VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

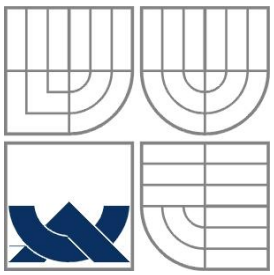
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO ANALÝZU HERNÍHO CHOVÁNÍ HRÁČŮ

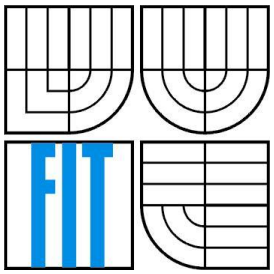
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE ONDŘEJ HOŠÁK
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO ANALÝZU HERNÍHO CHOVÁNÍ HRÁČŮ

SYSTEM FOR ANALYSIS OF PLAYER BEHAVIOR

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

ONDŘEJ HOŠÁK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VLADIMÍR BÁRTÍK, Ph.D.

BRNO 2008

Abstrakt

Bakalářská práce je zaměřena na analyzování herního chování hráčů a hodnocení jejich herní výkonnosti. Tato analýza probíhá na základě nahraných záznamů od jednotlivých uživatelů. Práce podrobně popisuje pravidla pro hodnocení těchto záznamů a určování herního chování. Pro vytvoření práce byla použita technologie ASP.NET, javascript a databáze MS SQL Server. Na internetovou prezentaci navazuje informační systém, který umožňuje jednoduchou správu obsahu prezentace a spravování uživatelských účtů. Během implementace designu byla věnována pozornost rozdílům mezi jednotlivými internetovými prohlížeči. Součástí práce je podrobná analýza a důsledný návrh, který dodržuje obecné standarty objektově-orientovaného programování.

Klíčová slova

ASP.NET, javasript , MS SQL, w3g parser, Dota Allstars, internetová aplikace

Abstract

Bachelor's thesis is focused on the analysis of players' behavior and ranking their gaming performance. This analysis is based on the replays which are uploaded by the users. The thesis describes in detail rules for rating the replays and analysing the players' behavior. There were used ASP.NET, javascript technologies and database MS SQL Server for programming. Internet presentation is followed by information system which enables simple management of presentation’s content and user’s accounts management. During design implementation attention was paid to differences among internet explorers. The thesis includes detailed analysis and projecting that meet general standards of object oriented programming.

Keywords

ASP.NET, javacript, MS SQL, w3g parser, Dota Allstars, web application

Citace

Hošák Ondřej: Systém pro analýzu herního chování hráčů. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Systém pro analýzu herního chování hráčů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením vedoucího pana Ing. Vladimíra Bártíka PhD., na základě použité literatury. Další informace mi poskytli Ing. Ladislav Ruttkay a hráči patřící do komunity hry Dota Allstars.Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……

Jméno Příjmení

12.5.2008

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu bakalářské práce panu Ing. Vladimíru Bártíkovi Ph.D za jeho připomínky, čas a trpělivost při tvorbě této práce. Dále bych rád poděkoval všem lidem, kteří mi pomohli s testováním vyvíjené aplikace.

Obsah

Obsah 1
Úvod 3
1 Dota All Stars 4
1.1 Popis mapy 4
1.2 Cíl hry 5
1.3 Důležité informace o hráči 6
1.3.1 Určení hodnocení (ratingu) záznamu 6
1.3.2 Vyhodnocení chování hráče 8
2 Analýza požadavků 10
2.1 Neformální specifikace 10
2.2 Funkce systému 10
2.2.1 Uživatelská část 10
2.2.2 Informační systém 11
2.3 Entity v systému 12
3 Použité technologie 14
3.1 Platforma .NET 14
3.2 ASP.NET 14
3.2.1 Popis práce platformy ASP.NET 15
3.2.2 Webové Formuláře 16
3.3 MSSQL Server (Express edition) 16
3.3.1 Jazyk SQL 17
3.4 Javascript 17
3.5 Kaskádové styly 17
4 Návrh Aplikace 19
4.1 Design 19
4.2 Struktura systému 19
4.2.1 Databáze 20
4.2.2 Aplikace 21
4.2.3 W3g Parser 22
5 Implementace 24
5.1 Použité nástroje 24
5.2 Design 24
5.3 Bezpečnost 24
5.3.1 Uložení hesel v databázi 25

5.3.2 SQL Injection 25
5.4 Aplikace 25
5.4.1 Autentizace 25
5.4.2 Přechod mezi stránkami 26
5.4.3 Naplnění obsahu stránky z databáze 26
5.4.4 W3g parser 26
5.4.5 Ošetření chybových stavů 26
6 Možnosti rozšíření systému 28
6.1 Vyhledávání a porovnávání mezi uživateli 28
6.2 Rozšíření editoru hrdinů 28
6.3 Nahrávání záznamů 28
6.4 Uložení více údajů ze záznamu 28
7 Závěr 30
Literatura 31
Seznam příloh 32

Úvod

Internetové aplikace jsou v dnešní době již nedílnou součástí našeho života. Díky přístupnosti internetu široké veřejnosti a zvyšující se počítačové gramotnosti jsou kladeny vyšší požadavky na internetové aplikace. Stále více je pozornost věnována dynamickým internetovým aplikacím namísto klasických statických html prezentací. Především se klade důraz na rychlost a bezpečnost.

Projekt, který jsem si zvolil realizovat jako svou bakalářskou práci, by mi měl dopomoci k rozšíření mých znalostí o technologii ASP.NET (implementované v platformě .NET). Jako implementační jazyk jsem si zvolil C#.

Námětem pro obsah internetové prezentace mi byla hra *Warcraft 3: The Frozen Throne*, konkrétně jedna z map pro tuto hru. Jako hráč bych ocenil systém, díky kterému bych si mohl vést statistiky o své herní činnosti a porovnávat se s ostatními hráči. Protože se mi však podobný systém nepodařilo nalézt, rozhodl jsem se jej vytvořit.

Cílem této bakalářské práce je tedy provést analýzu systému, který umožní uživatelům analyzovat jejich herní styl a sledovat také herní styl ostatních uživatelů. Další částí této práce je vytvoření nástroje pro načítání informací z formátu w3g (formát záznamů z počítačové hry *Warcraft 3*).

Při tvorbě této bakalářské práce byl velký důraz kladen zejména na design a kompatibilitu mezi jednotlivými internetovými prohlížeči. Dále se pak práce soustředí na správu profilu jednotlivých uživatelů v systému. Aplikace je doplněna informačním systémem, který usnadňuje práci administrátorům.

V první kapitole je popsána mapa *Dota AllStars* a její základní rysy. Čtenář se zde dozví, co je cílem mapy, jaké informace bude potřeba vyčíst z formátu w3g, jak se bodově hodnotí záznamy a na čem záleží analýza chování hráče.

Druhá kapitola se zabývá důkladnou analýzou požadavků na systém. Na základě specifikace jsou popsány jednotlivé části systému a jejich funkce. Po důkladném rozboru jsou pak vytvořeny modely systému, které nám při vývoji aplikace pomohou.

Třetí kapitola popisuje technologie použité při vytváření aplikace. Je zde stručně popsáno, jak celá architektura .NET pracuje, a také jsou uvedeny její přednosti.

Ve čtvrté kapitole je popsána struktura a návrh aplikace. Čtenář se seznámí se způsobem, jakým celá aplikace pracuje. Jsou zde postupně popsány jednotlivé vrstvy systému.

Pátá kapitola se zabývá implementací celého systému s důrazem na nástroj pro zpracování záznamů a zpracovaní těchto údajů pro statistiky a určení herního chování hráče.

Šestá kapitola nastiňuje možné rozšíření systému.

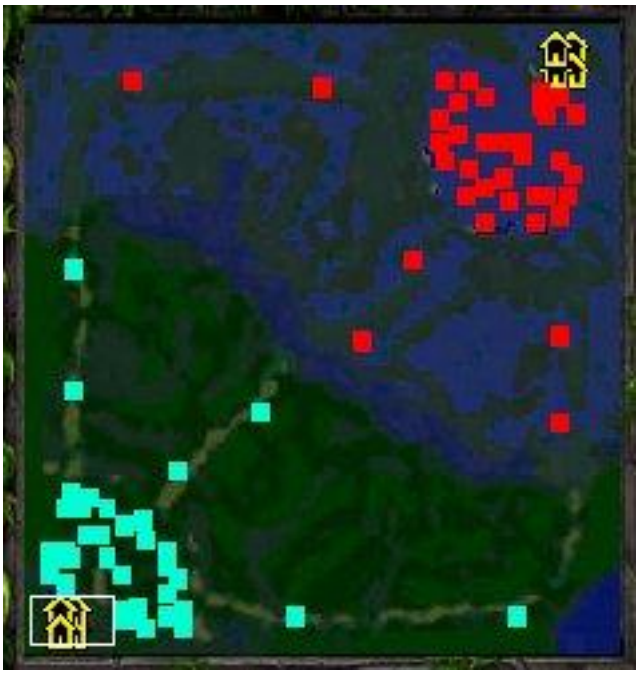
Obsahem závěrečné kapitoly je zhodnocení celé práce, zejména pak naplnění jejích cílů a shrnutí nalezených poznatků.

1 Dota All Stars

Dota (Defence of the Ancients) All Stars je vlastní mapou pro počítačovou hru z fantasy prostředí *Warcraft 3: The Frozen Throne* (Blizard Entertaiments 2003). Tato vlastní mapa byla vytvořena v roce 2005 člověkem s přezdívkou *Ice Frog*. Během následujících let se stala velice populární a v roce 2008 získala své zastoupení i na *WCG* (*World Cyber Games*), což je akce považována za olympiádu v počítačových hrách.

Momentálně je *Dota AllStars* ke stažení ve verzích 6.51 a 6.48b, přičemž verze 6.48b je pokládána za obecně vyrovnanou jak pro všechny strany, tak i pro každého hrdinu ve hře,a hraje se na turnajích a oficiálních akcích. Verze 6.51 je pak vývojovou verzí určenou pro testování nových předmětů, hrdinů nebo terénu mapy. Nová verze se objevují vždy v rozmezí 2 – 3 měsíců.

1.1 Popis mapy



Obrázek 1: Mapa Dota AllStars

V této mapě proti sobě stojí dvě strany - *Sentinel* (Ochránci) a *Scourge* (Horda). Obě strany mají svou základnu se svým divem. Ochránci mají jako div *The World Tree* a Horda má ve své základně *The Frozen Trone*. V základně se nacházejí obchody s různými předměty a také fontána, ve které se hráčův hrdina narodí. Základny mají celkem 3 vstupy a u každého z těchto vstupů se nachází budovy pro výrobu vojáků. Hráči nemají žádnou možnost, jak tuto výrobu ovlivnit (vojáci chodí ve skupinkách automaticky každých 50 vteřin). Tyto budovy lze pouze zničit.

