VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

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

WEBOVÁ PLATFORMA PRE TVORBU HIER

WEB PLATFORM FOR GAME DEVELOPMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE FILIP GULÁN
AUTHOR

VEDOUCÍ PRÁCE Ing. RADEK BURGET, Ph.D.
SUPERVISOR

BRNO 2016

Zadání bakalářské práce/18549/2015/xgulan00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Gulán Filip**
Obor: Informační technologie
Téma: **Webová platforma pro tvorbu her**
Web Platform for Game Development
Kategorie: Web

Pokyny:

1. Seznamte se s technologiemi HTML5 a JavaScript se zaměřením na tvorbu online her.
2. Prostudujte běžné funkční prvky existujících her jako např. správa hráčů, tabulka skóre, odměn, apod.
3. Navrhněte architekturu webového portálu umožňujícího vývojářům tvořit nové hry s využitím opakovaně použitelných komponent.
4. Implementujte navržený portál. Po dohodě s vedoucím implementujte zvolenou množinu komponent.
5. Implementujte jednoduchou demonstrační hru s využitím vytvořeného portálu.
6. Proveďte testování výsledného řešení a zhodnoťte dosažené výsledky.

Literatura:

- Gutmans, A., Rethans, D., Bakken, S.: Mistrovství v PHP 5, Computer Press, 2012
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3

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

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

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

Vedoucí: **Burget Radek, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

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

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Cieľom tejto práce je návrh a realizácia modernej webovej platformy pre tvorbu hier. Webová platforma je určená, ako pre vývojárov HTML5/Javascript hier, tak aj pre hráčov týchto hier. Súčasťou práce je okrem webového portálu aj API, ktoré sprístupňuje služby webového portálu vývojárom hier. Táto platforma je na serverovej strane postavená na skriptovacom jazyku PHP za použitia rámca Nette. Na klientskej strane je využité HTML, CSS, Javascript a pre účely responzívneho dizajnu je využívaný frontendový rámec Bootstrap. Hra, ktorá demonštruje API, je napísaná v jazyku Javascript s využitím rámca Phaser.

Abstract

The aim of this work is to design and implement modern web platform for creating games. The web platform is designed both for HTML5 / Javascript games developers and for players of these games. The work also includes a web portal and API that makes services of Web portal accessible for games developers. On server-side this platform is based on scripting language PHP using Nette framework. On the client side is used HTML, CSS, Javascript, and for the purpose of responsive design is used front-end framework Bootstrap. The game that demonstrates the API is written in JavaScript using the Phaser framework.

Kľúčové slová

webová platforma, API, HTML5 hra, Nette, Bootstrap, Phaser

Keywords

web platform, API, HTML5 game, Nette, Bootstrap, Phaser

Citácia

GULÁN, Filip. *Webová platforma pre tvorbu hier.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Burget Radek.

Webová platforma pre tvorbu hier

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Radka Burgeta, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

. . . . . .
Filip Gulán
16. mája 2016

Poďakovanie

Týmto by som sa chcel poďakovať vedúcemu práce pánovi Ing. Radku Burgetovi, Ph.D za odborné vedenie a cenné rady, ktoré mi pomohly pri tvorbe tejto práce. Zároveň ďakujem všetkým, ktorý ma podporovali behom štúdia.

© Filip Gulán, 2016.
Táto práca vznikla ako školské dielo na FIT VUT v Brně. Práca je chránená autorským zákonom a jej využitie bez poskytnutia oprávnenia autorom je nezákonné, s výnimkou zákonne definovaných prípadov.

Obsah

1 Úvod **3**

2 Analýza existujúcich riešení **4**
2.1 Steam 4
2.2 Google Play Game Services 5
2.3 Apple Game Center 6
2.4 Facebook Game Services 6
2.5 Clay.io 7
2.6 Ostatné 7

3 Návrh riešenia **8**
3.1 Analýza požiadaviek na komponenty 8
3.2 Návrh webovej platformy 9
3.3 Návrh databáze 16
3.4 Návrh aplikačného rozhrania 17
3.5 Návrh Hry 18

4 Použité technológie **20**
4.1 Serverová časť 20
4.2 Databázová časť 21
4.3 Klientská časť 22
4.4 Aplikačné rozhranie 23
4.5 Hra 25

5 Implementácia **26**
5.1 Implementácia webovej platformy 26
5.2 Implementácia aplikačného rozhrania 28
5.3 Implementácia demonštračnej hry 30

6 Testovanie **31**
6.1 Návrh testovania 31
6.2 Priebeh a výsledky testovania 32

7 Záver **33**

Literatúra **34**

Prílohy **35**
Zoznam príloh 36

A Obsah CD **37**

B Návod na inštaláciu **38**
B.1 Inštalácia 38

C Dokumentácia k API **39**
C.1 Funkcie 39
C.2 Príklady 41

D Ukážky z implementácie **42**

Kapitola 1

Úvod

V minulosti v oblasti herného priemyslu existovalo niekoľko webových platforiem, ktoré sprostredkovali rôzne služby pre vývojárov hier. Za zmienku stoja napríklad tabuľky najlepších hráčov, správa odmien, pokročilé štatistiky prístupov, vzdialené úložisko dát a iné. Tieto služby boli orientované iba na webové hry, vytvorené technológiou Flash. Keďže samotná technológia Flash pomaly upadá, tak aj tieto služby pomaly zanikajú, alebo už zanikli. Takýmto príkladom môže byť líder sveta Flashových hier, menom Mochimedia. Mobilné platformy, ako Android a iOS, alebo platforma Steam, začali sami oficiálne podporovať tieto služby už pri svojom vzniku. S rozmachom moderných webových technológií a ich následným rozšírením na chytré mobilné zariadenia, vznikla možnosť tvorby webových hier, ktoré by fungovali v každom modernom webovom prehliadači, zahrňujúc nielen prehliadače v desktopových systémoch, ale aj v mobilných a tabletových. Príchodom týchto HTML5/Javascript hier neexistovala žiadna webová stránka ponúkajúca podobné vyššie zmienené služby pre vývojárov.

Príchodom webového portálu Clay.io v roku 2012 sa na vývojárov Javascriptových hier dočasne usmialo šťastie. Clay.io ponúkalo rôzne služby pre herných vývojárov, bez nutnosti mať státisíce prístupov do hry, alebo mať hru špičkovej grafickej úrovni. Služby boli dostupné pre všetkých, bez rozdielu. Avšak, začiatkom roka 2015 sa Clay.io začalo orientovať iným smerom a z tejto webovej služby sa stal viac portál pre hráčov, ako pre vývojárov. Už viac nebolo možné integrovať API do hociktorej hry, a všetky hry, ktoré chceli byť umiestnené na portáli Clay.io a využívať jeho služby boli vyberané podľa prísnych kvalitatívnych kritérií. Taktiež hry využívajúce API bolo možné hrať iba na webovej stránke Clay.io a vývojár nesmel distribuovať hru na iné webové portály. Týmto sa Clay.io uzavrelo a zaradilo sa vedľa iných svetových distribútorov HTML5 hier, ako holandská Boostermedia a Spilgames, alebo nemecký Softgames.

Táto bakalárska práca vznikla z dôvodu neexistencie otvorenej, modernej webovej platformy pre tvorbu hier pre všetkých bez rozdielu, ktorá by svoje služby ponúkala zadarmo. Táto webová platforma je orientovaná nielen na vývojárov Javascriptových hier, ktorý prostredníctvom API majú možnosť využívať služby platformy, ale aj na hráčov týchto vytvorených hier. Hráči by mali byť schopný hrať hry na akejkoľvek platforme, či už desktopových osobných počítačoch, alebo moderných chytrých telefónoch. Z toho dôvody by mala minimálne časť webového portálu určená hráčom byť do veľkej miery responzívna.

Nasledujúca kapitola sa venuje analýze už existujúcich riešení pre iné platformy. Po nej nasleduje kapitola venujúca sa návrhu riešenia. V ďalšej kapitole sú zhrnuté technológie, ktoré boli použité v rámci tejto práce. Implementácia je zhrnutá v rovnako pomenovanej kapitole. A na záver, návrh a priebeh testovania je zhrnutý v kapitole Testovanie.

Kapitola 2

Analýza existujúcich riešení

V tejto kapitole budú predstavené konkurenčné, existujúce a predovšetkým známejšie riešenia určené zväčša pre iné platformy, ako je platforma, pre ktorú je riešenie tejto práce určené. Určite existuje oveľa viac alternatív, ale vymenúvať a rozobrať všetky nie je účelom tejto práce.

2.1 Steam

Steam je herná platforma od spoločnosti Valve Corporation určená predovšetkým k digitálnej distribúcií hier a vytvoreniu komunity hráčov týchto hier. Steam klient je podporovaný na rôznych platformách. Na počiatku podporoval iba platformu Windows, neskôr svoje pôsobenie rozšíril na platformu Mac a nakoniec na Linuxové distribúcie. Pochopiteľne nie všetky hry nachádzajúce sa na Steam platforme sú hrateľné vo všetkých operačných systémoch. Momentálne samotná platforma obsahuje viac ako 3500 hier a teší sa početnej komunite hráčov. [6]

Pre možnosť distribúcie hry na Steam je vývojár nútený si zakúpiť prístup k službe Steam Greenlight, ktorý stojí v dobe vzniku tejto práce 90 EUR. Poplatok je jednorazový a vývojár nie je obmedzený počtom publikovaných hier v tejto službe. Po publikovaní hry v službe Steam Greenlight sa hra avšak nenachádza ešte na samotnej platforme. Užívatelia vďaka tejto službe môžu hlasovať za hru a hra sa na platformu Steam dostáva až po získaní určitého počtu hlasov. Je teda na samotnom vývojárovi vynaložiť úsilie, aby bola hra čo najlepšie spropagovaná a všimol si ju a samozrejme aj hlasoval čo možno najväčší počet oslovených ľudí. [4]

Steam pre vývojárov hier, ktorých hry boli schválené komunitou Steam Greenlight pre možnosť distribúcie hry na platforme, ponúka svoje Steamworks API. Steamworks API dovoľuje vývojárom hier integrovať rôzne funkcie ako štatistiky, odmeny, autentifikácia, tvorba zápasov, peer-to-peer networking, Steam cloud úložisko, Valve anticheat[1], hlasový chat a v neposlednej rade aj DRM[2]. Všetky tieto funkcie je možné dohľadať na oficiálnej stránke Steamworks API. [7]

[1] Software, ktorý slúži ako prevencia proti neférovému získaniu výhody v online hre pomocou softwaru tretích strán

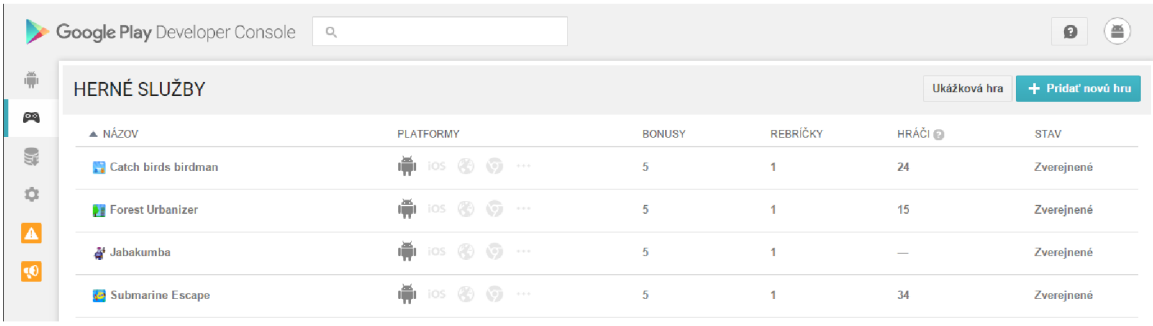
[2] Digital rights management - metódy, ktorých účelom je kontrolovať používanie obsahu digitálnych médií

Obr. 2.1: Ukážka steam klienta na Windows

2.2 Google Play Game Services

Po predstavení operačného systému Android, vyvinul Google svoj internetový obchod Google Play, ktorý ponúka nielen aplikácie a hry pre Android, ale po novom aj knihy, hudbu a filmy. Vývojári môžu momentálne získať prístup na publikovanie aplikácií v tomto obchode za poplatok 20USD, ktorý sa platí jednorazovo, pričom reálne nie sú limitovaný počtom aplikácií a ani kvalitou výslednej publikovanej aplikácie.

Okrem toho, Google dáva možnosť vývojárom hier integrovať do ich diel svoje Play Game Services API. Zprvu bolo API limitované iba na Android hry, v súčasnosti však podporuje už aj webové aplikácie. Pred samotným použitím API si vývojár musí najprv definovať hru v Google Play Developer Console a poprípade ďalšie doplnkové funkcie, ktoré chce integrovať a využívať. API ponúka možnosti použitia odmien, skóre tabuliek, multiplayeru, uloženia dát v cloude, udalosti, úlohy a iné. Ďalšie informácie sú k nájdeniu v oficiálnej dokumentácií. [3]



Obr. 2.2: Ukážka Google Play Developer Console - zoznam existujúcich hier

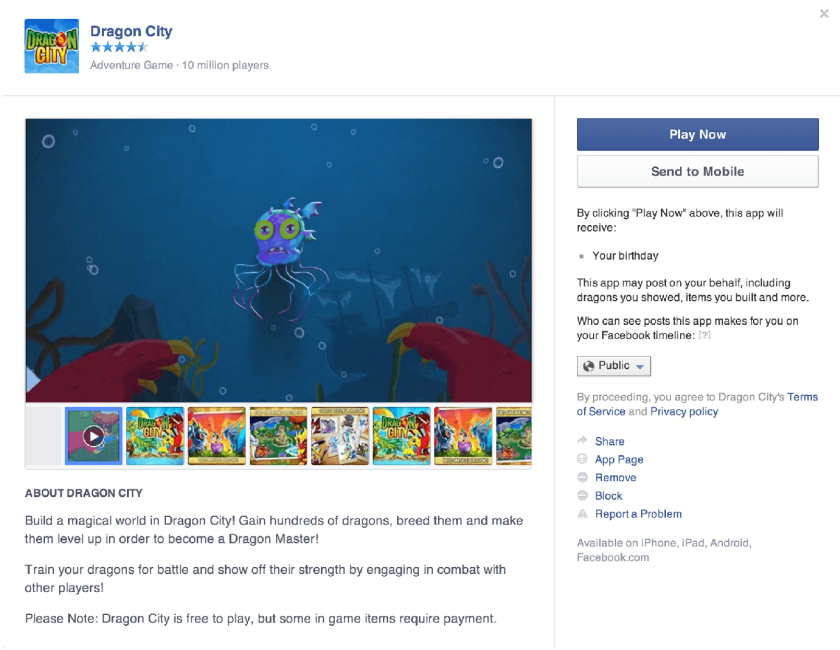
2.3 Apple Game Center

Rovnako ako Google má svoj internetový obchod s Android aplikáciami, tak aj Apple má svoj App Store, s aplikáciami pre svoje zariadenia. Na rozdiel od Google Play, vývojár pre prístup do App Store musí platiť ročne buď 99 USD za Apple Developer Program, alebo 299 USD za Apple Developer Enterprise Program. Viac informácií, ako aj rozdiely v týchto programoch, je možné nájsť na oficiálnej stránke Apple.

Apple tiež ponúka svoje Game Center API. API je integrovateľné do iOS hier a taktiež do OS X. Game Center umožňuje integrovať skóre tabuľky, odmeny, pozývanie priateľov do hry a spúšťanie multiplayer hier pomocou systému automatického vytvárania hier[3]. [1]

2.4 Facebook Game Services

Za zmienku určite stojí aj najväčšia a najpopulárnejšia sociálna sieť Facebook, ktorá od svojho vzniku v roku 2004, pridala nespočetné množstvo vylepšení, medzi ktorými sa objavili aj hry. V tomto momente to už zďaleka nie je len sociálna sieť, ako by sa mohlo na prvý pohľad zdať. Vývojári majú tak možnosť na tejto sociálnej sieti publikovať svoje hry a dokonca integrovať špeciálne funkcie ponúkané vývojárom vďaka Facebook Game Services API. API je integrovateľné do širokého spektra platforiem a enginov od Flash a HTML5 hostovaných na Facebooku, ktoré sú možné nájsť v Facebook App Store, až po mobilné hry platformy Android a iOS. API štandardne ponúka odmeny, skóre tabuľky a štatistiky + ďalšie rozširujúce funkcie sociálnej siete zahrnjúc zdieľanie na nástenke profilu užívateľa, alebo notifikácie. Záznam hry nachádzajúcej sa v Facebook App Store, môžete vidieť na obrázku 2.3. [2]



Obr. 2.3: Záznam hry v Facebook App Store

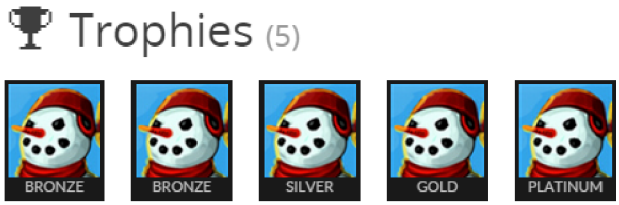
[3]Matchmaking

2.5 Clay.io

Clay.io je webový projekt, zameraný na Javascript/HTML5 hry, ktorý vytvoril v roku 2012 Austin Hallock. Z prvu bol tento projekt rovnomerne orientovaný, ako na vývojárov hier, tak aj na hráčov týchto hier. Hry mohli byť na platforme hostované zadarmo. Platforma neobsahovala žiadne kvalitatívne obmedzenia čo sa týka uverejňovania hier a jej služby boli zadarmo. Bolo možné do hry integrovať vlastné Javascriptové API, ktoré sprístupňovalo skóre tabuľky, odmeny, cloud úložisko dát, cross promotion[4], reklamy tretích strán. Okrem toho na základe už vložených informácií Clay.io dokázalo generovať rôzne konfiguračné súbory pre webové obchody ako Chrome Web Store, alebo Firefox Marketplace. Začiatkom roka 2015 sa však situácia zmenila a Clay.io sa uzavrelo. Začalo sa orientovať na vyššiu kvalitu hier a k API už mohli pristupovať iba oprávnení schválení vývojári. Publikovať hry s takýmto API bolo po novom tiež možné iba na webe Clay.io a v špeciálnej Clay.io aplikácií, ktorá je určená pre Android, poprípade na ďalších weboch, alebo v aplikáciách, avšak iba na tých pár vybraných, s ktorými Clay.io spolupracuje. [12]

2.6 Ostatné

Pre webové hry postavené na technológií Flash, existovalo a stále ešte existuje hneď niekoľko herných portálov, určených predovšetkým pre hráčov, ktoré ale taktiež ponúkajú vývojárom možnosť integrovať rôzne služby konkrétneho webového portálu do hry. Príkladom môže byť zaniknutý Mochimedia, alebo stále existujúce herné webové portály Newgrounds.com, Gamejolt.com, alebo Kongregate.com. Nevýhodou týchto portálov je to, že použitie ich API, je často limitované iba pre použitie na danom webovom portále. Všetky zmienené portály ponúkali, alebo stále ponúkajú obdobné funkcie API, ako tomu bolo v predchádzajúcich častiach.



