Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Vytvoření webové hry za pomocí MVC architektury

Bakalářská práce

Autor: Radek Jirsa
Studijní obor: Aplikovaná informatika

Vedoucí práce: doc. RNDr. Petra Poulová, Ph.D.

Hradec Králové duben 2015

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 27. dubna 2015 . . . . . .

Radek Jirsa

Abstrakt

Cílem této práce je vytvoření webové hry za pomocí Yii Frameworku. Zvolený framework je založen na MVC architektuře.

V práci je popsán postup, jakým byl finální framework zvolen. To vše je podloženo argumenty, proč tomu tak bylo. Ve druhé části je vybraný framework představen a jsou ukázány jeho hlavní přednosti. Ve třetí části najdeme návrh aplikace, její implementace a optimalizace.

Výsledkem práce pak je webová aplikace, která bude volně dostupná a zároveň odlišná od jiných her na trhu.

Klíčová slova

Yii Framework, MVC architektura, webová hra, vývoj webových aplikací

Abstract

The goal of this bachelor thesis is creation of browser game using Yii Framework. Selected framework is based on MVC architecture.

In the thesis describes the procedure, how was the final framework elected. Everything is supported by arguments as to why this was so. In the second part is selected framework introduced and they are shown its main advantages. In the third part we find application design, its implementation and optimization.

The result of the bachelor thesis is a web application that will be freely available and also will be different from other games on the market.

Keywords

Yii Framework, MVC architecture, web game, development of web application

Obsah

1. Úvod

Webové technologie i internetový obsah od svého vzniku dostály již mnoha změn. V dřívějších dobách zde byly jen statické stránky postavené pomocí čistého HTML jazyka spolu se jazykem CSS pro stylování obsahu. Dnes na internetu najdeme převážně již dynamické weby spravované CMS systémy jako je například Wordpress, Joomla! a další. Existují však weby, pro které je použití CMS nemyslitelné. Úprava CMS pro daný účel webu by zabrala mnoho zdrojů, ať už finančních, nebo značné lidské úsilí. Proto si jednotlivé společnosti vyvíjely vlastní systémy, které by byly pokud možno univerzální, a jejich kód se dal využít ve více projektech. Psaní jednoho a toho samého kódu pořád dokola je značně neefektivní a poskytuje velké riziko výskytu nějaké chyby, ať už bezpečnostní, nebo implementační. Velká část kódu pro většinu webových aplikací opakuje. Například přihlašování, zabezpečení, práce s databázovými modely, práce s menu atd. Proto se zrodila myšlenka zrodu systému, který by tyto neduhy eliminoval.

Díky těmto podnětům došlo k postupnému vzniku několika prvních frameworkům. Co je to framework? Jedná se o samostatnou aplikaci, která poskytuje základ každého většího projektu – webové aplikace. Využitím takovéhoto frameworku se značně eliminuje riziko chyb a doba, která je potřeba na vývoj aplikace. Jednotlivé části aplikací se pak dají využít i v jiných projektech, což opět zefektivňuje tvorbu webové aplikace.

1.1. Cíle práce

Frameworků je dnes široká řada a není lehké si mezi nimi zvolit ten pravý. Cílem této práce je vytvoření webové hry, která bude postavena právě za pomocí frameworku. Požadavky na framework byly takové, že musí splňovat MVC architekturu.

1.2. Struktura práce

V první části práce je ukázán postup jakým byl finální framework zvolen a je zde popsáno několik pádných argumentů, proč tomu tak bylo. V další části je zvolený framework představen a jsou ukázány jeho hlavní přednosti. Ve třetí části je ukázán návrh aplikace, její implementace a optimalizace. Závěr je věnován shrnutí výsledků práce a zhodnocení stanovených kritérií.

1.3. Předpokládaný výsledek práce

Hlavním výsledkem této práce je webová aplikace, která bude volně přístupná z internetu. Aplikace bude webová hra, která bude odlišná od jiných her na trhu. Hra by měla odstranit neduhy již dostupných her tak, aby byla uživatelsky přívětivá.

2. Historie MVC

Architektura MVC byla vynalezena v 70. letech 20. století norským profesorem Trygvem Reenskaugem ve výzkumném a vývojovém centru XEROX PARC v Kalifornii.

První významný dokument obsahující informace o MVC byla příručka „*Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk -80*“ zveřejněná na přelomu srpna a září roku 1988. Smalltalk byl tehdy jedním z prvních čistě objektových programovacích jazyků. (3)

Dnes se již moc nepoužívá, ale stále je populární zejména v oblasti výuky a své příznivce si najde pro psaní podnikových aplikací. (2)

MVC je klíčový pohled v celé oblasti grafických uživatelských rozhraní. Byl jednou z prvních architektur, která popsala a implementovala softwarové konstrukce z hlediska jejich odpovědnosti. MVC byl také první návrhový vzor, který byl významný, protože definoval komponenty namísto použití konkrétních implementací – každý controller měl jistý soubor zpráv, na které musel reagovat stejně jako každé View, ale jinak zde nebylo žádné omezení pro to, co tyto komponenty dělaly či jak to dělaly. View byly závislé na svých modelech a reagovaly na jednotlivé změny zprávami buď úplným překreslením Modelu, nebo jen inteligentním selektivním překreslením obsahu. V originálním znění View zobrazoval reprezentaci všech nebo jen část objektu, který mohl být složený a který obvykle měl Model definovaný jako „superclass“. Vývojáři aplikací se tak ocitli v situaci, kdy psali velké množství kódu a Views. Ty se nelišily ani tak v samotné implementaci jako spíše v drobnostech jako barva tlačítka. (3)

V této době proto vznikly zásuvná Views. Jednalo se o obecný kód a seznam Views, jejichž údaje byly specifikovány v metodách s mnoho parametry využitých k jejich vytvoření. Zásuvná Views značně snížila počet individuálních Views a Controllerů potřebných k jejich obsluze v klasické aplikaci. Zejména proto, že většina rozhraní byla složena jen z textu a tabulek. (3)

V roce 1991 došlo k předělání systémových nástrojů a podkladových tříd MVC. Zásuvná Views zmizela a místo nich se objevili GUI komponenty, tak jak je známe, například CheckBox, field, button atd. Každá komponenta byla implementována jako malé rozhraní s vlastním Controllerem. Třídy obsahující jen text či seznamy se staly o mnoho jednoduššími a přehlednějšími, než v předchozím použití zásuvných Views. Nápad použití flexibilních generických komponent k budování rozhraní byl rozšířen o nové druhy komponent, které se v budoucnu staly standardem pro sadu nástrojů tvorby GUI. (3)

Oddělená struktura zobrazovací, řídící a datové části vznikla zejména kvůli tehdejšímu hardwaru. Jednalo se ještě o sálové počítače, kdy každá jednotka zpracovávala něco jiného. Jedna se starala o grafický výstup, jedna kontrolovala uživatelské akce a další se starala o přístup k datům. V dnešní době web pracuje na podobném principu. Vstup a výstup je oddělený – vstupem jsou URL a výstupem je HTML. Proto se dnešní návrh MVC o moc neliší od toho původního MVC ve Smalltalku.

3. MVC Architektura

MVC (Model-View-Controller) architektura nebo také Model 2 je architektonický vzor pro tvorbu aplikací. Jedná se o softwarovou architekturu oddělující její hlavní části na tři navzájem nezávislé komponenty, kterými jsou datový model aplikace (dále jen Model), řídící rozhraní aplikace (dále jen Controller) a uživatelské rozhraní (dále jen

View). Tím se odlišuje od Modelu 1, který odděluje pouze datový model a uživatelské rozhraní s řídící logikou. (4)

Z tohoto návrhu vyplývá, že je poměrně jednoduché jednotlivé komponenty editovat, protože úprava jedné komponenty má minimální vliv na zbylé dvě. Tím se stává aplikace, tvořená pomocí této architektury, jednodušší a zároveň je možné jednotlivé komponenty znovu použít na jiných místech aplikace nebo za cenu minimálních úprav i v jiném podobném projektu. Pokud programátor striktně dodrží veškerá pravidla této architektury, pak bude výsledkem jeho práce přehledný kód o třech vrstvách, kde každá bude dělat práci, pro kterou byla primárně navržena. Aplikace bude rychlá, výkonná a zároveň by se nemělo stát, že aplikace bude ohrožena nějakou bezpečnostní chybou, protože veškerá aplikační logika je důsledně oddělená od klienta.

- **Model**

  Reprezentuje data a business logiku celé aplikace. Ve skutečnosti se jedná o její datový základ. Model tedy umožňuje přístup k datům a tvoří výkonovou část aplikace. Většinou bývá nejsložitější komponentou samotné aplikace. Model sám o sobě data nijak neformátuje, pouze je udržuje a spravuje. Model disponuje pevným rozhraním, díky kterému Controller komunikuje s Modelem. (5)

  Pokud Controller vyžaduje nějaká data například z databáze, pak pomocí právě tohoto rozhraní požádá Model. Ten jen zprostředkuje přenos z databáze a předá data Controlleru, který si o ně vyžádal. Model jako takový o existenci zbytku aplikace neví, pouze poskytuje služby, které mohou data získat či změnit. Poměrně velká množina akcí vyvolaná uživatelem (například: přihlášení, jakákoliv změna dat v databázi) používá ke své činnosti právě Model.

  Jednou ze zásadních chyb začínajícího programátora je ta, že chápe Model pouze jako datový model aplikace. V MVC architektuře se jedná o mnohem více než o datový model, jsou zde například business pravidla aplikace. Ty ověřují například to, že „Akce2“ nemůže proběhnout dříve než je „Akce1“ dokončena (Například: nelze využít plných služeb aplikace, pokud nebyla dokončena registrace). Dále by zde měla být umístěna i validační pravidla. Jedná se například o kontrolu vyplnění povinných položek. Nebo když textový vstup pro počet položek může obsahovat pouze číslo. Je poměrně časté, že se tyto validace objevují ve View, například ASP.NET k tomu přímo i svádí, ale to není správně. Validaci totiž potřebujeme na více místech, a pokud dojde k její implementaci do View, pak dojde k duplikaci kódu a aplikace se tak stává neefektivní. (7)

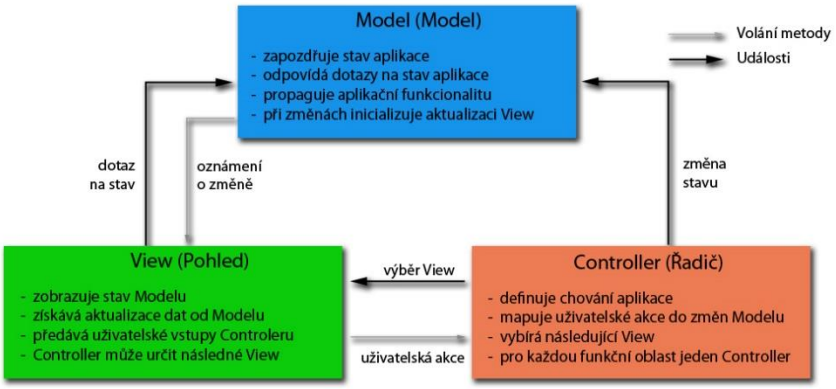
- **View**

  View je vrstva aplikace, která se stará o vykreslení grafického rozhraní pro uživatele. Výstup této vrstvy je ve většině případů čisté HTML. Má tedy na starosti finální zobrazení dat uživateli, které získá za pomocí Modelu. Zároveň se stará i o předání nových dat modelu od uživatele, třeba za pomocí formuláře. V praxi můžeme mít pro jeden Model (Controller) více Views. To je vhodné zejména pokud potřebujeme vynutit jiné uživatelské prostředí pro mobilní telefony, jiné pro

tablety a jiné pro klasický desktop. Hlavní výhodou zobrazovací vrstvy je to, že HTML nemusí být jediný výstup, který dokáže vykreslit. View dokáže zobrazit data ve formátech jako je XML či JSON. (7)

- **Controller**

Stojí mezi View a Modelem aplikace. V prostředí webové aplikace se nejčastěji skládá ze dvou hlavních částí. První je tzv. Front Controller. Ten zachytává uživatelské akce a všechny HTTP požadavky, které zpracuje a předá konkrétním Controllerům, což je ona druhá část. Ty v závislosti na druhu akce vyvolávají příslušné události. Controller tedy reaguje i na změny ve View (kliknutí na odkaz, odeslání formuláře apod.). Například při kliknutí na odkaz ve webové aplikaci zajistí načtení příslušného View, které se nachází za odkazem. Nebo v případě odeslání formuláře, předá dat z formulářových prvků Modelu, který data uloží do databáze. (7)



Obrázek 1- MVC Architektura (Zdroj: Vlastní)

Na obrázku je jasně vidět, jak spolu všechny 3 vrstvy úzce spolupracují. Dále je znázorněno volání metod jednotlivých vrstev či co se děje při uživatelské akci. Jak je z obrázku patrné, jedná se o uzavřený ekosystém, kde jsou všechny vrstvy na sobě závislé, nicméně pokud dojde k záměně kódu jedné vrstvy, neovlivní chování ostatních vrstev.

3.1. Proč využít MVC Framework

Framework je sám o sobě funkční aplikací. Obsahuje širokou škálu předdefinovaných funkcí, které je možno bez problémů využít. Tyto funkce a především samotné kódy frameworku jsou odzkoušeny širokou komunitou uživatelů (programátorů), a proto jsou velice dobře optimalizovány jak z hlediska výkonu tak i bezpečnosti, což nemusí platit o našem vlastním kódu, který by potencionálně mohl být v naší vlastní aplikaci bez použití frameworku. Oddělení jednotlivých vrstev architektury zajišťuje snadnou editaci kódu s minimálním vlivem na okolní vrstvy. Zároveň zamezuje duplicitu kódu, například duplicitu validačních funkcí pro formuláře. Tyto výhody ovšem platí, pokud chce vyvíjet nějaký větší projekt. Osobní webová prezentace o pár stránkách bude vždy rychlejší, pokud použijeme čisté HTML spolu s PHP. Proto je dobré si rozmyslet, co budeme vyžadovat od naší budoucí aplikace a k tomu použít vhodný nástroj.

Existuje mnoho frameworků, některé se specializují přímo na tvorbu malých projektů, jedná se o takzvané „micro-frameworky“. Jejich kód není složitý, ale na druhou stranu neposkytují takové možnosti jako třeba dva níže zmíněné. Vše je tedy na zvážení programátora a vhodného výběru frameworku, pomocí kterého se bude aplikace vyvíjet.

3.2. Proč nevyužít MVC Framework

Dříve než začneme framework využívat, je nutné si přečíst jeho technický manuál, zjistit jaké možnosti a funkce daný framework nabízí. Framework jako takový je silný nástroj pro tvorbu aplikací, ale může být problém se s ním naučit pracovat. S tímto problémem úzce souvisí i samotný výběr frameworku. I když většina ve výsledku dělá to samé, jejich principy a způsoby programování v nich se mohou více či méně lišit. Každý projekt vytvořený pomocí frameworku musí obsahovat všechny knihovny daného frameworku, což může být problém u některých free-hostingů, které poskytují malý prostor pro data. I jeho složitost a obsáhlost může být na obtíž. Například samotný Yii Framework obsahuje necelých 2000 souborů a to může být také problém u free-hostingů, které mají omezení na počet souborů na FTP. U některých frameworků může nastat problém, co se rychlosti aplikace týče. Samotný framework využívá ke své práci mnoho pomocných objektů a komponent, což při špatné optimalizaci může mít za následek pomalé načítání aplikace.

4. Volba Frameworku

Webová aplikace bude postavena na technologii PHP frameworku. Proč ale právě tato technologie? Tuto otázku si jistě položí mnozí z Vás. Většina aplikací tohoto typu funguje díky jazyku JAVA, který je také multiplatformní, a mnozí mohou říci, že i lépe použitelný pro tento účel. Vzhledem k tomu, že se webovým aplikacím věnuji téměř už od základní školy, tak jsem si zvolil jazyk PHP, ve kterém mám již nějaké zkušenosti. Ale psát aplikaci tohoto typu v čistém PHP je značně neefektivní a tato aplikace by asi nikdy nemohla vzniknout za tak krátkou dobu.

