VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÝ HERNÍ PORTÁL

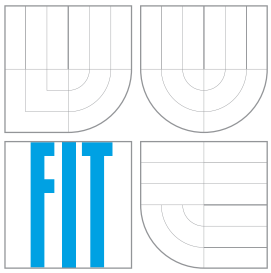
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN KUBÍČEK

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÝ HERNÍ PORTÁL

WEB GAME PORTAL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN KUBÍČEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RADEK BURGET, Ph.D.

BRNO 2015

Abstrakt

Tato bakalářská práce pojednává o návrhu a přestavbě již existujícího systému webového portálu ImmortalFighters. Detailně je analyzován stávající systém za účelem odstranění chyb už v samotném počátku návrhu přestavby. V rámci návrhu bylo vypracováno jak nové schéma databáze, tak i nová struktura webu. Byla prozkoumána podstata a herní logika fantasy hry Dračí doupě a možnost využití moderní objektově - orientované technologie k její implementaci do internetové podoby. Byla provedena analýza možných technologií, jež disponují objektově - orientovaným návrhem, a bylo by je tedy možno využít při samotné implementaci. Podle jasně stanovených kritérií z nich pak byla vybrána technologie nejvhodnější. Práce dále představuje chystané změny, které vychází z předchozí analýzy a osobních zkušeností nabytých v rámci dlouhodobého působení na webovém portálu. V praktické části je návrh realizován pomocí vybraných technologií – PHP rámce Symfony 2 a Doctrine 2 rámce objektově - relačního modelu tak, aby byla zřetelná značná vylepšení a základní rozdíly oproti stávajícímu řešení.

Abstract

The subject of this thesis is the design and reconstruction of the already existing ImmortalFighters web portal system. The thesis analyses in detail the current system in order to prevent errors right from the start of the proposed changes in design. As part of the proposed change, a new database and a new site structure were created. The nature and logic of the Dungeons & Dragons fantasy game and the possibility of using modern object-oriented technology for use in the web-based versions were explored. An analysis was made of possible technologies that are or have the capability of object - oriented design, and could therefore be used for the actual implementation. The most suitable technology was then chosen for this according to very clearly defined criteria. The work also presents the proposed changes that arise from analysis and personal experience accumulated from a long - term presence on the web portal. The practical part of the proposal is done using special technology – the Symfony 2 and Doctrine 2 PHP frameworks of an object - relational model, so that significant improvement and fundamental differences as compared the current solutions were clearly seen.

Klíčová slova

Dračí doupě, hra na hrdiny, webový herní portál, Symfony 2, PHP, ORM, Doctrine 2

Keywords

Dungeons and Dragons, RPG, web game portal, Symfony 2, PHP, ORM, Doctrine 2

Citace

Martin Kubíček: Webový herní portál, bakalářská práce, Brno, FIT VUT v Brně, 2015

Webový herní portál

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Radka Burgeta, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Kubíček
19. května 2015

Poděkování

Zde bych rád poděkoval svému vedoucímu bakalářské práce Ing. Radku Burgetovi, Ph.D. za jeho čas, odborný dohled, konzultace a vedení.

Obsah

Kapitola 1

Úvod

V současné době v oblasti herního průmyslu, a především v odvětví s RPG hrami, stojí v popředí hry s co nejrealističtější grafikou, zpracováním a zajímavými příběhy, které tvoří celé týmy scenáristů. Hry jsou však omezeny fantazií vývojářského studia a hráči si v nich nemohou vychutnat naprostou volnost a svobodu. Proto hráči stále vyhledávají weby, kde je možné najít a hrát RPG bez větších omezení a velkých hardwarových nároků.

I když webový portál ImmortalFighters v tomto trendu pokračuje, ztrácí své hráče a není momentálně schopen konkurovat ostatním webům zabývajících se fantasy hrou Dračí doupě. Z části je to způsobeno tím, že se web soustředí pouze na českou a slovenskou komunitu, ale jsou zde i závažnější aspekty mající za následek odliv hráčů. Jedním z důvodů je zastaralý grafický design webového portálu, který způsobuje problémy při použití na přenosných zařízeních menších rozměrů. Další vadou je pak zastaralý systém, který hráče velmi omezuje, a neumožňuje jim takovou volnost, jakou očekávají, či dokonce kompletní nefunkčnost některých herních mechanismů nutných pro plný požitek ze hry.

Tak vznikla iniciativa navrhnout a vyvinout nový systém s použitím moderních technologií a poznatků získaných dlouhodobým hraním Dračího doupěte. Nový systém by měl hráči poskytnout úplnou volnost a veškeré možnosti, jako je tomu u papírové verze hry. S novým systémem vznikne i nová líbivá grafická podoba webových stránek. Tato kombinace by pak mohla udržet stávající hráče a přilákat nové. Obsah webu je ovšem tak obrovský, že daleko překračuje rozsah této bakalářské práce. V práci bude tedy prezentováno pouze základní jádro nutné k velmi omezenému hraní hry.

V následujících sekci je detailně představeno stávající řešení a hlavní problémy, se kterými se momentálně web potýká. Je zde nastíněno i řešení, případně návrh, jak daným problémům do budoucna předejít. Následuje pak kapitola s analýzou metod, podle kterých by se dal nový systém naimplementovat. Technologie jsou porovnávány mezi sebou a podle specifických pravidel se hledá technologie nejvhodnější. Dále jsou v této kapitole detailně popsány zvolené technologie a jejich největší přínos pro novou verzi systému. Následuje popis samotného návrhu webového portálu, který se skládá z dílčích částí, jež tvoří návrh databáze a reprezentace dat, struktura webového portálu a řešení jednotlivých dílčích modulů, a v neposlední řadě také návrh nového rozložení uživatelských rolí. V implementační části je pak popsáno, jak byl výše zmíněný návrh implementován do technologie, a jaké problémy bylo při jejich implementace potřeba řešit.

Kapitola 2

Stávající řešení

Cílem této práce je přepracování již existujícího webového portálu ImmortalFighters[1], respektive jeho vnitřního systému, jenž je značně zastaralý. Web vznikl v roce 2002 odloučením od jiného webového portálu jako snaha převést papírovou verzi hry Dračí doupě do internetové podoby. Funguje na bázi jednoduchého principu psaní příspěvků do specializovaného fóra. Samotný systém webového portálu však nebyl dost dobře navržen, což se neblaze projevilo i na práci při jeho pozdějších úpravách. Původní systém neměl žádnou hierarchii, a pokud v pozdější fázi vývoje webu vznikla potřeba přidat nový modul, bylo nutné se starému systému přizpůsobit.

To, že ve starém systému nefungovala žádná hierarchie, bylo zapříčiněno tím, že od samotného vzniku webu se zdrojový kód systému webového portálu významně nezměnil, pouze se při přechodu na novější verzi PHP upravovaly funkce tak, aby vyhovovaly novým standardům. Docházelo tak ke vzniku mnoha bezpečnostních a herních problémů. Při provozování webu nebylo totiž využíváno nových komplexnějších technologií, dnes dostupných. Standard původního systému byl udržován a upravován tak, aby byl schopen „běžet“ na nových technologiích. Velmi podobně je na tom webový portál i po designové stránce. Stále se využívá velice zastaralé a dnes již nepoužívané technologie tabulkového layoutu[2]. Takové řešení postrádá jakékoliv prvky responzivního webového designu[3], na což je v současnosti kladen stále větší důraz.

2.1 Dračí doupě

Dračí doupě je českou mutací známější světové vyprávěcí hry Dungeons & Dragons založené na principu tzv. Role-playingu[4]. Náplň hry je velmi jednoduchá. Hráči si na začátku zvolí pohlaví a jednu z předpřipravených ras a povolání. Následně si dotvoří charakter postavy dle své libosti a doplní hodnoty potřebné k hraní, které ukládají pravidla v závislosti na vybrané rase a povolání. Následuje samotná hra, kdy jeden z hráčů, zvaný též Pj[5], vypráví ostatním příběh, tedy to, co jejich postavy prožívají či v jaké situaci se zrovna nacházejí. Hráči pak v reakci prezentují, jak se jejich postavy chovají a co dělají. Za to jsou ohodnoceni zkušenostními body, které jim při určitém počtu dovolí přestoupit na vyšší úroveň,

[1] Dostupné na http://www.immortalfighters.net

[2] layout - rozložení webové stránky

[3] responzivní web design - stylování html dokumentu, které zajistí, že se daná stránka korektně zobrazí na všech nejrůznějších zařízeních

[4] Role-play - hraní a vžívání se do fiktivní postavy

[5] Pj - Pán jeskyně, jeden z hráčů, který nehraje za postavu a řídí celý příběh

Obrázek 2.1: Grafická podoba uživatelského profilu na stávajícím webu

a zprostředkuje možnost se vydávat na složitější výpravy. Hráči ve svých reakcích mohou využívat spoustu herních aspektů, jako jsou například *itemy*.

Itemy tvoří značnou část samotné hratelnosti, jsou to veškeré předměty, na něž může hráč v průběhu hry narazit nebo je nějaký způsobem získat (např.: jako odměnu, zakoupením či vyrobením). Předměty mohou mít obyčejný nebo magický původ. Pokud mají magický původ, mohou disponovat různými bonusy či postihy, jež mají efekt na postavu, která předmět používá. S těmito předměty pak může hráč dělat cokoliv dle své chuti, nebo co mu pán jeskyně dovolí. Tyto prvky dračího doupěte tak vyžadují notnou dávku představivosti a fantazie. Ta může být v případě hráčů a vlastně i pána jeskyně neomezená. To dává značnou volnost a prostor vzniknout situacím, které nepokrývají ani originální pravidla.

Vzhledem k tomuto faktu se hra velice špatně převádí do pevné internetové či elektronické podoby. Uveďme zde příklad: Hráč disponuje kouzelnými kalhotami. Pokud si je správně nasadí na nohy, dostane za ně určitý bonus. Může však nastat situace, kdy se hráč rozhodně kalhoty nosit místo opasku, a ztrácí tedy daný bonus, který poskytují. Hráč se může také rozhodnout nasadit si kalhoty na hlavu, a místo bonusu je potrestán postihem. Výše postihů nebo bonusů se může lišit v závislosti na dané rase, povolání či kombinaci těchto faktorů. Proto je potřeba stanovit si určitá pravidla nad rámec oficiálních pravidel, anebo některá hodně abstraktní či obšírná pravidla úplně vypustit. Avšak ani tento krok nezajistí, že dosáhneme stoprocentní shody. Z tohoto důvodu si webový portál definoval pár svých vlastních pravidel, která částečně vycházejí z oficiálních pravidel.

2.2 Hierarchie webu

Web je postaven na modelu s využitím modulů, kdy každá významnější podstránka tvoří vlastní nezávislý modul. Momentálně se na webu nachází přes 70 modulů k nim zhruba 120 podpůrných skriptů. Většina z nich však není využívána ať už z důvodu, že daný modul zanikl, anebo nebyl nikdy dokončen a pořádně naimplementován. Modul má vlastní vnitřní logiku, jež dokáže zobrazit nebo získat data, avšak samostatně fungovat nedokáže. Je totiž závislý na předpřipravených datech, která obstarávají centrální skripty. Tato pětice centrálních skriptů se stará o zpracování veškerých funkcí, dotazů na databázi a propojení všech potřebných modulů. Využívají k tomu i mnoho malých podpůrných include[6] skriptů.

