VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ANOTACE OBRAZU A VIDEA FORMOU HRY

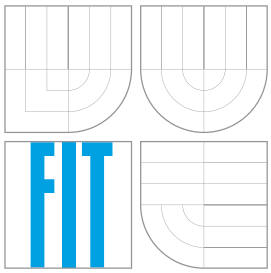
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. et Bc. ONDŘEJ SKOWRONEK

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ANOTACE OBRAZU A VIDEA FORMOU HRY

IMAGE AND VIDEO ANNOTATION AS A GAME

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. et Bc. ONDŘEJ SKOWRONEK
AUTHOR

VEDOUCÍ PRÁCE — Doc. RNDr. PAVEL SMRŽ, Ph.D.
SUPERVISOR

BRNO 2014

Abstrakt

Tato práce se zabývá problémem získávání anotací k obrazu a videím. Tento problém řeší za pomocí crowdsourcingové přístupu. Pro řešení tohoto problému byly v práci navrženy a implementovány crowdsourcingové hry pro získávání anotací. Tyto hry v testování prokázali velkou kvalitu získaných anotací a oblíbenost u uživatelů. Spuštění těchto her v širším měřítku by mohlo přispět k vytvoření velké databáze anotovaných videí a obrázků.

Abstract

This master thesis is oriented on a problem of creating video and image annotations. This problem is solved by crowdsourcing approach. Crowdsourcing games were designed and implemented to make solution of this problem . It was proven by testing that these games are capable of creating high quality annotations. Launching these games on a larger scale could create large database of annotated videos and images.

Klíčová slova

crowdsourcing, anotace obrazu, anotace videa, bounding box

Keywords

crowdousrcing, image annotation, video annotation, bounding box

Citace

Ondřej Skowronek: Anotace obrazu a videa formou hry, diplomová práce, Brno, FIT VUT v Brně, 2014

Anotace obrazu a videa formou hry

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Doc. RNDr Pavla Smrže Ph.D.

.....
Ondřej Skowronek
28. května 2014

Poděkování

Chtěl bych poděkovat svému vedoucímu panu Doc. RNDr Pavlovi Smržovi Ph.D. za poskytnutí odborné pomoci.

Obsah

Kapitola 1

Úvod

Systémy na rozpoznávání obrazu jsou dnes na velice vysoké úrovni. Jejich výsledky se dramaticky zlepšily díky rychlému vývoji počítačových technologií a výzkumu v oblasti zpracování obrazu. Dokážou spolehlivě rozpoznat obličeje, postavy a další. To vede k jejich širokému praktickému využití. Přesto musí systémy na rozpoznávání obrazů překonat mnoho dalších problémů, například: velké množství cílových objektů, hodně vzdálené nebo zakryté cíle. Dalším problémem je třeba také to, že systém na rozpoznávání obrazu dokáže rozpoznat pouze skutečné objekty, abstraktní objekty rozpoznat nedokáže. [19].

Přesto mají programy ještě velké nedostatky, které se dají zlepšit. Jednou z možností jak vylepšovat rozpoznávací algoritmy, je metoda strojového učení. Pro použití této metody je potřeba, aby rozpoznávací algoritmy měly k dispozici databázi obrázků s již rozpoznanými výsledky. Programy poté dokážou rozpoznat podobné obrázky, které mají v databázi a upravit podle toho výsledky.

Tento přístup je efektivní, ale má jeden velký problém. Je potřeba, aby tyto databáze s obrázky a jejich anotacemi měly obrovský rozsah. Čím větší databázi budou mít algoritmy k dispozici, tím lepších výsledků mohou dosáhnout. Není ale jednoduché je vytvořit. Klasický způsob, kdy se vyškolí a vytrénují speciální anotátoři, kteří budou obrázky anotovat, je finančně velmi neefektivní. Na vytvoření dostatečně velkých databází je totiž potřeba velký počet těchto anotátorů.

Tento problém se snaží řešit crowdsourcing přístup, kdy se využívá vědomostí široké škály lidí. Využití těchto lidí ale neprobíhá tak, že by dostávali finanční odměnu, nýbrž jsou odměňování tím, že se vytvoří takové podmínky, aby bylo anotování příjemným zážitkem. Jinými slovy vytvoří se crowdsourcing hra, jejímž hraním se budou vytvářet anotované obrázky. V hrách je velmi důležité motivovat uživatele, aby ji nepřestal hrát a tím vytvářel co nejvíc anotací.

Crowdsourcing hrami se bude zabývat i tato diplomová práce. Cílem této práce bude seznámit se s problematikou crowdsourcing her. Zjistit, jak by byly crowdsourcing hry vhodné pro získávání anotací pro algoritmy strojové učení. Cílem je vybrat způsob získávání anotací, které budou užitečné, a bude je moci vytvářet jakýkoliv nezkušený uživatel.

Zjistit, jaké hry by bylo vhodné vytvořit, jejímž hraním by byla tvorba nejvhodněji anotovaných obrázků. Po analýze a seznámení se s problematikou je hlavním cílem navrhnout a implementovat takovéto hry. Pro hry bude důležité, aby byly zábavné a lákavé pro účastníky, protože s větším počtem hráčů vznikne větší počet anotací. Taky výsledné anotace budou kvalitnější a budou lépe využitelnější pro algoritmy strojového učení.

Tato diplomová práce se bude skládat z několika kapitol. V další kapitole se budu zabývat problematikou crowdsourcingu a algoritmů strojového učení. Udělám průzkum již

existujících crowdsourcingových her a s těmi nejúspěšnějšími seznámím čtenáře.

V třetí kapitole určím potřebné technologie, které využiji pro implementaci herního systému. Také zde popíšu, jak dané technologie fungují a proč jsem si je vybral.

Čtvrtá kapitola bude obsahovat popis implementace herního systému. Implementace popíšu z pohledu klienta i serveru.

V páté kapitole přiblížím jednotlivé části implementovaného herního systému. Budou zde popsány všechny jeho výhody, které svým uživatelům nabízí. Bude zde nastíněno jak vypadá výsledná UI a jak s ní uživatelé mohou pracovat.

Šestá kapitola se bude zabývat uživatelským testováním, popíšu zde jakým způsobem jsem uživatelské testování prováděl. Závěrem této kapitoly zmíním jaké byly připomínky uživatelů a jaké jsem kvůli nim udělal změny.

V předposlední kapitole vyhodnotím přínos implementovaných her pro sběr dat. Bude zde vysvětleno jak probíhalo testování implementovaných her a jaké jsem získal v rámci testování data.

Poslední kapitolou je závěr, ve kterém jsou shrnuty dosažené výsledky.

Kapitola 2

Crowdsourcing

2.1 Crowdsourcing

Crowdsourcing je postaven na základním, ale mocném konceptu: Úplně každý má potenciál zapojit se do projektu a vytvořit nějakou hodnotu. Základní myšlenkou crowdsourcingu je využití široké veřejnosti pro splnění úkolů, na které by bylo třeba si někoho najmout a zaplatit mu za jeho práci [17]. Pracovní síla zdarma ovšem není jedinou výhodou crowdsourcing přístupu, další výhodou je také využití masové škály rozličných lidí. Vznik crowdsourcingu jde ruku v ruce s rozvojem internetu, který většinou slouží jako komunikační kanál a který umožňuje, aby se mohl připojit úplně každý. Typické znaky pro crowdsourcing jsou následující:

- Společnost s problémem či úkolem, který potřebuje splnit
- Komunita, která je schopna a ochotna se podílet na plnění daného úkolu
- Online prostředí umožňující komunitě pracovat a komunikovat se společností
- Spolupráce mezi společností a komunitou je pro obě dvě strany prospěšná

Poslední bod je velmi důležitý, neboť bez toho by neměla komunita důvod pro společnost pracovat. Pro společnost je tedy důležitým úkolem správně komunitu motivovat.

2.2 Strojové učení

Algoritmy strojového učení mohou vyřešit otázku, jak plnit důležité úlohy zobecňováním z příkladů. Toto je často jednodušší a cenově efektivnější než manuální programování. Čím více příkladů bude mít algoritmus k dispozici, tím složitější problémy může vyřešit. Díky tomu jsou algoritmy strojového učení široce používány ve výpočetní technice a v dalších oborech. Algoritmy strojové učení jsou používány při webovém vyhledávání, spam filtrech, cílenému dodávání internetové reklamy, detekování podvodníků, na trhu s akciemi, při modelování léčiv a mnoha dalších aplikacích [13].

2.3 Návrh anotovaných obrázků

Anotace k obrázkům, které popisují, co daný obrázek obsahuje, jsou velice užitečné. Pomáhají totiž takovéto obrázky jednoduše vyhledat. Anotací obrázků se například zabývá

Google, který poté umožňuje uživatelům vyhledávat obrázky podle klíčových slov. Anotování těchto obrázků ale není dokonalé, Google totiž používá především pro anotaci obrázků text, který se nachází na dané stránce s obrázkem. Ne vždy ovšem daný text souvisí přímo s daným obrázkem, někdy totiž ani k obrázku žádný text neexistuje. Potom dochází k tomu, že uživatel daný obrázek nenajde nebo najde obrázek, který se vůbec nevztahuje k hledané věci. Proto jsou kvalitní anotace k obrázkům velice užitečné a cenné, neboť není jednoduché je získat.

2.4 Návrh anotací videí

Neexistuje mnoho crowdsourcing her, které by získávaly anotace k videím. Většina crowdsourcingových her se vztahuje pouze na získávání anotací k obrázkům. Vytváření anotací k videím je také složitější. Uživatel musí shlédnout celé video, aby ho mohl anotovat, což se ve velkém počtu videí promítne na rychlosti anotace. Přesto jsou anotace k videím také užitečné. Podobně jako u obrázků, lze rovněž využívat anotace k videím pro rychlejší vyhledávání v archivu videí. Takové vyhledávání by například využil známý server s videi Youtube. Zde lze vyhledávat videa pouze pomocí názvů, které jim dali jejich tvůrci. Uživatelé však bohužel nemusí vložené video pojmenovat nejvhodnějším způsobem, což se projeví na obtížnějším vyhledávání videí. V případě anotací všech videí by se služby vyhledávání videí na Youtube zlepšily. Youtube by tak mohlo zavést i službu vyhledávání jednotlivých částí videí, kdy by se nalezla potřebná činnost i v třeba hodinu trvajícím videu.

2.5 Crowdsourcing projekty

Získávat anotace k videím a obrázkům za využití široké masy uživatelů není nápad úplně nový. Několik projektů postavených na této myšlence již už vzniklo. Chtěl bych zde některé z nich zmínit, protože jsou dobrým zdrojem pro inspiraci.

2.5.1 ESP game

ESP game je crowdsourcingová hra, která slouží pro anotování obrázků. V této hře se setkávají dva anonymní hráči, kteří pozorují stejný obrázek. Poté musí popsat, co se na obrázku nachází, snaží se popsat objekt tak, aby napsali to stejné co spoluhráči. V případě, že se shodnou, lze považovat za jasné, že tento objekt se na obrázku opravdu nachází. Při výzkumu tohoto způsobu anotace zjistili vývojáři ESP game, že je velice úspěšný, většina takto získaných anotací byla správných. Tento projekt byl dokonce tak úspěšný, že tuto hru odkoupil Google, aby ji využil pro vytvoření služby pro vyhledávání obrázků podle textu [27]. Proto jsem se rozhodl, že bych mohl vytvořit hru, na podobném principu.

2.5.2 Peekaboom

Pro machine learning algoritmy, jsou úžasnější takové anotace u obrázků, které obsahuje vymezení, kde se daný objekt nachází. Tento typ anotací se snaží získávat hra Peekaboom. Tuto hru hrají dva hráči, jeedn je Peek a druhý je Boom. Úkolem hráče Boom je postupně ukazovat hráči Peek části obrázku. Peek vidí jenom odkryté části a má za úkol uhádnout slovo, které se na obrázku nachází. Toto slovo nemusí být jenom objekt, může pouze s obrázkem souviset. Proto je úkolem Booma, aby Peeka dobře nasměroval a vybral vztah toho slova s daným obrázkem. Boom může také nasměrovávat Peeka tím, že osvětlí určité části

obrázku a nebo bude říkat u Peekových typů jestli se blíží nebo vzdaluje. Pro testování výsledků hry Peekaboom porovnávali tvůrci bounding boxy získané hrou a bounding boxy, které dobrovolníci vyznačili na obrázcích. Zjistili, že mezi těmito bounding boxy jsou rozdíly jenom minimální, takže data získané touto hrou vyhodnotili jako dostatečně kvalitní [28].

2.5.3 Guess What?

Guess What? je Facebooková hra ve které je hráčům ukázáno krátké video. Potom dostanou o tomto videu otázku. Cílem této hry je získat body. Hráči získávají určitý počet bodů za každou zodpovězenou otázku, ale aby se jim to povedlo, musí uhádnout jak na tyto otázky odpoví většina lidí. Jsou dva typy otázky, uzavřené a otevřené. Počet získaných bodů u otázek s možnostmi závisí na počtu stejných odpovědí ostatních hráčů . U otázek otevřených dostanou hráči 100 bodů za nejčastější odpověď, 75 bodů za již existující odpověď a 25 bodů za úplně novou odpověď. Výzkumníci stojící za touto hrou zjišťovali míru shodnosti mezi hráči. Použili proto Krippendorffovu metodu, která vrátila výsledek shodnosti jako 0.702, což značí velmi dobrou shodnost [25].

2.5.4 Steve museum

Steve museum je projekt, do kterého je angažováno společně několik muzeí. Tyto muzea se snažili nabídnout své kolekce (fotografie soch, obrazů a dalších muzejních předmětů) širokým masám uživatelů pomocí internetových stránek. Narazili ale na problém, jak uživateli svou širokou databázi nejrůznějších výtvorů nabídnout. V počátečních fázích nabízeli uživatelům své kolekce pomocí rozhraní, které spíše připomínalo databázi pro personál. Pro uživatele bylo velice složité najít objekt, který ho zrovna zajímá, v obrovské databázi dat, bez jakékoliv možností efektivního vyhledávání.

Proto vznikl tento projekt, který má za úkol pomoc muzeím označit všechny materiály dodatečnými informacemi, podle kterých by pak bylo snadné tyto materiály vyhledávat. Aby toho docílili, rozhodli se využít širokou masu lidí. Úkolem dobrovolníků, kteří chtěli pomoct tomuto projektu, pak bylo ke každému obrázku přiřazovat klíčová slova, napsat tedy takové objekty, kterými se obrázek vyznačoval [11].

2.6 Imagenet

Imagenet je velká obrázková databáze. Obrázky jsou zde spojeny se slovy (synsety) ze slovníku Wordnet. Imagenet má data přístupná pro širokou veřejnost. Tato databáze byla vytvořena, aby dala všem výzkumníkům dostatečný počet obrázků pro jejich výzkum. Cílem tohoto projektu je, aby každé slovo ze slovníku Wordnet obsahovalo více než 1000 obrázků. Takový způsob anotování obrázků, kdy se propojí obrázek s konkrétním slovem ze slovníku, shledávám jako neužitečnější, proto jsem se rozhodl, že mnou sbírané anotace budou vypadat stejně. Každý obrázek bude mít přiřazeno jedno slovo ze slovníku Wordnet. Tato stránka je rovněž vhodným zdrojem pro použití testovacích obrázků pro mou hru [12].

2.7 Wordnet

Wordnet je slovník, jehož základní jednotkou jsou synsety. Synset je soubor synonym. Synsety ve Wordnetu mají mezi sebou určité vazby. Tyto vazby můžou být například takové, že jeden synset značí součást, ze které je tvořen další synset. Jeden synset může spojovat

další synset tak, že jeden synset je zobecněná verze dalšího synsetu. Wordnet je tedy veliká síť navzájem propojených slov [14].

Kapitola 3

Návrh

Tato kapitola se zabývá návrhem hry. Podle mého následujícího návrhu, obsahujícího mé očekávání od dané hry, se rozhodnu, na jakou platformu tuto hru vytvořím a jaké konkrétní technologie použiji na její implementaci.

3.1 Návrh hry

Hráči budou mít určitý počet bodů a level. Tento level se jim bude zvyšovat podle toho, jak se jim bude dařit hru hrát. Rostoucí level tak bude sloužit jako motivace pro další hraní a anotování. Hra není realtime, proto nebude třeba řešit žádný složitý síťový protokol. Tahový způsob hry také odlehčí práci vykonávanou serverem, protože není třeba vyměňovat si pakety mezi klientem a serverem několikrát za sekundu, jak je v real-time hrách běžné. Očekávaná herní grafika je velmi jednoduchá. Stačí, aby hráči viděli anotované obrázky či videa.

3.1.1 Hra pro získávání anotací k obrázkům pro dva hráče

Cílem této hry je získávat popis věcí, které se nacházejí na obrázku. Tato hra se hraje ve dvou hráčích. Oba hráči dostanou stejný obrázek. Potom budou muset oba dva v určeném časovém limitu napsat věci, které se na obrázku nacházejí. Hráči by měli uvažovat stejně a snažit se napsat věc, kterou napíše i druhý hráč. Cílem hry totiž je, aby se oba dva hráči shodli na popisu obrázků.

Hráči nejsou za špatné typy nijak penalizováni, to je motivuje k tomu, aby napsali co nejvíc objektů. Kladné body dostanou ale pouze za věci shodující se s druhým hráčem. Předpokládá se, že když dva hráči napíšou stejnou věc, tak se daný objekt na obrázku skutečně vyskytuje a lze ho poté uložit do databáze jako anotaci.

Každý obrázek bude použitý vícekrát různými dvojicemi hráčů, čím více dvojic se shodne, tím lze považovat anotaci za správnější. Hráči by měli vybírat konkrétní slova z corpusu Wordnet, to poslouží jednak k větší kvalitě anotací a jednak by tento systém měl zabraňovat situaci, kdy hráči napíšou stejné slovo, akorát v jiném pádě, a systém tak nenajde shodu. Při návrhu této hry jsem se hodně inspiroval ESP game.

3.1.2 Mód pro jednoho hráče

Vytvořím mód hry pro získávání anotací k obrázkům, která bude ale pouze pro jednoho hráče. Tato hra bude důležitá především při rozjezdu hry, kdy nebude dostatek hráčů a bude

tak obtížné, aby někdo našel druhého hráče, se kterým by si tuto hru mohl zahrát. To umožní novým uživatelům si hru hned vyzkoušet a v případě, že se jim hra zalíbí, budou mít motivaci, aby si našli nějakého protihráče, se kterým by hru hráli.

Tento mód bude probíhat obdobně jako hra dvou hráčů, pouze s tím rozdílem, že objekty, které hráč napíše, se budou porovnávány s objekty, na kterých se shodli hráči v předešlých hrách pro dva. Čím více hráčů se shodlo na tom, že daný objekt na obrázku skutečně je, tím více za něj dostane bodů. Celkový počet bodů za tento typ hry, bude menší, než za hru ve dvou hráčích, aby byli hráči více motivováni hrát se soupeřem.

Nevýhodou tohoto módu hry je to, že tímto způsobem nelze získávat nové anotace k obrázkům. Bude moci pouze zvyšovat kvalitu již existujících anotací tím, že se zvýší výskyt již zadaných slov nacházejících se na daném obrázku.

3.1.3 Rozšířené získávání anotací

Vytvořil jsem návrh hry, která je velice podobná hře předešlé. V tomto případě ale hráči nebudou zadávat pouze objekty, které vidí, nýbrž budou vytvářet anotace k obrázkům tím, že budou popisovat určitou činnost, která se na obrázku děje. Hráči budou zadávat nejen podstatná jména, ale také slovesa, značící činnost, kterou objekty na obrázku vykonávají.

Chtěné anotace k obrázkům mohou například vypadat takto: Skákající žába, plavající hroch či běžící jelen. Nejvíce bodů hráči získají, pokud se shodne sloveso i podstatné jméno. Hráči budou nicméně odměněni také tehdy, pokud se shodne pouze sloveso nebo pouze podstatné jméno.

Pro tuto hru, bude také existovat obdobný mód pro jednoho hráče.

3.1.4 Hra pro získávání anotací k videům

Tato hra bude fungovat tak, že dva hráči budou hrát kooperativně. Každý hráč dostane na začátku hry od aplikace několik slov značících nějakou akci. Z těchto slov si každý hráč vybere takové slovo, které mu bude nejvíce vyhovovat. Každý hráč bude mít jiné slovo a nebude tak vědět, jaké slovo dostal jeho spoluhráč.

Úkolem obou hráčů po té bude nalézt takové krátké video, které by nejlépe vyjadřovalo toto sloveso. Videa se budou vyhledávat na serveru Youtube. Když hráči naleznou vhodné video, tak potom zkopírují URL videa a vloží ho do hry.

U videa hráči nastaví danou sekvenci, tedy v jakém čase začíná daná činnost. Je stanoveno omezení, že tato činnost může být dlouhá pouze 7 sekund, proto není třeba nastavovat konec sekvence. Tato doba by měla být dostatečně dlouhá na to, aby šlo poznat, o jakou činnost se na videu jedná.

Hra zpracuje tento odkaz a pouští druhému hráči dané video. Tento hráč musí podle videa uhádnout, o jaké sloveso jde. Bude mít více možností, aby mohl napsat také synonyma dané činnosti.

Pokud sloveso uhádne, pak oba dva hráči dostanou body. Touto hrou tak bude možné získávat videa a jejich anotace. Tato hra není složitá na implementování a výhodou tohoto konceptu je také fakt, že není potřeba získávat neanotovaná videa, protože hráči tato videa získají sami. Jediné, co je potřeba je mít nějakou databázi použitelných sloves. Pro generování sloves využiji slovník Wordnet.

3.1.5 Hra pro získávání bounding boxů

Bounding box je vymezení části obrázku. V případě anotování obrázku, se používají bounding boxy, aby se vymezilo, kde se v rámci obrázku nachází nějaký objekt. Vytvořím takovou hru, která bude mít za úkol takovéto bounding boxy k obrázkům a objektům získávat. Hra se bude hrát ve dvou hráčích, což poslouží jako kontrolní mechanismus, že bounding boxy jsou na správném místě. Hráči dostanou obrázek a jeden objekt, který by se měl na obrázku nacházet. Poté bude jejich úkolem tento objekt na obrázku vyznačit. V případě že bude těchto objektů na obrázku více, hráči budou muset umístit více bounding boxů.