Před pár lety mi bylo doporučeno, ať využiji PHP framework. Díky němu je prý práce na webových aplikacích značně zjednodušena, kód se stane efektivnější, bezpečnější a znovupoužitelnější. Toto se ovšem ukáže jako pravda až v momentě, kdy se s daným frameworkem naučíte opravdu pracovat. Pracovat ve skutečnosti znamená porozumět samotnému frameworku, jak je napsán, jak funguje a co a jak mohu použít. Proto by měla nejprve nastat fáze výběru vhodného frameworku.

Tato fáze je jedna z nejdůležitějších na celém vývoji. Existuje poměrně velká řada velkých i malých frameworků a každý má svá pro a proti. Z tradičních starších frameworků vychází nové, do kterých se implementují nové funkce a moderní mechanismy dané doby ihned při vývoji. Zatímco do těch starých se tyto funkce dostávají postupem času v podobě updatů. Nevýhody mladých frameworků jsou zejména ty, že nedisponují takovou komunitou uživatelů, a proto je těžší dostat se k výsledku při řešení nějakého zásadního problému.

Hlavním cílem tohoto srovnání bylo zjištění, který z těchto dvou kandidátů je lepší a který bude vhodnější pro výsledný projekt vytvoření webové hry.

4.1. Vhodní kandidáti

Do mnou vybraného užšího výběru se nakonec dostali dva kandidáti. Jedním byl poměrně mladý Yii Framework, pocházející původně od čínských vývojářů. Popularita tohoto frameworku vzrostla poměrně rapidním tempem a dnes je jedním z nejpoužívanějších PHP frameworků. Jako druhý byl vybrán Nette. Jedná se o český projekt s poměrně silnou základnou příznivců a velkou komunitou po celém světě.

4.1.1.Yii

Vznik tohoto projektu odstartoval v roce 2008 pokus opravit zásadní chyby a nedostatky v tehdejším frameworku PRADO. Jako například neoptimalizované načítání složitých stránek. PRADO byl těžkopádný, poměrně těžký na naučení, pochopení a mnoho ovládacích prvků šlo jen těžko nebo vůbec přizpůsobit potřebám programátora. Po deseti měsících soukromého vývoje byla v červnu roku 2008 vydána první alfa verze Yii. Růstu publicity dosáhl až v roce 2010, kdy vyšla stabilní verze 1.1, která je oficiálně podporována až doteď. (8)

Na konci roku 2014 oficiálně vyšla dlouho očekávaná verze 2.0, která přináší výrazně zlepšenou bezpečnost a celkově optimalizovanější chod aplikace. Zásadním rozdílem těchto dvou majoritních verzí je verze PHP, kterou vyžadují. Druhá verze Yii vyžaduje pro správné fungování PHP ve verzi 5.4.0 či novější, zatímco první verzi Yii stačilo PHP verze 5.1.0. To je také jedním ze zásadních problémů upgradu Yii z verze 1.1 na verzi 2.0. Tyto verze bohužel nejsou kompatibilní a většina funkcí již byla změněna a došlo také ke změně adresářové struktury základní aplikace.

4.1.2.Nette

Jedná se o český opensource projekt, který byl oficiálně uvolněn v roce 2008. Původním autorem je, v České republice poměrně známý, David Grudl. Teď se o vývoj stará organizace Nette Foundation. (9)

Dnes se již projekt nachází ve verzi 2.2.7, která vyžaduje PHP verze 5.3.1. Hlavní předností projektu je velký důraz, který je kladen na eliminaci bezpečnostních hrozeb jako je Cross-site scripting (XSS), Cross-site requers forgery (CSRF), session hijacking, session fixation a jiné. Autoři projektu slibují strmou křivku učení, která zaručuje rychlé naučení práce s Frameworkem. Další důležitým faktorem Nette je implementace moderních technologií jako jsou Asynchronous JavaScript and XML (AJAX) a Asynchronous JavaScript and JSON (AJAJ), Dependency injection, Search engine optimization (SEO), Cool-URL a další. (10)

Hlavní nevýhoda Nette je jeho nekompatibilita se staršími verzemi.

4.2. Bezpečnost frameworku

Hlavní parametr, kterému by měli věnovat pozornost všichni, kdo se rozhodují pro nějaký nový systém, je bezpečnost. Vaše aplikace je bezpečná natolik jako její nejslabší

článek. Můžete udělat nějakou bezpečnostní chybu vy, což je věc, která se bohužel stává, ale nechá se poměrně jednoduše opravit. Ale pokud budete svou aplikace stavět na bezpečnostně „děravém“ frameworku, pak se můžete snažit sebevíc, ale Vaše aplikace již bezpečná nebude. Zaměřil jsem se především na věci typu XSS, CSRF, Cookies/Session hijacking.

4.2.1.Cross-site scripting (XSS)

XSS je útok, při kterém je stránka napadena útočníkem většinou prostřednictvím nezabezpečených vstupů a JavaScriptového kódu. Útočník je schopen díky této chybě podstrčit vlastní skript, který se provede na napadené stránce a uživateli tak může třeba podvrhnout jiný obsah, získávat citlivá osobní data, znefunkčnit stránku či obcházet její bezpečnostní prvky. (11)

Většinou se rozděluje na dva druhy podle metody útoku:

- První je útok, který bývá označen jako non-persistent. Neboli je to útok, kterým napadneme danou webovou stránku pomocí změny v parametru URL adresy a změna tak není trvalá.
    - Víme, že aplikace vypisuje obsah parametru v URL
      http://domena.cz/?hledej=slovo
      => stránka zobrazí, že hledáme „slovo“
    - Upravíme proto parametr
      http://domena.cz/?hledej=<script>alert('Úspěšný XSS útok.');</script>
      => Za předpokladu, že nám stránka vypíše JavaScriptové hlášení, pak není vstup ošetřen a stránka je jednoduše napadnutelná
    - Pokud je uživatel pozorný, může si povšimnout jasně podezřelé URL a na stránku přes tento odkaz nevstoupit. Ovšem i tento problém jde maskovat přes různé „URL_Shortening“ weby, které takto nápadný parametr zamaskují do zkrácené URL, ze které nevyčtete, kam Vás směřuje.
- Druhý typ útoku je označován jako persistent či stored, protože trvale modifikuje napadnutou stránku.
    - Jedná se o útok, kdy programátor špatně ošetřil vstup například textového pole ukládaného do databáze. Například při vkládání komentáře k článku je připojen podobný script, jako byl použit výše v útoku prvního typu.
      „Ahoj, máš krásné <script>alert('Toto je úspěšný XSS útok.')</script>stránky!“

Pokud se skript provede, pak má útočník vyhráno a může libovolně modifikovat obsah Vašich vlastních stránek

4.2.2.Cross-site request forgery CSRF

CSRF je jedna z metod útoku, kdy se využívá především sociálního inženýrství a znalosti aplikace, kterou útočník chce napadnout. Útok pak probíhá tak, že útočník zná patřičné URL (například pro smazání článku) a jeho identifikátor (nejčastěji nějaké id). Najde nějakého přihlášeného uživatele, který má práva pro mazání článků

a přesvědčí ho, aby klikl na nějakou inkriminovanou stránku. Ta stránka na první pohled nijak nesouvisí s aplikací, ve které je přihlášen, ale na pozadí načítání stránky se odešle požadavek na smazání článku. Pokud daná aplikace nemá ochranu proti CSRF, požadavek se provede pod přihlášeným uživatelem a uživatel o tom ve skutečnosti vůbec neví. Tento útok může být proveden jak pomocí GET, tak pomocí POST HTTP požadavku. Útočník tak, prostřednictvím uživatele s příslušným oprávněním, může napáchat poměrně velké škody v aplikaci. (12)

4.2.3.Cookies/Session hijacking

Session hijacking, také známé jako Cookies hijacking je útok, při kterém útočník buď odcizí privilegovanému uživateli tzv. „SESSION" známé také jako „session key" nebo podstrčí svoje „SESSION ID" a použije ho pro přístup k informacím či službám v dané aplikaci, aniž by znal heslo uživatele. Tento útok je zejména kombinován za pomocí síťového útoku „Man_in_the_Middle", kdy je útočník mezi uživatelem a serverem, na který uživatel odesílá data. Útočník pak za pomocí sniffovacího programu zachytává veškerá data a je tak schopen ukrást uživatelovo „SESSION ID". (45)

Tyto tři zásadní a nebezpečné útoky jak Yii tak Nette poměrně efektivním způsobem eliminuje a není tedy nutné je nějak ve velké míře řešit. Pokud neudělá programátor nějakou zásadní elementární chybu, problémy od těchto útoků aplikaci nehrozí.

4.3. Jak začít s vytvářením aplikace ve frameworku

Pro uživatele, který s frameworkem začíná, je důležitá informace, zdali má daný framework nějaký přehledný tutoriál jak framework nainstalovat a začít s ním pracovat. Dalším faktorem je bezpochyby přehledná dokumentace, která je při vývoji aplikace nepostradatelná.

4.3.1.Nette

Na oficiálních stránkách má Nette vystavený návod pro prvotní instalaci aplikace. Nechybí ani přehledné informace k vytvoření Vaší první aplikace. Co se dokumentace týče, obsahuje pouze omezené větve, o kterých se dozvíte podstatné informace, ale celkovou dokumentaci budete hledat marně. Nette ovšem slibuje strmou křivku učení, takže příručka poskytuje nejspíše základ k pochopení fungování Nette a zbytek se dá odvodit od těchto základů. Případně je zde komunitní fórum, na kterém je většina zásadních problému přehledně vysvětlena.

4.3.2.Yii

Yii přistupuje k tomuto tématu jiným způsobem. Prvotní instalací vás provede celkem 5 videí, ve kterých se dozvíte veškeré podstatné informace, jak pracovat s frameworkem a jak napsat svojí první aplikaci. Pro verzi 2.0 přibyl oddělený „2.0 Guide", který je víceméně stejný jako tutoriál u Nette, nicméně disponuje kompletní a kvalitní dokumentací a obsahově je na vyšší úrovni.

Z mého pohledu se mi více líbil přístup Yii, který Vás naučí napsat své první aplikace opravdu během několika minut. I zmíněný „2.0 Guide“ mi přišel přehlednější a vyspělejší než u Nette. Na druhou stranu příručka Nette je kompletně přeložená do češtiny, což vyhovuje „neangličtinářům“. Ovšem první zklamání přijde při navštívení komunitního fóra, na kterém se, kvůli rozrůstání mimo české hranice, bez znalosti anglického jazyka stejně neobejdete.

4.4. Možnosti rozšíření (moduly/pluginy)

Možnost rozšíření základní aplikace pomocí již hotových řešení tzv. pluginů či modulů je rozhodně velké plus při vytváření aplikací. Proč psát něco, co je již napsáno a je volně k dispozici? Případně když mi celé řešení nevyhovuje, mohu jej jednoduše upravit podle svých představ, což vyžádá méně času než psát dané rozšíření od samotného začátku. Většina autorů poskytuje i poměrně obstojnou dokumentaci ke svým modulům, takže není až takový problém je upravit, případně začít hned využívat. Zaměřil jsem se tedy na fakt, zdali oba frameworky tyto rozšíření podporují a zdali již jsou již nějaké dostupné.

Oba dva nabízejí poměrně slušnou škálu doplňků. V nabídce jsou vizuální doplňky, například doplněk pro tvorbu menu, propojení se sociálními sítěmi, modul ochrany proti odeslání formuláře robotem (Re-Captcha). Také zde najdeme třeba doplňky pro práci s lokalizacemi, tématy a spoustu dalších. Když srovnám nabídku, co poskytuje repositář s doplňky u Nette a u Yii, pak Yii jasně vyhrává. Je vidět, že za ním stojí mnohonásobně větší komunita, která doplňky přidává, včas reaguje na nové verze a aktualizuje je.

4.5. Technologie

Oba výše vybrané frameworky mají základ v jazyku PHP. Konkrétně se jedná o objektový návrh postavený na MVC architektuře. Pokud chceme být důslední, pak je dobré zmínit, že Yii je MVC Framework zatímco Nette je MVP Framework. Oba dva také úzce využívají AJAX, který je potřeba při dynamických změnách stánek bez nutnosti kompletního načtení celé stránky.

4.5.1. MVC nebo MVP?

Většinou tu byla řeč o MVC architektuře, ale co znamená MVC Framework? MVC Framework je aplikace postavená tak, aby ctila zásady MVC architektury. Nette je ale MVP Framework, jaký je v tom tedy rozdíl?

MVP neboli Model-View-Presenter vychází z výše uvedené MVC architektury. Dalo by se říct, že MVP je konkrétní druh MVC. Také má tři základní vrstvy: Model, View a Presenter. Zásadní změna se odehrává v zobrazovací vrstvě. Ve View jsou zde oproti MVC odchytávány uživatelské akce. V důsledku druhu akce je pak volána konkrétní metoda v Presenteru. Ten pak vyvolá změnu View. View má tedy přímou vazbu na Presenter a plně kontroluje uživatelský vstup a výstup. To má ve výsledku přinést snadnější udržování prezenční vrstvy.

Podle odpovědnosti View se dá ještě MVP Framework rozdělit na dva vzory:

- **Supervising Controller**

  Jedná se o vzor, který přesně odpovídá popisu výše a jak vypadala původní implementace tzv. Dolphin Smalltalk MVP vzoru. Tedy vrstvě View je předána zodpovědnost za zobrazení dat z modelu. Ačkoliv se ve skutečnosti jedná o Presenter, autor návrhu Martin Fowler zřejmě nebyl o správném názvu zcela přesvědčen a tak zvolil to, co mu přišlo nejlepší. (7)

- **Passive View**

  Passive View je oproti tomu úplný opak. Smyslem tohoto vzoru je udělat View pokud možno co „nejhloupější". Není zde ani jednoduché zobrazování dat z modelu. Veškeré zobrazování je zde zprostředkováno pomocí Presenteru, který dostal oproti výše uvedenému vzoru větší pravomoci. (7)

4.6. Výkon

Kvalita frameworku jde do jisté míry také poznat na jeho výkonu. Výkon se odráží zejména ve velikosti konkrétního frameworku a množství podporovaných funkcí. Oba dva vybrané jsou si velice podobné, a proto je dobré vědět, jestli nějaký netrpí nějakým výkonnostním nedostatkem.

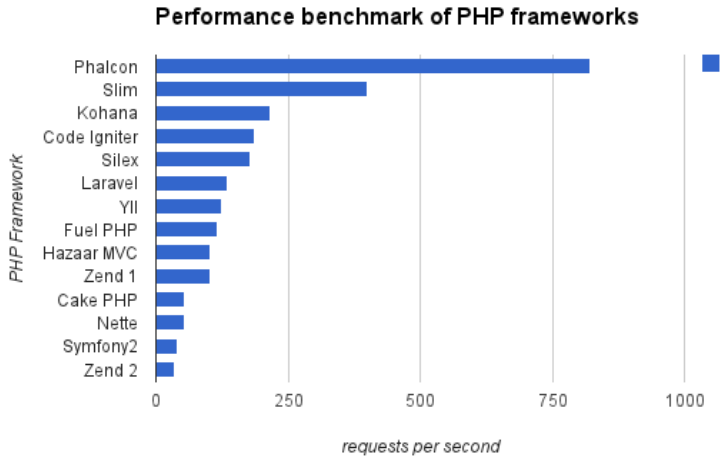
Pro porovnání byl zvolen jeden zahraniční a jeden tuzemský uživatelský benchmark. Vzhledem k tomu, že oba se na výsledku víceméně shodly, pak bych je oba považoval za relevantní.

Benchmark – jedná se o výkonnostní test, kdy jsou simulovány uživatelské požadavky na aplikaci. Tyto požadavky jsou totožné, jakoby seděl uživatel a prováděl je sám, ovšem v benchmarku se jich za tu samou dobu provede několikanásobně více. Což dovoluje simulovat nápor uživatelů.