[6]include - vkládání části kódu nebo skriptů z externího souboru

2.2.1 Skripty

Většina z těchto skriptů je velmi obsáhlá a nepřehledná, mnohdy také obsahují funkce, které už se dávno nepoužívají, což zhoršuje jejich čitelnost. Tento způsob centrálních skriptů není příliš výhodný. Stačí udělat chybu jen v jednom z nich a celý webový portál není schopen provozu.

functions.inc.php - Nejobsáhlejší spouštěný skript obsahuje většinu funkcí využívaných napříč celým webovým portálem. Jako příklad funkcí můžeme uvést konverzi dat do vhodné formy získaných pomoci dotazů na databázi či funkce zajišťující bezpečnost a kontrolu pravomocí nebo také převodní funkce (čas, text). Ve výjimečných případech skript používá i přístup k databázi a dotazy s ní spojené.

action.php - Tento skript z velké části obsluhuje komunikaci s databází. Zpracovává získaná data z formulářů nebo z různých modulů pomocí metod *POST* a *GET*. Na základě takto získaných dat je pak tvořena série dotazů na databázi. V tomto skriptu se neprovádí žádné bezpečností kontroly. S výjimkou kontroly, zdali jsou data korektně zadána.

w_action.php - Skript fungující na podobné bázi jako **action.php** ovšem s tím rozdílem, že tento skript zpracovává dotazy, které se týkají pouze postav. Zajišťuje tedy přestup na vyšší úroveň, automatické korekce postavy a vlastně jakoukoliv modifikaci hráčské postavy. Rovněž se zde vyskytuje mnoho problémů v oblasti zabezpečení.

to.php - První čistě praktický skript. Stará se o jakékoli vkládání příspěvků, překódování, jejich formátování do html značek a naopak. Také slouží k zálohování starých nepotřebných vláken fóra a převedení již dohraných příběhů do formátu html pro pozdější prohlížení.

table.inc.php - Skript zastupující v jisté abstrakci objekt. Jeho hlavní funkcí je získání veškerých dat o uživateli a herní postavě. Ta jsou následně uložena do globálních proměnných, aby mohla být využívána na celém webovém portálu. Toto řešení není nejvhodnější, neboť skript dává k dispozici všechny data získaná z databázových tabulek, což obnáší jistá bezpečnostní rizika.

include scripty - Sada malých specifických skriptů, které se starají o jednotlivé speciální vlastnosti daných povolání a ras. Každé povolání disponuje vlastní sadou, která se pak vkládá v závislosti na kombinaci povolání, rasy a úrovně dané herní postavy.

2.2.2 Moduly

Odlišných modulů lze na webovém portále nalézt značné množství, dělí se na dva základní typy:

herní - Slouží pro samotné hraní Dračího doupěte, nejvýznamnějšími moduly toho typu jsou uživatelský profil, který zároveň představuje osobní deník hráče, a Quest[7]

neherní - Slouží spíše k odreagování hráčů nebo k jiným neherním účelům a nemá vliv na samotnou hru. Příkladem tohoto neherního modulu je fórum.

[7] Quest - modul podobný fóru, ve kterém hráči prožívají svůj příběh

Stěžejní moduly:

Fórum - Nejdůležitější neherní modul umožňující hráčům komunikovat mezi sebou, či vyjadřovat svůj návrh na příslušném threadu[8]. Největší problém tohoto modulu představuje fakt, že zde hráč vystupuje jako postava, za kterou hraje, nikoliv za sebe samotného. Dochází zde však ke kombinaci dat jak herní postavy, tak z účtu daného hráče.

Quest - Modul založený na principu speciálního fóra s tím rozdílem, že do fóra přispívá pouze malá vybraná skupina hráčů, jenž v daném příběhu figuruje, a PJ, který hru řídí. Je zde také množství vylepšení, která by měla sloužit k lepšímu dotvoření atmosféry příběhu. Většina z nich však nefunguje, i přestože je tento modul stěžejní pro celé hraní.

Osobní deník - Představuje stavební modul celého webového portálu. Zajišťuje zobrazení osobního deníku postavy hráče. Deník postavy je vázán na účet daného hráče, a když zahyne herní postava, hráč je nucen založit si nový účet. Toto řešení není úplně vhodné a nese s sebou řadu komplikací. V některých případech je totiž nutné kromě dat důležitých pro danou herní postavu navíc získat i další data, která se týkají samotného účtu hráče, a neměla by mít nic společného s herní postavou. Ani samotné řešení herní postavy a získávání dat o ní není nikterak elegantní. Místo použití objektu, do kterého by se daná data o postavě nahrála, se využívá specializovaného skriptu (2.2.1).

2.2.3 Databáze a MySQL

Pro databázi je na webu využita technologie MySQL, ale ani zde nejsou žádné náznaky propracovaného návrhu databáze. Tabulky nemají k dispozici primární klíče (a většinou ani cizí) a disponují spíše záznamem, jenž má stejný název jako identifikátor jiného záznamu v jiné tabulce. V takovém případě není možné použít ani reference. Tento fakt znemožňuje využívat vestavěných *JOIN* funkcí na spojování tabulek a vše je nutno vytvářet ručně záznam po záznamu. To dává ve výsledku obrovské množství přístupu do tabulek a velikou režii dat.

Například výše zmíněný skript **table.inc.php**(2.2.1), který se stará o data pro osobní deník a postavu, potřebuje využít zhruba sedmi samostatných SQL dotazů, než je schopen získat většinu potřebných dat. Navíc většina dat v záznamech není uložena v normálové formě, a tudíž se při každém přístupu musí složitě analyzovat a převádět do vyhovujícího formátu.

2.3 Nedostatky a chyby

Níže popsané chyby a nedostatky vznikly častým střídáním programátorů, nepromazáváním starých, nedokončených a nevyužívaných modulů a nedodržováním jednotného stylu při psaní kódu. Dalším aspektem je příliš vágní či pouze omezené řešení daných problémů a chyb. Následkem toho může dojít ke vzniku dalších chyb, které se na sebe postupem času „nabalují“. Jejich původ je pak hluboce zakořeněn v kódu a velmi špatně se dohledává prvotní chyba, kterou by stačilo řádně opravit, aby byl celý problém vyřešen. Aby vše

[8] thread - vlákno či specifické téma daného fóra

fungovalo tak jak má, jsou do kódu přidávány kličky a výjimky, a to i za cenu vzniku jiných chyb či bezpečnostních děr. Ty však fungují jen do doby, než přijde významnější úprava modulu nebo důležitých funkcí, které modul využívá. Taková oprava pomocí výjimky pak přestane fungovat, a to proto, že se do té doby velmi křehká konstrukce oprav začne hroutit.

2.3.1 Chyby v zabezpečení webu

Po bezpečnostní stránce webový portál dosti zaostává. Trpí totiž na všeobecně známé útoky, jakými jsou XSS, SQL Injection, krádež session nebo CSRF. Většina takto napadnutelných míst byla sice opravena, ale ani tyto opravy nebyly příliš účinné a daly vzniknout dalším bezpečnostním hrozbám. Ty sice vyžadující větší úsilí ze strany potencionálních útočníků, avšak stále ohrožující bezpečnost webového portálu. Nicméně na webu v současnosti existuje více než dvacet procent bezpečnostně nedostatečně zajištěných modulů, přes které mohou být útoky vedeny. Až do první poloviny roku 2014 se hesla uživatelů uchovávala jen v prostém textu bez jakéhokoliv hashování. Jakýkoliv schopnější uživatel se mohl k datům dostat pomocí útoku SQL Injection a následně s nimi manipulovat. Kromě těchto útoků má webový portál i bezpečností hrozby pramenící z nedostatečného návrhu a nedostatečného zpracování bezpečnostní ochrany.

Jedním z nejzávaznějších bezpečnostních nedostatků je absence jakékoli kontroly oprávnění a pravomocí ve skriptech **action.php** a **w_action.php** (2.2.1). Většinou se využívá metody *GET*, v níž se data ke zpracování posílají v URL adrese. Ani zde není snaha data nějakým způsobem šifrovat, takže lze velmi jednoduše rozpoznat, co daná metoda udělá a jaká data posílá. Potencionální útočník pak může takový požadavek opakovat s tím rozdílem, že původní jméno zamění za své či jiné, a tak obohatí danou postavu. Podobně je tomu i u metody *POST*, ale taková operace již vyžaduje jisté odborné znalosti a navíc i specializovaný software. Následky jsou u obou metod stejné. Stačilo by přitom aplikovat jednoduchou kontrolu, zdali má uživatel, který odeslal požadavek, na tuto akci příslušná práva. Kontrolou uživatele odesílajícího požadavek by došlo k řešení i dalšího problému, který umožňuje modifikovat svou vlastní herní postavu. Chybí zde totiž i kontrola, zdali uživatel, jenž odeslal požadavek, není tentýž uživatel, na něhož se požadavek aplikuje. V minulosti tato chyba způsobovala to, že se někteří hráči obohacovali na předmětech a herní měně. V těchto a některých dalších bodech došlo k podchycení a opravení dané chyby, avšak stále zde zůstávají místa, kde je možné tuto chybu zneužít.

Dalším bezpečnostním pochybením je práce se session[9]. Ta by měla automaticky zanikat po odhlášení uživatele, ale není tomu tak. O odhlášení uživatelů se stará *cron*[10], který provádí hromadné odhlašování uživatelů po uplynutí určitého časového úseku. Session obsahuje veškerá data o uživateli, kromě jeho vlastní identifikace jako uživatele, tím pádem cron neví, které session mají být zničeny. Z nějakého důvodu rovněž není nastavena exspirace samotné session. Uživatel tedy může přistupovat do modulů nebo provádět úkony, jako kdyby byl přihlášený, avšak pozbývá jakéhokoliv identifikátor určující jeho identitu. Tato zásadní chyba spolu s absencí kontroly oprávnění dává potencionálnímu útočníkovi do rukou prostředky, kterých se dá velmi efektivně využít bez zanechání jakýchkoliv stop. Existuje také možnost tuto chybu využít k session hijackingu[11].

[9] session - možnost držení informací o přistupujících uživatelích
[10] cron - automaticky spouštěný skript v daných časových intervalech
[11] session hijacking - metoda využívající session oběti

2.3.2 Nedostatky v systému a jejich dopad na hru

Kromě bezpečnostních chyb trpí web i různými herními nedostatky, které znepříjemňují hru.

Častou chybou jsou náročné SQL dotazy a následné parsování obrovského množství dat do použitelného formátu. To způsobuje nejen dlouhé načítání webové stránky, ale i poměrně velkou režii na přenos takovýchto dat. S náročnými SQL dotazy je spojeno i nebezpečí, že takto složité konstrukce trpí náchylností k sebemenším chybám jak v samotných dotazech, tak i v datech uložených v databázi. Tento fakt může způsobovat i občasnou nedostupnost některých funkcí nebo modulů právě kvůli špatně vykonanému SQL dotazu.

Kvůli nevalně provedenému návrhu schématu databáze a neužitím normálové formy vzniká mnoho dalších problémů. Ty se řeší ručním zásahem do databáze, a tedy částečnou změnou nebo obcházením celého schématu. Zde dost záviselo na tom, který programátor měl daný problém na starost. Příklad špatného návrhu můžeme sledovat například u přístupových práv na některá fóra, která jsou navržena jako číselné konstanty. Tyto číselné konstanty nepokrývají všechny možné kombinace, a tak se může stát, že daný uživatel, jenž by měl mít přístup do tohoto fóra, jej nemá, protože se při návrhu nepočítalo s tím, že taková kombinace přístupových práv může vzniknout. Tento konkrétní případ se pak neřešil na úrovni databáze, ale na implementační úrovni, kdy byla jeho přístupová práva v kódu převedena na korektní kombinaci, a zaslána funkci, která má na starost právě kontrolu těchto práv. Je to ovšem pouze dočasné řešení, protože v budoucnu bude těchto uživatelů se speciální kombinací přístupových práv přibývat. Velmi podobný problém se řešil u přístupu do některých speciálních modulů, zde však stačila jemná úprava verifikační a přístupové logiky. Tento postup však nelze aplikovat u fór, jelikož jejich vnitřní struktura není tak triviální, jako u těchto speciálních modulů.

Asi nejvýznamnějším nedostatkem z pohledu uživatele je nefunkčnost některých částí určitých povolání v samotné hře. Následkem toho nejsou jistá povolání téměř hratelná, jelikož moduly pro jejich správnou hratelnost jsou buď v dezolátním stavu, anebo nejsou vůbec implementovány. Ať už pro svou složitost, či nerealizovatelnost v tomto stavu systému. Jako příklad může sloužit alchymistická laboratoř nebo správná funkčnost hráčských mazlíčků, kteří u některých postav tvoří podstatnou část hratelnosti.