Ještě před samotným začátkem hry se hráči rozdělují do týmů, které většinou čítají kolem pěti hráčů. Je ovšem možné hrát i v týmu o třech hráčích nebo tzn. duely (1vs1).

Po načtení mapy určí hráč, který je označen jako zakladatel hry, herní mód. Po určení módu hry si každý hráč vybere nebo je mu přidělen hrdina (viz příloha Herní módy). Je možné vybrat si z 90 hrdinů a volbu už nelze během hry měnit.

Mezi dvěma základnami vedou tři hlavní cesty. Součástí těchto cest jsou i obranné věže, kterých je na mapě celkem 18. Na každé cestě je po šesti věžích, přičemž obě bojující strany mají tři věže na své straně. Poslední věž vždy brání vstup do základny a je nejsilnější, směrem k soupeřově základně věže slábnou. Na mapě se nacházejí i další cesty, ale ty jsou bez věží. Prostor mezi cestami je vyplněn lesem, ve kterém je viditelnost slabší než na cestách. V lese se nacházejí stanoviště příšer tzv. *creep spots*, které lze využít k získání zkušeností a zlatých mincí. Dále se zde nacházejí ještě další obchody s předměty (*secrets shops*).

Po zvolení hrdiny si hráči v týmu rozdělí cesty, které budou hlídat. Toto rozdělení už je součástí strategie každého týmu stejně tak jako volba hrdinů, umožňuje-li ji mód hry.

1.2 Cíl hry

Cílem hry je zničit div v nepřátelské základně. Toho lze dosáhnou až po prolomení obrany na jedné ze tří výše zmíněných cest. K tomu, aby hráči mohli prolomit obranu na jedné z cest, je nutné získat zlaté mince a zkušenosti, ale také je třeba oslabit nepřítele. Toho lze dosáhnout například zabíjením jeho hrdinů a ochranou vlastních hrdinů (spoluhráčů) a vojáků. Pro protivníka jsou pak důsledkem těchto akcí nižší zisky zkušeností i zlatých mincí.

Zkušenosti lze získat za zabití nepřátelského hrdiny, nepřátelského vojáka nebo také za zabití neutrálních jednotek v lese. Zabití však není nezbytně nutné, protože k získání zkušeností stačí stát poblíž zmíněných postav. Pokud dojde k zabití, získávají se tak i zlaté mince.

Zkušenosti slouží k navýšení úrovně hrdiny, přičemž počáteční úroveň je 1 a maximální 25. Za každou získanou úroveň lze hrdinu naučit nové schopnosti nebo zvýšit úroveň schopnosti, kterou již hrdina umí. Každý hrdina se pak může naučit až pěti schopnostem, z nichž jedna je hlavní. Úroveň této hlavní schopnosti lze zvýšit pouze jednou během získání šesti úrovní.

Zlaté mince jsou potřeba pro nákup předmětů, které jsou důležitější než úroveň hrdiny, protože díky nim lze hrdinu posílit mnohem více než naučením nové schopnosti. Předměty dávají bonusy nebo určitým způsobem posilují schopnosti hrdinů, kteří je nesou. Některé předměty mají svou vlastní schopnost, kterou přidají hrdinovi. Každý hrdina u sebe může v jednu chvíli nést maximálně šest předmětů. Ve hře je pak přes 100 různých předmětů a jsou rozděleny do obchodů podle síly předmětu. Některé z nich se dají kombinovat a vznikají tak další mocnější předměty.

1.3 Důležité informace o hráči

Pro určování chování budeme využívat klasických statistik. U toho analyzování chování se nabízí možnost využití některé z metod pro získávání znalostí z databáze. Nicméně protože tyto metody se dají aplikovat až na velké objemy dat upustil jsem od implementace některé z metod v bakalářské práci. Získávání znalostí nabízí možnosti pro nalézání zajímavých modelů dat a vzorů. Nad implementací metod pro získávání znalostí by mělo smysl uvažovat až při rozšiřování systému, kde by bylo k dispozici více dat.

Informace, které potřebujeme pro tvorbu statistik uživatele a pro bodové ohodnocení hry jsou následující:
Hraný hrdina
Dosažená úroveň hrdiny
Nakoupené předměty
Zabití ostatních hrdinů (*Kills*)
Počet smrtí
Počet zabitých nepřátelských vojáků včetně neutrálních jednotek (*Creep kills*)
Počet zabití vlastních vojáků (*Denies*)

1.3.1 Určení hodnocení (ratingu) záznamu

Hodnocení se odvíjí od aritmetického průměru hodnocených údajů, které jsme definovali jako důležité pro tvorbu statistiky. Jediný údaj, se kterým nepracujeme, je hraný hrdina. Ostatní údaje ohodnotí body od jedné do deseti, přičemž každý z údajů má stejnou váhu. Zde je nutné uvést, že při stanovování bodových zisků u jednotlivých hodnot jsem vycházel jednak ze svých zkušeností s hraním této hry a dále z porady s hráči, kteří byli komunitou určeni jako jedni z nejlepších hráčů v České republice. Meze hodnot byly určeny po dohodě s dotazovanými hráči.
Jednotlivé údaje se hodnotí následujícím způsobem:

Dosažená úroveň hrdiny – Bodový zisk vychází z délky hrané hry. Průměrný hráč dokáže získat jednu úroveň za dobu 3 minut. Tedy jako střední hodnotu stanovíme čas 3 minuty a bodový zisk 5 bodů (střed intervalu bodových hodnot). Pokud hráč dosáhne úrovně 25 (maximální úroveň), započte se jako konečný čas, za který byla získána tato úroveň. Poté vyčíslíme průměrný čas potřebný pro získávání úrovní jako podíl konečného času a získané úrovně. Po získání průměrného času se provede bodové ohodnocení podle těchto pravidel :

- Průměrný čas je vyšší nebo roven 4 minutám : zisk 0 bodů
- Průměrný čas je nižší nebo roven 2 minutám : zisk 10 bodů
- Mezi těmito hodnotami roste bodový zisk lineárně

Nakoupené předměty – K nakupování předmětů jsou potřeba peníze a u každého předmětu je známá jeho cena. Proto se hodnocení odvíjí od cen předmětů, které hráč nakoupil během hry. Střední hodnota je nákup předmětů za 350 zlatých mincí během jedné minuty. Opět určíme průměrnou hodnotu získaných zlatých mincí za minutu výpočtem podílu ceny všech předmětů a času hry. Následně ohodnotíme podle těchto pravidel:

- Průměrná hodnota získaných zlatých mincí je menší rovna 200: zisk 0 bodů
- Průměrná hodnota získaných zlatých mincí je větší rovna 500: zisk 10 bodů
- Mezi těmito hodnotami roste bodový zisk lineárně

Zabití ostatních hrdinů – Tento údaj vypovídá o síle hrdiny v dané hře. Čím silnější hrdina byl, tím více protihráčů zabil, přičemž průměrný hráč během hry zabije devět nepřátelských hrdinů. Proto tuto hodnotu určíme jako střední hodnotu. Nabízí se možnost spojení tohoto údaje s počtem smrtí hráče a teprve tuto dvojici pak hodnotit, ovšem každý z těchto údajů vypovídá o jiných schopnostech hráče. Zabití vyžaduje úplně jiné schopnosti hráče než jsou schopnosti potřebné pro přežití. Proto jsem se rozhodl tyto údaje hodnotit odděleně a to následujícím způsobem:

- Počet zabitých hrdinů je vyšší nebo roven 15 : zisk 10 bodů
- Počet zabitých hrdinů je menší nebo roven 3 : zisk 0 bodů
- Mezi těmito hodnotami roste bodový zisk lineárně

Počet smrtí – Počet smrtí hráče ve hře je odvozen od jeho schopnosti útěku z nepřátelské léčky nebo přežití v boji. Vyžaduje znalost mapy a všech jejich temných míst, což jsou místa, kde vás protihráč nevidí. Zvládnutí technik úniku/přežití není jednoduché, a proto jsem zde volil jiný systém bodového hodnocení.

- Pokud hráč ani jednou během hry nezemře bude oceněn 10 body.
- Za každou smrt se odečte 1,4 bodu
- Nelze získat méně než 0 bodů (7 smrtí)

Počet zabitých nepřátelských vojáků a neutrálních jednotek – Za každou zabitou jednotku získává hráč odměnu v podobě peněz. Tato odměna je v rozmezí 40-50 zlatých mincí. I přesto, že při přidělování výše odměny hraje svou roli náhoda, můžeme toto zanedbat, jelikož zisk zlatých mincí hráče je ovlivněn i dalšími faktory (dostává zlaté mince průběžně). Pokud hráč zemře, je mu odečtena určitá suma (podle úrovně). Z těchto důvodů není údaj hodnocen z pohledu peněz ale pouze jako informace o tom, jak se hráč soustředí na zabíjení nepřátelských vojáků. Průměrný hráč dosáhne 125 zabitých vojáků během hry a tato hodnota je tedy střední hodnotou.

- 180 a více zabití : zisk 10 bodů
- 70 a méně zabití : zisk 0 bodů
- Opět roste bodový zisk mezi těmito údaji lineárně

Počet zabití vlastních vojáků – Tato hodnota vypovídá o tom, nakolik je hráč schopen zabít, resp. obětovat vlastní jednotky. Pokud totiž hráč zlikviduje vlastní jednotku, pak za ni protivník získá

méně zkušeností a je také připraven o možnost získat za její zabití zlaté mince. Jelikož je poměrně obtížné dosáhnout zabití vlastní jednotky, zvolili jsme zde nižší střední hodnotu a to číslo 20.

- 0 zabití vlastních vojáků : zisk 0 bodů
- 40 zabití vlastních vojáků : 10 bodů
- Mezi hodnotami roste bodový zisk lineárně

1.3.2 Vyhodnocení chování hráče

Analýzu chování hráče je možné rozdělit na dvě části. První z nich je analýza agresivity hráče a druhou je analýza stylu hry. U agresivity rozlišujeme tři stavy: agresivní, neutrální a pasivní. Analýzu agresivity je potřeba provádět podle poměru mezi počtem zabitých hrdinů a počtem smrtí ve hře. Čím je poměr vyšší, tím je hráč méně agresivní, protože bojuje jen když si je jist, že vyhraje. Samozřejmě jsou podstatné i hodnoty, ze kterých tento poměr vychází. Rozdíl si vysvětlíme na následujícím příkladu.

Mějme hráče číslo jedna, který skončí s výsledným skóre 14:7, a mějme druhého hráče, jehož koncové skóre je 2:1. Poměr je u těchto hráčů sice stejný, ale každý má naprosto odlišné chování. Hráč číslo jedna hraje agresivněji a je větším přínosem pro tým, tím že se účastní většiny bojů na mapě. Naproti tomu hráč číslo dva hraje znatelně pasivněji. Bojuje spíše jen v případě, když je donucen situací.