Poté co oba dva hráči vyznačí objekt na obrázku, systém vyhodnotí, zda jsou bounding boxy na stejném místě. Samozřejmě se bude počítat s určitou tolerancí, protože je velmi nepravděpodobné, že by hráči umístili bounding box na pixel přesně.

3.2 Platforma

Dalším krokem k vytvoření návrhu pro crowdsourcing hry bylo zvolení platformy, na které bude hra fungovat. Protože je cílem této práce vytvořit hru, která bude mít co nejvíce hráčů, bylo potřeba zvolit takovou platformu, která je dobře přístupná pro všechny potenciální uživatele. Instalace hry by tedy měla být jednoduchá, v optimálním případě žádná. Platforma by také měla být rozšířená a populární mezi uživateli. K možnostem, mezi kterými jsem se rozhodoval, jestli budou vhodné pro tuto hru, patřilo vytvoření hry jako mobilní aplikace nebo vytvoření hry jako webové aplikace. Obě dvě tyto platformy mi přišly pro tuto hru velice vhodné, protože oba tyto způsoby jsou v dnešní době velmi oblíbené, tudíž by zde neměly být žádné hranice, které by bránily novým potenciálním hráčům si tuto hru zahrát. Vytváření aplikací pro obě dvě platformy je také velice pohodlné, protože existuje spousta knihoven, nástrojů a frameworků, které zjednodušují vývoj pro tyto platformy.

3.2.1 Mobilní aplikace

Mobilní aplikace je dobrou volbou pro crowd-sourcingovou hru. V dnešní době doprovází mobilní aplikace obrovský rozmach, při kterém se jejich počet razantně zvyšuje a mají stále rostoucí počet uživatelů. Výhodou je rovněž to, že jsou snadno přístupné na internetu a jejich instalace je velmi rychlá. Obzvlášť hry jsou velice populární mezi mobilními aplikacemi. Přesto však mají mobilní aplikace určité nevýhody. Je třeba je vytvářet pro specifický operační systém, tudíž by daná hra byla dostupná pouze pro majitele telefonů s určitým operačním systémem, což je velkou nevýhodou. Další nevýhodou je, že psání na telefonu je z uživatelského hlediska velice nepohodlné. Jelikož tato hra bude používat textové vstupy, mohlo by psaní na telefonu odradit uživatele od hraní této hry.

3.2.2 Webová aplikace

Webové aplikace mají velkou výhodu v přístupnosti k potencionálnímu uživateli. Fungují ve všech operačních systémech stejně a rozdíly mezi internetovými prohlížeči nejsou zásadně omezující. Další výhodou je i to, že webové aplikace lze spustit jak na mobilech, tak i na stolních počítačích. Nevýhodou webových aplikací je to, že vývoj složitějších her není tak snadný. Navíc běh webových aplikací ve webovém prohlížeči je méně efektivní než běh desktopové aplikace.

3.2.3 Výběr platformy

Z těchto dvou platforem jsem dal nakonec přednost vytváření hry coby webové aplikace. Rozhodl jsem se tak proto, že webové aplikace jsou totiž mnohem více přístupné všem uživatelům. Velkou výhodu spatřuji také v tom, že tyto aplikace lze spouštět v mobilních zařízeních, stejně tak jako v počítačích. Proto může webová aplikace využít všech uživatelů, kteří by byli hlavní cílovou skupinou v případě volby vývoje mobilní aplikace. Webový prohlížeč v dostatečné míře umožňuje, aby hráči viděli anotované obrázky nebo videa. Vytváření textových popisů také není nijak omezeno webovou aplikací. Proto je vyvíjení crowd-sourcingové hry jako webové aplikace nejvhodnější volbou.

3.3 Programovací prostředí

Vývoj webové aplikace lze rozdělit do dvou fází. A to na vývoj frontendu a backendu. Do frontendu spadá vytvoření aplikace, která bude spuštěna u uživatele v internetovém prohlížeči. Do backendu spadá vývoj serveru a tedy ta část aplikace, která bude uschovávat všechny získané anotace a bude synchronizovat komunikaci mezi více uživateli.

3.3.1 Frontend

Pro vývoj frontendu neexistuje tolik alternativ jako pro vývoj backendu. Internetové prohlížeče nativně podporují jen několik technologií. U ostatních technologií, nepřímo podporovaných v internetových prohlížečích, je třeba vybrat takové, které jsou kompatibilní s většinou druhů zařízení a prohlížečů. Pro vytvoření frontendu jsem se rozhodl použít klasické webové technologie: HTML5, Javascript a CSS. Jednou z dalších alternativ, kterou by bylo možno použít pro vývoj webové aplikace, je Flash.

Flash

Flash je grafický vektorový program používající se pro tvorbu interaktivních her. Využití Flashe jsem shledal jako nevyhovující a zbytečné. Zbytečné z toho důvodu, že v HTML5 a Javascriptu lze vytvořit všechno potřebné pro plánovanou crowdsourcingovou hru a Flash nepřináší žádné nové možnosti. Nevyhovující z toho důvodu, že je potřeba, při použití Flashe, aby si uživatel nainstaloval přídavný plugin, potřebný k samotné hře. Další nevýhodou je také to, že Flash nefunguje ve všech zařízeních. Například u zařízení vyráběných firmou Apple se rozhodli technologii Flash nepodporovat vůbec, což představuje velkou množinu potenciálních uživatelů. Další nevýhodou je také pomalý načítací čas dané stránky a také špatná přístupnost textu pomocí vyhledávačů [20].

HTML 5

HTML je značkovací jazyk, který se používá v dokumentech pro webové stránky. HTML jazykem lze vyjádřit jak má webová stránka vypadat, jaké zde mají být objekty a jak mají být umístěny. Úkolem internetových prohlížečů je přečíst kód napsaný v jazyce HTML a stránku podle toho zobrazit. Základním kamenem HTML je element.

HTML má za sebou několikaletý vývoj, jeho nejnovější verze je HTML 5. HTML 5 přináší novinky jako vkládání videa či zvuku do stránky. Tohoto šlo ještě nedávno docílit pouze pomocí využití technologie Flash. Další užitečnou utilitou je element canvas. Tento element slouží jako plátno, do kterého lze vykreslovat nejrůznější věci. Lze do něj vykreslovat

přímky, obrázky, kružinice a další. Tento element proto využiji pro hru na vykreslování bounding boxů [23].

Javascript

Javascript je jednoduchý interpretovaný programovací jazyk, který je objektově orientovaný. Všeobecné jádro tohoto jazyka je naimplementováno ve většině webových prohlížečích a je určeno pro programování webu s přidáním objektů reprezentujících okno prohlížeče a jeho obsah. Klientská verze Javascriptu umožňuje vkládat spustitelný obsah do webových stránek - to znamená, že díky Javascriptu nemusí být webové stránky pouze statické HTML. Javascript svou syntaxí připomíná programovací jazyk C++, v tom ale jeho podobnost končí. Javascript je prototypově založený, neexistují zde žádné třídy. V tomto jazyce není třeba definovat typy proměnných, je totiž netypový [15].

CSS 3

CSS neboli kaskádové styly je jazyk používaný při vytváření webových stránek. CSS bylo vytvořeno, aby se rozdělila prezentace stránky do jiného dokumentu, než je struktura stránky. Toto rozdělení umožňuje jednodušší zpracování dokumentů popisujících strukturu webové stránky (HTML). Výhodou také je zjednodušení vytváření stránek, protože CSS dokument je ve většině případů v jednom souboru a definuje vzhled pro všechny ostatní HTML dokumenty. Nemusí se tak ve všech ostatních HTML dokumentech opakovat stejné značky pro stejný vzhled stránky [22]. V současnosti nejnovější verzí CSS je CSS 3. CSS 3 umožňuje vytvářet moderní a čistý design pro nynější webové stránky, jednou z novinek je například vytváření animovaných prvků. Animace bez CSS 3 mohly být vytvářeny pouze pomocí Javascriptu nebo Flashe.

Framework pro CSS

Abych si zjednodušil vývoj webové aplikace, tak jsem se rozhodl, že využiji nějaký framework, který řeší vzhled stránky. Díky tomu nebudu muset sám tvořit úplně celý design, což je velice zdlouhavá a náročná práce. Pro svou aplikaci jsem se rozhodl, že použiju framework Bootstrap. Tento framework je ze všech podobných frameworků nejpopulárnější.

BootStrap

Bootstrap je velice užitečný frontend framework, který je zcela zdarma. Tvůrci tohoto frameworku jsou vývojáři z Twitteru. Bootstrap se skládá ze sady nástrojů určených pro vývoj webových stránek. Obsahuje HTML a CSS designové šablony a také Javascriptové skripty. Bootstrap definuje vzhled tlačítek, formulářů a navigačních panelů. Výhodou Bootstrapu je konzistence uživatelského prostředí.

Další velkou výhodou Bootstrapu je jeho zaměření na responzivnost. Responzivnost znamená, že webová aplikace mění rozpoložení objektů vzhledem k šířce okna, ve kterém jsou zobrazeny. Tato vlastnost je obzvlášť užitečná pro mobilní aplikace, kdy má uživatel mnohem užší obrazovku než na počítači a tak je uživatelské rozhraní, kde je více sloupců, pro něj nevhodné. Pro tento framework existuje několik různých variací. Základ těchto variací je stejný, liší se ale barevnou kombinací a lehce pozměněnými designovými prvky. Ne všechny tyto variace jsou však zdarma, nemusí totiž pocházet od oficiálních tvůrců Bootstrapu [2].

JQuery

Základní bootstrap bez rozšíření požaduje, aby aplikace obsahovala knihovnu JQuery, jinak javascript pro komponenty bootstrapu nebudou fungovat. Javascript je v komponentách potřebný pro vysouvací menu, přehazování stránek v carouselu, rolování elementů, vyskakování modalů a dalších interaktivních efektů, které uživatelům zjednoduší práci s aplikací.

JQuery je javascriptová knihovna, která zjednodušuje práci v javascriptu. Velkou výhodou této knihovny je její kompatibilita s většinou prohlížečů, proto se vývojáři nemusí tolik soustředit, zda jejich skripty v javascriptu fungují všude, ale mohou se spolehnout na knihovnu JQuery, která to zařídí za ně. Z tohoto důvodu této knihovny využili i vývojáři Bootstrapu. JQuery také obsahuje mnoho funkcí pro práci s elementy jako jejich skrývání, pohyb po obrazovce, postupné mizení, drag & drop a další [10].

Angular-UI

Abych mohl využít všech komponent, které mi Bootstrap nabízí a zkombinoval je s frameworkem Angular, použiji ve webové aplikaci knihovnu Angular-UI. Výhodou je to, že nebudu muset kombinovat kódy pro JQuery s kódem pro Angular. Všechny komponenty totiž spolupracují s Angular framework a jsou s ním plně kompatibilní. Kód tak bude čistější a plně podle filosofie Angular frameworku. Díky použití této knihovny bude potřeba knihovny JQuery úplně minimální [1].

3.3.2 Javascript Framework

Pro vytvoření frontendu aplikace jsem se rozhodl použít javascriptový framework. Použití javascriptových frameworků totiž přináší efektivnější vývoj klientské aplikace. Při vývoji tak není třeba zabývat se detaily, které nejsou pro aplikaci důležité, a umožňují zaměřit se pouze na důležité části. Z dostupných frameworků jsem si vybral framework AngularyJS. Díky tomuto frameworku je možné vytvářet single page webové aplikace. Single page webové aplikace jsou vhodnější pro webové hry, protože uživatel nemusí při každé akci čekat na načtení celé stránky znova. Aplikace je díky tomu velice plynulá.

3.3.3 AngularJS

AngularJS je Model View Controller framework zaměřený na klientskou část webových aplikací a napsaný v JavaScriptu. Běží ve webovém prohlížeči a velice dobře usnadňuje vývojářům vytvářet moderní, single-page webové aplikace. Tento framework má všeobecné využití. Silnou stránkou AngularJS je inovativní šablonovací systém, jednoduchý vývoj a velmi pevná inženýrské praxe. Na šablonovacím systému stojí za povšimnutí:

- Šablony jsou napsány v jazyce HTML
- Šablony nepotřebují DOM refresh, AngularJS se sám postará o načítání HTML stránek, kdy jsou potřeba. Pro načítání HTML šablon se využívá Ajax
- V šablonovacím systému lze vytvořit vlastní komponenty. Je tak možné vytvořit pro prohlížeč nové tagy s obsahem vytvořeným vývojářem.

Šablonovací systém je sice nejsilnější stránkou, ale ne tou jedinou. AngularJS není pouze šablonovací systém, ale má i ostatní utility a užitečné služby pro vývojáře. AngularJS využívá dependency injection, což pomáhá k vytvoření opětovně použitelných částí kódů, protože jednotlivé prvky webové aplikace lze jednoduše zapojit do jiného celku [21].

Služby

Služba je objekt v javascriptovém programu, který se zde vyskytuje pouze jednou. Registrování objektů jako služby tak umožňuje definovat důležité objekty pouze jednou a využít je kdekoliv v rámci programu. Služby jsou tedy napsány podle návrhového vzoru jedináčka. Služby si může vytvářet vývojář sám a nebo použít ty, které jsou ve frameworku Angular již vytvořeny [21]. Zde je několik užitečných defaultních služeb, které jsou velice užitečné při vytváření aplikace:

- `$http` - služba umožňující posílání HTTP request
- `$timeout` – služba, používaná pro plánování spouštění funkcí až po určitém časovém zpoždění
- `$scope` – služba, která umožňuje controlleru komunikovat s pohledem
- `$location` – služba, která slouží na přesměrovávání stránek
- `$log` – služba na vypisování výpisů do konzole

Direktivy

Direktivy umožňují rozšiřovat jazyk HTML o vlastní elementy nebo atributy. Direktivy fungují v Angular takovým způsobem, že pokud Angular překladač při zpracování HTML kódu narazí na známou direktivu, tak ji nahradí patřičným HTML kódem. V Angularu je spousta předdefinovaných direktiv, které jsou velmi užitečné a dají se využít ve většině aplikací. Zde jsou základní používané direktivy:

- `ng-hide` – direktiva atributu, která podle pravdivostní hodnoty skryje daný element
- `ng-show` – direktiva atributu, která podle pravdivostní hodnoty zobrazí element
- `ng-app` – direktiva, která naváže modul do aplikace
- `ng-repeat` – direktiva, která umožňuje vložit hodnoty ze seznamu
- `input` – direktiva, která se tváří jako klasický element tag, ale přepíše ho tak, aby byl tento input jednoduše napojen na scope a mohl být použit i při automatické validaci formulářů

Největší výhodou direktiv je to, že každý vývojář si může vytvářet své vlastní. Direktivy se používají k vytvoření složitých komponentů. Komponenty umožňují skrýt komplexní DOM strukturu, CSS a jejich chování. Komponenty by se měly vytvářet tak, aby byly využitelné i v jiných místech aplikace.

3.3.4 Backend

Vybírání správné technologie pro programování serverové části bylo mnohem složitější. Existuje zde totiž více použitelných programovacích jazyků než pouze Javascript. Pro každý jazyk také existuje i řada další webových frameworků. Při výběru programovacího jazyka jsem se rozhodoval mezi těmito třemi kandidáty [24].

- PHP

- Python
- Ruby

Tyto jazyky mají mnoho společného. Všechny tři jazyky jsou objektově orientované, jsou dynamicky typované a jsou interpretované.

PHP

Nejsilnější stránka programovacího jazyka PHP tkví v obrovské komunitě. PHP je totiž nejvíce používaný jazyk pro vývoj webu. Nevýhodou ale je, že tento jazyk není moc striktní, je mnoho způsobů jak dělat některé věci, což rozhodně nepomáhá čistotě kódu. Unikátní vlastností PHP je přímé vkládání zdrojového kódu do HTML souborů. Tato vlastnost je výhodná při vyvíjení malých stránek či aplikací. Při vyvíjení větších aplikací se tato vlastnost prakticky nepoužívá. Syntaxe jazyka PHP je podobná jazyku C, což je výhodou pro programátory, kteří chtějí začít s PHP a mají zkušenosti s programovacím jazykem C. PHP má slabý typový systém.

Ruby

Ruby je čistě objektivní jazyk. Veškeré datové typy jsou zde objekty. Tento jazyk je silně inspirován jazykem SmallTalk. PHP a Python nejsou čistě objektové jazyky, existuje zde několik primitivních datových typů. U Ruby tomu tak není, zde je objektem úplně všechno. Ruby je také velmi dynamický jazyk, v běhu programu jdou změnit metody všech objektů. Kombinace těchto přístupů však nemusí být vždy výhodná, primitivní objekty jdou přetypovat kdekoliv v programu a v případě importování cizích knihoven, se potom tyto primitivní objekty mohou chovat jinak, než by programátor očekával.

Python

Python je jazykem, který se nesoustředí přímo na vyvíjení webových aplikací. Přesto ale dnes už existují spolehlivé prostředky, jak tento jazyk použít i v tomto směru, což umožňuje WSGI: „Web Server Gateway Interface“. Python má velkou standardní knihovnu, dobrou a kvalitní dokumentaci. Pro Python také existuje velké množství externích modulů v dobré kvalitě. Python se hodně soustředí na čitelnost a znovupoužitelnost kódu (PEP-8). Toho dosahuje svou jedinečnou syntaxí: Bloky kódu se vytvářejí odsazením textů. Proto je kód v Pythonu vždy přehledný. Python také vyniká malým počtem klíčových slov, což snižuje složitost použití jazyka. Python má silný typový systém.

3.3.5 Srovnání programovacích jazyků pro backend

Dle [24] byly tyto jazyky srovnány v několika disciplínách. Těmito disciplínami byly: čitelnost, výkon, použitelnost, bezpečnost a popularita. Jako nejčitelnější jazyk zvítězil Python díky nucenému odsazování řádků. Na prvním místě z hlediska bezpečnosti zůstali společně Python a Ruby, protože PHP má několik bezpečnostních chyb. V popularitě zvítězilo PHP, protože komunita PHP je nesrovnatelně větší než u ostatních jazyků. Na druhém místě v popularitě skončil Python a na třetím Ruby. Výkonově skončily všechny tři jazyky stejně, při provádění kódu se stejnou funkcí na stejném hardwaru se kód provedl v přibližně stejném čase.

Největší použitelnost vyhrál jazyk Ruby, díky možnosti dynamicky změnit metody za běhu programu. Všechny tyto faktory jsem vzal v potaz při vybírání nejvhodnějšího jazyka. Ve výsledku jsem si vybral jazyk Python. Python se celkově umístil ve všech disciplínách nejlépe. Ve všech disciplínách, kromě popularity, předčil jazyk PHP. Ve srovnání s Ruby zvítězil v popularitě a čitelnosti a prohrál pouze v použitelnosti. Python jsem si také vybral z toho důvodu, že PHP mi nevyhovuje kvůli bezpečnostním chybám a nedokonalému designu programovacího jazyka. Od Ruby mě odrazuje přílišná dynamičnost, která může znesnadnit programování.

3.4 Výběr Python Framework

Při plánování Python frameworku, který využiji pro svou práci, jsem uvažoval nejvíce nad frameworky Django a Flask. Dále jsem uvažoval i o použití frameworku Twisted, který je určen pro multiplayerové online hry. Frameworky Django a Flask jsou multifunkční frameworky se širokou funkcionalitou. Jsou v nich vytvořeny základní funkce, které se využívají prakticky ve všech webových aplikacích. Těmito funkcemi jsou např. napojení k databázi a práce s nimi, vytvoření uživatelů a jejich autentizace, užitečné funkce pro práci se session a cookies, základní bezpečnostní ochrana.

Kombinace frameworku Djanga a Flasku by byla nevhodná, protože se jedná o frameworky se stejným účelem. Twisted framework by ale šel zkombinovat s jedním z těchto frameworků, protože se stará čistě o komunikaci po síti.

3.4.1 Flask

Flask je malý ale mocný microframework pro Python. Filosofie Flasku je taková, že se udržuje co nejosekanější jádro a pro jakoukoliv další funkcionalitu lze využít pluginy. Výhoda Flasku je v jeho jednoduchosti a nekomplexnosti [26].

3.4.2 Twisted

Twisted je framework čistě napsaný v Pythonu sloužící pro programování sítových služeb a aplikací. Hlavním konceptem tohoto frameworku je vytvoření neblokujícího asynchronního serveru. Přestože má i Python několik funkcí pro asynchronní komunikaci se serverem, tento framework je přenáší do vyšší úrovně. Twisted je navržen tak, aby podporoval klientskou i serverovou část a běžel na široké škále operačních systémů a platforem. Twisted poskytuje API pro TCP, SSL, UDP a Unixové sockety a další možné způsoby komunikace. Tento framework je vhodný pro multiplayerové online hry [18].

3.4.3 Django

Django je vysoko-úrovňový framework, který umožňuje vytvářet webové aplikace s relativně nízkým počtem řádků zdrojového kódu. Django je jednoduché, flexibilní a s jeho pomocí lze navrhovat řešení s malou režií. Django je postavené na objektově orientovaném jazyce Python. Umožňuje tak uživatelům vytvářet aplikace, které dokážou řešit mnoho druhů problémů. V Djangu je důležitá znovupoužitelnost všech komponent, řídí se principem DRY neboli "don't repeat yourself". Django je bezplatný open-source framework [16].

3.4.4 Závěr výběru frameworku

Nakonec jsem se po zvážení těchto možností rozhodl pro framework Django. Django má velkou výhodu v tom, že jeho vývoj trvá již několik let a proto je velice stabilní a vyzkoušený. Také se mi zamlouvalo to, že Django je mnohem více komplexní než Flask. Nemusím tedy hledat různé pluginy pro základní věci, ale mám veškerou funkcionalitu v jednom. Rozhodl jsem se také vynechat framework Twisted, protože jeho použití by bylo komplikovanější díky tomu, že by bylo třeba řešit dva servery. Jeden server se samotnou hrou, který by byl v Twisted a druhý pro Django, který by řešil věci mimo hru jakými jsou registrace, přihlašování a statistiky.