Z důvodu pomalejšího a obtížnějšího způsobu získávání úrovní a zkušeností oproti běžnému hraní se s vyššími úrovněmi v internetové verzi zpočátku nepočítalo. V posledních dvou letech se však na webu rozrostl počet postav převyšující dříve implementované úrovně a systém musel být značně upravován a s ním částečně i některé tabulky v databázi, aby bylo možné s tak vysokými úrovněmi hrát. Pořád zde ale přetrvávají určité nedostatky, které se odstraňují až ve chvíli, kdy se na ně přijde tak, že herní postava přestoupí na vyšší úroveň.

Posledním viditelným nedostatkem je grafická podoba stávajícího webu, která, jak už bylo zmíněno dříve, stále využívá zastaralého tabulkového rozložení stránky. Ten už se v dnešní době nepoužívá a navíc postrádá jakékoliv prvky moderního web designu, jako je například responzivnost. Přidává zbytečný kód do HTML šablony a formátování takové šablony je pak velmi obtížné a nepřehledné. Navíc v dnešní době se veškeré formátování provádí v externím kaskádovém stylu. Ten pak umožňuje definovat různé způsoby zobrazení pro určitá zařízení pomocí notace *@media* [2], což tabulkové rozložení neumožnuje. Tabulkové rozložení stránky se ani neumí vypořádat s příliš velkým obsahem.

Tento problém vzniká především na fórech a v Questech, kde hráči vkládají příliš dlouhé příspěvky nebo rozměrné obrázky. Ani po grafické stránce webový portál příliš nevyčnívá,

právě naopak, využívá staré formáty obrázků, navíc ve velmi nevhodném rozlišení. Takto načítaných unikátních obrázků na každé stránce je zhruba kolem třiceti. Ani tento fakt nepřidává na rychlosti načítání stránky či nenáročnosti na datový tok požadovaný při zobrazení dané stránky.

Kapitola 3

Technologie

Jelikož se celý systém webového portálu tvoří od začátku, nabízela se otázka, zdali by nebylo vhodné pro novou implementaci systému zvolit jiné modernější kombinace programovacích jazyků. Byly zde jen dva základní požadavky, a to, aby použitá technologie fungovala na principu server-side a disponovala možností použít objektový návrh. Jazyků splňujících tyto požadavky je velmi mnoho a bylo tedy nutné stanovit další požadavky, které by daný výběr zúžili. Zaměřil jsem se tedy na ty nejpoužívanější, protože čím méně je jazyk rozšířený a čím více je exotický, byť by splňoval všechny zadané požadavky, vzniká u něj možnost, že by za pár let mohl přijít o podporu nebo rovnou zaniknout. Podobný problém by mohl vzniknout i u příliš starých jazyků, zde ovšem riziko, hlavně u těch rozšířenějších, není tak vysoké. Nabízela se možnost použití i některých mladých programovacích jazyků, které vyvinuli weboví giganti, jako například Facebook nebo Google pro své specifické potřeby. Avšak tyto jazyky potřebují buď specializovaný hardware, software, anebo jsou příliš specifické pro naše potřeby.

PHP - Původní programovací jazyk, ve kterém byl systém webového portálu naprogramován. Ve prospěch tohoto jazyka hraje i fakt, že je stále nejrozšířenějším programovacím jazykem pro webové aplikace a je podporován na všech webhostinzích. Je to skriptovací interpretovaný slabě typovaný jazyk.

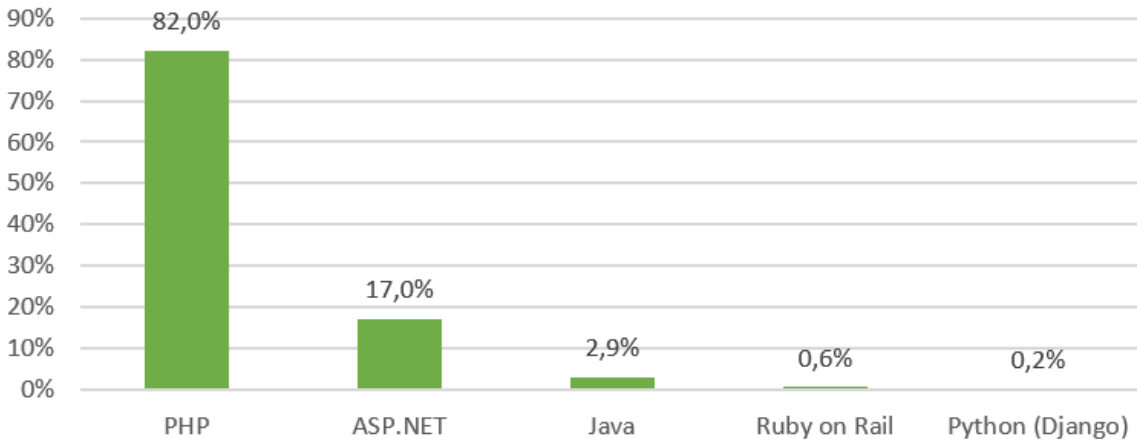
Java - Je multiplatformní, třídně-založený programovací jazyk, jenž je z části interpretovaný. Před samotnou interpretací, kterou zajišťuje *Java Runtime*, se totiž zdrojový kód kompiluje. Tento jazyk je staticky a silně typovaný. To mu neumožnuje takovou pružnost, jakou disponuje PHP. Rámce založené na této technologii jsou velmi oblíbené díky své rychlosti, což je výhodnější zejména u webových stránek s vysokým počtem návštěvnosti.

Python - Flexibilní, komplexní a vysoce škálovatelný interpretovaný skriptovací jazyk. Díky jeho přísným požadavkům na strukturu kódu je výsledný kód aplikace velmi přehledný a čistý. Obsahuje mnoho standartních knihoven, které umožní rychlý vývoj aplikace. Navíc disponuje speciálním rámcem *Django*, jenž slouží primárně pro vývoj webových aplikací.

Ruby - Jednoduchý interpretovaný skriptovací jazyk. Je velmi podobný jazyku Python. Díky tomu, že je tento jazyk koncipován tak, aby ho zvládli i začátečníci, je v něm možné velice rychle vyvíjet nové aplikace. Pro tyto účely byl vytvořen speciální

rámec *Ruby on Rails* sloužící k vývoji webových aplikací. Ačkoliv je tento programovací jazyk velmi silný vývojový nástroj, jeho podpora na mnohých webhostizích stále chybí.

ASP .NET - Neboli *Active Server Pages* je součástí *.NET rámce*, který využívá jazyka *C#*. Obsahuje knihovny generující HTML kódy. Ačkoliv je toto řešení velmi rozšířené, má určitou nevýhodu. Ke svému běhu totiž vyžaduje server s operačním systémem Windows.



Obrázek 3.1: Graf použití server-side programovacích jazyků na webových aplikacích
Zdroj: w3techs.com/technologies/overview/programming_language/all

3.1 PHP

Jako implementační jazyk bylo nakonec ponecháno PHP, a to hlavně z důvodu jeho velké popularity a rozšířenosti. Dá se tedy předpokládat, že jeho vývoj nebude v dohledné době ukončen, disponuje také velmi širokou podporou. Neklade žádné specifické požadavky na hardware. Jelikož se z větší části jedná o textově orientovanou hru, která ke svému chodu nevyžaduje zobrazení 3D grafiky nebo využití přídavných apletů, není třeba volit speciální programovací jazyk. PHP tak splňuje všechny zadané požadavky a bylo i nevysloveným přáním majitele webu zachovat stávající jazyk, za předpokladu, že se nenajde vhodnější řešení. Další důležitým rozhodnutím je to, jakou podobu PHP využít.

Čisté PHP - Hlavní výhodou tohoto řešení je naprostá volnost a možnost naprogramovat si funkce dle vlastních potřeb. To dává vzniknout velmi čistému kódu bez nepotřebných rozšíření či kusů kódu. Avšak tato výhoda je zároveň i jeho nevýhodou. Nutnost vytvářet veškeré funkce od nuly je velmi časově náročná. Navíc takto napsané funkce nemusejí být správně zabezpečené a dostatečně otestované, a mohou se v nich vyskytovat bezpečnostní hrozby. Sktruktura kódu není vyžadována, a ani neexistují daná pravidla, jak správně kód psát. Což při práci ve více lidech a nedodržování žádných konvencí, či kombinací obojího, dává za výsledek velmi nečitelný a špatně udržitelný kód. Nespornou výhodou je ovšem rychlost a relativně nízká paměťová a výkonnostní náročnost. To avšak platí pouze pro malé aplikace. U rozsáhlejších a složitějších aplikací je pak rozdíl roven skoro nule.

Rámec - je aplikace vyvíjená více programátory nebo komunitou a nejčastěji se vyznačuje pevně danou strukturou kódu. Poskytuje základní funkčnost jako je komunikace s databází, generování formulářů, zabezpečení a oprávnění uživatelů. Odpadá tedy nutnost tzv. „znovuobjevení kola“, kdy je v takovém rámci většina základních věcí už předem implementovaná a není potřebné se těmito podřadnými úkoly zabývat. Dostává se většího prostoru věnovat se důležitějším problémům. Samotný rámec má už od základu strukturu, která se většinou během psaní sama udržuje, což zajistí, že kód je kvalitní a dobře čitelný. To dává možnost takovou aplikaci velice snadno a dlouhodobě udržovat. Je vyvíjen a používán kolektivem lidí, díky tomu je valná většina rámců velmi dobře otestovaná a bez chyb. Také je zde velká pravděpodobnost, že specifické rozšíření, které bude potřeba implementovat do systému, už někdo naprogramoval, což ušetří potřebný čas na jeho vývoj a testování. Navíc, většina rámců obsahuje návrhový vzor, nejčastěji pak MVC. MVC, neboli také Model-view-controller, je návrhový vzor rozdělující aplikaci do třech základních nezávislých komponent. **Model** - aplikační (datový) objekt, **view** - prezentační objekt (grafika) nebo také uživatelské rozhraní a **controler** - mapující vstupy na akce (řídící a funkční logika). To umožňuje znovu použitelnost kódu [6]. Díky MVC je možné do jisté míry modifikovat komponenty bez zásadního vlivu na ostatní komponenty. Rámec jako takový má však i své nevýhody. Jelikož je to komplexní systém obsahující spoustu rozšíření a doplňků, jeho rychlost není příliš velká. Takové množství potřebných souborů se pak samozřejmě negativně projeví u nároků na paměť a výkon. Toto řešení není tak čisté jako vlastnoručně napsaný kód, který obsahuje pouze nutné náležitosti.

3.1.1 Existující rámce

V dnešní době existuje snad největší počet rámců založených právě na PHP. Valná většina z nich si je velmi podobná, liší se pak ve věcech jako je počet implementovaných knihoven a komponent, nebo verzí použitého PHP. Mnoho z těchto rámců, jsou už několikáté verze v pořadí. Snaží se tak odstranit nedostatky předešlé verze a vyladit svůj systém tak, aby co nejvíce odpovídal potřebám uživatelů. Například implementováním nebo podporou většiny známých a používaných technologii, jako je *AJAX*.