Vycházíme-li z těchto poznatků, pak za agresivního hráče považujeme člověka, který má vysoké skóre. Pokud hráč dosáhne záporného skóre (např. 11:13), tak již není přínosem pro tým, ale i přesto lze jeho chování považovat za agresivní. Skóre, kterého dosáhne neutrální hráč, se pohybuje v celkovém součtu kolem hodnot 10 (7:3, 5:5).

Dále se pak ale musíme zaměřit na samotný poměr mezi počtem zabití a počtem smrtí. Agresivní hráči se většinou nesoustředí na poměr, ale spíše na počet zabitých hrdinů. I když si nejsou jistí vítězstvím v souboji, stejně bojují. Proto agresivní hráči nemívají příliš vysoký poměr, většinou se pohybuje do hodnoty 1,5. Neutrální hráči nevyhledávají souboje přímo, ale nejsou-li nutně přesvědčeni o vlastní prohře, bojují. Naproti tomu pasivní hráči se spíše snaží vyhnout přímým bojům a spíše hledají již zraněného či oslabeného protivníka, kterého snadno zabijí. Z toho důvodu dosahují lepších poměrů v konečném skóre.

Na základě výše zmíněných skutečností sestavíme pravidla pro určení chování hráče.

- Má-li hráč konečné skóre v součtu větší než 15 a jeho poměr je do 1,5, pak ho považujeme za agresivního.
- Neutrálně hrající hráč má konečné skóre v součtu patřící do rozmezí 5 -15 a jeho poměr je v rozmezí 1,5 – 2.

- Ostatní hráči jsou považování za pasivní.

Nyní následuje analýza stylu hry. U stylu hry rozlišujeme dvě možnosti: farmář a bojovník. Jako farmáře určíme hráče, který má vysoký počet zabitých nepřátelských vojáků. Farmář se většinou snaží příliš nezapojovat do soubojů, spíše v poklidu zabíjet jednotky a našetřit co nejvíce zlatých mincí. Poté nakoupí silné předměty a začne bojovat. Takové typy hráčů většinou mívají velmi vysoký počet zabitých nepřátelských vojáků a vysoký poměr mezi zabitím a smrtí (14:2). Proto jako farmáře určíme hráče, který splní následující 2 podmínky. Všechny ostatní označíme jako bojovníky.

- Má více než 160 zabitých nepřátelských vojáků.
- Poměr zabití a smrtí dosahuje hodnot větších než 5

2 Analýza požadavků

2.1 Neformální specifikace

Aplikace slouží k uložení a hodnocení záznamů o herní aktivitě. Celá aplikace je rozdělena do dvou základních částí - internetová prezentace a informační systém.

Internetová prezentace rozlišuje celkem dva druhy uživatelů, tedy registrované a neregistrované, a je rozdělena do čtyř částí. První částí je fórum, které je s právem čtení přístupné pro všechny uživatele. Pouze registrovaní uživatelé pak mají možnost psát příspěvky a nová fóra mohou zakládat výhradně uživatelé s administrátorskými právy. Další částí prezentace je seznam hrdinů, kteří jsou v nabídce mapy *Dota AllStars*. Obdobnou součástí prezentace je také stránka se seznamem všech předmětů v mapě *Dota AllStars*. Poslední částí prezentace je stránka o uživateli. Obsahem této stránky jsou informace o uživateli vyplněné při registraci a statistické údaje získané ze záznamů, přičemž svou stránku uživatele mají pouze registrovaní hráči, ale viditelná je každému návštěvníkovi prezentace. Dále je zde také seznam všech záznamů nahraných daným uživatelem s možností jejich prohlížení.

Informační systém, do kterého mají přístup pouze uživatelé s administrátorskými právy, slouží ke spravování celého systému. Primárně je určen k obsluhování fóra a administraci údajů o uživatelích. Nabízí také možnost editovat informace o hrdinech a předmětech.

U uživatelů je potřeba evidovat uživatelské jméno, heslo, přezdívku a email jako povinné údaje. Jako nepovinné údaje pak jméno, příjmení a ICQ. U záznamů potřebujeme znát hraného hrdinu, dosaženou úroveň u hrdiny, seznam předmětů, počet zabitých nepřátelských hrdinů, počet smrtí hrdiny hraného hráčem, počet zabitých nepřátelských jednotek, počet zabitých vlastních jednotek, čas hraní a příznak vítězství.

Je vhodné, aby byla aplikace kompatibilní s co největším počtem internetových prohlížečů, minimálními požadavky jsou kompatibilita s prohlížeči Internet Explorer a Mozilla Firefox.

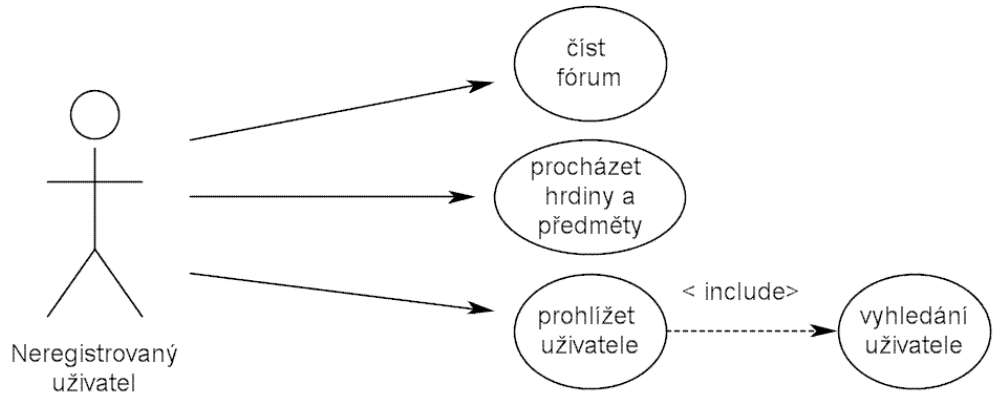
2.2 Funkce systému

2.2.1 Uživatelská část

Neregistrovaný uživatel

Za neregistrovaného uživatele pokládáme jakéhokoliv návštěvníka internetové aplikace. Neregistrovaný uživatel má právo číst fórum, prohlížet stránku se seznamem hrdinů i stránku se

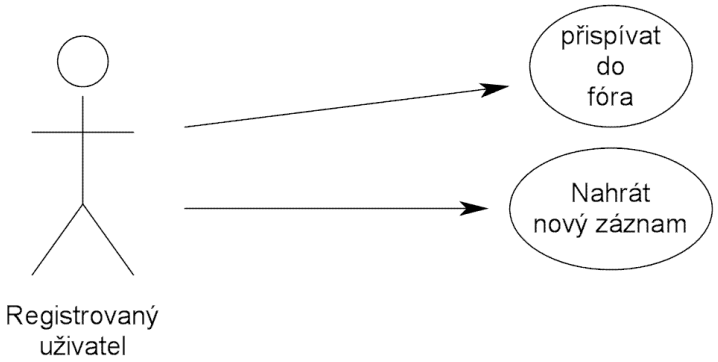
seznamem předmětů. Dále může neregistrovaný uživatel prohlížet záznamy a statistiky libovolného zaregistrovaného uživatele. Stejně tak může mezi uživateli vyhledávat. Má možnost se zaregistrovat.



Obrázek 2: USE-CASE diagram neregistrovaného uživatele

Registrovaný uživatel

Registrovaný uživatel je takový uživatel, který řádně vyplnil registrační formulář a správným způsobem se přihlásil. Má všechna práva jako neregistrovaný uživatel. Navíc ale může vkládat nové záznamy o herní aktivitě do svého profilu a to formou nahrání souboru (ve formátu w3g). Registrovaný uživatel může přispívat do fóra svými příspěvky a editovat svůj profil.



Obrázek 3: USE-CASE diagram registrovaného uživatele

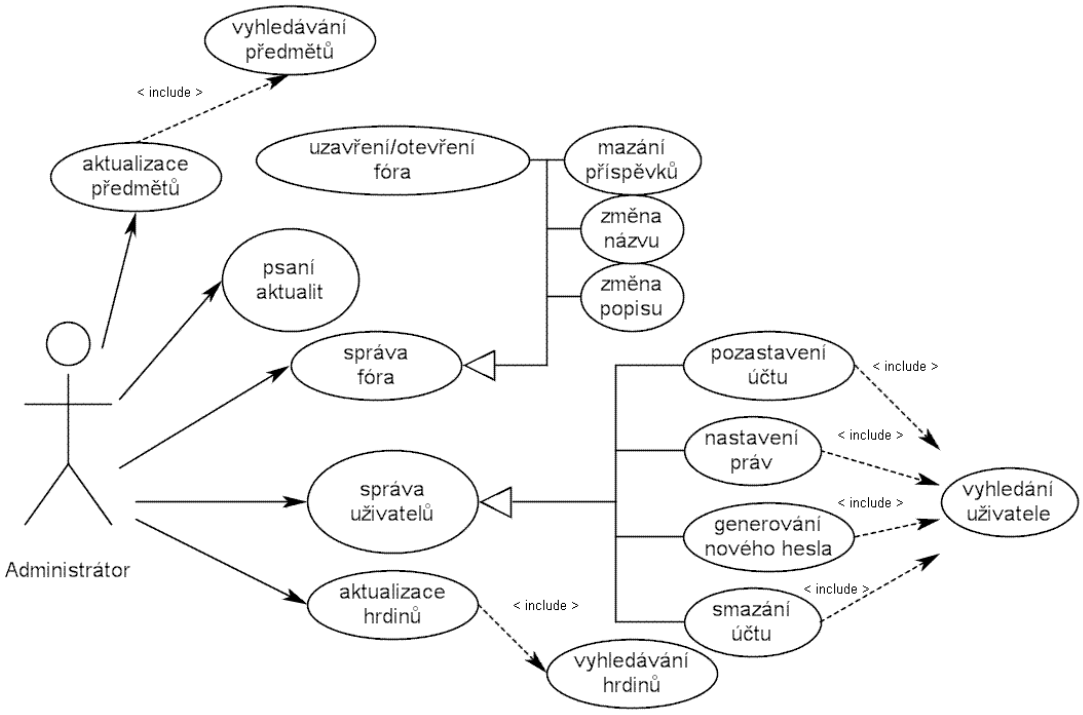
2.2.2 Informační systém

Administrátor

Administrátor má stejná práva jako registrovaný uživatel, ale navíc má umožněn přístup do informačního systému. Může získat svá práva od jiného administrátora a přitom si zachovat stejné uživatelské jméno i heslo. Jednou z hlavní náplní práce administrátora je udržování fóra. Každý administrátor má své fóra, o která se stará, což znamená, že mezi jeho pravomoci patří změna názvu a popisu fóra nebo uzavírání a otevírání pro psaní příspěvků. Je zodpovědný za obsah, a proto má také právo smazat libovolný příspěvek z fóra, jehož je moderátorem.