Jednodušší řešení je vytvořit celý backend aplikace ve frameworku Django. Nemusí se tak řešit ani předávání informací mezi těmito servery. Hra, kterou chci vytvořit, totiž nebude náročná na výměnu informací mezi uživatelem a serverem. Výměna těchto informací nemusí probíhat několikrát v rámci sekundy, hráči totiž mezi sebou interagují ve vysokých intervalech, pouze na začátku a na konci hry, kdy si vymění výsledky. Proto není třeba mít pro tuto komunikaci speciální framework.

3.5 NLTK

Rozhodl jsem se pracovat se sítí slov Wordnet, proto budu potřebovat API, které mi umožní s touto sítí slov pracovat. Rozhodl jsem se pro Nature Language Toolkit. NLTK je hlavní platforma pro vytváření Python programů pracujících s lexikálními daty. Tento soubor nástrojů opatřuje rozhraní pro jednoduchou práci s více než 50-ti korpusy a lexikálními zdroji. Jeden z těchto korpusů je také mnou chtěný Wordnet. Tento nástroj je multiplatformní, funguje v Mac OS X, Linuxu i ve Windows.

NLTK je bezplatný, open-source projekt, řízený komunitou. Existuje verze tohoto nástroje pro Python 3, proto ho můžu použít ve své aplikaci. Pomocí tohoto nástroje můžu vyhledávat synsety, získávat vztahy mezi synsety, dále můžu vyhledávat jednotlivá synonyma a jejich definice [9].

3.6 Databáze

Aplikace bude potřebovat spolehlivé úložiště dat. Spolehlivost úložiště dat je obzvlášť potřebná, aby se neztratily pracně získané anotace. Databáze bude také sloužit pro ukládání údajů o hrách a o uživatelích. Je potřeba, aby databáze byla rychlá, protože každý HTTP request bude potřebovat přístup do databáze. Také by databáze měla mít vyřešenu pokročilou konkurečnost dotazů. Předpokládám, že uživatelé budou upravovat záznam o stavu hry zároveň, takže v případě nedostatečného konkurentního systému, může dojít k chybám.

Rozhodoval jsem se mezi použitím databáze MySQL a PostgreSQL. Obě tyto databáze jsou relační a pro práci s nimi se používá jazyk SQL. Vybral jsem si tyto databáze, protože obě dvě jsou spolehlivé a mají dlouhou historii spokojených uživatelů. Také jejich spuštění není problematické, na většině serverů je jejich instalace samozřejmostí.

3.6.1 MySQL

MySQL je oproti PostgreSQL odlehčenější databázový systém. Sice již obsahuje pokročilé databázové funkce, přesto ale nedosahuje takových možností jako PostgreSQL. Výhodou takového odlehčeného přístupu je mírně vyšší rychlost. Mysql má několik typů tabulek.

Ty nejzákladnější typy nepodporují transakce a mají jen ty nejzákladnější funkce. Tímto způsobem se Mysql snaží být přívětivá i pro nezkušené vývojáře [4].

3.6.2 PostgreSQL

Postgresql je vyzrálý a pokročilý databázový systém, který plně podporuje pokročilé databázové funkce jako procedury, pohledy, triggery a další. Souběžnost přístupů je zde řešena v mnohem větší hloubce [5].

3.6.3 Závěr rozhodnutí

Rozhodování mezi těmito databázemi nebylo příliš důležité, obě dvě databáze jsou si sobě velice podobné a obě dvě by dobře posloužily k účelům mé hry.

Přesto jsem se rozhodl, pro databázi PostgreSQL, protože její systém pro souběžné dotazy, je pokročilejší než v MySQL. Předpokládám, že vzhledem k tomu, že mé hry budou pro více hráčů, kteří budou zároveň přistupovat do stejného záznamu, se bude pokročilý systém řešící souběžnost přístupů velmi hodit .

Návrh databáze

Vytvořil jsem návrh databáze, tento návrh lze vidět v B.1

Kapitola 4

Implementace

4.1 Implementace klienta

Klientská část byla vytvořena pomocí HTML5, CSS a Javascriptu. Javascript kód jsem implementoval za použití frameworku AngularJS. Pro jednodušší design jsem použil framework Bootstrap. Pro spojení frameworku Bootstrap s Angular jsem použil knihovnu Angular-UI.

V rámci Angularu byl vytvořen hlavní modul *rootModule*. Tento modul má závislosti na ostatní podpůrné moduly. Podpůrné moduly obsahují direktivy a služby, které jsem vytvořil. Tento modul je nastaven tak, aby se automaticky spustil s celou aplikací.

4.1.1 Direktivy

Vytvořil jsem Angular direktivy na opakovatelné problémy ve webové aplikaci. Vytvoření direktiv mi zjednodušilo práci a výrazně přispělo k čistotě zdrojového kódu.

MyLoading

Tato direktiva se skládá čistě z HTML šablony. Je to direktiva elementu. Na místě vložení tagu se poté automaticky vloží HTML šablona s animovaným objektem signalizujícím práci aplikace. Tato animace obsahuje defaultní direktivu *ng-hide*, s pomocí které se dá jednoduše nastavit, kdy se tato animace zobrazí uživateli. Zda má být animace vidět, se nastaví pomocí proměnné *hideLoading*.

MyAlert

Vložením této direktivy se vloží šablony pro aplikační hlášky. Vytváření hlášek se nastavuje pomocí 3 proměnných umístěných na *$scope*. Proměnnou *alertShow* se nastavuje, zda se má hláška zobrazit. Proměnnou *alertMessage* se nastaví text hlášky a proměnnou *alertType* typ hlášky. Jsou dva typy hlášek – chybové a oznamovací.

PasswordMatch

Toto je direktiva atributu, používá se vložením atributu *passwordMatch* do elementu. Element musí být *input*. Tento atribut má jeden parametr a to jméno elementu, který musí být stejný jako položka s tímto atributem. Tato direktiva po té sleduje veškeré změny na

elementu s daným jménem a porovnává to s hodnotou elementu, jehož je atributem. Pokud se obě dvě hodnoty neshodují, vyhodí se do formuláře validační error *unique*.

ExperienceBar

Tato direktiva slouží pro vygenerování HTML kódu, který zobrazuje uživateli jeho postup v úrovních. Nejdůležitější součástí této direktivy je progress bar. Progress bar je komponenta z Angular-UI. Pro direktivu jsou důležité 4 proměnné, které je třeba umístit na *$scope*. Tyto proměnné jsou oldPoints, newPoints, oldLevel, newLevel. Při nastavení těchto proměnných se spustí animace v progress baru, který z hodnoty *oldPoints* vyroste na hodnotu *newPoints*. Pokud uživatel získal nový level, tedy když se proměnné *oldLevel* a *newLevel* neshodujou, tak se provede taková animace, kdy se progress bar celý vyplní a zobrazí se hláška informující uživatele, že dosáhl nového levelu. Po chvilce progress začne znovu růst z nuly a dojede na počet bodů (*newPoints*), který má.

autoFillSync

Tato direktiva řeší problém Angularu s vlastností autofill nacházející se v moderních internetových prohlížečích. Tento problém je takový: Uživatel si uloží v prohlížeči své přihlašovací údaje a později když najede na danou stránku, tak prohlížeč za něj tyto údaje vyplní. V Angularu ale existuje taková chyba, že tyto údaje se vyplní, ale do Angular modelu se nenačtou. Proto jsem musel vytvořit tuto direktivu, která je umístěna ve formě atributu v políčkách formuláře pro přihlašování. Tato direktiva dá prohlížeči čas půl sekundy, aby mohl automaticky vyplnit údaje. Potom načte tyto údaje do Angular modelu. V této direktivě se ověřuje, zda za tuto půlku sekundy uživatel nestačil již něco vyplnit. Pokud ano, vezme se hodnota, kterou vyplnil uživatel.

autocompleteFor

Tato direktiva řeší druhou část problému spojeného s vlastností autofill. Angular model se neaktulizuje ani v případě, že uživatel si u jednoho políčka formuláře vybere z našeptávače hodnotu (např. uživatelské jméno) a prohlížeč mu automaticky doplní pasující hodnotu do políčka formuláře druhého (např. heslo).

Tato direktiva je také ve formě atributu, který se dává k druhému, automaticky vyplňovanému políčku formuláře. Zde je třeba do atributu nastavit hodnotu jména prvního políčka formuláře, jehož hodnota je uživateli našeptávána prohlížečem. Kód této direktivy poté pracuje tak, že pozoruje, jestli se změnila hodnota políčka formuláře, jehož jméno je nastavené v atributu. Pokud ano, dá prohlížeči půl sekundy, aby mohl automaticky vyplnit políčko formuláře, kde se atribut direktivy nachází. Poté vezme hodnotu tohoto formuláře a nastaví ji do Angular modelu.

4.1.2 Angular-UI

Kvůli frameworku Bootstrap jsem použil knihovnu Angular-UI, abych mohl využívat všechny komponenty bez JQuery a ony lépe spolupracovaly s frameworkem AngularJS. Tato knihovna se aktivuje v Angular Frameworku velice snadno, je zde vytvořen modul *angular-ui* s veškerou funkcionalitou. Tento modul jsem vložil do seznamu závislostí k hlavnímu modulu spouštěného aplikací. Na výběr jsem měl dvě možnosti, jak použít tuto knihovnu, a to buď

s externími šablonami, nebo s šablonami napsanými přímo v kódu. Já jsem se rozhodl, že použiji druhou možnost, protože jsem šablony komponent nepotřeboval měnit.

4.1.3 Služby

Pro aplikaci jsem vytvořil několik vlastních služeb, které se starají o její chod. Výhodou používání služeb je jejich schopnost uložit nějakou proměnnou v rámci aplikace a pak k ní kdykoliv přistoupit. Se službami se pracuje výhradně pomocí jejich funkcí. Každá služba má nějakou konkrétní oblast působnosti.

Game

Vytvořil jsem službu *Game* a umístil jsem ji do modulu *GameProvider*. Tato služba zajišťuje továrnu na vytvoření funkcí starajících se o průběh hry. Tato služba zajišťuje hru pro všechny typy her. Hlavní funkce této hry je ukládání dat, které budou potřebovat další controllery v průběhu hry.

Součástí této služby je funkce pro odstartování hry. Účelem této funkce je přesměrování stránky na správnou stránku hry. V rámci této funkce se zkontroluje typ hry a podle toho se nastaví lokace přesměrování. Každý typ hry má totiž své vlastní šablony a controllery.

Po odstartování hry z lobby se do této služby ukládají i základní data o hře. Tedy počet kol, typ hry, první obrázek či sloveso. Pro každý parametr, který tato služba ukládá, existuje i funkce, která ho vrátí. Takto mohou controllery přistupovat k potřebným proměnným v místech, kde je potřeba.

Při dohrání jednoho kola, se používá tato služba pro ukládání výsledků uživatelů, tedy jaké objekty uložili oni a jaké objekty uložili jejich spoluhráči. Také se zde ukládá, jaký bude další obrázek v dalším kole.

Podobně jako při startu hry, se zde nachází funkce přesměrování do stránky s tabulkou se skórem. Všechny typy hry totiž mají odlišnou šablonu pro závěrečné skóre. Služba zjišťuje kam přesměrovat hru, díky tomu, že má nastavenou proměnnou s typem hry.

Po dokončení hry, zde existuje funkce, která vrátí pole s daty o všech hraných kol. Mezi tyto informace patří počet získaných bodů v kole, napsané objekty a napsané objekty oponentem. Data o všech hraných kolech, se ukazují v závěrečných statistikách.

GamesChannel

Naimplementoval jsem službu *GamesChannel*. Tato služba je součástí modelu *GamesChannelProvider*. Tato služba zajišťuje posílání zpráv pro stránku se seznamem her. Pokud se totiž stane něco ve hře, tak se hra zruší a přesměruje se právě zde. Tato služba potom slouží k tomu, aby uživatel dostal informaci, proč se hra zrušila. Služba poskytuje dvě hlavní funkce, jednu pro nastavení zprávy a druhou pro vybrání zprávy. Při vybrání zprávy se zpráva smaže, aby se zpráva neopakovala při opětovném příchodu na stránku s hrami.

LoginChannel

Vytvořil jsem model *LoginChannelProvider*, jehož součástí je služba *LoginChannel*. Tato služba slouží pro komunikaci s controllerem pro stránku k přihlašování. Pomocí této služby lze nastavit upozornění, které se na stránce objeví po přesměrování. Tato služba také nabízí funkci na uložení lokace, kam se má stránka po přihlášení přesměrovat. Při přihlášení se

zavolá jiná funkce, která vezme uloženou lokaci a zároveň ji smaže, aby při příštím přihlášení se zde stránka nepřesměrovávala znovu.

User

Vytvořil jsem svůj vlastní model *UserProvider*, jehož součástí je služba *User*. Tato služba je továrnou na vytvoření sady funkcí pro práci s přihlášeným uživatelem na straně klienta. Tato služba se využívá všude, kde se přihlašuje nebo odhlašuje uživatel. Také se používá tehdy, když je potřeba zjistit, zdali je uživatel přihlášen, a v případě existujícího přihlášeného uživatele, získat jeho jméno. Po úspěšném přihlášení se v této službě nastaví, že byl uživatel přihlášen a jeho jméno zapíše do privátních proměnných této služby. Existuje zde také funkce pro odhlášení, která jméno vynuluje a nastaví status do stavu nepřihlášen.

Title

Služba titul je součástí modelu *TitleProvider*. Tato služba slouží k změně titulky stránky. Umožňuje tak vysílat upozornění uživatelům, kteří nemají soustředěnou stránku, že se ve hře stalo něco nového. Toho se využívá zejména v lobby. Tato služba má dvě funkce, první funkce nastavuje zprávu, která bude blikat v titulku. Druhá funkce ukončuje blikání titulku a vrací ho do normálního stavu.

4.1.4 Konfigurace

Bylo potřeba nakonfigurovat několik modulů dostupných v základní verzi AngularJS, aby pracovaly podle mých požadavků. Moduly jsem konfiguroval přes jejich rozhraní. Konfigurace se provádí ihned po spuštění aplikace.

NgRoute

NgRoute je modul, který se stará o směrování stránek ve frameworku AngularJS. Tento modul je třeba nakonfigurovat tak, aby se při různých URL načítala správná stránka. Module lze konfigurovat přes jeho interface *$routeProvider*. V konfiguraci se k dané URL nastavuje, jaký se má použít controller a jaká se má použít šablona. U některých stránek zde bylo potřeba nastavit i proměnné, které se posílají přes URL.

Interpolate provider

Služba *$interpolateProvider* umožní konfigurovat kompiler Angularu. To je užitečné v případě kombinace frameworku Django a AngularJS. Oba dva tyto frameworky mají totiž šablonovací systém, kam se vkládají proměnné takovým způsobem, že se uzavřou do dvojitých složených závorek ({{ *proměnná* }}). Aby se tyto šablonovací systémy nekryly, využil jsem této služby, abych pozměnil v kompileru způsob, jakým v šablonách vyhledává proměnné. Nakonfiguroval jsem tuto službu tak, že se v mé aplikaci vkládají proměnné v šablonách následujícím způsobem: {$ *proměnná* $}

HTTP interceptor

Ve frameworku AngularJS existuje služba se jménem *$httpProvider*, kterou jsem použil i ve své aplikaci. V rámci konfigurace této služby lze vytvořit interceptor všech HTTP response přicházejících ze serveru. Takto jsem mohl nastavit přednastavené reakce klienta na HTTP

response s určitými stavovými kódy. Tuto službu jsem nastavil tak, aby odchytávala tyto kódy:

- 401 – Unauthorized
- 403 – Forbidden
- 500 – Internal Server Error

V případě že přijde HTTP response s kódem 403, znamená to, že CSRF ochrana odhalila u poslaného HTTP requestu vadný CSRF kód. Tento případ může nastat při skutečném útoku, ale taky může dojít k nějaké chybě u klienta. Proto, pokud se odchytne takový HTTP response, webová aplikace se přesměruje na stránku, která informuje uživatele o tomto stavu a o tom, že musí znovu zapnout aplikaci, aby mohl pokračovat.

Pokud se vyskytne na serveru nějaká chyba v kódu, pošle zpátky klientovi HTTP response s kódem 500. Nastavil jsem tedy odchytávání HTTP response s tímto kódem, abych informoval klienta o tom, že se stala nějaká nečekaná chyba na serveru a z toho důvodu aplikace nefunguje správně.

Nakonfiguroval jsem službu *$httpProvider*, aby ve vytvořeném interceptoru měla přístup k službě *UserProvider*. Interceptor tak při odchytu HTTP response s kódem 401 může ve službě nastavit, že uživatel není přihlášen.

4.2 Implementace serveru

Pro implementaci serveru jsem využil frameworku Django. Server funguje na takovém principu, kdy odchytává všechny HTTP request, které jdou na jeho doménu. U všechn HTTP request zkoumá URL, na jakou jsou poslány. Podle cesty v URL, server rozhodne kterou třídu použije pro jeho zpracování.

4.2.1 Knihovny

Na serveru jsem implementoval určité třídy, které mi pomáhají zpracovat více než jeden typ HTTP requestů.

RequestParser

Tato třída slouží k tomu, aby se získaly z HTTP Requestu poslané informace. V HTTP requestu jsou informace poslány ve formátu JSON, proto se používá modul Pickle, aby se převedla data do datové struktury, s kterou se dá v Pythonu pracovat. Tato třída se používá u všech HTTP Requestu, které na server dojdou.

ResponseMaker

Tato třída umožňuje jednoduché vytváření odpovědí pro klienta. Tato třída vytváří nový HTTP Response. Třída kontroluje, zda při zpracování HTTP Requestu nedošlo na serveru k nějaké chybě, která by nastavila chybový kód. Chybu může například vyvolat špatný format dat zaslaných klientem. Pokud dojde k nějaké chybě, zkontroluje její kód.

Pokud je to kód signalizující, že nepřihlášený uživatel poslal žádost, aby server udělal něco, co mohou pouze přihlášení uživatelé, tak potom tato třída nastaví u nově vytvořeného HTTP Response stavový kód na 401. Pokud je to jiný typ chyby, stavový kód se nastaví

jako 401. V případě, že při zpracování nedošlo k žádné chybě, nastaví se stavový kód na 200. Tato třída se stará také o to, aby HTTP Requesty byly naplněny o jejich obsah.

GetUser

Tato třída se používá na částech serveru, kde mají přístup pouze přihlášení uživatelé. Tato třída se postará o to, aby zkontrolovala, zda byl uživatel přihlášen. Pokud je uživatel přihlášen, systém si ho načte do vnitřní proměnné, díky které můžou třídy zpracující HTTP request přistupovat k uživatelským datům. Pokud uživatel není přihlášen, tak vytvoří pro uživatelé anonymního uživatele, ke kterému ho přihlásí.

FindGame

Tato třída může být použita, po použití třídy *GetUser*. Úkolem této třídy je najít hru, ve které se uživatel nachází. Tato třída má parametry, značící status hry, kterou má najít. V případě, že nenajde žádnou hru spojenou s uživatelem a daným statusem hry, nastaví chybový kód. Chybový kód se nastaví, i pokud bude takových her nalezeno více. V opačném případě, si uloží záznam hry do vnitřní proměnné, pomocí které pak může třída zpracovávající HTTP request manipulovat s nalezenou hrou.

4.2.2 Middleware

Používání middlewaru je nedílnou součástí používání Djanga. Middleware lze chápat podobně jako pluginy. Middleware jde totiž jednoduše zapnout a vypnout podle situace. Lze také používat Middleware od jiných vývojářů než od oficiálních tvůrců Djanga. V tomto případě je samozřejmě třeba být obezřetný při jeho používání [3]. Já ve své aplikaci používám tento Middleware:

- SessionMiddleWare
- CommonMiddleware
- AuthenticationMiddleware
- XFrameOptionsMiddleware
- CsrfViewMiddleware

CommonMiddleware

CommonMiddleware má několik různých funkcí. Jednou z jeho funkcí je starost o URL.

Session MiddleWare

Django umožňuje plnou podporu pro anonymní sessions. Díky tomuto Middleware lze ukládat a získávat zpět libovolná data pro jednotlivé uživatele webové aplikace. Data jsou ukládána na serveru a excerpují tak zasílání a přijímání cookies. Cookies obsahují pouze session ID, nikoliv data samotná.

CsrfViewMiddleware

Tento Middleware se stará o ochranu proti Cross Site Request Forgery. Stará se o generování CSRF tokenů. A v případě zaslání HTTP Requestu kontroluje, zda obsahuje CSRF token a jestli je tato hodnota správná. V opačném případě se middleware postará o ignoraci takového requestu a vytvoření HTTP response se stavovým kódem 403.

XFrameOptionsMiddleware

Tento Middleware se stará o ochranu proti ClickJackingu.

AuthenticationMiddleware

Jedná se o Middleware, jehož práce spočívá v tom, že se stará o zjišťování, který uživatel poslal HTTP request. Po přihlášení uživatele se ukládá záznam o uživateli do session. Aby se uživatel identifikoval, server mu pošle HTTP response s cookies *sessionid*, který si uživatel uloží a použije v budoucí komunikaci. Poté lze pomocí tohoto Middlewaru na serveru jednoduše zjistit informace z HTTP requestu, zda ho posílá přihlášený uživatel a o jakého uživatele se jedná. Vývojář tak nemusí řešit detaily s přihlašováním a jeho bezpečností.

