VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

API ONLINE WEBOVÝCH APLIKACÍ

API FOR ONLINE WEB APPLICATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JAN ČAPEK
AUTHOR

VEDOUCÍ PRÁCE Ing. JAN SAMEK, Ph.D.
SUPERVISOR

BRNO 2016

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Čapek Jan**
Obor: Informační technologie
Téma: **API online webových aplikací**
API for Online Web Applications
Kategorie: Umělá inteligence

Pokyny:

1. Prostudujte a popište standardy, metody a návrhové vzory pro tvoru aplikačních programových rozhraní (API). Zaměřte se především na API pro online webové/internetové aplikace (sociální sítě, komunitní weby apod).
2. Vytvořte přehledovou studii možností a použitelnosti jednotlivých existujících API různých online aplikací, zaměřte se především na popis komunikačního protokolu a dále na typ a strukturu dat, které jednotlivé API poskytují.
3. Navrhněte a implementujte netriviální aplikaci, která bude demonstrovat použití některých z vámi popsaných API vybraných webových aplikací.
4. Zhodnoťte výsledky vámi vytvořené práce a diskutujte možná rozšíření.

Literatura:

- Brian Mulloy. *Web API Design: Crafting Interfaces that Developers Love*. 2014
- Socials API. *ProgrammableWeb* [online]. 2014. Dostupné z: http://bit.ly/1rubIgH
- MEMON, Nasrullah. *Data mining for social network data*. London: Springer, c2010, ix, 215 p. Annals of information systems, v. 12. ISBN 9781441962874

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2 zadání.

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

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

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

Vedoucí: **Samek Jan, Ing., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Tato bakalářská práce se zaměřuje na popis a návrh aplikačních rozhraní. V práci jsou popsány architektury aktuální doby, ale jsou okrajově popsány i architektury historické. Součástí je rovněž proces návrhu a implementace aplikace, která by některé rozhraní využívala.

Abstract

This bachelor thesis is focused on describing and designing application interfaces. The thesis describes architectures of current age and also marginally describes legacy architectures. The part of the work is also process of design and implementation of application, which would utilize some interface.

Klíčová slova

Web API, REST, API, Webová architektura, Architektura API

Keywords

Web API, REST, API, Web architecture, Architecture of API

Citace

ČAPEK, Jan. *Api Online Webových Aplikací.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Samek Jan.

Api Online Webových Aplikací

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jana Samka Ph.D. Další informace mi poskytla Yvonne Chavez, zaměstnankyně společnosti Hi-Rez Studios. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Jan Čapek
17. května 2016

Poděkování

Děkuji Ing. Janu Samkovi Ph.D za odborné vedení, za pomoc a rady při zpracování této práce. Mé poděkováni patři též Yvonne Chavez, která mi vyšla vstříc při získávání údajů pro výzkumnou část práce a všem testujícím, kteří obětovali svůj čas při testování aplikace.

© Jan Čapek, 2016.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1 Úvod **3**

2 Standardy používané pro API **4**
2.1 Architektura orientovaná na zdroje . . . 5
2.1.1 REST . . . 5
2.2 Achitektura orientovaná na služby . . . 7
2.2.1 XML-RPC . . . 7
2.2.2 SOAP . . . 9
2.3 Formáty používané pro reprezentaci dat . . . 11
2.3.1 JSON . . . 11
2.3.2 XML . . . 12
2.3.3 Prvky a syntaxe . . . 12
2.4 Praktiky . . . 14

3 Existující API **17**
3.1 API her a herních platforem . . . 17
3.1.1 Hi-Rez Studios Smite API . . . 17
3.1.2 Valve Corporation Steam API . . . 19
3.1.3 Riot Games API . . . 20
3.2 API sociálních sítí . . . 22
3.2.1 Facebook for developers . . . 22

4 Implementace **24**
4.1 Úvod . . . 24
4.2 Smite . . . 24
4.2.1 Základní informace . . . 24
4.2.2 Principy . . . 25
4.3 Návrh aplikace . . . 26
4.3.1 Použité technologie . . . 27
4.3.2 Databázový model . . . 28
4.4 Aplikace . . . 31
4.4.1 Algoritmus pro výpočet výhry . . . 31
4.4.2 Uživatelské rozhraní . . . 33
4.5 Testování . . . 37
4.5.1 Návrh . . . 37
4.5.2 Zjištěné informace . . . 38
4.5.3 Zpětná vazba uživatelů . . . 39

5 Závěr **40**

6 Literatura **41**

A Obsah CD **43**

Kapitola 1

Úvod

V dnešní době jsou webové technologie aktuálním trendem. Někteří uživatelé ovšem nechtějí pouze konzumovat obsah poskytovaný webovými stránkami či jinými aplikacemi. V takovém případě existují pro tvořivé uživatele webová aplikační rozhraní, které poskytují možnost přistupovat k datům jiným způsobem. Dovolují jim tvořit své vlastní aplikace založené na obsahu, který poskytuje původní tvůrce.

Tato bakalářská práce popisuje jednotlivé architektury použité pro aplikační rozhraní. Největší ohled je brán na architekturu REST, která je dnes využívána pro takřka všechna moderní aplikační rozhraní. Jsou popsány nejen formáty dat, ale ukázány i praktiky, kterých se ve vývoji moderních API využívá. Zastaralé technologie ale nejsou zanedbány a jsou v práci rovněž okrajově popsány. Jednou z částí této práce je i přehled několika aplikačních rozhraní. Je zde popsáno, co takové rozhraní může nabízet. V této sekci je i demonstrováno, že aplikační rozhraní nejsou využitelná ryze pro existující webové stránky, ale poskytují je i jiné aplikace. Tato práce se tudíž nesoustředí na typické aplikační rozhraní, které poskytují například sociální sítě. Prozkoumává méně probádané odvětí, API pro hry a herní platformy. V implementační kapitole se poté podrobněji nahlédne na jedno existující herní API. Konkrétně se jedná o počítačovou hru Smite od společnosti Hi-Rez Studios. Cílem aplikace je stahování dat dostupných v API a vypočítat z nich pravděpodobnost hráče na vítězství. Další funkcionalitou je zobrazení informací, které mu pomohou optimalizovat svou strategii. V této kapitole je popsán návrh této aplikace, použité algoritmy a uživatelské rozhraní. Testování není opomenuto a výsledky tohoto procesu se rovněž nachází v příslušné sekci práce. V závěru jsou poté velmi stručně shrnuty dosažené výsledky.

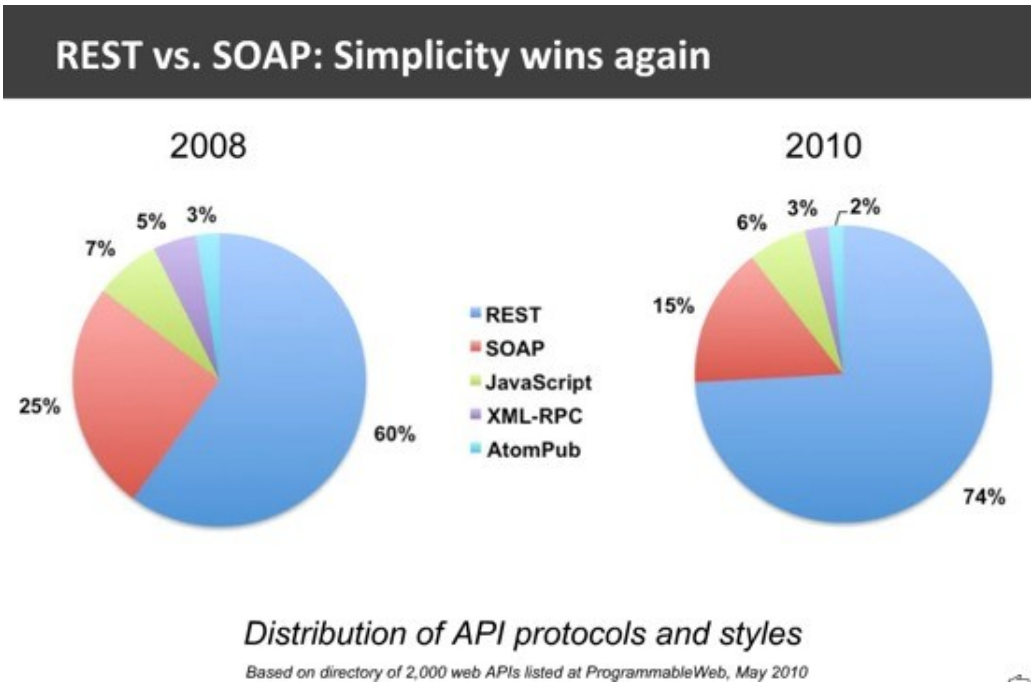
Kapitola 2

Standardy používané pro API

Tato kapitola se zabývá různými návrhy a přístupy pro realizování aplikačních rozhraní. Popisuje jaké standardy se pro serializaci dat používají a protokoly zapouzdřující přenášená data. Nejdůkladněji je rozebrána architektura orientovaná na zdroje, protože právě ta je dnes nejrozšířenější.

Aplikační rozhraní lze definovat jako množinu různých metod, typicky poskytované přes nějaký komunikační protokol, které umožňují uživateli přistupovat ke zdrojům či službám dostupným na vzdáleném serveru. Zatímco v minulosti byly takové API často zaměřené na služby, tak dnes se od této praktiky ustupuje a přes rozhraní je přistupováno přímo ke zdrojům.

Následující graf z roku 2010 ukazuje, že dříve používané technologie SOAP a XML-RPC ustupují jednoduššímu aplikačnímu rozhraní RESTful.



Obrázek 2.1: Růst podílu návrhu REST [3]

2.1 Architektura orientovaná na zdroje

Účelem této architektury je zpřístupnit různé zdroje uložené na serveru. Zdroji jsou u webových aplikací uvažovány různé spustitelné skripty anebo libovolně vnořené datové struktury a kolekce, které může klient přistupující přes rozhraní využívat ke svým dalším účelům. V dnešní době se pro realizaci API používá nejčastěji.

2.1.1 REST

REST (Representational State Transfer) je architektura rozhraní určená pro webové služby. REST se vyvinul současně s protokolem HTTP, který se tudíž stal téměř výhradně standardním přenosovým médiem pro RESTful systémy. Volbě protokolu také nahrál fakt, že REST definoval ve své disertační práci Roy Fielding, který je jedním ze spoluautorů protokolu HTTP. REST oproti API zaměřeným na služby určuje, jak se přistupuje k datům. Všechny zdroje mají vlastní identifikaci, kterou je v případě protokolu HTTP URI (uniform resource identifier).

Definice

Jednu z nejpřesnějších definicí uvedl Bill Burke ve své knize RESTful Java with JAX-RS [1]. Uvádí, že klíčové vlastnosti určující REST jsou následující:

- **Adresovatelnost zdrojů** - klíčovou abstrakcí dat a informací je zdroj a každý takový zdroj musí být adresovatelný pomocí URI.
- **Jednotnost a vynucenost rozhraní** - k manipulaci zdrojů se používá malá množina dobře navržených metod.
- **Orientovanost na reprezentaci** - zdroje identifikované jednou URI mohou mít různé formáty. Různé platformy mohou vyžadovat různé formáty. Prohlížeč může požadovat HTML, JavaScriptové aplikace mohou požadovat JSON a jiné aplikace formát XML.
- **Beze stavová komunikace** - škálování bez stavové aplikace je jednodušší.
- **Hypermedia As The Engine Of Application State (HATEOAS)** - přechody mezi stavy aplikace jsou řízeny formáty dat.

Erik Wilde [18] tuto definici celkově minimalizuje na několik již z výše zmíněných principů. Tvrdí, že REST se zaměřuje především na to, aby bylo možné přistupovat ke zdrojům skrze URI, a aby byla komunikace bez stavová. To znamená, že všechny potřebné stavové informace jsou obsaženy v interakci mezi serverem a klientem a server si nemusí udržovat relaci.

Principy

Shrnutí informací z definic uvedených výše zmíněnými autory a další principy.

Bezestavovost
Bezestavovost určuje, že server nepotřebuje udržovat s klientem žádnou relaci. Všechny informace, které by mohl server potřebovat, jsou zaslány v požadavku. Výhodou je menší paměťová i výpočetní náročnost a lze tedy celkový systém snáze škálovat.

Mezipaměť
Mezipaměť je uvedena na stranu klienta, aby se snížilo vytížení serveru. Při využití tohoto principu je v odpovědi od serveru uvedeno, zda je možné získaná data uložit. Takto lze omezit množství dat přenášené po síti a zároveň snížit celkové nároky na systém díky eliminaci interakce mezi klientem a serverem. Nevýhodou je však existující možnost, že data v mezipaměti se budou lišit od dat, které by klient získal při zaslání dotazu.

Adresovatelnost
Adresovatelnost je princip, který praví, že všechny zdroje by měly dostupné skrze unikátní adresu. V případě webových aplikací se používají Unique Resource Identifiers (URI). V reálných případech se snaží API udržovat v určité hierarchii. Takto může uživatel intuitivně dedukovat cestu adresu jiným zdrojům.

Jednotnost rozhraní
Jednotné rozhraní poskytuje pouze určitou množinu operací, aby však měl klient dostatečné možnosti, tak je v případě REST využíváno operací CRUD – Create, Read, Update a Delete. Takovou jednotnost splňuje protokol HTTP, a proto je používán pro REST nejčastěji. Celá interakce mezi klientem a serverem je popsána základními metodami HTTP.

Níže jsou popsány metody, které jsou pro rozhraní REST používána. Zatímco metody jako GET a DELETE se používají stejným způsobem pro téměř všechna aplikační rozhraní, tak například POST a PUT se mohou svým použitím lišit.

- **GET** popisuje požadavek o načtení dat nebo informace ze zdroje. Na tento požadavek je v hlavičkách v odpovědi přiložena informace popisující data obsažené v těle zprávy.
- **PUT** popisuje požadavek o upravení existujícího na zadané adrese URI. Server si obsaženou reprezentaci objektu přečte a vytvoří a aktualizuje již existující položku. V některých případech lze využívat pro vkládání velkého objemu dat na server.
- **POST** může být použit pro vytvoření nového zdroje a v závislosti na dané implementaci API, také k aktualizaci již existujících zdrojů, avšak Leonard Richardson [4] uvádí, že rozdíl mezi PUT a POST je v tom, že PUT rozhoduje o tom, na jaké URI se nový zdroj bude nacházet, zatímco POST se používá, když o výsledné URI rozhoduje server.
- **DELETE** je odeslán v případě, že je potřeba zdroj smazat. Klient by neměl odesílat žádné data reprezentující cílový objekt.