Další důležitou činností je spravování uživatelů. Administrátor může smazat účet libovolného uživatele, včetně administrátorů (pochopitelně vyjma sebe sama). Může účet pozastavit, což pro daný účet znamená, že ho lze stále prohlížet, ale jeho vlastník se nemůže na tento účet přihlásit. Tato akce by měla sloužit jako výstraha uživatelům zejména při nevhodném chování, jakým může být například používání nevhodných slova ve fóru nebo opakované nahrávání stejných záznamů. Další činností, kterou může administrátor provádět, je vygenerování nového hesla uživateli a zaslání na jeho email.

Administrátor se ještě stará o udržování aktuálních informací o hrdinech a předmětech. Posledním právem je psaní aktualit.



Obrázek 4: USE-CASE diagram pro administrátora

2.3 Entity v systému

Celý systém se skládá z entit *Actualities, Dues, Forum, Heroes, ShopDealers, SkillInfo, Skills, Taverns, UserReplay, UserReplaySkills, UserReplayItems, Users*. Každá z entit má svůj primární klíč ID, podle kterého jsou jednotlivé záznamy jednoznačně identifikovatelné.

Actualities (aktuality) slouží pro informování uživatelů o změnách, úpravách a novinkách v prezentaci nebo systému.

Forum a *Dues* (příspěvky) jsou entity, které jsou využívány pro obsluhu fóra stejně jako pro přispívání nebo čtení fóra. Entita *Forum* je v podstatě seznam všech fór vytvořených v systému. Obsahuje název, informaci o uzavřenosti/otevřenosti, datum vytvoření, popis a identifikaci administrátora, který se stará o dané fórum. Entita *Dues* je potom seznamem příspěvků do daného fóra.

Heroes, *Skills*, *SkillInfo* jsou entity, které slouží pro práci s hrdiny. *Heroes* je základní seznam všech hrdinů. U hrdinů potřebujeme evidovat jméno, povolání a jeho základní atributy i jejich přírůstky při získání vyšší úrovně a v neposlední řadě také jejich příslušnost k taverně. Jelikož každý hrdina má rozdílný počet schopností, většinou však čtyři, je potřeba tyto schopnosti evidovat zvlášť. Každá schopnost má několik úrovní (3 – 4) a na každé úrovni má rozdílné atributy. To je důvodem pro zavedení entity *SkillInfo*.

Taverns (taverny) jsou místa, kde se dají hrdinové najmout. Každá taverna patří k jedné z bojujících stran. Pomocí této entity pak dokážeme rozhodnout, kam daný hrdina patří.

ShopDealers (obchodníci) nabízejí jednotlivé předměty, které se dají koupit. Tato entita je důležitá pouze pro zařazení jednotlivých předmětů na stránce s předměty.

Users (uživatelé) jsou jednou z hlavních entit v systému. U uživatelů potřebujme evidovat uživatelské jméno, heslo, email a přezdívku jako povinné atributy. Mezi nepovinnými atributy jsou pak jméno, příjmení nebo ICQ. Pro účely systému evidujeme ještě atributy aktivity daného uživatele, resp. účtu, a souboru jeho práv.

UserReplay, *UserReplayItems*, *UserReplaySkills*. Po nahrání záznamu uživatelem je potřeba uložit informace ze záznamu pro pozdější analýzu. U záznamu ukládáme údaje, které jsou podstatné z hlediska analýzy (viz 1.3). *UserReplayItem* je seznam předmětů, které hráč nakoupil během hry. Jelikož těchto předmětů může být neomezený počet, je lepší vytvořit samostatnou entitu pro tento seznam. *UserReplaySkill* je dalším seznamem a to seznamem schopností. Znalost postupu hráče ve výběru schopností je důležitá nejen pro analýzu, ale také pro ostatní uživatele, kteří se budou chtít podívat, jak daný uživatel hraje za jednotlivé hrdiny. Celá situace je názorně zobrazena v ER-digramu, který lze nalézt v přílohách

3 Použité technologie

3.1 Platforma .NET

Pokud mluvíme o platformě .NET, většinou máme na mysli její infrastrukturu. Tato infrastruktura se skládá celkem ze čtyř částí. Jsou to .NET Framework , Microsoft Visual Studio, .NET Enterprise Servers a Microsoft Windows .NET. Infrastruktura platformy zahrnuje všechny technologie, které tvoří prostředí pro vytváření a spouštění aplikací. Ta část platformy, která umožňuje vytváření aplikací, se nazývá .NET Framework.

Systém .NET Framework je složen ze dvou částí: běhového systému CLR (*Common Language Runtime*) a knihoven tříd známých pod zkratkou BCL (*Base class library*). Knihovna je přehledně členěna do jmenných prostorů (každý obsahuje sadu tříd) rozdělených podle použití. K této knihovně mají přístup všechny programovací jazyky. Běhový systém si můžeme představit jako virtuální stroj, ve kterém pracují aplikační funkce platformy .NET a je jádrem celé infrastruktury. V tomto běhovém prostředí mohou být bez problémů spouštěny aplikace psané v různých programovacích jazycích. Jde o princip, který je někdy nazýván „spolupráce mezi jazyky“ (*cross-language interoperability*). Všechny jazyky, které chceme využít pro vývoj aplikací běžících na CLR, musí mít překladač dodržující specifikaci CLS (*Common Language Specification*). Jde o jakýsi souhrn pravidel a pokud překladač jazyka tato pravidla nedodržuje, pak není kompatibilní s platformou .NET. Mezi jazyky dodržující CLS patří např. C#, C++ .NET nebo Visual Basic .NET.

Společnost Microsoft se snažila usnadnit práci výrobcům překladačů, a proto vyvinula „mezijazyk“ MSIL (*Microsoft intermediate language*). Pro kompilaci programů pro systém .NET je vstupem překladačů zdrojový kód v daném jazyce a výstupem právě tento „mezijazyk“. Při prvním spuštění aplikace zajistí CLR její překlad do strojového kódu počítače pomocí překladače JIT (*just in time*). MSIL je tedy sám o sobě plnohodnotný jazyk podobný assembleru, vyjma krajních situací se ale sám o sobě k programování nepoužívá.

3.2 ASP.NET

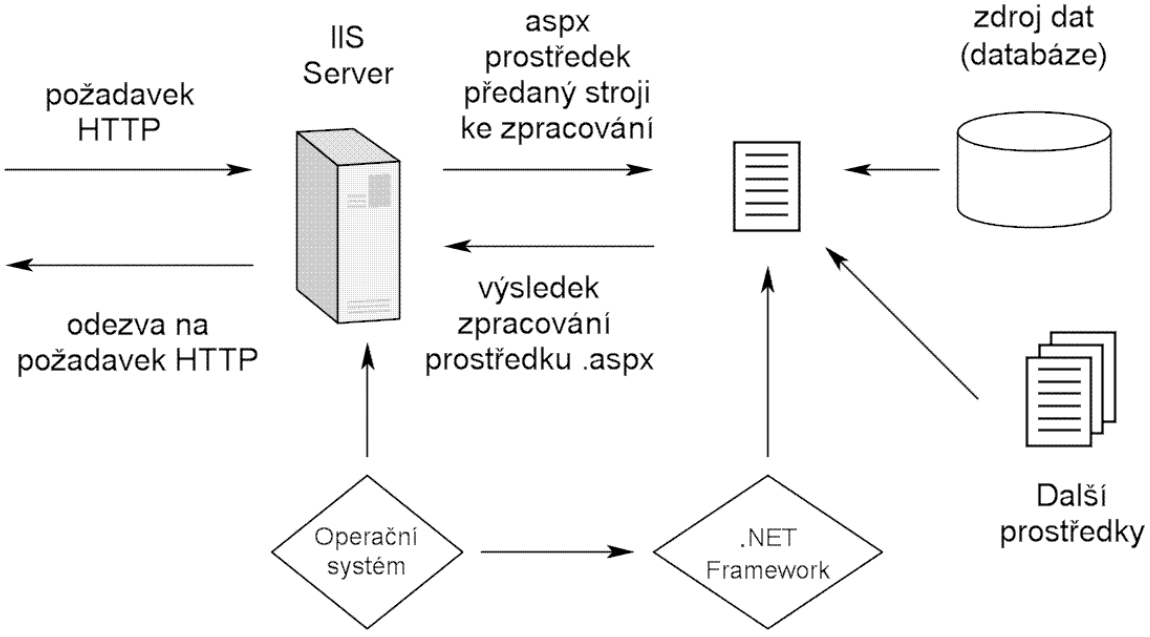
ASP.NET (Active server pages) je platforma pro vytváření, nasazování a běh internetových aplikací. Mezi její hlavní přednosti, oproti ostatním technologiím pro tvorbu interaktivních internetových stránek, patří zejména rychlost generování výstupu. Pro tvorbu v této platformě se využívají kompilované jazyky, přičemž nejznámějšími zástupci mezi těmito jazyky jsou C# a Visual Basic.NET.

Rozsáhlá podpora od strany Microsoftu a počet předpřipravených komponent, které jsou ekvivalentní s klasickými html prvky, činí tuto platformu velmi výhodnou co se týče rychlosti vývoje aplikace. Samozřejmě ASP.NET přichází s mnohými dalšími komponentami, které nahrazují některé html konstrukce (formuláře a jiné). Díky těmto pozitivům dané platformy je vidět, že se uplatní spíše při vývoji systémů, které jsou v porovnání s naší aplikací rozsáhlejší.

Užití pokročilejších programovacích jazyků, které se pro vytváření ASP.NET aplikací používají, má jeden nezanedbatelný klad, a tím je kompletní objektový model. ASP.NET nabízí přístup ke všem jeho komponentám a ovládacím prvkům na webových stránkách jako k objektům, jejich vlastnosti a funkce jsou tak srovnatelné s prvky určenými pro systém Windows, proto jsou stránky ASP.NET generující HTML často nazývány Web Forms.

3.2.1 Popis práce platformy ASP.NET

Jako každá jiná webová technologie má i ASP.NET svůj vlastní server, který na základě požadavků protokolu HTTP odesílá požadované informace zpět k uživateli. Tímto serverem je IIS (*Internet Information Services*). Tento server je podobný serveru *Apatche*, který využívá jazyk PHP. ASP a stránky jsou uloženy na server s příponou .aspx.