4.3 Komunikace serveru s klientem

Frontend komunikuje s backendem pomocí AJAXu. Klient pošle serveru HTTP request. HTTP request se odesílá na konkrétní URL. Server podle toho, na jakou URL je HTTP request poslán, žádost zpracuje. Při posílání na konkrétní URL server očekává, že HTTP request bude typu POST a v těle žádosti budou přídavná data. Zachování kontextu v komunikaci se serverem a frontendem se děje pomocí Session. Server si o každém uživateli vede záznam a toho uživatele identifikuje pomocí *sessionid* uložené v cookies. Na každou žádost server odpovídá klientovi pomocí HTTP response. Pokud je všechno v pořádku, pak má daný HTTP response stavový kód 200. Data posílaná serveru klientem i klientem serveru jsou ve formátu JSON.

4.3.1 Detekce odpojených hráčů ze hry

Během hry probíhá veškerá komunikace obou hráčů přes server. Se serverem oba dva hráči komunikují přes HTTP protokol. Tento protokol však není příliš vhodným protokolem pro síťovou hru. Nelze jednoduše zjistit, kdy se druhý hráč odpojil.

Jediným způsobem, jak určit, že hráč už není připojený ke hře, je nastavit, aby hráči serveru neustále posílali serveru zprávu o tom, že jsou stále připojení. A pokud server nedostane nějakou určitou dobu zprávu o hráči, tak ho začne považovat za odpojeného.

Tento způsob má úskalí v tom, že doba, kterou server musí dát hráči na posílání zpráv, by měla být dostatečně dlouhá. Poslat zprávu serveru může totiž trvat v přetíženém provozu i několik sekund, a když se nastaví doba příliš krátká, může server poté odpojit hráče, který je stále ve hře. Proto jsem tuto dobu nastavil na 40 sekund. Tato doba zase není ideální pro druhého hráče, který až po 40 sekundách zjistí, že hraje s odpojeným hráčem.

Proto jsem se snažil vytvořil mechanismy, které označí hráče za odpojeného dříve. Jedním z těchto mechanismů je odchytávání události o změně lokace stránky v controllerech,

starajících se o průběh hry. Pokud controller zjistí, že má být přesměrován a přitom přesměrování není součástí hry, tak odešle automaticky HTTP request na server s žádostí o odpojení.

Server, když přijme takový požadavek, si nalezne přihlášeného uživatele a hru, ve které se nachází. Poté ho z této hry odpojí. Pokud je tvůrcem hry nebo je hra už spuštěna, pak server hru zablokuje. Spuštěnou hru totiž nelze hrát pouze v jednom hráči. Díky tomuto mechanismu se druhý hráč může dozvědět velice rychle, že se jeho spoluhráč odpojil.

4.4 Bezpečnost

Bezpečnost je velice důležitým prvkem ve všech webových aplikacích. Hráči při registraci posílají serveru své osobní údaje, jako například emailovou adresu a heslo. Tyto údaje by neradi poskytli třetí straně. V případě, že by se hráči dověděli, že server není důvěryhodný, tak by je to mohlo odradit od další registrace a hraní her. Proto jsem musel řešit i základní bezpečnost uživatele. Protože jsem použil framework Django, tak jsem některé bezpečností problémy mohl řešit pouze okrajově, neboť Django tyto problémy vyřešilo za mne. Jedním z těchto problémů byla ochrana proti Cross Site Reference Forgery.

4.4.1 Cross Site Request Forgery

Cross Site Request Forgery patří mezi 20 nejvíce zneužívaných bezpečnostních chyb. Přesto se ochraně proti Cross Site Request Forgery věnuje málo pozornosti. Cross Site Request Forgery probíhá tak, že útočníkova stránka nabádá prohlížeč nic netušícího uživatele, aby poslal HTTP request na důvěrnou stránku, jako kdyby to byla běžná interakce uživatele s touto stránkou. Zneužívá tak připojení uživatele a stav cookies v uživatelově prohlížeči, aby se připojil jako uživatel. Po připojení může útočník vykonávat veškeré operace jako uživatel a zneužít to ve svůj prospěch. Proti Cross Site Request Forgery se dá bránit několika různými způsoby [8]. Na obrázku 4.1 lze vidět schéma CSRF útoku.

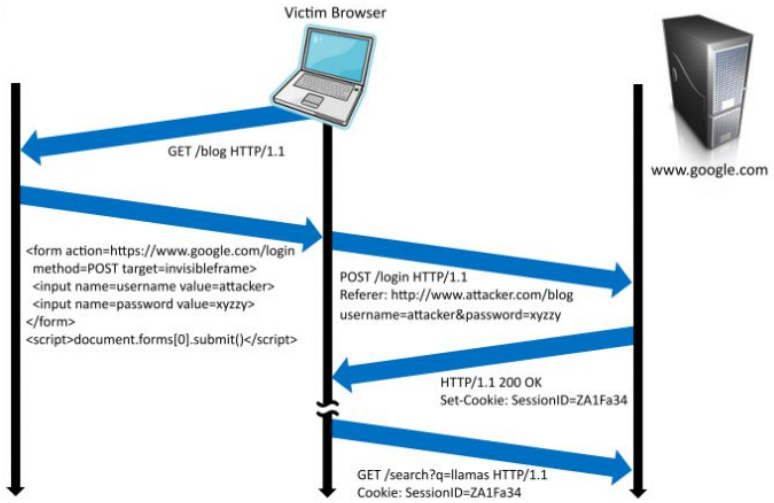
4.4.2 Custom HTTP Headers

Vlastní HTTP hlavičky mohou být použity jako ochrana pro CSRF, protože prohlížeče zabraňují posílání vlastních hlaviček z jedné webové adresy na jinou webovou adresu, ale stránkám vytvářet své vlastní HTTP hlavičky za použití XMLHttpRequest objektu ne. Při použití této CSRF ochrany musí stránka kontrolovat hlavičky HTTP requestů a zavrhovat všechny ty, které nemají správnou hlavičku.

4.4.3 Secret Validation Token

Jedním ze způsobů, jak se bránit proti CSRF útokům, je posílat dodatečné informace v každém HTTP požadavku, který může být použit k posouzení, zda požadavek přišel z autorizovaného zdroje. Tento "validační token"by mělo být velice složité uhádnout pro útočníka, který nemá přístup k uživatelovu účtu. Pokud HTTP requestu chybí validační token nebo nemá očekávanou hodnotu, server by měl tento HTTP request odmítnout.

Je několik různých druhů technik, rozhodujících jak generovat hodnotu validačního tokenu. Jednou z možností je generovat tento token náhodně. Server tuto hodnotu vygeneruje při první návštěvě uživatele a ten si ji uloží ve formě cookies.



Obrázek 4.1: Cross Site Reference Forgery schema, zdroj: [8]

4.4.4 Problém s Cross Site Request Forgery

Pro Django existuje speciální Cross Site Request Forgery Middleware, který se stará o ochranu proti CSRF útokům. Tento Middleware používá ochranu pomocí tajného validačního tokenu. Při prvním přístupu náhodně vygeneruje hodnotu, kterou použije jako validační token. Posílání tajného tokenu se v Djangu provádí tak, že se v šabloně nastaví speciálním tagem *{ % csrf_token % }* .

To při renderování šablony do HTTP response nastaví cookies s hodnotou tohoto tokenu. Poté je při posílání informací ve formulářích zpátky na server typické vytvořit skrytý input s hodnotou CSRF tokenu. Hodnota CSRF tokenu se dá získat a vložit do šablony pomocí speciální značky *$csrf_token* .

Ve své aplikaci ale využívám javascriptového frameworku angular, který má své vlastní šablony. Využívá Django šablonovacího systému pouze pro titulní stránku. Proto jsem nastavil *{ % csrf_token % }* v titulní stránce, aby se vygeneroval validační token. Problém nastal při posílání dat serveru, při přijetí přes CSRF ochranu. Tento problém jsem vyřešil tak, že při startu aplikace načtu validační token z cookies. Poté využiju služby *$http*, která má atribut *default.post* a do toho nastavím hodnotu validačního tokenu pod klíč *X-CSRFToken*. Takovéto nastavení způsobí, že veškeré HTTP requesty zasílané přes framework angular budou mít hlavičku *X-CSRFToken* se správnou hodnotou validačního tokenu.

Další problém nastává, když se uživatel přihlásí, poté se totiž hodnota validačního tokenu změní. Protože se při přihlášení neposílá žádná nová Django šablona s nastaveným validačním tokenem, musel jsem nastavit, aby se hodnota nového validačního tokenu odesílala v HTTP response reagující na požadavek k přihlášení. Když klientovi přijde HTTP response po úspěšném přihlášení, nastaví se, aby se, stejně jako předtím se všemi HTTP requesty, posílala nová hodnota validačního tokenu.

4.5 Nasazení na server

Výsledná aplikace běží na serveru knot08, který je pod správou Fakulty informačních technologií VUT Brno. Na tomto serveru běží operační systém UNIX. Používal jsem SSH pro přístup na vzdálený server, abych zde svou aplikaci zde správně nakonfiguroval a spustil.

4.5.1 Python

Nejdůležitějším požadavkem pro běh aplikace byla instalace Python interpretu. Pro aplikaci jsem použil Python verze 3.2. Pro spuštění aplikace by byl ale pouhý Python interpret nedostatečný. Potřeboval jsem přídavné balíčky pro práci s databází pro Django framework a pro vytvoření virtuálního prostředí. Pro správné nakonfigurování potřebných balíčků jsem použil nástroj Pip.

4.5.2 Pip

Pip je nástroj pro správu balíčků v Pythonu. Díky tomuto nástroji lze jednoduše přidávat a odebírat všechny balíčky. Nástroj získává zdrojové kódy požadovaných balíčků ze vzdálených repositářů. Ve většině případů dokáže Pip najít v repositářích všechny balíčky, které uživatel potřebuje. Jednoduchá instalace probíhá pomocí příkazu *pip install jméno_balíčku* Využil jsem Python balíčky a to:

- django
- psycopg
- virtualenv
- nltk

Virtualenv

Tenhle balíček bylo třeba nainstalovat jako první. Virtualenv je nástroj sloužící k vytvoření izolovaného prostředí. Protože jsem hru konfiguroval na serveru, kde běželo souběžně mnoho dalších Python aplikací, bylo třeba balíčky, které jsou potřebné pro aplikaci, nějakým způsobem izolovat, aby nekolidovaly s ostatními nainstalovanými balíčky. Tento nástroj tak umožní mít vždy specifickou verzi potřebnou pro správný běh aplikace. Práce s tímto nástrojem vypadá tak, že se pomocí příkazu *virtualenv* vygeneruje složka se soubory virtuálního prostředí a s nástroji pro práci s ním. V této složce se automaticky vytvoří binární soubor *activate*, jehož spuštěním se aktivuje virtuálním prostředí. Po této akci při použití nástroje se všechny instalované balíčky automaticky nainstalují do dané složky s virtuálním prostředím. Také při použití interpretu Python se použijí pouze balíčky virtuálního prostředí.

Django

Django balíček je balík, který je potřebný pro použití frameworku Django. Použil jsem v tuto dobu nejnovější verzi Djanga a to verzi 1.6.1. Bylo potřeba použít přídavné balíčky a to balíčky Django, Psycopg2.

Psycopg2

Psycopg2 je adaptér k databázi PostgreSQL pro programovací jazyk Python. Psycopg2 je malý balíček a jeho používání je rychlé a velice stabilní. Tento balíček je odlišný od ostatních databázových adapterů tím, že je určen pro těžké multi-vláknové aplikace, které vytvářejí a používají velký počet kurzorů najednou a také používají konkurentní dotazy. Psycopg2 také poskytuje asynchronní operace.

NLTK

NLTK je knihovna s rozhraním pro práci s několika slovníky. Součástí NLTK je i slovník Wordnet. Proto jsem musel nainstalovat tuto knihovnu, abych umožnil aplikaci přistupovat k funkcím pro práci se slovníkem Wordnet.

Kapitola 5

Stránky aplikace

Zde popíšu implementaci jednotlivých stránek, jak fungují z pohledu klienta i z pohledu serveru. U každé stránky napíšu, k čemu slouží a jaké problémy jsem musel řešit při jejich tvorbě.

5.1 Menu

Pro vytvoření menu jsem využil komponentu navigační listy z frameworku Bootstrap. Tato komponenta mi umožňuje vytvářet rolovací lišty v menu. Menu má odlišné položky podle toho, zda je uživatel přihlášen nebo ne. Pokud uživatel přihlášen není, má možnost si vybrat ze 4 položek:

- Home - úvodní stránka
- Statistiky - statistiky všech hráčů a jejich skóre
- Registrace - registrace nových uživatelů
- Login - přihlašovací formulář

Abych zjistil, jestli je uživatel přihlášen nebo ne, tak využívám mnou vytvořenou službu *User*, ze které načítám jméno uživatele a pravdivostní hodnotu, zda je přihlášen či nikoliv. Pokud je uživatel přihlášen, tak se mu v levé straně menu nabídnou tyto položky:

- Home - úvodní stránka
- Singleplayer - hra pro jednoho hráče
- Multiplayer - hra pro více hráčů

Uživateli se tak skryly položky pro registraci a přihlášení, protože tyto položky přihlášený uživatel nepotřebuje. Pravá strana menu po přihlášení nezůstane prázdná a objeví se zde dvě položky:

- Jméno přihlášeného uživatele - vede ke stránce s detaily uživatelského účtu
- Odhlášení - stránka pro odhlášení uživatele

O veškerou logiku v menu se stará controller *UserBarCtrl*. Zobrazování položek v menu není ale jeho jediným úkolem, tento controller se stará o komunikaci se službou *User*. Je to jediné místo v aplikaci, které přímo službu *User* informuje o tom, že se uživatel přihlásil nebo odhlásil. Pokud totiž dojde k přihlášení nebo odhlášení uživatele kdekoliv v aplikaci, nekomunikuje se přímo se službou *User*, ale provede se pouze broadcastování události typu *loggedIn* nebo typu *loggedOut*. Při broadcastování události typu *loggedIn* se signalizuje aplikaci, že se uživatel přihlásil. S touto událostí se posílá proměnná se jménem přihlášeného uživatele. V menu se při odchytu této události zavolá funkce služby *User*, aby se v této službě nastavilo, že se uživatel přihlásil. Při broadcastování události typu *loggedOut* se signalizuje v rámci aplikace odhlášení uživatele. Tato událost nemá žádné dodatečné informace. Při odchytu se zavolá funkce služby *User* a nastaví se, že se uživatel odhlásil. Na první pohled vypadá, že by bylo jednoduší, kdyby se všude při přihlášení nebo odhlášení přímo komunikovalo se službou *User*. To by ovšem controlleru *UserBarCtrl* znemožnilo ihned rozpoznat, že se uživatel zrovna přihlásil nebo odhlásil, a tak by se v menu zobrazovaly špatné položky a špatné jméno uživatele. Takhle, díky odchytu událostí spojených s přihlašováním nebo odhlašováním, controller hned může změnit menu podle aktuální situace.

5.2 Home

Toto je klasická home stránka, jejímž účelem je přivítat nového uživatele. Je zde text vysvětlující účel této webové aplikace a tlačítka pro rychlou hru jednoho hráče. Tyto tlačítka jsou zde tady proto, aby když uživatel poprvé otevře, tak aby si mohl hru ihned vyzkoušet. Úvodní stránku lze prozkoumat zde A.1.

5.3 Login

Tato stránka slouží pro přihlašování uživatelů. Pro přihlašování se používá klasický formulář se dvěma položkami: S první položkou pro uživatelské jméno a s druhou položkou pro heslo. Využívá se zde pouze validace, která zjišťuje, zda položky s uživatelským jménem a heslem nejsou prázdné. Při neprázdnosti položek se vyhodí validační error *required*. V případě výskytu validačního erroru se vyřadí z činnosti tlačítko pro přihlášení.

Při odeslání žádosti pro přihlášení se zkontroluje na straně serveru, zda uživatel s daným jménem existuje. Pokud ano, pokusí se zjistit, zda se shodují hesla. Veškerá hesla v databázi jsou zahešovaná pro zvýšenou bezpečnost uživatele, porovnávají se tedy pouze heše obou hesel. V případě shody hesel se odešle informace, že požadavek byl úspěšný a stránka se přesměruje v klasickém případě do stránky s výběrem her. Kam přesně se má stránka přesměrovat, zjistí pomocí použití služby *LoginChannel*

V případě neúspěchu se pošle zpátky číslo chyby a aplikace vypíše patřičnou chybovou hlášku. Jsou zde použita direktiva pro vypisování hlášek *myAlert*. Direktiva *myAlert* je napojena na službu *LoginChannel*. To umožní jiným stránkám nastavit, jaké hlášky se zde mají vypisovat. Náhled na stránku pro přihlášení lze vidět tady A.6

5.4 Registrace

Vytvořil jsem registrační formulář pro nové uživatele. V této aplikaci není sice povinné se nejprve zaregistrovat, předtím než začnou hrát, ale je to pro ně výhodné, protože se jim

budou ukládat body a budou postupovat v úrovních. Registrační formulář je jednoduchý, jsou zde pouze 4 položky:

- Login
- Email
- Password
- Password match

Využil jsem frameworku Angular pro validování políček ve formuláři ještě před odesláním žádosti o registraci. To umožní uživatelům snadnější práci s registračním formulářem, protože hned poznají, jestli něco ve formuláři vyplnili špatně. Je zapnuta validace u všech políček, zda-li nejsou prázdná, jinak se vyhodí validační error *empty*.

Všechny položky registračního formuláře jsou povinné. Dále jsem implementoval automatickou validaci položky pro email, zdali je email ve správném tvaru. Při nesprávném tvaru emailu je vyhozen validační error *email*.

Všechny tyto validační mechanismy jsou implementovány ve frameworku Angular a jdou jednoduše použít. Pouze pro validování obou položek pro heslo jsem musel vytvořit svou vlastní direktivu, která kontroluje obě položky pro heslo a v případě nesrovnalosti vyhodí validační error *required*. Pokud je vyhozen některý z validačních errorů, objeví se jeho popis u položky formuláře, která tento error vyhodila. Také v případě nevalidního formuláře se zablokuje tlačítko na odesílání žádosti, aby uživatel nemohl odeslat formulář s nevalidními položkami.

Po odeslání žádosti o registraci se provede validace i na straně serveru. Validace na straně serveru kontroluje, jestli jsou poslaná data ve správném tvaru, protože validace v javascriptu lze jednoduše obejít a poslat data serveru v jakékoliv formě. Proto nelze mít validaci pouze v prohlížeči.

Oproti klientské straně se na serveru navíc ověřuje, jestli požadovaný login není už zabrán jiným uživatelem. V případě špatné validace na straně serveru se pošle klientovi chybový kód. Podle chybového kódu aplikace zobrazí patřičnou chybovou hlášku. V případě, že se odešle žádost, která projde veškerou validací, vytvoří se v databázi nový uživatel s patřičným emailem a heslem. Uživatel je poté přesměrován do přihlašovacího okna, aby se mohl ihned přihlásit. Registrační stránka lze vidět A.7.

5.5 Statistiky

Toto je stránka se statistikami všech uživatelů. Uživatel může přijít na tuto stránku a podívat se, jak si vedou ostatní hráči. U každého hráče je totiž informace o tom, jak velké je jeho skóre.

Při příchodu uživatele na tuto stránku se odešle HTTP request na server. Server přijme tento request a načte z databáze všechny uživatele. Tyto uživatele seřadí podle velikosti jejich levelu, od toho s největším po toho s nejmenším. Klientské části se odešlou zpátky pouze jména hráčů a jejich levely. Na stránce se poté všichni hráči zobrazí do tabulky.

Uživatel zde má možnost vyhledat si pouze jednoho konkrétního hráče. Je zde vyhledávací formulář, do kterého stačí vložit pouze podřetězec vyskytující se v hráčově jméně. Vyhledávání hráčů probíhá kompletně na straně klienta. Implementoval jsem vyhledávání hráčů za použití angular filtrů.

Na této stránce se kvůli přehlednosti zobrazí maximálně 10 hráčů, ostatní hráče lze zobrazit za použití lišty se stránkami. Tato lišta se stránkami pochází z UI-Angular a umožňuje uživatelům, aby mohli jednoduše prolistovat celý záznam se všemi statistikami hráčů. Rozvržen stránky lze vidět na A.8.

5.6 Detaily uživatele

Na této stránce se zobrazují uživatelům detaily jejich účtů. Je zde zobrazeno jejich uživatelské jméno, email, level a jejich skóre. Při spuštění této stránky se serveru odešle žádost o detaily přihlášeného uživatele. Server po přijetí žádosti načte *SessionID* uložené v cookies. Pomocí Session identifikuje daného uživatele. V případě, že se jedná o legitimního přihlášeného uživatele, tak načte z databáze jeho uživatelské jméno, email a skóre. To také pošle zpátky klientovi, aby se mu to mohlo zobrazit.

Na této stránce lze také změnit si heslo. Nejprve je třeba zobrazit formulář pro změnu hesla a to kliknutím na patřičné tlačítko. Po kliknutí se roztáhne formulář pro změnu hesla. Roztáhnutí formuláře je provedeno za pomoci knihovny UI-bootstrap.

Tento formulář má 3 položky – položka pro staré heslo, položka pro nové heslo a položka pro zopakování nového hesla. Uživatel zde musí znovu zadat své heslo kvůli bezpečnosti, aby se předešlo případům, kdy se uživatel zapomene odhlásit a někdo jiný mu změní heslo a ukradne účet. Je zde použita direktiva *passwordMatch* u položky pro opakování nového hesla. Slouží pro ověření, zda jsou obě hesla stejná. Všechny 3 položky jsou povinné, proto se u formuláře validuje, jestli položky nejsou prázdné, v případě prázdnosti některé z položek se vyhodí validační error *required*. Po validaci formuláře lze odeslat požadavek na změnu hesla.

Server po přijetí požadavku na změnu hesla nejprve ověří uživatele podle SessionID. Poté se ověří, zda se heslo, které bylo posláno na server, shoduje s heslem přihlášeného uživatele. V případě neshody se odešle zpátky HTTP response s kódem chyby a s kódem 400. Dle kódu chyby se v aplikaci vypíše chybové upozornění pro uživatele. Pokud je heslo správné, tak se změní na heslo nové. Ve formuláři jsou sice dvě položky pro nové heslo, ale serveru se posílá hodnota pouze jedné z nich. Veškerá kontrola toho, jestli jsou obě dvě hesla stejná, se tedy provádí pouze na straně klienta. Je zde použita také direktiva *myLoading*. Tato direktiva je aktivní ve dvou případech: při přístupu na stránku a při odesílání požadavku na změnu hesla. Jak stránka vypadá lze vidět na A.13.