Obr. 2.4: Ukážka odmien na stránke Gamejolt.com

Boli vymenované riešenia, asi pre takmer všetky webové, desktopové a mobilné platformy. Je ale nutné podotknúť, že herné konzoly tiež nezostávajú pozadu a taktiež obsahujú väčšinu z vymenúvaných komponent, čo je samozrejme prirodzené, keďže tieto komponenty často pridávajú hre na atraktivite. Dobrým zástupcom je Xbox, alebo Playstation. Avšak možnosti stredných, vôbec neuvažujúc malých, herných štúdií a vývojárov, ktorý chcú na týchto platformách publikovať sú dosť obmedzené, keďže sú akceptované poväčšine iba veľké AAA tituly[5], a keď Indie hry[6], tak naozaj veľmi kvalitné, alebo populárne.

[4]Forma propagácie, kedy je zákazníkom jednoho preduktu predstavovaný podobne zameraný produkt

[5]V hernom premysle označuje herné tituly, s najväčším rozpočtom, levelom propagácie, alebo s najvyšším hodnotením od kritikov

[6]Nezávislá hra vyvíjaná bez finančnej podpory vydavateľa

Kapitola 3

Návrh riešenia

Táto kapitola popisuje návrh riešenia tejto bakalárskej práce. Najprv pojednáva o analýze požiadaviek na komponenty, ktoré sú zahrnuté v API, a na celkovú štruktúru webovej platformy. Kapitola ďalej popisuje návrh webovej platformy z hľadiska funkcionality a následne z hľadiska grafického užívateľského rozhrania. Hneď nato nasleduje návrh databáze a je predstavený jej konceptuálny model. Na konci kapitoly je predstavený návrh samotného API a návrh jednoduchej HTML5 hry, ktorá má za úlohu demonštrovať funkcie tohto API.

3.1 Analýza požiadaviek na komponenty

Požiadavky na komponenty čiastočne vychádzajú z predchádzajúcej kapitoly, o Analýze už existujúcich riešení. Boli vybrané komponenty z daných existujúcich riešení také, ktoré sa najviac opakovali v daných službách, teda boli v určitom zmysle žiadané, a ktoré boli reálne použiteľné na tvorenej webovej platforme. Okrem toho bol zostavený orientačný, doplňujúci dotazník, cielený práve na vývojárov hier a špeciálne na vývojárov webových html5 hier. Dáta boli zberané na fórach[1], kde sa združujú vývojári takýchto hier a v špeciálnych skupinách sociálnych sietí[2], ktoré obsahujú takto odborovo orientovaných ľudí. V dotazníku odpovedalo 47 ľudí. Títo ľudia mali vybrať 3 komponenty, ktoré pokladajú pre seba, ako vývojárov za najužitočnejšie. Pritom bolo jedno z akého hľadiska, či z hľadiska toho, že komponenta má obohacovať hru napríklad o tabuľku skóre, či odmeny a tak jej pridávať na atraktivite, alebo naopak má slúžiť skôr vývojárovi ako napríklad štatistiky prístupov, alebo kontrola domény, na ktorej sa daná hra nachádza. Výsledky môžete vidieť na obrázku 3.1. Dáta slúžia iba pre predstavu, čo najviac vývojárom chýba, alebo naopak, ktoré komponenty z vymenúvaných pokladajú z ich hľadiska za najdôležitejšie, najprínosnejšie.

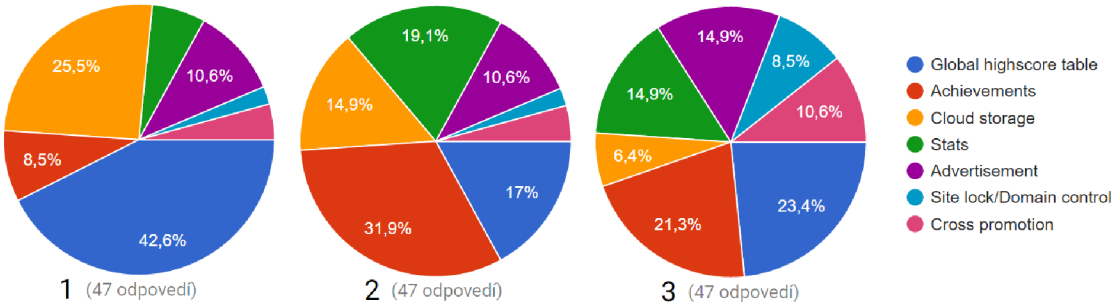
Na základe tohto dotazníku, ale predovšetkým na základe analýzy už existujúcich riešení, bola vybraná nasledujúca množina komponent.

- Tabuľky skóre
- Odmeny
- Kontrola domén
- Štatistiky prístupov

[1] http://www.html5gamedevs.com/, https://www.scirra.com/forum/
[2] HTML5 Mobile Game and App Development a Game Development Community CZ/SK na Facebooku

- Úložisko v cloude

Niektoré komponenty neboli vybrané z dôvodu, že neboli pokladané vývojármi za až tak podstatné (napríklad cross-promotion), alebo sa nehodili k danému projektu (napríklad chat, ktorý by v hre na mobilných zariadeniach, s nie až tak veľkými obrazovkami, mohol byť viac na obtiaž, než k úžitku). Ďalšia komponenta, s ktorou sa v tomto projekte nepočíta, aj keď v dotazníku bola táto možnosť pomerne chcená, je integrácia reklám do hry. Takáto komponenta by bola nad rámec tejto práce a muselo by sa okolo nej riešiť niekoľko ďalších vecí, ako je získanie inzerentov, správa reklamných kampaní, a ďalšie z väčšej miery netechnické veci.

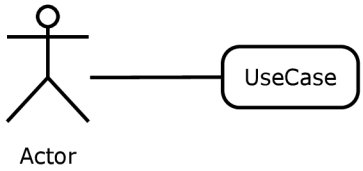


Obr. 3.1: Výsledky dotazníku

3.2 Návrh webovej platformy

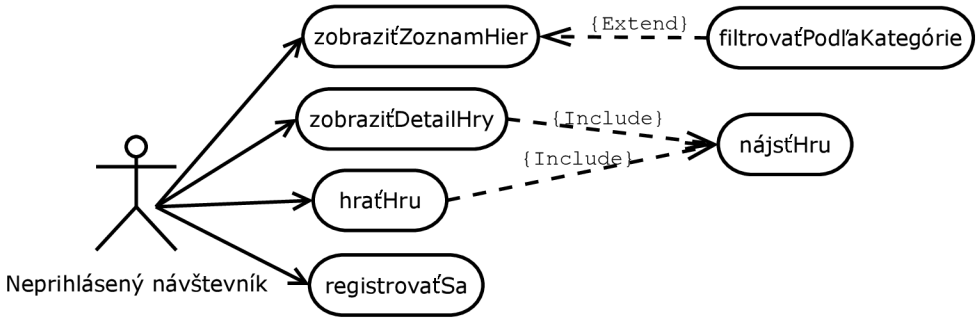
Návrh webovej platformy, bol vypracovávaný po analýze existujúcich herných platforiem, herných klientov a webových portálov, ktoré sú čiastočne uvedené v kapitole 2. V prvom rade bolo treba premyslieť, k čomu všetkému by mala daná webová platforma slúžiť, čo všetko by na nej malo byť realizovateľné, aké možnosti by ponúkala pre vývojárov hier, a aké možnosti zase pre hráčov týchto hier.

Tento návrh, toho, čo všetko môže daný užívateľ so svojou rolou na portáli vykonávať, bol spracovaný pomocou Use Case Diagramu (Diagramu prípadov užitia), ďalej iba UCD, ktorý zobrazuje chovanie systému, tak ako ho vidí užívateľ. Účelom tohto diagramu, je popísať funkcionalitu systému, teda čo sa od systému očakáva. Diagram hovorí o tom, čo má systém, v tomto prípade naša webová platforma, vedieť, ale nehovorí už o tom, ako sa to bude vykonávať. UCD sa skladá, z prípadov užitia (Use Case), aktérov (Actor) a vzťahov medzi nimi. Prípad užitia je sada niekoľkých akcií, ktoré vedú k dosiahnutiu určitého cieľa. Aktér je rola, ktorá komunikuje s jednotlivými prípadmi užitia. Zjednodušene povedané, UCD je diagram, ktorý zobrazuje to, čo všetko daná užívateľská rola môže v systéme vykonávať. Na obrázku 3.2 môžete vidieť naľavo aktéra a napravo prípad užitia.



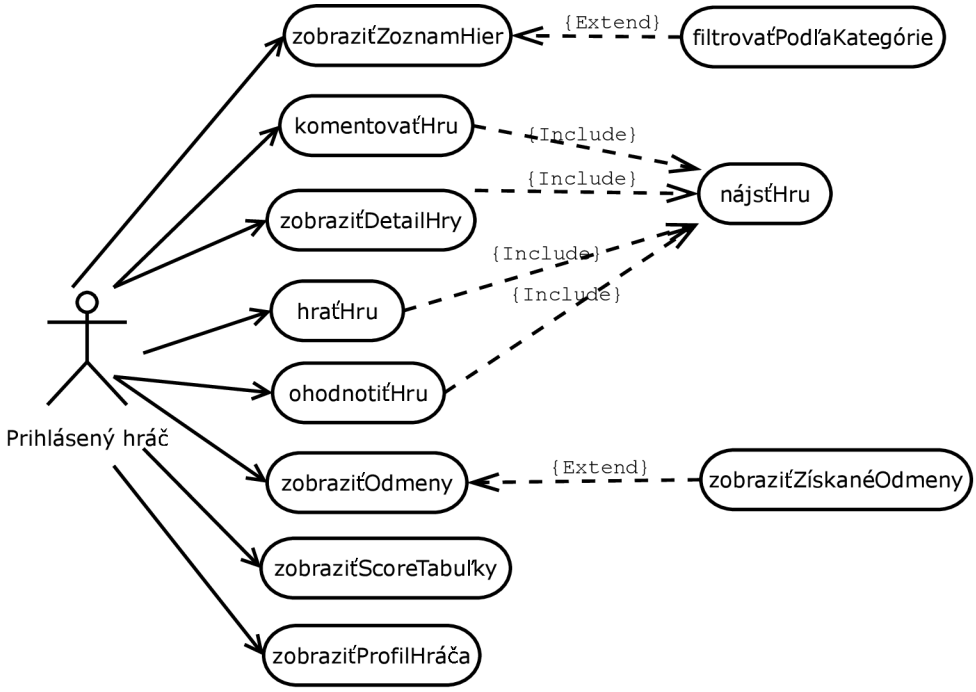
Obr. 3.2: Use Case Diagram, naľavo aktér, napravo prípad užitia

Prvou rolou, ktorou sa užívateľ stáva po príchode na webovú platformu, je rola neprihláseného návštevníka. Prirodzene, neprihlásený návštevník musí mať obmedzenú množinu toho, čo môže na danej stránke vykonávať. V prípade tejto bakalárskej práce, môže byť neprihlásený užívateľ napríklad hráč, ktorý sa ešte nezaregistroval, alebo sa z nejakého dôvodu nechce registrovať. Na portál prišiel vyložene skrátiť dlhú chvíľu, skóre tabuľky najlepších hráčov ho nemotivujú k súťaženiu s ostatnými hráčmi a ani odmeny nepokladá za motivujúci prvok hry. Keďže platforma by mala byť optimalizovaná pre prenosné zariadenia, ako mobily a tablety, tak to môže byť hráč s mobilným zariadením, čakajúci napríklad v ordinácií lekára, alebo na autobusovej zastávke. Takýto hráč, teda potrebuje jediné. Potrebuje zobraziť zoznam hier, aby si vybral hru, ktorú chce hrať a prípadne potrebuje filtrovať zoznam hier podľa nejakých kategórií, ako napríklad žánru a podobne. Po vybraní hry si chce prečítať základné informácie o hre, inštrukcie ako hru ovládať a v neposlednom rade náhľady, alebo iné grafické materiály, pochádzajúce najlepšie priamo z danej hry. Po tom všetkom chce samozrejme daný hráč hrať samotnú hru umiestnenú na portáli. Neprihláseným návštevníkom, môže byť však aj hráč, ktorý sa chce na daný portál prihlásiť, poprípade sa zaregistrovať a mať tak možnosť využívať všetku funkcionalitu daného portálu, určenú pre rolu prihláseného hráča. Posledným prípadom, ktorý môže vystupovať ako neprihlásený návštevník je, užívateľ, ktorý chce na tomto portáli získať rolu vývojára hry a mať tak možnosť sa podieľať na tvorbe herného obsahu, alebo využívať iné funkcie určené vývojárom. Diagram prípadov užitia, ktorý zhrnuje tento odsek je na obrázku 3.3.



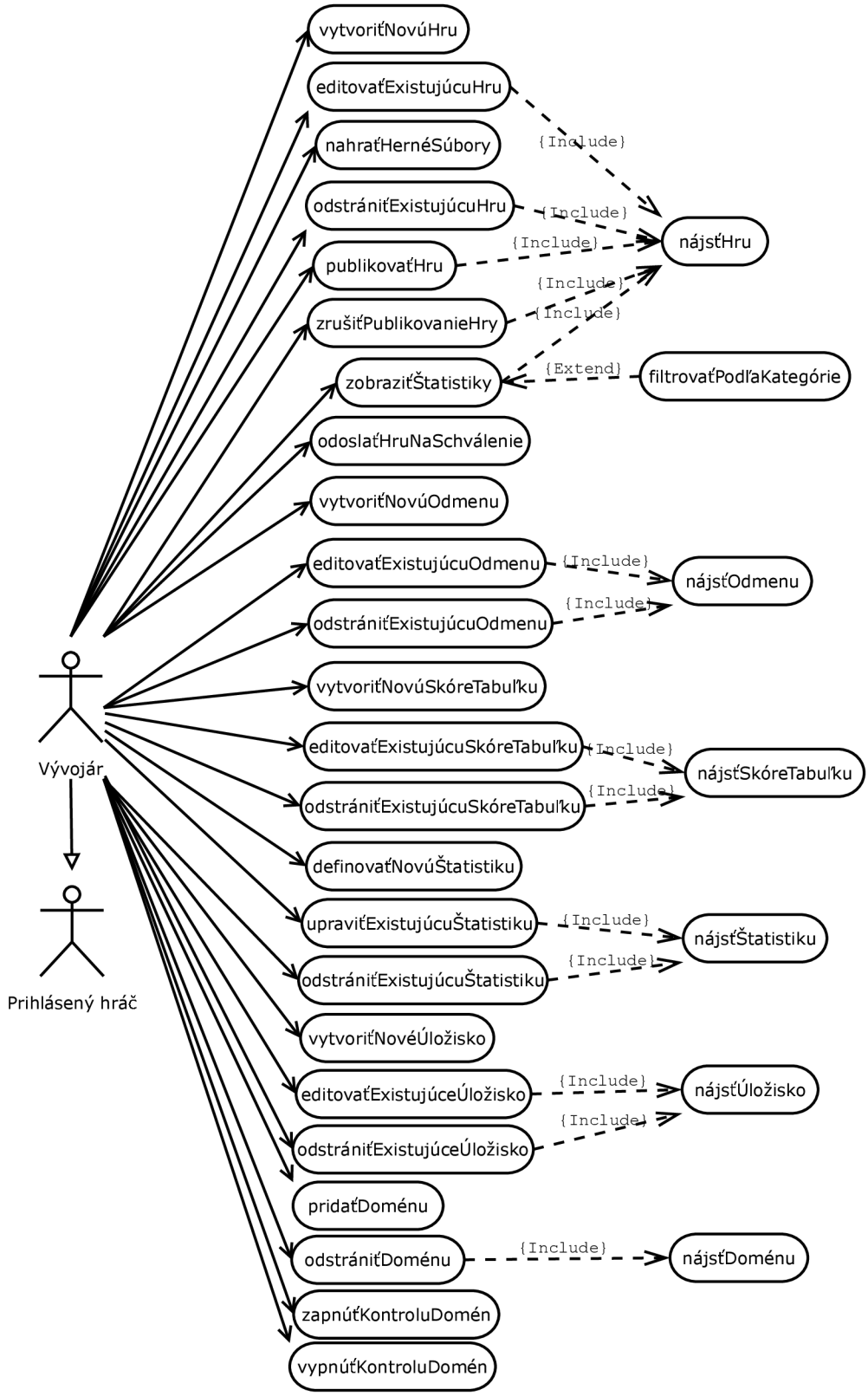
Obr. 3.3: Use Case Diagram neprihláseného užívateľa

V poradí druhou rolou, ktorou sa užívateľ môže stať v rámci systému, je rola prihláseného hráča. Neprihlásený návštevník sa prihláseným hráčom stáva od chvíle, kedy sa prihlási k svoju účtu hráča. Prihlásený hráč môže vykonávať na portáli všetky akcie, ktoré môže vykonávať rola neprihláseného návštevníka, teda zobrazovať zoznam hier, filtrovať ich, zobrazovať detaily o hre a hrať hru. K tomu ešte môže vykonávať ďalšie akcie, na ktoré je potrebné byť prihlásený. Prihlásený hráč môže hrať hry pod svojim účtom. Tým pádom môže hráč získavať v danej hre odmeny a umiestňovať sa v skóre tabuľkách najlepších hráčov a tým nepriamo súťažiť s ďalšími prihlásenými hráčmi. Okrem toho, prihlásený hráč má možnosť číselne ohodnotiť hru, alebo zdieľať svoje pocity z hry s ostatnými , publikovaním svojho komentáru na stránke s hrou. Hráč by mal mať možnosť tiež prezerať si svoj hráčsky profil a profil ostatných hráčov. Use case diagram, ktorý popisuje akcie prihláseného hráča môžete vidieť na obrázku 3.4.