Obrázek 5: Pohled na technologii ASP.NET

Součástí souboru .aspx mohou být tyto části:

- Instrukce serveru pro zpracování stránky
- Kód v některém z jazyků, které jsou podporovány platformou ASP.NET
- Libovolný ovládací prvek (komponenty) platformy ASP.NET
- Prvky značkovacího jazyka HTML

3.2.2 Webové Formuláře

Celá funkčnost prostředí ASP.NET je založena na formulářové technologii a komunikaci mezi jednotlivými formuláři. V záhlaví každého zdrojového souboru stránky aspx musíme stanovit direktivy pro server. Direktivy jsou vloženy mezi značky <% a %> v následujícím formátu.

```
<%@ Page Language="C#" Title="Dota website Login"
    MasterPageFile="~/MasterPage.master"
    CodeFile="Default.aspx.cs"
    Inherits="_Default"
    AutoEventWireup ="true"%>
```

Bez těch direktiv se stránka nepočítá za platnou a její zpracování skončí chybou. Nyní si projdeme jednotlivé direktivy a jejich význam pro server.

Language – určuje serveru, v jakém z jazyků je psána třída pro obsluhu stránky

Title – titulek stránky

MasterPageFile – určuje stránku, do které patří právě generovaný obsah. Tato direktiva není povinná, pokud nepoužíváme systém *Master pages*.

CodeFile – informuje server, ve kterém souboru má hledat kód pro obsluhu stránky

Inherits – třída pro obsluhu stránky

AutoEventWireup – umožňuje automatické odesílání prvků na stránce. Například při vybrání možnosti z prvku <*asp:dropdownlist runat="server"*/> známém jako prvek <input type="*select*" /> z klasického HTML. Výchozí hodnotou je *false*. Pokud chceme hodnotu změnit, použijme tuto direktivu.

Každá z komponent, které je možné využít při tvorbě stránky, musí obsahovat direktivu runat="server", pokud chceme, aby ji zpracovával server. Pokud tato direktiva není neuvedena, pak se předpokládá, že je prvek zpracováván na straně klienta.

3.3 MSSQL Server (Express edition)

MS SQL Server je databázový nástroj vytvořený firmou Microsoft umožňující ostatním programům (na lokálním počítači nebo přes síť) ukládat, zpracovávat a vyhledávat větší objemy dat. Pro vývoj naší aplikace jsme využili jeho Express verzi, která je dodávána společně s instalací Microsoft Visual studio Express Edition. Systém řízení báze dat MS SQL Server je spolehlivý a robustní natolik, aby mohl tvořit databázovou vrstvu aplikací podnikových informačních systémů strategického významu. Celý systém obsahuje mnoho užitečných aplikací s grafickým uživatelským rozhraním usnadňujícím vývojáři jeho práci, tyto nástroje však nejsou dostupné v jeho Express verzi.

3.3.1 Jazyk SQL

V 70. letech minulého století probíhal ve firmě IBM výzkum relačních databází. Vývojáři potřebovali vytvořit sadu příkazů pro práci s těmito systémy, které by byly syntakticky podobné přirozené řeči. Vznikl tak jazyk SEQUEL, který byl později standardizován a přejmenován na SQL (*Structured Query Language*). Prvním standardizovaným jazykem bylo v roce 1986 SQL-86. Pro jeho nedostatky byl později upraven a vznikly postupně jazyky SQL-92 a SQL-99.

3.4 Javascript

Javasript je scriptovací jazyk se základním objektovým modelem, který se interpretuje na straně klienta. To znamená, že se jeho kód odešle společně s HTML dokumentem a jeho interpretaci zajišťuje internetový prohlížeč. Universální jádro javasriptu je obsaženo ve všech internetových prohlížečích. Tento jazyk se využívá zejména pro validaci formulářů, tzn. kontroluje obsah polí podle určitých pravidel a nepovolí odeslání formuláře, dokud nejsou všechna pole řádně vyplněna. Mezi jeho další využití lze uvést možnost dynamické změny vzhledu HTML prvků na stránce. Syntaxe javacriptu je velice podobná jazyku C, ale jedná se o objektový jazyk, který využívá především objektů prohlížeče a zabudovaných objektů.

Javascript má však i jednu podstatnou nevýhodu. Uživatel může zakázat svému prohlížeči interpretování javacriptu. Proto je důležité s touto možností počítat zejména při zpracování formulářů. Je vhodné je ošetřit pomocí kódu, který se zpracovává na straně serveru.

3.5 Kaskádové styly

Kaskádové styly slouží pro vylepšení vzhledu internetových stránek, protože samotný značkovací jazyk HTML neumožňuje ovlivnit vzhled prvků na stránce a ani to není jeho cílem. Z počátku vývoje internetových stránek se příliš nepočítalo s grafickým ztvárněním stránek. Ovšem s rostoucí oblibou internetu bylo pro firmy a tedy i vývojáře potřeba nějakým způsobem své dílo prezentovat a upozornit na něj.

Proto se do HTML jazyka začaly postupně přidávat atributy značek, které umožnily změnit vzhled jednotlivých prvků (barva, písmo apod.). Velkou nevýhodou tohoto způsobu ale byla nemožnost nastavit hodnotu nějakého atributu pro všechny prvky určitého typu a potřeba nastavovat atributy pro každý prvek jednotlivě. To vedlo ke vzniku značně nepřehledného kódu, jehož jakákoliv úprava znamenala nemalé úsilí pro vývojáře.

Jako reakce na tyto problémy vznikly v roce 1996 CSS (*Cascade Style Sheets*) neboli kaskádové styly. Na počátku se jim nedostávalo velké podpory ze strany internetových prohlížečů, protože starší verze nedokázaly plně spolupracovat s touto novou technologií.

Hlavní výhodou CSS je možnost oddělení kódu (obsahu) a grafického vzhledu stránky. Kaskádové styly se připojí ke stránce pomocí direktivy *<link rel="stylesheet" type="text/css" href="cesta k souboru" />*, která se vloží do hlavičky dokumentu HTML *<head></head>*. Po připojení stylů ke stránce je možné jednotlivým prvkům na stránce přiřadit třídu, jejíž vlastnosti se určují v souboru s kaskádovými styly. Tyto soubory lze opět využít při tvorbě jiných stránek .

4 Návrh Aplikace

4.1 Design

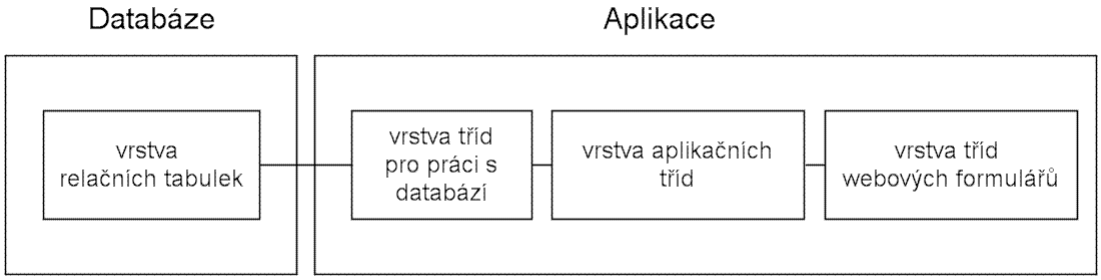
Vzhled prezentace je zasazen do prostředí hry *Warcraft 3*, aby ladil s obsahem stránek. Celý design je navržen tak, aby byl dobře zobrazitelný i při nižších rozlišeních obrazovky. V dnešní době stále používá ještě nemalá část uživatelů rozlišení obrazovky 800x600 pixelů. Z tohoto důvodu má celý design šířku 750px, aby byl dobře zobrazitelný i při tomto rozlišení.

Při navrhování designu je důležité dodržovat určitá pravidla, která se vztahují k dobré čitelnosti písma a barevných kombinací písma a pozadí. Barva písma by měla být dostatečně kontrastní proti pozadí. Dále se musí věnovat pozornost rozložení prvků na stránce, aby byla navigace pro uživatele jednoduchá a mohli se jednoduše orientovat v obsahu stránek.

Pro vytvoření stránek jsme použili systém *Master pages*, který nám usnadňuje rozdělení stránek na více částí. Systém *master pages* nám umožňuje vytvořit *master* stránku a v ní vyznačit místa, jenž se budou dynamicky měnit za běhu aplikace. Naše stránka *master* obsahuje hlavičku s obrázkem, který se dynamicky mění podle uživatelem vybraného obsahu. Dále obsahuje menu a zápatí stránky, přičemž tyto součásti jsou statické. Poslední částí *master* stránky je obsah, který se generuje podle uživatelova požadavku. Design byl koncipován tak, abychom mohli využít možností systému *Master pages*.

4.2 Struktura systému

Celý systém je rozdělen na několik na sobě v podstatě nezávislých částí. Celou situaci názorně popisuje následující obrázek.



Obrázek 6: Struktura systému

Systém navržený podle tohoto schématu přináší pro vývoj aplikace mnoho výhod, z nichž mezi hlavní patří dobrá strukturovanost a flexibilita. V budoucnu budeme moci aplikaci lehce upravovat nebo rozšiřovat, aniž bychom museli zasahovat do všech jejich částí. Změníme-li například databázový server, zasáhnou změny pouze vrstvu tříd pro práci s databází. Toto rozdělení systému

navíc umožňuje rozdělit aplikaci na více logických částí, na jejichž vývoji se mohou podílet různí lidé nebo týmy lidí. Stačí pouze, aby se domluvili a zachovali rozhraní mezi jednotlivými vrstvami. Jeden tým může mít na starosti práci s tabulkami a implementaci metod pro tyto třídy. Další tým pak využije tyto metody pro generování obsahu prezentace.

4.2.1 Databáze

Databáze obsahuje všechna potřebná data pro obsah prezentace a pro potřeby informačního systému. Databáze ale neobsahuje informace potřebné pro čtení záznamů ve formátů w3g.

4.2.1.1 Vrstva relačních tabulek

Pro lepší přehlednost a strukturovanost jsou data rozložená do více tabulek, které jsou navzájem propojené na základě cizích klíčů. Následuje seznam tabulek s jednotlivými sloupci a využitými datovými typy. Primární klíče jsou vždy označeny kurzívou.

Users – slouží pro uložení záznamů o uživatelích systému

Sloupce: *IDUser (Int)* , login (Varchar), password (Varchar), admin (Bit), name (Varchar), surname (Varchar), email (Varchar), DateCreation (Datetime), ICQ (Decimal), nickname (Varchar), ActiveFlag (Bit)

UserReplay – tato tabulka slouží pro uložení údajů získaných z nahraného záznamu

Sloupce: *IDReplay (Int)*, IDUser (Int), IDHero (Int), Rating (Float), DateUpload (Datetime), WinFlag (Bit), HeroKill (Int), Deaths (Int), CreepKills (Int), CreepDeny (Int), Lenght (Int)

UserReplaySkills – je seznamem schopností hrdiny, které byly vybrány hráčem během hry. Nemá primární klíč, slouží jako pomocná tabulka.