Při volbě správného rámce je rozhodujících několik zásadních faktorů, jimiž jsou robustnost, flexibilita, schopnost generovat kód, databázové vrstvy, šablonovací nástroje, učící křivka a náročnost [3]. Robustnost daného systému určuje počet implementovaných knihoven a komponent. Zde platí pravidlo, že robustnější systémy bývají všeobecně náročnější na hardware a disponují nižší rychlostí. Při nedostatečném výkonu nebo při požadavku na rychlejší rámec je vhodnější volit méně robustní systém, u něhož je směrodatná míra flexibility, tedy v jakém časovém intervalu jsme schopni chybějící komponentu naprogramovat. Řada rámců také disponuje možností generovat kusy kódu pomocí konzolových příkazů, například *Controler*. Šetří to čas, a navíc udržuje kód stále přehledný a strukturovaný. Dalším faktorem je použití databázových vrstev, které už má většina rámců implementovaných ve svém systému, anebo je využito aplikací třetích stran. Nespornou výhodou je použití šablonovacích nástrojů, pomocí kterých lze lépe oddělit aplikační vrstvu od vrstvy prezentační. Poslední, a velmi zásadním prvek je učící křivka a náročnost samotného rámce, tedy za jak dlouho je schopen uživatel se v něm orientovat, a s ním následně pracovat. V tomto bodě se odráží i nutnost toho, aby daný rámec disponoval kvalitní a srozumitelnou dokumentací, která může čas potřebný na učení výrazně zkrátit.

Zend Framework 2 - Je již druhá verze jednoho z nejpoužívanějších rámců vyvíjená pod hlavičkou samotných vývojářů PHP. Jedná se o velmi robustní systém, který už v základu obsahuje velké množství komponent. I přes svou robustnost je překvapivě rychlý díky použití moderních technologií a „vyladěnému“ kódu. Databázová vrstva je zde implementována pomocí jednoduchého *ZendDB* , plnohodnotná *ORM*[1] tak jako šablonovací nástroj nejsou obsaženy přímo v základním balíku, ale lze je velice snadno doimplementovat [4]. Zend má velmi rozvinou komunitu, a není tedy problém najít zde většinu potřebných komponent, které nejsou obsaženy v základním balíku. Díky poměrně špatné dokumentaci není rámec vhodný pro začátečníky, a jeho učící křivka je poměrně dlouhá.

Nette - Je český rámec řadící se do skupiny robustnějších systémů a zároveň nejpoužívanější rámec u nás. Využívá moderní koncept a návrh. Má vlastní, velmi jednoduchou a rychlou databázovou vrstvu nazvanou *NotORM*, která je přizpůsobena přímo pro potřeby Nette [4]. Obsahuje i vlastní šablonovací nástroj jménem *Latte*. Jeho velkou nevýhodou je absence jakéhokoliv generátoru kódu, který by v tak robustním rámci ušetřil spoustu času. Disponuje i českou dokumentací, již zpracování není ovšem nijak „valné“. Vše zachraňuje poměrně ochotná a aktivní komunita na oficiálním fóru. Díky tomu není učící křivka až tak dlouhá, a také díky velmi jednoduchému kódu se dá poměrně rychle naučit. Je tedy vhodný pro větší projekty a také pro začátečníky.

Symfony 2 - Druhá verze velmi oblíbeného robustního rámce, který je vytvořen především pro co největší úsporu času během psaní kódu. V základní verzi neobsahuje mnoho komponent, ty potřebné se však dají stáhnout. Je velmi moderní a založen na technologii kde je vše bráno jako balíček (*Bundle*). Postrádá vlastní databázovou vrstvu, ta je ale nahrazena využitím *Doctrine 2* už v základní verzi. V základní implementaci se také nachází *TWIG*, což představuje šablonovací nástroj, který je maximálně výkonný a rychlý [4]. Jeho hlavní výhodou je propracovaný generátor kódu, který dokáže spolupracovat i s databázovou vrstvou. Jeho nevýhodou je však velmi velká časová náročnost, zapříčiněná některými jeho striktními návrhy. Snížit časovou náročnost se nedaří ani přes velmi kvalitně zpracovanou dokumentaci a nepřeberné množství video návodů, jak s tímto rámcem pracovat.

CakePHP - Představuje kompaktní rámec, který v základním provedení obsahuje většinu nutných komponent. Za inspiraci si bere programovací jazyk *Ruby on rail* a částečně využívá jeho konceptů. Díky své jednoduchosti má velmi přehledný zdrojový kód a je spíše vhodnější pro menší projekty jako jsou například e-shopy, či redakční systémy. Existuje možnost jej velice jednoduše zakomponovat do Zend rámce. Databázová vrstva má pouze jednoduché *ORM*, validace dat probíhá přímo v modelu. Dále pak rámec disponuje vlastním generátorem kódu, který umožňuje s pomocí několika příkazů vytvořit databázové tabulky [4]. Mezi jeho nevýhody patří absence šablonovacího systému a poměrně obtížné implementování komponent třetích stran, které nejsou upraveny pro tento rámec. Přes jeho kompaktnost není rychlost nikterak závratná. Je poměrně jednoduchý na použití a notnou dávkou tomu přispívá velmi dobře zpracovaná dokumentace, která obsahuje i praktické příklady.

[1] ORM - Objektově relační mapování

CodeIgniter - Je minimalistický a jednoduchý rámec, vyvíjený pouze firmou nikoliv komunitou, která se snaží o zpětnou kompatibilitu k PHP4, a tak je rámec dosti zastaralý. Obsahuje pouze nejnutnější a nejžádanější komponenty, což mu zaručuje velmi malou náročnost na hardware a velmi vysokou rychlost. Rámec obsahuje pouze jednoduchý nástroj na správu databáze bez validace vstupních údajů. Také nedisponuje žádným šablonovacím nástrojem, ani generátorem kódu. Vlastní komponenty se implementují velice snadno, ovšem problém opět představuje pro komponenty třetích stran [4]. Díky velmi propracované a detailní dokumentaci s názornými ukázkami se lze naučit velmi rychle.

	Databázová vrstva	Generátor kódu	Učící křivka	Šablonovací nástroje	Robustnost	Rychlost a náročnost na hw	Dokumentace	Modifikovatelnost
Zend Framework 2	3. ZendDB	3.	4.	3. SMARTY	1.	3.	5.	1.
Nette	2. NotORM	-	2.	2. Latte	3.	2.	4.	3.
Symfony 2	1. Doctrine 2	1.	5.	1. TWIG	2.	5.	2.	2.
CakePHP	4. ORM	2.	3.	-	4.	4.	3.	5.
CodeIgniter	-	4.	1.	-	-	1.	1.	4.

Obrázek 3.2: Porovnání jednotlivých rámců podle jejich vlastností

3.1.2 Symfony

Podle výše uvedených kritérií a porovnání bylo nakonec vybráno za hlavní implementační rámec Symfony 2. Hlavní jeho výhodou je čistota kódu a nutnost dodržovat striktní struktury, které rámec udává. Z počátku může být těžší si dodržování struktury osvojit, ale rámec tomu dosti napomáhá a není problém si na tento fakt později zvyknout. Zakládá se na rozdělení podle *frontendu*[2] a *backendu*[3], ty se pak dělí na jednotlivé moduly, v tomto rámci nazvané jako balíčky. Ty se ještě následně dělí na jednotlivé akce.

Právě použití této moderní technologie založené na balíčcích je nespornou výhodou. Každá komponenta v rámci je právě takovýto balíček, který lze v případě potřeby stáhnout a připojit do systému bez sebemenších problémů. Pokud daný balíček nesplňuje specifické požadavky, existuje možnost jej modifikovat dle potřeb. Celý rámec je totiž postaven na modelu *open - source*[4]. To se však výrazně podepisuje na rychlosti samotného rámce, pro jeho zrychlení je tak vhodné použít některý z dostupných *PHP akcelerátorů*, který pomůže v jeho samotném zrychlení. Je koncipován pro co nejrychlejší a nejčistší vývoj celé aplikace, k čemuž dopomáhá velice propracovaný generátor kódu. Generátor kódu je schopen na pár kliknutí a příkazů vytvořit celou vnitřní logiku jednoho bloku včetně samotných tabulek. Tento fakt však s sebou nese i jisté komplikace a to v podobě nutnosti přístupu k příkazové řádce serveru, na kterém je umístěn systém. Tu rámec přímo využívá nejenom při generování kódu, ale také i při jiných činnostech jako je například údržba celého rámce.

Většina českých poskytovatelů webhostingu však tuto službu neposkytuje, což dost výrazně komplikuje nasazení rámce do reálného provozu nebo nutnost změnit poskytovatele webhostingových služeb. Další nevýhodou je pak jeho poměrně velká složitost a delší učící křivka, potřeba mít alespoň malé povědomí o objektově orientovaném návrhu, anebo znalost nějakého z vyšších programovacích jazyků. Malou výhodou zde může být velmi kvalitně

[2] frontend - uživatelsky viditelná část webu
[3] backend - logická část webu, zpracovává data
[4] open-source - model svobodného softwaru. Takový software se dá volně upravovat dle potřeb.

zpracovaná dokumentace, nepřeberné množství video návodů a také poměrně aktivní komunita, která působí například na *stackoverflow*[5]. I přes tak početnou a aktivní komunitu může nastat situace, že specifický balíček, jenž vyžadujeme, zatím nikdo neimplementoval do rámce. Symfony v takovém případě umožňuje velmi jednoduché a rychlé naimplementování vlastního balíčku či upravení komponenty třetích stran.

Tento rámec vyvíjí a financuje společnost *SensioLabs*. Symfony je z velké části inspirováno jinými programovacími jazyky jako *Ruby on rail* nebo *Django (Python rámec)*. Aktuální verze nese číslo 2.6 a je jí využito při zpracování této práce. O samotný vývoj se částečně stará velmi početná komunita, která je v tomhle ohledu dost aktivní. Přispívá k tomu i fakt, že samotný rámec nebo jeho jednotlivé komponenty jsou použity v nejrůznějších systémech, například *Drupal*, *phpBB*, *Joomla!*, *Composer*. Ty vývojem svých komponent přispívají i k vývoji samotného rámce. Snaží se při tom používat nejmodernější technologie, které PHP ve své nejvyšší verzi používá. Symfony také disponuje spoustou známých komponent, které jsou většinou obsaženy v základním balíku. Z těch nejdůležitější uveďme *Doctrine 2*, která slouží k ovládání k *ORM* vrstvy. *PHP unit* na testování rámce a šablonovací nástroj *TWIG* usnadňující tvorbu grafické podoby stránky.

Většina konfiguračních souborů je zapsána ve speciálním *YAML* formátu, existuje ale i možnost využití jiných formátů například *XML* nebo *PHP*. Implicitně se však využívá *YAML*. Ten je primárně určen pro serializaci strukturovaných dat a sestaven tak, aby příkazům rozuměl jak člověk, tak stroj. To poskytuje ohromnou výhodu právě při psaní konfiguračních souborů, které disponují přehlednou hierarchií srozumitelnou všem. Pevně dána je i struktura rámce, ta se s jistou abstrakcí velice nápadně podobná struktuře *MVC*. Obsahuje čtyři hlavní adresáře:

app - Stará se o chod celého rámce, jeho nastavení a konfiguraci. Obsahuje další čtyři podadresáře a spoustu souboru včetně *AppKernel.php* zajišťujícího správu balíčků v rámci a *console* jakožto spustitelný soubor, přes nějž se řídí celý rámec v příkazové řádce. *Config* sdružuje veškeré konfigurační soubory, ať už se jedná o nastavení samotného rámce (*config.yml*), parametry nutné k připojení spojení mezi databázemi (*parameters.yml*), nastavení přesměrování (*routing.yml*), hierarchii přístupových práv na web, anebo bezpečnost (*security.yml*). *Resources* obsahující překlady chybových hlášení, obecné šablony a vlastní upravené chybové stránky (například 404). Dále jsou zde umístěny adresáře *cache* a *logs*.

src - Uchovává většinu kódu. Právě tento adresář v sobě uchovává většinu z návrhového vzoru *MVC*, jenž rámec používá. Hlavním obsahem jsou jednotlivé uživatelsky vytvořené balíčky, které obsahují hlavní logiku celého systému. V balíčcích se pak nacházejí další podadresáře. *Controller* má na starosti řízení jednotlivých akcí příslušných modulů. A jak už jeho název napovídá, v návrhovém vzoru zastupuje funkci controleru. *Entity* obsahuje modely, které se primárně chovají jako PHP objekty a umožnují práci s daty. *Resources* adresář zajištující view, tedy vzhled prezentovaný uživateli, a disponuje mnoha dalšími podadresáři, jež mohou obsahovat specifické šablony modulů, kaskádové styly, obrázky a ve výjimečných případech i vlastní směrování. *Form*, zde se uchovávají funkce na vykreslování a práci s formuláři.