Reprezentace
Reprezentace popisuje formáty dat, které jsou použity při přenosu mezi serverem a klientem. Je zde však velmi velký rozdíl mezi zdrojem a jeho reprezentací. Zdroj jako takový lze reprezentovat jakýmkoliv vhodným formátem. Nejčastěji používané reprezentace jsou JSON, XML a HTML.

HATEOAS
Odpověď serveru vždy obsahuje data a odkazy v závislosti na stavu aplikace. Výstupem tedy nejsou pouze data, ale i odkazy na další zdroje, které mohou vést k dalším zdrojům.

Struktura v HTTP

Následující příklad demonstruje použití REST API v přenosovém protokolu HTTP. Požadavek zobrazuje metodu GET, která značí prosté získání dat ze zdroje. Dále následuje URL adresující zdroj. Doplňujícím parametrem je API klíč. Poté následuje verze protokolu a hlavičky.

```
GET /api/lol/euw/v1.4/summoner/by-name/Thiesius?api_key=f5c2d857
    -0000-0000-0000-2761fe73281d HTTP/1.1
Host: euw.api.pvp.net
User-Agent: Mozilla
Accept-Language: en-US
Accept-Charset: ISO-8859-1,utf-8
Connection: close
```

Odpovědí na požadavek výše je zpráva OK, která značí, že vše proběhlo v pořádku. Dále následují hlavičky odpovědi a po nich samotná data, která byla získána z uzlu.

```
HTTP/1.1 200 OK
Content-Length: 139
Access-Control-Allow-Headers: Content-Type
Access-Control-Allow-Methods: GET, POST, DELETE, PUT
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=UTF-8
Date: Fri, 15 Jan 2016 22:02:05 GMT
Server: Jetty(9.1.5.v20140505)
Vary: Accept-Encoding, User-Agent
Connection: keep-alive Connection: close

{"thiesius": {
   "id": 79477711,
   "name": "Thiesius",
   "profileIconId": 26,
   "revisionDate": 1445080124000,
   "summonerLevel": 1
}}
```

2.2 Achitektura orientovaná na služby

Tento návrhový vzor popisuje komponenty aplikací, které poskytují služby ostatním nezávislým komponentám skrz nějaký komunikační protokol. V případě webových aplikačních rozhraní tato architektura není tolik využívána.

2.2.1 XML-RPC

XML-RPC je protokol pro vzdálené volání procedur (RPC – Remote procedure call), který používá XML pro zakódování svých volání a zpravidla protokol HTTP jako transportní mechanismus. XML-RPC lze uvažovat jako sekvenci požadavků přenášené ve formátu XML, které vyvolají na vzdáleném stroji výpočty, jejichž výsledky jsou opět ve formátu XML vráceny ve formě odpovědi.

Způsob vyhodnocování

- Klient specifikuje jméno metody a parametry zabalí je do XML a pomocí HTTP POST zprávy je zašle na cílový server.
- HTTP server obdrží POST požadavek a předá jej ke zpracování XML-RPC.
- XML-RPC získá z XML jméno metody a parametry a zavolá příslušnou metodu předávajíc extrahované parametry.
- Výsledek je opět obalen XML a je vrácen serveru, který odešle odpověď na dříve zaslaný POST požadavek.
- Klient zpracuje XML a získá návratovou hodnotu a program může pokračovat v práci.

Datové typy

Volání vzdálených metod je velmi podobné jako v standardním programování. Při volání metody jsou přiložena data ve formě parametrů a v odpovědi jsou obsažena data jako návratová hodnota. V odpovědi může být právě jedna hodnota.

Základní typy

Celá čísla

Reprezentují 32-bitové čísla se znaménkem v rozmezí od -2147483648 a 2147483647. V XML-RPC jsou zapisována jako

```
<value><int>cislo </int></value>
```

Čísla s plovoucí desetinnou čárkou

Reprezentují čísla s dvojitou přesností podle standardu IEEE 754 o velikosti 64-bitů. Jsou zapisována jako

```
<value><double>cislo </double></value>
```

Boolean

Reprezentují logické hodnoty. Nabývá pouze hodnot 0 a 1, kde 1 je uvažována jako true (pravda). Logické hodnoty jsou zapisovány následovně

```
<value><boolean>hodnota</boolean ></value>
```

Řetězce

Reprezentují textové řetězce. Znaky, které jsou vyhrazené jazykem XML je nutné zapisovat pomocí předdefinovaných entit – jako například & pro &. Lze je zapisovat dvěma způsoby.

```
<value><string>Nejaky retezec!</string ></value>
```

Nebo využít toho, že řetězec je implicitní typ

```
<value>Nejaky retezec!</value>
```

Datum

Datum a čas jsou v XML-RPC zakódovány jako element typu dateTime.iso8601

```
<value><dateTime.iso8601>CCYYMMDDTHH:MM:SS</dateTime.iso8601></value>
```

Jako příklad lze použít hodnotu 19930608T12:46:25, která představuje čas 8. 6. 1993 12:46:25

Binární

Netisknutelné znaky je nutné zakódovat kódováním base64. Binární data zapisujeme následovně

```
<value><base64>QmFrYWzDocWZa2EgQVBJ</base64></value>
```

Složené typy

Pole

XML-RPC umožňuje seskupovat data jako sekvenci hodnot. Od konvenčního programování se však liší tím, že nemusí být stejného typu a nejsou indexovány obvyklým způsobem.

```
<value>
  <array>
    <data>
      <value><int>1</int></value>
      <value><int>12</int></value>
      <value><int>7</int></value>
    </data>
  </array>
</value>
```

Struktury

XML-RPC rovněž umožňuje seskupovat data do pojmenovaných položek. Od konvenčního programování se však liší tím, že nemusí být stejného typu a nejsou indexovány obvyklým způsobem.

```
<struct>
  <member>
    <name>Prvek1<name>
    <value><int>1</int></value>
  </member>
  <member>
    <name>Prvek2<name>
    <value><string>Testovaci retezec</string></value>
  </member>
</struct>
```

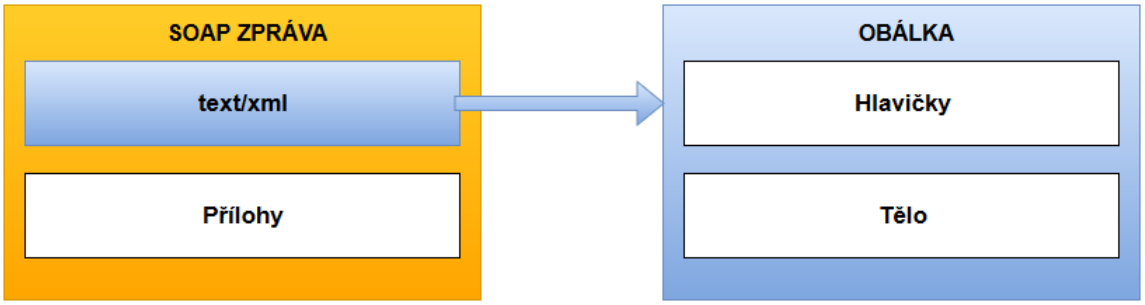
2.2.2 SOAP

SOAP (Simple Object Access Protocol) je protokol pro výměnu strukturované informace. Podobně jako XML-RPC používá datový formát XML a k přenosu používá nejčastěji protokol HTTP, ačkoliv není úplně závislý na protokolu.

Ačkoliv je SOAP uvažován jako následník XML-RPC, tak i v tomto případě není tak často využívaný jako API založené na REST.

Formát

Zpráva SOAP je XML dokument skládající se ze tří částí – obálky, hlavičky a těla.



Obrázek 2.2: Formát SOAP zprávy

Obálka

Obálka je nejvyšší element XML dokumentu obalující zprávu a musí být ze jmenného prostoru SOAP protokolu. Může obsahovat nepovinný element definující hlavičku a musí obsahovat tělo.

Hlavička

Nepovinný element, ve kterém lze specifikovat další možnosti pro SOAP, jako například bezpečnost a směrování.

Tělo

Tělo je povinnou částí SOAP zprávy a obsahuje data. Formát dat není určený, a proto nemusí být nutně ve formátu XML.

Následující ukázka představuje volání metody pomocí protokolu HTTP.

```
POST /InStock HTTP/1.1
Host: www.stock.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 299

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http:www.w3.org/2001/12/soap-envelope"
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
    <soap:Body xmlns:m="http://www.stock.org/stock">
        <m:GetStockPrice>
            <m:StockName>IBM</m:StockName>
        </m:GetStockPrice>
    </soap:Body>
</soap:Envelope>
```

2.3 Formáty používané pro reprezentaci dat

Formát dat přenášený z aplikačního rozhraní většinou není pevně stanovaný použitou architekturou a návrhem. Nejčastěji používané formáty jsou u webových API JSON (JavaScript Object Notation) a XML (Extensible Markup Language). Protože různé aplikace mohou vyžadovat různé formáty, tak by webové aplikační rozhraní mělo poskytovat možnost volby datového formátu.

2.3.1 JSON

JavaScript Object Notation je otevřeným standardem, který se zaměřuje na snadnou čitelnost a snížení velikosti dat odeslaných při komunikaci. Struktura dokumentu ve formátu JSON se skládá z párů jméno-hodnota. Je nezávislý na jazyce a využívá notace sdílené v jazycích C, C++, Java atd.

Funkcionalita

JSON byl především navržen tak, aby byl jednoduchý a univerzální. Jeho další vyzdvihovanou vlastností je, že je pro člověka snadno čitelný. Jazyk JSON rovněž vlastní velmi jednoduchou gramatiku, proto je parsování dokumentů výkonově nenáročné.

Standard ovšem nepodporuje reference na objekty, čímž se efektivně vylučuje použití cyklických struktur. Funkcionalita referencí lze rozšířit pomocí Frameworku Dojo [7].

V JSONu lze vytvářet i komplexní struktury, jako asociativní pole a libovolně vnořené objekty obsažené v objektech. Každý objekt pak také může obsahovat libovolný platný datový typ.

Na rozdíl od XML však není v jazyce doplněna podpora pro validaci hodnot a struktury. Lze však doplnit externí specifikaci nazvanou JSON Schema.

Datové typy a syntaxe

Platnými datovými typy jsou následující

- čísla (nekonečno nelze vyjádřit),
- desetinná čísla (zapsané ve vědecké notaci),
- řetězce (včetně podpory Unicode znaků),
- logické hodnoty Boolean,
- pole,
- objekty (asociativní pole neboli páry název-hodnota),
- prázdný datový typ (null).

V následujícím bloku je příklad dat formátovaných v JSONu.

```
{
  "poleSJednimPrvkem":[
    "cislo": 3,
    "dalsiCislo": 1784,
    "vnorenePole": [
      {
        "retezec": "BP API 1",
        "desetinneCislo": 14.5
      },
      {
        "dalsiRetezec": "BP API 2",
        "desetinneCislo2": 20.8e-4
      }
    ]
  ]
  "prazdny": null
}
```

Syntaxe jazyka JSON je velmi jednoduchá. Kořenový uzel dokumentu JSON musí být objekt nebo pole. Objekty jsou vyznačeny složenými závorkami . Identifikátory jsou obaleny uvozovkami a následovány středníkem a hodnotou. Prvky sestavující objekty jsou odděleny čárkou. Pole je vyznačeno hranatými závorkami [] a hodnoty v něm obsaženém jsou odděleny čárkou. Komentáře nejsou v současné revizi jazyka JSON dovoleny.

2.3.2 XML

Extensible Markup Language je značkovací jazyk, který definuje pravidla pro zakódování dokumentu tak, aby jej přečetl stroj i člověk. XML se především soustředí na obecnost použití. Je méně kompaktní než výše zmíněný JSON, ale zase poskytuje prostředky pro validaci dokumentu.

Funkcionalita

Jazyk XML nebyl navržen, aby sloužil pouze jako serializační formát a proto je oproti jazyku JSON méně kompaktní a zavádí vyšší komplexnost. Je možné v něm vyjádřit libovolné datové struktury. Dokument se skládá z prvků, jejich hodnot a případně i jejich atributů. Pro ověřování dokumentů existují reference na DTD (Document Type Definition).

2.3.3 Prvky a syntaxe

Jazyk XML oproti svému protějšku JSON neobsahuje přímo datové typy. V následující sekci jsou vypsány pouze některé nejdůležitější části XML definované podle specifikace [16]

- **Deklarace**
  XML dokument může o sobě na počátku poskytnout informace jako například verze XML a kódování.

- **Značka**
  Začíná znakem < a je ukončena znakem >. Rozlišujeme otevírací značku <znacka> zavírací značku </znacka> a značku prázdného elementu <znacka />

- **Elementy**
  Je složen z otevírací a ukončovací značky anebo ze značky definující prázdný element. Znaky mezi značkami tvoří obsah elementu a mohou obsahovat i dále vnořené podelementy, kterým se říká děti.

- **Atribut**
  Dodatečná hodnota, kterou lze zapsat k otevírací značce anebo k značce prázdného elementu. <znacka nejakyAttribut="BP API"dalsiAttribut="BP API" /> <znacka nejakyAttribut="15"dalsiAttribut="10»OBSAH</znacka>

- **Komentáře**
  Na rozdíl od jazyka JSON lze v jazyce XML zapisovat komentáře. Komentáře sice nelze vnořovat, ale zase do nich lze zapisovat jakékoliv symboly. Komentáře začínají řetězcem „<!–“ a jsou ukončeny řetězcem „–>“.