Sloupce: IDReplay (Int), IDSkill (Int)

UserReplayItems – je vytvořena pro podobné účely jako tabulka UserReplaySkills, ale ukládá záznamy o předmětech.

Sloupce: IDReplay (Int), IDItem (Int)

Skills – seznam všech schopností hrdinů, které se objevují v mapě Dota AllStars

Sloupce: *IDSkill (Int)*, Heroname (Varchar), SkillName (Varchar), Description (Varchar), Passive (Bit), Ultimate (Bit)

SkillsInfo – určena pro uložení informací o každé úrovni schopnosti

Sloupce: *IDSkillInfo (Int)*, SkillName (Varchar) , SkillLevel (Int), SkillDescription (Varchar), Cooldown (Int), ManaCost (Int)

Heroes – seznam všech hrdinů

Slopce: *IDHero (Int)*, Name (Varchar), HeroFunction (Varchar), Range (Int), PrimaryAttribute (Varchar), Speed (Int), DamageMin (Int), DamageMax (Int), StrenghtBase (Int), AgilityBase (Int), InteligenceBase (Int), StrenghtPerLvl (Int), AgilityPerLvl (Int), InteligencePerLvl (Int), IDTavern (Int)

Taverns – tato tabulka slouží pro zařazení hrdinů do skupin a do jedné z bojujících stran.

Sloupce: *IDTAvern (Int)*, TavernName (Varchar) , Sentinel (Bit)

Items – obsahuje všechny předměty, které se dají koupit.

Sloupce: *IDItem (Int)*, ItemName (Varchar), ItemDesription (Varchar), IDShopDealer (Int)

ShopDealers – pomocná tabulka pro zařazení předmětů do skupin. Toto zařazení vychází z mapy Dota AllStars.

Sloupce: *IDShopDealer (Int)*, ShopDealerName (Varchar), Recepices (Bit)

Actualities – aktuality na úvodní stránce prezentace

Sloupce: *IDActuality (Int)*, IDUser (Int), Text (Varchar), DateCreation (Datetime)

Forum – obsahuje všechna fóra (bez příspěvků)

Sloupce: *IDForum (Int)*, ForumName (Varchar), Moderator (Int), DateCreated (Datetime), ActiveFlag (Bit), ForumDescription (Varchar)

Dues – příspěvky pro jednotlivá fóra

Sloupce: *IDue (Int)*, ID ForumParent (Int), IDUser (Int), text (Varchar), DateCreated (Datetime), ActiveFlag (Bit)

4.2.2 Aplikace

4.2.2.1 Vrstva tříd pro práci s databází

Tato vrstva slouží pro práci s databází a je uzavřena do vlastního jmenného prostoru *db*. Je tvořena z abstraktních tříd, tzn. že nelze vytvořit instanci těchto tříd. Všechny třídy obsahují statické metody pro práci s databází. Jejich volání probíhá na základě jmen s příslušnými parametry.

Každá metoda komunikující s databází potřebuje vytvořit připojení k dané databázi. K tomu slouží různé třídy podle typu databázového serveru. My pracujeme s MS SQL Server a používáme třídu *SqlConnection*. Aby se tyto třídy mohly připojit k serveru, vyžadují připojovací řetězec (*Connection String*), jenž obsahuje údaje o umístění serveru, přihlašovacích údajích a typu autorizace.

Abychom při změně databázového serveru nemuseli v každé metodě měnit tento připojovací řetězec, umístíme ho do konfiguračního souboru naší aplikace. Tento soubor se jmenuje Web.config. Při každé inicializaci aplikace je tento připojovací řetězec načten statickou třídou cGlobal a do její vlastnosti *ConnectionString* je uložen připojovací řetězec. Tuto vlastnost potom využívají všechny metody, které tento řetězec vyžadují pro připojení k databázi. Z tohoto důvodu je změna databázového serveru velice jednoduchá, stačí pouze změna jednoho parametru v souboru Web.config.

4.2.2.2 Vrstva aplikačních tříd

Tato vrstva obsahuje třídy, jejichž instancí jsou objekty s informacemi o jednotlivých entitách v systému. Je uzavřena do jmenného prostoru *bc*. Třídy jsou pojmenovány podle použití, jako příklad

uvádím třídu *cForum*, která odpovídá stránce fórum v prezentaci. Každá z těchto tříd má několik konstruktorů, z nichž jeden vytvoří pouze prázdnou instanci této třídy. Ostatní potom podle parametru (ID, jméno hrdiny apod.) s využitím databázových tříd vytvoří instanci a naplní ji požadovanými daty. Jednotlivé třídy mají vlastní metody pro potřeby práce s instancemi těchto tříd, ale každá obsahuje metody *write* (zápis do databáze) , *update* (aktualizace databáze) a *delete* (mazání záznamů z databáze).

Užití tohoto objektového modelu nám umožňuje větší abstrakci a rozšiřitelnost systému. Jednotlivé třídy obsahují privátní proměnné, k nimž mají přístup metody těchto tříd, a jsou obaleny metodami *get* a *set* pro přístup ostatních objektů k těmto vlastnostem.

4.2.2.3 Vrstva webových formulářů

Webové formuláře jsou rozhraním mezi systémem a uživatelem, jejich prostřednictvím komunikuje uživatel s danou aplikací. Každý formulář je rozdělen na dvě části. První z nich je HTML kód, kam můžeme vkládat různé prvky jazyka HTML a komponenty platformy ASP.NET, v případě potřeby využívat CSS stylů. Druhou je pak třída, která se stará o obsluhu stránky. Tato třída má definované metody pro obsluhu prvků, které se vyskytují na stránce (tlačítka, formuláře). Každá z těchto tříd má metodu *PageLoad*, která je volána po inicializaci všech prvků na stránce a umožňuje tedy naplnění těchto prvků daty podle požadavku uživatele.

Další technikou, jenž zde využijeme, je vytváření vlastních komponent (*user controls*) s příponou .aspx. Tímto způsobem můžeme upravit existující komponenty, pokud nám jejich chování nevyhovuje tak, jak je předdefinováno. Lze také vytvořit novou komponentu a definovat prvky, které bude obsahovat společně s jejich chováním a funkcí. Na tyto komponenty je potřeba formulář upozornit pomocí direktivy v záhlaví formuláře. Její syntaxe vypadá následovně.

<%@ Register TagPrefix="předpona tagu" TagName="jméno tagu" Src="cesta k souboru .ascx" %>

Použití:

<předpona tagu:jméno tagu Id=" " runat="server"></ předpona tagu: jméno tagu >

4.2.3 W3g Parser

Při vytváření nástroje používaného pro nalezení informací ze záznamu jsem vycházel z nástroje, který je volně stažitelný na internetu (*http://code.google.com/p/w3gparser/*). Tento nástroj je určen pro čtení záznamů z klasické hry *Warcraft 3*. Pro účely vyvíjené aplikace je potřeba tento nástroj přepracovat tak, aby dokázal číst záznamy, které se vytvářejí při hraní mapy *Dota Allstars*. V hlavičce záznamu je vždy uveden název mapy, na které se hra hrála. U mapy *Dota AllStars* je uvedena tato mapa v hlavičce společně s verzí mapy. Toho využijeme pro validaci záznamů (přijímáme pouze

záznamy s mapou *DotaAllstars*) a pro načtení souboru xml, který se váže k vybrané verzi mapy. Záznam je rozdělen na dvě na sobě v podstatě nezávislé části

4.2.3.1 Hlavička

Hlavička obsahuje informace o všech hráčích, kteří se hry účastnili. Obsahuje ještě další informace, ale ty nejsou z pohledu našeho systému důležité. Jedinými údaji, které v hlavičce ještě využíváme, jsou délka hry a počet bloků s akcemi hráčů. Hlavička má vždy stejnou velikost.

4.2.3.2 Akce

Tato část záznamu je rozdělena na bloky s jedinečnými identifikátory v hexadecimálním kódu. Každý z těchto bloků značí jinou akci provedenou hráčem a má proměnou délku. Pomocí těchto bloků dokážeme určit, jak se hráč pohyboval po mapě nebo jaký předmět koupil či jakou schopnost použil. Každý předmět, hrdina i schopnost mají opět své identifikátory *(32 bitový integer*), které jsou v rámci mapy jedinečné. Tyto identifikátory se převádějí na řetězce (4 znaky). Jelikož ale tvůrce mapy postupně přidává předměty, hrdiny a schopnosti nebo mění jejich vlastnosti, mění se tím i jejich identifikátory. Pro vyřešení této problematiky se využívá souborů ve formátu xml, přičemž každý obsahuje informace pro jednu z verzí mapy. V těchto souborech je vždy seznam všech předmětů, hrdinů a schopností, které se vyskytují v dané verzi mapy, jejich cena a identifikátor.

5 Implementace

5.1 Použité nástroje

Microsoft Visual Studio Express Edition

Volně dostupný nástroj pro tvorbu aplikací firmy Microsoft. Toto prostředí je určeno jak pro tvorbu webových aplikací tak i pro tvorbu ostatních aplikací a obsahuje i vlastní databázový server.

Microsoft SQL Query Analyser

Nástroj pro testování SQL dotazů.

Adobe Photoshop 7.0 CE

Nástroj, který jsem využili při grafickém návrhu aplikace, a při tvorbě kaskádových stylů.

5.2 Design

Celý design je implementován pomocí technologie kaskádových stylů. Tyto styly jsou uloženy v jednotlivých souborech, jejichž názvy jsou shodné s názvem stránky, pro kterou jsou určeny. Celý kód je členěn do DIVů (značka pro blok) a komponent platformy ASP.NET, které jsou pomocí kaskádových stylů pozicovány. Při vytváření designu jsem se setkal s několika odlišnostmi mezi jednotlivými internetovými prohlížeči. Hlavní problémy se projevily při používání vlastností *padding* (vnitřní okraj) a *margin* (vnější okraj) .

Při použití vlastnosti *padding* a pevné šířky DIVu dochází u prohlížeče Firefox k přičtení *paddingu* k celkové šířce, zatímco IE ponechá zadanou šířku. Tato odlišnost se dá řešit více způsoby, jedním z nich je vložení DIVu, u kterého potřebujeme pevnou šířku, do dalšího DIVu, kterému nastavíme vlastnost *padding* a neurčíme šířku.

Problémy související s vlastností *margin* se objevují v IE (*double margin bug*). Tyto problémy vznikají při použití plovoucích (*float*) prvků tak, že vlevo plovoucí prvek má dvojnásobný levý vnější okraj, analogicky vpravo plovoucí prvek má dvojnásobný pravý vnější okraj. Řešením toho problému je přidání atributu *display* s hodnotou *inline* tomuto prvku nebo tento prvek vložit (bez marginu) do DIVu, kterému dáme *padding* s požadovanou hodnotou.