[5]Dostupné na http://stackoverflow.com/questions/tagged/symfony2

vendor - Představuje místo, do nějž jsou ukládány všechny balíčky využívané rámcem a také komponenty třetích stran. Je zde možnost nevyhovující balíčky předělat k obrazu svému, ovšem obsahuje nepřeberné množství složek, které se pak dále složitě větví a daná komponenta nebo balíček se hledá velmi obtížně.

web - Nachází se zde všechny potřebné soubory pro uživatelské zobrazení. Obsahuje podsložky *bundle*, ve kterých jsou všechny soubory (většinou obrázky a kaskádové styly), potřebné k zobrazení pro všechny balíčky. Z důvodu ušetření místa je možné použít pouze reference na tyto soubory v jejich původním umístění, a není tedy potřeba je duplicitně kopírovat do této složky. *CSS* složka může obsahovat speciální kaskádový soubor, který představuje spojení všech potřebných kaskádových stylů do jednoho. Je však nutné po každé změně kaskádových stylů, které obsahuje, vygenerovat tento speciální soubor znova.

3.2 Databáze

V dnešní době lze nalézt velké množství databázových technologií, které poskytují efektivní ukládání dat. Vzhledem k původnímu řešení a předpokladu velkého počtu tabulek, jež potřebují vzájemné propojení pomocí vazeb, je vhodné zvolit *relační model*. Data jsou v něm strukturována do tabulek. V takovém případě odpovídají relacím řádky tabulky a atributům jména sloupců. Tabulka tak obsahuje pouze atomická data, a to taková data, která jsou jednoduchá a nelze je dále dělit. Také je zde obsaženo schéma relace (tabulky). Schéma je tvořeno názvem relace, výčtem jmen atributů a definicí odpovídajících domén [8].

Nejznámějším a nejpoužívanějším jazykem je *SQL*, neboli strukturovaný dotazovací jazyk. Je velice podobný přirozenému lidskému jazyku a jeho syntax je tedy velmi dobře čitelný. Tento jazyk je jedním z nejužívanějších jazyků pro webové aplikace. Existuje tedy nepřeberné množství databázových systémů, využívajících ke komunikaci tento jazyk. Ze zmíněných systémů je potřeba vybrat ten, který disponuje referenční integritou, která je potřebná pro udržení vztahů v tabulkách propojených pomocí relace. Je také důležité, aby daný systém disponoval vlastnostmi ACID, které zajišťují *Atomičnost* - každá transakce je buď dokončena celá, anebo vůbec, *Konzistence* - konzistence databáze, správná reflexe reálného stavu, *Izolovanost* - více probíhajících transakcí je zpracováváno izolovaně tak, jako kdyby byly zpracovány postupně, a ne najednou. *Trvalost* - všechny změny budou po úspěšném dokončení transakce trvalé [8]. Nabízí se tedy použití dvou ověřených databázových systémů:

PostgreSQL - Moderní objektově - relační databázový systém, který se snaží v sobě skloubit všechna potřebná vylepšení. Obsahuje například dědičnost, což je poměrně vzácná vlastnost v databázových systémech. Je velice rozšířený a sdružuje kolem sebe velmi aktivní komunitu.

MySQL - Také velice často používaný systém, je relativně jednoduchý a velmi nenáročný na údržbu. Hlavní výhodou je jeho rychlost, která při správné konfiguraci může dosáhnout až dvounásobné rychlosti oproti ostatním systémům [5]. Není však na takové úrovni jako výše zmíněný PostrgreSQL. Díky jeho podpoře samotným Symfony 2, jednoduchosti, rychlosti a možnosti chybějící vylepšení doimplementovat na softwarové úrovni byl vybrán jako hlavní databázový systém.

Jelikož web nevyužívá jenom dynamická data, nabízí se zde možnost použít kromě techonlogií založených na jazyce SQL i jiné systémy. Například *XML* či *NoSQL* pro statická a tabulková data, kterých se hře nachází spousta.

3.2.1 ORM

Použitím relační databáze a objektově - orientovaného programovacího jazyka s sebou přináší jistá úskalí a nutnost propojit tyto dvě diametrálně odlišné technologie. Ačkoliv obě technologie využívají *entit* znamená to pro každou technologii něco jiného. V případě PHP je to chápáno jako samotný objekt, kdežto v relační databázi představuje entit řádek či množinu řádků v dané tabulce. Existují proto databázové vrstvy jako je právě *ORM*, což je objektově orientované mapování. Toto mapování umožní vzájemnou konverzi tak, aby bylo možné použít objektově - orientovaný programovací jazyk společně s relační databází, která se v zásadě provádí automaticky. Lze jej provádět jak manuálně, tak poloautomaticky za předkladu, že máme specifické požadavky při konverzi. Taková úprava se pak provádí na úrovni zdrojového kódu.

Vrstva pracuje na principu převodu jednotlivých tabulek včetně jejich relací na odpovídající objekt v daném programovacím jazyce. Pozbývá tedy nutnosti pracovat v SQL a tvořit samotné SQL dotazy. Nese to i jistou nevýhodu, kterou představuje nemožnost přistupovat k datům jinak než pomocí identifikátoru v podobě primárního klíče. Samotné ORM se snaží co nejvíce usnadnit vývoj celé aplikace a podporu použitého MVC návrhového vzoru. A to úplnou automatizací veškerých operací jako je čtení, zápis, mazání či úprava dat v tabulkách. Vrstva také umí automaticky převádět datové typy jednolitých položek, které se mohou v programovacím jazyce a relační databázi lišit, anebo nemusí vůbec existovat, jako v případě absence JSON formátu v MySQL databázi. Tím je zajištěna i jistá interní validace dat, ve které se ještě před uložením samotných dat odstraní nebezpečné znaky, jež by mohly způsobit potencionální bezpečnostní hrozbu. Využívá také možnosti zapouzdření některých funkcí do samotného objektu (*GET* a *SET*) [7]. ORM vrstva má také na starosti trvalé (persistentní) uchování dat z objektu pomocí databáze. To znamená, že data, se kterými objekt operoval, nebudou ztracena, ale uložena do relační databáze a je možné je opětovně využít i po restartu aplikace či je může využít jiný uživatel. S tím však vzniká problém, kterým trpí většina databázových vrstev. Nutnost provedení speciální režie kvůli serializaci, případně deserializaci daných dat a také větší paměťové náročnosti. Vrstva si totiž musí uchovávat jak data v původní podobě, s níž pracuje relační databáze, tak i data přemapovaná na objekt. Tím se výrazně celá operace zpomaluje oproti čisté implementaci relační databáze. Opět je i zde poměrně dlouhá učící křivka a poměrně složitá implementace, které se dá předejít pomocí speciálních generátorů kódu, jež jsou implementovány v některých PHP rámcích.

3.2.2 Doctrine 2

Doctrine 2 je jedním z ORM rámců, kterými Symfony 2 disponuje už v základním balíčku. Je implementován jako čisté a transparentní ORM bez jakýchkoliv speciálních funkcí. Díky tomu je možné sledovat co se v něm děje a nikdy neprobíhá žádná softwarová magie. Využívá návrhového vzoru *Data Mapper*, který je reprezentován pomocí *Entity Manageru*. *Entity* v něm uchovávají strukturu tabulek a jejich atributů. Je možné je použít jako přístup k datům a bez nutnosti přístupu k samotným tabulkám lze i nahradil SQL příkazy za *DQL*[6]

[6]Doctrine Query Language (DQL) - dotazovací jazyk využívající především entit a objektů

notaci, která pak využívá názvů entit a jejich proměnných místo názvů tabulek a atributů. Tímto dokáže úplně oddělit databázi od aplikace. Entita k definici svých vlastností využívá speciálních anotací, které se zapisují do komentáře. Ovšem při použití Entit je nutné už při návrhu uvažovat o relační databázi jako o Entitě a ne jako o tabulkách, což může být někdy velmi složité. Doctrine 2 se skládá ze 3 různých vrstev.

Common Shared Libraries - obsahuje základní funkce, rozhraní, třídy a knihovny, které jsou zapotřebí k vytvoření mapovacího objektu. Tato vrstva je schopna fungovat nezávisle na ostatních. Lze ji tedy použít i samostatně.

DBAL (DataBase Abstraction Layer) - knihovna, která poskytuje abstraktní vrstvu pro relační databáze. Odděluje databázi od zbytku aplikace. Zavadí DQL do rámce. DBAL je závislá na předešlé vrstvě a nelze ji bez ní použít.

Object Mappers - vrstva zajištující mapování objektů na relační databázi, jejich načítání a persistivnost. O tuto funkci se stará *Object Manager*, využívá k tomu obě předešlé vrstvy, bez kterých by to nebylo možné [1].

Kapitola 4

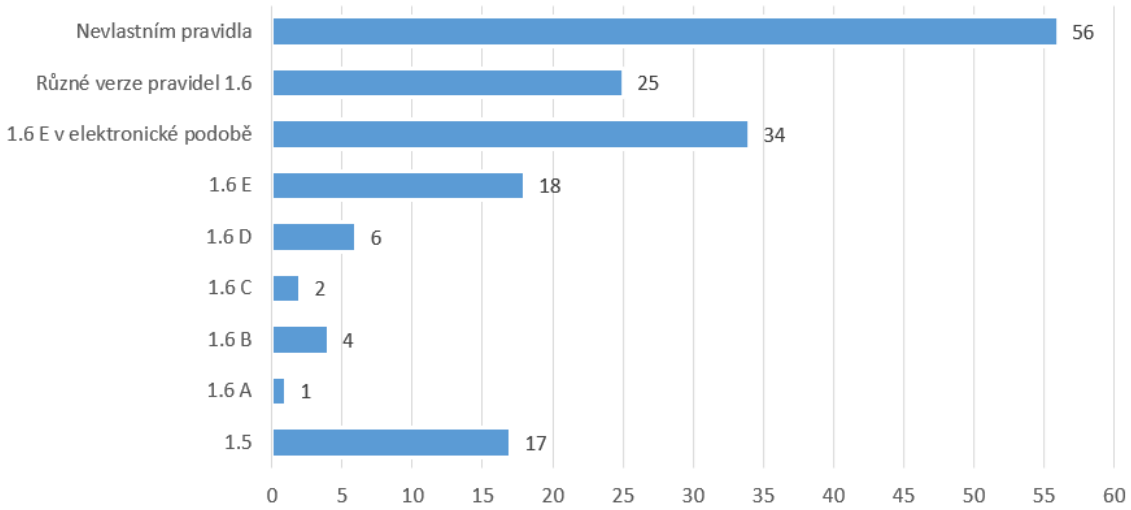
Návrh

Stávající systém je velmi rozsáhlý a jeho kompletní přetvoření daleko přesahuje rozsah bakalářské práce. Bylo tedy nutné vybrat pouze důležité moduly, které budou implementovány, a na nichž se odrazí celá budoucí hierarchie webového portálu. Tyto modulu musí být zvoleny tak, aby měly určitou logiku, která je vyžadována k částečnému hraní hry. Navíc zde působí globální faktory, jež je potřeba zohlednit už při návrhu, aby v pozdějších fázích vývoje nezpůsobily komplikace.