4.6.1. Systemsarchitect.net

Testovány byly základní projekty, tzv. demo projekty, které najdeme u každého frameworku. Ty by měly poskytnout základní obraz o frameworku a jeho funkčnosti. Jednotlivé projekty byly umístěny na pronajatém virtuálním serveru hostovaném společností Amazon. To z důvodu odrušení chyb nesprávným nastavením systému a zkreslení výsledku při použití běžného hardwaru. K testování jednotlivých aplikací byl využit ApacheBench, což je nástroj simulující uživatelské požadavky a přístupy na web. Nastavení benchmarku bylo následovné: *N = 500, C = 20.* To znamená, že bylo vysláno celkem 500 HTTP požadavků s možností kontinuálního zpracování až 20 požadavků.

Test dopadl s jasným výsledkem pro Yii. Yii dokázal zpracovávat 123,5 požadavků za vteřinu zatímco Nette odbavil „pouze" 53,48 požadavků. Pro srovnání je přiložen graf se srovnáním ostatních světově nejpopulárnějších frameworků.



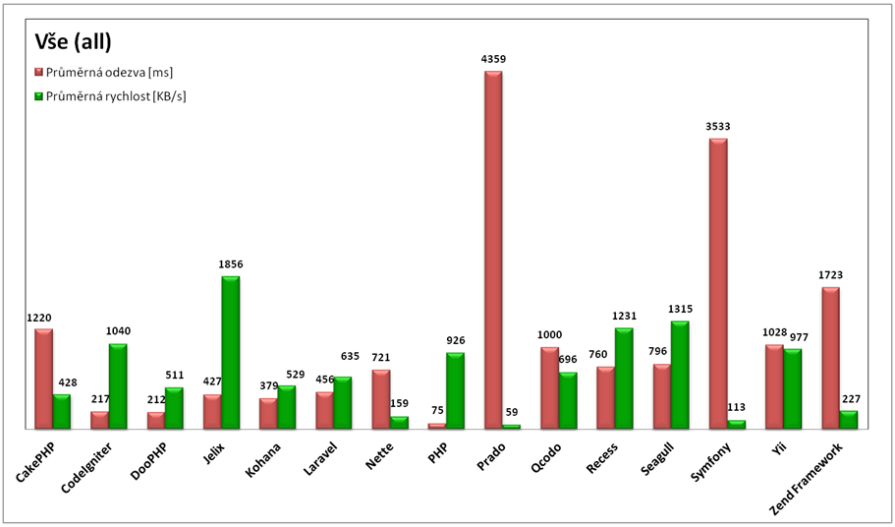
Obrázek 2- Test výkonosti frameworků (Zdroj: http://systemsarchitect.net/)

Phalcon spolu se Slim frameworkem jsou jednoduché, silně ochuzené frameworky bez možností implementace doplňků a nasazení na složitější projekty. Jsou vhodné pro malé projekty typu osobní webová prezentace. Fungují na bázi relativně čistého OOP PHP, takže jak je vidět, jejich hlavní přínos je zejména v rychlosti.

4.6.2.Zdrojak.cz

Český test populárních frameworků byl proveden na 2 strojích, kde jeden sloužil jako serverový a jeden jako klientský, ze kterého byly vysílány požadavky na server a framework. Test zahrnoval využití 4 nejzákladnějších databázových funkcí (SELECT, INSERT, UPDATE, DELETE), které se periodicky opakovaly. Celkem test zahrnoval 3000 požadavků od každé funkce, ve výsledu bylo provedeno tedy 12000 žádostí na server. Ty byly sledovány a z nich byl vypočten průměr požadavků, které zvládl server s konkrétním frameworkovým projektem odbavit. Také byla sledována průměrná odezva, za kterou byly požadavky provedeny.

Výsledek byl poměrně překvapivý, kdy Yii sice zvítězilo s lepší propustností, ale zároveň mělo horší průměrnou odezvu. Z čehož nám vyplývá jasný výsledek, že Nette je sice rychlý, ale pokud bude zpracovávat více požadavků, tak ztrácí poměrně dost na svém výkonu. Vše je poměrně pěkně vidět na následujícím obrázku.



Obrázek 3 – Počet požadavků odbavených za průměrnou dobu
(Zdroj: zdrojak.cz/clanky/orm-test-php-frameworku-php-zaver)

4.7. Licence

Pokud se rozhodnete vyvíjet webový software ať už komerční či nekomerční, pak je důležité zkontrolovat, jak je framework, na kterém Vaše aplikace bude postavena, licencován. Většinou se zde nesetkáme s nějakými problémy. Většina frameworků je licencována pomocí New BSD License.

„Jedná se o licenci pro svobodný software a je zároveň jednou z nejsvobodnějších licencí vůbec. Umožňuje volné šíření licencovaného obsahu, přičemž vyžaduje pouze uvedení autora a informace o licenci, spolu s upozorněním na zřeknutí se odpovědnosti za dílo. Zkratka BSD označuje „Berkeley Software Distribution" – obchodní organizaci při University of California, Berkeley, která tuto licenci vyvinula a používala pro práce nad operačním systémem BSD." (44)

BSD licence dovoluje komerční využití, včetně využití v proprietárním softwaru bez zveřejněného zdrojového kódu. Díla založená na dílech licencovaných pod BSD dokonce mohou být zveřejněna pod komerční licencí, pouze musí dodržet podmínky licence, tzn. v programu uvádět informaci o autorech a zřeknutí se odpovědnosti. (44)

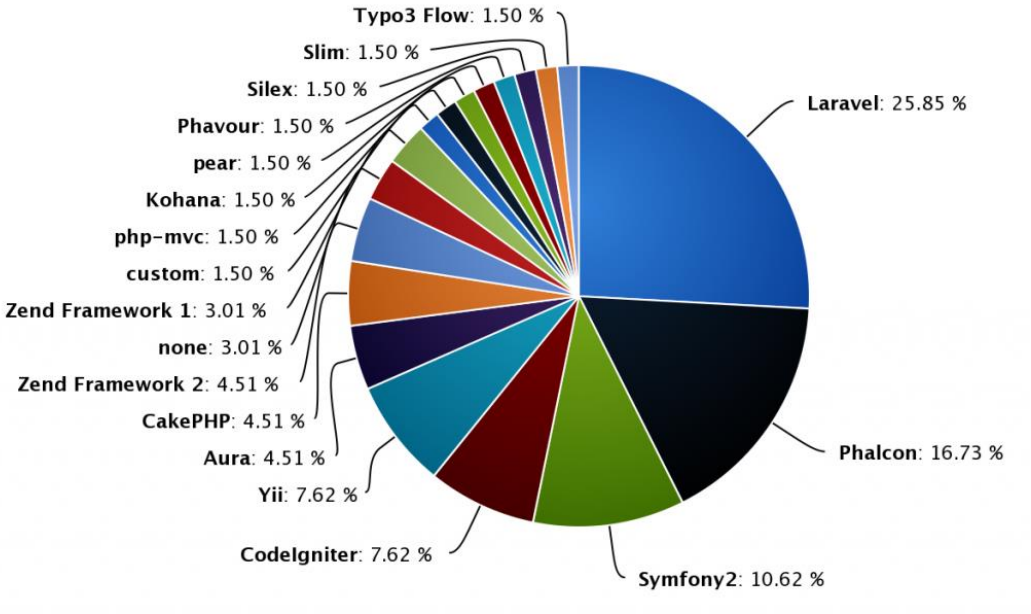
Jak Nette tak i Yii je licencován touto New BSD Licencí. Takže pokud dodržíme podmínky této licence, nemusí se za jejich využití platit či žádat o souhlas autorů. A to platí i pro komerční aplikace.

4.8. Závěrečné rozhodnutí

Jak již bylo řečeno, každému může vyhovovat jiný framework, kvůli jeho přístupu a stylu, jakým se s ním pracuje. Jeden framework se nedá použít jako podklad pro e-shop a zároveň jako základní kód pro své osobní stránky. Respektive možné to je, ale musíte správě volit mezi výkonem a funkcemi. Zde totiž končí vlastnosti frameworku a začínají schopnosti programátora, který aplikaci píše. I se sebelepším frameworkem programátor, který nedokáže optimalizovat své kódy, nenapíše rychlou a stabilní

aplikaci. Proto je důležité zvolit framework tak, aby vyhovoval schopnostem a znalostem programátora, který je dokáže onen stavební materiál (myšleno framework) proměnit svým umem na stabilní a výkonnou aplikaci.

Po několika pokusech a důkladném projití tutoriálů obou frameworků byl nakonec zvolen Yii. Jedná se o můj subjektivní názor. Tento framework se mi svým „stylem“ psaní kódů zalíbil a dnes bych si už práci bez něj nedovedl představit. Tento názor se mnou zřejmě sdílí více lidí, protože v roce 2014 byl Yii jedním z 5nejpoužívanějších frameworků na internetu. Což může doložit i tento graf, který je výsledkem rozsáhlé studie portálu Sitepoint, který se zabývá webovými technologiemi.



Obrázek 4 – Uživatelsky nejoblíbenější frameworky
(Zdroj: http://www.sitepoint.com/best-php-frameworks-2014)

Zde je vidět, že Yii se umístil na 5. místě s podílem necelých 8%.

5. Představení Yii

Jak již bylo řečeno, Yii je poměrně mladý framework vyvíjený od roku 2008. Podle oficiálních stránek se tento framework, vyvíjený původně čínskými vývojáři, vyslovuje jako Yee nebo [Ji:]. Jedná se o akronym pro „Yes it is!“ (Ano to je). To je, jak uvádí jeho autoři, často přesná a nejvýstižnější odpověď na dotazy nových uživatelů Yii. (19)

„Je rychlý? … Je bezpečný? … Je profesionální? … Je ten pravý pro můj další projekt? … Ano, to je!“ (19)

5.1. Přednosti Yii

Hlavní přednosti Yii vyplývají z jeho integrovaných funkcí. Například defaultně implementované ověření vstupů z formulářů. Parametry pro kontrolní funkce se navíc samy nastaví v závislosti na datových typech atributů v databázové tabulce.

Možnost jednoduchého spravování designů webu je s Yii snadnou záležitostí. V Yii je přímo implementován mechanismus pro snadnou správu témat či vzhledů, který umožní rychlou změnu vzhledu aplikace.

Poměrně zajímavou funkcí je možnost lokalizace a internacionalizace. Lokalizace ("L10n") znamená upravení aplikace pro danou zemi, jak na úrovni jazyka, tak musí dojít i ke změně času, formátu data a třeba měny. Internacionalizace neboli zmezinárodnění ("i18n") je přizpůsobení aplikace pro více jazykových vstupů a výstupů. Systém je považován za internacionalizovaný v plném rozsahu, pokud je možné měnit jazyk uživatelského rozhraní za běhu aplikace. (20)

Zejména pro vývoj se hodí funkce přepnutí aplikace do „debuging" módu. Tento mód umožňuje přehledné vypsání a hlavně trasování chyby až na úroveň, kterou si zvolíte. Budete tak vědět, kde se chyba vyskytla poprvé a jaký měla následek na chování aplikace. Tento nástroj je při ladění aplikace nepostradatelný.

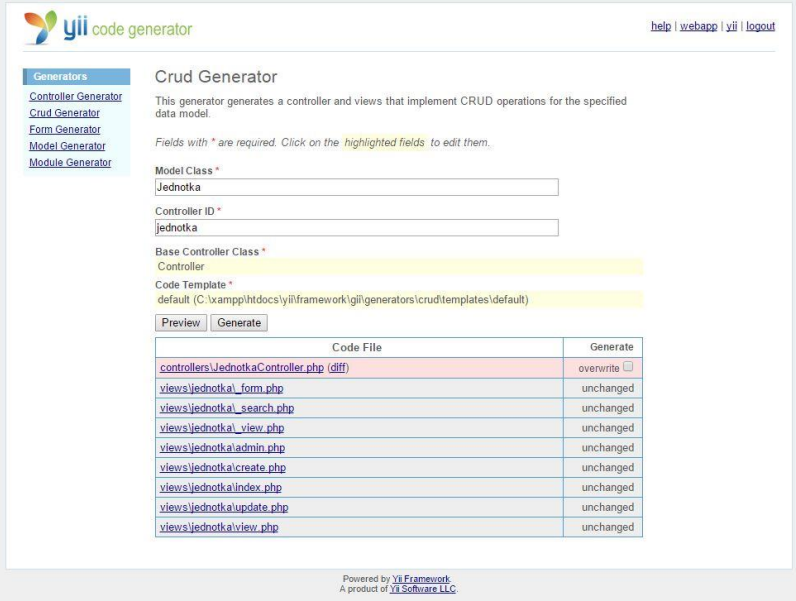
Zajímavostí tohoto frameworku je ta, že jeho kódy jsou kompatibilní například s PEAR či Zend Frameworkem. Takže můžete aplikaci psanou v Zend Frameworku jednoduše migrovat na Yii bez nutnosti větších úprav. (21)

Yii podporuje funkci ukládání dat, stránek a dynamického obsahu do mezipaměti. Pokud se nezmění obsah stránky, například při načítání článku z databáze, je stránka načítána z mezipaměti. Výsledkem je rychlejší načtení stránky bez nutnosti přístupu do databáze. Úložiště mezipaměti se dá jednoduše změnit dle potřeb uživatele, aniž by se musel měnit kód aplikace. (21)

5.2. Gii tool

Nakonec byla nechána funkce, která je opravdu velice silným nástrojem Yii a zaslouží si více pozornosti. Jedná se o nástroj zvaný Gii. Gii je z části automatický nástroj sloužící ke generování kódu aplikace. Automaticky zvládne vygenerovat Model, Controller, formulář a dokonce i celou CRUD strukturu aplikace. Díky tomuto nástroji dojde k vygenerování nejdůležitějších prvků systému během několika kliknutí. Například Model je generován na základě databázové tabulky, se kterou bude později spolupracovat. Díky provázanosti modelu s databázovou tabulkou ještě před jeho vytvořením dojde při generování zároveň i k ošetření vstupů. Z datových typů atributů v tabulce se v modelu vytvoří ověřovací funkce tak, aby nedošlo k uložení jiných dat do databáze, než očekává.

Generování CRUD struktury je obrovské ušetření práce, například když vytváříte aplikace typu blog či osobní stránky. CRUD je ve skutečnosti zkratka pro 4 nejvyužívanější operace pro práci s daty v databázi: Create, Read, Update a Delete. Těmto metodám odpovídají ve standardu SQL metody: INSERT, SELECT, UPDATE a DELETE. Gii za Vás vytvoří celou část aplikace, ve které bude možnost zobrazit, upravit, vložit a mazat příspěvky v databázi, tak jak jsou definovány databázovou tabulkou. Jediné co potřebuje Gii k vygenerování CRUD struktury je již vygenerovaný Model. Gii samozřejmě generuje veškeré soubory ze šablony, proto je jednoduché, pokud Vám nevyhovuje výchozí šablona ji pozměnit nebo si vytvořit vlastní.



Obrázek 5 - Ukázka generování CRUD (Zdroj: Vlastní)

Gii je opravdu silným nástrojem a dokáže ušetřit spoustu času. Nemusíte psát často opakované části kódu znovu a můžete se soustředit na funkčnost a optimalizace celé aplikace. Ačkoliv Gii přináší opravdu velké ušetření času, není schopen aplikaci napsat za Vás. I tak se tento nástroj brzy stane Vaším oblíbeným.