Následující úryvek dokumentu demonstruje prvky XML.

```
<?xmlversion="1.0" encoding="UTF-8"?>
<Array0fplayerQueueStats
    xmlns="http://schemas.datacontract.org/2004/07/SmiteApi" xm1ns:i="
        http://mAcw3.org/2001/XMLSchema-instance">
    <playerQueueStats>
        <Assists>0</Assists>
        <Deaths>4</Deaths>
        <God>He_Bo</God>I

        <ret_msg i:nil="true" />
    </playerQueueStats>
    <playerQueueStats>
        <Assists>0</Assists>
        <Deaths>2</Deaths>
        <God>Ah Muzen_Cab</God>
        <ret_msg i:nil="true" />
    </playerQueueStats>
    <playerQueueStats>
        <Assists>0</Assists>
        <Deaths>4</Deaths>
        <God>Loki</God>
        <ret_msg i:nil="true" />
    </playerQueueStats>
</ArrayofplayerQueueStats>
```

Validace

Dokument může být validován, k tomu slouží odkaz na takzvaný DTD (Document Type Definition). Všechny elementy a atributy poté musí splňovat gramatická pravidla, která jsou specifikovaná v odkázaném dokumentu.

Použitím DTD lze specifikovat jaké druhy elementů se mohou nebo musí v dokumentu vyskytovat, v jakém pořadí se mohou objevit a jaké atributy mohou mít.

Následující příklad demonstruje v horní části definované DTD popisující strukturu dokumentu. Po ní následuje samotný XML dokument, který splňuje Document Type Definition.

```
<!ELEMENT god (name, nickname*, skills)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT nickname (#PCDATA)>
<!ELEMENT skills (skill, skill, skill, skill)>
<!ELEMENT skill (skillname, skilldescription, skillmanacost)>
<!ELEMENT skillname (#PCDATA)>
<!ELEMENT skilldescription (#PCDATA)>
<!ELEMENT skillmanacost (#PCDATA)>


<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE clovek SYSTEM "example_smite.dtd">
<god>
    <name>Agni</name>
    <skills>
        <skill>
            <skillname>Noxious Fumes</skillname>
            <skilldescription>
              DESC1
            </skilldescription>
            <skillmanacost>60/70/80/90/100 mana</skillmanacost>
        </skill>
        <skill>
            <skillname>Flame Wave</skillname>
            <skilldescription>
              DESC2
            </skilldescription>
            <skillmanacost>60/70/80/90/100 mana</skillmanacost>
        </skill>
    </skills>
</god>
```

2.4 Praktiky

U realizace API nezávisí jen na použité architektuře. Závisí i na určitých dalších faktorech, které se mezi použitými technologiemi prolínají. Ačkoliv tyto praktiky jsou považovány za užitečné, tak nejsou závazné a neobjevují se ani u aplikačních rozhraní implementovaných velkými společnostmi. Některé tyto praktiky popsal ve svém dokumentu Brian Mulloy [8] a budou shrnuty níže.

Intuitivnost rozhraní
Pokud jsou objekty, které API zpřístupňuje, odhadnutelné i bez dlouhé studie dokumentace, lze o rozhraní říci, že je intuitivní. Jedno objekt by měly adresovat maximálně dvě URL. Jedna zpřístupňující celou skupinu a druhá zpřístupňující jeden konkrétní prvek skupiny.

```
/lide                    /lide/1486
```

Podstatná jména oproti slovesům
V případě užívání sloves se do API zavádí nepotřebná komplexita. Využívání zvyků z kon-

venčních programovacích jazyků a nazývat zdroje jako metody začínající slovem get je tedy nevhodné.

```
/getAllPlayers    /getPlayer        /getOnlinePlayers
```

oproti

```
/players          /players/Peterson        /players?state=online
```

Jednotné číslo oproti množnému číslu
Zde nelze s určitostí doporučit, zda je vhodnější používat jednotných čísel anebo použít číslo množné. Je však vhodné zvolit jeden přístup a držet se ho – tím se stane aplikační rozhraní více intuitivní, protože uživatel může lépe odhadnout adresu jiného objektu.

Menší míra abstrakce
Pro objekty je lepší volit konkrétní názvy než abstraktní formy. Z názvu /objects nebo /items není úplně možné pro uživatele určit, jaká reprezentace mu bude v odpovědi vrácena.

Asociace
Mezi jednotlivými objekty mohou být relace. Je dobrým zvykem ji vyjádřit pomocí adresy například následujícím způsobem

```
/objekt1/identifikator/objekt2
```

Konkrétněji lze demonstrovat na příkladu. Uvažujme objekt reprezentující množinu dosažených úspěchů

```
/achievments
```

a konkrétního hráče

```
/player/2058
```

Lze mezi nimi vytvořit relaci

```
/player/2058/achievments
```

Atributy
Určité objekty lze izolovat použitím atributů, které obsahuje. V adrese se k tomu účelu zpravidla využívá znak „?“ a jednotlivé atributy jsou odděleny symbolem „&”.

```
/players?level=28&status=online&activeGame=true
```

Chybová hlášení
Rovněž je potřeba zvolit vhodný způsob, kterým jsou uživateli ohlašovány chyby. Dobře navržená chybová hlášení umožní uživateli pohodlně používat aplikační rozhraní.

Jelikož je nejpoužívanějším protokolem přenosu HTTP, tak lze využít návratových kódů obsažených ve specifikaci. K popsání výsledku operace stačí tři různé kódy

- Operace proběhla úspěšně.
- Aplikace provedla něco špatně – chyba na straně klienta.
- Server provedl něco špatně – chyba na straně serveru.

K tomu lze využít kódy 200 Success, 400 Bad Request a 500 Internal Server Error. Pokud vývojář potřebuje více chybových oznámení, tak může vybrat další podle specifikace HTTP.

Verze API

Před vydáním API je vhodné zvolit systém verzování. Lze zvolit z několika přístupů. Nejvhodnějším je přidání verze přímo do adresy.

```
/v1/players/Alkaline
```

Vývojář by se ale však měl vyhnout přidávání tečky jako v1.2. Volitelně pak lze vložit verzi do hlaviček HTTP požadavku.

Kapitola 3

Existující API

3.1 API her a herních platforem

V současné době se dostávají aplikační rozhraní i do sféry digitální zábavy jako jsou počítačové hry a jejich distribuce. Zájem o počítačové hry roste, a proto se vývojáři těchto her a platforem snaží poskytnout vývojářům třetí strany nástroje pro tvorbu vlastního obsahu. Takovým nástrojem může být i webové rozhraní, ze kterého mohou uživatelé API čerpat data z online her anebo informace z prodejních platforem. Použitím těchto rozhraní nemusí vývojáři třetích stran získávat data pochybným způsobem, jako je například odposlouchávání a dešifrování komunikačního protokolu anebo vytahování omezených informací z oficiálních webových stránek aplikace.

3.1.1 Hi-Rez Studios Smite API

Co je to Smite API

Smite [6] je počítačová hra pro více hráčů vyvíjená společností Hi-Rez Studios. Řadí se do žánru MOBA (Multiplayer Online Battle Arena), kde spolu dvě družstva hráčů soupeří o vítězství. Základna hráčů je v současné době natolik vysoká, že se vývojáři rozhodli implementovat aplikační rozhraní zpřístupňující různé statistické i informační údaje ze hry.

Popis aplikačního rozhraní

Aplikační rozhraní je vystavěné na architektuře REST a používá k přenosu protokol HTTP. Rozhraní se striktně nedrží zásadami specifikované v kapitole praktiky, která je zmíněna v této práci. Používá totiž spíše programátorský přístup a v názvech svých adres používá sloveso get.

Základními URL jsou:

PC: `http://api.smitegame.com/smiteapi.svc`
Xbox: `http://api.xbox.smitegame.com/smiteapi.svc`

Rozhraní nemůže libovolně používat každý. Nejprve je nutné získat informace pro vývojáře. Tyto informace lze získat vyplněním registračního formuláře. Registrační formulář https://fs12.formsite.com/HiRez/form48/secure_index.html (23.12.2015) Přístupové informace se skládají z identifikátoru vývojáře a klíče. Následující informace slouží jako příklad a nejsou platné.

DevId: 1521
AuthKey: 2694AF04D3DD0C11B6FC1B4D489E020E

Než lze aplikační rozhraní využít je nutné vytvořit relaci. K tomu slouží metoda createsession. Relace slouží k autentizaci, bezpečnosti a statistickým účelům.

Na rozhraní jsou rovněž nastaveny následující limity:

```
Počet současných relací: 50
Relací za den:  500
Čas jedné relace:   15 minut
Počet požadavků za den:   7500
```

Aplikační rozhraní se od jiných rozhraní liší tím, že nepoužívá vrstvu SSL/TLS k šifrování přenášeného obsahu. Aby nebyly odhaleny případnému útočníkovi přístupové informace, tak se s každým požadavkem vypočítává podpis. Pro vytvoření podpisu se používá hashovací algoritmus MD5. Podpis je sestaven z identifikátoru vývojáře, hledaného zdroje, klíče od API a časového razítka ve formátu UTC následujícím způsobem:

```
md5_hash("1521" + "createsession" + "2694
    AF04D3DD0C11B6FC1B4D489E020E" + "20120927183145");
```

Aplikační rozhraní podporuje získání dat v reprezentaci JSON a XML. Formát dat si uživatel volí přidáním názvu žádaného formátu za název metody. Získání dat pak může vypadat následovně:

```
// Obecne
metoda[format]/{devId}/{podpis}/{relace}/{casoveRazitko}
// Konkretne
getdatausedJson/1521/82F89CADA10742048E9D8C015F542C36/
    BCA3F095F56A5904E18226ED76ECC497/20120927183145
```

Časové razítko a podpis zavádí však do API nepotřebnou komplexitu, která narušuje intuitivnost rozhraní. Lepším řešením by pravděpodobně byla změna protokolu z HTTP na HTTPS a tím by odpadla nutnost řešit skrytí klíče od API tím, že je vložen do MD5 podpisu.

Funkce

API poskytuje přibližně následující funkcionalitu:

- Informace o tom, kolik dat uživatel API již vyčerpal.
- Informace o e-sportech, jako kdy utkání probíhalo, jaká družstva se utkala, jak se nazýval turnaj a zda už je utkání odehráno.
- V případě, že má hráč povolenou sníženou diskrétnost profilu, lze získat seznam jeho virtuálních přátel.
- Lze získat čistě statistiky pro jednotlivé hráče – identifikátor hrdiny, jaké úrovně hráč s tímto hrdinou dosáhl a další.
- Poskytuje čistě informativní data o hrdinech obsažených ve hře - schopnosti včetně popisek, typ schopnosti, druh poškození a další parametry a popisky. Dále seznam předmětů, jejich parametry a různé další informace.

- Tipy pro hráče, jako jaké předměty by měl hráč za dané hrdiny používat.
- Získání informací o odehraném utkání – jména hráčů, jaké předměty za své hrdiny použili, výsledek utkání a další informace.
- Lze zjistit, zda daný hráč právě hraje hru a informace o této hře.
- Je možné vypsat seznam utkání za časové období a jejich stav.
- Získání informací o lize a sezónách jako například žebříček hráčů v lize a jejich statistiky.
- Vypsání hráčových předešlých her a statistik v těchto hrách. Dále dosažené úspěchy hráče, kdy se naposled přihlásil do hry a zda momentálně hraje.
- Hledání klanů a zjištění informací o klanu jako kdo je zakladatelem, počet členů, seznam hráčů a další informace

Rozhraní Smite API slouží především k získávání informací a lze přes něj vytvořit statistické weby shromažďující informace o hráčích, zobrazující výkonnostní grafy, porovnání mezi hráči a ukazující aktuální dění ve hře.

Příkladem webové aplikace využívající Smite API je například SmiteGuru[1]

3.1.2 Valve Corporation Steam API

Co je to Steam API

Steam [13] je platforma vyvíjená společností Valve Corporation, která je primárně určená k digitální distribuci her a softwaru. Poskytuje rovněž zajištění serverové infrastruktury pro hry pro více hráčů a také se snaží poskytnout komunikační zázemí pro hráče. Uživatel se zde může zapojovat to různých činností. Aplikační rozhraní se tedy snaží vytvořit přístup k různým informacím a prvkům, které služba Steam poskytuje.

Popis aplikačního rozhraní

Data poskytované službou Steam jsou zpřístupněný přes webovou službu založené na RESTful architektuře. K přístupu k většině dostupných koncových bodů není potřeba speciální oprávnění, ale některé z nich vyžadují API klíč. Takový klíč se poté specifikuje jako parametr při interakci s rozhraním. K přenosu se nabízí protokol HTTP i zabezpečená verze HTTPS.

Základní URL adresa rozhraní je `http://api.steampowered.com/`

Obecný přístup k prvkům nabízených rozhraním je následovný:

```
http://api.steampowered.com/<nazev_rozhrani>/<metoda>/v<verze>/?key=<
    klic>&format=<format>
```

Data jsou reprezentovatelné třemi různými formáty a to konkrétně JSON, XML a VDF. Formát VDF (Valve Document Format) je speciálním formátem navrženým společností Valve, jehož dokumentaci lze nalézt na webech společnosti Valve [14]

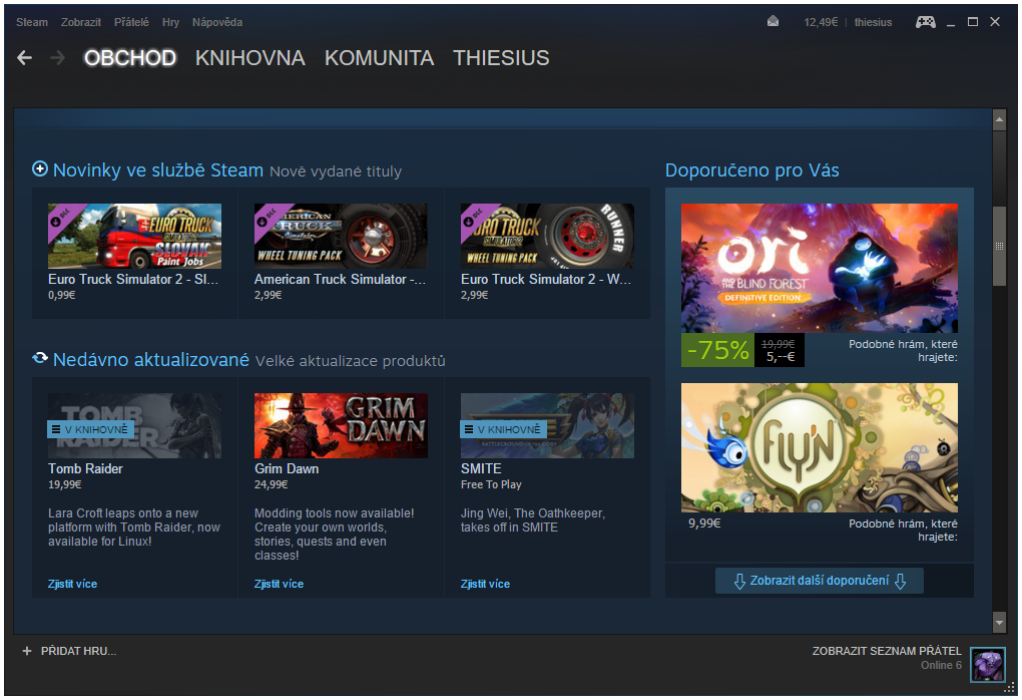
[1] Dostupné na `http://smite.guru/`

Funkce

Steam API poskytuje velké množství různých dalších koncových bodů.