Jedním z faktorů je volba pravidel Dračího doupěte, podle kterých se bude tvořit veškerá vnitřní logika. Pravidla hry zahrnuje spoustu verzí a to i díky svému dlouhodobému vývoji. Pravidla se však mohou lišit buď pouhými detaily, anebo obsahovat marginální rozdíly, které mohou mít zásadní vliv na celý herní systém. Pozdější aplikace takto rozsáhlých změn může mít fatální dopad na celý systém. Tento fakt lze pozorovat na stávajícím systému, kdy se v průběhu let měnila pravidla podle jejich aktuálnosti, což zanechalo v systému spoustu herních chyb a nuancí. Je proto více než důležité si už v začátku určit podle jakých pravidel se bude daný systém tvořit. Toto rozhodnutí však nestojí na mě jako vývojáři, nýbrž na samotné herní komunitě. Herní komunita dostala možnost vyjádřit svůj názor v anketě, která běžela na webu. Z grafu (4.1) lze usuzovat, že nejvíce lidí, kteří vlastní pravidla, disponují verzí s označením *1.6 E*, což je také poslední oficiální verze před ukončení vývoje a také jediná verze, k níž existuje i elektronická verze pravidel. Z tohoto důvodu i po následné dohodě s majitelem webu a administrátory, bylo rozhodnuto o využití této verze pravidel při vývoji.

Dalším důležitým prvkem je propracovaná struktura webu. Zde se ovšem nestačí zaměřit pouze na moduly zahrnuté v bakalářské práci, ale pohlížet na celou věc jako celek a vypracovat strukturu tak, aby bylo možné do ní v budoucnu bez větších problémů integrovat nové moduly a propojit je s těmi stávajícími. Jako vhodné řešením se zde jeví naimplementovat každou významnější stránku, tedy modul, jako samostatný objekt. Ten pak může být děděn. Každý z modulů tak bude mít vlastní kontrolér, což zajistí čistotu kódu a hierarchii. Dané řešení obnáší však i jisté nevýhody jako je například větší paměťová náročnost při zmíněném počtu objektů.

Vzniká zde i snaha úplně oddělit herní postavu od uživatele, což by ve výsledku znamenalo, že v herních modulech vystupuje uživatel jako postava a ve zbytku jen jako uživatel. Navíc, pokud postava umře, odpadá nutnost zakládat s novou postavou i nový účet. Tato snaha zkvalitní hraní za postavu, jelikož se někteří hráči bojí hrát za svou postavu i z důvodu, že by při úmrtí postavy přišli o svoji unikátní přezdívku, která se v tuto chvíli používá jako přihlašovací jméno a zároveň i jméno postavy. Výrazně tak pomůže oddělit neherní část webu od té herní.



Obrázek 4.1: Graf udávající počet uživatelů daných verzí pravidel
Zdroj:http://www.immortalfighters.net/?module=ankety

Zapotřebí je také vymyslet nové řešení přístupových práv a práv administrátorů na webu. Musí být ovšem zachována původní reprezentace těchto práv a pouze se upraví logika tak, aby vyhovovala novému systému a pokrývala problémy, které vznikaly ve stávající verzi (2.3.2).

4.1 Struktura webu

Při návrhu nové struktury se částečně vychází ze současného řešení zavedeného na stávajícím webu. Použije se tedy model, jenž rozděluje jednotlivé moduly (stránky) na dvě velké kategorie. Herní, kde moduly mají přímý vliv na hru a ovlivňují pouze herní postavu, nikoliv hráče, a neherní moduly, které naopak vůbec nesouvisí s hrou a hráč zde není reprezentován svou herní postavou, nýbrž sám sebou, tedy jako hráč. V této kategorii mohou být moduly pro ukrácení času anebo sloužit k vytváření herního obsahu (různé editory kouzel, předmětů atd.). Obě kategorie by pak měly disponovat jinými daty. Herní data by měla čerpat informace o uživateli z objektu *Character* a neherní by naopak měla využívat informace o uživateli z objektu *User*. Struktura webu je tak následující:

Hlavní stránka - příchozímu uživateli se zobrazí stránka, na které se dozvídá něco o samotné hře a může se přihlásit či zaregistrovat. Po úspěšném přihlášení je přesměrován na tuto hlavní stránku, odkud vede rozcestník na další moduly. Ty jsou ovšem přístupné pouze pro přihlášené uživatele. Také se zde dozvídá nejnovější novinky a dění na webu.

Forum - uživateli jsou zobrazena všechna dostupná vlákna (thready) a jejich aktuální stav udávající změnu na daném vlákně (přidání nebo odebrání příspěvků). Podle daných přístupových práv se pak uživateli vlákno zobrazí či nikoliv, je oprávněn do něj pouze nahlížet, anebo má kompletní přístup a může tedy vkládat i příspěvky. Ty pak může upravovat či mazat a ostatní uživatelé si tyto příspěvek mohou i oblíbit. Každé

vlákno slouží specifickým účelům, nejčastěji však ke komunikaci, řešení problémů či sdílení důležitých věcí.

Quest - je velmi podobný vláknu fóra. Počet uživatelů, kteří mohou do daného Questu, přispívat je omezený a jeden uživatel (Pán jeskyně) vede hru. Zde už ale píší uživatelé příspěvky za svou postavu, nikoliv za sebe jako hráče. Dané příspěvky má pak Pán jeskyně možnost ohodnotit zkušenostmi v závislosti na tom, jak je daný příspěvek kvalitní a zdali odpovídá správnému role-playingu postavy. Příspěvky je možné odkrýt pouze vybraným hráčům, čímž dochází k tzv. „hernímu šeptání“. Uživatelům, kteří nejsou v daném Questu, se zobrazí pouze text viditelný všem.

Uživatelský (herní) profil - stránka zobrazující herní údaje zvoleného hráče. Pokud si uživatel prohlíží svůj vlastní profil, jsou mu zobrazeny všechny dostupné informace. Pokud si však prohlíží cizí profil, zobrazují se mu pouze údaje, které by mohl v normálním světě vypozorovat. Například oblečení a zbraně, které postava drží. Nevidí však ani jeho životy, vlastnosti či povahu. Existuje jedna výjimka. Pokud si uživatel, který je Pán jeskyně prohlíží profil hráče v jeho Questu, vidí profil tak, jako by si prohlížel svůj vlastní a navíc v něm může provádět změny.

Obchod - zde může uživatel nakoupit či prodávat různé herní předměty, které pak využije v příbězích.

4.2 Hierarchie uživatelských rolí

Zde vzniká problém, že původně využitá hierarchie uživatelských rolí musí zůstat zachována, a tedy být modifikovaná tak, aby se dala použit v novém systému a zároveň pokryla možnost přidat nové role, což do této doby vůbec nebylo možné. Existují tyto následující role:

Uživatel - tato role je přiřazena ihned po registraci.

Certifikovaný uživatel, který finančně přispěl na chod serveru, moc se neliší od běžného uživatele, získává pouze určité výhody.

VIP uživatel je bývalý administrátor či cech mistr či uživatel, který přispěl významnou měrou na chod webu.

Cech mistr vede cech, což je herní obdoba společenství.

Nižší admin - editor určitého herního modulu.

Admin - většinou starší hráč, který má pod sebou nižší adminy. Spravují jeden konkrétní herní modul, například tvoření předmětů nebo hádanek.

Specializovaný admin má na starosti speciální odvětví webu, jakým je bezpečnost.

Hlavní admin uživatelé, jež se starají o chod celého webu, přerozdělují práci a zajištují kontrolu.

Programátor zpravidla nehrající uživatel starající se o vývoj webu a opravy vzniklých chyb.

Role mají svoji speciální zkratku a jsou odlišeny podle barevných štítů, jež se zobrazují u jména. Tyto role lze rozšířit o další funkce, které se většinou týkají specializace daného administrátora v daném modulu. k tomu všemu může uživatel ještě získat další roli v případě, že je Pán jeskyně, což jej opravňuje vést Quest.

Začátečník - uživatel ve výcviku, učí se, jak správně tvořit příběh a jak vést hráče. Musí vést úspěšně do konce jeden Quest pod vedením *experta*

Pokročilý - uživatel, jež může vést Quest a získat za něj zkušenosti.

Expert - velmi zkušený uživatel, který má právo cvičit nové adepty.

I tyto role mají svou zkratku a označení, podle kterého lze poznat, jak zkušený Pán jeskyně uživatel je. Tyto role nabízejí nepřeberné množství kombinací. Návrh není v zásadě špatný, má ovšem své chyby. Ty se dají odstranit drobnou úpravou, která zahrnuje rozdělení rolí do tří hlavních tříd: *štít* neboli postavení na webu, *PJ* role určující status zkušenosti daného hráče a *speciální* dávající uživateli speciální práva nad rámec jeho role štítu. Hierarchii je dále možno specifikovat pomocí tří abstraktních rolí, jež jsou zobrazeny v diagramu užití (4.2). Diagram tak nezahrnuje speciální třídu, která je pro něj příliš specifická.

4.3 Reprezentace dat v databázi

Jak je vidět v E - R diagramu (4.3), schéma databáze je velmi obsáhlé, přičemž v tomto stavu pokrývá zhruba třicet procent celkové hratelnosti. Je to způsobeno zejména nepřeberným množstvím různých spojovacích tabulek, které tak umožnují jistou modularitu.

Představení entit a jejich funkcí, kterou jsou pak převedeny pomocí ORM na objekty. Všechny tyto entity disponují unikátními identifikátory:

User - uchovává veškeré nezbytně nutné informace o uživateli sloužící k jeho identifikaci a vstupu na webový portál. Dále zahrnuje také IP adresu, ze které se uživatel přihlásil, jeho poslední aktivitu na webu, kvůli automatickému odhlašování a zdali je daný uživatel přihlášen. V poslední řadě také obsahuje role, kterými uživatel disponuje. Tyto informace slouží k identifikaci při zobrazování autora příspěvku v daném vlákně a k identifikaci vlastníka herní postavy.

Thread - sdružuje data o vlákně, název, který slouží k identifikaci a také popis daného vlákna. Zahrnuje jednotlivá práva na zápis, čtení a editaci celkového počtu příspěvků, které jsou v příslušném vlákně a jeho poslední změnu, která umožní kontrolovat, zdali od poslední návštěvy uživatele byla ve vláknu provedena změna.

postThread - jednotlivé příspěvky ve vlákně. Obsahují identifikátory příslušného vlákna, ve kterém byly zapsány a identifikátor uživatele, jenž příspěvek napsal. Dále pak čas kdy byl příspěvek napsán a dále samotný text příspěvků. Jeho vlastní identifikátor a identifikátor uživatele pak figurují ve spojovací tabulce označující příspěvky, které si některý z uživatelů oblíbil (*postThreadLikes*).

Character - ústřední entita zahrnující v sobě všechna nutná data pro hraní Dračího doupěte. Neobsahuje však data, která se dají dopočítat na softwarové úrovni (v rámci PHP objektu). Disponuje vlastním jménem postavy, které může být jiné než přihlašovací jméno uživatele a identifikátor uživatele, jenž danou postavu vlastní. Samotný