5.3. Řízení požadavků v Yii

Ve většině MVC implementacích mají požadavky na aplikaci následující cyklus:

I. Prohlížeč odešle požadavek na server s MVC aplikací
II. Controller zpracuje žádost
III. Controller naváže spolupráci s Modelem a zažádá o data
IV. Controller vyhledá View a předá mu data
V. View připraví data (nejčastěji jako HTML) a vrátí je prohlížeči k zobrazení

Implementace MVC v Yii se od tohoto modelu nijak neliší. V Yii je nejprve příchozí požadavek od prohlížeče přijat front Controllerem. Ten analyzuje požadavek a rozhodne, kam by měl být v aplikaci poslán k dalšímu zpracování. Ve většině případů front controller identifikuje specifickou metodu v Controlleru, kterému je požadavek směřován. Tato metoda se zaměří na příchozí data a zajistí komunikaci s modelem a provede ostatní potřebnou business logiku aplikace. Nakonec připraví požadovaná data a odešle je View. Ten pak data upraví v souladu s konkrétním rozvržením a designem a vrátí je prohlížeči k zobrazení. (22)

5.4. Struktura projektu Yii

Balíček s Yii, který je možné stáhnout přímo z oficiálních stránek, obsahuje samotné soubory frameworku a 3 demo projekty. Ty je možné přímo nahrát na server

a otestovat tak na nich funkčnost a možnosti Yii. Také je lze využít jako základ pro svůj projekt. Yii ale samozřejmě umí vytvořit úplně nový projekt. Jak se takový projekt vytváří, je popsáno v samotné kapitole v části implementace aplikace. Po vytvoření nové čisté aplikace bude mít následující strukturu.

```
www/
  |-- Yii_aplikace/                  <- Kořenový adresář webu
          |-- assets/                <- Obsahuje zdrojové soubory (Mezipaměť)
          |-- css/                   <- Soubory s kaskádovými styly
          |-- themes/                <- Soubory s komplexními tématy
          |-- protected/             <- Adresář webové aplikace
          |       |-- components/    <- Komponenty potřebné k běhu aplikace
          |       |-- config/        <- Konfigurační soubory aplikace
          |       |-- controllers/   <- Jednotlivé Controllery
          |       |-- models/        <- Jednotlivé Modely
          |       |-- runtime/       <- Popůrné soubory pro běh aplikace
          |       |-- views/         <- Jednotlivá Views
          |
          |-- yii/                   <- Složka se soubory frameworku
          |-- index.php              <- Spouštěcí soubor
```

Obrázek 6 - Ukázka adresářové struktury nové aplikace (Zdroj: Vlastní)

Nejdůležitější je adresář protected, který obsahuje veškeré uživatelské kódy pro Views, Controllery a Modely. Dále zde najdeme konfigurační soubor, ve kterém se nastavuje routování URL, přístup do databáze, automatické načítání komponent, konfigurace Gii a mnoho dalších nastavení. Tato složka samozřejmě není přístupná zvenčí, protože obsahuje citlivá data. Přístup k této složce je zabezpečen pomocí souboru „.htaccess“.

Soubor „index.php“ slouží jako hlavní soubor, který spouští samotný framework. Jeho kód je jednoduchý. Definuje pouze cestu k frameworku a cestu ke konfiguračnímu souboru. Jediným úkolem je pak jen volání metody pro start Yii.

```php
// cesta k souborům frameworku a konfiguračnímu souboru
$yii=dirname(__FILE__).'/yii/framework/yii.php';
$config=dirname(__FILE__).'/protected/config/main.php';

require_once($yii);
Yii::createWebApplication($config)->run();
```

Obrázek 7 - Ukázka souboru index.php (Zdroj: Vlastní)

5.5. Debugování kódu

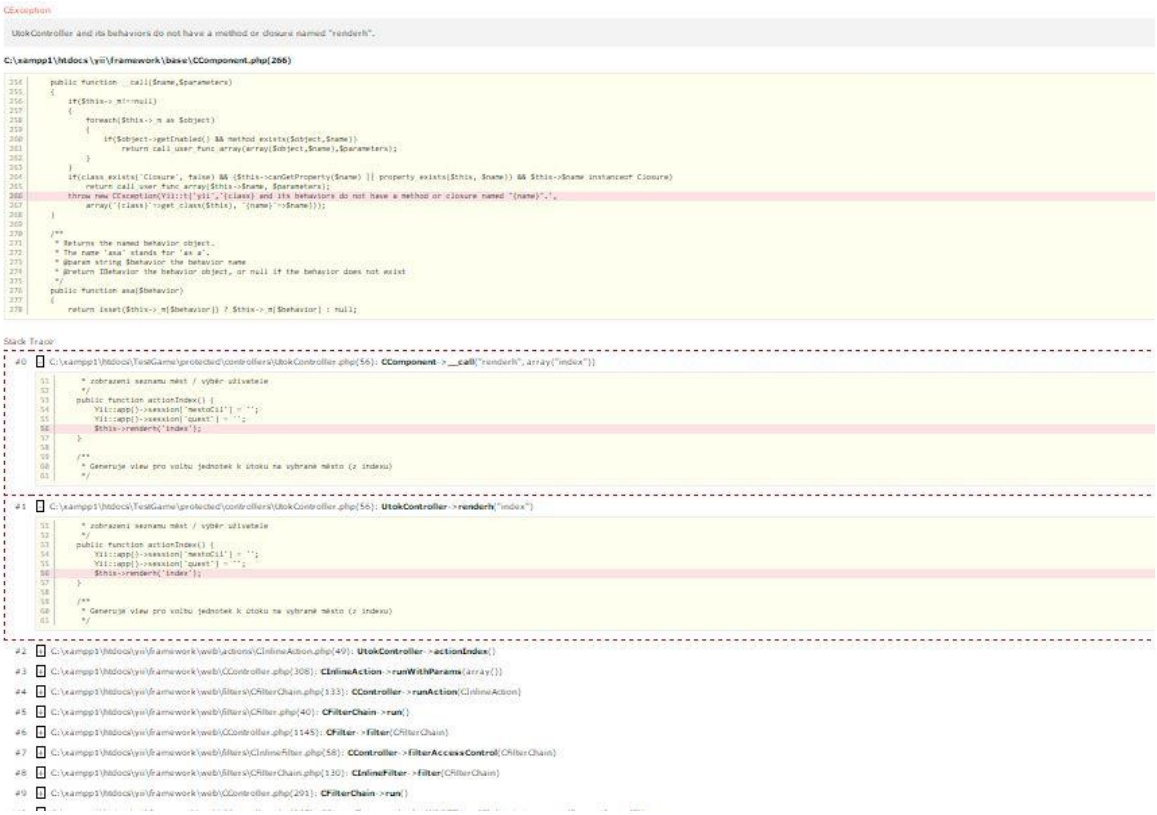
Pro vývoj jakéhokoliv systému je vždy potřeba využít nějaký nástroj, který nám bude pomáhat s laděním a odstraňováním chyb. Yii naštěstí obsahuje vlastní vestavěný nástroj na debugování kódu. Jediné co je pro zapnutí debugovacího modu potřeba, jsou následující řádky kódu umístěné v souboru „index.php“.

```
// aktivace dobugovacího módu
defined('YII_DEBUG') or define('YII_TRACE_LEVEL',true);
// definuje kolik úrovní bude zobrazeno ve výpisu
defined('YII_TRACE_LEVEL') or define('YII_TRACE_LEVEL',3);
```

Obrázek 8 - Nastavení debugovacího módu (Zdroj: Vlastní)

Prvním řádkem kódu se aktivuje onen debugovací mód. Tento mód je vhodné zapnout pouze při vývoji. Pokud je aplikace využívána pro ostrý provoz, je doporučováno ho mít vypnutý. A to z důvodu, že výpis z tohoto nástroje umožní potenciálnímu útočníkovi odhadnout fungování naší aplikace.

Druhý řádek kódu umožnuje definovat kolik úrovní je zobrazeno v ladícím výpisu. Výpis při chybě pak bude vypadat nějak následovně.



Obrázek 9 - Ukázka výpisu z debugovacího nástroje (Zdroj: Vlastní)

5.6. Routování URL

Yii nabízí poměrně jednoduchý, ale komplexní systém pro routování požadavků. Routování požadavků je ve skutečnosti překládání mezi URL a metodou Controlleru. To znamená, že router správně deleguje metodu pro konkrétní Controller spolu s jejími. (23)

Výhoda tohoto routovacího systému je zejména ve značném zkrácení URL. To usnadňuje život jak uživatelům, tak i vyhledávačům.

Routování v Yii se nastavuje za pomocí konfiguračního souboru „*main.php*“. Zde je URL manažer, který dokáže směrovat Cool-URL na konkrétní Controllery a jejich metody. URL manažer hojně využívá regulérních výrazů, díky kterým je schopen správně delegovat konkrétní požadavky. Na následujícím obrázku je ukázka URL manažera.

```
'urlManager'=>array(
        'urlFormat'=>'path',
        'showScriptName'=>false,
        'rules'=>array(
                '<controller:\w+>/<id:\d+>'=>'<controller>/view',
                '<controller:\w+>/<action:\w+>/<id:\d+>'=>'<controller>/<action>',
                '<controller:\w+>/<action:\w+>'=>'<controller>/<action>',
        ),
),
```

Obrázek 10 - Ukázka routovacího manažera v Yii (Zdroj: Vlastní)

Pole s pravidly nám definuje konkrétní situace, kde vlevo je formát URL adresy a vpravo je pravidlo, jak delegovat tento požadavek. Za zmínění také stojí parametr „*showScriptName*“, který v nastavení na „*false*“ skrývá z URL „*index.php*“. Tento systém tedy nahrazuje nepřehledný mode_rewrite, který se nastavoval pomocí souboru „.htaccess“.

5.6.1.SEO - URL (Cool-URL)

Jak bylo psáno výše SEO URL neboli Cool-URL jsou adresy, které mají parametry předávány bez jména proměnné, ve které jsou předávány. Jednotlivé parametry jsou od sebe odděleny lomítky. Tato slova oddělená lomítky se pak považují za klíčová slova pro vyhledávače. Adresy jsou lépe zapamatovatelné a mohou i vylepšit pozici ve vyhledávači. (25)

- Příklad standartní URL v Yii:
    - *http://yii_aplikace.cz/index.php?r=uzivatel/view&id=1*

Kde parametr „r“, znamenající routa, odkazuje na konkrétní Controller „uzivatel“ a na metodu tohoto Controlleru jménem „view“ s parametrem „id“. Tato metoda má za úkol zobrazit uživatele zadaného parametrem „id“, nemá proto nic společného se zobrazovací vrstvou View, jedná se pouze o shodu jmen.

- Příklad URL se zapnutým routování v Yii:
    - *http://yii_aplikace.cz/uzivatel/1*

Jak je vidět, adresa se razantně zkrátila a je mnohem přehlednější a pro uživatele pochopitelnější. Ještě je možné nahradit parametr „id“ třeba přihlašovacím jménem a vyhledávat podle něj. Pak bude naprosto jasné, že za adresou „*http://yii_aplikace.cz/uzivatel/admin*“ se skrývá náhled uživatele s přihlašovacím jménem „admin“.

6. Návrh aplikace

6.1. Požadavky na aplikaci

Při návrhu aplikace je vhodné si stanovit požadavky jak ty funkční tak ty non-funkční. Tyto požadavky stále promítat do návrhu aplikace a pak i do samotné implementace návrhu. Na začátku byly stanoveny tyto požadavky:

6.1.1. Funkční požadavky

- **Pohled běžného uživatele**

  Aplikace musí v prvé řadě umožnit registraci uživatele. Po dokončení registrace bude mít uživatel k dispozici paletu nástrojů pro správu svého města. Bude mít možnost město rozšiřovat prostřednictvím stavění nových budov. I zde bude platit klasický vztah, že čím více budov bude ve městě postaveno, tím větší bude produkce a více surovin k dalšímu stavění. Na obranu města a k podnikání útočných výprav budou k dispozici vojenské jednotky. Ty bude uživatel cvičit a tím získá potřebnou útočnou či obranou sílu. Jednotky pak bude také možné vyslat na útok, kde bude možné si zvolit jeden z těchto dvou typů útoků: Dobývací a špionážní. Uživatel bude mít také k dispozici komunikační kanál, který bude realizovaný prostřednictvím pošty. V systému bude implementován také systém úkolů, které může uživatel plnit. Tento systém úkolů by měl hraní zpříjemnit a uživateli nabídnout zajímavé bonusy.

- **Pohled správce**

  Aplikace musí být plně spravovatelná z webového prostředí bez nutnosti úpravy kódu. Uživatele bude možné dočasně zablokovat. To se bude dít v případě opakovaných pokusů o neoprávněné akce či za porušení podmínek aplikace. Správce také bude mít možnost při odesílání pošty zvolit její druh. Bude tak jednoduché vytvořit oznámení či systémovou zprávu, která bude zobrazena všem.

- **Odlišnosti od ostatních her**

  Tato aplikace nebude uživatele omezovat při stavění či trénování dlouhým čekáním, jak je tomu zvykem u tradičních her. Tyto dvě činnosti budou omezeny pouze aktuálním množství surovin a maximálním prostorem v městském skladišti či v kasárnách. Suroviny uživatel bude získávat prostřednictvím tzv. „přepočtu“, který by měl nastávat jednou za den. Tím je docíleno toho, že uživatel není nucen u hry sedět de facto pořád, aby něčeho ve hře dosáhl.

Každý uživatel bude mít své vlastní skóre. Toto skóre bude ovlivňovat možnost na někoho zaútočit, čímž by měla redukovat nesmyslné útoky, po kterých Vás hra přestane bavit. A to vše jen proto, že hráč, který hraje o rok déle, je natolik silný, že porazí všechny okolo a Vám tak hru znechutí. Tento fakt se děje v drtivé většině webových her a nutí Vás tím zaplatit za bonusové věci, čímž redukujete časovou

ztrátu na silnější spoluhráče. Proto hráč bude moci zaútočit na jiného hráče jen v případě, že skóre útočníka je stejné nebo nižší než skóre obránce.

6.1.2.Non-funkční požadavky

Je potřeba zajistit především plynulý chod aplikace a splnit veškeré požadavky, které si nárokuje Yii pro svůj běh. Proto je potřeba zajistit server, který bude splňovat tyto požadavky:

- PHP verze 5.5.x a novější
- Podpora MySQL
- PHP funkce mail()
- Podpora .htaccess
- Podpora plánovače (CRON)
- Prostor pro soubory o velikosti alespoň 150MB

Tyto požadavky splňují dnes asi již všechny free-hostingy, proto s nimi nebude problém. Největší problém je s plánováním událostí, respektive automatickým spuštěním události. Tato možnost v tarifech, které jsou zadarmo, není moc často vidět. Nicméně pro funkčnost aplikace je tato funkce potřeba. Jsou potřeba alespoň dvě automaticky spouštěné události.

Zásadní požadavky jsou na bezpečnost aplikace. Hesla uživatelů musí být šifrovaná, tak aby se při případném útoku nedala použít. Veškeré uživatelské vstupy musí být důsledně ošetřeny tak, aby nebylo možné provést SQL Injection či podobné útoky. Systém musí umožňovat přehledně nastavit jednotlivá oprávnění pro každého uživatele zvlášť, případně hromadně definovat práva za pomocí vytvořením uživatelské skupiny, respektive přidělením jednoho a toho samého oprávnění více uživatelům. V systému musí být ošetřeno volání metod, které může spouštět pouze uživatel s konkrétními právy, tak aby nedošlo k neoprávněnému zásahu do aplikace. Pokusy o takovéto neoprávněné pokusy by mohly být evidovány.