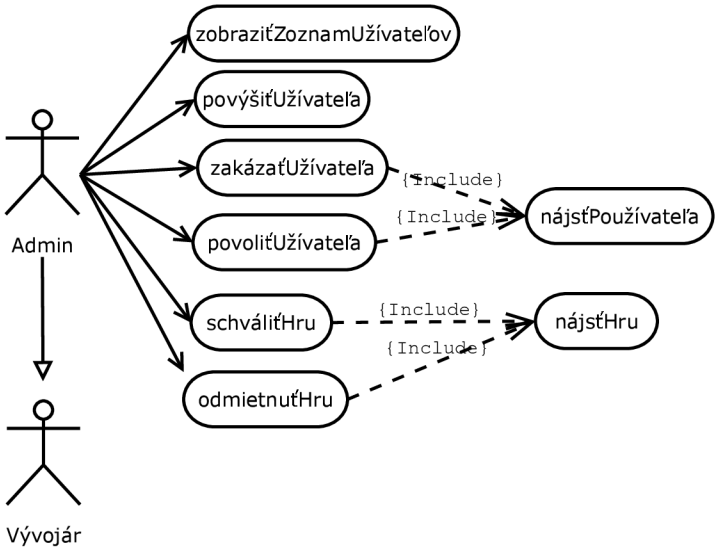
Obr. 3.4: Use Case Diagram prihláseného hráča

Treťou, a asi aj najvýznamnejšou rolou v rámci tejto práce je rola vývojára. Užívateľ rolu vývojára nadobudne, zaregistrovaním/prihlásením sa do svojho vývojárskeho účtu. Vývojár má povolené vykonávať to isté, ako predošlé dve role, neprihlásený návštevník a zároveň prihlásený hráč. V pomyslenej hierarchii, je táto rola zase o niečo vyššie, ako spomenuté predošlé dve. Rola vývojára je schopná vytvárať a definovať nové hry, editovať informácie o hre, či už existujúcej, alebo novo vytváranej. Vývojár, ďalej musí byť schopný nahrať herné súbory, obrázky k hre, ikony a iné promočné materiály, potrebné pre platformu. Musí byť schopný, zatiaľ nepublikovanú hru, publikovať/poslať na schválenie a už publikovanú hru, odstrániť zo zoznamu hier, viditeľných pre neprihláseného návštevníka a prihláseného hráča. Ďalej, tento užívateľ, ktorý pracuje pod touto rolou musí byť schopný zobraziť štatistiky prístupov o hre, poprípade, zmeniť typy zobrazovaných štatistických grafov. Jednou z najvýznamnejších funkcií portálu, ku ktorej má táto rola prístup je práca s API. Presnejšie sa jedná o definovanie API komponent. Táto práca s API, zahrňuje definovanie, vytváranie nových odmien, editovanie už existujúcich a taktiež možnosť ich odstránenia. Umožňuje prácu s tabuľkami skóre, ich definovanie, vytváranie, editovanie, odstránenie, ich rôzne nastavenia... Tiež umožňuje definovať cloud úložisko, editovať ho, alebo ho kompletne odstrániť. V neposlednom rade, by mal byť vývojár schopný definovať, na ktorej doméne je možné hru s API spustiť a na ktorej nie, poprípade túto funkcionalitu vypnúť/zapnúť.



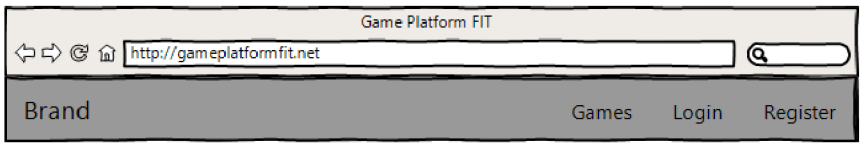
Obr. 3.5: Use Case Diagram vývojára

Poslednou, avšak nie nevýznamnou rolou, ktorá sa v pomyslenom hierarchickom rebríčku právomocí nachádza úplne najvyššie, je rola administrátora webovej platformy. Bežný užívateľ nemá možnosť stať sa administrátorom, žiadnou akciou z jeho strany. Užívateľa môže povýšiť na administrátora, iba iný administrátor, alebo správca databáze, ktorý má prístup k tabuľkám užívateľov a ich rolí. Administrátor má všetky práva a môže vykonávať všetky akcie a riadiť všetky udalosti na platforme, ktoré mu grafické užívateľské prostredie dovoľuje vykonávať. Teda užívateľ, ktorý nadobudol rolu administrátora, môže vykonávať všetko to čo neprihlásený návštevník, prihlásený hráč a zároveň aj vývojár. Okrem toho, takýto užívateľ, má na starosti správny chod platformy, teda by mal mať možnosť deaktivácie a znovu aktivácie užívateľských účtov, v prípade, že by konkrétny užívateľ nejak porušoval pravidlá platformy. Tento užívateľ, by mal mať možnosť editácie všetkých hier a všetkých užívateľských profilov, nielen tých, ktoré mu patria. Ďalšou významnou úlohou, ktorú má rola administrátora na starosti, je schvaľovanie a odmietanie vývojármi vytvorených a odoslaných hier na schválenie.



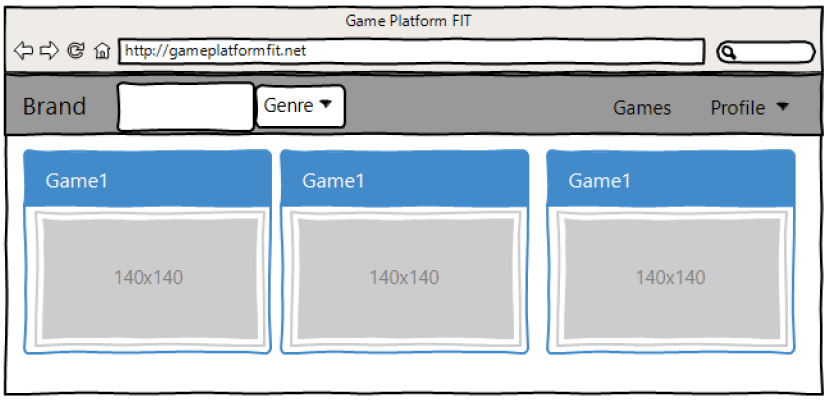
Obr. 3.6: Use Case Diagram administrátora

Návrh grafického užívateľského rozhrania môže byť kľudne rozdelený podľa rolí, teda na stránky, kam pristupuje administrátor, kam zase developer, a kam hráč a návštevník webu. Podľa toho sa dá potom rozhranie správne pripraviť a navrhnúť. V prvom rade je ale dobré si uvedomiť, ktoré prvky majú všetky 3 role spoločné. Určite to bude menu v hlavičke, ktorého návrh môžete vidieť na obrázku 3.7. Táto hlavička by mala obsahovať nejaké textové, alebo obrázkové logo naľavo a základné menu napravo. Pričom obsah v menu napravo, by bol generovaný podľa toho, v akej roli užívateľ vystupuje. V prípade, že nie je užívateľ prihlásený/registrovaný, tak by sa tam mal nachádzať formulár na prihlásenie/registrovanie.



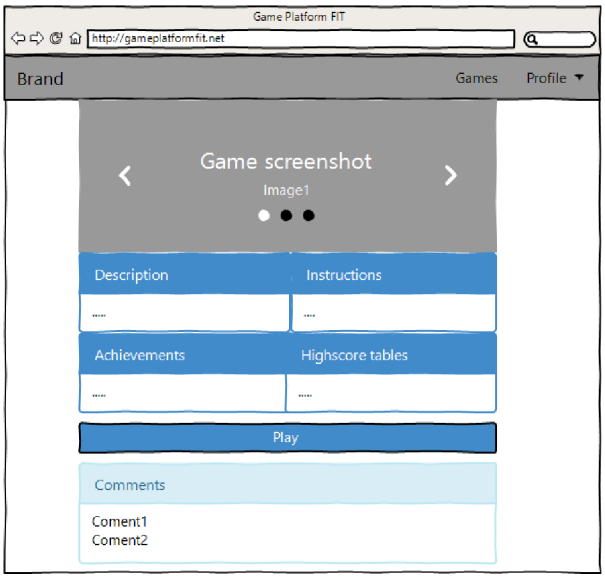
Obr. 3.7: Hlavička s logom a menu

Stránky určené pre hráčov hier by boli: stránka so zoznamom hier, detail hry a poprípade profil hráča. Stránka so zoznamom hier by obsahovala prirodzene jednotlivé hry. Každá hra by tu bola reprezentovaná názvom a svojím náhľadovým obrázkom, nie snímkou z hry, skôr nejakou graficky príťažlivou upútavkou k hre. Zoznam hier by mal byť filtrovateľný podľa jednotlivých kategórií. Návrh môžete vidieť na obrázku 3.8.



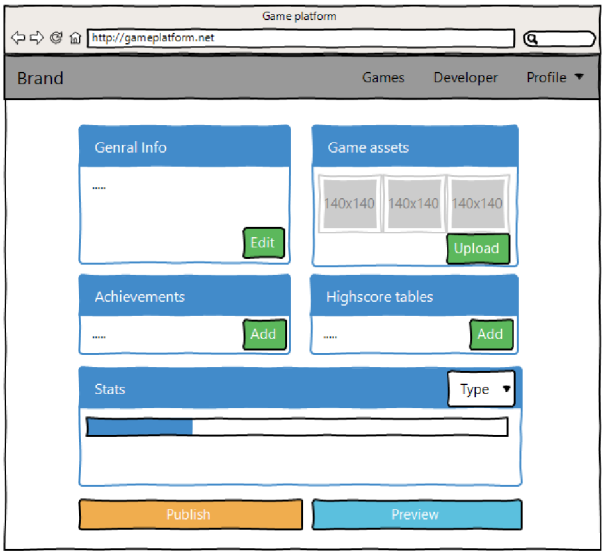
Obr. 3.8: Stránka s hrami

Detail hry by mal obsahovať, slide show obrázkov, už zo samotnej hry. Slideshow by sa mala sama meniť, poprípade by mal užívateľ možnosť ručne prechádzať obrázky, pomocou šípiek na stranách. Ďalej by tam boli základné informácie o hre, ako popis a inštrukcie, ktoré by boli zobrazené kvôli grafickej príťažlivosti v paneloch. V rovnakých paneloch, by mali byť zobrazené aj odmeny a tabuľky skóre. Prihlásený hráč by mal tiež vyznačené, ktoré odmeny získal, a ktoré nie. Riadok, so získanou odmenou, by bol vyfarbený na zeleno, s odmenou, ktorú ešte nezískal, zase na červeno. Pod tým všetkým, by sa malo nachádzať tlačidlo „Play“, slúžiace na spustenie hry. Úplne na konci, by sa mali nachádzať pridané komentáre k hre od užívateľov a formulár, kde by samozrejme prihlásený užívateľ mal tiež možnosť komentovať. Celý zjednodušený návrh môžete vidieť na obrázku 3.9.



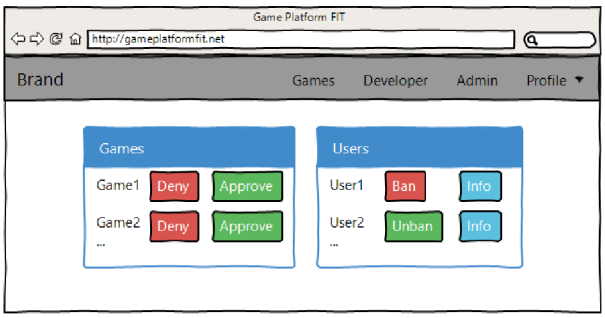
Obr. 3.9: Detail hry

Stránky, ku ktorým by pristupoval vývojár hry, by boli tie isté, ako stránky, ku ktorým pristupuje hráč. Okrem toho by mal mať prístup do vývojárskej konzole, kde by bol zoznam hier, ktoré daný vývojár vyvíja a detail, alebo lepšie povedané úprava/tvorba existujúcej/novej hry. Vývojárska konzola so zoznamom hier, by bola takmer totožná z hľadiska GUI ako už popísaný zoznam hier určený pre hráča, takže ju nemá zmysel popisovať podrobnejšie. Stránka s detailom vyvíjanej hry, by pozostávala z panelov. Každý panel, by mal slúžiť pre iný druh informácií. Jeden by bol pre hlavné informácie o hre, druhý pre grafické materiály k hre, tretí k odmenám, tabuľkám skóre a zobrazenie štatistík. Informácie v každom paneli by boli editovateľné poprípade vytvoriteľné po kliknutí na tlačidlo „Edit“, po ktorom by sa vyvolalo modálne okno, s príslušným editačným formulárom.



Obr. 3.10: Úprava hry v vývojárskej konzole

Admin by mal samozrejme prístup ku všetkému, v rámci platformy. Teda, by mohol využívať stránky navrhnuté ako pre hráča, tak aj pre vývojára hier. Admin taktiež potrebuje nejakú administrátorskú konzolu, aby mohol spravovať hry a užívateľov. Mal by v tejto konzole mať možnosť vidieť hry, ich status, alebo presnejšie povedané, či sú schválené, alebo neschválené a mať možnosť ich jednoducho schváliť, alebo zamietnuť z publikácie. V zozname užívateľov, by mal mať zase možnosť jednotlivým užívateľom odopierať prístup na stránku, poprípade prístup povoľovať. Mockup administrátorskej konzole je možné vidieť na obrázku 3.11.



Obr. 3.11: Konzola administrátora

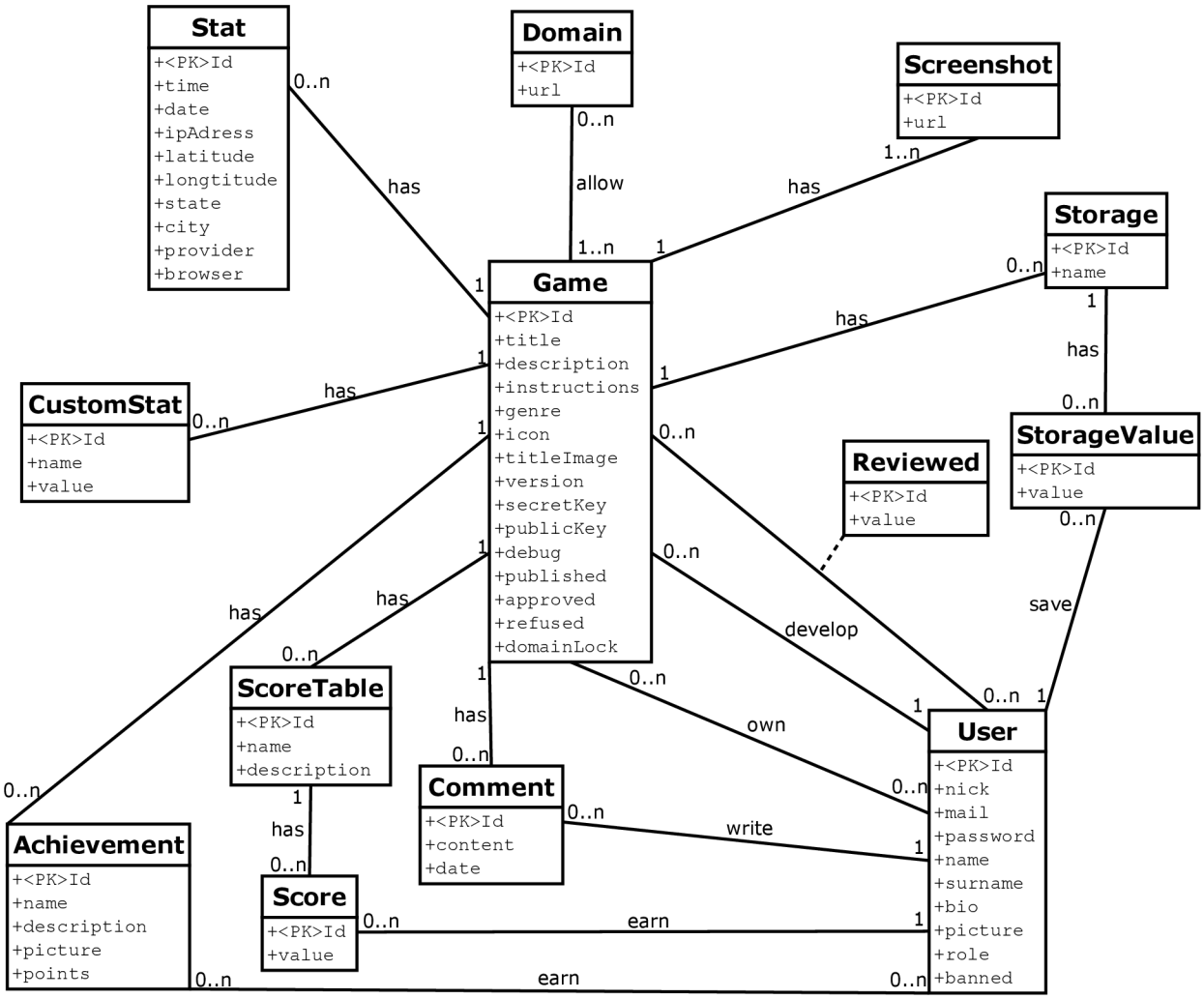
3.3 Návrh databáze

Návrh databáze vychádza v určitej miere z predchádzajúcej kapitoly o návrhu webovej platformy, kde boli predstavené a namodelované diagramy prípadov užitia. Tieto 2 kapitoly teda spolu úzko súvisia, keďže užívateľ chce na webovej platforme vykonávať rôzne akcie a častokrát, alebo aj skoro vždy, požaduje, aby sa informácie niekde uchovali, poprípade aby už uložené informácie od niekal získal. Táto kapitola teda do istej miery obsahuje informácie už uvedené v kapitole o návrhu webovej platformy, s tým rozdielom, že informácie sú interpretované tak, aby z nich bolo možné urobiť návrh databáze. Z nasledujúceho popisu by malo vyplývať, aké informácie v databázy je potrebné uchovávať.