5.3 Bezpečnost

Bezpečnost systému a ochrana dat uživatelů před jejich zneužitím dnes patří mezi klíčové faktory. Přístup do částí, kde jsou zneužitelné informace, je povolen pouze uživatelům, kteří na to mají právo.

Jedním z nejpoužívanějších způsobů takového ověření je kombinace uživatelského jména a hesla. Při vyplňovaní hesla do formuláře je samozřejmostí jej zobrazovat ve skryté formě (nejčastěji jsou znaky nahrazeny hvězdičkami). Uživatelské jméno a heslo jsou uloženy v databázi v tabulce uživatelů.

5.3.1 Uložení hesel v databázi

Při uložení hesel v databázi je potřeba tyto data nějakým způsobem zakódovat, aby v případě úniku informací nebylo možné získat platné kombinace uživatelských jmen a hesel. Využijeme tzv. zahashované podoby hesla, která heslo podle určité hashovací funkce zakóduje na řetězec čísel a písmen. Až tento nový řetězec se teprve vkládá do databáze.

Mezi nejznámější hashovací funkce patří SHA1 a MD5. Při přihlašovaní uživatele do systému se odeslané heslo zahashuje stejnou funkcí, která hashuje heslo při registraci, a teprve tento hash se porovnává s databází. Využití této techniky má však i svou nevýhodu. V případě, že uživatel zapomene své heslo, nemáme možnost toto heslo zpětně zjistit. Jedinou možností v tomto případě je vygenerování nového hesla a zaslání na emailovou adresu zadanou při registraci. Po vygenerování nového hesla si ho uživatel může změnit ve svém profilu.

5.3.2 SQL Injection

Dalším bezpečnostním rizikem, dnes často využívaným na internetu, je zneužití pomocí tzv. sql injection (injekce). Při tomto typu útoku se útočník snaží vložit potencionálně nebezpečný SQL dotaz do formulářových vstupů nebo přímo do url stránky. Daná akce související s tímto zneužitím pak většinou provede více operací nad databází. Útočník tak může například vymazat všechna data v databázi. Obranou před tímto druhem útoku je používání parametrizovaných příkazů pro práci s databází, kde každý parametr má datový typ odpovídající jednomu z SQL datových typů .

5.4 Aplikace

5.4.1 Autentizace

ASP.NET nabízí mnoho způsobů jak provést autentizaci uživatele. Při návrhu systému jsem se rozhodl pro autentizaci pomocí sessions. Po úspěšném přihlášení uživatele se vytvoří instance třídy *cUser* s informacemi o přihlášeném uživateli a je vytvořena session s hodnotou tohoto objektu. Session jsou proměnné, které se uchovávají na straně serveru a každý klient je spojen se svými sessions pomocí jednoznačného řetězce (*SessionID*) pro každou relaci. Tento identifikátor je pak mezi serverem a klientem předáván pomocí *cookie*. Pří zobrazovaní stránek, kde má registrovaný uživatel na rozdíl od neregistrovaného práva navíc, testuji v metodě *PageLoad* session na přítomnost objektu

typu uživatel, a pokud je test úspěšný, pak uživatele považuji za přihlášeného. Pokud test není úspěšný, práva jsou omezena nebo je uživatel přesměrována na úvodní stránku.

5.4.2 Přechod mezi stránkami

Pro přechod mezi stránkami je převážně využíváno klasických HTML odkazů, ale při jejich využití se ztrácí veškeré výhody ASP.NET. Proto jsem v některých případech využil komponenty *<asp:LinkButton>* a *<asp:button>*, které se při stisku odesílají na server, a tam podle jejich atributu *onclick="nazev_funkce"* zavolají požadované funkce, díky kterým lze provézt potřebný kód, a teprve poté uživatele přesměrovat pomocí metody Response.Redirect ("nazev_stranky.aspx").

5.4.3 Naplnění obsahu stránky z databáze

Pro naplnění obsahu stránek jsem využíval zejména komponentu *<asp:Table>*, která je podobná klasické tabulce z jazyka HTML. Technologie ASP.NET totiž neumožňuje libovolné přidání nové komponenty za běhu aplikace – všechny komponenty, které využíváme musí být definované před zavoláním metody *PageLoad()* v této metodě pak můžeme určit viditelnost, obsah nebo třídu kaskádových stylů jednotlivých komponent. Proto při potřebě vykreslení několika obrázků nebo textů používám tabulku, kde lze programově přidat další řádky za běhu aplikace. Samozřejmě pro tyto obrázky a texty můžeme použít jen prvky jazyka HTML. Tabulka sice není přiliž vhodná pokud přihlížíme k přístupnosti internetových stránek, ale na druhou stranu se tak dají prvky umístit, aby nedošlo k různému zobrazení stránek v rozličných prohlížečích.

5.4.4 W3g parser

Implementace toho nástroje byla časově velice náročná, zejména protože neexistuje žádná dokumentace k formátu záznamů. Při tvorbě jsem vycházel z projektu, který byl určen pro jinou verzi a bylo tedy potřeba jej upravit, aby zpracovával mapu *Dota AllStars*. V této mapě se na rozdíl od ostatních map objevují nové bloky, u nichž je potřeba určit význam. Toto určování jsem prováděl na základě ručního sledování záznamů a zapisování informací a poté jsem je porovnával s výstupem mého parseru.

Další komplikací bylo určení jednotlivých identifikátorů pro předměty, schopnosti a hrdiny. Poté, co jsem objevil blok, kde si hráč kupuje předmět, jsem již opět podle sledování záznamů zjistil jednotlivé identifikátory. Analogicky to probíhalo u schopností a hrdinů.

5.4.5 Ošetření chybových stavů

Používání každé aplikace se nevyhne chybovým stavům, ale pro programátora je potřeba tyto chybové stavy vhodně ošetřit, aby neměly vliv na uživatele nebo chod celé aplikace. Mezi nejčastější příčiny patří výpadek databáze, špatně implementovaný algoritmus nebo chybné zadání údajů. Pro

tyto potřeby jsou všechny potenciálně kritické sekce v aplikaci uzavřeny blokem *try, catch, finally*. Příkladem může být využití u databázových tříd, které v bloku try provádí veškerou komunikaci s databází, následuje blok nebo více bloků catch, jenž chytají nastalé výjimky při práci s databází. Následuje blok finally prováděný za všech okolností a v tomto bloku ukončíme veškerou komunikaci s databází.

6 Možnosti rozšíření systému

6.1 Vyhledávání a porovnávání mezi uživateli

Prezentace obsahuje pouze možnost zobrazení odehraných her a jejich prohlížení. Možné rozšíření by spočívalo v možnosti vyhledávání uživatelů podle počtu odehraných her, podle průměrného hodnocení či podle nejhranějšího hrdiny. Dále by se dala vytvořit část prezentace, která by obsahovala seznam všech registrovaných uživatelů. V této části by pak uživatelé mohli nastavit různé filtry, díky nimž by bylo možné najít uživatele s podobnými statistikami, tedy s podobným chováním. Samozřejmostí by pak bylo řazení podle bodového hodnocení nebo podle herní aktivity. Příkladem takového filtru by mohlo být „Najdi hráče, který často hraje hrdinu *Zeus* a jako třetí předmět staví *Bracer*".

6.2 Rozšíření editoru hrdinů

Editor hrdinů nyní umožňuje pomocí javasriptu pouze zvyšování úrovní u vybraného hrdiny a pozorování růstu jeho atributů. Tento editor by se dal rozšířit o možnost zvolení schopnosti hrdiny při získání nové úrovně (některé totiž zvyšují i atributy hrdiny). Další možností je propojit editor hrdinů s prezentací předmětů.

Pomocí javascriptu by zde uživatel mohl hrdinovi přidat předmět a opět pozorovat, jaký vliv to bude mít na jeho atributy. To se dá využít při hledání nejlepší skladby předmětů pro daného hrdinu.

6.3 Nahrávání záznamů

Pro zautomatizování nahrávání záznamů do systému by se dala vytvořit jednoduchá aplikace, která by běžela na pozadí. Tato aplikace by se starala o testování adresáře *Replays*, který obsahuje instalace hry *Warcraft 3*. Tato hra totiž po odehrání jakékoliv hry uloží do výše zmíněného adresáře záznam s názvem LastReplay.w3g. Tohoto chování by pak využívala navrhovaná aplikace. Vždy při přepsání (změny času a data) souboru LastReplay.w3g by tento soubor odeslala do systému. Samozřejmě by tento program musel obsahovat přihlašovací údaje do systému.

6.4 Uložení více údajů ze záznamu

Nyní se při nahrání záznamu do systému soustředí parser pouze na údaje, které jsme na začátku práce označili za nejdůležitější. Ovšem v záznamu jsou obsaženy další informace, které by pro nás mohli mít potencionální význam. Takovou informací může být například údaj o cestě, na které hráč hrál,

nebo můžeme evidovat, kolik času hráč strávil v lese. Znalost cesty je možné využít k určení nejoblíbenější pozice na mapě a údaj o lese by se dal využít k lepšímu určení herního chování. Na základě těchto údajů již by se dalo uvažovat o využití nějaké z metod pro získávání znalostí z dat. Záznam obsahuje nepřeberné množství dalších údajů, jako například počet akcí hráče, což lze využít pro výpočet APM (*actions per minute*). Podobných informací se v záznamu nachází skutečně mnoho, ale není cílem této práce je zde všechny zmiňovat.

7 Závěr

Úkolem bakalářské práce bylo vytvořit systém, který analyzuje herní chování na základě záznamů nahraných uživateli a shromažďuje informace o herní aktivitě hráče. Přestože jsem základní analýze aplikace předcházející samotnému návrhu prezentace a systému věnoval dostatek času, musel jsem se k ní poměrně často vracet a upravovat některé ne zcela vhodně navržené části.Systém byl navržen do více vrstev, ale i tak bylo potřeba během implementace udělat drobné odchylky od toho návrhu. To jen dokazuje, jak je důležité věnovat analýze zpracovávaného projektu dostatek času.

Aplikace bude zavedena do praxe během následujících měsíců (nyní se testuje) a počítá se s jejím dalším rozšířením, které by však vzhledem k návrhu nemělo být problematické. Ve fóru je zřízeno speciální vlákno, kde mohou hráči vznášet podměty na rozšíření této aplikace. Testování komunitou hráčů mapy *Dota Allstars* poukazuje na přínos tohoto systému pro hráče, a podle jejich ohlasu je zřejmé, že se mi podařilo vytvořit systém, který najde využití u mnoha hráčů. Neméně důležitým přínosem je nástroj pro zpracování záznamů, který může sloužit pro rozhodčí zápasů ke kontrole pravidel hry (předávání předmětů mezi hráči). Nicméně předtím, než bude systém umístěn na internetu k využití širokou veřejností, bude potřeba ještě doladit rozdíly mezi záznamy, hrané pod různými módy (mění se délky bloků).