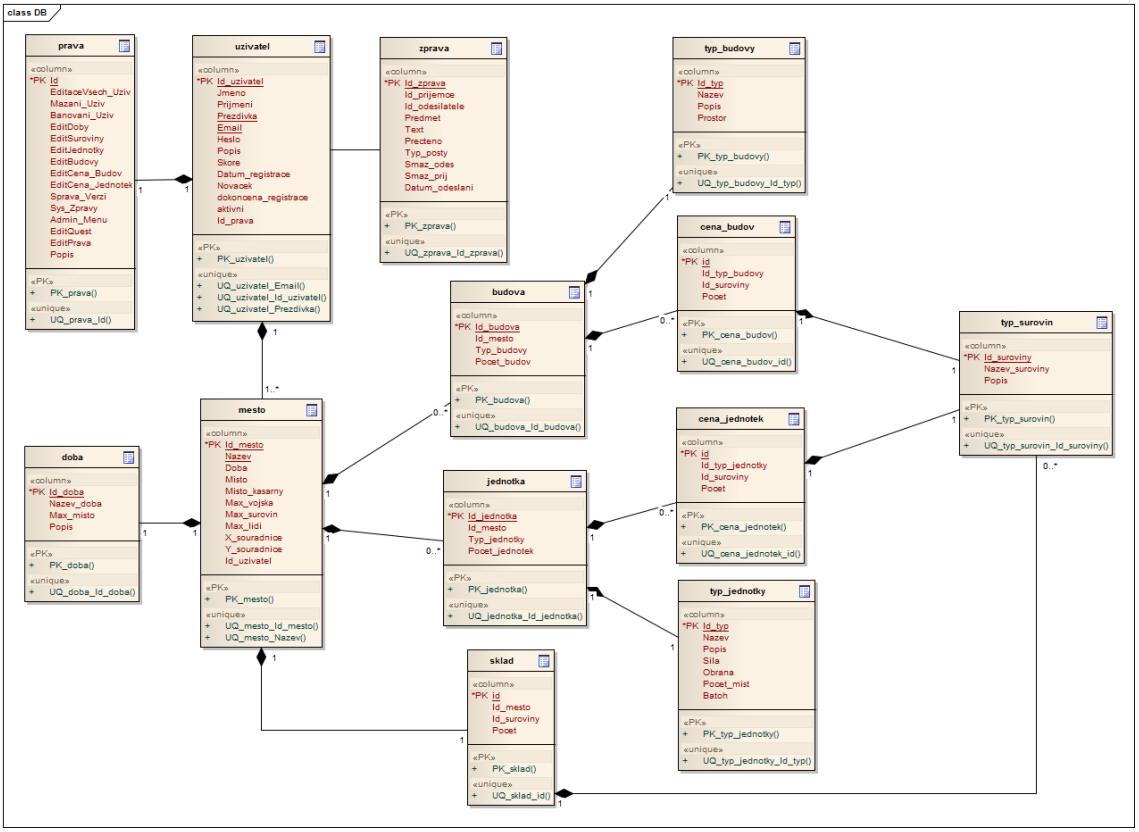
Přístupy do databáze by měly být optimalizovány a dotazy pokud možno sjednoceny tak, aby nedocházelo k několikanásobným zbytečným přístupům. Pokud to nebude nutné, měly by se z databáze vybírat pouze data, která jsou potřebná pro danou stránku.

Zvolený framework má poměrně dobře vymyšlený systém uchovávání dat pomocí session, čehož by bylo výhodné využít zejména u dat, která jsou často využívaná, čímž by odpadla nutnost je pořád stahovat z databáze.

V aplikaci nebudou použity žádné flashové animace. Veškerá grafika bude řešena na úrovni statické grafiky buď za pomocí obrázků nebo CSS. Webová aplikace musí být validní dle standardu CSS3 a HTML5.

6.2. Základní databázový model

Databáze je základem pro takovýto druh aplikací, bez ní by nemohla fungovat. Model je sám o sobě poměrně rozsáhlý, proto je zde ukázán jenom základní model, na kterém jsou vidět jednotlivé vztahy mezi konkrétními databázovými tabulkami.



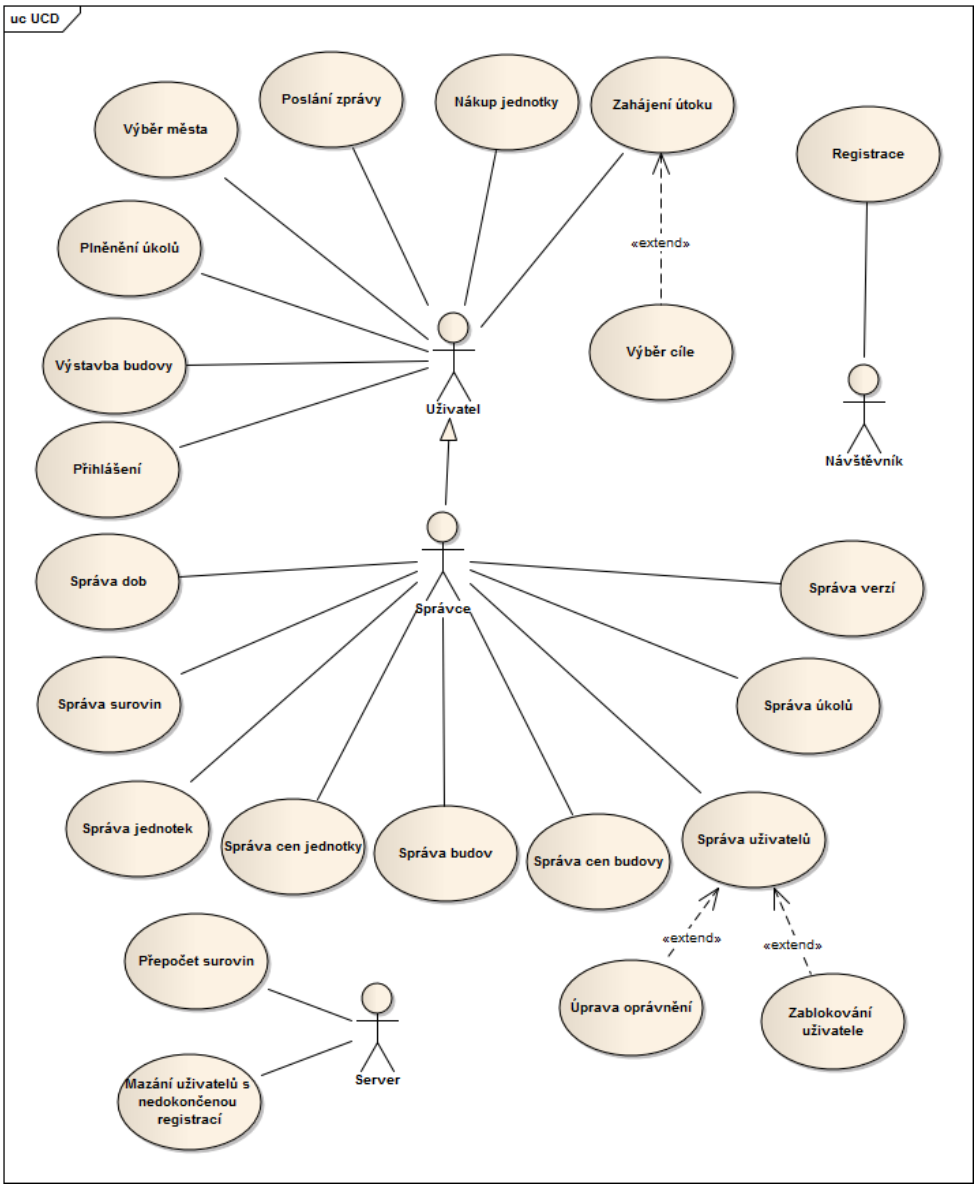
Obrázek 11 - Ukázka databázové struktury (Zdroj: Vlastní)

Pokud se na model podíváme blíže, vidíme, že každý uživatel dostane svá oprávnění. Uživateli je také k dispozici komunikační modul pro komunikaci s ostatními uživateli. Každý uživatel vlastní jedno nebo více měst. Město vždy patří do jedné konkrétní doby. Součástí města je sklad, ve kterém se nachází jednotlivé suroviny. Na město jsou také vázány konkrétní budovy a jednotky.

Celá databázová struktura je ve skutečnosti ještě o něco složitější. Zde byla pro názornost zjednodušena a byly zde zobrazeny opravdu základní prvky systému.

6.3. Struktura systému (UCD)

Pomocí diagramu typových úloh je zobrazena základní funkčnost celé aplikace. Typové úlohy byly dekomponovány z funkčních požadavků. Tento diagram slouží zejména pro představu, co může běžný uživatel v aplikaci dělat a jaké k tomu potřebuje funkce. Díky tomuto diagramu bude značně zjednodušena implementace samotné aplikace.



Obrázek 12 - Ukázka typových úloh v systému (Zdroj: Vlastní)

Jak je z tohoto diagramu patrné, celý systém bude obsahovat poměrně dost výkonných funkcí a proto bude potřeba psát optimalizovaný kód, aby nedocházelo k dlouhým prodlevám mezi spuštěním skriptu a jeho následnému provedení. Aby byla aplikace přehledná, bude nutné oddělit administrační a uživatelské prostředí. Aktér Server bude v systému zastoupen onou automaticky spouštěnou funkcí, což bude mít na starosti funkce CRON.

6.4. Návrh GUI

Jak je psané v požadavcích, hra bude postavená na statické jednoduché grafice. Při tvorbě designu byl použitý již hotový grafický template, který je možné zdarma použít na vlastních projektech. Tento template byl upraven tak, aby vyhovoval potřebám aplikace. Zejména byla upravena natolik, aby vyhovovala standardům HTML5 a CSS3.

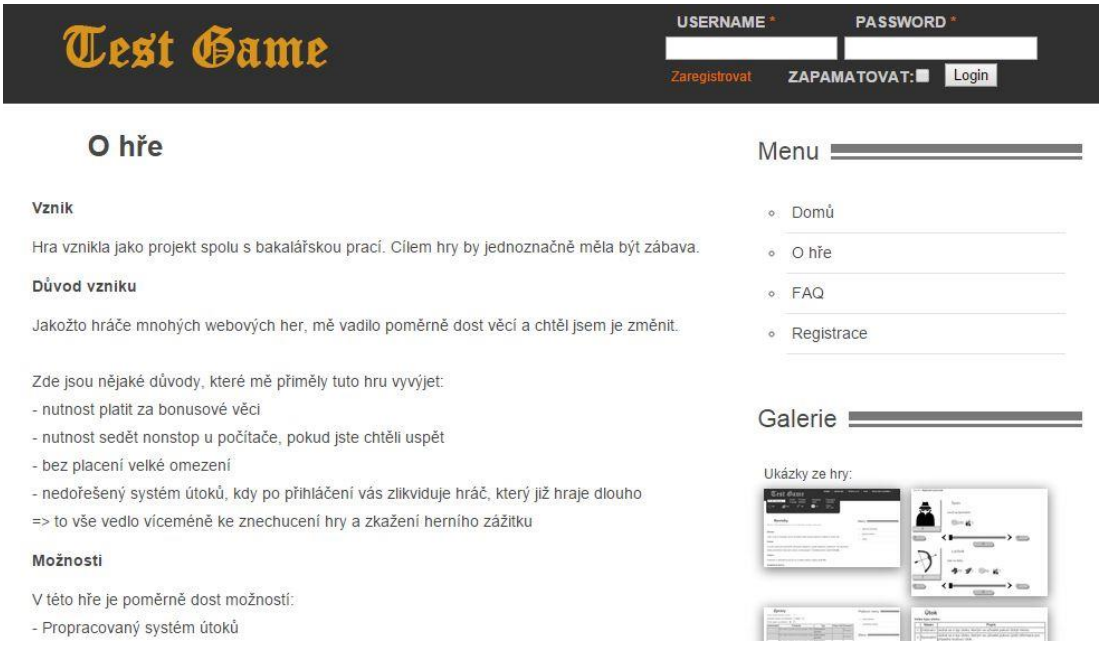
Aplikace bude rozdělena na 3 části:

- První bude informační sekce webu, která bude dostupná všem nepřihlášeným uživatelům, bude obsahovat přehledné informace o hře.
- Druhou částí bude samotná hra dostupná přihlášenému běžnému uživateli
- Poslední částí bude administrační rozhraní aplikace. To bude dostupné pouze pro privilegované uživatele.

Všechny části budou mít jednotný design vycházející z daného templatu. Lišit se budou zejména obsahem stránky a nabídkou hlavního či bočního menu.

6.4.1.Sekce pro nepřihlášené

Sekce pro nepřihlášené uživatele obsahuje jen pár stránek, na kterých je možné se dozvědět důležité informace o hře. Dále je zde přihlašovací formulář, případně odkaz na registrační formulář. Nepřihlášený uživatel si také může zobrazit informace o aktuální verzi aplikace. Do ostatních sekcí nemá nepřihlášený uživatel přístup. To je řešeno na úrovni Controlleru, který zajišťuje, že sekce, které potřebují autorizaci, budou bez přihlášení nepřístupné. Náhled na webovou aplikaci bez přihlášení vidíme na následujícím obrázku.



Obrázek 13 - Ukázka herního prostředí (Zdroj: Vlastní)

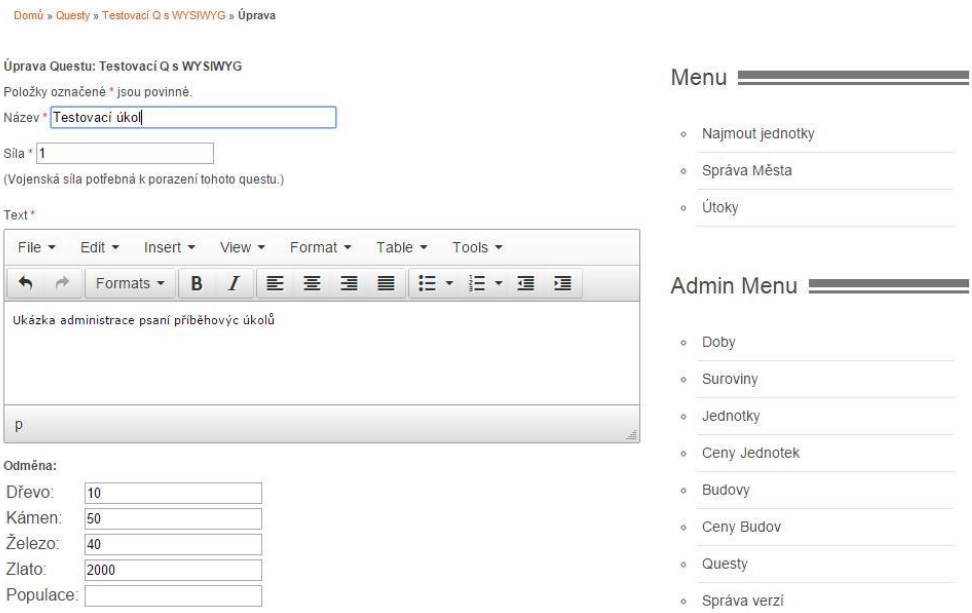
6.4.2.Sekce pro přihlášené uživatele

Když se hráč přihlásí, hra mu nabídne mnohem větší možnosti. Zobrazí se informace o jeho městě, poště. Má pak přístup k vlastní hře a může v rámci svých práv ovládat hru. Jak vypadá hra po přihlášení, vidíme na následujícím obrázku.

Obrázek 14 - Ukázka hry po přihlášení (Zdroj: Vlastní)

6.4.3.Sekce administrace

Administrátor (moderátor) má, díky vyšším oprávněním, větší možnosti správy hry. Může editovat budovy, jednotky. To se bude dít zejména k balancování jednotek. Také může vytvářet a upravovat příběhové úkoly. Proto má zobrazeno další menu, díky kterému se dostane k administraci jednotlivých prvků hry. To vše je vidět na následujícím obrázku.