Návrh databáze bol modelovaný pomocou Entity relationship diagram (ERD) s Unified modeling language notáciou. ERD sa využíva v softwarovom inžinierstve pre abstraktné a konceptuálne znázornenie dát. Pri modelovaní takéhoto diagramu vzniká jeden z typov konceptuálnych, alebo schematických dátových modelov systému, v tomto prípade relačnej databáze, a požiadaviek na ňu z hora dole. V prípade že sa jedná o návrh webovej platformy, portálu alebo iného vo svojej podstate informačného systému, ktorý je založený na databázy, tak sa konceptuálny model v neskoršej fázy mapuje na logický dátový model (napríklad relačný model), a ten je ďalej mapovaný na fyzický dátový model. Existuje niekoľko bežných notácií využívaných pre ERD určené pre rôzne typy návrhov, ako logické, fyzické alebo konceptuálne. Jedny zo známych konvencií pre písanie diagramov sú Chenova notácia, Crows Foot notácia alebo už spomenutá UML notácia.

Najhlavnejšiu úlohu v tomto projekte zastáva hra. Hra je teda v diagrame akási centrálna entita, bez ktorej by nemalo zmysel tento databázový návrh, a vôbec celý projekt robiť. Hra obsahuje názov, popis, inštrukcie, ikonu, titulný obrázok, dvojicu kľúčov, aktuálne nahranú verziu hry na portáli, jeden herný žáner a príznaky, či je vyvíjaná, publikovaná, schválená, odmietnutá, a či využíva funkciu uzamknutia na domény. Hra môže mať niekoľko snímok obrazovky, minimálne ale 1 musí byť prítomný vždy. Daný snímok obrazovky patrí vždy jednej konkrétnej hre. Snímok obrazovky obsahuje url, kde sa nachádza v rámci serveru. Ďalej môže byť hre priradených niekoľko domén, daná doména môže prislúchať práve jednej hre. Doména obsahuje svoju url. Hra tiež obsahuje ďalšie funkcie spojené prevažne s API, ako štatistika, vlastná štatistika, cloud úložisko, tabuľky skóre a odmeny. Štatistika prislúcha práve jednej danej hre a hra môže mať niekoľko štatistík. Štatistika v sebe obsahuje informácie o IP adrese, čase, dátume, zemepisnej šírky a dĺžky, štáte, mesta, internetovom poskytovateľovi a použitom internetovom prehliadači. Okrem toho ale hra môže mať definované vlastné štatistiky. Týchto štatistík môže byť hneď niekoľko a daná vlastná štatistika prislúcha práve jednej hre. Vlastná štatistika obsahuje svoje meno a hodnotu. Úložísk môže mať hra taktiež niekoľko a jedno konkrétne úložisko patrí iba jednej hre. Úložisko má v sebe uložené hodnoty, ktorých môže byť viac, jedna hodnota úložiska patrí jednému úložisku a zároveň jednému konkrétnemu užívateľovi. Odmeny obsahujú meno, popis, url obrázku, a body. Daná odmena patrí k jednej hre a hra odmien môže mať viacero. Užívateľ tieto odmeny má možnosť získať a môže ich získať toľko, koľko odmien existuje. Posledným z API tabuliek sú skóre tabuľky. Skóre tabuliek môže mať hra niekoľko a jedna skóre tabuľka patrí k jednej hre. V skóre tabuľke sa nachádzajú hodnoty, ktorých môže mať skóre tabuľka veľa, ale jedna konkrétna hodnota skóre patrí práve jednej skóre tabuľke a zároveň jednému užívateľovi. Užívateľ je ďalšia dôležitá entita, ako je už vidieť z predošlých riadkov. Užívateľ v sebe nesie informácie o svojej prezývke, menu, priezvisku, heslu, e-mailu, biografií, obrázku, o svojej roli v rámci systému a príznak či má zákaz prístupu na stránku, alebo nie. Hra sa môže nachádzať u hocijakého počtu hráčov v ich osobných knižniciach. Táto osobná

knižnica v podstate predstavuje zoznam obľúbených hier. Hru by mal mať možnosť užívateľ do tejto knižnice pridávať, v prípade, že sa tam ešte nenachádza, alebo v prípade, že sa tam už nachádza, by ju mal mať možnosť z nej odstrániť. Hráč môže mať vo svojej knižnici hocikoľko hier, alebo v nej nemusí mať žiadnu hru. Užívateľ by ďalej mal mať možnosť hru ohodnotiť a komentovať. Hra môže byť hodnotená a komentovaná od hocijakého počtu prihlásených užívateľov. Užívateľ môže komentovať jednu hru niekoľko krát, avšak hodnotiť ju môže iba raz, a potom môže svoje hodnotenie už iba upravovať. Danú konkrétnu hru môže vyvíjať iba jeden užívateľ.



Obr. 3.12: Návrh databáze ako ER diagram

3.4 Návrh aplikačného rozhrania

API by malo byť implementované z väčšej časti na serverovej strane, z menšej časti na strane klienta. Malo by fungovať tak, že klient inicializuje spojenie so serverom pomocou AJAX a CORS. V XHR správe pošle nejaký identifikátor, pravdepodobne špeciálne vygenerovaný reťazec tzv. API kľúč. Server by mal následne odpovedať, ideálne v nejakom formáte, ktorý server dokáže vygenerovať a klientská strana zase čítať. Na tento účel bol vy-

braný JavaScript Object Notation, skrátene JSON, ktorý sa používa pre takýto typ prenosu dát. Výhodou je, že je platformne nezávislý, a to aj napriek tomu, že používa Javascriptovú syntax. Server by teda mal odoslať odpoveď o tom, či spojenie akceptuje a preniesť ďalšie potrebné dáta. Odoslaná JSON správa by mala vyzerať tak, že by sa tam nachádzala akási položka *status*, ktorá by vypovedala o tom, ako skončila operácia vykonávaná na serveri, či skončila úspechom, poprípade neúspechom, a akým presne.

```
1 {
2     status: '0',
3     result: 'some data'
4 }
```

Daný neúspech by bol definovaný dopredu určeným číslom. V položke *result* by zase boli klientom požiadané dáta, v prípade že operácia skončila úspechom a tieto dáta by sa na klientskej strane v danej funkcií spracovali do užívateľsky prijateľnej podoby, a následne by boli ďalej predané do aplikácie

Položka status/návratový kód	Význam
0	Operácia na serveri skončila úspešne
1	Operácia na serveri skončila neúspešne
1<	Ďalšie, špecifickejšie neúspešné skončenie operácie

Tabuľka 3.1: Tabuľka návrhov návratových kódov

Z vybraných komponent pre použitie v API vyplýva, že by v API mali byť implementované nasledujúce funkcie:

- Inicializácia API
- Meranie štatistík prístupov
- Kontrolovanie domén
- Práca s užívateľským účtom
- Práca s tabuľkou skóre
- Práca s odmenami
- Práca s cloudovým úložiskom

3.5 Návrh Hry

Pri tvorbe návrhu hry bolo brané na vedomie, že hra má byť jednoduchá, čo sa týka použitých programovacích techník, keďže hra má slúžiť predovšetkým na demonštráciu webovej platformy a jej API. Vývojár, ktorý by si chcel eventuálne hru rozobrať, by sa nemal snažiť pochopiť príliš zložitý kód. Z toho dôvodu, boli hneď vylúčené zložité herné žánre a na výber sa ponúkali skôr klasické hry (rôzne logické stolné hry, ako pexeso, puzzle...), alebo pomerne nové obľúbené a populárne mobilné casual hry[3] (napríklad Flappy Bird, TimberMan, alebo rôzne Match3 klony), ktoré sú v niektorých prípadoch na implementáciu ešte jednoduchšie, ako spomínané klasické žánre. Ďalším dôležitý kritériom pri návrhu demonštračnej hry

[3]Herný žáner cielený na občasných hráčov hier, vyznačujúci sa väčšinou predovšetkým veľmi jednoduchými pravidlami

bolo, aby v hre boli implementovateľné všetky navrhnuté funkcie API. Posledným, avšak nie až tak dôležitým kritérium bolo, aby celá grafická stránka hry bola ľahko vytvoriteľná aj pre človeka, ktorý nie je primárne grafikom. Ideálne by teda hra mala byť bez zložitých animácií a ďalších v tomto prípade zbytočných zložitostí.

Po zhodnotení možností, bola ako demonštračná hra zvolené pexeso. Dôvodom je to, že pexeso je pomerne ľahko implementovateľné. Dajú sa do neho integrovať všetky navrhnuté API funkcie a pre tvorbu grafiky nie sú potrebné žiadne pokročilé grafické skúsenosti, nehovoriac o animáciách. Hra by sa mala skladať z niekoľkých obrazoviek, z hlavného menu, z obrazovky, na ktorej bude zobrazená tabuľka najlepších hráčov s ich skóre, obrazovka slúžiaca ako akási sieň trofejí, v ktorej budú zobrazené získané odmeny a obrazovka, na ktorej sa bude odohrávať samotná hra. Na obrazovke hry by bola zobrazená hracia plocha + nejaký základný HUD[4], informujúci o tom, koľko kariet zostáva otočiť, koľko hráč nahral bodov, a aké je doposiaľ najväčšie nahrané skóre daného hráča.

Hráč má v tejto hre za úlohu nájsť všetky rovnaké dvojice hracích kariet. Ak by našiel rovnakú dvojicu, tak by získal +10 bodov. Ak by otočil dvojicu rozdielnych kariet v jednom kole, tak by stratil -1 bod. Takto by mohol získavať skóre a následne po otočení všetkých rovnakých dvojíc, by jeho skóre bolo odoslané na server a hráč by bol presmerovaný na obrazovku s tabuľkou najlepších hráčov. Odmeny by mohli byť implementované napríklad tak, že hráč by po prvej hre dostal odmenu, potom by mohol dostať odmenu, ak by nahral skóre väčšie ako 100 bodov, väčšie ako 200 bodov a tak podobne. Cloudove úložisko by zase mohlo v tejto hre slúžiť, ako počítadlo, koľko hier daný hráč odohral.

[4]Head-up display, metóda, ktorá vizuálne odovzdáva informácie hráčovi, ako súčasť užívateľského rozhrania hry

Kapitola 4

Použité technológie

Pri implementácií platformy bolo nutné si dobre premyslieť, čoho sa má dosiahnuť a tak špecifikovať požiadavky na technológie, ktoré sa budú využívať. Po zhrnutí požiadaviek sa zistilo, že bude potrebný programovací jazyk pre serverovú časť, programovací a značkovací jazyk pre klientskú časť, jazyk, v ktorom bude implementované API pre vývojárov, databázu a databázový jazyk a napokon jazyk, v ktorom bude implementovaná hra demonštrujúca funkcie webovej platformy.

4.1 Serverová časť

Pod pojmom serverová časť sa myslí všetko to, čo sa vykonáva na serveri. Teda v prípade tejto práce to je hlavne generovanie webovej stránky platformy a práca s dátami uloženými v databázy. Okrem týchto vecí sa používa serverová časť pri práci s API[1], kedy sa využíva AJAX pre dotazovanie sa na server.

Pri výbere jazyka pre serverovú časť sa ponúkalo hneď niekoľko možností. Bolo na výber medzi PHP, Python, Java, Ruby, alebo ASP . NET. Napokon bolo vybrané PHP a zavážili hlavne vlastnosti, ktoré sú spomenuté nižšie.

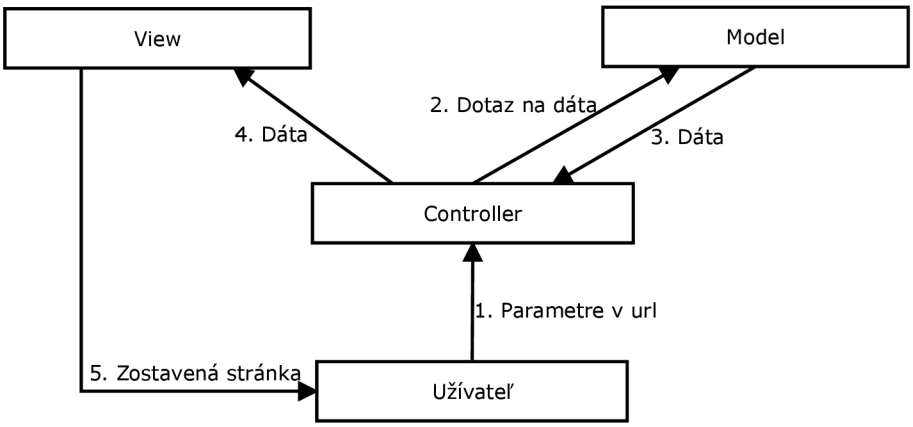
- **PHP -** Hypertext Preprocessor je široko používaný a univerzálny open source[2] skriptovací jazyk, ktorý sa obzvlášť používa na vývoj webových aplikácií a je možné ho vložiť do HTML súboru. Syntax jazyka je veľmi podobná syntaxi jazyku C. Od novších verzií sa PHP stal objektovo orientovaným a je v ňom možné používať OOP[3] postupy. Tento jazyk beží takmer na všetkých operačných systémoch od UNIXu cez Windows až po Mac OS X. [11]

  PHP bol zvolený pre serverovú časť hlavne kvôli tomu, že to je najrozšírenejší serverový jazyk a takmer všetci poskytovatelia hostingu ponúkajú webhosting práve s PHP a teda nenastáva problém s výberom poskytovateľa. Okrem toho PHP podporuje mnohé databázy ako MySQL, Oracle a PostgreSQL. Práve kombinácia PHP s Apache serverom a MySQL databázou je jedna z najobľúbenejších.

- **Model Controler View architektúra -** MVC je architektonický vzor, ktorý sa uchytil predovšetkým pri tvorbe webových aplikácií a informačných systémov. Je súčasťou populárnych webových rámcov, ako napríklad Nette, alebo Zend. Základnou

[1] Application programming interface
[2] otvorený kód
[3] objektovo orientované programovanie

myšlienkou je oddelenie logiky od výstupu. MVC pozostáva, ako už názov napovedá, z 3 typov komponentov. Z modelu, pohľadu (view) a kontroléru. Model obsahuje všetku logiku aplikácie. Môžu sa v ňom nachádzať výpočty, alebo databázové operácie. Komponent pohľad sa zase stará o zobrazovanie výstupu užívateľovi. Obsahuje minimálne množstvo logiky, ktorá je pre výpis nutná. Tretí a posledný typ je kontrolér, ktorý komunikuje s užívateľom, modelom aj pohľadom a komponenty navzájom prepája. Príklad takejto architektúry a komunikácie medzi jednotlivými komponentami je zobrazený na nasledujúcom obrázku. [14]



Obr. 4.1: Príklad MVC

- **Nette -** PHP je síce samo o sebe mocný programovací jazyk, ale v prípade zvolenia čistého PHP, by programátor musel riešiť veľké množstvo vecí, ktoré niekto v minulosti už dávno efektívnejšie vyriešil. Okrem toho by si programátor musel dávať veľký pozor a ošetrovať početné množstvo bezpečnostných rizík. Kvôli týmto veciam bol pre účel práce vybraný aplikačný rámec Nette.

  Nette je rámec napísaný v PHP5, ktorý sa zameriava na pohodlnejšie a rýchlejšie programovanie webových aplikácií a elimináciu bezpečnostných rizík. Podporuje MVC a OOP. Z veľkej časti je založený na vytváraní znovu použiteľných komponentov. K Nette patrí aj šablónovací systém Latte, ktorý zapuzdruje PHP v HTML súboroch do špeciálnych Latte makier a tým eliminuje bezpečnostné riziká. Tracy a Tester zase slúžia na hľadanie a odlaďovanie chýb. Pôvodným autorom je David Grundl, ale v súčasnosti sa o rozvoj rámcu stará Nette Foundation. Je ponúkaný pod licenciou GNU GPL a licenciou Nette (obdoba BSD licencie). Rámec sa teší veľkej obľube a taktiež početnej komunite v Čechách a na Slovensku. Minimálne požiadavky, pre správnu funkčnosť rámca, sa vyžaduje PHP verzie 5.3.1 a vyššej. Všetky požiadavky je možné overiť oficiálnym checker.php skriptom. [10]

4.2 Databázová časť

Databáza, ako prostriedok pre uloženie dát, bola zvolená hlavne kvôli vlastnosti ACID[4] a pre prístup viacerých užívateľov k databázy súčasne. Okrem iného, databázy šetria pamäťové miesto tým, že pri dobrom návrhu sa v databázy nenachádzajú redundantné dáta.

[4]atomickosť, konzistencia, izolovanosť a trvácnosť

Manipulácia s dátami je pomerne rýchla, pretože sa do medzi pamäte neukladá celá databáza. Nad databázou sa dajú vykonávať zložité dotazy a spájať dáta z viacerých tabuliek. Na výber boli rôzne alternatívy, ako napríklad PL/SQL, T-SQL, alebo vybraný MySQL.

MySQL je otvorený, viac užívateľský SQL relačný databázový server, ktorý je implementovaný vo viacerých programovacích a hlavne serverových jazykoch, napríklad v skriptovacom jazyku PHP. Je to databázový relačný systém, teda každá databáza v MySQL je tvorená z jednej alebo viacerých tabuliek, ktoré majú riadky a stĺpce. Riadky udávajú jednotlivé záznamy, stĺpce zase dátové typy jednotlivých záznamov. Práca s takouto databázou je vykonávaná pomocou takzvaných dotazov.