Mým osobním přínosem z této práce je obohacení o využití objektově orientovaných technik při vytváření internetových aplikací a seznámení s technologií .NET.

Literatura

[1] Archer, T.: Myslíme v jazyku C#, Grada Publishing 2002

[2] Microsoft Corporation: Vytváříme zabezpečené aplikace v ASP.NET, Computer Press 2004

[3] Lacko, L.: ASP.NET A ADO.NET 2.0, Computer Press 2006

[4] Škultéty, R.: JavaScript - programujeme internetové aplikace, Computer Press 2001

[5] Vše o tvorbě internetových stránek. Dokument dostupný na URL *http://www.jakpsatweb.cz/* (duben 2008).

[6] On-line podpora pro technologie firmy Microsoft. Dokument dostupný naURL *http://www.msdn.cz/* (duben 2008).

Seznam příloh

Příloha 1. Herní módy Dota AllStars

Příloha 2. ER- diagram

Příloha 3. Štábní kultura

Příloha 4. Adresářová struktura

Příloha 5. CD se zdrojovými kódy

Příloha 1: Herní módy Dota AllStars

Na začátku hry *Blue player* (modrý hráč – většinou zakladatel hry) zvolí systém, jakým se přidělí všem hráčům hrdinové. Módy se vkládají do jedné řádky a píší se jako zpráva ostatním hráčům. Tato zpráva má následující formát *-apidsh* nebo *-dm -ar –sc* (módy lze kombinovat). Není potřeba se obávat o pořadí psaných módů, hra si poradí sama nebo Vás upozorní (dá další šanci pro zadání), pokud je mód nesprávný/nekompatibilní.

Módy psané na začátku hry:

- AR (All random)

Náhodně vybere za hráče hrdinu. Patří k nejhranějším módům.

- TR (Team random)

Přiřadí hráči náhodně hrdinu podle strany, za kterou hraje.

- TP (Team pick)

Hráči mohou vybírat hrdiny pouze z taveren patřících straně, za kterou hrají.

- AP (All pick)

Hráči si vybírají ze všech hrdinů. Patří k nejhranějším módům.

- LM (League mode)

Mód, který se hraje na oficiálních akcích, jakými jsou ligy nebo turnaje.

- MM (Mirror match)

Oba týmy dostanou stejné hrdiny.

- DM (Dead match)

Jediný herní mód, který mění hrdiny během hry. Při smrti hráče se jeho hrdina pohřbí a hráči je přidělen další. Hra může kromě klasického zničení divu skončit i vybráním všech hrdinů.

- ID (Item drop)

Pokud hráč umře, vypadne mu libovolný předmět z inventáře na zem. Této mód se téměř nehraje.

- EM (Easy mode)

Upravený mód, který zvyšuje získané zkušenosti a peníze. Upravuje věžky, aby byly slabší. Slouží celkově ke zkrácení hry.

- SC (Super Creeps)

Tento mód přidává na normální hry jednotky, které jsou extrémně silné a střídavě pomáhají jednotlivým stranám.

- NP (No powerups)

Ve hře se nevyskytují powerupy.

- MR (Mode random)

Vybere náhodný mód.

- SH (Same hero)

Všichni hráči dostanou stejného hrdinu.

- AI (All inteligence)

Vybírání probíhá pouze z hrdinů, jejichž zaměření je na inteligenci.

- AA (All agility)

Stejné jako AI, ale výběr je zaměřena na obratnost.

- AS (All strenght)

Stejné jako AA a AI, ale výběr je soustředěn na sílu.

- SP (Shuffle players)

Náhodně prohodí hráče v týmech.

- VR (Vote random)

Zobrazí výběr ze tří palet hrdinů označených Option 1, Option 2 a Option 3. Příkazem *-option X*, kde X je číslo výběru, zvolíte konkrétní paletu hrdinů. Každý hráč má právo vybrat jednu možnost. Vítězí paleta s největším počtem hlasů. Hrdinové budou náhodně přiděleny **uvnitř** jednotlivých teamů.

- RD (Random draft)

Náhodně zvolí do dvou taveren po 20-ti hrdinech, ze kterých si postupným výběrem (1 2 2 2 2 1) hráči volí hrdiny. Funguje jako kombinace AR a AP.

- SD (Single draft)

Zobrazí výběr ze tří hrdinů. Příkazem *–pick X*, kde X je číslo vybraného hrdiny, si hráč zvolí hrdinu.

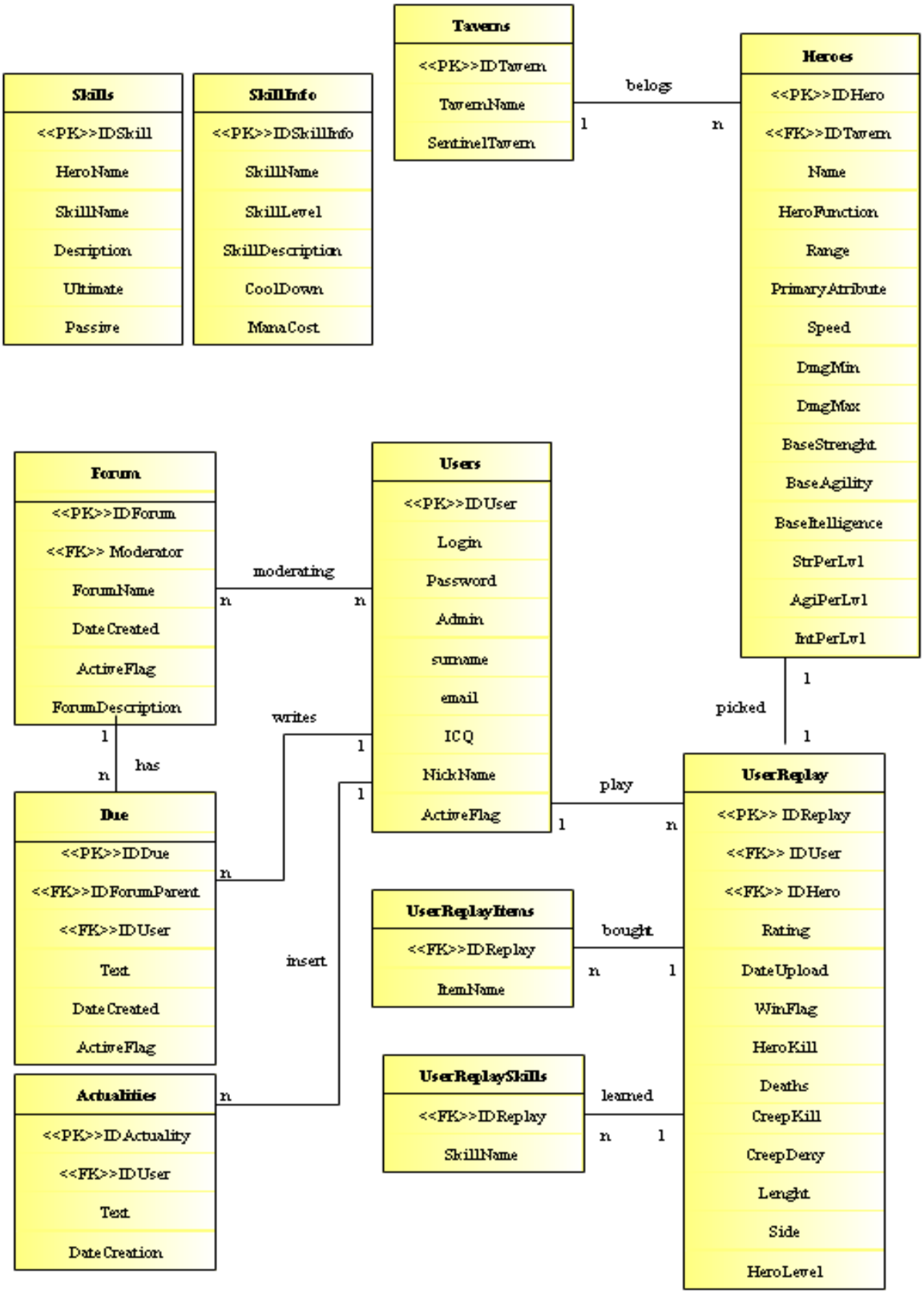
- DU (Duplicate mode)

Každý hrdina může být vybrán vícekrát, tzn. nezmizí z nabídky, pokud si ho někdo zvolí.

- WTF

Zábavný mód, který upravuje cooldown a cenu many u kouzel na 0.

Příloha 2: ER-diagram



Příloha 3: Štábní kultura

Pro vyšší přehlednost zdrojového kódu, by se měla dodržovat jistá pravidla pro jeho formátování a pojmenování jednotlivých proměnných, metod, tříd, parametrů atd.

V projektu jsem se držel těchto pravidel.

- pojmenování tříd začíná vždy písmenem c, následuje pojmenování třídy, začínající velkým písmenem

př. cForum

- objekty začínají písmenem o, následuje pojmenování objektu, začínající velkým písmenem

př. oAutor

- parametry metod začínají písmenem A, následuje jméno parametru začínající velkým písmenem

př. AAutor

- názvy metod začínají písmenem, následuje pojmenování metody začínající velkým písmenem, pokud se skládají z více slov, začíná každé nové slovo velkým písmenem

př. NactiDataZFormulare

- lokální proměnné začínají písmenem značící jejich datový typ (i int, s string, atd.), následuje jméno proměnné začínající velkým písmenem

př. iPocetCyklu

- privátní proměnné třídy začínají podtržítkem, následuje název začínající velkým písmenem

př. _Id

- veřejné vlastnosti tříd začínají velkým písmenem

př. Id

Příloha 4: Adresářová struktura

- dotawebsite – kořenový adresář projektu
  - App_code – adresář pro umisťování našich zdrojových kódů
    - classes – adresář pro jednotlivé třídy
      - Bc – aplikační třídy
      - Db – databázové třídy
      - W3gParser – třídy pro pársování záznamů
  - Controls – uživatelské ovládací prvky
  - Bin – soubor s knihovnami potřebnými pro běh w3g parteru
  - Tmp – adresář, do kterého se dočasně ukládají právě zpracovávané záznamy
  - Log – adresář se souborem o logování a povedených změnách v systému
  - Xml – obsahuje xml soubory pro různé verze mapy Dota Allstars
  - JS – adresář se soubory javascriptu
  - Css – kaskádové styly
    - IS – kaskádové styly pro informační systém
  - img – obrázky využívané v designu stránek
    - heroes – obrázky všech hrdinů ve hře
      - detail – obrázky, které se načítají v editoru hrdinů
    - items – obrázky předmětů
    - shopdealers – obrázky prodavačů
    - skills – obrázky schopností hrdinů
    - userphotos – uživatelské obrázky (pro vlastní stránku)
  - IS – adresář se stránkami informačního sytému