Obrázek 15 - Ukázka administrace (Zdroj: Vlastní)

7. Implementace aplikace

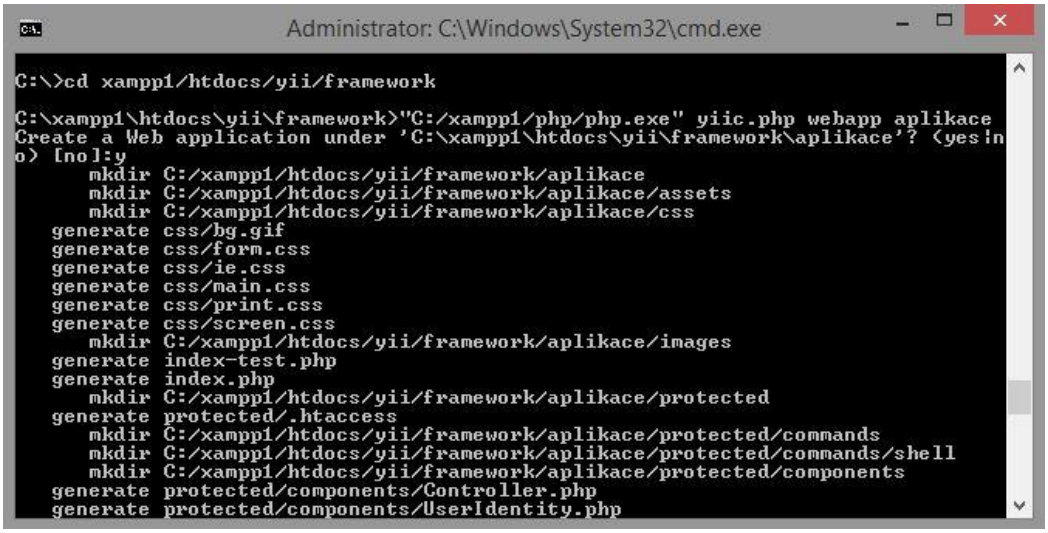
Nakonec byl použitý framework Yii ve verzi 1. 1. 16. Nová verze tohoto frameworku vyšla bohužel až v poměrně pokročilé části vývoje této aplikace. Vzhledem k vzájemné nekompatibilitě jsem proto byl nucen zůstat u této verze. Tato verze je však i nadále podporována a budou na ní vycházet opravné updaty.

7.1. Vytvoření aplikace

Vytvoření nové aplikace v Yii je poněkud nezvyklé a neprobíhá úplně standartním způsobem, jak jsme zvyklí. Yii má opět nástroj, který za nás vytvoří základní skeleton aplikace spolu se všemi potřebnými soubory. Tento nástroj se jmenuje Yiic a využívá příkazové řádky. K vytvoření základní aplikace je tak třeba jediný příkaz:

- *„Yii/framework/yiic.php webapp nazevAplikace“*

Kde první část je cesta k souboru yiic.php, který je zodpovědný za vytvoření základního skeletonu aplikace. Druhým je parametrem webapp, ten zajistí, že bude vygenerována webová aplikace a posledním parametrem je název aplikace, který se pak bude objevovat ve výsledné aplikaci. Vše je názorně ukázáno na následujícím obrázku.



Obrázek 16 - Vytvoření nové aplikace v Yii (Zdroj: Vlastní)

Dále je potřeba otevřít soubor „main.php“, který se nachází v adresáři „protected/config/“. Zde se musí nastavit přístup k databázi. Po těchto krocích je aplikace již plně funkční.

7.2. Vytvoření CRUD pomocí Gii

V této aplikaci je ve velké míře využíváno klasického modelu, kdy se záznamy pouze přidávají, mažou a upravují. Nejvíce těchto stejných akcí nalezneme v administračním rozhraní. Proto je vhodné nechat si vygenerovat CRUD akce pomocí vestavěného nástroje Gii. To nám ušetří hodně práce a času. Jediné, co je potřeba upravit jsou

Controllery. Do těch musíme přidat omezení týkajících se oprávnění. Případně se musí poupravit akce Controllerů, tak aby nám vyhovovali. Dále je vhodné v modelu upravit atributy pro popisky vstupů případné vztahy mezi tabulkami. Ve finále je administrace některých tabulek v databázi vytvořena během několika minut a je možné se věnovat věcem, které jsou pro systém stěžejní místo toho, aby se trávil dlouhý čas nad věcmi, které jsou pro systém ne až tak důležité.

Jak vypadá prostředí nástroje Gii je názorně ukázáno v kapitole 5.2. Gii vygeneruje plnohodnotný a funkční systém pro administraci a zobrazování záznamů z databáze. Pak už jen záleží na programátorovi aplikace, zdali mu bude základní funkcionalita postačovat či nikoliv.

7.3. Přihlášení uživatelů

Uživatelé se v aplikaci budou přihlašovat pomocí kombinace jednoznačného uživatelského identifikátoru a hesla. Uživatel si bude moci zvolit, respektive systém bude umožňovat, přihlášení buď podle registrovaného e-mailu, nebo podle zadaného uživatelského hesla. Tento systém je výhodný proto, že si uživatel zvolí to, co je pro něj jednodušší. Obecně je známo, že si uživatelé pamatují více svůj e-mail, nežli přihlašovací jména, která většinou mění a nepoužívají jedno pro všechny servery. Na druhou stranu uživatelské jméno bývá kratší než e-mail a tím ušetří uživateli pár úhozů na klávesnici při přihlašování. Proto byla navržena tato možnost volby.

Yii řeší přihlašování a identifikaci uživatelů v systému pomocí třídy CUserIdentity. Jedná se o základní třídu pro reprezentaci identit, které jsou ověřovány na základě nějakého uživatelského jména a hesla. V základu tato třída předpokládá, že uživatelské jméno je unikátní, a proto je použité jak identifikátor konkrétní identity. (26)

Tato třída ovšem primárně nedokáže řešit autentizaci pomocí databáze. Proto je nutné vytvořit vlastní třídu, která bude dědit vlastnosti původní třídy. Původní třída implementuje metodu authenticate(), která má na starosti ověřit identitu uživatele na základě zadaných přihlašovacích údajů. Tu je nutné ve zděděné třídě překrýt a doplnit vlastní kontrolní mechanismus pro přihlášení uživatele. Na obrázku vidíme ukázku kódu.

```php
<?php
class UserIdentity extends CUserIdentity
{
const ERROR_STATUS_BANNED = 4;

public function authenticate()
{
        //Uživatel zadal přezdívku
        $u = Uzivatel::model()->findByAttributes( array('Prezdivka'=>$this->username));

        //Uživatel zadal Email
        if (!$u){
        $u = Uzivatel::model()->findByAttributes( array('Email'=>$this->username));
        }

        if (!$u) {                                          // Neplatný Email/Přezdívka
            $this->errorCode = self::ERROR_USERNAME_INVALID;
        }
        else if (!$u->matchesPassword($this->password)) {   // Neplatné Heslo
            $this->errorCode = self::ERROR_PASSWORD_INVALID;
        }
        else if ($u->aktivni == 0) {                        // Uživatel je blokován
            $this->errorCode = self::ERROR_STATUS_BANNED;
        } else {


            $this->_id = $u->Id_uzivatel;
            $this->username = $u->Prezdivka;
            $this->errorCode = self::ERROR_NONE;
```

Obrázek 17 - Ukázka UserIdentity sloužící pro přihlášení uživatele (Zdroj: Vlastní)

7.4. Bezpečnost uživatelů

Každá webová aplikace, pokud uchovává uživatelská hesla, by měla dodržovat jistá pravidla, aby nedošlo k jejich zneužití. Nejhorší situace, která se může stát je ta, že se vám někdo dostane do databáze a stáhne si veškeré informace o uživatelích včetně jejich hesel. Je obecně známo, že většina uživatelů používá na webových stránkách jedno heslo a stejné uživatelské jméno. Bohužel to tak bývá i v případě e-mailového účtu. Takže pokud útočník získá uživatelovo heslo, první věc, kterou zkusí, je přihlášení na registrovaný email pomocí hesla, které bylo v databázi.

To, že takto unikne část nebo celá databáze se stává a stává se to i velkým firmám typu E-Bay, Google atd. Proto je důležité myslet dopředu a udělat vše proto, aby nemohl útočník heslo zneužít. Není totiž nic horšího, než psát uživatelům omluvný e-mail, že jejich heslo bylo prozrazeno a vy tak doporučujete, aby si ho všude změnili.

Proto se k uložení hesel používají takzvané otisky. Otisk hesla (Hash) je řetězec znaků, který je vygenerovaný podle nějakého kryptografického hašovacího algoritmu. A v případě ověření uživatele při přihlášení se jen ověřují ony otisky a ne hesla samotná. K vytvoření otisku se využívá několik algoritmů. Nejznámější je MD5 a SHA.

7.4.1.MD5

MD5 je popsán v internetovém standardu RFC 1321 a vytváří hash o velikosti 128 bitů. Byl vytvořen v roce 1991 Ronaldem Rivestem, aby nahradil dřívější hašovací funkci MD4, který byl klasifikován jako nedostatečně bezpečný. (27)

Vzhledem ke své malé délce (pouze 128 bitů) byla už v roce 2004 oznámena úplná kolize pro MD5 hash.

„*Kolize je stav, kdy hash, kontrolní součet nebo otisk kryptografické hašovací funkce přiřazuje stejný výstup různým vstupním datům.*“ (28)

MD5 byl proto mírně zdokonalen a byly přidány soukromé a veřejné klíče. Ovšem i tak byly nalezeny kolize. Navíc díky upravenému algoritmu pro nacházení kolizí bylo možné nalézt kolizi do jedné minuty a to na běžně dostupném hardwaru. Proto bylo doporučeno začít využívat jiný algoritmus, a to konkrétně SHA.

MD5 je tak díky své jednoduchosti efektivně napadnutelný dvěma způsoby. Ten první je, že se náhodně generují hesla a porovnávají se jejich otisky. Vzhledem k výkonnosti dnešního hardwaru je možné na procesorovém jádře o taktu 3,2GHz generovat přibližně 8 milionů MD5 otisků za vteřinu ze vstupu o velikosti 16 bajtů. (29)

Druhá možnost je využití kolize, kdy nepotřebujeme znát původní řetězec, protože ho nahradíme vlastním, který má stejný výstupní otisk jako původní řetězec.

7.4.2.SHA

„SHA navrhla organizace NSA (Národní bezpečnostní agentura v USA) a vydal NIST (Národní institut pro standardy v USA) jako americký federální standard. SHA je rodina pěti algoritmů: SHA-1, SHA-224, SHA-256, SHA-384 a SHA-512. Poslední čtyři varianty se souhrnně uvádějí jako SHA-2. SHA-1 vytvoří obraz zprávy dlouhý 160 bitů. Čísla u ostatních čtyř algoritmů značí délku výstupního otisku v bitech. SHA se používá u několika různých protokolů a aplikací, včetně TLS a SSL, PGP a SSH, ale i pro kontrolu integrity souborů nebo ukládání hesel. Je považována za nástupce hašovací funkce MD5.“ (30)

„Bezpečnost SHA-1 byla poněkud zpochybněna kryptografickými odborníky nebo například i firmou Google, ačkoli nebyly oznámeny žádné útoky na varianty SHA-2. Varianty SHA-2 jsou algoritmicky stejné s algoritmy SHA-1 (ten v roce 2014 používá 90% webů). Proto jsou snahy o vývoj vylepšených hašovacích funkcí.“ (30)

I SHA tedy trpí problémem, že je snadno generovatelný na běžném hardwaru jako MD5, je sice o něco pomalejší, ale jen v řádu jednotek procent. Dnes již existují rainbow table s nejběžnějšími slovníkovými hesly i pro SHA.

7.4.3.BCrypt

BCrypt je hašovací algoritmus, který vychází ze šifrovacího algoritmu Blowfish. Výsledný hash se skládá z několika částí. Z prefixu "$2a$", "$2y$" nebo "$2x$", který indikuje, že se jedná o Bcrypt hash v modulárním kryptovacím formátu. Zbytek hashe se pak skládá z parametru ceny, 128 bitového řetězce salt a 192 bitového hashe původní hodnoty. (31)

Bcrypt algoritmus závisí na „Eksblowfish“ algoritmu, který je zobrazen na následujícím obrázku.

```
EksBlowfishSetup(cost, salt, key)
    state ← InitState()
    state ← ExpandKey(state, salt, key)
    repeat (2^cost)
        state ← ExpandKey(state, 0, key)
        state ← ExpandKey(state, 0, salt)
    return state
```

Obrázek 18 - Ukázka kódu algoritmu Eksblowfish (Zdroj: en.wikipedia.org/wiki/Bcrypt)

Počet průchodů algoritmu je závislý na parametru ceny, kdy platí, že počet opakování se rovná 2^cenu algoritmu. Tento návrh tak dává tomuto algoritmu jedinečnou vlastnost, že ze dvou stejných řetězců nevznikne jeden a tentýž otisk.

Všechny výše uvedené metody jde ještě pro vyšší bezpečnost zdokonalit použitím „Salt“. Jedná se o náhodný řetězec znaků, který programátor daného systému nechá vygenerovat, případně vymyslí. Tento Salt je pak vždy stejný a vždy se přidává k heslu. Z kombinace heslo + salt se pak tvoří onen otisk a ten je teprve ověřován v databázi. Tato metoda je účinná pouze v tom případě, že útočník má jen k dispozici otisky hesel. Ty pak porovnává pomocí vlastních otisků ze slovníkových hesel či náhodně generovaných hesel. Útoku, který využívá předpočítané tabulky s otisky hesel se říká Rainbow Table attack. Vzhledem k tomu, že k heslu je přidán onen neznámý salt, je tento útok vždy neúspěšný. Ovšem pokud se útočník již dostal ke kódu naší aplikace, není problém salt zjistit a tato ochrana pak postrádá smysl.

Základem je tedy dostatečně silný šifrovací algoritmus, který sice nikdy nedokáže uživatele zastavit, ale dokáže ho zbrzdit natolik, že se mu nevyplatí hesla rozšifrovat. A to je i náš cíl. Proto volíme kombinaci BCrypt + salt. Tento algoritmus sice „platí“ za svou bytelnost zvýšenými nároky na výkon serveru, ale bude využíván tedy vždy jen jednou a to při přihlášení, a proto jsou tyto jeho nevýhody zanedbatelné. (34)

7.5. Registrace uživatelů

V aplikaci je využita dvoufázová registrace, která bude vyžadovat ověření e-mailové adresy k dokončení registrace.

7.5.1. První krok

Uživatel se chce zaregistrovat, a proto je po kliknutí na odkaz přesměrován na registrační formulář. Nejdůležitější prvky jsou textové vstupy pro e-mail a uživatelské jméno. Ty jsou povinné a zároveň musí být zaručeno, aby byly unikátní. Na následujícím obrázku vidíme ukázku tohoto formuláře.

Obrázek 19 - Ukázka registračního formuláře (Zdroj: Vlastní)

Zajištění unikátnosti e-mailu a uživatelské přezdívky je zaručeno dvěma způsoby. První je „client-site“, neboli ověření přímo u uživatele ještě před odesláním formuláře. Tento způsob funguje následovně. Po vyplnění vstupu je pomocí JavaScriptu odeslán požadavek na server. Ten je zpracován v případě unikátnosti nevrací žádný výsledek, zatímco v opačném případě vrací hlášení o chybě. Tato kontrola má také za následek nedovolení odeslání formuláře, dokud nejsou oba požadované vstupy unikátní. Tato ochrana je poměrně efektivní a pohodlná pro uživatele, protože vidí ihned po vyplnění, že je něco špatně. Ovšem tuto ochranu lze poměrně snadno vyrušit zablokováním JavaScriptu v prohlížeči (třeba jen pro konkrétní stránku). A proto je zde implementováno ještě jedno bezpečnostní opatření.