- Seznam her, které uživatel platformy Steam vlastní.
- Úspěchy hráče, kterých dosáhl v konkrétní hře.
- Vypsání novinkek připnuté na kartě hry.
- Lze získat profil uživatele obsahující jeho identifikační údaje, obrázek a momentální stav.
- V určitých případech lze získat i privátní data uživatele jako je jeho reálné jméno.
- Lze získat seznam přátel uživatele.

Steam API nezpřístupňuje pouze platformu Steam jako takovou, ale rovněž přes něj lze spravovat i statistiky v hrách od Valve jako je Team Fortress 2 a nebo DotA 2.



Obrázek 3.1: Oficiální aplikace Steam

3.1.3 Riot Games API

Co je to Riot Games API

Riot Games API zapouzdřuje informace týkající se hry League of Legends [11]. Ta se podobně jako Smite řadí do žánru MOBA her, které jsou dnes velmi populární. League of Legends je momentálně považovaná za jednu z nejúspěšnějších her vůbec a proto studio poskytlo jiným vývojářům možnost získávat přes aplikační rozhraní různé informativní i statistické údaje.

Popis aplikačního rozhraní

Aplikační rozhraní je vystavěné na architektuře REST a používá k přenosu protokol HTTPS. Aplikační rozhraní je stejně jako hráčská komunita rozdělené na regiony. Základní URL adresa API je teda určena jako:

```
https://{region}.api.pvp.net/api/lol/{region}
```

Podporované regiony jsou v současné době:

Kód	Region
Br	Brazílie
Eune	Severovýchodní Evropa
Euw	Západní Evropa
Kr	Korea
Lan	Severní Latinská Amerika
Las	Jižní Latinská Amerika
Na	Severní Amerika
Oce	Oceánie
Ru	Rusko
Tr	Turecko

Tabulka 3.1: Regiony Riot Games API

Pro použití rozhraní je nutné získat klíč od API. Pro získání klíče stačí registrovat účet u Riot Games a zažádat o něj. Není nutné zakládat relaci. Při každém požadavku je pouze nutné přikládat klíč. Ten se poté vkládá přímo do URL požadavku.

```
/api/lol/euw/v1.2/champion?api_key=f5c2d857-0000-0000-0000-2761fe73281d
```

Na rozhraní jsou nastaveny následující základní limity:

```
10 požadavků každých 10 sekund
500 požadavků každých 10 minut
```

Aplikační rozhraní podporuje získání dat pouze v reprezentaci JSON. Poskytované zdroje jsou velmi dobře dokumentované a je možné jednotlivé funkce je možné po přihlášení ozkoušet přímo přes webový prohlížeč, intuitivnost rozhraní je tedy poměrně vysoká. Osvědčená praktika, které se API nedrží je verzování, jednotlivé komponenty rozhraní jsou verzovány zvlášť a není použit systém celých čísel.

Funkce

API poskytuje přibližně následující funkcionalitu

- Seznam hrdinů obsažených ve hře, a zda je momentálně možné za ně hrát i v ligových hrách.
- Je možné získat informace o právě probíhající hře, jako například účastnící, režim hry a zakázaní hrdinové.
- Lze získat seznam doporučených hrdinů pro nováčky.

- Jsou dostupné statistické informace o hráči – nedávno odehrané hry, kdy byla hra odehrána, za jakého hrdinu hráč hrál a jak si ve hře vedl.
- Lze vypsat jména hráčů účastnících se ligy a další informace o ligových zápasech.
- Stahování čistě informativních dat o hrdinech. Data zahrnují schopnosti včetně popisků, typ schopnosti, druhy poškození a další parametry definující daného hrdinu.
- Lze stáhnout seznam předmětů obsažených ve hře a jejich parametry a popisky.
- Jsou dostupná data o mapách a dalších statických prvcích hry.
- Aktuální informace o spuštěných serverech jsou také dostupná.
- Statistická data o odehraných hrách a jejich detaily – lze například získat informace o tom, jaké předměty hráči za svého hrdinu zakoupili apod.

Příkladem webové aplikace využívající Riot Games API je například http://www.lolking.net/

3.2 API sociálních sítí

Aplikační rozhraní jsou rovněž velmi rozšířená ve sféře sociálních sítí. Dnes se zapojuje do sociálních sítí téměř každý, kdo má dostupné internetové připojení. Aplikační rozhraní dostupná pro tyto weby či aplikace slouží k jednoduché tvorbě aplikací třetích stran, které například stahují informace o uživatelích a dávají je do různých vztahů.

3.2.1 Facebook for developers

Co je to Facebook for developers

Facebook for developers [4] je jak již název napovídá sada prostředků pro práci se sociální sítí Facebook, která je dnes bezesporu největší. Díky své popularitě dochází k tvorbě velkého množství aplikací, které poskytované API využívají.

Popis aplikačního rozhraní

Aplikační rozhraní dovoluje využívat relací mezi uživateli a získávání informací o profilech. Rovněž dovoluje přistoupit k přihlášenému profilu a publikovat různé aktivity na jejich profilu. Uživatel často vytvořené aplikace musí často dovolit případné změny v souladu s jejich nastavením zabezpečení. Typicky lze utilizovat informace jako je obecný profil, přátelé, skupiny, fotografie, oblíbené položky a nastávající události.

Architektura REST se stala základem pro toto rozhraní a k přenosu je použit bezpečný protokol HTTPS.

Dokumentaci aplikačního rozhraní lze nalézt

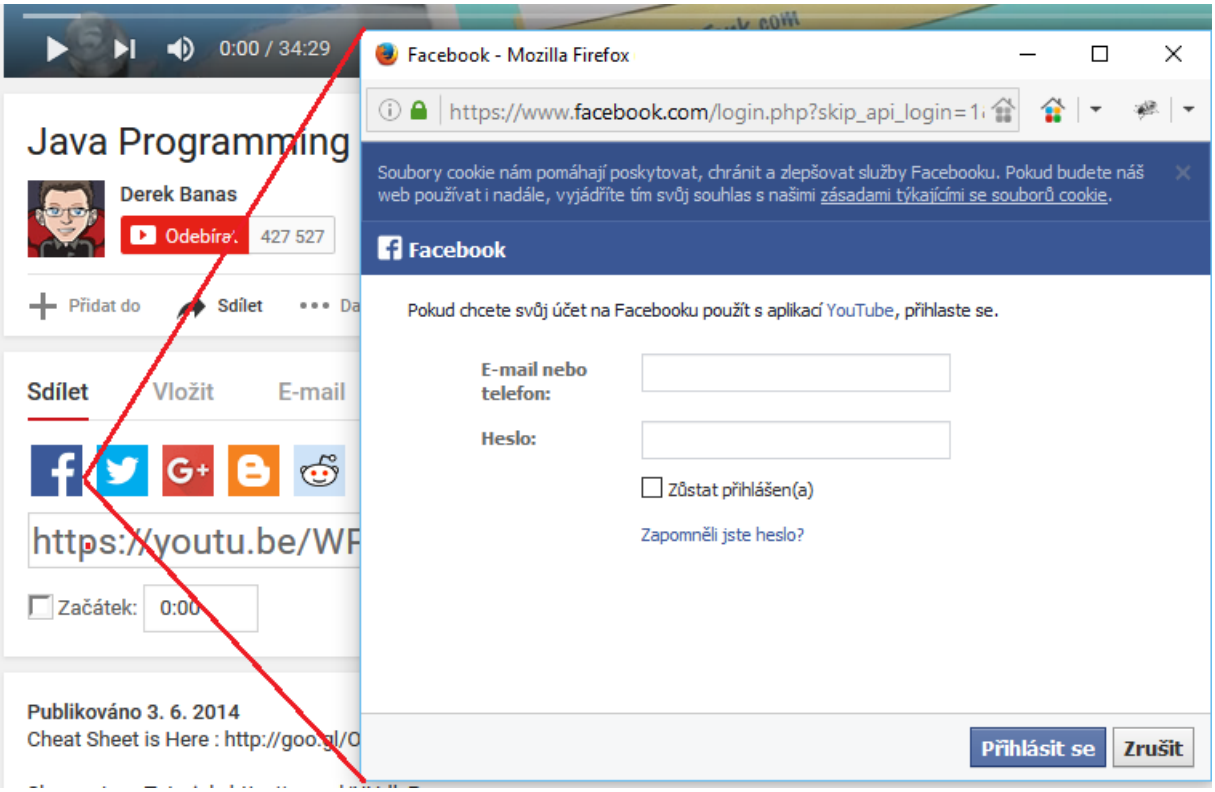
```
https://developers.facebook.com/docs/apis-and-sdks
```

K přístupu k aplikačnímu rozhraní a k většině informací musí aplikace nejprve získat takzvaný access token [5].

Postup k získání takového přístupového tokenu je následující

- Klient zažádá o přístup a povolení skrze vývojový kit a přihlašovací dialog.

- Klient se přihlásí a schválí jednotlivá povolení k přístupu.
- Aplikace získá přístupový token a může jej využít k manipulaci s profilem.



Obrázek 3.2: Přihlašovací dialog pro získání access tokenu

Funkce

Graph

Aplikační rozhraní Graph je jádrem platformy Facebook. Umožňuje vývojářům číst a zapisovat data do Facebooku. Facebook Graph API představuje graf, který zobrazuje jednotlivé objekty grafu, jako jsou lidé, události, fotky a zájmy. Rovněž popisuje spojitosti mezi těmito objekty. Takovými spojitostmi mohou být vztahy mezi uživateli, sdílený obsah a označení na fotkách.

Authentication

Facebook Authentication je nástroj pro vývojáře, který jim dovoluje nabídnout jejich aplikacím možnost interakce s Graph API pro právě přihlášeného uživatele. Rovněž poskytuje způsob jak se jedním přihlášením autentifikovat na webových, mobilních i aplikacích vytvořených pro stolní počítače.

Kapitola 4

Implementace

4.1 Úvod

Tato bakalářská bude demonstrovat použití aplikačního rozhraní společnosti Hi-Rez Studios pro jejich počítačovou hru Smite. Konkrétně se bude snažit aproximovat výsledek utkání dvou družstev hráčů na základě historie jednotlivých hráčů stažené z API.

Motivací pro výběr tohoto rozhraní byla nízká zkušenost vývojářského studia, které dosáhlo s touto hrou paradoxně velkého úspěchu a vybojovala si na trhu MOBA her třetí místo. Analyzovat strukturu dat poskytovaných API se pak jevilo jako zajímavou problematikou, dokud se nachází samotné rozhraní v ranné fázi. Motivací pro samotnou aplikaci byla možnost ověřit, zda známé algoritmy pro výpočet pravděpodobnosti na výhru nachází uplatnění i v počítačových hrách a zahrnout do výpočtu i další faktory, které mohou ovlivnit výsledek hry.

4.2 Smite

4.2.1 Základní informace

Smite je počítačová hra žánru MOBA, což znamená Multiplayer Online Battle Arena. Je vyvíjena společností Hi-Rez Studios pro operační systém Windows, konzoli Xbox One a aktuálně je plánovaná i podpora pro konzoli Playstation 4. Hra se liší od ostatních MOBA her tím, že se nehraje obdobně jako strategie, kde se na herní plochu hledí z vrchu. Ve hře Smite se zvolená postava, kterou hráč reprezentuje, ovládá z třetí osoby.

Hráč se zapojuje do týmové bitvy, v různých herních režimech. Vždy se jedná o utkání alespoň jednoho až pěti hráčů v každém týmu, který se snaží dosáhnout určitého cíle. Ovládaná postava se ve hře jsou bohové, které jsou obrazem bohů z reálných panteonů. Každá taková virtuální postava má různou sadu schopností.

Ve hře aktuálně registrováno přes 7,7 milionu hráčů a denně se odehraje přibližně 250000 utkání. Hra se tedy velmi rychle stala E-Sportem. Na prvním světovém šampionátu ve hře Smite se vybrala kolektivní cena 2,6 milionu dolarů, což byla v tento čas třetí nejvyšší cena v eSportech, která byla kdy nabídnuta. Vítěz turnaje si tedy odnesl něco málo přes 1,3 milionu dolarů.

4.2.2 Principy

Smite se skládá z několika herních režimů - největší z nich je Conquest. Hráči jsou ve všech režimech formování do dvou družstev. Vhodné složení týmu, spolupráce a vhodná strategie je vždy klíčem k vítězství. Pokud se daří hráčům dobře skloubit schopnosti jimi zvolených bohů, tak se jejich šance na vítězství zvyšuje. Hráči plní úkoly spojené s daným typem hry a snaží se současně paralyzovat druhé družstvo tím, že zabijí jeho členy. To je i jeden ze způsobů, jakým lze ve hře získat zlato. Dalším způsobem je zabíjení počítačem ovládaných vojáků, které se automaticky tvoří pro každé družstvo.

Za získané zlato lze zakoupit předměty, které mají různé vlastnosti a dále ovlivňují průběh a výsledek utkání. Pokud jedno z družstev nabude dojmu, že již pro ně není možné ukořistit vítězství a nebo pokud je hra nebaví, mohou hlasovat a vzdát se svému soupeři.