5.7 Hledání hry

Na této stránce jsou všechny dostupné hry vytvořené ostatními hráči. Vybírají se pouze hry určené pro více hráčů. Při příchodu na stránku odešle klient HTTP request na server. Server vyhledává v databázi všechny hry, které ještě nezačaly.

Tyto hry poté pošle server zpátky klientovi. V seznamu her odeslaných klientovi jsou jména her, jejich stvořitelé a typ hry. Typ hry je zde reprezentován číselným kódem. Proto musí klient upravit seznam her, aby změnil číselný kód hry za řetězec, který je pro uživatele mnohem více informačně vypovídající než číselný kód. K tomu se využívá služba *Translator*. Všechny hry jdou vidět na stránce v přehledné tabulce. Hráč se může připojit do dané hry takovým způsobem, klikne na její název. To ho přesměruje do lobby dané hry.

Aby uživatelé vždy viděli na této stránce aktuálně vytvořené hry, tak se serveru posílá žádost o seznam her opakovaně. Abych toho docílil, využil jsem služby *$timeout*. Nastavil

jsem pomocí ní, aby se tato událost opakovala co 3 sekundy. Hráči tak každé 3 sekundy uvidí aktuální hry, ke kterým je možné se připojit.

Na tuto stránku se přesměrovávají stránky pro přihlášení po úspěšném přihlášení. Také pokud je uživatel v lobby nebo v průběhu hry a stane se nějaká událost, která ukončí hru či vyhodí uživatele z lobby (v běžném případě dochází k takové události při neočekávaném odpojení druhého hráče), tak se aplikace přesměruje na tuto stránku. Po takovém přesměrování je třeba informovat uživatele, proč k němu došlo. K tomu se využívá služba *GameChannel*. Pro vytváření informačních a chybových hlášek se využívá direktiva *myAlert*. Vzhled stránky lze vidět na A.4.

5.8 Multiplayer vytváření hry

Pro vytváření hry existuje formulář, ve kterém jsou tyto tři položky – jméno hry, typ hry, počet kol. Všechny položky jsou povinné, proto se vyhodí validační error *required* vždy, když nějaká položka chybí. Jméno hry může uživatel zvolit libovolně. Je zde nastaveno pouze omezení, že jméno hry musí mít aspoň jedno písmeno a maximální počet písmen pro jméno hry je 40. Uživatel má na výběr ze čtyř typů hry:

- Tag Image! – Hra s hádáním objektů na obrázku
- Tag Image Extreme! – Hra s hádáním činností, které se dějí na obrázku. Zde je třeba uhádnout sloveso a podstatné jméno
- Tag Video! - Hra s hádáním činnosti, která se děje na videu
- Find Tag! - Hra s hledáním objektu na obrázku.

Položka pro počet kol má defaultní hodnotu 5, což jsem stanovil jako přiměřený počet kol ve hře, která tak není ani moc dlouhá ani příliš krátká. Pro počet existují dva validační mechanismy. První validační mechanismus ověřuje, zda je v položce číslo. Pokud ne, vyhodí se validační error *number*. Druhý validační mechanismus ověřuje, zda je číslo větší než nula.

Pouze pro typ hry *Tag Image!*, *Tag Image Extreme!* a *Find Tag!* je možné hrát více kol. Proto položka pro maximální počet kol zmizí za použití direktivy *ng-show*, pokud se nastaví typ hry videa.

Po splněních všech validačních mechanismů lze odeslat požadavek na server. Při zpracování požadavku se tlačítko znovu zablokuje, aby uživatel nevytvořil her více.

Po odeslání požadavku serveru se ověří, zda velikost jména neporušuje omezení velikosti, zda je typ hry známý a zda počet kol je číslo větší než 0. Při porušení těchto omezení se odešle chybový kód, který vypíše uživateli patřičnou hlášku.

Při správně zadaných údajích se vytvoří hra. Klientovi se odešle úspěch jeho požadavku a ten přesměruje stránku do lobby pro vytvořenou hru. Vzhled stránky lze vidět na A.2.

5.9 Lobby

Toto je stránka s lobbym hry. Zde se připojují hráči, předtím než začne hra. Herní lobby je určeno pouze pro dva hráče. Hráč se může dostat do stránky s herním lobby dvěma způsoby — buď vytvoří svou vlastní hru nebo najde nějakou hru v seznamu her.

Při příchodu na tuto stránku odešle klient HTTP request serveru s žádostí o připojení se k lobby. Zasílaný HTTP request je metody POST, protože obsahuje ID lobby, ke které se

chce klient připojit. ID lobby je zde předáno v rámci aplikace pomocí URL. Klient extrahuje z URL za pomocí použití modulu *ngRoute*.

Server po přijetí žádosti klienta zkontroluje, zda se chce k lobby připojit přihlášený uživatel. V případě přihlášeného uživatele se server pokusí nalézt hru s daným ID. Při nalezení hry zkontroluje, zda tato hra nebyla ještě spuštěna. Dále také server zjišťuje, kolik hráčů je k dané hře připojeno.Během připojování tedy může dojít k následujícím chybám. Nespuštěná hra s daným ID nebude existovat. Ve hře už budou oba dva hráči připojeni.

V případě těchto chyb se pošle klientovi HTTP response s chybovým kódem a stavovým kódem 400. Z chybového kódu klient zjistí co se stalo špatně a o této události uživatele informuje patřičnou hláškou. V případě že nedojde k žádné takovéto situaci, hráč se připojí ke hře.

Využil jsem služby *$timeout*, abych zde naplánoval událost, která neustále kontroluje stav lobby. Každé 2 sekundy se odešle HTTP request serveru s žádostí o aktuální stav lobby.

Při přijetí takovéto žádosti stačí serveru zjistit, o jakého uživatele se jedná. U přihlášeného uživatele se pokusí zjistit, v jakém lobby je uživatel připojen. Poté zjistí, jestli je v lobby připojen i druhý uživatel. Seznam obou uživatelů se odešle zpátky klientovi. U každého uživatele je v seznamu jeho jméno a ID.

Může ale nastat situace, kdy uživatel není tvůrcem hry, a tvůrce hry se odpojí. Po odpojení tvůrce hry se hra automaticky zablokuje a tak server nenalezne žádnou hru, ve které by uživatel byl. Při této situaci odešle server klientovi zpátky HTTP response se stavovým kódem 400 a s chybovým kódem, který klienta bude informovat o tom, co se stalo.

Když klient dostane zpátky seznam obou uživatelů, upraví změny na stránce. Uživatel tak pozná, že se někdo odpojil či připojil do lobby, ve kterém je.

Tvůrce hry má možnost vyhodit z lobby uživatele, s kterým nechce hrát. Vyhodí ho tak, že klikne na ikonku s křížkem, která je hned vedle jména připojeného uživatele. Po kliknutí na ikonku se odešle HTTP request serveru o odpojení druhého uživatele. Server z requestu zjistí, o jakého přihlášeného uživatele se jedná a v jaké hře je připojen. Ověří, zda je uživatel tvůrcem dané hry, a jestli tak má právo vyhazovat druhého hráče ze hry. Poté druhého hráče odpojí ze hry. Klientovi se v případě úspěšně vyhozeného hráče odešle pouze HTTP request se stavovým kódem 200 a s žádnou další informací. Klient po příjmu takové zprávy ihned aktualizuje lobby a začne ukazovat na stránce, že místo uživatele je prázdné.

Když vyhozený hráč pošle žádost o aktuální stav lobby, tak mu server odpoví s chybovým kódem. Přijetí chybového kódu přesměruje uživatele pryč z lobby.

Tvůrce hry má dostupné tlačítko na spuštění hry. Při zmáčknutí tohoto tlačítka se odešle HTTP request serveru. Server přijme tento požadavek a nastaví status hry na spuštěný. Poté vybere z databáze informace o dané hře. Pokud je typ hry anotace obrázků nebo anotace akcí na obrázku, tak server vybere obrázky, které se budou anotovat. Vybere takový počet obrázků, jaký je nastavený počet kol. Server po té odešle HTTP response klientovi s informacemi o spuštěné hře.

Spoluhráč je informován o startu hry z pravidelných zpráv o aktuálním stavu lobby. Pokud hra začala, tak klientovi přijde ve zprávě pokyn ke startu hry. Tato zpráva bude obsahovat také hlavní informace o hře.

Když klient dostane pokyn ze serveru, že hra začala, ať už ve zprávě o aktuálním stavu lobby nebo protože poslal žádost o start hry, tak si všechny přijaté informace o hře uloží přes službu *Game*, aby byly i později k dispozici. Tato služba se využívá také na přesměrování do správné stránky s hrou. Vzhled stránky lze vidět na A.5.

5.9.1 URL pro připojení

Uživatel má k dispozici tlačítko s ikonkou vlaječky, které po kliknutí aktivuje modal. V modalu se zobrazí URL adresa dané lobby. URL se získá za využití služby *$location* Pokud uživatel odešle tuto URL adresu nějakému svému známému, tak ten se jednoduchým vložením této adresy do prohlížeče připojí ihned do lobby. Tento mechanismus je zde pro usnadnění vyzývání ostatních hráčů.

5.9.2 Uživatelské detaily

U jmen uživatelů v lobby existují také tlačítka s ikonkou siluety člověka. Tato tlačítka aktivují také modal. Tento modal zobrazí detaily daného uživatele. Pro získání těchto detailů se pošle serveru HTTP request. Tento HTTP request je metody POST a obsahuje ID hledaného uživatele. ID hledaného uživatele má controller modalu k dispozici z controlleru pro lobby.V detailech uživatele se ukáže jeho jméno a nastřádané body. Ve všech modalech existuje tlačítko OK, které daný modal zavře. Modal se dá také zrušit klávesou ESC.

5.9.3 Chat

V levém panelu jsem implementoval chat. Chat slouží pro komunikaci mezi uživateli před začátkem hry. Zpráva, kterou chce uživatel poslat svému spoluhráči, se píše do textového inputu. Po kliknutí na tlačítko se odešle HTTP request na server. HTTP request používá metodu POST, v jeho těle je umístěná zpráva.

Server identifikuje uživatele z HTTP requestu. Nalezne lobby, ve kterém se uživatel nachází, a poté vytvoří záznam v databázi s danou zprávou, která bude přiřazena danému uživateli a lobby. U každé zprávy je uložen čas, kdy se vytvořila. To umožňuje seřadit zprávy dle pořadí, v jakém byly poslány.

Nové zprávy v chatu se získávají v HTTP requestu společně s informacemi o stavu lobby. Server se podívá do databáze pro dané lobby a vybere z ní všechny zprávy. Zprávy se seřadí podle času. Klientovi se pošlou zprávy v seznamu, kde každá položka obsahuje text zprávy a jméno uživatele, který zprávu napsal. To se také zobrazí na stránce, uživatel tak vidí, které zprávy napsal on a které zprávy napsal spoluhráč.

5.10 Hra pro jednoho hráče - vytváření hry

Tato stránka je velice podobná stránce pro vytváření hry více hráčů. Ve formuláři ale chybí položka pro jméno hry — v případě hry jenom jednoho hráče tato položka není třeba, protože hra se nikde nezobrazuje a ani se k ní nelze nijak připojit.

Ve formuláři jsou dvě položky — typ hry a počet kol. Jsou zde pouze tři typy hry:

- Tag Image! – Hra s hádáním objektů na obrázku
- Tag Image Extreme! – Hra s hádáním akcí, které se dějí na obrázku. Zde je třeba uhádnout sloveso a podstatné jméno
- Tag Video! – Hra s hádáním akcí, které se dějí na videu. Zde je třeba uhádnout sloveso

U počtu kol se klasicky nastaví, kolik kol má hra trvat. Toto políčko formuláře má automatickou validaci sloužící ke kontrole, zda je zadaný prvek číslem. Další validace je na to, aby se ověřilo, zda číslo je větší než nula. V případě, že uživatel napíše něco jiného než číslo, formulář nepůjde odeslat a zobrazí se odpovídající hláška.

Při úspěšné validaci formuláře lze odeslat požadavek na server. Na serveru se validuje zda byla vybraná hra ve správném rozsahu a jestli počet kol je opravdu kladné, celé číslo. V úspěšném případě se vytvoří daná hra.

Když klient příjme odpověď serveru, využije službu *Game* a přesměruje stránku do stránky obsahující hru. Vzhled stránky lze vidět na A.3.

5.11 Herní stránka - Find Tag!

5.11.1 Kreslení bounding boxů

V této hře se hraje pouze s obrázky, které už někdo anotoval. To proto, aby systém věděl, co se na obrázku nachází.

Na herní stránce se zobrazí hledaný objekt a obrázek na kterém se tento objekt nachází. Úkolem uživatele je označit dané slovo na obrázku. Uživatel má 45 sekund na to, aby tento úkol splnil.

Obrázek je na stránce vykreslen pomocí nového HTML 5 elementu canvas. Toho bylo docíleno kvůli tomu, aby se dalo na obrázek kreslit. Když uživatel klikne na plátno a pohne myší, tak se bude na canvasu vykreslovat čtverec z původního bodu do bodu, kde se nachází myš. Při druhém kliknutí, se čtverec uloží. Uživatel tak může na plátně vytvořit libovolný čtverec, kterým zakreslí hledaný objekt. Obdélníků může uživatel vytvořit libovolný počet. Obdélníky se ukládají do proměnné, která se později odešle na server. Obdélníky jsou při ukládání reprezentovány souřadnicemi vrchního levého bodu a spodního pravého bodu.

Vedle obrázku se nachází panel se dvěma tlačítky. První tlačítko s ikonkou koše slouží pro vymazání všech čtverců z plátna. Druhé tlačítko slouží pro okamžité ukončení tahu.

Při vypršení limitu nebo stlačení tlačítka pro konec tahu se posle serveru HTTP Request se souřadnicemi obdélníků. Server příjme tyto souřadnice a uloží si je do databáze. Po té zkontroluje, jestli druhý uživatel také ukončil svůj tah. V případě, že ještě neodeslal tak pošle klientovi zpátky odpověď, že musí počkat.

Když oba dva hráči ukončí svůj tah, tak dojde k vyhodnocení jejich obdélníků. Každý obdélník jednoho hráče se porovnává s obdélníky druhého hráče. Porovnávají se oba dva vrcholy s vrcholem druhého obdélníku. Pokud oba dva vrcholy jsou v blízkosti 50 pixelů, od vrcholu obdélníků druhého, tak jsou pak obdélníky považovány za shodné. Bodování za bounding boxy probíhá následovně. Důležitým číslem je počet obdélníků, které hráč či jeho spoluhráč označili, vybírá se to vyšší *(maximumPoslanychObdelniku)*. Dalším číslem, které se použije při výpočtu výsledku, je číslo shodných obdélníků *(shodujiciSeObdelniky)*. Výpočet bodů poté proběhne pomocí této rovnice 5.1. Vzhled stránky lze vidět na A.12.

$$pocetBodu = 200 \times \frac{maximumPoslanychObdelniku}{shodujiciSeObdelniky} \tag{5.1}$$

Shodné bounding boxy jsou hledané anotace k obrázku, server je proto uloží do databáze k danému slovu a k danému obrázku.

Server poté zjistí, jaké slovo a jaký obrázek, budou hráči hrát v dalším kole. Poté pošle klientovi zpátky data o tom, kolik získal hráč bodů, jaké bounding boxy jsou shodné s jeho oponentem, jaký je nový obrázek a slovo. Klient použije službu *Game*, aby si tyto data uložila. Za použití stejné služby se pak přesměruje do stránky s výsledky.

5.11.2 Stránka s výsledky

Klient nejprve získá potřebná data pomocí služby *Game*, aby mohl zobrazit výsledky kola. Podobně jako u her Tag Image! a Tag Image Extreme! je i zde na stránce počítadlo jaký je aktuální obrázek a kolik obrázků bude celkem. Je zde také použita direktiva *experienceBar*, která zobrazuje uživateli jeho postup v úrovních. Hráč má pouze 10 sekund, aby zjistil jaké má skóre. Poté se hra přesměruje na nový obrázek. Jestli hráč nechce čekat 10 sekund, tak může použít tlačítko, které ho k novému obrázku přesměruje okamžitě.

Změnou oproti ostatním hrám je zde zobrazování obrázku ze hry. Na tomto obrázku jsou vykresleny všechny bounding boxy obou hráčů. Bounding boxy mají ale různé barvy.

- světle zelená - barva bounding boxů hráče, které jsou shodné s bounding boxy spoluhráče
- tmavě zelená - barva bounding boxů spoluhráče, které jsou shodné s bounding boxy hráče
- červená - barva bounding boxů hráče, které nejsou shodné s žádným jiným bounding boxem spoluhráče
- vínová - barva bounding boxů spoluhráče, které nejsou shodné s žádným jiným bounding boxem hráče

Tato legenda k barvám obdélníků se zobrazuje také hráčům.

5.12 Herní stránka - Tag Image!

V případě hry pro jednoho hráče se ke hře vygenerují obrázky, ke kterým už existuje nějaká anotace, získaná ve hře více hráčů. V případě hry pro více hráčů, se generuje obrázky úplně náhodně.

Na herní stránce se zobrazí formulářové políčko pro zadávání objektů a obrázek, který se anotuje.

Při napsání objektu do políčka formuláře se zobrazí modal s definicemi. Tento modal odešle serveru požadavek o zaslání synsetů z Wordnetu. Když server příjme požadavek tak zkontroluje jaký typ synsetů klient chce. Klient je v této hře nastaven tak, že chce pouze podstatné jméno. Na serveru se tedy využije funkce nástroje NLTK, aby z corpusu Wordnet vrátil pole synsetů, který odpovídá danému slovu. Server z pole synsetů vytvoří pole, které bude obsahovat pro každý synset jeho identifikační kód, definici a první ze synonym. Toto upravené pole odešle server zpátky klientovi. Pokud nástroj NLTK nenalezne žádné odpovídající synsety, odešle se klientovi prázdné pole.

Klient v modalu zobrazí hráči nabízené synsety. Pokud je pole synsetů prázdné, zobrazí se hláška, že dané slovo nebylo nalezeno. Hráč může kliknutím vybrat jeden synset a ten se mu tak přidá do tabulky. V tabulce může vidět hráč všechny synsety, které pro daný obrázek vybral. V případě, že hráč vybral špatný synset, můžu ho zrušit kliknutím na ikonku křížku.

Hráč má 45 sekund na to, aby vybral všechny synsety, které se na obrázku nacházejí. Poté se odešle serveru HTTP request se seznamem synsetů hráče. V případě že hráč, nechce tak dlouho čekat, můžu kliknout na tlačítko send. Po té se synsety odešlou serveru okamžitě.

Hra se hraje tímto způsobem stejně pro multiplayer i singleplayer. Akorát po odeslání synsetů serveru už existují rozdíly. Vzhled stránky lze vidět na A.9

Multiplayer

Server příjme všechny synsety a uloží je do databáze. Poté zkontroluje, jestli pro danou hru poslal své synsety i druhý hráč. Pokud ne, tak odešle klientovi zprávu, že druhý hráč není připraven. Když klient zjistí, že druhý hráč není připraven, přesměruje se na čekací stránku. Tato stránka generuje periodicky HTTP requesty na server, aby zjistila, jestli druhý hráč ukončil svůj tah.

V případě že oba dva hráči ukončili svůj tah, server porovná jejich výsledky. Synsety obo dvou hráčů se musí shodnout, aby hráči byly připsány body. Pokud se synsety shodnou, uloží se v databázi jako anotace k danému obrázku. Pokud už taková anotace existuje, zvětší se její počet použití. Za každé shodlé synsety dostanou hráči 100 bodů. Výsledek tohoto kola se odešle zpátky klientovi. Klient si ho uloží pomocí služby *Game* a také pomocí této služby přesměruje hru do tabulky se skóre.

Singleplayer

Hráč odešle synsety serveru. Server načte z databáze anotace patřící k danému obrázku. Z těchto anotací vybere tu, která má největší počet výskytů k danému obrázku. Tento počet si uloží do proměnné (*pocetVyskytuNejcastejsiAnotace*). Poté projede seznamem přijatých synsetu. Každý synset se pokouší nalézt v databázi, na místě, kde jsou uložené již vytvořené anotace (vytvořené hrou více hráčů) k danému obrázků. Pokud najde shodnou anotaci, uloží si počet jejich výskytu (*pocetVyskytuShodneAnotace*). Počet výskytů této anotace následně zvýší o jedna. Poté pro tento vstup vypočte počet bodů podle této rovnice 5.2

$$pocetBodu = 50 \times \frac{pocetVyskytuNejcastejsiAnotace}{pocetVyskytuShodneAnotace} \tag{5.2}$$

Následně se výsledky pošlou zpátky klientovi. Ten použije službu *Game*, aby si je uložil a aby přesměroval stránku do stránky se skórem.

5.13 Herní stránka - Tag Image Extreme!

Tato hra je podobná hře Tag Image!. Na herní stránce se také objeví obrázek pro získání anotací. Formulářové políčka pro zadávání vstupu od uživatele jsou zde ale dvě. V této hře je třeba zadávat nejen objekt ale i činnost, kterou objekt provádí. Jedno políčko je tedy na sloveso a druhé na podstatné jméno. Při zadání vstupu, se zde objeví také modal. Tento modal pošle HTTP request serveru žádost o synsety. Serveru pošle v HTTP requestu sloveso i podstatné jméno.

Server poté vyhledává pomocí NLTK nástroje, shodující se synsety sloves a podstatných jmen. V případě úspěchu se odešlou klientovi neprázdné dvě pole synsetů. Tyto pole se před odesláním zpracují podobě jako ve hře Tag Image!. V poli tedy bude jméno synsetu, jedno ze synonym a definice synsetu.