Druhý způsob ověření již klasicky probíhá na serveru. V Yii se právě pro tyto účely využívají pravidla, která jsou umístěna v modelu. V následující ukázce kódu vidíme pravidla právě pro registraci uživatele.

```php
public function rules() {
    return array(
        array('Email, EmailZ, licence, Prezdivka' ,'required', 'on'=>'registrace'),
        array('Jmeno, Prijmeni, Prezdivka, Email, EmailZ', 'length', 'max' => 50),
        array('Heslo, password_st', 'length', 'max' => 80),
        array('Popis', 'safe'),
        array('Email, Prezdivka','unique'),
        array('Email, EmailZ', 'email'),
        array('licence', 'compare', 'compareValue' => true, 'on'=>'registrace',
              'message' => 'Pro dokončení registrace musíte souhasit s podmínkami!' ),
        array('EmailZ', 'compare', 'compareAttribute'=>'Email', 'on'=>'registrace', 'message'=>'E-maily se musí shodovat!'),
        // Ověření CAPTCHA kódu
        array('verifyCode', 'captcha', 'on'=>'registrace', 'allowEmpty'=>!CCaptcha::checkRequirements()),
    );
}
```

Obrázek 20 - Ukázka validace formuláře v Modelu (Zdroj: Vlastní)

Z výše uvedených pravidel jasně vyplývá celá validace registračního formuláře. E-mail, E-mail znovu, licence a přezdívka jsou položky povinné. Pak následuje jejich validace ohledně velikosti. Nejdůležitější validační pravidlo se skrývá pod parametrem „unique“, který zajišťuje onu unikátnost uživatelského jména a e-mailu. Pak následuje validace, zdali uživatel souhlasí s podmínkami serveru a ověření správnosti CAPTCHA kódu. Pokud validace odhalí v nějakém kroku chybu, je uživateli vrácen formulář spolu s popisem chyby.

Yii má validační pravidla obohacená ještě o parametr „on“. Ten frameworku říká, v jaké akci má být ono pravidlo využito. To je vhodné zejména v případě, kdy máme více formulářů, které se zobrazují při různých akcích, a potřebujeme stejné vstupy validovat jinak nebo formuláře obsahují jiné povinné vstupy.

Po úspěšném odeslání přijde uživateli na zadaný email vygenerované heslo, díky kterému se bude moci přihlásit a pokračovat v registraci. U uživatele se uchovává datum a čas přihlášení. Jeden z důvodů, proč je tento údaj uchováván, je ten, že pokud uživatel nedokončí registraci do 48hodin od odeslání registračního formuláře, je účet smazán.

7.5.2. Krok druhý

Pro dokončení registrace se uživatel musí přihlásit do 48 hodin od registrace. Pokud tak učiní, je při přihlášení přesměrován na formulář, kde dokončí svou registraci. Zde si uživatel zvolí název svého městečka a je po něm také vyžadována změna hesla.

Název městečka musí být unikátní, a proto je kontrolován jako v předešlém případě uživatelská přezdívka či E-mail. Tento formulář není ničím zajímavý, to podstatné se děje až po jeho odeslání a úspěšné validaci.

Po úspěšném odeslání formuláře je uživateli vytvořeno městečko, kterému jsou náhodně vygenerovány souřadnice, naplněn sklad základním počtem surovin a přiděleny základní jednotky a budovy. Toto je důvod, proč se toto děje až v druhém kroku, protože je jednodušší po nedokončené registraci (třeba úmyslně) smazat jeden záznam s uživatelem z jedné tabulky, než přibližně 17 záznamů z pěti různých tabulek. Tím je zajištěna i jistá ochrana proti zahlcení databáze.

Uživatel je následně odhlášen (respektive nikdy ani přihlášen nebyl) a přesměrován na hlavní stránku, kde se může přihlásit již s novým heslem a začít hrát.

7.6. Správa uživatelů

Správa uživatelů je nástroj, díky kterému může uživatel upravit svůj profil. Jak moc dokáže upravit profil uživatele, závisí na jeho oprávněních. Pokud se jedná o běžného uživatele, dokáže zobrazit pouze svůj profil. I kdyby znal identifikační číslo jiného uživatele, systém ověřuje totožnost uživatele. Proto se pod jiným identifikačním číslem zobrazí stále jeho profil. Na svém profilu vidí registrovaný E-mail, přezdívku a úroveň oprávnění. To jsou údaje, které uživatel nemůže změnit. Dále vidí vyplněné jméno, příjmení a popis. Tyto údaje si uživatel může libovolně měnit.

Poslední věcí, kterou může uživatel změnit, je jeho heslo. V administraci vidí uživatel dva textové vstupy. Jeden je pro zadání starého hesla a druhý je pro zadání nového hesla. Pokud není ani jeden vstup vyplněn, systém zachová původní heslo. Pro úspěšnou změnu hesla je nutné znát původní heslo. Pokud je vyplněno správně staré heslo a je vyplněn i vstup s novým heslem, je heslo úspěšně změněno. Změnu hesla takovýmto způsobem technicky umožnuje samotné Yii. V modelu uživatelů se nachází dvě funkce, které pracují s heslem. První je funkce „afterFind()“. Ta umožnuje, že po vyhledání záznamu uživatele je heslo uloženo do jiné proměnné. Proto může zůstat vstup pro heslo prázdný. Druhá funkce se jmenuje „beforeSave()“. Ta se stará o zpětné uložení hesla v případě, že nedochází k žádné změně hesla. Obě funkce jsou názorně vidět na následující ukázce.

```
function afterFind() {
        $this->_oldPassword = $this->Heslo;
        $this->Heslo = '';
        return parent::afterFind();
    }
```

Obrázek 21 - Ukázka funkce afterFind() (Zdroj: Vlastní)

```
function beforeSave() {
        //Při Updatu kdy nebylo vyplněno heslo => Neměním heslo
        if (!$this->Heslo) {
            $this->Heslo = $this->_oldPassword;
            return parent::beforeSave();
        //Uživatel mění heslo => kontrola zdali je staré heslo správné
        }elseif ($this->matchesPassword($this->password_st)) {
                $this->Heslo = $this->getHash($this->Heslo);
                $this->password_st = '';
                return parent::beforeSave();
        }else {
                $this->addError('password_st', 'Zadali jste špatné původní heslo!');
        }
    }
```

Obrázek 22 - Ukázka funkce beforeSave() (Zdroj: Vlastní)

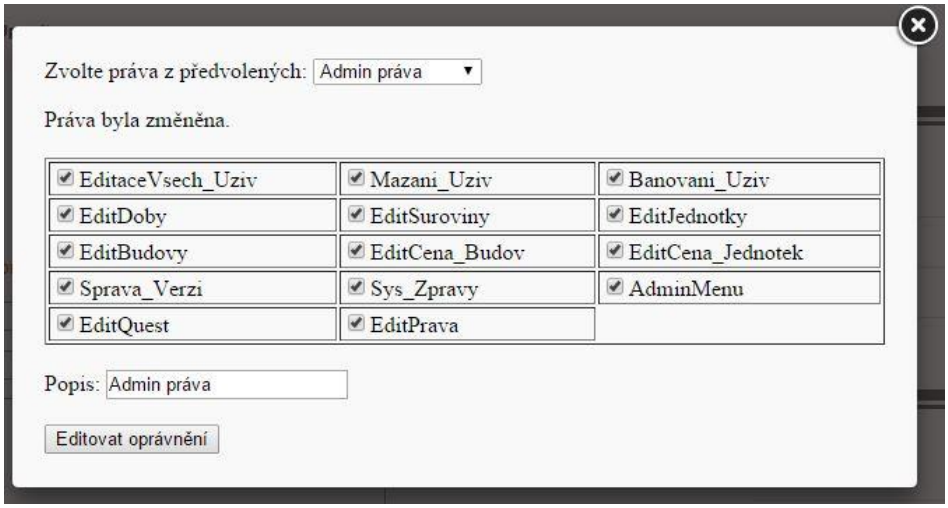
V administraci je možné další rozšíření, kdy by si uživatel upravoval čas přepočtu surovin. Bohužel tato možnost není implementována z důvodu omezení serveru. Další možnost na rozšíření aplikace by byla ta, že uživatelské profily by byly veřejné (při zachování soukromí uživatelů – nebyl by zobrazován E-mail atd.). Uživatelé by tak věděli, kolik měst jiní uživatelé mají, případně jaké mají skóre.

Pro oprávněné uživatele je správa uživatelů o něco zajímavější. Zásadní změna je v tom, že oprávněný uživatel vidí a může upravit i účty jiných uživatelů. Oprávněný uživatel také může zablokovat uživatele v případě, že poruší podmínky serveru.

Administrátor disponuje ještě jednou funkcí. Může měnit oprávnění uživatelů. V databázi jsou předdefinovány tři základní oprávnění:

- Administrátorská práva
  - Uživatel s tímto oprávněním může v systému prakticky vše, přes editaci uživatelů až po editaci samotné hry
- Moderátorská práva
  - Moderátor je funkce, která pomáhá administrátorovi spravovat webovou aplikaci a má o poznání nižší práva
- Uživatelská práva
  - Uživatel nemůže nijak zasahovat do systému a nemůže měnit žádné údaje kromě svých vlastních

Administrátor také může v rámci systému definovat nová práva přímo na míru konkrétnímu uživateli. Z těchto práv se pak vytvoří další pravidlo, které lze přidělit více uživatelům. Přidělování práv vidíme na následujícím obrázku:



Obrázek 23 - Ukázka přidělování práv (Zdroj: Vlastní)

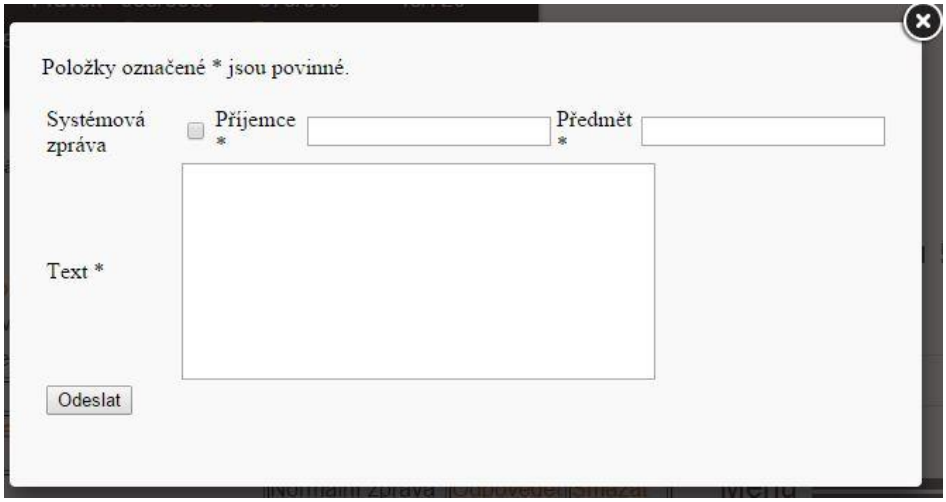
Přidělování práv je zobrazeno pomocí fancyboxu. Ten načte iframe s příslušným formulářem pro výběr oprávnění. Pokud edituje uživatel práva sám sobě, dojde k okamžité změně práv.

7.7. Komunikační modul

V aplikaci je implementován komunikační modul, který umožní komunikaci mezi uživateli pomocí textových zpráv. Pokud má uživatel potřebná oprávnění, modul mu umožní odeslat globální zprávu. Ta se poté zobrazí všem uživatelům. Tyto zprávy se dají využít například k rozeslání systémových zpráv či jako oznámení.

Odeslání nové zprávy je řešeno pomocí nového okna otevřeného ve fancyboxu. Zde je jednoduchý formulář se vstupem pro uživatelskou přezdívku, která zde slouží pro

identifikaci příjemce. Pokud je tento vstup nesprávně vyplněn a zpráva je odeslána, pak je uživateli vypsáno oznámení o chybě. Vstup s přezdívkou je vymazán, ale předmět zprávy a vlastní text je zachován. Ověřování na straně klienta zde není z důvodu zamezení spamu. V opačném případě, když je přezdívka zadána správně, je zpráva odeslána a uživateli je zobrazeno oznámení o úspěchu akce.



Obrázek 24 - Náhled na formulář pro psaní nové zprávy (Zdroj: Vlastní)

Podobně funguje zobrazení příchozí či odchozí zprávy. Zprávy se otvírají také v novém okně otevřeném ve fancyboxu. Zde je však pevný design a uživatel si tak může zprávu pouze přečíst. Nově příchozí nepřečtené zprávy jsou uživateli oznamovány v hlavním menu ve formě počtu zpráv, které nejsou přečtené. S tímto faktem se pojí jedna vlastnost. Uživatel, který zprávu odešle, vidí, zdali si ji příjemce již přečetl nebo tak ještě neučinil.

Uživatel se v komunikačním modulu pohybuje pomocí přídavné boční nabídky, kde si volí, zdali chce vidět odeslané či přijaté zprávy. V přijatých zprávách má uživatel volbu rychlé odpovědi. Po kliknutí na „odpovědět“ se uživateli předvyplní příjemce zprávy a předmět. Uživatel také může zprávy mazat. Zde jsem zvolil možnost, že zprávy budou pouze danému uživateli zneviditelněny. Nebudou tak fyzicky smazány. Toto řešení jsem zvolil kvůli případnému řešení sporů, vulgarit či porušení podmínek serveru.

Komunikační modul dále obsahuje jednoduché filtrování pošty v rámci konkrétní přezdívky. Uživatel si tedy může zobrazit poštu od nebo pro konkrétního uživatele. Tento filtr funguje na jednoduchém formuláři a „$_SESSION“ proměnné, která nám zajišťuje uchovávání id uživatele, kterého chceme zobrazit. Seznam uživatelů, ze kterých je možné vybrat v komponentě „select“, je závislý na přijaté/odeslané poště. Jsou zde zobrazeni tedy jenom ti uživatelé, se kterými uživatel komunikuje.

Stránkování je poměrně efektivně řešeno pomocí vestavěného widgetu v Yii. Ten funguje tak, že se nastaví limit na počet záznamů na stránku a počet záznamů, které chceme rozdělit na stránky. Kód pro Controller vypadá pak následovně:

```
//Zjisti počet záznamů
$count = Zprava::model()->count($criteria);
//Nastav widget pro stránkování
$pages = new CPagination($count);
$pages->pageSize = Yii::app()->session['strankovani'];
$pages->applyLimit($criteria);
//Načti data z DB
$Zpravy = Zprava::model()->findAll($criteria);
$this->render('prijate', array('Zpravy' => $Zpravy, 'pages' => $pages, 'count' => $count, 'pageSize' => $pages->pageSize));
```

Obrázek 25 - Ukázka kódu pro stránkovací widget (Zdroj: Vlastní)

Pro upřesnění, proměnná „$criteria“ obsahuje SQL podmínky pro samotný dotaz. V tomto případě je obsahem proměnné klauzule WHERE a ORDER, podle které jsou zprávy v databázi vyhledány. „PageSize“ má na starosti počet záznamů zobrazených na stránce. Zde je vidět, že počet je uložen v SESSION proměnné, kterou uživatel může ovlivnit dle svých preferencí. Zásadním pro fungování stránkování je řádek, kdy se aplikují limity na daná kritéria pro SQL dotaz. Pokud je následovně po tomto zavolán SQL dotaz, jsou záznamy rozděleny do stránek dle nastaveného počtu záznamů na stránku.

Ve view se pak už jen přidá gui, ve kterém je výběr stránek a zvýraznění aktuální stránky. Vše jednoduše podle kódu, který je vidět na následujícím obrázku. Proměnné „$pages“, „$count“, „$pageSize“ jsou ty samé se stejnými hodnotami jako na ukázce výše v Controlleru.

```
<?php
$this->widget('CLinkPager', array(
    'currentPage' => $pages->getCurrentPage(),
    'itemCount' => $count,
    'pageSize' => $pageSize,
));
?>
```

Obrázek 26 - Gui pro stránkovací widget (Zdroj: Vlastní)

7.7.1. Fancybox

Již několikrát zde byl použitý pojem fancybox. Jedná se o plugin třetí strany, který slouží k zobrazování obrázků, případně jiných dat či stránek v designem oddělené sekci, umístěné nad celou stránkou. Jeho možnosti jsou poměrně rozsáhlé a je proto možné zobrazit nejen obrázky, ale i iframe se stránkou, media typu video na YouTube či nějaký flashový kód.

Tento plugin podléhá GPL licenci a proto je možné ho využít i pro komerční účely případné kódy libovolně editovat. Samotná editace je pak poměrně jednoduchá a přehledná. Plugin také poskytuje různé nastavení přechodů a designů. Také je možné obrázky seskupit a nechat je zobrazovat jako celou galerii. Zde jsou možnosti jako zobrazení popisků obrázků, případně miniatury ve spodní straně stránky pro lepší orientaci.