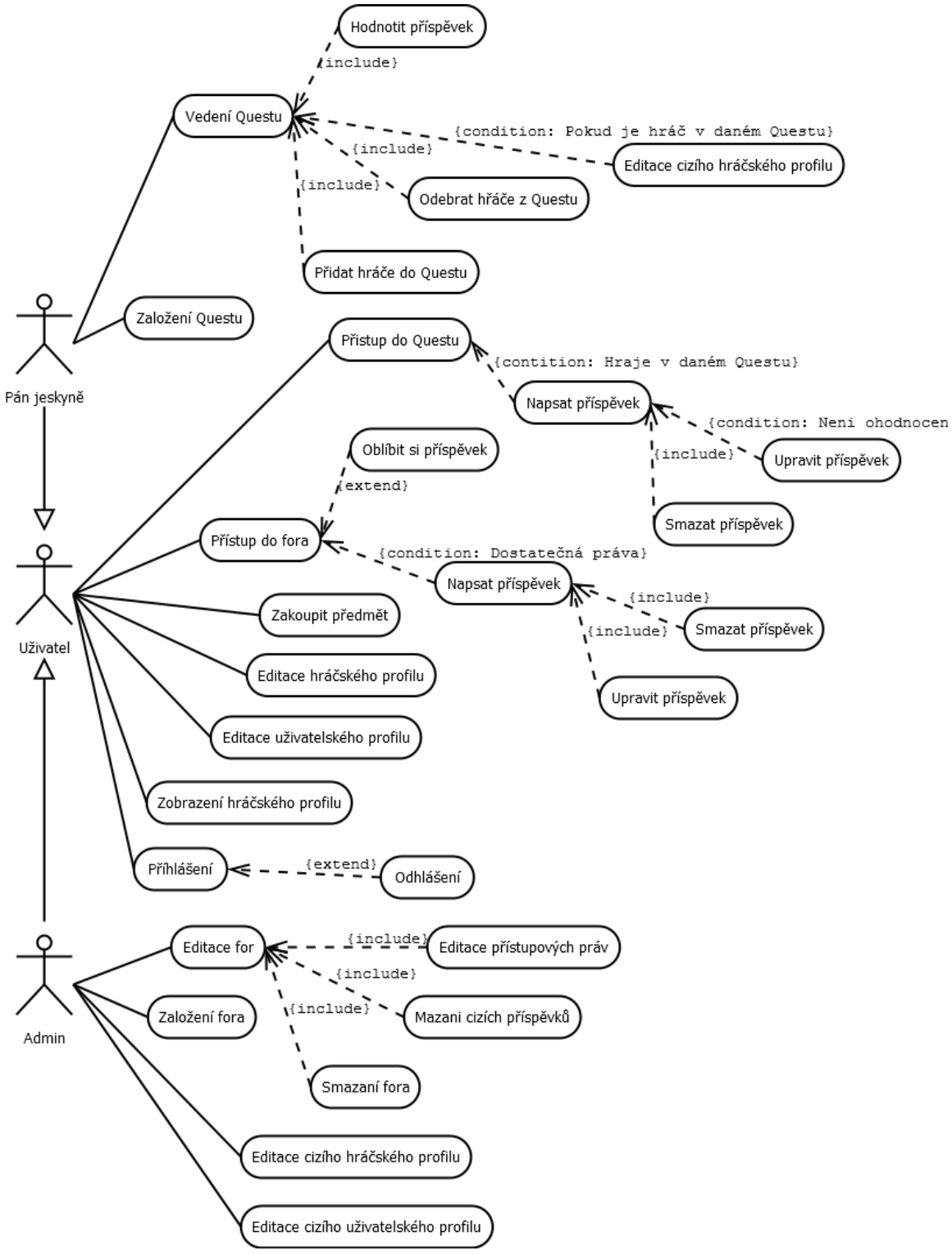
identifikátor herní postavy figuruje jako cizí klíč v některých spojovacích tabulkách, jako jsou například tabulka zajišťující kouzla (*characterSpell*), kterými daná postava disponuje, anebo spojovací tabulka itemů vlastněných postavou (*characterItem*).

Quest - tabulka prototypově velmi podobná tabulce **Thread**. Zde se ovšem uchovávají data o daném příběhu, identifikátor herní postavy, který jej vede (tedy, kdo je PJ), název příslušného příběhu, jeho popis, maximální počet hráčů a další informace nutné k založení příběhu, které dány interními pravidly webu pro založení příběhu. Na jednotlivé záznamy je pak navázána spojovací tabulka (*characterInQuest*), jenž určuje identifikátory herních postav v daném příběhu, jejich aktuální počet životů, magenergie a povolení různých funkcí, které mohou vykonávat během samotného příběhu.

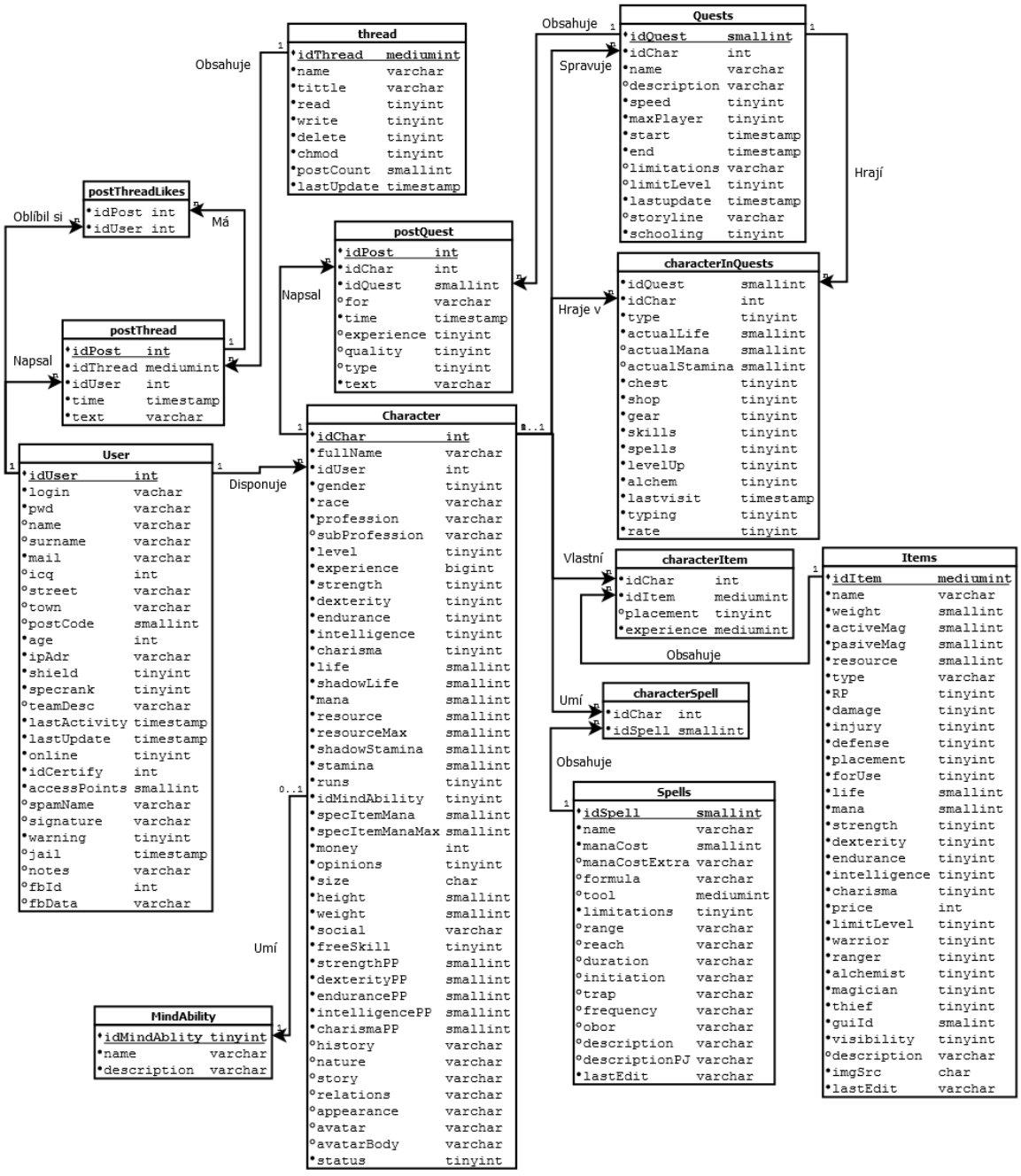
postQuest - obsahuje jednotlivé záznamy o příspěvcích v příběhu. Využívá k tomu dvou cizích identifikátorů, a to příslušného Questu a uživatele, jenž napsal příspěvek. Zahrnuje také čas vložení příspěvku a pro koho jediný příspěvek určen. Obsahuje také informaci o zkušenostech získaných za daný příspěvek a o jeho kvalitě. Je zde také položka, která určuje, zda je příspěvek herní, neherní a do budoucna se zde počítá se zavedením speciálních typů příspěvků, jako jsou příšery, kouzla atd. A samozřejmě obsahuje samotné znění daného příspěvku.

Spells - tabulka zaštiťující kouzla, která mohou herní postavy využívat. Zde je použit obecný formát tak, aby každé povolání nemuselo mít zvláštní tabulku s kouzly, ale existovala jedna obecná. Proto, zde není většina atributu povinná. Povinné jsou jenom obecné náležitosti, jako je název daného kouzla, jeho manová náročnost, rozsah, dosah a popisy kouzla. Další položky jsou volitelné, jelikož každé povolání využívá jiná data u svých kouzel. Kouzelníci potřebují znát magické zaklínadlo, kdežto Hraničáři potřebují znát předmět, jenž musí být použit pro správné účinkování kouzla. Identifikátor kouzla je pak vyžíván ve spojovací tabulce *characterSpell*, která určuje, jaký uživatel disponuje danými kouzly.

Items - sdružuje data o předmětech využívaných ve hře. Opět je tato tabulka velmi obecná a má mnoho nepovinných atributů, kvůli rozličným druhům předmětů vyskytujících se v herním světě. Obsahuje ale i povinné atributy jako je jméno, typ daného předmětu, váha, popis, informace, zda nepatří nějakému specifickému cechu, viditelnost v deníku postavy, adresu na grafickou reprezentaci (obrázek), omezení pro jisté povolání, cena, a zdali se jedná o herní, či neherní itemy (odměny za různé neherní soutěže, jako je například zlatý brk za literární soutěž). Jeho identifikátor je taktéž využit ve spojovací tabulce *characterItem* určující vybavení herní postavy.



Obrázek 4.2: Abstraktní Use Case diagram



Obrázek 4.3: E-R diagram

Kapitola 5

Implementace

Rámec dosti zjednodušuje automatické psaní zdrojového kódu, které však není schopno pokrýt všechny speciální případy, které hra představuje. Proto bylo nutné některé náležitosti upravit tak, aby vyhovovaly našim potřebám. Některá řešení bylo třeba vymyslet od úplného začátku a v jiných se dalo využít už naimplementovaných interních funkcí využitých technologií, jež prošly jistou modifikací.

5.1 Implementace balíčků v Symfony 2

Velmi zajímavou části je samotná implementace modulů v Symfony 2. Moduly jsou řešeny jako jednotlivé balíčky ve složce src (3.1.2), to ovšem znamená, že každý významnější modul i jeho separátní části by měly mít vlastní balíček. V tuto chvíli by tento fakt nepředstavoval velký problém, ovšem do budoucna by s přibývajícími moduly vznikalo i mnoho nových a leckdy zbytečných balíčků. Ty se dají sloučit do jednoho funkčního bloku a ten pak implementovat jako obecný balíček. Tím je odstraněna nutnost do každého balíčku složitě přidávat referenci na ostatní využívané balíčky, protože jsou umístěny v jednom jmenném prostoru obecného balíčku, a mají tedy k sobě vzájemný přístup.

ForumBundle - balíček sdružující v sobě všechny objekty *Thread* a *Post*, jež jsou nutné k zobrazení vláken a jednotlivých příspěvků v nich. Dále obsahuje jejich kontroléry a celou logiku, která je zapotřebí při manipulaci s objekty. Představuje také formuláře sloužící k odeslání nebo editaci příspěvků a vytvoření či editaci celého vlákna. V poslední řadě také obsahuje šablony jednotlivých stránek a jejich kaskádové styly nutné ke grafické reprezentaci. Je zde vytvořena speciální logika práv, o níž se zmíním v následující sekci (5.2).

UserBundle - Balíček obsahující objekt *User* a *Character*. Jeho funkcí je zajištění jejich vzájemného propojení a vyhledávání uživatelů. Má také na starosti zobrazení uživatelského profilu, registraci nového uživatele a celou přihlašovací logiku včetně validace dat a přesměrování do vnitřního systému. Do budoucna by měl obsahovat i modul zpráv, administrativní profil či hřbitov postav. Umožňuje tak uživateli vlastnit více herních postav. Avšak pouze jedna může být aktivní, ostatní jsou v takovém případě pozastaveny a nelze s nimi hrát, nicméně v systému stále existují. Tímto krokem lze zabránit neustálému odlivu postav, kdy hráče přestane daná kombinace povolání a rasy bavit. Má možnost tuto postavu upozadit a vytvořit si novou.