Klient po přijetí zprávy serveru zobrazí v modalu dvě tabulky se synsety. V jedné tabulce jsou synsety sloves a v druhé jsou synsety podstatných jmen. Když uživatel označí sloveso a podstatné jméno, modal se ukončí a vybrané synsety se přidají do tabulky. Tyto synsety jdou smazat stejně jako ve hře v Tag Image!.

Uživatel má 45 sekund na to, aby označil všechny objekty, vykonávající nějakou činnost.

Po odeslání vybraných synsetů serveru už existují rozdíly mezi hrou pro více hráčů a hrou pro jednoho hráče. Jak vypadá tato hra lze vidět na A.10

Hra pro více hráčů

Oba dva hráči musí ukončit svůj tah, poté dojde k ohodnocení jejich výsledků. Kontrolují se zadané synsety hráčů mezi sebou. Kontrolují se mezi sebou dvojice slovesa a podstatného jména. Pokud se shodnout synsety s podstatným jménem i slovesem, dostanou za to hráči 100 bodů. V případě, že se shodnou jenom se slovesem nebo podstatným jménem, dostanou za to polovinu. V případě se úplného shodnutí se dvojice považuje jako správná anotace a uloží se do databáze.

Hra pro jednoho hráče

Dvojice podstatného jména se porovnávají s hodnotami dvojic uložených v databázi po úspěšném shodnutí. V případě že se shodnou, tak získají za uloženou dvojici počet bodů podle rovnice 5.2

5.14 Tabulka se skóre

Tato stránka zobrazuje skóre ke hrám Tag Image! i Tag Image Extreme!. A to jak u verze pro jednoho hráče, tak i pro více hráčů. Pro jednotlivé hry je zde pouze několik rozdílů.

Klient načte pomocí služby *Game* výsledky hry. Dále načte počet bodů, které hráč ve hře získal. Poté se použije direktiva *ExperienceBar*, která ukáže progress bar, který se posune o získaný počet bodů. V případě nově získaného levelu, tuto informací zobrazí.

Na této stránce jsou i tabulky, ve kterých jsou napsané synsety, které hráči resp. hráč posílal.

Tabulka je zde pouze jedna, pokud je pro hru jednoho hráče. V první tabulce jsou synsety, které zadal hráč. U každého synsetu je zobrazen počet bodů, který byl za tento synset získán. V případě hry Tag Image! je v každém řádku jenom synset podstatného jména, v případě hry Tag Image Extreme! jsou zde synsety dva, jeden pro sloveso a druhý pro podstatné jméno.

V případě hry pro více hráčů je zde i druhá tabulka. Ta také obsahuje synsety, ale takové které odpověděl druhý hráč.

Synsety, za které byly získány body, jsou napsány zelenou barvou. Ty za které nebyly přiděleny žádné body, jsou napsány červenou barvou.

Hry pro více hráčů mají časový limit, kdy hráč může zůstat na této stránce. Časový limit je 10 sekund. Na stránce je tento zbývající čas ukazován uživateli.

Pokud uživatel chce další hru, je zde pro to tlačítko, které ho dostane do další hry okamžitě.

Pokud byl dohraný poslední obrázek, tak zde není žádný časovač ani tlačítko pro další hru. Je zde ale tlačítko pro přechod do stránky se statistikami celé hry. Vzhled stránky lze vidět na A.8.

5.15 Statistiky hry

Ve stránce se statistikou hry, se ukážou statistiky celé odehrané hry. Data se statistika se zde získávají pomocí služby *Game*. Je zda zobrazen celkový počet bodů za všechny kola. Pro každé kolo zde existuje řádek v tabulce, kde jde vidět počet získaných bodů a je zde tlačítko s ikonkou lupy pro zobrazení daných objektů, které ve hře hráč zadal. V případě

hry pro více hráčů je zde i obdobné tlačítko pro zobrazení daných objektů druhým hráčem. Jak vypadají statistiky lze vidět na A.8

5.16 Herní stránka - Tag Video! pro více hráčů

5.16.1 Návrh videa

Pokud se spustí hra Tag Video!, tak server použije nástroj NLTK a vygeneruje 16 náhodně vybraných sloves pro oba dva hráče. Server odešle tuto skupinu 16-ti sloves klientovi každého hráče. Klient zobrazí 16 sloves patřící danému hráči a skupinu sloves druhého hráče si uloží na další použití pomocí služby *Game*. Slovesa jsou zobrazeny jako tlačítka, mezi kterýma jde libovolně přepínat. Hráč si musí z 16-ti sloves vybrat takové sloveso, které mu nejvíc vyhovuje. Poté má za úkol prohledat server youtube a nalézt vhodné video, které znázorňuje dané sloveso. Když nalezne toho video, uloží odkaz tohoto videa do formulářového políčka.

Klient neustále pozoruje změny v tomto formulářovém políčku. Při jeho změně se pokusí pomocí regulérního výrazu získat z dané URL kód youtube videa. Tento kód se vloží do komponenty, vytvořené vývojáři youtube na zobrazení videa. Hráč si tak může zkontrolovat, zda vložil správné video. U videa lze nastavit i čas, kdy má začít, a to pomocí dvou formulářových políček pro minuty a sekundy. Video bude trvat pouze 7 sekund, to je dostatečný čas, aby se dala znázornit nějaká činnost a zároveň, aby video nebylo příliš dlouhé.

Uživatel není časově limitován s odesláním videa. Ve formuláři se validuje, zda je políčko pro odkaz na video neprázdné a zda políčka pro minuty a sekundy obsahují pouze nezáporné čísla. Při splnění validace může uživatel odeslat video a své slovo. Hráč, který odeslal video jako první, musí čekat, dokud video neodešle i druhý hráč. Když jsou oba dva hráči připraveni, přesměruje se stránka do stránky s hádáním slovesa.

5.16.2 Hádání slovesa

Při přesměrování na stránku s hádáním slovesa, odešle server klientu kód youtube videa. Klient tento kód použije pro přehrání správného videa za použití komponenty od vývojářů youtube. Klient také zobrazí na stránce 16 dříve uložených sloves, z kterých vybíral spoluhráč.

Hráč by si měl přehrát video a podle toho uhádnout sloveso. Slovesa se vybírají v panelu tlačítek. Při vybrání slovesa, se sloveso zobrazí i s jeho definicí.

Když je hráč spokojen s výběrem slovesa, může odeslat formulář. Sloveso se poté odešle serveru. Znovu se čeká až odešle svou odpověď i druhý hráč. Když oba dva hráči odešlou svou odpověď, zpracují se výsledky. Ověřuje se, zda hráči vybrali stejné sloveso, jaké jejich oponent vybral při hledání videa. Za uhádnutí slovesa, získávají oba dva hráči 150 bodů. Server po té vrátí klientovi výsledky hry. Výsledky hry se uloží pomocí služby *Game* a pomocí této služby se i hra přesměruje do stránky s výsledky.

5.16.3 Výsledky hry

Po přesměrování na stránku s výsledky hry, si klient načte výsledky hry pomocí služby *Game*. Na stránce s výsledky hry se potom zobrazí, kolik hráč získal za celou hru bodů, zda uhádl správně sloveso a zda správně uhádl sloveso jeho spoluhráč. Na této stránce je použita direktiva *experienceBar*, která upozorní hráče na to, že postoupil v úrovni a také

to, kolik mu chybí do nové úrovně. Na stránce jsou také zobrazeny slova, které hráči hádali a slova, které byly správnou odpovědí.

5.17 Herní stránky - Tag Video! pro jednoho hráče

5.17.1 Hádání slovesa

Při spuštění této hry, se vyberou z databáze videa pro celou hru. Tyto videa jsou získávány z videí získaných při hře Tag Video! pro více hráčů. U videí je tedy i známa skutečná činnost, která se na videu děje (hráči podle této činnosti videa hledali).

Vybere se tolik videí, kolik je počet kol. Na herní stránce se zobrazí formulářové políčko a promítané video. URL promítaného videa získá klient pomocí služby *Game*.

Do formulářového políčka mají hráči za úkol napsat sloveso, které se hodí na činnost zobrazovanou ve videu na stránce. Když uživatel zadá slovesa, klient pošle požadavek serveru o seznam synsetů shodující se s názvem hledaného slovesa. Hráč poté specifikuje význam slovesa a odešle ho serveru.

Na serveru se odeslané sloveso porovná se správným slovesem, které na videu je. Pomocí nástroje NLTK se porovná podobnost těchto sloves v rámci corpusu Wordnet. Podle míry podobnosti sloves bude hráč odměněn body, pokud hráč napíše úplně stejné sloveso jaké má být, dostane plný počet bodů, tedy 75. Pokud hráč uhádne video úplně přesně, zvýší tak relevantnost anotace k videu.

Bodové hodnocení se odešle zpátky klientovi, který přesměruje uživatele na stránku s výsledky.

5.17.2 Stránka s výsledky

Na tento stránce se zobrazí uživateli počet bodů kolik získal. Bude zde také zobrazena míra podobnosti mezi slovesem co zadal a mezi slovesem, které je správné. Jako na každé stránce se výsledky je i zde použita direktiva *experienceBar*, která zobrazí uživateli jeho postup v úrovních.

Na stránce je tlačítko, které hráče přesměruje k novému videu.

Kapitola 6

Uživatelské testování

Vytváření hry pro široké masy lidí není jednoduchá činnost. Musel jsem se ve své práci soustředit i na to, aby aplikace byla jednoduše použitelná a uživatelsky přátelská. Proto jsem provedl uživatelské testování, abych se dozvěděl připomínky a mohl uživatelské rozhraní zlepšit.

6.1 Testovací scénáře

Abych mohl provést uživatelské testování, musel jsem nejdříve analyzovat vytvořenou webovou aplikaci. Cílem analýzy bylo zjistit, jaké testovací scénáře by bylo vhodné vytvořit. Testovací scénáře obsahují typickou modelovou situaci, ve které se nachází noví uživatelé testované stránky. Testovací scénář také má určitý cíl, jež je třeba pro dokončení scénáře splnit. Tyto testovací scénáře se poté dají testovacím uživatelům, aby se podle nich chovali. Pro splnění scénářů by měl testovací uživatel udělat většinu základních úkonů, které jsou na stránce potřeba [6].

6.1.1 Hraní hry pro jednoho

Prvním testovací scénař zní následovně: Uživateli bylo doporučeno jeho přáteli, že je tato hra zábavná a řekli mu tu, ať si zkusí zahrát hry pro jednoho. Cíle pro splnění tohoto scénáře tedy je, aby si uživatel zahrál hry pro jednoho hráče.

6.1.2 Změna hesla

Druhý testovací scénář je takový: Uživatel někomu půjčil svůj účet, ale už nechce, aby tento člověk měl k jeho účtu přístup. Cílem tohoto scénáře je tedy, aby si uživatel v systému změnil heslo.

6.1.3 Vytvoření hry pro hru s více hráči

Třetí scénář má takovéto znění: Uživatelovi bylo doporučeno, aby si zahrál hru Tag Video! s nějakým dalším hráčem, protože je to dobrá zábava. Tento scénář uživatel splní, pokud si zahraje hru Tag Video! v módu pro více hráčů.

6.2 Testování

Pro uživatelské testování jsem si vybral 5 testovacích uživatelů. Těmto uživatelům jsem dal instrukce, aby se v aplikaci chovali podle daných tří scénářů. Poté jsem tyto uživatele pozoroval jak jednají a na jaké objekty klikají. Uživatelům jsem dal instrukce, aby mysleli nahlas a říkali, co právě dělají a nad čím přemýšlí. Takto poznám, jak uživatelé reagují na prvky UI a co shledávají jako matoucí. V případě, že byly třeba testovány hry, pro více hráčů, tak jsem zastoupil druhého uživatele. Na konci testování jsem se uživatelů zeptal, jaké mají připomínky. Toto testování by mi mělo přinést důležité informace o chybách mého uživatelského rozhraní [7].

6.3 Připomínky

Po zpovídání hráčů jsem získal tento seznam připomínek:

- Při hraní hry pro anotace obrázku se v tabulce výsledku ukážou objekty, které zadal uživatel a které zadal jeho spoluhráč. Objekty, které zadal uživatel, byly barevně označeny, aby uživatel zjistil, které se neshodovaly s spoluhráčem a které byly správné. Objekty, které zadal spoluhráč, byly vypsány klasickou černou barvou. To vedlo ke zmatení testovacího uživatele, který tak měl pocit, že pouze on dostává body a spoluhráč ne
- Čas hry byl pro všechny uživatele, kteří se ke hře dostali poprvé, příliš krátký. Než se stačili orientovat a zjistit, jak se hra hraje, tak uběhlo celé kolo.
- Uživatelům se nelíbilo velké množství definic, které se ji dávalo na výběr, při hře s popisováním obrázků. Velice je zdržovalo číst popis např. 10-ti definic při zadání nového slova.
- Uživatelé byli v lobby zmateni ikonkou vlajky, nedokázali odhadnout, že jim tato vlajka vygeneruje link, který mají poslat druhému hráči.
- Při registraci byli uživatelé nespokojeni, že nemůžou mít jméno dlouhé pouze jedno písmeno.
- Ve hře Tag Video! nevěděli k čemu vybírat slovo a co mají dělat.
- Uživatelé byli nespokojeni kvůli faktu, že musí příliš hledat, kde mají kliknout, i když si chtějí hru jenom rychle vyzkoušet
- V případě, kdy uživatelé se snažili přihlásit, tak byli zmateni z chybové hlášky. Nevěděli, že nezadali správné heslo.
- Uživatelům se nelíbilo, že se musí před hrou registrovat a přihlašovat.

6.4 Změny vykonané díky testování

- Změnil jsem barvu textu, kterou byly vypisovány neúspěšné objekty druhého hráče na červenou a zelenou, aby uživatel věděl, že i jeho protivník je postihován stejně.

- Prodloužil jsem čas, jaký mají uživatelé pro jednotlivé kolo. Abych hra nebyla příliš dlouhá, při hře dvou zkušených hráčů, tak jsem přidal do hry tlačítko pro ukončení tahu, aby hráči nemuseli čekat než se tah ukončí sám.
- Nechal jsem vypisovat uživatelům pouze tři první definice, z toho důvodu, že jedna z těchto tří první definic ve většině případů popisuje hledané slovo.
- Ikonku vlaječky jsem vyměnil za ikonku řetězce (anglicky je řetěz link, tedy jako odkaz). To by mělo lépe signalizovat, že uživatel po kliknutí dostane odkaz, který mohou poslat svým kamarádům.
- Změnil jsem omezení na serveru, takže nyní mohou mít uživatelé jméno, které je dlouhé pouze jedno písmeno. Není totiž žádný důvod proč uživatele takhle omezovat.
- Zlepšil jsem popis k hře Tag Video!. Hned na začátku je uživateli řečeno, že musí vybrat nejvhodnější sloveso, které půjde jednoduše naleznout na Youtube.
- Na úvodní stránku jsem umístil tlačítka na odstartování rychlé hry pro jednoho hráče, takhle si noví uživatelé zapnou stránku a jedním klikem můžou hned hrát.
- Chybová hláška před špatném přihlášení pouze suše konstatovala, že se přihlášení nepodařilo. Proto jsem nastavil server, aby vracel víc chybových kódu a mohl tak uživatele informovat, že zadal nesprávné heslo.
- Opravit tuto připomínku bylo nejsložitější, předělal jsem celý systém, aby i nepřihlášení uživatelé měli přístup do části aplikace, kde mohli pouze přihlášení. Změnil jsem lobby a zobrazování hráčů, aby ukazovalo nepřihlášené hráče jako Anonymní.

Kapitola 7

Testování naimplementovaných her

Provedl jsem testování mnou implementovaných her, abych mohl vyhodnotit přínosy těchto her pro sběr dat. Cílem testování bylo také to, abych zjistil, jak se uživatelům dané hry zamlouvají. Testoval jsem multiplayerové verze všech mnou vytvořených her: Tag Image!, Tag Image Extreme!, Video Tag! a Find Tag!.

Pro testování jsem využil 16 lidí, kteří měli za úkol si tyto jednotlivé hry zahrát. Tito lidé hráli hry po dvojicích.

V klasické hře, se hraje s obrovským počtem obrázků, které jsou vybírány náhodně. Já jsem ale kvůli testování připravil speciální sady obrázků. Všechny tyto obrázky jsem získal ze stránek projektu Imagenet. Speciální sady jsem vytvořil zvolil z toho důvodu, abych poznal, jak lidé na jednotlivé obrázky odpovídali a abych mohl lépe porovnat, zda se podařilo obrázky takovýmto způsobem anotovat.

7.1 Tag Image!

Pro tuto hru jsem zvolil 10 obrázků. Obrázky se lišili počtem objektů, který se na nich vyskytuje. Na některých obrázcích byl zobrazen velký počet objektů. Takové obrázky jsem vybral z toho důvodu, abych zjistil, jak uživatelé jsou schopni pochytit všechny objekty, které se tam nacházejí.

Uživatelé hráli hru s 5-ti obrázky. Celkem bylo zahráno 8 her. Každý obrázek byl tedy v průměru anotován 4 dvojicemi uživatelů. V tabulce 7.1 jde vidět rozvržení získaných anotací k jednotlivým obrázkům. Z her bylo tedy získáno 48 shodných popisků k 10 obrázkům. Což dělá zisk 4,8 anotací k jednomu obrázku.

Nejvyšší počet získaných anotací k jednomu obrázku bylo 7. Tento počet získali dva obrázky. Jeden z těchto obrázků lze vidět na 7.1. Získané anotace k tomu obrázku byly: Pes, ocas, moře, pláž, balón, písek a voda. Jak lze vypozorovat, jdou získat pomocí této hry široké informace o obrázcích.

To je velký počet získaných anotací, i přes relativně nízký počet zahraných her u jednoho obrázku. Díky systému, že se musí shodnout oba dva uživatelé, nebyly k žádnému obrázku přiřazena slova, která by se tam nevyskytovala. Správnost anotací byla ověřována pomocí databáze Imagenet, kde lze dohledat obrázky podle synsetů podstatných jmén.

Zajímavým faktem je, že v některých obrázcích se shodl o dost větší počet uživatelů, že je zde jeden předmět než jiný. Například na tomto obrázku 7.2 se shodly všechny dvojice uživatelů, že se zde nachází stan. Ale pouze jedna dvojice se shodla, že se zde nachází kamení. Obě tyto informace jsou pravdivé, ale jak lze posoudit, stan je na obrázku opravdu

Obrázek	počet anotací
1	5
2	5
3	4
4	7
5	2
6	5
7	7
8	4
9	5
10	4

Tabulka 7.1: Tabulka s počty anotací u obrázků

Obrázek 7.1: Obrázek s největším počtem anotací

dominantnější objekt než kamení. Podle počtu shodnutí se dá tak určovat relevantnost označení. Této relevantnosti označení lze například efektivně využít ve vyhledávání obrázků, kdy lze uživatelům podsouvat jako první obrázky, kde se daný předmět vyskytuje s větším počtem nalezení a je teda na tom to obrázku výraznější než ostatní objekty.

Tato hra se tedy ukázala jako vhodnou hrou k získávání anotací k obrázkům.

7.2 Tag Image Extreme!

V této hře bylo připraveno 10 obrázků. Na obrázcích vždy byl nějaký objekt v nějaké situaci. Zde byly výsledky horší než ve hře Tag Image!. Pouze 5 obrázků mělo shodnou anotaci a to pouze jednu. V této hře je velice těžké se shodnout s podstatným jménem i slovesem. I tyto anotace byly správné. Nízkému počtu shodnutí lze přičíst i faktu, že je těžké jednoznačně určit, jaká činnost se na obrázku provádí.

Zajímavým poznatkem je také fakt, že podobně jak u předešlé hry jsou některé činnosti jednoznačnější než jiné. Na jenom obrázku se shodli všechny páry hráčů, že zde objekt vykonává určitou činnost, zatímco u jiného obrázku se shodl jenom jeden pár. Obrázek s největším počtem shodnutí lze vidět zde 7.3, kdy se uživatelé jednoznačně shodli, že je zde plavající medvěd.

Obrázek 7.2: Obrázek s velkým počtem shodnutí jednoho objektu

Obrázek 7.3: Obrázek s největším počtem shodnutí jedné činnosti

Tato hra už není tak efektivní na získávání anotací k obrázkům jako ta předešlá, přesto jde pomocí ní získat o obrázku zajímavé informace, které předešlou hrou získat nelze.

7.3 Video Tag!

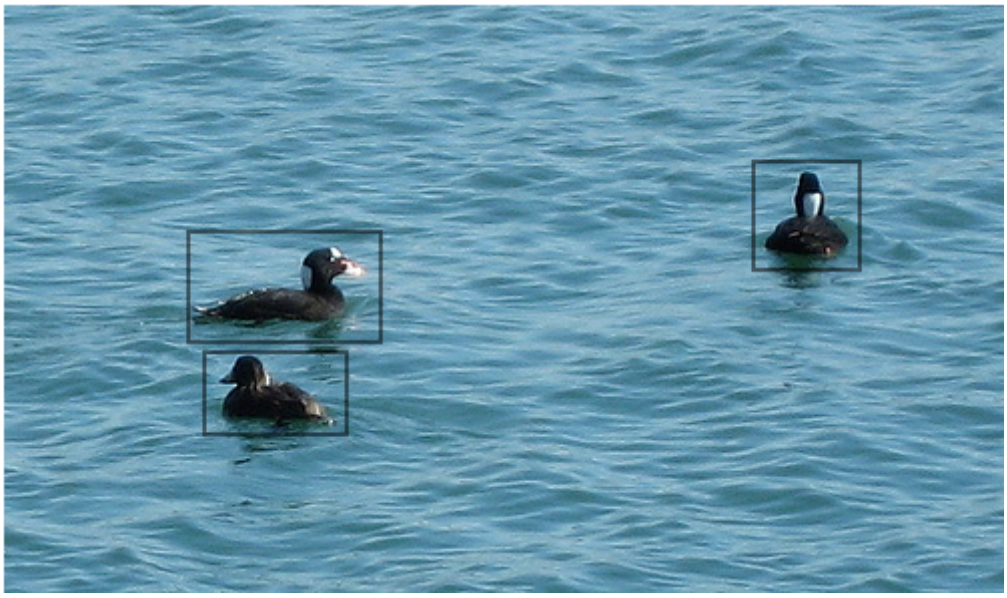
Pro tuto hru jsem nemusel připravovat žádné speciální data, tato hra je v získávání videí plně v režii uživatelů. Tuto hru hrálo 8 párů uživatelů. Pouze odkaz na videa, kdy druhý uživatel vybral správně činnost na daném videu, se ukládala. Z 8 her se získalo 14 videí.