4.3 Klientská časť

Klientská časť pokrýva všetko to, čo je zobrazené užívateľovi. Teda v tomto prípade je myslená kompletná stránka, ktorej zostavovanie sa dialo z väčšej miery na servery. Aby boli informácie určené pre užívateľa zobrazené v nejakej prijateľnej a užívateľsky prívetivej podobe, tak sa vyžaduje okrem značkovacieho jazyka HTML použiť aj CSS kaskádové štýly. Keďže moderná prezentácia informácií na webových portáloch vyžaduje aj určitú ľahkosť a dynamiku, tak k tomuto účelu bol použitý celosvetovo rozšírený programovací jazyk Javascript. Nasledujúce informácie vychádzajú z internetovej stránky [9].

- **HTML -** Hyper Text Markup Language je značkovací jazyk určený pre popis webových dokumentov, hlavne webových stránok. Najprv bol iba veľmi jednoduchá podmnožina jazyka SGML[5], ale neskôr sa z neho vyvinul samostatný štandard. HTML dokument sa popisuje pomocou HTML tagov. Každý typ HTML tagu slúži na popísanie odlišného typu obsahu dokumentu.

  V minulosti sa HTML používal na definovanie štruktúry dokumentu a aj jeho výzoru. V súčasnej dobe je HTML určené iba na zmienenú definíciu štruktúry.

- **CSS -** Kvôli tomu, že HTML sa začalo používať iba na definovanie štruktúry dokumentu a nie jeho výzoru, bol vyvinutí Cascading Style Sheets. CSS popisuje ako budú HTML tagy zobrazené na obrazovke, papieri, alebo inom médiu. CSS šetrí veľa času tým, že môže popisovať rozmiestnenie viacerých webových stránok naraz.

- **Javascript -** Javascript je skriptovací, prototypovo založený jazyk, ktorý sa v súčasnosti používa hlavne pri tvorbe dynamických webových prezentácií a webových stránok. Javascript beží na klientskej strane webu. Je spúšťaný väčšinou vo webových prehliadačoch. V súčasnosti Javascript podporuje väčšina webových prehliadačov a je najpoužívanejší skriptovací jazyk používaný na klientskej strane.

  Bola by chyba myslieť si, že Javascript svoje uplatnenie pri tvorbe webov, má iba na strane klienta. Javascript je naozaj všestranný jazyk a svoje uplatnenie nájde aj na serveri napríklad v podobe Node.js. Pre účely tejto práce je avšak použitý klasický klientský Javascript spúšťaný vo webovom prehliadači.

- **JQuery -** JQuery je ľahká a rýchla Javascriptová knižnica, ktorej hlavným cieľom je, aby bolo použitie Javascriptu na webových stránkach omnoho jednoduchšie. Taktiež zjednodušuje komplikované veci Javascriptu, ako AJAX volania a manipuláciu s DOM. Okrem toho obsahuje metódy na manipuláciu s CSS, HTML udalosti, efekty

[5] Standard Generalized Markup Language

a animácie a ďalšie nápomocné programy. Je rozšíriteľná pomocou pluginov a na väčšinu známych vecí už existuje JQuery plugin. Knižnica zároveň rieši tzv. cross-browser problémy, kedy jedna vec sa môže v rôznych prehliadačoch správať odlišne.

Je to najobľúbenejšia a najrozšírenejšia Javascriptová knižnica, čomu dáva za pravdu nielen početná komunita okolo, ale aj to, že je využívaná najväčšími firmami ako Google, či Microsoft.

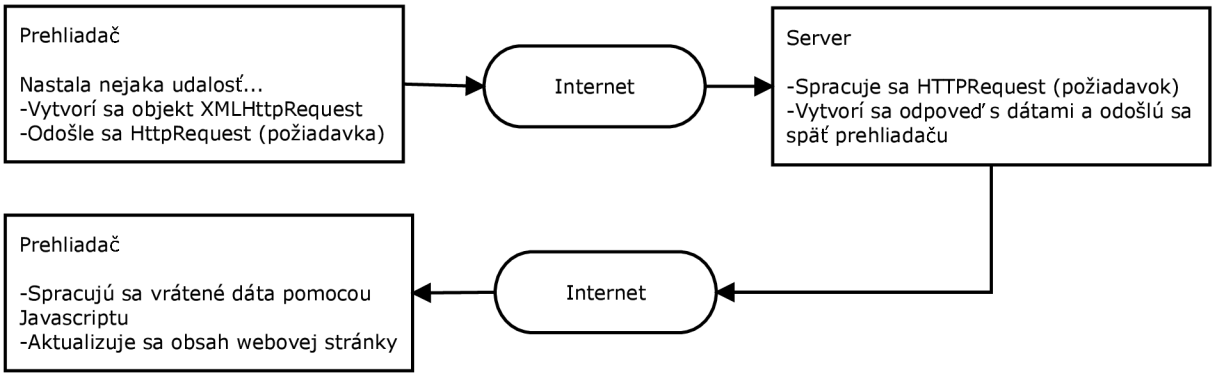
- **Bootstrap -** Bootstrap je frontendový aplikačný rámec pre rýchle a ľahké vyvíjanie responzívnych webových stránok. Obsahuje predpripravené dizajnérske typografické šablóny, založené na HTML a CSS. Hodí sa predovšetkým na prototypovanie, hlavne kvôli značnej neoriginalite vytvorených stránok, keďže väčšina stránok založená na tomto rámci vypadá veľmi podobne. Na druhú stranu, jednoduchými modifikáciami si môže skúsený dizajnér upraviť Bootstrap k svojmu obrazu.

- **Chart.js -** Chart.js je ľahká Javascriptová knižnica, pomocou ktorej je tvorba grafov z daných dát veľmi jednoduchá. Na zobrazovanie grafov používa HTML tag <canvas> do ktorého sa grafy vykresľujú. Na výber je z 6 najbežnejších typov grafov. Takto vytvorené grafy sú plne responzívne, takže môžu byť správne zobrazené, ako na stolných počítačoch s veľkou obrazovkou, tak aj na malých mobilných zariadeniach.

4.4 Aplikačné rozhranie

Aplikačné rozhranie, ďalej iba API, je v tomto prípade myslené, ako knižnica, pomocou ktorej vývojári hier pristupujú k funkciám webovej platformy. Knižnica je implementovaná v Javascripte a k funkciám platformy, ktoré sú implementované na serveri sa pristupuje dotazovaním technológiou AJAX.

Asynchronous Javascript and XML je súhrnné označenie technológií, ktoré umožňujú meniť obsah webových stránok bez nutnosti znovu načítania celej stránky zo servera. Hlavná výhoda spočíva v tom, že sa prenáša značne menšie množstvo dát. AJAX nie je samostatný programovací jazyk, ani technológia sama o sebe, ako by sa mohlo zdať. Je to skôr kombinácia niekoľkých prvkov. Je založený na internetových štandardoch. Používa kombináciu HMLHttpRequest pre získanie dát zo servera a Javascript/DOM pre zobrazenie a použitie dát. [8]



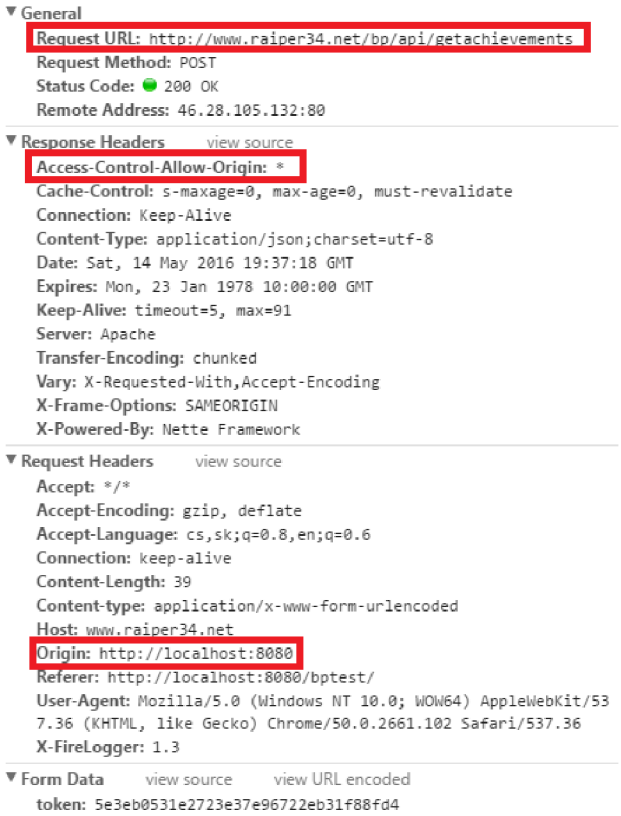
Obr. 4.2: Neblokujúca komunikácia so serverom pomocou AJAXu

Klasický Javascript je limitovaný tzv. Same origin policy. Z toho dôvodu nie je možné prijímať odpovede na dotazy z domény inej, ako je doména z ktorej bola požiadavka vyslaná.

Tento známy, hore popísaný problém rieši Cross-origin resource sharing. CORS je vlastne mechanizmus, ktorý povoľuje zakázané zdroje na webovej stránke, a tak je možné prijímať odpovede na požiadavky z inej domény, ako z domény z ktorej zdroje pochádzajú. Pre použitie tohto mechanizmu musí nastať zmena pri tvorbe požiadavku na klientovi a tiež je treba povoliť CORS na servery. Prehliadač musí posielať nastavenie Origin: doména v HTTP hlavičke. Na serveri zase musí byť nastavené povolenie Acces-Control-Allow-Origin: doména, alebo * pre povolenie všetkých domén. Zjendodušene CORS funguje nasledovne:

1. CORS požiadavku iniciuje klientský Javascriptový kód.
2. Prehliadač vloží dodatočné HTTP hlavičky do požiadavku pred tým, ako je odoslaný na server.
3. Server vloží do odpovede HTTP hlavičky, ktoré indikujú, že požiadavka je povolená.
4. Ak je požiadavka povolená, prehliadač odošle odpoveď klientskému Javascriptovému kódu.

Ak hlavičky vrátené serverom, neexistujú, alebo nie sú očakávané, odpoveď je zamietnutá a klient si nemôže túto odpoveď pozrieť. Na servery môže byť špecifikované, ktoré domény môžu robiť požiadavky a ktoré metódy a ktoré hlavičky sú povolené v HTTP požiadavke.



Obr. 4.3: Detail HTTP požiadavku a odpovede

Na obrázku 4.3 je možné vidieť detail HTTP požiadavky vyslanú z localhostu (položka Origin) na určitý server (položka Request URL) a odpoveď na túto požiadavku, kde je vidieť, že bola pridaná hlavička Access-Control-Allow-Origin. Tieto informácie boli prebrané z publikácie [13] hlavne z kapitoly "The Core of CORS", ktorá je voľne dostupná.

4.5 Hra

Pre demonštračnú hru sa ponúkalo buďto použiť klasický Javascript, čo ale nie je príliš rozšírená forma tvorby Javascript/HTML5 hier, alebo použiť jeden z početného množstva aplikačných rámcov, ktoré sú určené špeciálne na tvorbu hier. Na výber bol Ease.js, Tree.js, Panda.js, melon.js, Kiwi.js, alebo Phaser.js a mnoho ďalších. Bol vybraný Phaser.js, pretože to je jeden z najpoužívanejších a najbežnejších Javascriptových rámcov určených pre programovanie hier, o čom svedčí hlavne početná komunita okolo tohto rámca a fakt, že aj mnohé firmy z oblasti herného biznisu, ako Boostermedia, pracujú práve s týmto rámcom.

Phaser je Javascriptový aplikačný rámec určený špeciálne pre tvorbu hier, ako desktopových, tak aj mobilných. Je založený na renderovacom rámci pixi.js. Phaser je ľahko osvojiteľný, rýchly, je ponúkaný zadarmo a má otvorený kód. Podporuje canvas[6] a zároveň aj webgl[7]. Obsahuje veľké množstvo predpripravených funkcií. Tieto funkcie sú spracované tak, aby ich použitie bolo jednoduché. Príkladom môže byť stavaný preloader, fyzika, animácie, častice, rôzne módy prispôsobenia veľkosti obrazovky a iné. [5]

[6]HTML prvok, slúžiaci k dynamickému skriptovateľnému vykresľovaniu grafických primitív a bitových máp

[7]Web-based Graphics Library - je softwarova knižnica rozširujúca Javascript. Umožňuje vytvárať 3D grafiku a kód je spúšťaný na GPU.

Kapitola 5

Implementácia

Tak, ako kapitola venujúca sa návrhu riešenia bola rozdelená do logických častí, tak aj táto kapitola pojednávajúca o konkrétnej implementácií je rozdelená do obdobných logických častí. Kapitola začína popisom implementácie webovej platformy, pokračuje implementáciou API a končí implementáciou hry. Každá táto časť popisuje podrobnejšie významné implementačné detaily, ktoré boli navrhnuté v kapitole 3.

5.1 Implementácia webovej platformy

Webová platforma, ako sa píše už v kapitole 4, je implementovaná za pomoci rámca Nette. Celá aplikácia, je teda rozdelená do 3 vrstiev. Model slúži v prípade tohto projektu predovšetkým pre prácu s databázou, so súbormi a na autentifikáciu užívateľa. Všetky vytvorené modely sa nachádzajú v zložke *app/mvc/models*. Model *Authentication* slúži na autentifikáciu užívateľa, zjednodušene povedané, na jeho prihlásenie do systému. Autentifikácia v skratke prebieha tak, že funkcia, ktorá autentifikáciu vykonáva dostane na vstup ako parametre meno a heslo užívateľa, ktorý sa pokúša prihlásiť. Potom sa pozrie do databázy a zistí, či existuje užívateľ s takým užívateľským menom a s takým odtlačkom hesla. Podľa toho je následne užívateľ buď prihlásený, alebo je jeho žiadosť o prihlásenie zamietnutá a je vyhodená výnimka. V modeli s názvom *Queries* sa nachádza manipulácia a všetky operácie s databázou, či už to je vkladanie, vymazávanie, aktualizácia, alebo získanie informácií z databázových tabuliek. Okrem toho sa tam nachádzajú aj funkcie na prácu so súbormi. Je to z toho dôvodu, pretože práca so súbormi je v týchto funkciách obsiahnutá na jednom riadku, veľmi zriedkavo na viacerých. Na základe operácie so súborom, buďto nahranie súboru na server, alebo vymazanie zo serveru je potom potrebná ďalšia manipulácia s databázou. Je potrebné aktualizovať informácie o tabuľke, buď zmazaním záznamu z tabuľky, jeho prepísaním, alebo jeho vytvorením. To už záleží podľa konkrétnej operácie so súborom. Všetky modely sú v rámci frameworku potom zaregistrované v config.neon konfiguračnom súbore ako služby, aby bolo možné používať dependency injection[1]. Spojujúca vrstva aplikácie sa nachádza v súbore *app/mvc/presenter*, kde každý presenter má svoj vlastný priečinok, v ktorom sa nachádza samotná trieda presenteru a všetky formuláre využívané v rámci tohto presentru, ktoré sú implementované v továrenskej triede a následne zaregistrované ako služby. Presenter má na starosti vlastne jedinú vec, získať požadované informácie z modelu volaním nejakej jeho konkrétnej funkcie a predanie získaných

[1] Technika pre vkladanie závislostí medzi jednotlivými komponentami programu tak, aby jedna komponenta mohla používať druhú, bez toho, aby na ňu mala v dobe zostavovania programu referenciu

informácií ďalej do pohľadu. V triede presenteru sa nachádza niekoľko typov metód, budú spomenuté iba najvýznamnejšie použité. Metódy začínajúce slovom *render*, slúžia na predanie informácií do pohľadu, ku každej render metód musí existovať práve jeden pohľad. Metódy typu *action*, sú obdoba metód typu *render*, avšak tieto metódy sa používajú tam, kde je potrebné urobiť určitý úkon, napríklad v databázy a následne sa presmerovať (teda nič sa nevykresľuje , nepredáva do pohľadu) na iný presenter, alebo inú *action/render* metódu. Posledným typom metódy, ktorý je v rámci implementácie presenteru použitý, je typ začínajúci slovom *createComponent*. Metódy tohto typu slúžia na vytvorenie formulárov, alebo iných komponent a ich následné predanie do pohľadu. Po predaní je potom možné v pohľade používať makrá *{control}*, na vykreslenie komponenty. Pohľady k presenterom sú uložené v *app/mvc/presenters/templates*. Priečinky, ktoré sa nachádzajú v tejto zložke sú pomenované podľa presentrov a súbory v nich, čo už sú samotné pohľady, sú pomenované podľa *render* metód (napríklad pre *render* metódu s názvom *renderDefault*, tu existuje pohľad menom *default.latte*). Pohľady, ako bolo niekoľko krát spomenuté, sa starajú o vykresľovanie dát, ktoré im boli predané zo spojovacej vrstvy a nachádza sa v nich maximálne logika, potrebná na vykreslenie. Vykreslenie získaných dát je realizované pomocou latte makier. Predchádzajúce riadky tejto kapitoly popisovali princípy aplikácie všeobecnejšie. Nasledujúce riadky naopak budú popisovať implementačne zaujímavejšie detaily. Náhľadové obrázky ku všetkým významným sekciám portálu sa nachádzajú v prílohe D. Na stránkach určených pre hráčov s nachádzajú nasledujúce prvky, ktoré stoja za popis z hľadiska implementácie.

- **Triedenie podľa žánru -** triedenie podľa žánru sa nachádza na hlavnej stránke v hlavnom menu. Užívateľ si môže zvoliť pomocou dropdown boxu, o aký herný žáner má záujem a všetky ostatné hry iného žánru sa zneviditeľnia. Toto filtrovanie sa deje na základe obsahu *data-genre* atribútu, ktorý panely s hrou obsahujú. Hrám, ktoré majú hodnotu data-genre atribútu inú, ako je požadovaný vybraný žáner sa nastaví pomocou Javascriptu *display: none*. V budúcnosti, ak by platforma obsahovala väčší počet hier, by bolo avšak dobré riešiť filtrovanie pomocou Ajaxu a načítať hry podľa žánru priamo zo serveru. Pri malom počte hier toto riešenie ale úplne postačuje.