Herní režimy

Ve hře existuje několik herních režimů, které se dají rozdělit do několika typů. Cvičební hry, kde hráč může trénovat své vlastní schopnosti ovládat hru. Kooperativní hry, kde se tým hráčů utkává proti počítačem řízeným oponentům, normální režimy, kde se hráči utkávají proti sobě a pro více zkušené hráče existují hodnocené hry, kde se hráči mohou probojovat až do sportovní scény.

Conquest
Conquest je považován za hlavním módem hry, kde se utkává deset hráčů ve družstvech po pěti. Mapa je ve formě tří cest vedoucí z báze každého týmu a džungle. Toto rozložení je typické pro většinu MOBA her. Tento režim je považován za jeden z nejtěžších a je vhodný pro zkušené hráče. Úkolem hry je plnit úlohy a získat tak dostatečnou výhodu a porazit titána protihráčů.

Arena
Aréna je druhý nejpopulárnější režim hry, kde se utkávají dvě družstva o pěti hráčích na obou stranách. Mapa se neskládá z typických několika cest, ale jedná se o jednu ucelenou bitevní plochu. Každé družstvo začíná z 500 body a různými úkoly a zabíjením protihráčů je soupeři snižují. Tým, který přijde o všechny body prohrává.

Joust
Joust je režim hry, kde se nachází pouze jedna cesta a hrají tři hráči proti třem soupeřům. Mapa je zjednodušená, ale založená na podobném principu jako Conquest. Vítězí tým, který porazí titána soupeře.

Assault
Assault je třetím nejpopulárnějším režimem hry, kde se spolu utkávají dva týmy s pěti hráči. Rozdílem oproti ostatním hrám je, že si hráči svého boha nevolí, ale je jim přidělen na základě náhody. Rovněž není možné se vracet do své základny nakupovat předměty dokud hráč nezemře a nečeká na znovu zrození. Vítězem se stává tým, který se probojuje skrz mapu a porazí titána soupeře.

Siege
Siege je režim hry podobný Conquestu. Jedná se o mapu pro 8 hráčů, kteří jsou opět

rozděleni do dvou družstev po čtyřech hráčích. Účelem hry je opět rozdrcení obrany soupeře a poražení jejich titána. Rozdílným prvkem jsou obléhací jednotky, které lze v tomto režimu po splnění určitých kritérií získat.

Match of the Day
MOTD nebo-li match of the day jsou speciální druhy her, jejichž náplň se mění každý den. Jejich unikátnost je typicky změna herní mechaniky. Příkladem je snížení času mezi sesílání kouzel. Nebo neomezená energie nebo neomezené zlato.

Clash
V režimu Clash se utkají dvě družstva o pěti hráčích. Tento režim hry má být kompromisem mezi režimem Arena a Conquest. Kombinuje úkolové prvky režimu Conquest s rychlejší a akčnější Arenou.

Obrázek 4.1: Obrázek ze hry. Režim hry Joust.

Na obrázku 4.1 je obrázek z probíhající hry. Jedná se o režim hry Joust, kde se utkávají tři hráči. Předměty postavené hráčem se zobrazují na spodní levé straně uživatelského rozhraní. Uprostřed lze vidět hráčem ovládaného boha a jeho schopnosti. Na levé straně v horním rohu lze vidět kompletní sestavu týmu, zatímco na pravé straně je sestava soupeřů.

4.3 Návrh aplikace

Při návrhu aplikace bylo nutné zvážit různé aspekty jako například přenositelnost a jakým způsobem budou uživatelé aplikaci využívat. Rovněž bylo nutné vzít v úvahu limity, které API na vývojáře ve své základní funkcionalitě klade.

4.3.1 Použité technologie

Snadné doručení obsahu uživateli se ukázal jako jedním z klíčových prvků pro tuto aplikaci. Aplikace pro desktopové operační systémy se jevila problematicky, protože by bylo nutné vytvářet pro každého uživatele lokání databázi nebo stahovat databázi z dedikovaného serveru. Pokud by se využilo lokální databáze, nastal by problém s distribucí přihlašovacích údajů od API, jak již bylo popsáno v kapitole věnované aplikačnímu rozhraní hry Smite. Tyto údaje nelze sdílet s jinými vývojáři či uživateli. Dalším problémem by bylo poskytování aktualizací aplikace koncovým uživatelům. Nejjednodušším způsobem jak se vyhnout těmto problémům bylo vytvořit webovou aplikaci.

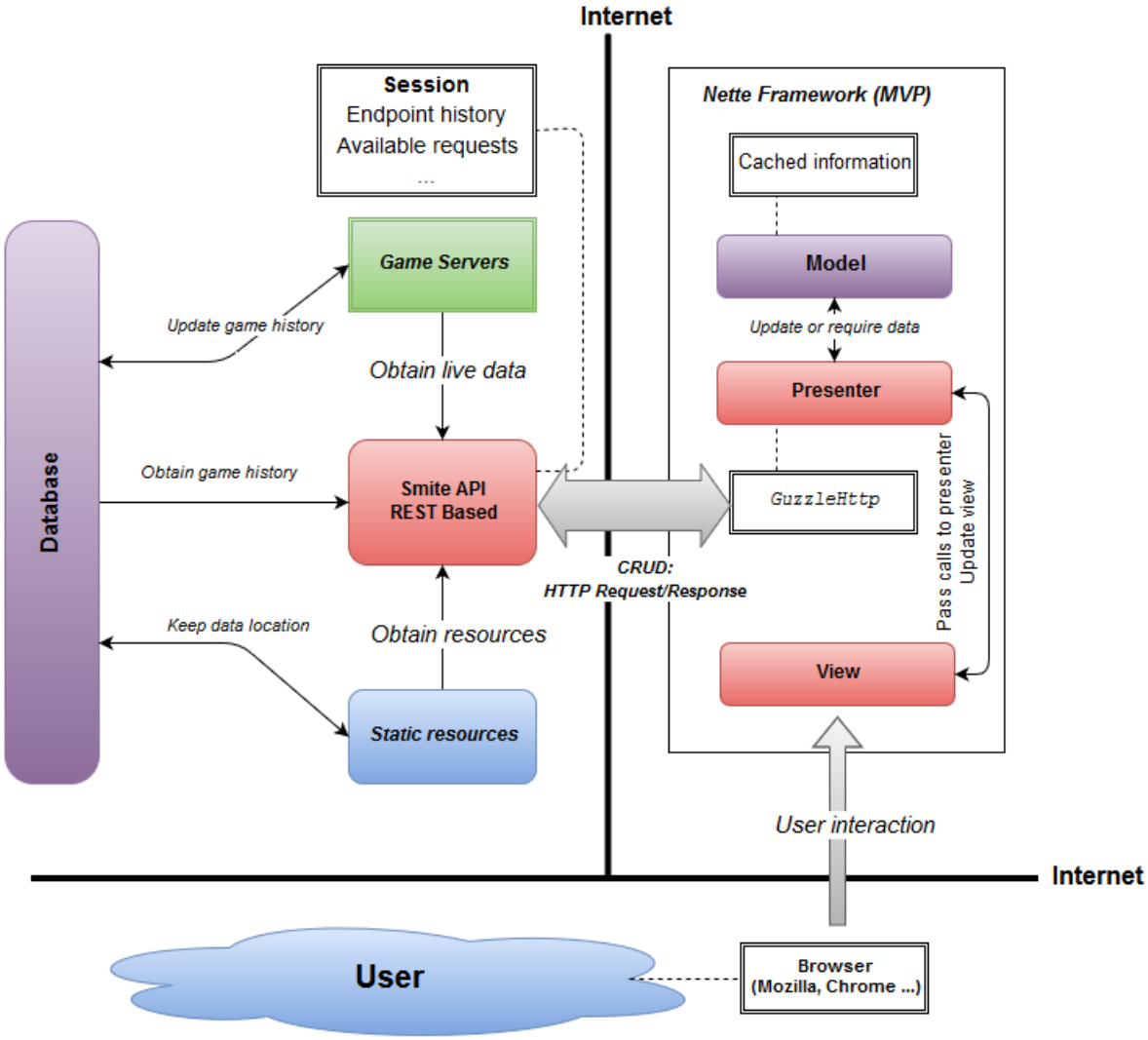
Pro tvorbu interaktivních webových aplikací se tedy jevil vhodný jazyk PHP [12] a pro ukládání dat stažených z API byl zvolen databázový systém MySQL[10]. Pro zachování referenční integrity je použit engine InnoDB. Pro snadné programování byl použit framework Nette [9], který je vybudován na architektuře MVP (Model-View-Presenter).

MVP pomáhá oddělit rozdělit aplikaci do tří vrstev. Na datový model, uživatelské rozhraní a řídicí logiku. Úpravy jednotlivých vrstev mají pak minimální vliv na ostatní.

View (pohled) převádí data reprezentovaná modelem do podoby vhodné k prezentaci uživateli. Controller (řadič) reaguje na události pocházející od uživatele a zajišťuje změny v modelu nebo v pohledu.

- Model zajišťuje přístup k datům a manipulaci s nimi.
- View (pohled) převádí data reprezentovaná modelem do podoby vhodné k prezentaci uživateli. A vytváří spojení mezi uživatelskými příkazy tak, aby presenter mohl nad daty vykonat nějakou akci.
- Presenter pracuje nad modelem i pohledem. Získává data z modelu a zpracovává je tak, aby mohly být zobrazeny ve View (pohledu)

K připojení k aplikačnímu rozhraní, které je, jak již bylo zmíněno, založeno na architektuře REST, je použita knihovna GuzzleHttp [2], která obaluje funkcionalitu HTTP klienta.



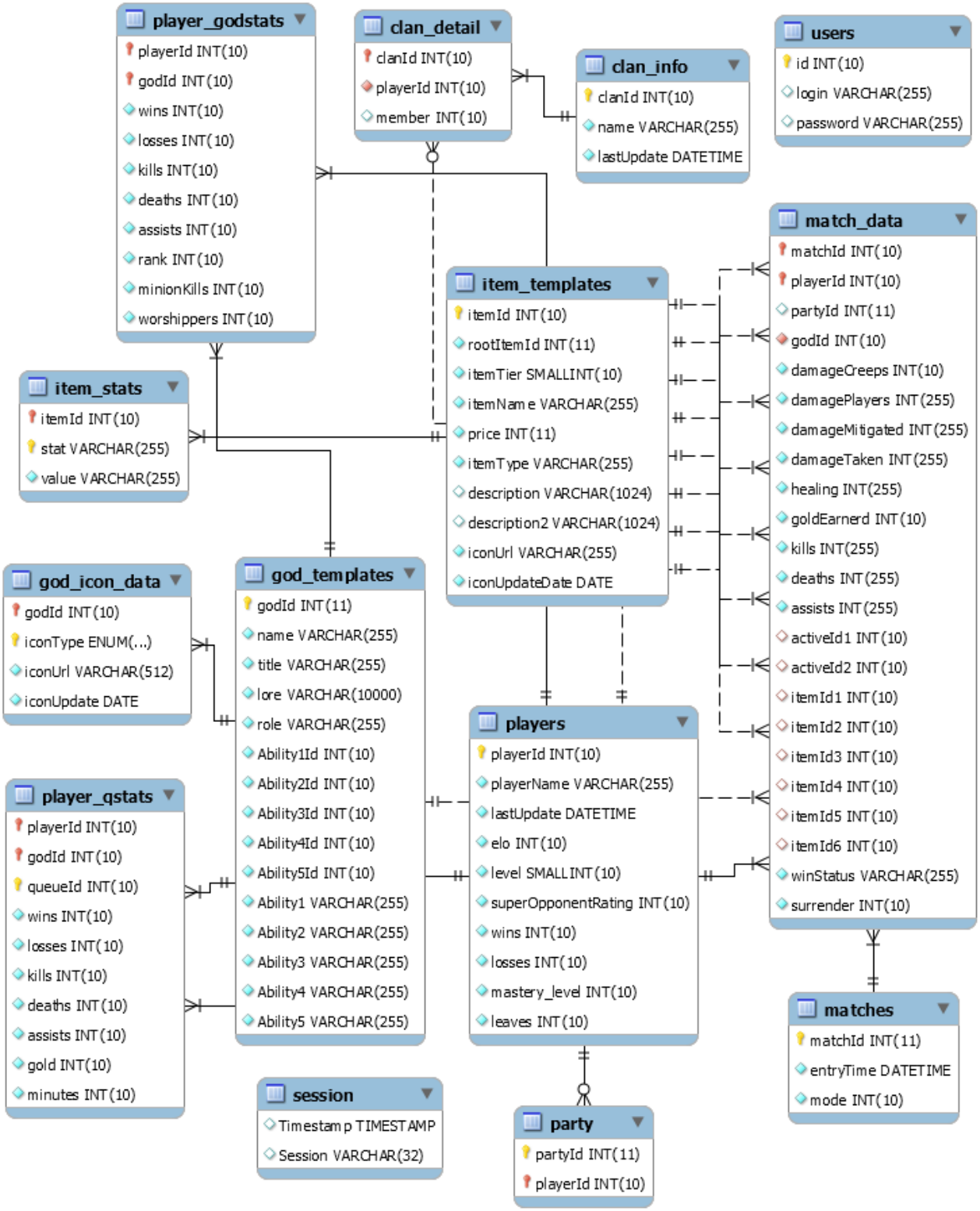
Obrázek 4.2: Diagram interakce

Na obrázku 4.2 je zobrazen diagram interakce všech zúčastněných stran. Uživatel navštíví aplikaci a je mu přes view zobrazen obsah. Po interakci jsou zavolány vazby v presenteru, který může nacachovaná data získat z modelu a nebo přes HTTP klient GuzzleHttp získat data z API. Na straně API se data získají v závislosti na požadavku z jednotlivých podsystémů. Po stažení dat se data ukládají do databáze, aby nebylo nutné je v nějakém časovém rozsahu získávat znovu.

4.3.2 Databázový model

Databázový model obsahuje všechny potřebné položky, které je vhodné ukládat při stahování dat z API. Slouží tedy jako datová cache, ale obsahuje i jiné data potřebné k správnému fungování aplikace. Některé položky aktuálně v aplikaci nejsou použity, ale při rozšiřování aplikace budou užitečné.