Z toho 12 videí bylo opravdu vhodně zvolených a opravdu na nich byla zmíněná činnost. Problém s videi je většinou takový, kdy uživatelé vyberou video, kde není daná činnost ale je toto sloveso vyjádřeno jinak. Například textem ve videu nebo je toto slovo v textu přehrávané písně. O těchto videích pak lze polemizovat, jestli je vhodné, aby byly anotovány tímto slovesem.

Každopádně bilance 12-ti videí k 14-ti videím je dobrá(0,85% správnost videí), proto i tato hra je užitečná, protože lze díky ní získat kvalitně anotované videa.

7.4 Find Tag!

Stejně jako u všech ostatních obrázkových her, i zde jsem vybral 10 obrázků. Uživatelé měli za úkol hledat objekty skrývající se na obrázku. Všechny obrázky byly aspoň jednou shodně označeny dvojicemi hráčů. Na některých obrázcích se nacházelo objektů více, proto na některých obrázcích bylo třeba označit více předmětů. Jako třeba na tomto obrázku 7.4. Na tomto obrázku museli hráči označit 3 objekty.



Obrázek 7.4: Obrázek s třemi objekty k zakreslení

Vytvořil jsem průměry souřadnic všech čtverečků, které se v rámci hry vyhodnotili jako pravdivé. Tyto souřadnice jsem umístil do tabulky 7.2.

Obrázky jsem schválně vybral z těch obrázků Imagenetu, ke kterým existuje anotace bounding boxů, abych mohl vyhodnotit získané výsledky. Tabulku referenčních hodnot lze vidět v 7.3

Naměřené průměrné hodnoty jsem porovnal s referenčními hodnotami. Zjistil jsem jejich rozdíly a ty porovnal s rozměry obrázku(x-ové souřadnice s šířkou a y-ové souřadnice s výškou). Rozdíly mezi hodnotami v pixelech a v procentech jdou vidět v tabulce 7.4

Z vypočítaných hodnot lze vypozorovat, že pouze na jednom obrázku je odchylka větší než 10%. Lze se domnívat že tato chyba, byla způsobena obtížným obrázkem letadlové lodi, kdy se špatně vymezují její hranice. Odchylky ostatních obrázků jsou velice nízké, proto lze považovat tuto hru, jako vhodnou možnost jak získávat za pomocí hráčů kvalitní bounding boxy vymezující jednotlivé objekty nacházející se na obrázku.

Objekt	x_{min}	y_{min}	x_{max}	y_{max}
Tuba s léky	292,25	27	371,25	153,75
Křídlo	198,75	149,75	520,25	399,25
Ryba	232,25	25,75	341,5	436,5
Nádoba na mléko	231	10	434	220
	197	195	478	492
	13	30	286	494
Pavouk	114,14	109,64	311,28	285,14
Preclík	111	19	315,75	208,75
	159	194,75	374	384
Letadlová loď	263,25	229,25	497	378
Zebra	234,5	76,5	521,5	376,5
	30,5	91,5	313,5	380
Kachna	93,25	162,5	188,375	218,25
	89,75	127,375	194,125	187,875
	374,25	73,375	442	151
Žába	165,25	76	352,25	330

Tabulka 7.2: Tabulka s průměry souřadnic bounding boxů označených uživateli

Objekt	x_{min}	y_{min}	x_{max}	y_{max}
Tuba s léky	269	2	344	128
Křídlo	167	146	499	374
Ryba	243	7	323	428
Nádoba na mléko	234	11	417	211
	200	201	498	496
	0	41	290	496
Pavouk	116	114	290	268
Preclík	103	4	296	196
	154	186	359	374
Letadlová loď	0	80	431	374
Zebra	220	57	499	374
	0	58	293	360
Kachna	107	174	166	210
	92	118	183	162
	378	82	423	124
Žába	165	69	332	313

Tabulka 7.3: Tabulka s referenčními souřadnice stáhnuté z Imagenet

7.5 Dotazník

Všem lidem, kteří se účastnili testování hry, jsem poslal krátký dotazník. První čtyři otázky se týkali jednotlivých her, účastníci měli ohodnotit na stupnice od jedné do desíti, jak jim přišel konkrétní typ hry zábavný. Pátá otázka se týkala snadnosti používání celé aplikace. Na stejné stupnici měli účastníci ohodnotit, jak bylo pro ně jednoduché aplikaci používat.

Objekt	$\Delta x_{min}\%$	$\Delta y_{min}\%$	$\Delta x_{max}\%$	$\Delta y_{max}\%$
Tuba s léky	4,65	6,67	5,45	6,87
Křídlo	6,35	1	4,25	6,73
Ryba	3,17	3,93	5,46	1,75
Nádoba na mléko	2,85	1,91	3,35	3,06
	3,8	4,62	2,75	3,31
	4,65	2,3	2,85	2,96
Pavouk	0,37	1,16	4,26	4,57
Preclík	1,6	4,02	3,95	3,42
	1	2,35	3	2,68
Letadlová loď	52,65	39,8	13,2	1,07
Zebra	2,9	5,2	4,5	0,67
	6,1	8,93	4,1	5,33
Kachna	2,75	3,92	4,48	2,82
	0,45	3,2	2,23	8,83
	0,75	2,94	3,8	9,22
Žába	0,05	1,88	4,05	4,56

Tabulka 7.4: Tabulka s odchylkama bounding boxů

Následující otázka se ptala uživatelů, jestli by si chtěli v budoucnu tuto hru zahrát znovu. Otázka měla předvolené možnosti "Ano určitě", "Možná"a "Ne". Předposlední otázka zkoumala, zda uživatelé používají tutoriál, před tím než začnou hrát. V tabulce 7.5 lze vidět průměr odpovědí na prvních pět otázek.

Otázka	**Hodnocení**
Tag Image!	6,8
Tag Image Extreme!	5,5
Video Tag!	8,5
Find Tag!	6,75
Ovladatelnost	7

Tabulka 7.5: Tabulka s výsledky dotazníku

Nejoblíbenější hrou se tedy stal Video Tag!. Naopak nejneoblíbenější hrou je Tag Image Extreme!. Uživatelé měli dle ankety pouze menší problémy hru ovládat. Zajímavé také je, že pouze 2 uživatelé použili tutoriál před tím, než začali hru hrát.

S výsledků usuzuji, že se hry uživatelům líbili a mohli by být použity i pro širší sběr dat.

Kapitola 8

Závěr

Práce se zabývá crowdsourcingovými hrami pro anotování videa a obrazu. Čtenáři práce byli seznámeni s tím, co to crowdsourcing je a jak jej lze použít ve formě hry pro získávání užitečných anotací. Čtenáři také byli seznámeni s dalšími crowdsourcingovými hrami a projekty jako jsou ESP game, Peekaboom, Steve Museum a Guess What?.

Následně byly navrženy nové crowdsourcingové hry vhodné pro získávání anotací k obrazu a videu. Byla navržena hra, kdy hráči popisují objekty na obrázku pro jednoho i více hráčů. Další takovou hrou byla hra, kde hráči popisují činnosti, prováděné objekty na obrázku. Pro získávání anotace k videu, byla navržena hra, kde dva hráči mají za úkol hledat videa na youtube, na kterých se děje určitá činnost. Pro anotaci videa byla navržena i hra pro jednoho hráče, která pracuje s nalezenými videi na youtube. Dále byla navržena hra, ve které dva hráči mají označit objekt, nalézající se na obrázku.

Tyto hry získávají anotace v podobném tvaru jako v projektu Imagenet. Obrázky a videa se spojují se synsety z korpusu WordNet. V případě poslední navržené hry se obrázky se synsety spojují s odpovídajícími bounding boxy.

Navržené hry byly implementovány. Jako nejvhodnější platforma pro crowdsourcingovou hru byla zvolena webová aplikace. Hry se skládají z klientské a serverové části. Klientská část byla vytvořena pomocí javascriptového frameworku Angular a frontend frameworku Bootstrap. Serverová část byla vytvořena v Pythonu ve frameworku Django. V aplikaci byla použita databáze PostgreSQL. Obrázky pro anotační hry byly získány ze serveru Imagenet. Veškerá slova vážící se k anotovaným obrázkům a videům jsou pro uživatele vybírána z korpusu Wordnet, což zvyšuje užitečnost získaných dat. Propojení Wordnetu a Pythonu bylo docíleno pomocí lexikálního nástroje NLTK.

Implementace byla v práci detailně popsána. Jsou zde zmíněny direktivy a služby frameworku Angular, které byly vytvořeny pro tuto aplikaci. Ze serverové části byly popsány užitečné knihovny. V práci je obsažen i popis jednotlivých stránek aplikace a způsob komunikace klienta se serverem.

Webová aplikace byla spuštěna na školním serveru knot08. Pro spuštění aplikace bylo potřeba nainstalovat Postgresql, Python a jeho balíčky – NLTK, Virtualenv, Psycopg2 a Django.

Hry byly otestovány 16 uživateli. Bylo získáno v průměru 4,8 anotací k jednomu obrázku, které měly stoprocentní správnost. Hra Tag Image Extreme! už měla výsledky horší, při testování se dokázali uživatelé shodnout pouze u poloviny obrázků, na druhou stranu takto získané anotace byly také správné. U hry Tag Video! bylo zjištěno, že vytváří anotovaná videa s 85% správností. Bounding boxy získány pomocí hry Find Tag! byly z 90% správné. U správných bounding boxů byla maximální nepřesnost 10% pixelů vzhledem k rozměrům

obrázku.

Z těchto informací vyplynulo, že hry Tag Image!, Tag Video! a Find Tag! jsou vhodné hry pro získávání anotací. U hry Tag Image Extreme! by se o tom dalo polemizovat, už i kvůli menší oblibě u hráčů. Většině testovaných uživatelů se hry líbily, proto existuje šance, že by si tyto hry mohly vybudovat velkou uživatelskou základnu, která by mohla vytvořit velkou databázi anotovaných obrázků a videí vhodnou pro další výzkum.

Literatura

[1] Angular-UI.
URL http://angular-ui.github.io/

[2] Bootstrap.
URL http://getbootstrap.com/

[3] Middleware.
URL https://docs.djangoproject.com/en/dev/topics/http/middleware/

[4] MySQL.
URL http://www.mysql.com/

[5] PostgreSQL.
URL http://www.postgresql.org/

[6] Scenarios.
URL http://www.usability.gov/how-to-and-tools/methods/scenarios.html

[7] An intro to usability testing by Amberlight Partners. jun 2013.
URL http://www.uxcolombo.org/blog/2013/06/an-intro-to-usability-testing-by-amberlight-partners/

[8] Barth, A.; Jackson, C.; Mitchell, J. C.: Robust defenses for cross-site request forgery. In *Proceedings of the 15th ACM conference on Computer and communications security*, ACM, 2008, s. 75–88.

[9] Bird, S.: NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, Association for Computational Linguistics, 2006, s. 69–72.

[10] Chaffer, J.: *Learning jQuery*. Packt Publishing Ltd, 2013.

[11] Chun, S.; Cherry, R.; Hiwiller, D.; aj.: Steve. museum: an ongoing experiment in social tagging, folksonomy, and museums. *Museums and the Web 2006*, 2006.

[12] Deng, J.; Dong, W.; Socher, R.; aj.: Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, IEEE, 2009, s. 248–255.

[13] Domingos, P.: A few useful things to know about machine learning. *Communications of the ACM*, ročník 55, č. 10, 2012: s. 78–87.

[14] Farreres, X.; Rigau, G.; Rodriguez, H.: Using wordnet for building wordnets. In *Proceedings of COLING-ACL Workshop on Usage of WordNet in Natural Language Processing Systems*, 1998, s. 65–72.

[15] Flanagan, D.: *JavaScript: the definitive guide*. "O'Reilly Media, Inc.", 2002.

[16] Forcier, J.; Bissex, P.; Chun, W.: *Python web development with Django*. Addison-Wesley Professional, 2008.

[17] Greengard, S.: Following the crowd. *Communications of the ACM*, ročník 54, č. 2, 2011: s. 20–22.

[18] Itamar, S.-T.: An Introduction to the Twisted Networking Framework. Leden 2004. URL http://www.onlamp.com/pub/a/python/2004/01/15/twisted_intro.html

[19] Javidi, B.: *Image recognition and classification: algorithms, systems, and applications*. CRC Press, 2002.

[20] Kouznetsova, S.: may 2013. URL http://svknyc.com/journal/2013/05/why-is-flash-bad-for-user-experience/

[21] Kozlowski, P.; Darwin, P. B.: *AngularJS Web Application Development*. Packt Publ, 2013, iSBN 978-1-78216-182-0.

[22] Lie, H. W.; Bos, B.: *Cascading style sheets*. Addison-Wesley, 1997.

[23] Pilgrim, M.: Dive into HTML5. *Dive Into HTML5*, 2010.

[24] Purer, K.: PHP vs. Python vs. Ruby–The web scripting language shootout. 2009.

[25] Riek, L. D.; O'connor, M. F.; Robinson, P.: Guess what? a game for affective annotation of video using crowd sourcing. In *Affective Computing and Intelligent Interaction*, Springer, 2011, s. 277–285.

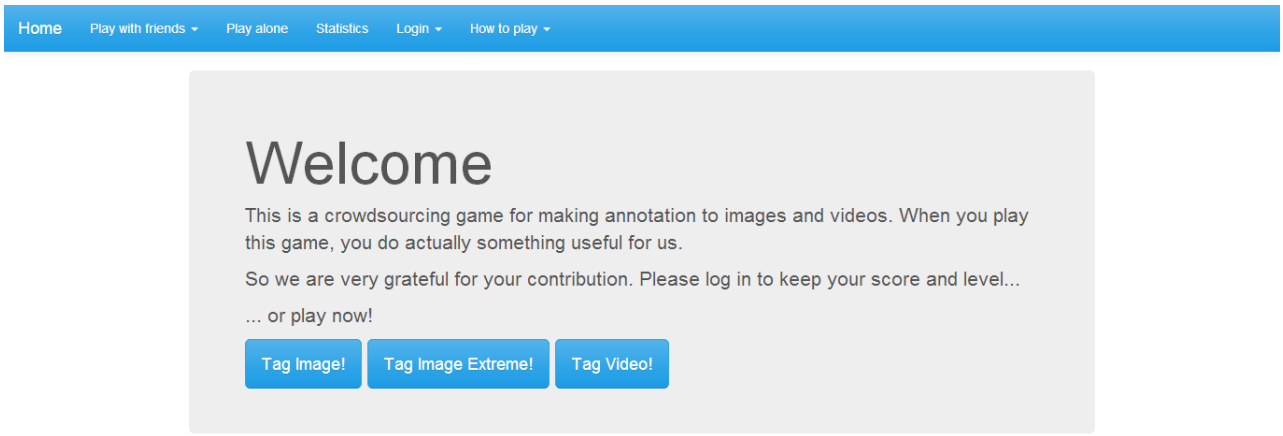
[26] Ronacher, A.: Flask. 2010. URL http://flask.pocoo.org/

[27] Von Ahn, L.; Dabbish, L.: Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, 2004, s. 319–326.

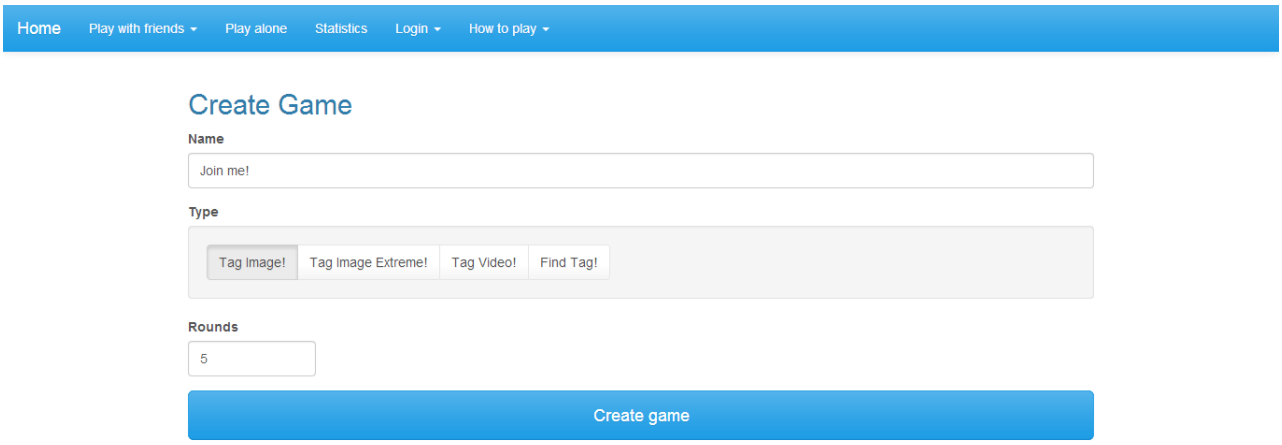
[28] Von Ahn, L.; Liu, R.; Blum, M.: Peekaboom: a game for locating objects in images. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, ACM, 2006, s. 55–64.

Příloha A

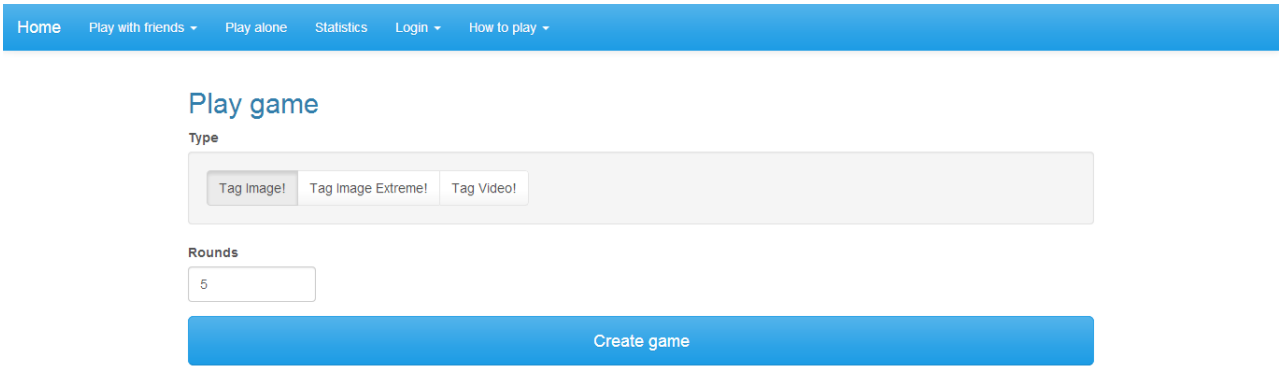
Screenshoty



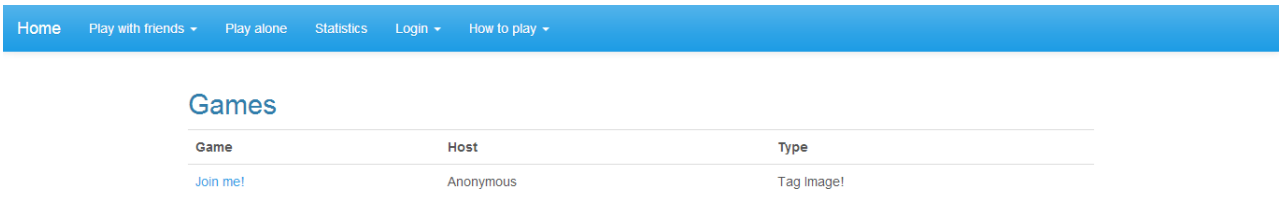
Obrázek A.1: Úvodní strana



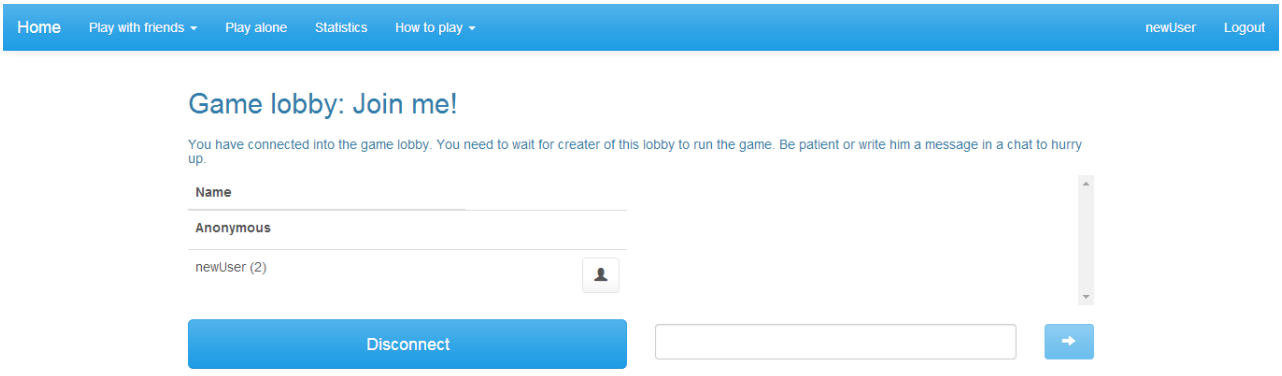
Obrázek A.2: Vytvoření hry – pro více hráčů