- **Hodnotenie hry** – hodnotenie hry sa nachádza v detaile hry. Užívateľ môže hru ohodnotiť buď 1 až 5 hviezdičkami. Podľa toho, na akú hviezdičku užívateľ klikne, taká hodnota sa prenesie na server pomocou Ajaxu a uloží sa do tabuľky reviewed. Stránku teda nie je potrebné načítať, keďže odoslanie hodnoty sa deje asynchrónne Ajaxom.

- **Spustenie hry** – samotná hra sa spúšťa z detailu hry buď v modálnom okne v prípade desktopových zariadení, alebo na samostatnej stránke v prípade mobilných a tabletových zariadení. Táto detekcia zariadenia sa vykonáva na klientskej strane pomocou Javascriptu, kedy sa kontroluje klientský operačný systém, a podľa toho sa vygeneruje udalosť po stlačení tlačidla na spustenie hry.

Stránky určené pre vývojárov sú o niečo bohatšie na implementačné detaily, ktoré stoja za zmienku, keďže vývoj hier tvorí ústrednú tému tejto práce.

- **Tvorba novej hry** – Tvorba novej hry sa skladá z viacerých krokov. Užívateľ definuje novú hru iba názvom (vytvorí sa nová položka hry v databázy). V databázy, táto nová hra, bude mať iba názov. Následne je užívateľ presmerovaný na detail hry, kde pred tým, ako odošle hru administrátorovi na schválenie, musí vyplniť všetky potrebné

informácie (v podstate sa iba editujú, prázdne polia v databázovom zázname). Všetky editácie informácií sa dejú v modálnych oknách. Pri tom, ako sa prvotne vytvorí záznam v databázy o hre, sa tiež vygeneruje API kľúč pre hru a uloží sa nová doména (url adresa s hrou na portáli) do zoznamu povolených domén. API kľúč je generovaný pomocou funkcie *md5()*, ktorej je dané na vstup id hry a následne sa vygenerovaný reťazec ešte skonkatenuje so samotným identifikačným číslom hry.

- **Nahrávanie súboru s hrou a obrázkov** – na nahrávanie súborov a obrázkov sú použité klasické Nette formuláre. V callbacku priradenému formuláru sa následne predajú hodnoty a súbory do modelu, kde sa vytvoria adresárové štruktúry podľa identifikačného čísla hry. Napríklad, pre hru s identifikačným čísom 1 sa vytvorí adresár pre herné súbory v zložke *upload/game/1* a pre obrázky *upload/images/1*. Po vytvorení adresárovej štruktúry sa do týchto adresárov presunú uploadnuté súbory. Uploadnuté obrázky, sú tiež pomenované podľa svojho identifikačného čísla, ktoré sa nachádza v databázi v tabuľke screenshots. Pri nahrávaní hry sa okrem spomenutých krokov kontroluje, či zip s hrou obsahuje index.html súbor, ktorý sa zobrazuje hráčom.

- **Kontrola domén** – zapnutie/vypnutie funkcie kontroly domén sa vykonáva pomocou Ajaxu, kedy sa na server posiela požiadavka, o uloženie príznaku, či je kontrola domén zapnutá, alebo vypnutá. Požiadavka sa mení vždy po prepnutí radio tlačidla v panely kontroly domén. Samotné povolené domény sa už pridávajú klasickým textovým Nette formulárom.

- **Štatistiky -** vizualizácia štatistík je implementovaná za pomoci knižnice Chart.js. Vývojár má možnosť nastaviť si typ zobrazovaných informácií, dátum od kedy a dátum dokedy chce dané informácie získať. Získanie informácií sa deje asynchrónne pomocou Ajaxu, kedy sa najprv posiela na server požiadavka, aké informácie, alebo presnejšie aké štatistiky je potrebné získať. Následne server pošle odpoveď, s už konkrétnymi štatistikami pre dané zvolené obdobie. Počas získavania informácií zo serveru je užívateľovi zobrazený načítací pás, aby mal prehľad o tom, že sa niečo deje.

- **Proces zverejnenia hry -** proces zverejnenia hry pre hráčov prebieha v niekoľkých fázach. V prvom kroku, kedy vývojár práve vytvoril hru, sa hra nachádza vo fáze editovania. Všetky príznaky v zázname hry (published, approved, refused) sú nastavené na 0. Admin v tejto fáze, nemá právo hru schváliť, ani odmietnuť. Po vyplnení všetkých informácií (informačné panely daných sekcií budú mať modrú farbu a nie červenú, ako to je pri chýbajúcich informáciách), je vývojárovi dovolené poslať hru na schválenie administrátorovi. Hre sa nastaví príznak published. Po odoslaní hry vývojárom má administrátor právo hru schváliť na publikovanie (nastavenie príznaku approved), alebo hru odmietnuť (nastavenie príznaku refused). O stave danej hry, je vývojár informovaný tým, že panel s hrou vo vývojárskej konzole je farebne odlíšený + je prítomná ikona, ktorá tiež informuje o tomto stave.

5.2 Implementácia aplikačného rozhrania

API je implementované ako Javascriptová knižnica, pre použitie na klientskej strane, ale jeho hlavná časť je implementovaná a vykonávaná na serverovej strane. Na strane serveru teda prebieha všetka komunikácia s databázou, ukladanie hodnôt do databáze, ich získanie z databázy, prihlasovanie a odhlasovanie užívateľa a celkovo všetka práca s API. API na

servery je implementované za pomoci PHP a je využitý rámec Nette. Keďže je Nette použitý aj pre implementáciu portálu a samotné Nette postačuje na tvorbu tohto API, nebolo potrebné rozmýšľať nad alternatívnym rámcom, alebo čistým PHP. Odoslanie odpovede sa deje za pomocou funkcie JsonResponse, ktorá je súčasťou rámca. Táto funkcia teda odošle odpoveď v JSON formáte, z dôvodu, aby odpoveď bola čitateľná pre klientskú stranu. V spomenutej Javascriptovej knižnici, sa nachádza iba samotné vytváranie požiadavku, ktorý sa odosiela na server, a taktiež spracovanie hodnôt, ktoré prídu v odpovedi od serveru. Požiadavka obsahuje parametre, podľa typu a charakteru volanej funkcie API a token, ktorý bol vytvorený pri inicializácií na serveri a následne predaný na klientskú stranu. Tieto dáta sa prenášajú v tele POST požiadavku a na serveri je teda možné k ním pristúpiť aj pomocou *$_POST["menoParametru"]*.

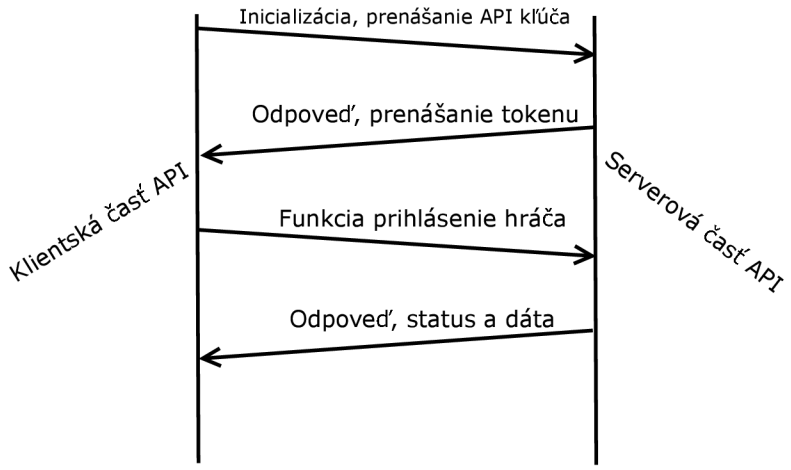
Činnosť API prebieha nasledovne. V prvom rade je potrebné si vytvoriť na klientskej strane objekt API. Následne na to, je možné zavolať inicializačnú funkciu, ktorá ako parameter na svoj vstup berie API kľúč získaný z vývojárskej konzoly (jeho tvorba bola popísaná vyššie v implementačných detailoch webovej platformy). Tento kľúč je odoslaný na server, a tam je zisťované, či existuje hra s takým API kľúčom, a či táto hra má povolené požiadavky z adresy, z akej požiadavka prišla. Ak všetko prebehne v poriadku, teda hra existuje a doména je povolená, tak sa vytvorí nový záznam v tabuľke dbSession, ktorá nahrádza klasické session, keďže CORS z jednej domény na druhú nepodporuje prenášanie cookies. Do tohto záznamu sa vygeneruje špeciálny unikátny token, identifikátor hry, vloží sa tam čas, adresa odkiaľ požiadavka prišla a v prípade prihlasovania sa sem vkladá aj identifikátor užívateľa. Token je tvorený pomocou funkcie *md5()*. Táto funkcia na svoj vstup berie skonkatenovaný API kľúč s adresou z ktorej požiadavka prišla, s aktuálnym časom, s identifikačným číslom hry a s identifikačným číslom dbSession. Po vytvorení *md5* reťazca je tento reťazec následne ešte skonkatenovaný s identifikačným číslom dbSession. Následne sa vygenerovaný token pošle naspäť do klientskej časti. Ostatné požiadavky, už samotné API funkcie, sú potom tvorené nielen z parametrov konkrétnych funkcií, ale aj tohto tokenu. Získanie informácií pre štatistiky a ich uloženie sa tiež deje počas inicializácie. Po inicializácií je už možné využívať funkcionalitu API. Všetky funkcie na serveri testujú zhodu adresy, ktorá bola uložená pri inicializácií s aktuálnou adresou, z ktorej prišla nová požiadavka. V prípade nezhody, je vrátený chybový kód 100. V prípade že sa adresy zhodujú, tak je vrátený špecifickejší kód a ďalšie možné dáta.

Všetky funkcie vracajú pole, kde na indexe *status* sa nachádza kód, ktorý značí ako operácia na serveri skončila (presne tak, ako bolo navrhnuté a popísané v kapitole o návrhu riešenia). Okrem toho, v prípade že sa očakávajú dáta a operácia skončila úspechom, tak na indexe *result* sa tieto dáta budú nachádzať. K tomuto poľu je možné pristúpiť v callbacku, ktorý bol predaný API funkcií. Okrem toho do tohto callbacku je možné predať tzv. kontext, alebo referenciu na objekt, ktorý chceme v callbacku využívať.

V API sú implementované nasledujúce funkcie (podrobnú dokumentáciu je možné nájsť v prílohe C):

- **initialize()** – funkcia slúžiaca na inicializovanie API na strane servera.
- **login()** – funkcia slúžiaca na prihlásenie hráča.
- **logout()** – funkcia slúžiaca na odhlásenie hráča.
- **isLogged()** – funkcia na zistenie, či je hráč prihlásený.
- **postScore()** – funkcia na vloženie hodnoty skóre do tabuľky.

- **getUserBestScore()** – funkcia na získanie najvyššieho skóre daného prihláseného hráča.
- **getScores()** – funkcia na získanie prvých X miest zo skóre tabuľky.
- **getAchievements()** – funkcia na získanie odmien a informácie o tom, či dané odmeny vlastní, alebo nevlastní aktuálne prihlásený hráč.
- **unlockAchievement()** – funkcia na získanie odmeny.
- **store()** – funkcia na uloženie hodnoty do definovaného úložiska.
- **load()** – funkcia na získanie odmeny z definovaného úložiska.



Obr. 5.1: Príklad komunikácie častí API.

5.3 Implementácia demonštračnej hry

Hra je implementovaná za pomoci rámca Phaser, ktorý bol predstavený v kapitole venovanej použitým technológiám. Celá hra je rozdelená do niekoľkých Javascriptových súborov. Javascriptové súbory v zložke *js/rooms*, obsahujú prototypy jednotlivých objektov herných obrazoviek, teda hlavné menu, obrazovka so skóre tabuľkami, obrazovka s odmenami a samotný level. V týchto obrazovkách sa vykresľujú jednotlivé grafické prvky, ako je text, obrázky a iné. V zložke *js/objects* sa zase nachádzajú súbory obsahujúce prototypy dôležitejších herných objektov, v prípade tejto práce je to predovšetkým objekt pexeso karty a objekt pexeso hracej plochy. Ostatné súbory, ktoré sa nachádzajú mimo týchto zložiek slúžia na všeobecnejšie účely. Za zmienku stojí Javascriptový súbor *boot.js*, ktorého jediná úloha je prichystanie hry pred samotným načítaním herných zdrojov (napríklad nastavenie módu zobrazenia, zarovnania...). Súbor *loader.js* slúži na načítanie všetkých herných zdrojov, teda obrázkov a prípadne zvukov. Celá hra sa spúšťa z *index.html* súboru. V tomto html súbore sa nachádzajú iba definície nevyhnutných css štýlov, inkludovanie Javascriptových súborov, základné html tagy (napríklad pre nastavenie titulku hry, ikony...) a spúšťanie *boot.js* súboru, ktorý sa stará už o všetko ostatné spojené s hrou.

Kapitola 6

Testovanie

Táto kapitola sa venuje testovaniu výsledného riešenia, predovšetkým testovania výslednej API knižnice. V prvom rade sa pojednáva o návrhu testovania, kde sú zhrnuté dôvody, prečo bol daný druh testov zvolený a následne na to je popísaný v ďalšej kapitole priebeh a výsledky zvoleného testovania.

6.1 Návrh testovania

Pri návrhu testovania bolo nutné si dopredu premyslieť, aké aspekty a časti tejto bakalárskej práce sú testovateľné, do akej miery sú testovateľné, a či vybraná časť na testovanie je vhodná vzhľadom na charakter práce. Po dôkladnom premyslení boli brané do úvahy tieto typy testovania:

- Testovanie užívateľského rozhrania hráčmi
- Testovanie užívateľského rozhrania vývojármi hier
- Testovanie užívateľského rozhrania administrácie
- Testovanie pochopenia API vývojármi hier (či je API pochopiteľné a ľahko integrovateľné do hry)
- Testovanie funkcionality API

Keďže práca nie je zameraná primárne na hráčov, a aj keď ich požiadavky do istej miery rieši, tak testovanie užívateľského rozhrania hráčmi bolo úplne vynechané. Testovanie užívateľského rozhrania administrácie bolo vynechané kvôli takmer totožným dôvodom, ako to bolo pri hráčoch. Testovanie užívateľského rozhrania vývojármi a testovanie, toho, či je API ľahko pochopiteľné a ľahko integrovateľné do hry boli prvými veľmi dobrými kandidátmi pre výber formy testovania. Bohužiaľ, ako sa ukázalo, tak proces tvorby hry je natoľko zložitý a komplexný, že testovacie vzorky, v podobe vývojárov hier, bolo nemožné nájsť. HTML5 hry sa tvoria, aj keď sa tak zdať nemusí, z drvivej väčšiny na komerčné účely. Autor hry, vývojár, vyvíja hru s cieľom predať jej exkluzívnu, alebo neexkluzívnu licenciu väčšej spoločnosti, ktorá si do hry integruje svoje vlastné API sama, alebo s pomocou vývojára. Spoločnosť sama ďalej hru distribuuje na svojich distribučných kanáloch. Z toho dôvodu, nebolo reálne si myslieť, že sa nájdu vhodný kandidáti na tento typ testovania, keďže aj samotný vývojár hry často stráca licenčné práva k hre. Kvôli tomu, autor tejto bakalárskej práce vytvoril svoju vlastnú testovaciu hru, ktorej návrh a implementácia ale

bola už popísaná v predchádzajúcich kapitolách. Keďže, táto testovacia hra ale nepokrýva funkcionalitu API na 100%, a okrem toho samotné testovanie API iba pomocou hry, by bolo až priveľmi zdĺhavé, bol vytvorený špeciálny Javascriptový test.

Súbory tohto Javascriptového testu pozostávajú z html súboru, ktorý slúži na vizualizáciu prebiehajúcich výsledkov a navrhnutej Javascriptovej testovacej knižnice. Táto knižnica obsahuje volania jednotlivých API funkcií a pokrýva všetku testovateľnú funkcionalitu. Po načítaní html súboru v prehliadači sa spustí funkcia *runTests()*, ktorá zavolá inicializačnú funkciu API. Funkcia vo svojom callbacku vráti hodnotu zo servera, *status* položku ktorá predstavuje návratový kód operácie. Práve táto hodnota je následne porovnávaná s očakávanou hodnotou. Podľa úspešnosti testu, sa vypíše pomocou Javasciptu do html súboru jeho stav, a aj čas, ktorý bol potrebný na túto operáciu (rozoberať nejako tento čas nemá zmysel, keďže je to hlavne v réžií servera) a následné zavolá ďalšiu funkciu API. Takto sa otestujú všetky funkcie API, presnejšie všetky možné návratové kódy. Testuje sa napríklad inicializácia s nesprávnym API kľúčom, inicializácia so správnym kľúčom, prihlásenie neexistujúceho užívateľa... Testovací skript je súčasťou priloženého CD/DVD nosiča.

6.2 Priebeh a výsledky testovania

Na základe navrhnutých testov bolo následne vykonané testovanie. Testovalo sa v prvom rade pomocou testovacieho skriptu. Testy prebiehali najprv na localhoste, kedy hra s Javascriptovou API knižnicou, a aj implementovaný portál so serverovou časťou API sa nachádzali na localhoste. Keď testy na localhoste skončili úspechom, tak bol webový server presunutý na reálny webový hosting pod určitou doménou, ale testovací skript zostal na localhoste. Keď skončil aj tento test úspechom, tak sa presunul aj testovací skript na tú istú doménu. Takto sa otestovalo API v rôznych podmienkach. Všetky testy skončili úspechom. Obrázok náhľadu na testovací skript je možné vidieť na obrázku.