QuestBundle - Balíček zaštiťující funkce nad Questem, jako je vytváření Questů nových, správa či jejich pozdější zálohování do speciálních formátů. Úzce spolupracuje s *UserBundlem*, který hojně využívá právě uživatelských dat. Ta používá jenom jako referenci, jelikož má vlastní tabulku, která udržuje stav atributů herní postavy pouze pro daný příběh. Tím dává možnost jednomu hráči figurovat ve více příbězích nezávisle na sobě. V každém z nich pak také může hráč disponovat jiným počtem životů, many a dalšími vlastnostmi, což na stávajícím webu nelze.

ItemBundle - Balíček, v němž jsou veškeré objekty uchovávající informace o předmětech nacházejících se ve hře a veškerou jejich logiku. Tedy informaci, kdo má daný předmět nasazený a na jakém místě. Do budoucna se snad povede implementovat velikost bonusu nebo postihu v závislosti na umístění popsaná v úvodní kapitole (2.1). Dále zde budou editory předmětů, které mohou pověření administrátoři přidávat do hry. Bude obsahovat i editor obrázku pro předměty nebo také třeba logiku obchodů a aukcí. V poslední řadě zde bude v budoucnu implementován modul alchymistické laboratoře dovolující herním postavám vyrábět předem připravené předměty.

5.2 Role a přístupová práva

Základní problém kombinace rolí a přístupových práv z něj plynoucí byl nastíněn v sekci hierarchie uživatelských rolí (4.2). Proto bylo více než nutné tyto kombinace pokrýt. V tomto ohledu pomáhá samotné Symfony 2. Již v základu disponuje hierarchií rolí s dědičností. To umožnuje ostatním rolím dědit ty s nižší hodnotou, a není tedy nutné pro každou roli definovat její vlastní pravidla, pouze rozšiřovat ta základní. *ROLE_USER* je základní rolí, jenž všechny ostatní dědí a vycházejí z ní. Čím výše se daná role nachází, tím více obsahuje rolí. To pak umožňuje tvořit nejrůznější kombinace, které jsou zapsány v *JSON* ve formátu: [‘Nejvyšší role’, ‘speciální role’, ‘základní role’]. S hierarchií jsou spojené i interní funkce, jež dovolují rekurzivně procházet toto pole a určit, zda uživatel disponuje příslušnými rolemi. Tyto funkce ušetří spoustu času při validaci oprávnění. Nelze jich však využít všude, například v dotazech to není možné.

Tento nedostatek dal vzniknout nutnosti implementovat vlastní funkci ověřující dané role. Zde však vznikl problém s formátem JSON. Bylo nutné provést konverzi na přijatelnější formát z důvodu, že nad JSON formátem nelze provádět podmíněné SQL dotazy. Jako formát byla nakonec zvolena bitová posloupnost, kde každý bit reprezentuje jednu roli, a je tedy možné je libovolně kombinovat, přičemž vznikají unikátní kombinace čísel. Navíc, pokud bude v budoucnu nutnost přidat další speciální roli, přidá se pouze její bit navíc. V SQL dotazech se pak používá logický *AND*, který určí, jestli daný hráč disponuje požadovanou rolí. Pro reprezentaci dat byla vytvořena speciální tabulka (5.1), jenž popisuje kombinace rolí, jejich binární hodnoty a v závorkách pak dekadickou hodnotu, která se používá například ve formulářích.

5.3 Bezpečnost a validace dat

Bezpečnost ve stávajícím řešení je docela zanedbávaná věc. Proto, a také částečně díky rámci, jsou zde bezpečnostní kontroly přísnější a vyskytují se hned na několika úrovních. První úrovní je samotné přesměrování (routování), kdy dochází ke kontrole oprávnění ke vstupu, a při nedostatečných právech je uživatel přesměrován na stránku s chybovým hláše-

	Programátor	Hlavní admin	Specialiozovaný admin	Admin	Nižší admin	Cech mistr	VIP uživatel	Certifikovaný uživatel	Uživatel
	000111111111 (511)	000011111111 (255)	000001111111 (127)	000000111111 (63)	000000011111 (31)	000000001111 (15)	000000000111 (7)	000000000011 (3)	000000000001 (1)
PJ	001111111111 (1023)	001011111111 (767)	001001111111 (639)	001000111111 (575)	001000011111 (543)	001000001111 (527)	001000000111 (519)	001000000011 (515)	001000000001 (513)
PJ2	011111111111 (2047)	011011111111 (1791)	011001111111 (1663)	011000111111 (1599)	011000011111 (1567)	011000001111 (1551)	011000000111 (1543)	011000000011 (1539)	011000000001 (1537)
PJ3	111111111111 (4095)	111011111111 (3839)	111001111111 (3711)	111000111111 (3647)	111000011111 (3615)	111000001111 (3599)	111000000111 (3591)	111000000011 (3587)	111000000001 (3585)

Obrázek 5.1: Přehled rolí a jejich binární reprezentace.

ním. Kontrola oprávnění je pak zavedena i v šablonách, a pokud uživatel nedisponuje potřebnými oprávněními, není mu ani zobrazeno tlačítko pro vykonání požadované akce.

Další úroveň je při samotném vykonávání akce, kdy opět probíhá kontrola oprávnění pro danou akci. Ta se vykoná před samotnou akcí a nemůže tak dojít k odesláním požadavku bez dostatečných práv. Poslední úrovní si pak zajišťuje samotné Symfony. To si na databázové úrovni ověřuje, že ukládaná data neobsahují potencionálně škodlivý kód a mají správný datový typ. V případě nesprávného datového typu pak dochází ke konverzi dat, aby se zamezilo chybnému ukládání dat. O toto se i částečně stará validace dat. Validace je rozdělena do několika úrovní. Tou první je základní validace datových typů ve formulářových vstupech pomoci *HTML5 validátoru*. Dále se jedná o validaci pomocí serveru. Při odesílání formuláře se zkoumá, zda jsou data validní a neobsahují potencionálně závadný kód, a odpovídají-li požadovaným vstupům, což se testuje pomocí *ORM vrstvy*. U registrace je pak ještě navíc zkoumáno za pomocí ORM notace *UniqueEntity*, zda jsou zadaná data unikátní, a nedojde tak k nežádoucí duplicitě dat a následné kolizi.

Kapitola 6

Závěr

V textu byla mnohokrát zmiňována náročnost a dlouhá učící křivka Symfony rámce a jeho technologií. To vše vyvažuje možnost po překonání prvotních obtíží tvořit velmi kvalitní a strukturovaný kód, na němž může pracovat i více vývojářů najednou s tím, že struktura kódu zůstane zachována. Jednou z hlavních nevýhod převažujících nad samotnou náročností je nutnost disponovat vlastním serverem nebo webhostingovou službou, jenž umožňuje přístup k příkazové řádce daného serveru, na němž je aplikace uložena. Bez toho totiž není možné Symfony rámec, a tedy ani výslednou aplikaci zprovoznit. To byl také problém, proč nedošlo k samotnému funkčnímu testování webové aplikaci.

Všechna tato fakta se neblaze podepsala na výsledné webové aplikaci, která je velmi strohá a umožnuje pouze základní funkce. I takto naimplementovaný systém stačil k tomu, aby ukázal svůj ohromný potenciál tkvící zejména ve velmi v propracovaném návrhu databáze, nové struktuře webu a použití objektů. Ale také k tomu, aby se Symfony ukázal, jak ohromně silným vývojovým nástrojem je. Vývoj systému tedy přesáhne rozsah bakalářské práce a bude stále vyvíjen a udržován až do doby, kdy jeho funkční část bude moci nahradit stávající systém, aniž by hráči pocítili výrazné změny nebo museli přijmout nějaká herní omezení. Pomůže tak vylepšit dosavadní hru a také přilákat nové hráče. Ohromný potenciál tkví hlavně v použití moderních technologií, které umožnují implementovat moduly, jež v minulosti nebyly možné, nebo byla jejich implementace velmi složitá. Už nyní je známo sedm takovýchto modulů. Ty by měly výrazně zkvalitnit prožitek z hraní. To je ovšem běh na dlouho trať, která vyžaduje programátorské dovednosti a velké množství času.

Literatura

[1] Doctrine oficiální dokumentace [online]. Dostupné z: http://www.doctrine-project.org/about.html, 2014 [cit. 2015-05-14].

[2] BURGET, R.: Tvorba webových stránek: Média a CSS 3 [online]. Dostupné z: https://www.fit.vutbr.cz/study/courses/ITW/private/prednasky/p08/prednaska.html, 2015 [cit. 2015-05-18], materiály k přednáškám.

[3] GAŠPARÍK, P.: Ako si vybrať ten najvhodnejší PHP framework? [online]. Dostupné z: http://www.zajtra.sk/programovanie/821/ako-si-vybrat-ten-najvhodnejsi-php-framework, 2012 [cit. 2015-05-08].

[4] GAŠPARÍK, P.: Veľký prehľad najpoužívanejších PHP frameworkov [online]. Dostupné z: http://www.zajtra.sk/programovanie/820/velky-prehlad-najpouzivanejsich-php-frameworkov, 2012 [cit. 2015-05-08].

[5] KOŠÁREK, L.: Výkonnostní srovnání relačních databází. Dostupné z: http://is.muni.cz/th/256433/fi_b_b1/Vykonnostni_srovnani_relacnich%_databazi.pdf, 2010.

[6] KŘIVKA, Z.: Principy programovacích jazyků a OOP: Návrhové vzory [online]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IPP-IT/lectures/2012-2013/IPP_08_DP_cz_ZK_2013-04-01.pdf?cid=9999, 2013 [cit. 2015-05-08], materiály k přednáškám.

[7] TRÖSTER, F.: ORM frameworky pro PHP5: Obecný úvod [online]. Dostupné z: http://www.zdrojak.cz/clanky/orm-frameworky-pro-php5-obecny-uvod/, 2010 [cit. 2015-05-14].

[8] ZENDULKA, J.: Studijní opora k předmětu Databázové systémy [online]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IDS-IT/texts/IDS_predn.pdf?cid=9355, 2005 [cit. 2015-05-13].

Dodatek A

Obsah CD

`src` - složka s kompletními zdrojovými kódy výsledné aplikace

`src/databaseBP.sql` - soubor obsahující SQL příkazy k vytvoření databáze

`src/readme.txt` - návod k instalaci aplikace

`text` - zdrojové soubory technické zprávy

`bp-xkubic34.pdf` - výsledná technická zpráva ve formátu PDF

Dodatek B

Manual

Tato kapitola obsahuje popis instalace rámce a aplikace na server

Požadavky

Pro spuštění a správné fungování aplikace je nutné disponovat *PHP* ve verzi 5.3.3 a vyšší. Dále také *MySQL server* ve verzi 4.2 a vyšší. Zdrojové kódy obsahují celý funkční rámec a jeho balíčky, jenž jsou uzpůsobeny tak, aby nebylo nutné do jejich struktury nijak zasahovat, či nevznikala nutnost přidávat další dodatečné balíčky.

Spuštění

Před samotným spuštěním aplikace je nutné naimportovat databázi s předem připraveným jménem *immortalfighters*, do níž se nahrají data ze souboru `src/databaseBP.sql`. Poté může následovat spuštění aplikace. To se děje přes příkazový řádek, přičemž je nutné nacházet se v hlavní složce samotné aplikace (`src/bakalarka/`) zadním příkazu:

```
php app/console server:run
```

Ten zajistí, že se rámec a všechny jeho potřebné balíčky spustí a uvedou aplikaci do chodu. Ta je pak dostupná z `http://localhost:8000`.

Před samotným testováním nebo vyvíjením aplikace je doporučeno synchronizovat PHP objekty a relační databázi. Obě schémata se porovnají a vyhodnotí se změny, které se následně promítnou do samotného zdrojového kódu nebo schématu databáze. Synchronizaci lze zajistit příkazem:

```
php app/console doctrine:schema:update --force
```

V případě problémů s nekorektním zobrazováním aplikace či nefungováním některých funkcí je vhodné promazat *cache* paměť rámce pomocí příkazu:

```
php app/console cache:clear --env=prod
```