Na obrázku 4.3 lze v EER diagramu všechny entity a vztahy mezi nimi.



Obrázek 4.3: EERD (Enhanced Entity Relatioship Diagram) databáze

Entita players
V entitě players jsou uloženy základní informace o hráči. Nachází se zde jeho aktuální elo (viz. sekce 4.4.1), počet výher a proher apod. Rovněž se zde nachází atribut, který definuje poslední změnu. Ten je nutný, protože kvůli technickému omezení API nelze stahovat informace o uživateli úplně vždy. Aktualizace probíhá pokud je poslední změna starší než dvě hodiny.

Entita matches
V entitě matches je uloženo unikátní číslo utkání, čas vzniku hry a režim hry.

Entita match_data
Zde se ukládají informace o odehraných utkáních a jejich účastnících. Jsou zde zaznamenány výkony jednotlivých hráčů, jimi reprezentovaný bůh a zda se stali vítězi.

Entita party
V této entitě se ukládají informace, které dokazují, zda jednotliví hráči spolu hrají.

Entita god_templates
V této tabulce databáze jsou uloženy elementární informace o bozích obsažených ve hře. Slouží především pro přívětivější uživatelské rozhraní, ale jsou zde prvky, které mohou ovlivnit výpočet, jako například role.

Entita god_icon_data
Toto je pouze pomocná entita, kde se ukládají informace o ikonách, které se zobrazují v uživatelském rozhraní. Je zde typ ikony, URL kde ji lze stáhnout a poslední aktualizace.

Entita clan_info a clan_detail
Tyto tabulky obsahují informace o jednotlivých klanech, ve kterých mohou být uživatelé členové.

Entita player_godstats a player_qstats
V těchto entitách jsou obsaženy statistické informace o výkonech hráčů za jednotlivé bohy. Tabulka player_qstats se liší v tom, že jsou informace seskupeny podle herního režimu.

Entita item_templates a item_stats
V těchto entitách se nacházejí informace o předmětech obsažených ve hře. Jejich popisky, URL ikony a další základní informace. Rovněž jsou pro ně uchovávány v entitě item_stats jejich atributy a síla těchto atributů.

Entita users
Tato tabulka slouží k uchovávání informací o administrátorských účtech.

Entita session
Relace s API má omezenou životnost. Tato tabulka zajišťuje, že je vždy použita funkční relace. A v případě, že je starší než patnáct minut, je založena relace nová.

4.4 Aplikace

Tato aplikace si dala za úkol z dat stažených z API vypočítat hráčovu šanci na výhru. Hráč tedy spustí hru a zahájí samotné utkání. Poté může navštívit webové stránky aplikace a zadat své jméno do vstupního formuláře. Aplikace ověří, zda se hráč vůbec nachází ve hře a pokud ano, zahájí proces zjišťování informací o spoluhráčích a protihráčích.

Informace o jednotlivých hráčích mohou být již poměrně aktuální (tzn. mladší než dvě hodiny) a jsou tedy použity ty, které jsou nacachovány v databázi. Může však i nastat případ, kdy jsou informace starší anebo vůbec neexistují a v takovém případě je nutné kontaktovat API a stáhnout aktuální informace.

Aplikace kvůli technickým omezením ze strany API aktuálně podporuje pouze režim hry Joust, kde se utkávají tři hráči proti třem. Algoritmy a datová struktura takto omezena není a v zásadě ji lze aplikovat na jakýkoliv herní režim.

4.4.1 Algoritmus pro výpočet výhry

Základním elementem pro výpočet výhry je určitým způsobem přiřadit každému uživateli nějaké výkonnostní ohodnocení. Takové výkonnostní ohodnocení lze nazvat elo. Čím více a čím lepších soupeřů hráč porazí tím větší elo získává. V opačném případě, když hry prohrává, elo postupně ztrácí.

Pro výpočet tohoto ohodnocení výkonu je použit algoritmus, který používají některé šachové ligy. Tento algoritmus se nazývá **pravidlo 400** [15] [17]. Ve své čiré podobě je velmi jednoduchý a výkon hráče je vypočten následovně

1. Za každého poraženého hráče přičti jeho elo navýšené o 400 bodů.
2. Za každého hráče, který tě porazil přičti jeho elo snížené o 400 bodů.
3. Pokud remizuješ, přičti elo hráče, proti kterému jsi hrál.
4. Výslednou sumu poděl celkovým počtem her.

$$\text{Vykon hrace (elo)} = \frac{\text{Soucet el vsech protihracu} + 400 * (Vyhry - Prohry)}{Vyhry + Prohry} \tag{4.1}$$

Základní šanci na vítězství lze pak z tohoto výkonnostního ohodnocení vypočítat jako:

$$\text{Sance hrace A} = \frac{1}{1 + 10^{\frac{EloB - EloA}{400}}} \tag{4.2}$$

kde EloB a EloA jsou ela patřící jednotlivým hráčům.

Jelikož ve hře Smite proti sobě nebojují samotní hráči, ale tým hráčů, je nejprve před výpočtem potřeba spočítat elo super-hráče, který reprezentuje celý tým. To lze spočítat snadno jako průměr el.

$$SuperElo = \frac{\sum_{i=1}^{n} eloPlayer_i}{n} \tag{4.3}$$

kde n je počet hráčů v týmu.

Každý hráč začíná se základním elem 1500, protože je to ideální hodnota, od které lze odečítat i přičítat tak, aby výsledek výpočtu byl stále reálný. Pokud by elo začínalo například na 0 bodech, tak by se musely upravit výpočty pro práci se záporným elem a to není žádané. Ve hře Smite je však nutné také uvažovat, že v počátečních fázích jsou mezi hráči obrovské rozdíly. Proto aplikace aktuálně uvažuje, že hráči s méně než třiceti hrami mají statické elo 1500. V reálném nasazení by bylo vhodné uvažovat hráče s alespoň 60 hrami. Aplikace tedy uvažuje, že hráč strávil hrou již nějaký čas. Přibližně osm odehraných hodin. Nově staženým profilům hráčů je elo inicializováno podle předpisu. Tím je zaručeno, že zkušenější hráči získají do budoucích výpočtů trochu statistické síly.

$$Elo = \frac{\text{Pocet odehranych her} * 1500 + 400 * (Vyhry - Prohry)}{Vyhry + Prohry} \tag{4.4}$$

Pro další kalkulace jsou již využity informace o hrách, které lze stáhnout z profilu hráče. Což bývá typicky 0-50 odehraných her.

Ve hře však čistě matematicky vypočtená zkušenost protivníka není jediným prvkem, který má vliv na vítězství. Dalšími prvky ovlivňující vítězství je též zkušenost protivníků se svým bohem a také zda spolupráce s ostatními hráči. Dalšími prvkem je vhodné složení týmu. Ačkoliv nelze zjistit z aktuálně probíhajícího utkání informace, zda hráči tvoří jednu ucelenou skupinu, lze nahlédnout do historie her a tuto informaci dohledat a případně započítat. Tyto dohledané informace, co též mají vliv na výsledek hry lze uvažovat jako koeficienty a je potřeba jimi ovlivnit šanci na vítězství popsanou výše.

Jak již bylo zmíněno výše, aplikace zpětně dohledává společnou souhru hráčů v týmu a může tedy odhadnout, zda hrají spolu. Tento faktor je nazván *chemie* - podle chemie, která je mezi hráči. Nejprve je spočítán faktor nazvaný *souhra*. Pro herní režim 3v3 funguje přibližně následovně. Pokud hráč odehraje se spoluhráčem deset samostatných her, souhra bude prostých deset bodů. Pokud odehraje s jedním pět her a s oběma 10 her, tak souhra lze vyjádřit jako:

$$Souhra = 5 * 1 + 10 * 2 \tag{4.5}$$

Obecný zápis, který platí i pro všechny herní režimy je tedy:

$$Souhra = \sum_{i=1}^{n} gamesTogether * i \tag{4.6}$$

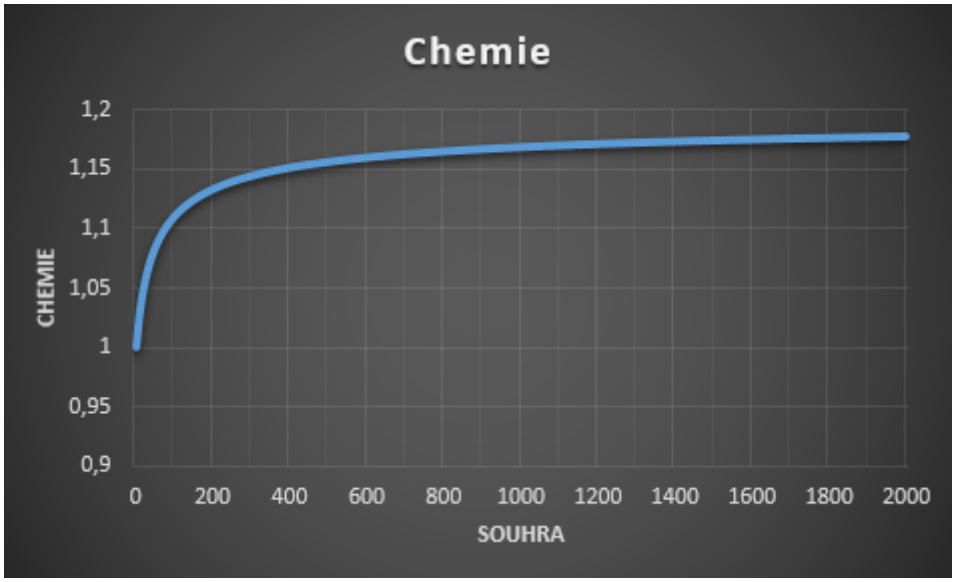
Kde n je počet spoluhráčů, kteří hráli společně v jednom týmu. Tento parametr se počítá právě takto, protože se snaží zvážit, že čím více hráčů hraje spolu, tím lepšího stmelení dosáhnou.

Chemie lze poté spočítat dle předpisu:

$$Chemie = \frac{-1}{\sqrt{Souhra + 25}} + 1.2 \tag{4.7}$$

$$Souhra \in <0, \infty)$$

$$Chemie \in <1, 1.20)$$



Obrázek 4.4: Průběh chemistry funkce

Funkce chemie má průběh lomené funkce. Je omezená 1.2 shora proto, aby tento parametr neovlivnil výpočet výhry příliš silně. Konstanta 25 posouvá celou funkci a zaručuje, že funkce pro souhru 0 vrací chemii 1. Základní myšlenkou této lomené funkce je hráčům hrajícím spolu vylepšit pravděpodobnost vítězství. Po velmi častém hraní ovšem jejich souhra už takový vliv na výsledek nemá a proto je v děliteli odmocnina.

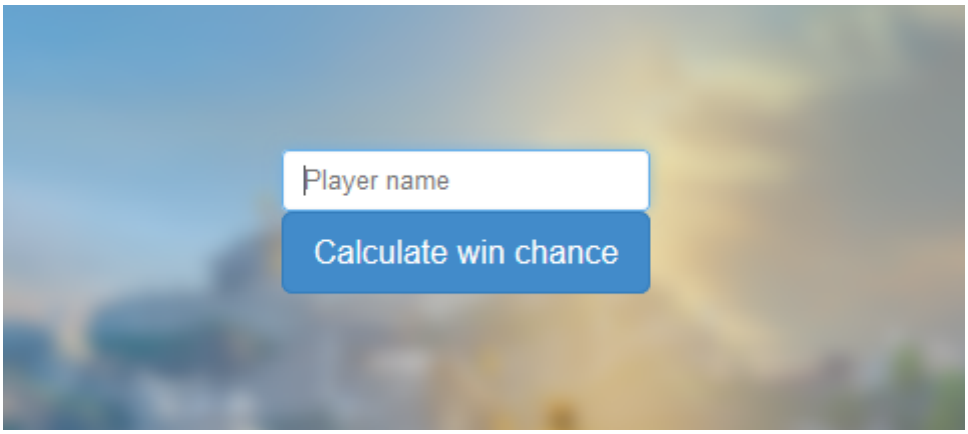
A výsledný ovlivňující koeficient pro uživatelův tým jako:

$$Coefficient = chemA - chemB + 1 \tag{4.8}$$

Kde chemA je chemie uživatelova družstva a chemB chemie jeho oponentů.