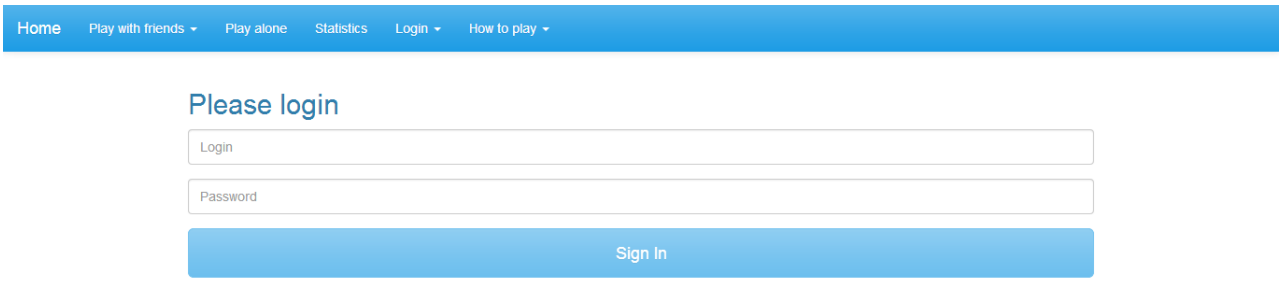
Obrázek A.3: Vytvoření hry - pro jednoho hráče



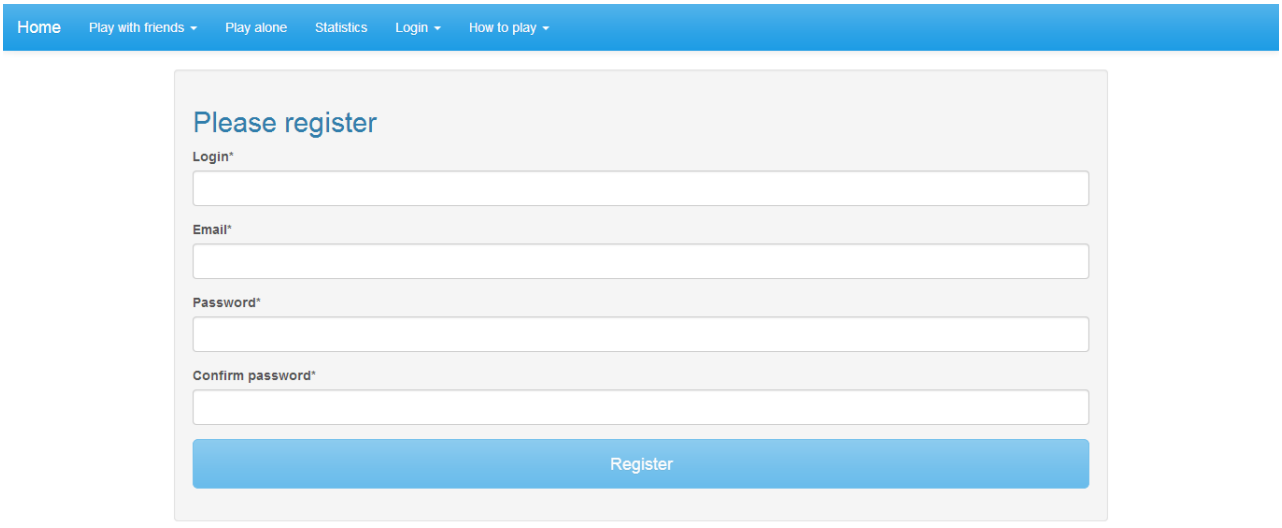
Obrázek A.4: Hledání her



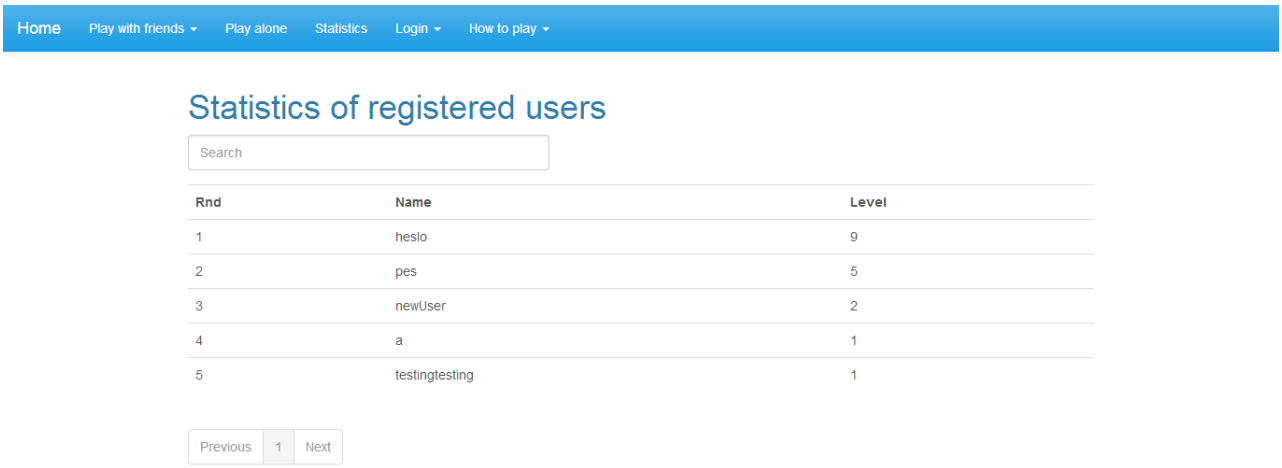
Obrázek A.5: Herní loby



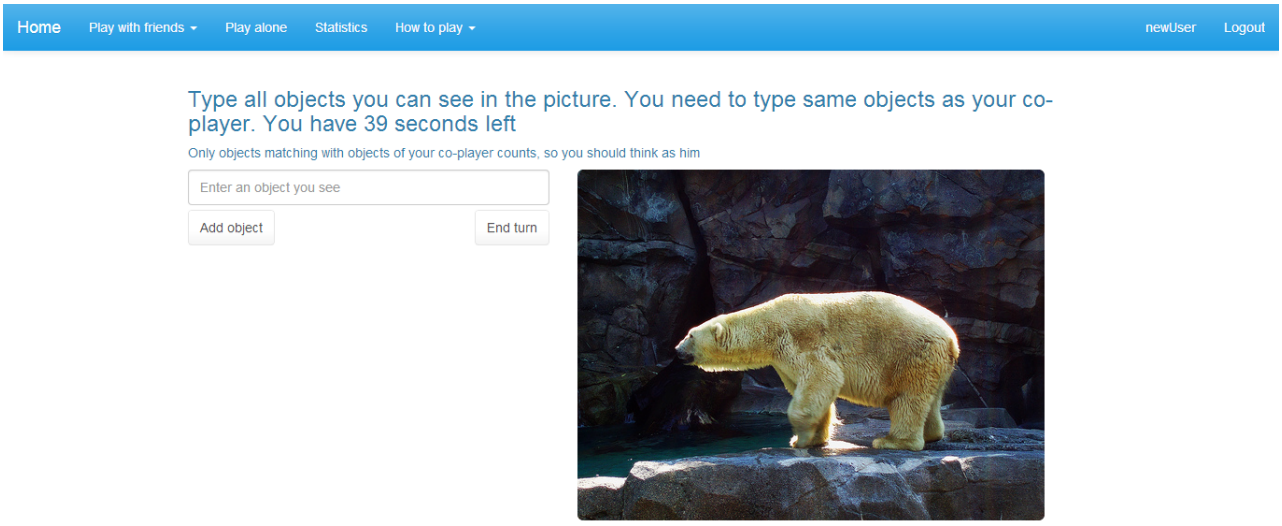
Obrázek A.6: Přihlášení



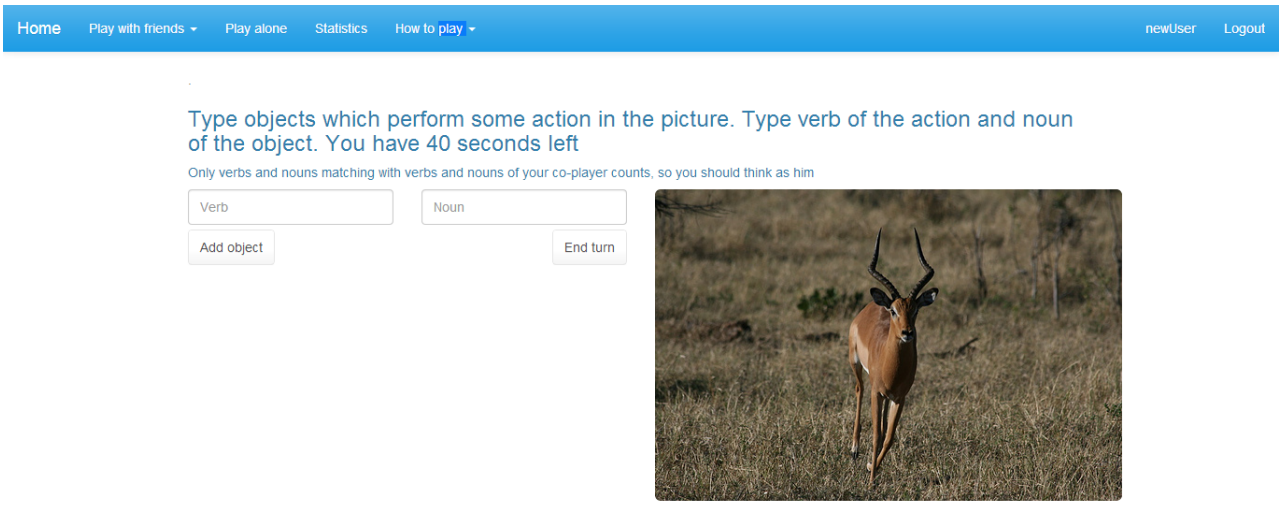
Obrázek A.7: Registrace



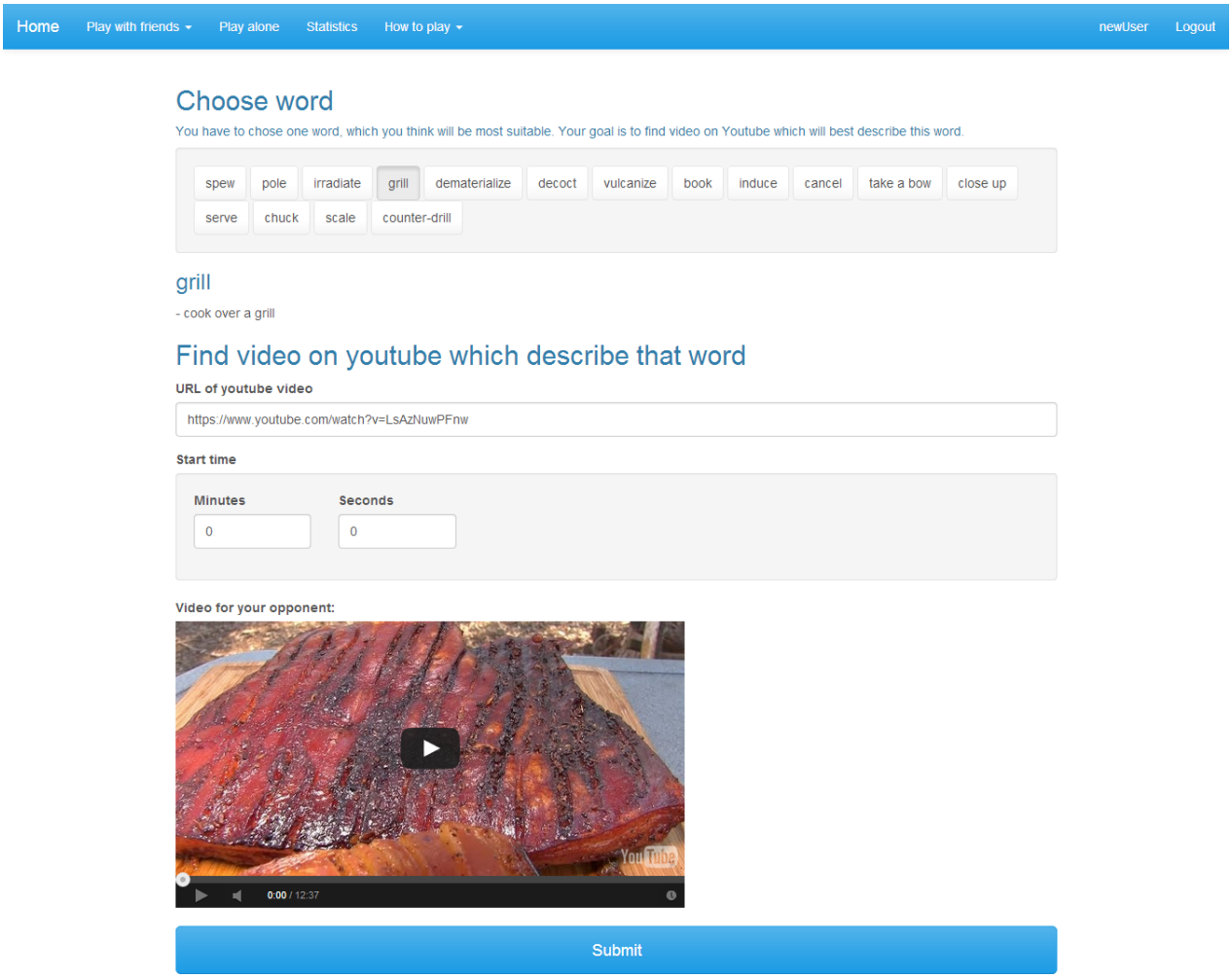
Obrázek A.8: Statistika



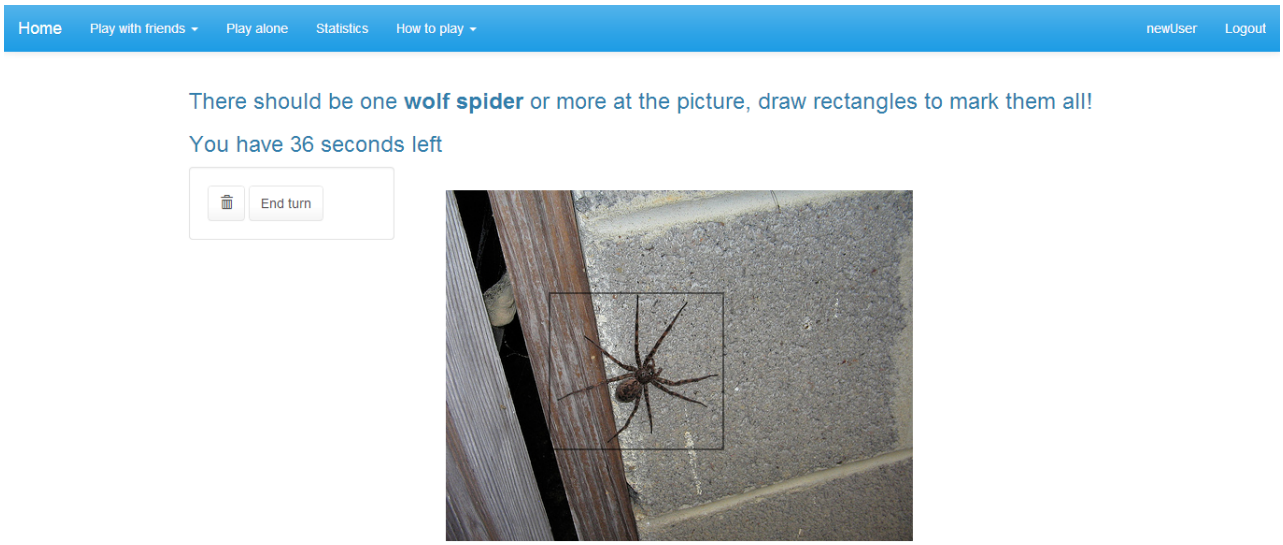
Obrázek A.9: Tag Image!



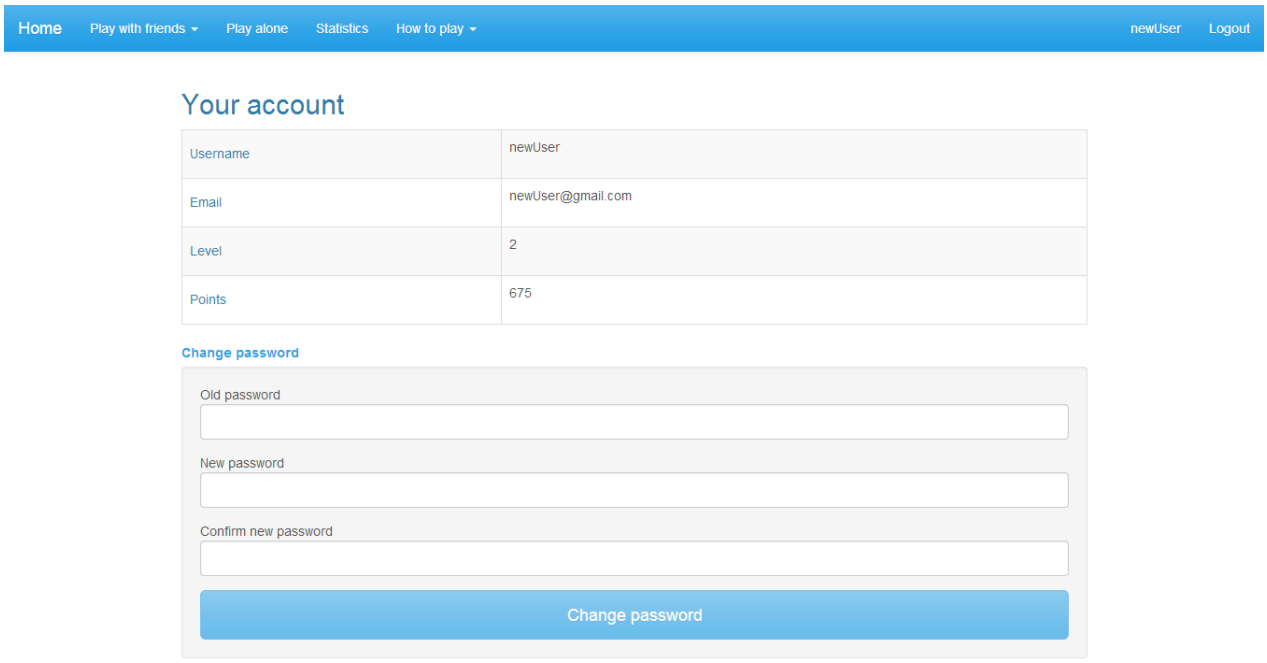
Obrázek A.10: Tag Image Extreme!



Obrázek A.11: Tag Video!



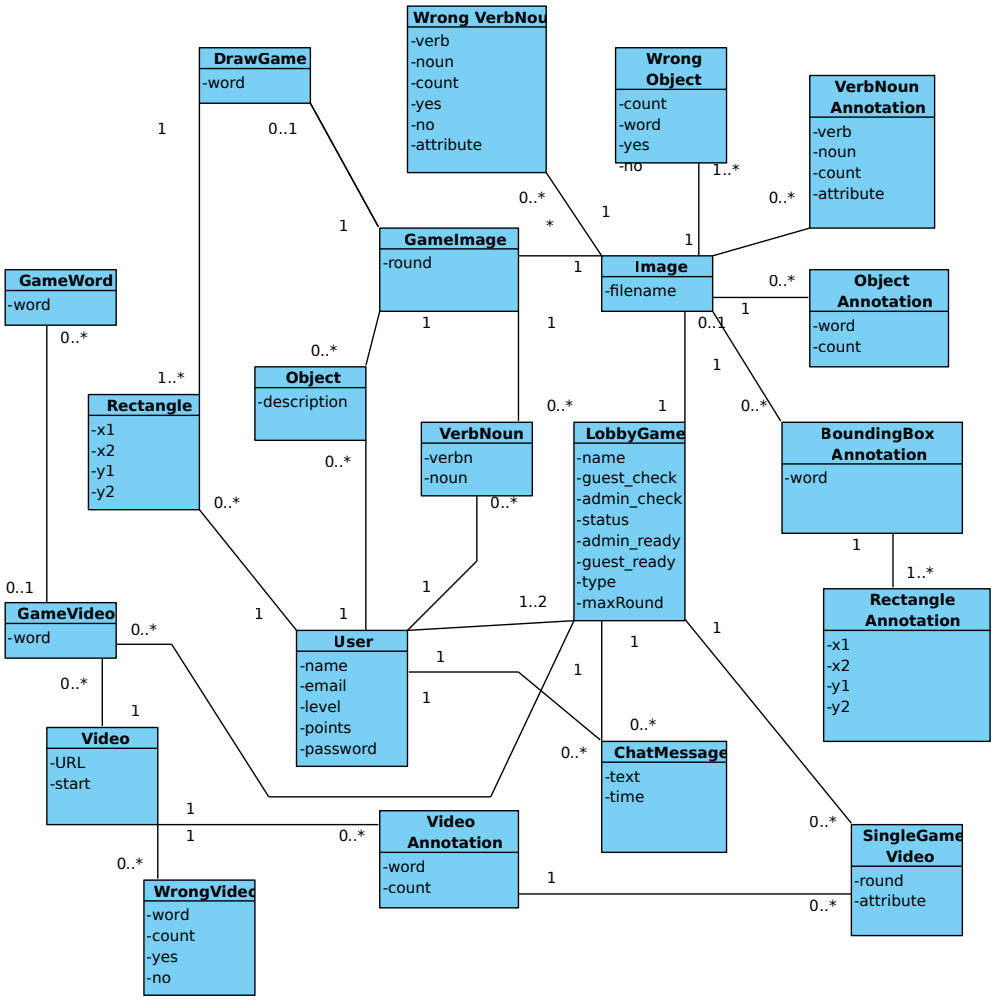
Obrázek A.12: Find Tag!



Obrázek A.13: Uživatelské detaily

Příloha B

ER diagram



Obrázek B.1: ER diagram aplikace

Příloha C

Obsah CD

Cesta	Popis
`/src/`	Zdrojové soubory webové aplikace
`/tex/`	Zdrojové texty této práce
`/projekt.pdf`	PDF soubor s touto prací
`/manual.pdf`	Manuál pro spuštění programu
`/plakat.png`	Plakát k diplomové práci