Test12 - Unlock nonexisting achievement: **passed** with status **1** in 241miliseconds

Test13 - Unlock existing achievement: **failed** with status **2** in 173miliseconds

Test14 - Unlock already owned achievement: **passed** with status **2** in 176miliseconds

Test15 - Load from nonexisting storage: **passed** with status **1** in 146miliseconds

Test16 - Load from existing storage but nonexisting value: **failed** with status **0** in 223miliseconds

Obr. 6.1: Ukážka testovacieho skriptu a jeho vyhodnocovania

Po úspešných testoch pomocou testovacieho skriptu bola vytvorená demonštračná hra a API sa testovalo už pomocou nej. Najprv sa testovalo podobne ako s testovacím skriptom, kedy hra bola uložené spolu s webovým serverom na localhoste, potom sa webový server presunul na hosting a následne v poslednom teste bola aj hra presunutá na hosting s doménou. Po skončení týchto testov, ktoré skončili úspechom bolo API považované za overené.

Kapitola 7

Záver

Cieľom tejto bakalárskej práce bol návrh a realizácia modernej webovej platformy pre tvorbu hier spolu s návrhom a implementáciou API, ktoré sprístupňuje funkcie webovej platformy vývojárom hier. Webová platforma je na serverovej strane implementovaná skriptovacím jazykom PHP za pomoci rámca Nette. Na klientskej strane je využité HTML, CSS, Javascript a pre účely responzivity je použitý rámec Bootstrap. Súčasťou práce je taktiež jednoduchá Pexeso hra, ktorá demonštruje funkcionalitu API a je implementovaná v Javascripte pomocou rámca Phaser.

Webová platforma je plne použiteľná pre hráčov hier, ktorý si prišli na platformu zahrať nejakú hru a súťažiť s inými hráčmi v najvyššom skóre, alebo získavať odmeny hraním hier. Ale hlavne je určená pre vývojárov týchto hier, ktorý vďaka API, majú možnosť využívať funkcie a komponenty webovej platformy. Vývojár hry má na platforme možnosť definovať nové hry, definovať rôzne komponenty týmto hrám, ako sú skóre tabuľky, odmeny, cloudové úložiská. Ďalej má možnosť kontrolovať domény, na ktorých sa hra môže nachádzať, má možnosť prehliadať štatistiky prístupov k hre, vytvárať a editovať informácie o hre, jej popis, spravovať grafické materiály a samotné herné súbory na platforme. Administrátor platformy má potom možnosť schvaľovať vývojármi odoslané hry, a spravovať hráčov a vývojárov.

Testovanie riešenia prebiehalo za pomoci testovacích nástrojov, vytvorených v rámci tejto práce. Testovalo sa hlavne API funkcionalita pomocou špeciálne navrhnutého Javascriptového testovacieho skriptu a po úspešnom vyhodnotení boli vykonané testy aj za pomoci demonštračnej hry. Všetky testy skončili úspechom.

Platforma má určite veľký potenciál. Môže byť jednak využitá ako multifunkčný herný portál určený aj pre hráčov, aj pre vývojárov, alebo ako osobné portfólio vývojára s možnosťou správy spomínaných funkcií a komponentov. Platforma je ľahko rozšíriteľná a v budúcnosti sa dá ľahko rozširovať o ďalšie komponenty v rámci API, alebo ďalšiu funkcionalitu v rámci platformy. Do budúcna je možné implementovať rôzne chatovacie miestnosti, systém automatického vytvárania zápasov, správu herných reklamných kampaní a mnoho iných.

Literatúra

[1] *Apple Developer - Game Center*. [online]. 2016 [cit. 2016-01-20]. Dostupné z: https://developer.apple.com/game-center/.

[2] *Facebook Developers - Game Services*. [online]. 2016 [cit. 2016-01-20]. Dostupné z: https://developers.facebook.com/docs/games/services.

[3] *Google Developers - Play Games Services*. [online]. 2016 [cit. 2016-01-20]. Dostupné z: https://developers.google.com/games/services/.

[4] *O službě Greenlight*. [online]. 2016 [cit. 2016-01-18]. Dostupné z: https://steamcommunity.com/workshop/about/?appid=765&section=faq.

[5] *Phaser - Desktop and Mobile HTML5 game framework*. [online]. 2016 [cit. 2016-02-14]. Dostupné z: http://phaser.io/.

[6] *Steam*. [online]. [cit. 2016-01-18]. Dostupné z: http://store.steampowered.com/.

[7] *Steamworks documentation*. [online]. [cit. 2016-01-18]. Dostupné z: https://partner.steamgames.com/documentation/api.

[8] *W3School - AJAX Introduction*. [online]. 2016 [cit. 2016-02-15]. Dostupné z: http://www.w3schools.com/ajax/ajax_intro.asp.

[9] *W3School Online Web Tutorials*. [online]. 2016 [cit. 2016-02-15]. Dostupné z: http://www.w3schools.com/.

[10] Grundl, D.: *Nette Framework - zvyšte svoji produktivitu*. [online]. 2009-03-10 [cit. 2016-02-12]. Dostupné z: https://www.zdrojak.cz/clanky/nette-framework-zvyste-svoji-produktivitu/.

[11] Gutmans, A.; Bakken, S. S.; Rethans, D.: *Mistrovství v PHP 5*. Brno: CP Books, 2005, ISBN 802510799x.

[12] Hallock, A.: *Clay.io blog*. [online]. 2016 [cit. 2016-01-25]. Dostupné z: https://clay.io/blog/.

[13] Hossain, M.: *CORS in action - creating and consuming cross-origin APIs*. ISBN 16-172-9182-X.

[14] Čápka, D.: *MVC architektura*. [online]. 2013 [cit. 2016-02-12]. Dostupné z: http://www.itnetwork.cz/navrhove-vzory/mvc-architektura-navrhovy-vzor/.

Prílohy

Zoznam príloh

A Obsah CD **37**

B Návod na inštaláciu **38**
B.1 Inštalácia 38

C Dokumentácia k API **39**
C.1 Funkcie 39
C.2 Príklady 41

D Ukážky z implementácie **42**

Príloha A

Obsah CD

Adresárová štruktúra CD

- *src* - Zdrojové súbory
  - *web* - Webový portál
  - *api* - Javascriptové API
  - *game* - Demonštračná hra
  - *test* - Testovací skript
  - *database* - Databázový skript
  - *demo* - Demo aplikácia s už prichystanými nastaveniami, obsahujúca hru, jej materiály a funkčné API. (pre ľahšiu inštaláciu a demonštráciu)
- *text* - Text práce

Príloha B

Návod na inštaláciu

Keďže aplikácia využíva rámec Nette a ten má isté požiadavky, tak server, na ktorom sa bude aplikácia nachádzať musí tieto požiadavky splňovať. Tieto požiadavky je možné overiť Requirements-Checker skriptom, viac informácií na stránke s oficiálnou dokumentáciou[1]. Ďalej je nutné mať v internetovom prehliadači povolený Javascript pre správny chod platformy a vôbec pre spustenie HTML5 hier. Demonštračná hra vyžaduje taktiež splnené určité požiadavky, v prvom rade je to existencia canvas tagu, znova viac informácií v repozitáry rámca[2].

B.1 Inštalácia

1. V prvom rade je nutné v phpMyAdmin, alebo inom, spustiť skript databaza.sql, ktorý sa nachádza na priloženom CD v priečinku database. Tento skript vytvorí potrebné tabuľky a naplní ich ukážkovými dátami.

2. Ďalej je nutné nahrať obsah priečinku demo (obsahuje demo aplikáciu s hrou a API, pre ľahkú demonštráciu) na server a upraviť súbor v app/config/config.local.neon s prístupovými údajmi k databáze.

3. Pre správne fungovanie vyvinutého aplikačného rámca je nutné ďalej nastaviť adresu, na ktorú sa bude API dotazovať. Ak je napríklad domovská stránka webu načítaná po zadaní adresy www.example.com, tak práve túto adresu treba nastaviť do API súboru, riadok 8 premenná server v súbore api/api.js. Okrem toho, je to potrebné nastaviť ešte v totožnom súbore nachádzajúci sa v upload/games/1/js/api.js pre správne fungovanie demonštračnej hry.

[1] https://doc.nette.org/cs/2.3/requirements
[2] https://github.com/photonstorm/phaser#requirements

Príloha C

Dokumentácia k API

C.1 Funkcie

- **initialize(key[, callback[, context]])** – funkcia slúžiaca na inicializáciu API na strane servera.
  - **Parametre funkcie:** API kľúč získaný z vývojárskej konzole (z detailu hry), callback, ktorý je vykonaný po získaní odpovede, referencia na objekt, ktorý chceme v callbacku využívať.
  - **Vracia:** data['status'] je 0 v prípade úspechu, 1 v prípade, že doména nie je povolená a 2 v prípade že hra neexistuje.
- **login(username, password[, callback[, context]])** – funkcia slúžiaca na prihlásenie hráča.
  - **Parametre funkcie:** meno a heslo prihlasovaného hráča, callback, ktorý je vykonaný po získaní odpovede, referencia na objekt, ktorý chceme v callbacku využívať.
  - **Vracia:** data['status'] je 0 v prípade úspechu, 1 v prípade, že je zadané nesprávne meno, alebo heslo a 2 v prípade, že užívateľ je už prihlásený. data['result'] vracia v prípade úspechu prezývku hráča.
- **logout([callback[, context]])** – funkcia slúžiaca na odhlásenie hráča.
  - **Parametre funkcie:** callback, ktorý je vykonaný po získaní odpovede, referencia na objekt, ktorý chceme v callbacku využívať.
  - **Vracia:** data['status'] je 0 v prípade úspechu, 1 v prípade, že užívateľ nie je prihlásený.
- **isLogged([callback[, context]])** – funkcia na zistenie, či je hráč prihlásený.
  - **Parametre funkcie:** callback, ktorý je vykonaný po získaní odpovede, referencia na objekt, ktorý chceme v callbacku využívať.
  - **Vracia:** data['status'] je 0 v prípade, že užívateľ je prihlásený, 1 v prípade, že užívateľ nie je prihlásený. data['result'] v prípade že je užívateľ prihlásený obsahuje jeho prezývku.
- **postScore(table, value)** – funkcia na vloženie hodnoty skóre do tabuľky.

  - **Parametre funkcie:** id tabuľky, do ktorej chceme vkladať skóre a hodnota skóre.
  - **Vracia:** data['status'] je 0 v prípade úspechu, 1 v prípade, že tabuľka neexistuje, 98 v prípade, že užívateľ nieje prihlásený.

- **getUserBestScore(table[, callback[, context]])** – funkcia na získanie najvyššieho skóre daného prihláseného hráča.
  - **Parametre funkcie:** id tabuľky, z ktorej chceme skóre získať, callback, ktorý je vykonaný po získaní odpovede, referencia na objekt, ktorý chceme v callbacku využívať.
  - **Vracia:** data['status'] je 0 v prípade úspechu, 1 v prípade, že tabuľka neexistuje, 98 v prípade, že užívateľ nieje prihlásený. data['result'] v prípade úspechu obsahuje najvyššiu skóre hodnotu užívateľa z danej tabuľky.

- **getScores(table, count[, callback[, context]])** – funkcia na získanie prvých X miest zo skóre tabuľky.
  - **Parametre funkcie:** id tabuľky, z ktorej chceme skóre získať, počet hodnôt, ktoré chceme získať, callback, ktorý je vykonaný po získaní odpovede, referencia na objekt, ktorý chceme v callbacku využívať.
  - **Vracia:** data['status'] je 0 v prípade úspechu, 1 v prípade, že tabuľka neexistuje. data['result'] v prípade úspechu obsahuje pole s užívateľmi a ich skóre.

- **getAchievements([callback[, context]])** – funkcia na získanie odmien a informácie o tom, či dané odmeny vlastní, alebo nevlastní aktuálne prihlásený hráč.
  - **Parametre funkcie:** callback, ktorý je vykonaný po získaní odpovede, referencia na objekt, ktorý chceme v callbacku využívať.
  - **Vracia:** data['result'] obsahuje odmeny.

- **unlockAchievement(achievement[, callback[, context]])** – funkcia na získanie-/odblokovanie odmeny.
  - **Parametre funkcie:** id odmeny, ktorú chceme získať, callback, ktorý je vykonaný po získaní odpovede, referencia na objekt, ktorý chceme v callbacku využívať.
  - **Vracia:** data['status'] je 0 v prípade úspechu, 1 v prípade, že odmena neexistuje, 2 v prípade, že užívateľ už odmenu vlastní, 98 v prípade, že užívateľ nieje prihlásený. data['result'] v prípade úspechu obsahuje danú získanú odmenu.

- **store(storage, value)** – funkcia na uloženie hodnoty do definovaného úložiska.
  - **Parametre funkcie:** id úložiska, do ktorého chceme ukladať hodnotu a samotná hodnota
  - **Vracia:** data['status'] je 0 v prípade úspechu, 1 v prípade, že úložisko neexistuje, 98 v prípade, že užívateľ nieje prihlásený.

- **load(storage[, callback[, context]])** – funkcia na získanie hodnoty z definovaného úložiska.

– **Parametre funkcie:** id úložiska, callback, ktorý je vykonaný po získaní odpovede, referencia na objekt, ktorý chceme v callbacku využívať.
– **Vracia:** data['status'] je 0 v prípade úspechu, 1 v prípade, že úložisko neexistuje, 2 v prípade, že hodnota v úložisku neexistuje, 98 v prípade, že užívateľ nieje prihlásený. data['result'] v prípade úspechu obsahuje užívateľovu hodnotu z úložiska.

C.2 Príklady

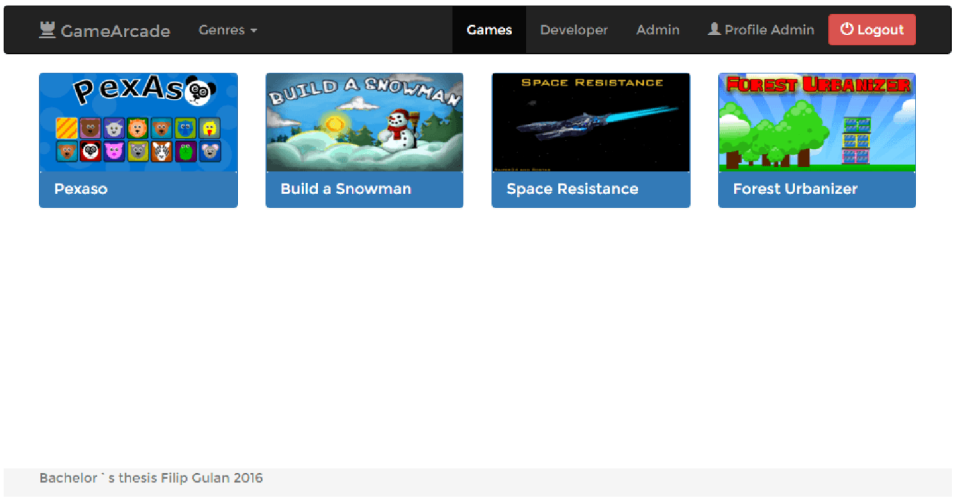
```
1  //Load value from storage
2  this.game.api.load(1, function(data, room){
3      if(data['status'] == 2) //value does not exist
4      {
5          //Store to storage
6          room.game.api.store(1, 1);
7      }
8      else //value exist
9      {
10         //Store to storage
11         room.game.api.store(1, parseInt(data['result']) + 1);
12     }
13 }, this);
```

```
1  //Get user best score from score table
2  room.game.api.getUserBestScore(1, function(data, room) {
3      room.game.playerBestScore = data['result']; //value from server
4      room.hudBestScore = room.game.add.text(room.game.width*(3/4) + 100,
           room.game.height, "Best Score: " + data['result'], { font: '20px
           Arial', fontWeight: 'bold', fill: "#000000", align: "center" });
5      room.hudBestScore.anchor.setTo(0.5,1);
6  }, room);
```

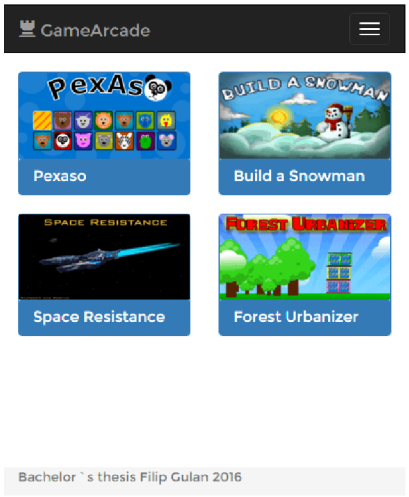
```
1  //Unlock achievement
2  this.game.api.unlockAchievement(2, function(data, game) {
3      if(data['status'] == 0) //achievement unlocked with success
4      {
5          achievement = game.add.text(game.width/2, game.height/2 - 100,
               data['result']['name'] + "\n Unlocked", { fontWeight: 'bold',
               font: '30px Arial', fill: "#000000", align: "center" });
6          achievement.anchor.setTo(0.5);
7          var fade = game.add.tween(achievement);
8          fade.to( { alpha: 0 }, 4000, "Linear", true);
9      }
10 }, this.game);
```