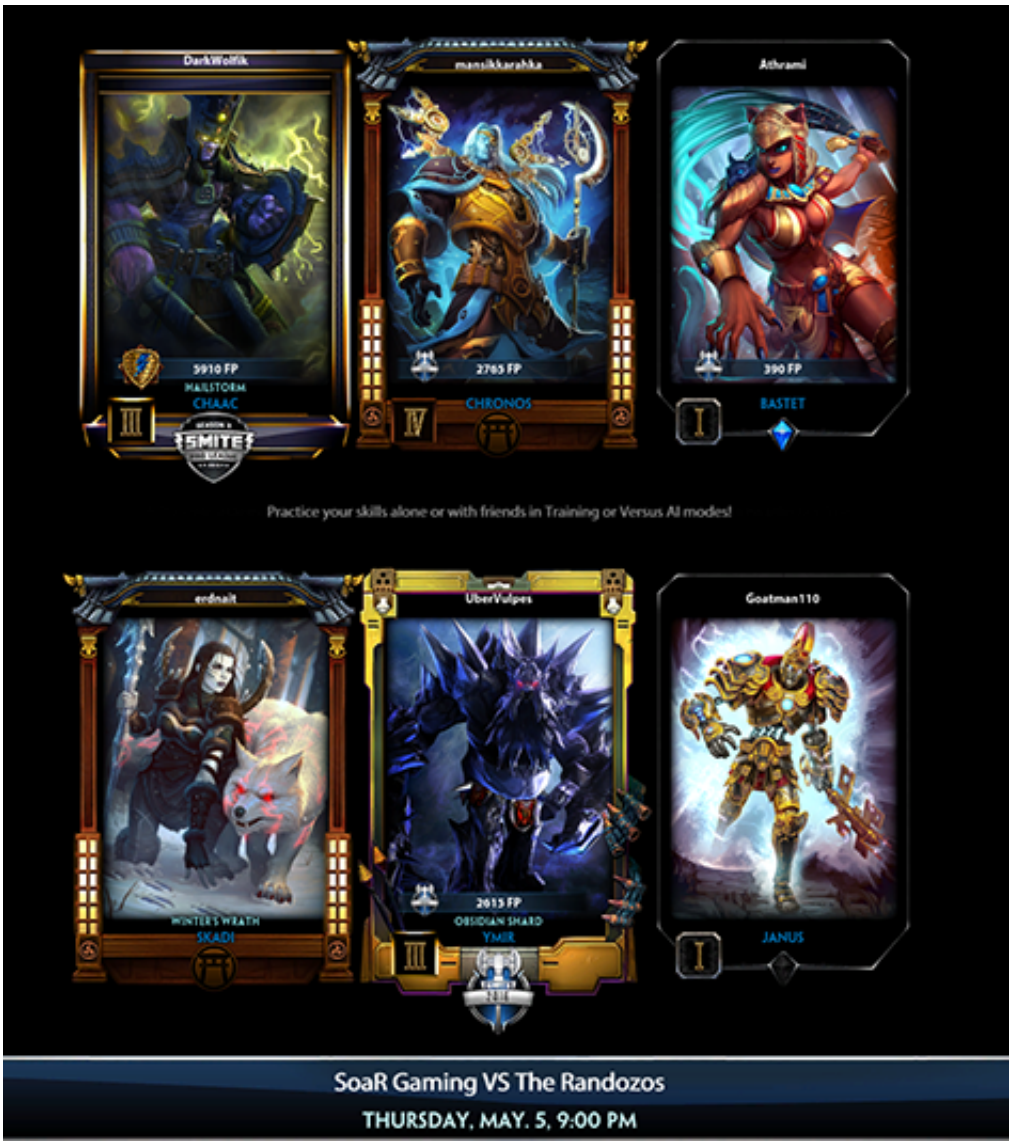
4.4.2 Uživatelské rozhraní

Uživatelské rozhraní by mělo být vytvořeno tak, aby bylo pro uživatele přívětivé a bylo na první pohled jasné, co se od něj očekává. Hráči je tedy na počátku zobrazen formulář tázající se na jméno hrajícího.



Obrázek 4.5: Vstupní formulář

Dalším krokem bylo nabídnout uživateli něco, co zná. Na obrázku 4.6 lze vidět obrazovku načítání ze hry. Je vidět rozdělení do dvou družstev a jaký hráč reprezentuje jakého boha. Rovněž zobrazuje i přibližnou zkušenost hráče se zvoleným bohem.



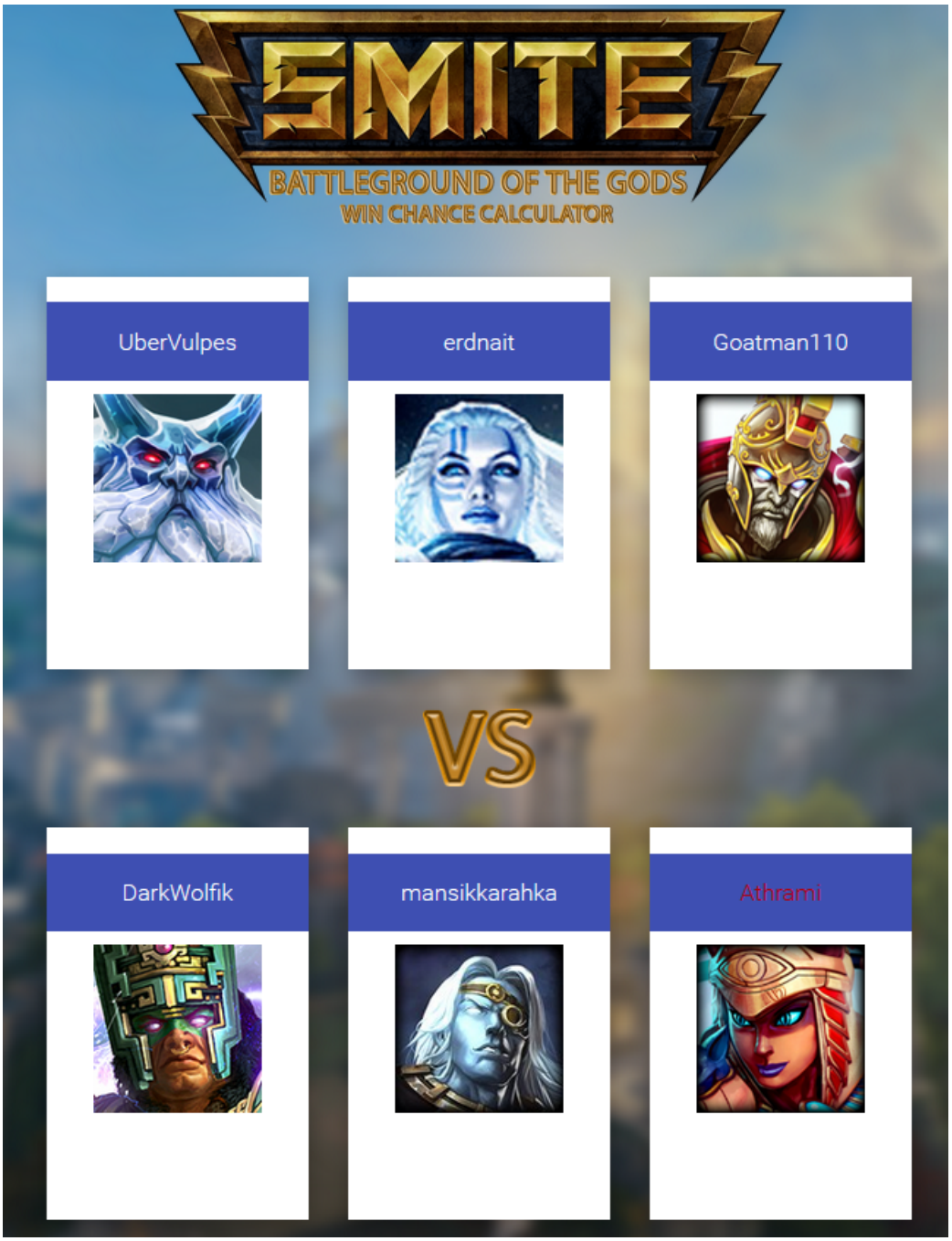
Obrázek 4.6: Načítací obrazovka ze hry, zobrazující obě družstva

Podobná zkušenost je nabídnuta hráči i v této aplikaci. Zobrazí se mu karty s jednotlivými bohy a reprezentující hráče. Ikony a další informace o bozích jsou staženy z API jednorázově a poté jsou dle potřeby aktualizovány. Na obrázku 4.7 lze vidět raný návrh, který by měl uživateli poskytnout podobný pocit, jaký má ve hře.

Nelze dosáhnout perfektní grafické shody, protože karty bohů jak je vidět ve hře by bylo nutné extrahovat z datových souborů hry a kompletně by se obešlo API. Koncový bod aplikačního rozhraní, který vrací informace o probíhajícím utkání rovněž neobsahuje natolik podrobné data, aby bylo možné určit jaký skin (obměna vzhledu boha) hráč používá.

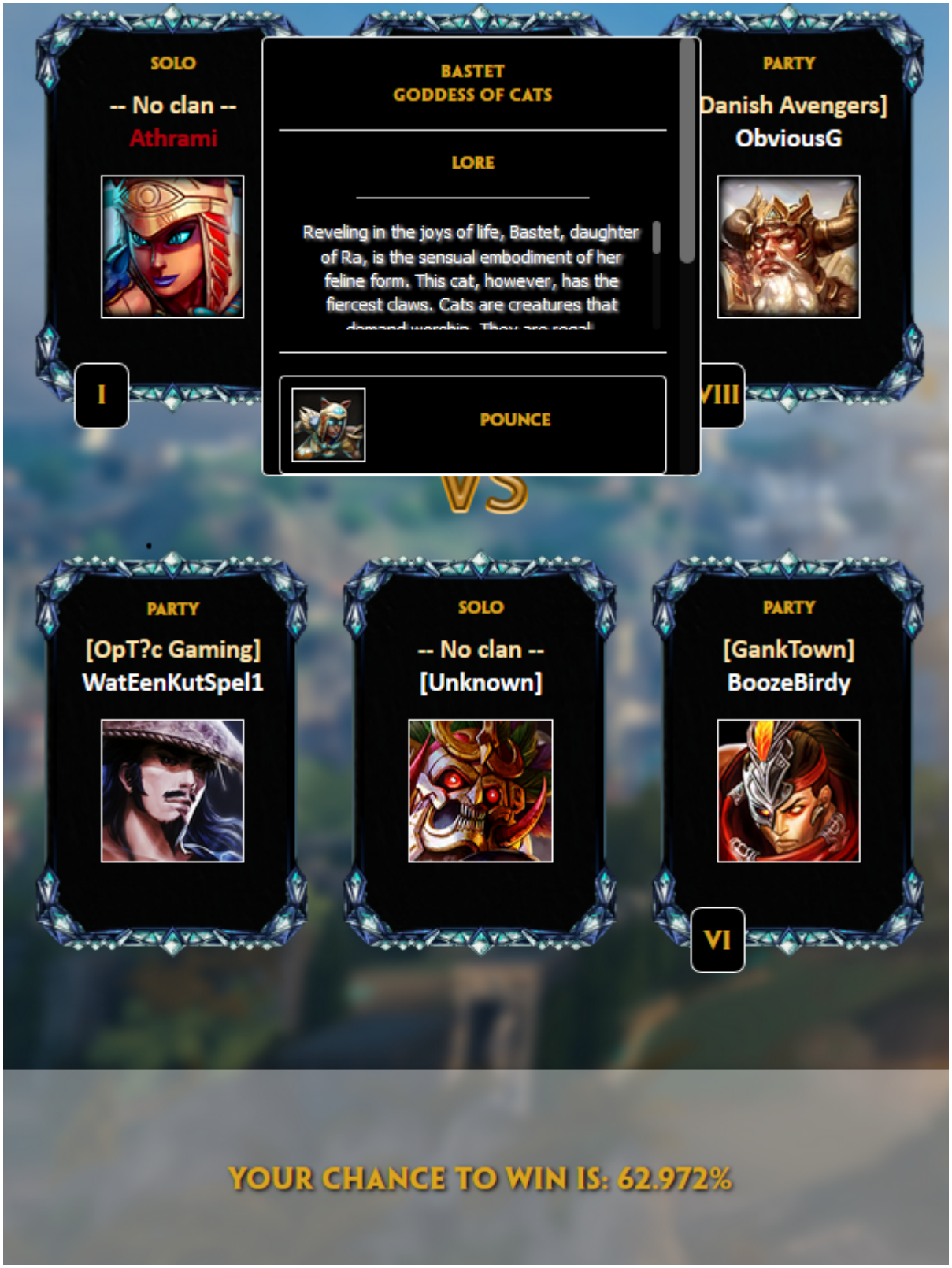
Lze však z databáze načíst informace o bohovi a při přesunutí kurzoru na ikonu boha zobrazit informace a schopnosti. To může být uživateli užitečné, pokud si není jist schop-

nostmi bohů svých oponentů.



Obrázek 4.7: Raný návrh rozhraní, rovněž zobrazující obě družstva

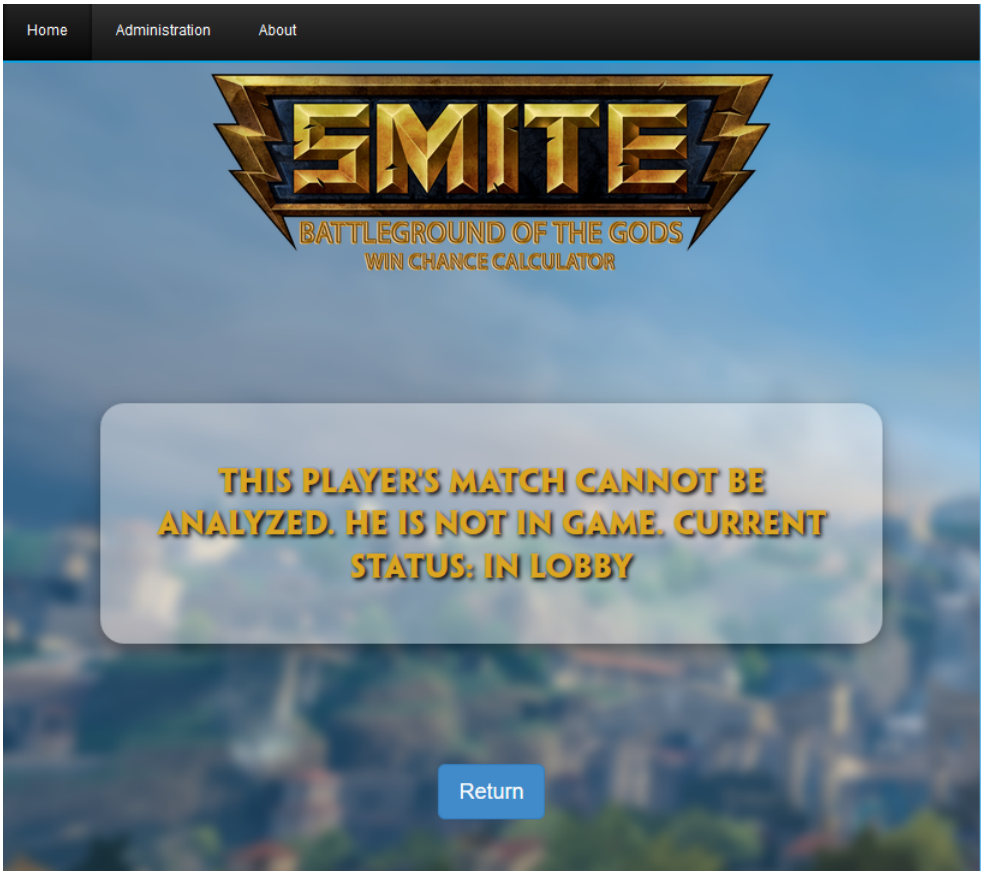
Další užitečným prvkem pro uživatele je, že mu lze zobrazit přibližné statistické informace o jeho oponentovi, včetně domněnky, zda soupeř vystupuje jako jedna skupina hráčů hrajících spolu. Detekce skupin uživatelů z historie her lze vidět na následujícím snímku z aplikace



Obrázek 4.8: Rozhraní ukazující informace k bohovi a detekci skupin a klanů

Hráč může pomocí těchto informací přizpůsobit styl své hry a maximalizovat svůj výkon.