Príloha D

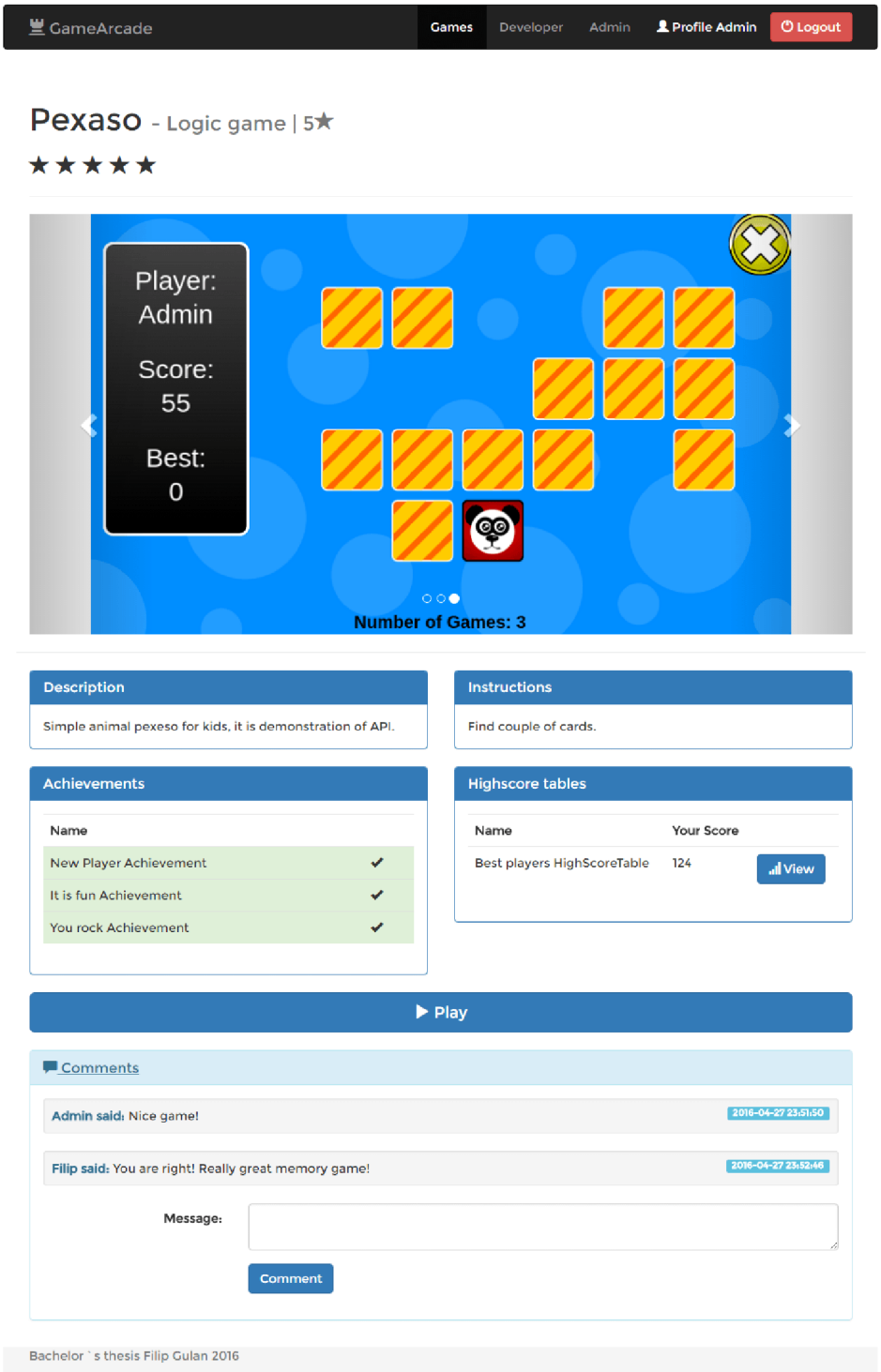
Ukážky z implementácie



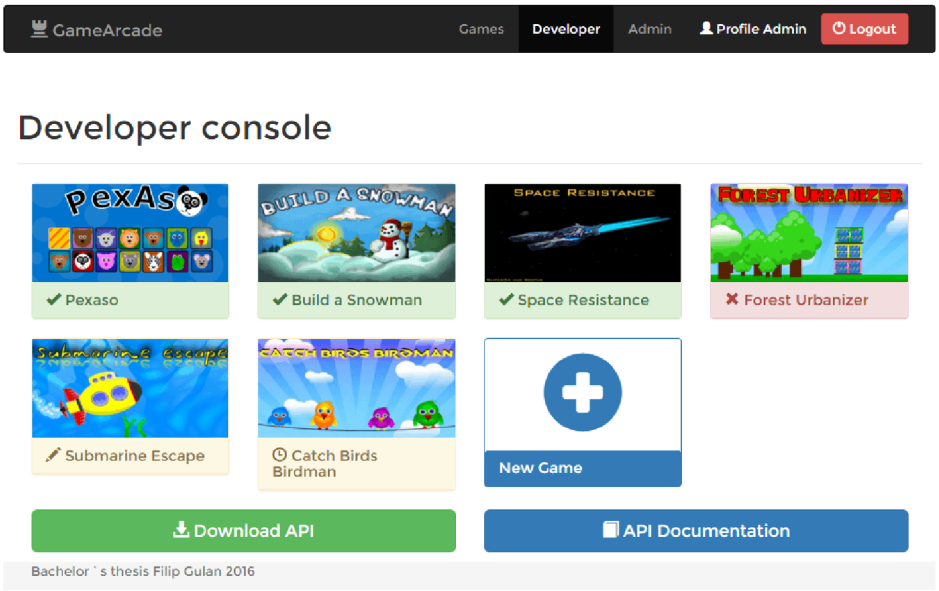
Obr. D.1: Ukážka obrazovky so zoznamom hier. Hore na lište vedľa textového loga sa nachádza filtrovanie podľa žánru.



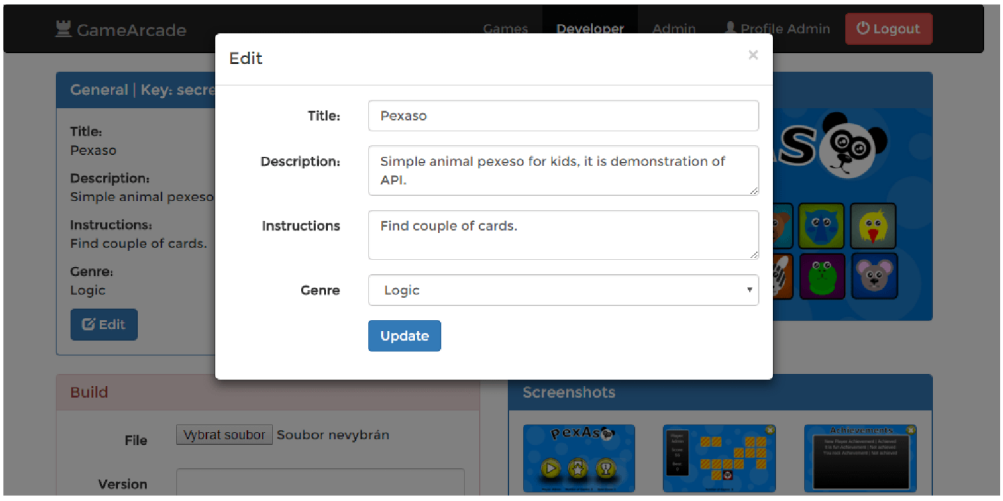
Obr. D.2: Ukážka zoznamu hier a jeho responzívnosti.



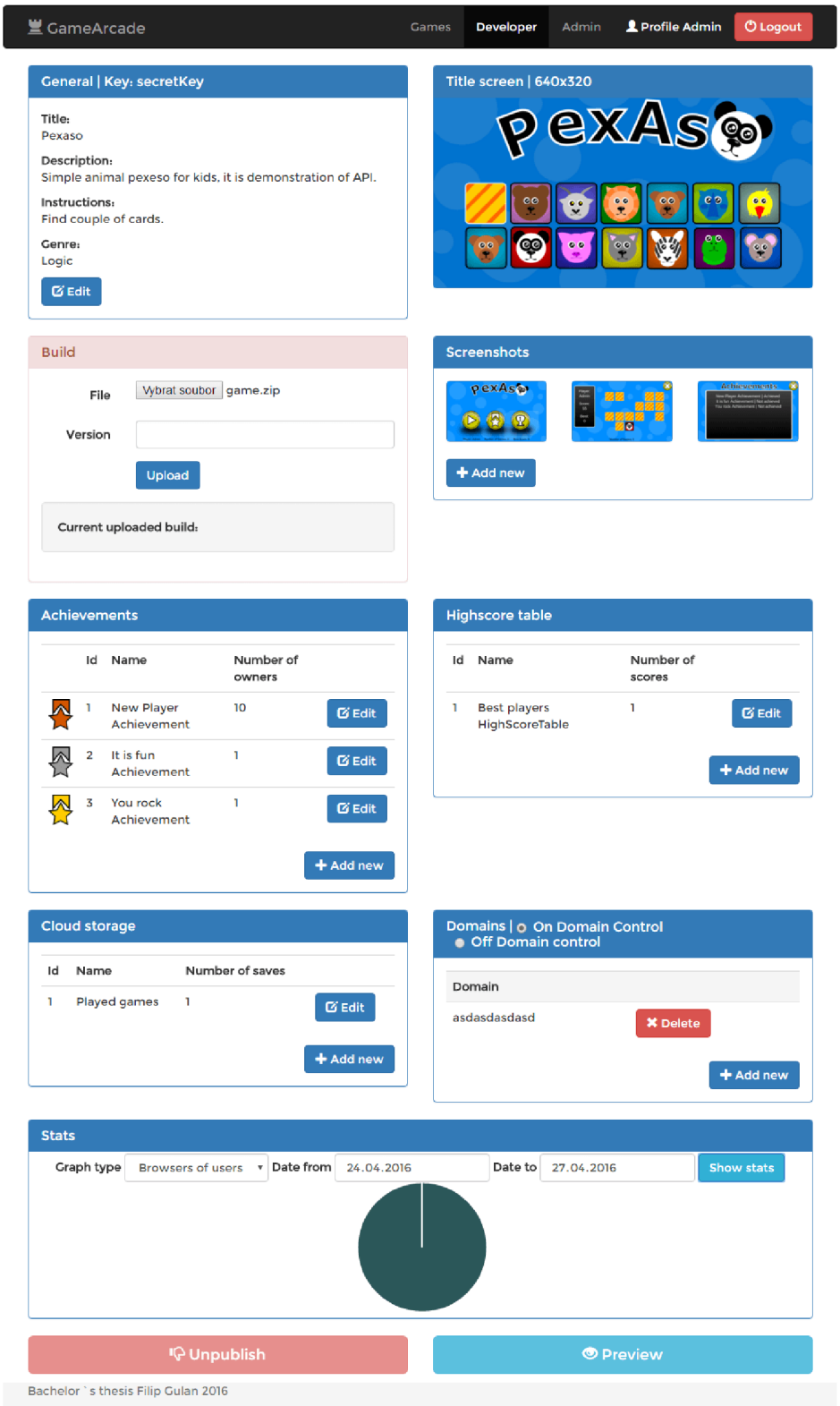
Obr. D.3: Ukážka detailu hry. Nachádzajú sa tu panely s s jednotlivými informáciami (napríklad panel s odmenami, tabuľkami...)



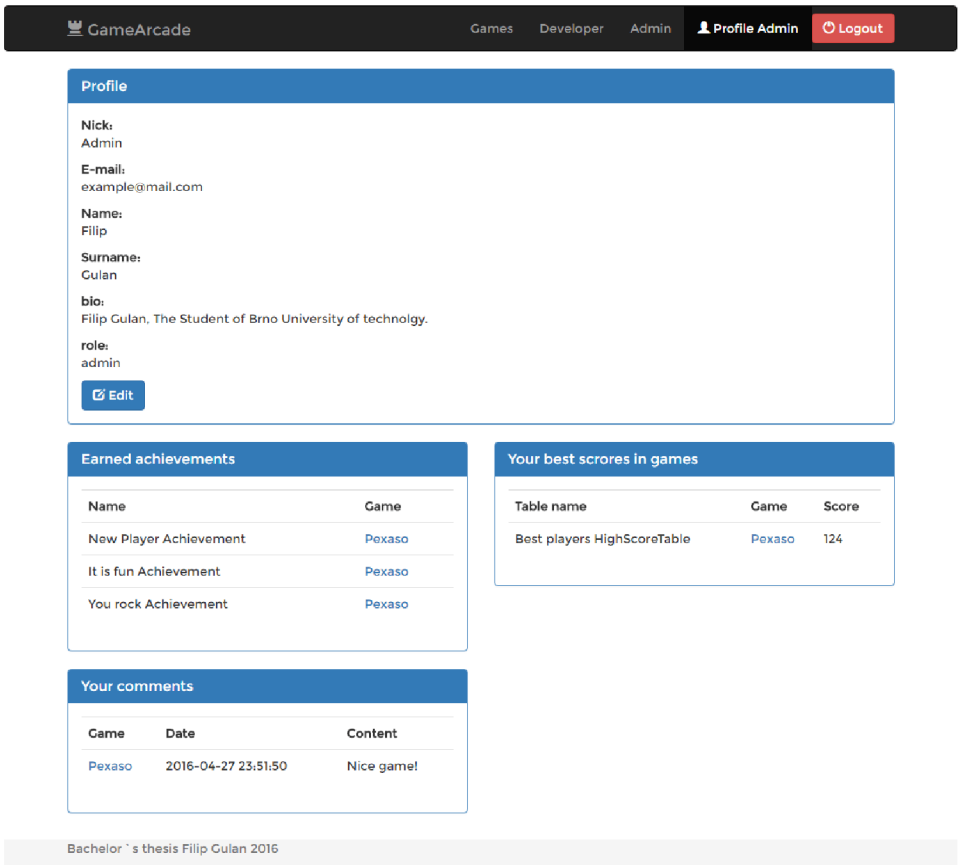
Obr. D.4: Ukážka vývojárskej konzole. Je možné vidieť hry v rôznom štádiu publikácie, ktorých panely sú farebne odlíšené.



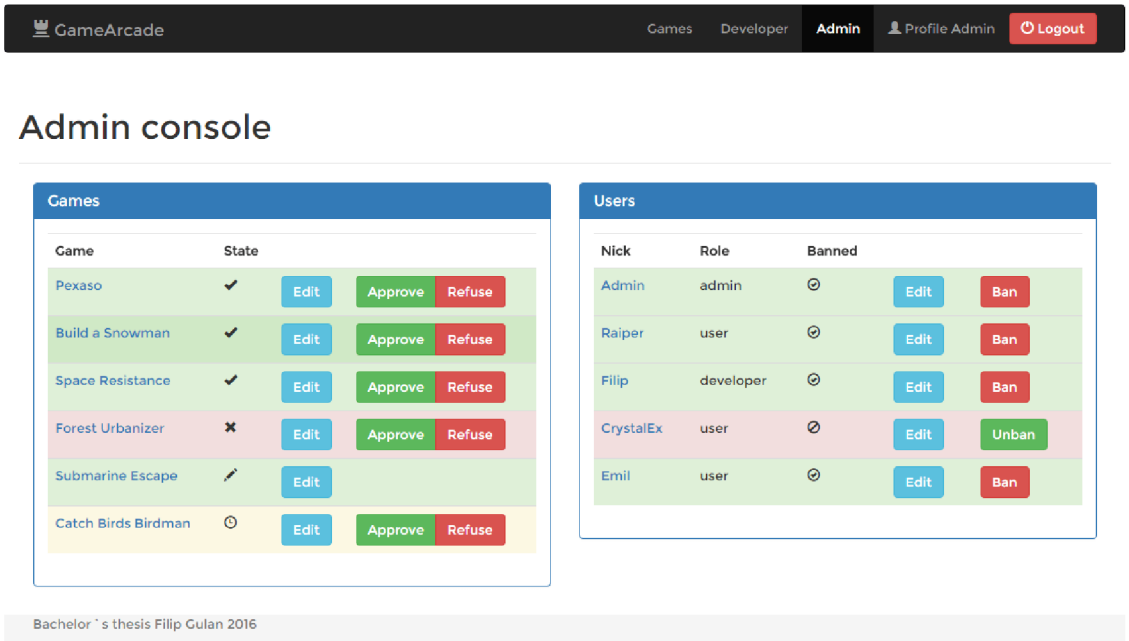
Obr. D.5: Ukážka úpravy informácií v detailu hry vo vývojárskej konzole pomocou modálneho okna.



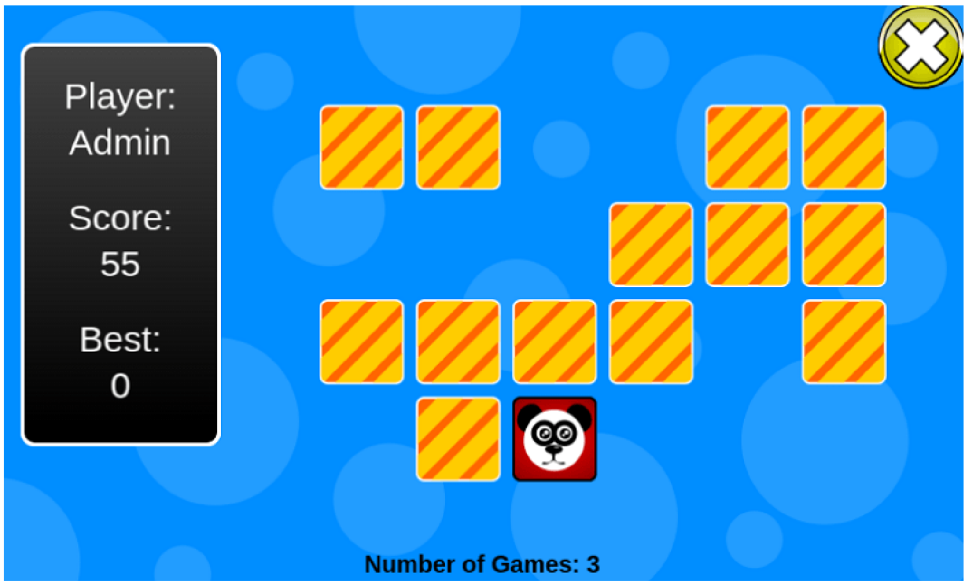
Obr. D.6: Ukážka detailu hry vo vývojárskej konzole. Je možné vidieť jednotlivé panely, ktoré logicky oddeľujú jednotlivé informačné a editačné sekcie.



Obr. D.7: Ukážka profilu užívateľa.



Obr. D.8: Ukážka administrátorskej konzole. Je možné vidieť farebne odlíšené hry v jednotlivých štádiách publikácie a užívateľov, taktiež farebne odlíšených, podľa toho, či majú prístup povolený, alebo zakázaný.



Obr. D.9: Ukážka levelu v demonštračnej hre. Naľavo je možné vidieť informačný panel, s informáciami, ktoré sú získané zo serveru pomocou implementovaného API.

Obr. D.10: Ukážka hernej obrazovky s odmenami. Obsahuje meno odmeny a informáciu o tom, či ju aktuálne prihlásený užívateľ získal.