Pokud si je vědom, že protivník hraje jako jeden tým, může předpokládat, že se soupeř pokusí o některou ze známých strategií, které mohou poskytnout jednomu družstvu výhodu.



Obrázek 4.9: Hráč není ve hře. Nelze mu zobrazit informace

Na obrázku 4.9 lze vidět i případ, kdy se uživatel nenacházel ve hře. V takovém případě nebo i případě, že hráč neexistuje je potřeba uživateli tuto skutečnost sdělit. Podobné zprávy se uživatel dočká i za předpokladu, že by zvolil jiný herní režim než Joust 3v3.

4.5 Testování

Nedílnou součástí vývoje je testování aplikace. K otestování této aplikace je potřeba navrhnout postup kterým se ověří několik aspektů aplikace. Přesnost výsledků - tedy zda aplikace dokáže skutečně odhadnout přibližně výsledek utkání a pokud ne, tak proč. Dalším aspektem je, zda uživatel je schopen bez obtíží aplikaci používat a chápe její výstupy.

4.5.1 Návrh

Jelikož aplikace poskytuje přesnější výsledky častějším používáním je dobré se při testování zaměřit na jednoho uživatele a získat o něm více informací a porovnat odhadnuté výsledky s reálnými.

Pro uživatelský test je navržený plán. Pro lepší zpětnou vazbu je ideální hráč třicáté úrovně (nejvyšší možné dosažitelné úrovně), která zaručuje, že má se hrou alespoň základní zkušenosti.

1. Spustit hru a zařadit se do fronty hráčů hledajících hru režimu Joust 3v3.
2. Zvolit svého boha a počkat na načítání hry a nebo přímo vstoupení do hry.
3. Přistoupit k aplikaci na zadané URL.
4. Vyplnění uživatelského jména hráče ve hře, pro kterého chce hráč spočítat šanci na výhru. Hráč by měl najít svou hru.
5. Pokud propočet proběhl bez problému hráč by měl být schopen odhadnout přibližnou šanci na výhru a zjistit jak silný soupeř je.
6. Hráč by měl porovnat a zhodnotit výsledek utkání s odhadnutým výsledkem.

Tento postup by měl být několikrát zopakován pro upřesnění výsledků.

4.5.2 Zjištěné informace

Pro testování bylo vybráno několik studentů hrajících hru Smite a dále několik hráčů, kteří se primárně zaměřují na herní režim Joust. Výsledky měření byly shrnuty do tabulky. Všechny měření proběhly na evropském serveru po získání dostatečného množství informací o hráčích a jejich obvyklých oponentech.

Data v tabulce představují testujícího hráče, jeho vypočítané elo před utkáním, v kolik hodin začal hru a délka samotné hry. Dále ukazuje zda hrál ve skupině a zda aplikace správně detekovala skupiny na obou stranách družstev. V poslední řadě je předpovězená hodnota a samotný výsledek utkání. Do tabulky byly vybrány takové hry, kde testující i protihráči hráli normálním způsobem a nesnažili se nějak zvrátit výsledek hry nesportovním chováním.

Testující	**Elo**	**Čas počátku**	**Délka hry**	**Skupina**	**Předpověď**	**Výsledek**
Student_1	1578	18:48	00:12:28	Ne/Ano	58.421%	Výhra
Student_1	1578	19:15	00:18:12	Ne/Ano	52.220%	Prohra
Student_1	1571	16:15	00:13:12	Ano/Ano	65.673%	Výhra
Student_2	1624	12:17	00:22:17	Ne/Ano	48.321%	Výhra
Student_2	1646	22:45	00:16:44	Ano/Ano	56.330%	Výhra
Student_2	1656	02:15	00:15:51	Ne/Ano	64.400%	Výhra
Hráč_1	1498	20:36	00:14:13	Ne/Ano	49.120%	Prohra
Hráč_1	1492	22:50	00:16:44	Ano/Ano	56.330%	Výhra
Hráč_2	1714	03:15	00:10:08	Ano/Ano	72.541%	Výhra
Hráč_2	1718	20:15	00:13:41	Ne/Ano	61.611%	Výhra
Hráč_3	1512	16:21	00:11:28	Ne/Ano	51.128%	Prohra
Hráč_3	1500	18:35	00:23:31	Ne/Ano	49.776%	Prohra

Tabulka 4.1: Výsledky testů

Z testování bylo zjištěno, že nelze se stoprocentní úspěšností určit, zda družstvo zvítězí, protože výsledky se obvykle pohybují v rozmezí 40-60%. Z toho lze například odvodit, že není úplně pravda, že by hra proti sobě velmi často stavěla nevyvážená družstva tak, jak se komunita okolo hry dohaduje. Dále je vidět, že v brzkých ranních hodinách, když je základna hráčů řidší, mají skupinky zkušenějších hráčů větší šanci na vítězství. Příčina tohot jevu je, že těmto skupinkám hra nemůže v adekvátním časovém rozsahu najít podobně silné soupeře.

Rovněž se objevil nepříjemný efekt, že pokud aplikace nemá k dispozici dostatečné množství informací o hráči, může poskytovat nepřesné výsledky. Pro dosažení lepších výsledků je vhodné, když jeden uživatel používá aplikaci častěji, čímž se získá více záznamů do databáze.

Aplikace byla rovněž vyzkoušena na ligové utkání, ale ukázalo se, že v profesionální sféře se nelze na výsledky spolehnout, protože v eSportech závisí na daleko více faktorech než pouze na těch, které jsou aktuálně v aplikaci obsaženy.

Nepříjemným zjištěním však byla limitace API, která se po intenzivním testování projevila i v herním režimu Joust 3v3. V produkčním nasazení by bylo nutné kontaktovat společnost Hi-Rez a nechat odstranit limit tak, jako byl odstraněn pro statistický web SmiteGuru, který byl již zmíněn dříve v sekci 3.1.1.

4.5.3 Zpětná vazba uživatelů

V testování se nelze ohlížet pouze na získané hodnoty, ale je nutné rovněž zhodnotit dojem uživatelů. Nejvíce uživatelé zmiňovali následující položky:

- Uživatelské rozhraní si uživatelé chválí a přijde jim dostatečně intuitivní.
- Uživatelé si cenní detekce skupin a zobrazení klanů, protože členství v klanu lze ve hře skrýt. Takto mohou lépe odhadnout koordinaci svého soupeře.
- Uživatelé si stěžují po prvním použití na velmi dlouhou dobu výpočtu. Ta je způsobená tím, že při prvním stažení všech informací o hráči může být odesláno až 55 dotazů na API, kde každý může trvat až 300ms.

Ze zpětné vazby uživatelů se ukázal největším problémem doba výpočtu. Získání čistých dat o všech zúčastněných hráčích může dohromady trvat v nejhorším případě i 300 sekund, což je velmi dlouhá doba. V případě, že jsou data o hráčích přibližně aktuální, celkový proces zabere daleko méně času.

Tento jev by možná bylo možné částečně obejít stahováním dat pro každého uživatele pomocí vláken, ale to by vyžadovalo pokročilou synchronizaci, aby se nestahovala jedna data z API dvakrát. Další možností by bylo zobrazit uživateli předběžnou statistiku a nástřel ze základních dat a po dokončení stahování všech informací by si uživatel nechal zobrazit zbytek informací a nový odhad.

Kapitola 5

Závěr

Cílem práce bylo popsat obecně vzory a architektury pro aplikační rozhraní a nad některým existujícím vytvořit aplikaci. Toho se podařilo dosáhnout a každá kapitola práce se poté zabývá danou částí problematiky.

V rámci bakalářské práce byly zjištěny informace o různých architekturách aplikačních rozhraní. Především byla popsána architektura REST, která je v dnešní době moderní a zároveň nebyly opomenuty ani architektury, které už patří do historie. Popsány byly i datové formáty, které rozhraní používají, a také byly rozepsány užitečné praktiky. Dále byly uvedeny existující API a nejpodrobněji se práce zaměřila na ty, které se týkají herních platforem. Okrajově bylo rozebráno i rozhraní největší sociální sítě na světě, Facebook. Z těchto rozhraní pak bylo vybráno právě jedno, které bylo použito pro implementaci aplikace.

V implementační části práce poté byla navržena aplikace, která by počítala pravděpodobnost výhry jednotlivých družstev pro hru zvanou Smite. Za výhodu aplikace je považováno zobrazení užitečných informací pro uživatele a ověření použitelnosti rozhraní. Bylo zjištěno, že přesnější šanci na výhru nelze z informací, které aktuálně API poskytuje, s jistotou zjistit. Vypočítané pravděpodobnosti se typicky pohybují v rozmezí 40-60%. Z tohoto samotného údaje nemůže hráč určit, zda zvítězí. Může však využít zobrazené informace a odhadnout, jaké úsilí bude muset vynaložit, aby dosáhl co nejlepších výsledků.

O samotném rozhraní od společnosti Hi-Rez bylo zjištěno také několik poznatků. Pro uživatele rozhraní je zřetelné, že je API stále ještě v rané fázi vývoje. Nedrží se osvědčených praktik a ani dostupnost a spolehlivost nejsou jeho klíčovými rysy. V některých případech poskytuje duplicitní data. Propojení jednotlivých částí systému však funguje korektně a vytvořená aplikace je důkazem, že lze nad tímto rozhraním realizovat i neobvyklé myšlenky.

Pokračováním práce by mohlo být rozšíření aplikace o další možné heuristiky, které by zohlednily ve výpočtu další prvky hry. Lze zahrnout například třídu bohů a nebo po více dosažených her hledat nejefektivnější možné kombinace předmětů, které by hráči mohly pomoci k vítězství.

Kapitola 6

Literatura

[1] Burke, B. *RESTful Java with JAX-RS 2.0.* 2. vyd. Beijing: O'Reilly Media, 2013. ISBN 978-1-4493-6134-1.

[2] Dowling, M. *Guzzle, an extensible PHP HTTP client* [online]. 2016 [cit. 2016-04-13]. Dostupné na: <https://github.com/guzzle/guzzle>.

[3] DuVander, A. *New Job Requirement: Experience Building RESTful APIs.* San Francisco: MuleSoft, 2010-06-09 [cit. 2015-11-01]. Dostupné na: <http://www.programmableweb.com/news/new-job-requirement-experience-building-restful-apis/2010/06/09>.

[4] Facebook Inc. *Facebook for Developers* [online]. Cambridge (Massachusetts): Facebook Inc., 2016 [cit. 2016-04-13]. Dostupné na: <https://developers.facebook.com/>.

[5] Facebook Inc. *Facebook for Developers* [online]. Cambridge (Massachusetts): Facebook Inc., 2016 [cit. 2016-04-05]. Dostupné na: <https://developers.facebook.com/docs/facebook-login/access-tokens/>.

[6] Hi-Rez Studios, Inc. *Smite* [online]. Atlanta: Hi-Rez Studios Inc., 2016 [cit. 2016-04-13]. Dostupné na: <http://www.smitegame.com/>.

[7] Kris, Z. *JSON Referencing in Dojo. SitePen* [online]. Palo Alto: SitePen Inc. [cit. 2015-12-25]. Dostupné na: <https://www.sitepen.com/blog/2008/06/17/json-referencing-in-dojo/>.

[8] Mulloy, B. *Apigee* [online]. Detroit: Apigee Corp [cit. 2015-12-22]. Dostupné na: <https://pages.apigee.com/rs/apigee/images/api-design-ebook-2012-03.pdf>.

[9] Nette Foundation. *Nette FrameWork* [online]. 2016 [cit. 2016-04-13]. Dostupné na: <https://nette.org/cs/>.

[10] Oracle Corporation. *MySQL* [online]. Redwood Shores: Oracle Corporation, 2016 [cit. 2016-04-13]. Dostupné na: <https://www.mysql.com/>.

[11] Riot Games. *League Of Legends* [online]. Santa Monica: Tencent, 2016 [cit. 2016-04-13]. Dostupné na: <http://eune.leagueoflegends.com/>.

[12] The PHP Group. *PHP: Hypertext Preprocessor* [online]. Cupertino (California): Zend Technologies, 2016 [cit. 2016-04-13]. Dostupné na: <http://php.net/>.

[13] Valve Corporation. *Steam* [online]. Bellevue (Washington): Valve Corporation, 2016 [cit. 2016-04-13]. Dostupné na: <http://store.steampowered.com/>.

[14] Valve Corporation. *Valve Developer Community* [online]. Bellevue (Washington): Valve Corporation, 2016 [cit. 2015-12-22]. Dostupné na: <https://developer.valvesoftware.com/wiki/KeyValues>.

[15] Vovk, B. *Chess Elo Rating in Simple* [online]. 2008-2015 [cit. 2016-03-03]. Dostupné na: <http://www.chesselo.com/index.html>.

[16] W3C. *Extensible Markup Language (XML) 1.0 (Fifth Edition)* [online]. Cambridge (Messachusetts): W3C, 2008 [cit. 2015-12-25]. Dostupné na: <http://www.w3.org/TR/REC-xml/>.

[17] Wikimedia Foundation Inc. *Elo rating system* [online]. San Francisco: Wikimedia Foundation Inc., 2016 [cit. 2016-03-03]. Dostupné na: <https://en.wikipedia.org/wiki/Elo_rating_system>.

[18] Wilde, E. *Putting Things to REST*. Barkeley: School of Information, 2007. Dostupné na: <http://dret.net/netdret/docs/wilde-irep07-015-restful-things.pdf>.

Příloha A

Obsah CD

- **latex_src** obsahuje zdrojový kód technické zprávy napsaný v LaTexu.
- **src** obsahuje zdrojový kód aplikace.
- **manual.txt** obsahuje popis potřebných nástrojů a postup ke zprovoznění aplikace.
- **projekt.pdf** je technická zpráva.