Implementace tohoto pluginu je velice snadná. Jediné, co je potřeba zajistit, je načtení potřebných JavaScritových souborů JQuery. K tomu je vhodné použít CDN (Content Delivery Network) server, což zajistí rychlejší načtení JQuery knihoven.

Dále je potřeba svázat odkazy s fancyboxem. To se provádí tak, že se do atributu class v HTML tagu přidá tato informace: „fancybox“. Tímto se docílí otevření ve fancyboxu. Můžeme přidávat ještě další parametry, které budou dále specifikovat otvíraný obsah jako například: „fancybox fancybox.ajax“.

7.8. Přepočet

Přepočet je akce systému, která musí probíhat automaticky a oddělena od uživatelů. Smysl přepočtu je ten, že se v určitý čas uživateli přičtou suroviny v závislosti na tom, kolik vlastní budov. Čím více budov bude uživatel vlastnit, tím více se mu připočte surovin. V systému bylo zavedeno jedno omezení, které zajišťuje, že uživatel nebude moci vlastnit více surovin, než je kapacita jeho skladu. Ten lze ovšem také rozšířit a kapacitu tak pro další přepočet zvýšit. Pokud není dostatek místa pro suroviny, je z důvodu efektivity algoritmu uživatel přeskočen.

Z kontrolních důvodů je přepočet logován. Do databáze je zapsáno kolik měst a kolik uživatelů bylo do přepočtu zahrnuto. Dále je zapsána doba trvání algoritmu a datum, kdy byl přepočet proveden. Pokud tedy správce zjistí, že doba přepočtu trvá déle, než obvykle, bude nutné tento problém řešit. Algoritmus pro přepočet je poměrně složitý. Mluvíme tu o algoritmu s cyklem „FOR“ vnořeným do třetí úrovně. Počet průchodů se pak počítá podle vzorce:

- M * N * L
    - M = Počet uživatelů
    - N = Počet uživatelových měst
    - L = Počet surovin, při nastavení města na dobu 1 se L = 5

Proto je vhodné rozšíření tohoto skriptu o možnost volby pro uživatele, kdy se jejich suroviny budou přepočítávat. Uživatel by si tak zvolil konkrétní hodinu, kdy mu proběhne přepočet. Tím se nároky na výkon rozmělní mezi jednotlivé hodiny. Případně by šlo ještě zavést další omezení, které zajistí rozložení zátěže. Nastavit limit pro maximální počet uživatelů přepočítávaných v konkrétní hodinu.

Aby se skript spouštěl pravidelně v určitý čas, je vhodné využít funkce CRON.

7.8.1. CRON

„*CRON je softwarový démon, který v operačních systémech automatizovaně spouští v určitý čas nějaký příkaz resp. proces (skript, program apod.). Jedná se vlastně o specializovaný systémový proces, který v operačním systému slouží jakožto plánovač úloh, jenž umožňuje opakované spouštění periodicky se opakujících procesů (např. noční běhy dávkových úloh při hromadném zpracování dat) apod.*“ (36)

Vytvoření CRONu se pak liší hosting od hostingu. Pokud máme přímý přístup k nastavení CRONu vypadá zápis nějak takto:

```
-  * * * * *  /cesta/ke/skriptu
```

Prvních pět hvězdiček definuje čas v následujícím pořadí (v závorkách jsou uvedeny povolené hodnoty):

1. Minuta (0–59)
2. Hodina (0–23)
3. Den v měsíci (1–31)
4. Měsíc (1–12)
5. Den v týdnu (0 = neděle, 1 = pondělí, …, 6 = sobota)

Tuto funkci poskytuje většina hostingů, ale je potřeba si dát pozor, zdali je poskytována i v neplacené verzi.

7.8.2. Zabezpečení

Vzhledem k tomu, že CRON vyžaduje cestu ke skriptu a není možné řešit oprávnění na úrovni hesla, je potřeba zajistit, aby nebyl skript spuštěn uživatelem. Proto je doporučováno vyžadovat nějaký parametr typu GET, který bude uživatelům neznámý. I já jsem zabezpečil skript tímto způsobem. Vyžaduji pro provedení skriptu určitý parametr, který když není splněný, tak se skript nezačne provádět.

7.9. Nákup budov

Nákup budov je v této aplikaci řešen pomocí klikatelné mapky, která je složena z obrázku a HTML tagu „map“. Této mapě jsou pak přiřazeny souřadnice, kde je mapka klikatelná a kam budou ony souřadnice odkazovat. Jedná se o prvek standardu HTML 5, proto zde byla dána v úvahu ještě druhá možnost a to použití oddílů (div), které budou napozicovány, tak aby tvořili celistvý obrázek. Nicméně tato metoda vyžaduje přesné rozřezání obrázku a následné napozicování, kvůli vytvoření odkazu na jednotlivých budovách. Vzhledem k náročnosti na pozicování a ne 100% kompatibilitě napříč všemi prohlížeči, byla zvolena možnost první a to použití tagu „map“.

7.9.1. Mapa

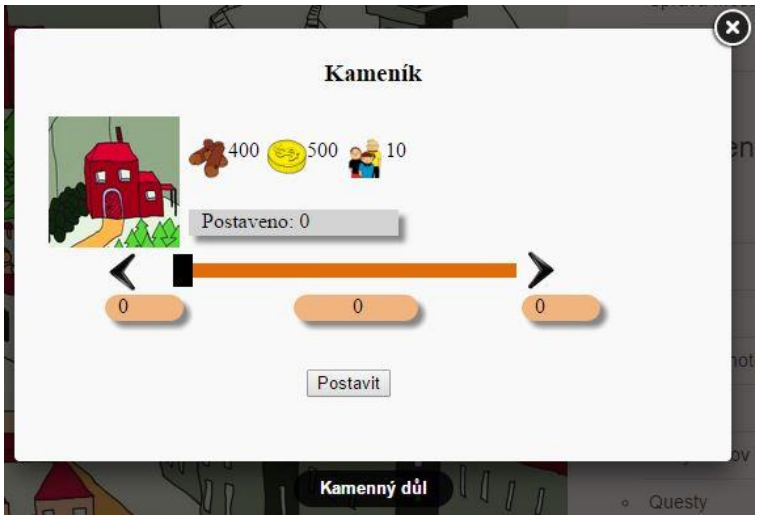
Na následujícím obrázku je zobrazeno použití HTML mapy. Nejdříve se vykreslí obrázek, který je s mapou spojen pomocí atributu „usemap“. Mapa pak obsahuje seznam odkazů se souřadnicemi, které vytyčují hranice odkazu. Pomocí atributu „shape“ volíme tvar ohraničení odkazu. Tvar, který budou hranice odkazu nabývat, můžeme volit z několika možností jako je kruh, obdélník či mnohoúhelník. V tomto případě byl zvolen jako tvar mnohoúhelník (poly). Ten je určen čtyřmi vrcholy. Každý z nich se skládá ze dvou hodnot a to z X-ové a Y-ové souřadnice.

```
<img src="/css/images/mesto.jpg" alt="mesto" usemap="#citimap" />
  <map name="citimap">
      <area shape="poly" coords="98,368,202,368,202,484,98,464" alt="Dřevorubec" title="Dřevorubec" href="zobraz/2">
      <area shape="poly" coords="230,207,309,207,309,298,230,298" alt="Železný důl" title="Železný důl" href="zobraz/5">
      <area shape="poly" coords="248,324,389,324,389,424,248,424" alt="Sklad" title="Sklad" href="zobraz/6">
      <area shape="poly" coords="329,207,487,207,487,298,329,298" alt="Kamenný důl" title="Kamenný důl" href="zobraz/3">
      <area shape="poly" coords="414,298,618,298,618,389,414,389" alt="Kasárna" title="Kasárna" href="zobraz/4">
      <area shape="poly" coords="414,389,5527,389,527,489,414,489" alt="Dům" title="Dům" href="zobraz/1">
  </map>
```

Obrázek 27 - Ukázka namapování souřadnic na obrázek (Zdroj: Vlastní)

7.9.2.Volba počtu budov

Jak již bylo zmíněné dříve, nákup budov je klíčový pro hraní hry. Budovy rozšiřují herní možnosti hráče, proto je vhodné, aby uživatel mohl stavět budovy ve větším počtu. Z tohoto důvodu byl zvolen systém, kdy uživatel má k dispozici táhlo (slider), kterým může nastavit počet budov, které bude stavět. Tento systém nám také částečně zamezí vkládání nesprávných, či dokonce zakázaných hodnot, jako při použití textového vstupu. Pro uživatele je tento systém výhodný zejména proto, že vidí maximální počet budov, které může postavit.



Obrázek 28 - Ukázka stavění budovy (Zdroj: Vlastní)

Na obrázku vidíme ukázku prostředí, kde uživatel kupuje nové budovy. Jak je z obrázku patrné, je použitý fancybox, který je schopen pomocí iframu zobrazit novou stránku bez nutnosti uživatele někam přesměrovávat. K vytvoření slideru bylo použito JQuery UI. Jedná se totiž o nejjednodušší způsob, jak tohoto prvku docílit. HTML 5 již také poskytuje slider, ten bohužel nebylo možné použít z důvodu nemožnosti dynamického přepisování hodnot maxima. Tento problém je popsán v následující kapitole popisující nákup jednotek.

Uživatel má tedy zvolený počet budov a odešle formulář.

7.9.3.Odeslání formuláře

Samotný proces stavění budov je rozdělen do několika kroků a je poměrně náročný. Je totiž nutné ověřit vstup od uživatele. To znamená znovu spočítat možné maximum pro postavení konkrétního typu budovy. Pokud je maximální hodnota poslaná uživatelem větší než maximum, které je možné postavit, je nutné vstup opravit. Tento krok, ač se může zdát zbytečný, je poměrně důležitý. Je nutné uživateli nevěřit

a předpokládat, že vstup nějak (úpravou HTML na jeho straně) pozměnil a odesílá hodnoty, které jsou větší než povolené.

Poté je vypočítána cena za všechny posílané budovy. Výsledná částka je pak následně odečtena ze skladu ve městě. Dále se musí upravit volný prostor pro stavění ve městě.

Posledním krokem je pak již jen uložení stavěných budov a přesměrování zpět na formulář.

7.10. Nákup jednotek

Nákup jednotek je řešen velice obdobným způsobem jako nákup budov. Ovšem nachází se zde pravděpodobně nejzajímavější část celé aplikace. Poměrně zásadní změnou je systém nákupu samotný. Zatímco u budov si nejdříve uživatel vybere konkrétní budovu, kterou pak rozšiřuje, zde má možnost nakupovat všechny jednotky najednou. To s sebou přineslo poměrně zajímavý problém.

7.10.1. Dynamická úprava maxima

Za předpokladu, že nabídneme uživateli při nákupu vybrat si z nějakého rozsahu, je nutné zajistit, aby byl rozsah vždy aktuální.

Při načtení stránky jsou pro každý typ jednotky vygenerovány maximální počty jednotek, které si uživatel vzhledem ke zdrojům, kterými disponuje, může dovolit. Pokud ale uživatel změní počet nakupovaných jednotek jednoho typu, je s velikou pravděpodobností možné, že si už nebude moci dovolit koupit stejný počet ostatních jednotek jako na začátku. Proto bylo nutné zajistit, aby po změně počtu nakupovaných jednotek došlo k přepočítání maximálních počtů ostatních jednotek.

Tento problém byl vyřešen pomocí vhodné kombinace JQuery slideru a AJAX dotazu na server. Proč zrovna tento slider a ne obyčejný, kterým disponuje HTML5? Potřeboval jsem docílit toho, aby po posunutí jedním sliderem se změnily ostatní slidery. Pro objasnění situace: Slider je definován minimální a maximální hodnotou a následně hodnotou, kterou uživatel zvolil z daného rozsahu. Tato hodnota zároveň určuje i pozici táhla na prvku. Takže pokud je změněno maximum, je nutné vzhledem k novému maximu překreslit prvek a pozici táhla.

Mé řešení pracuje v několika krocích. Nejdříve je vytvořen samotný formulář, kde každá jednotka obsahuje svojí tabulku s potřebnými informacemi a sliderem. Tomu je při vytváření nastaveno několik základních parametrů. Kód tohoto prvku je zobrazen na následujícím obrázku.

```
$('#7').slider({
            animate:true,value:0, min:0, max:0,step:1,
            slide: function(event, ui){ $('#7_t').html(ui.value);},
            change: function(event, ui){
                     $('#7_v').attr('value', ui.value);
                    fired(7);}
                });
```

Obrázek 29 - Ukázka kódu slideru (Zdroj: Vlastní)

Na příkladu je vidět, že se slider skládá ze dvou základních funkcí: „slide“ a „change“. Funkce slide má za úkol měnit polohu při hnutí s táhlem. Funkce change naopak zobrazuje uživateli hodnotu, kterou si zvolil a zároveň volá funkci „fired()“, ta se stará o přepočítání maxim všech sliderů.

Funkce fired je funkce volaná pokaždé při pohnutí s jakýmkoliv sliderem. Vše, co funkce potřebuje k přepočítání maxima, jsou hodnoty nastavené na všech sliderech, aktuální stav surovin ve skladu a cena jednotlivých typů jednotek. Nejdříve je proto potřebné poskytnout JavaScriptu datovou strukturu typu pole s informacemi o skladu. Vzhledem k tomu, že sklad je uložený v databázi, se kterou JavaScript neumí přímo spolupracovat, bylo nutné nejdříve data získat pomocí PHP spolu se SQL dotazem. Poté byla využita funkce, které je určená ke konverzi PHP pole na JavaScriptové pole. Jedná se o funkci „json_encode()“. Na následujícím obrázku je ukázáno, jak tato funkce pracuje.

```
<?php
  $arr = array('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5);
  echo json_encode($arr);
?>


Výše uvedený příklad vypíše:

{"a":1,"b":2,"c":3,"d":4,"e":5}
```

Obrázek 30 - Ukázka funkce json_encode() (Zdroj: http://php.net/manual/en/function.json-encode.php)

Stejným způsobem je vypsán i ceník jednotlivých jednotek. Algoritmus tak má dvě potřebné informace ze tří. Posledním krokem je projít veškeré slidery. K tomu je využita funkce „each“ pracující pod JQuery. Jedná se o ekvivalent PHP funkce „foreach“.

```
$(".data-slider").each(function () {
            var id = parseInt($(this).attr('id'));
            var typ = parseInt($("#" + id + "_typ").val());
            var poc = parseInt($(this).val());
            jQuery.each(cena[typ], function (surovina, pocet) {
                sklad[surovina] = sklad[surovina] - (pocet * poc);
            });
        });
```

Obrázek 31 - Přepočítání volných surovin (Zdroj: Vlastní)

Na předchozím obrázku je ukázka kódu z výše popisovaného algoritmu. Tato část postupně projde veškeré slidery a odečte příslušné suroviny ze skladu podle počtu jednotek nastaveného na slideru.

V této části již máme upravený sklad surovin a potřebujeme vrátit maximální počet surovin dle nového stavu ve skladu. Opět musíme projít každý slider a postupně přes dotaz typu POST odeslaného prostřednictvím AJAX. V dotazu jsou odeslány informace o skladu a typu jednotky. PHP funkce v controlleru již pak vypočítá nové maximum a vrátí ho zpět. Posledním krokem je pak už jen úprava hodnot jednotlivých prvků slideru. Nejdříve nastavení parametru „max“, který při změně upraví pozici táhla slideru a pak textový výpis pro uživatele. Při tomto návrhu ale ovšem nastal problém. Po přepočítání maxima algoritmus zároveň přepočítal maximum již nastavených prvků. Proto je nutné k maximu vždy přičíst onu nastavenou hodnotu, ovšem pokud je větší než nula. Ukázku kódu vidíme níže.

```
$(".data-slider").each(function () {
    var id = parseInt($(this).attr('id'));
    var typ = parseInt($("#" + id + "_typ").val());
    var poc = parseInt($(this).val());
    if (id !== fire) {
        $.post('jednotka/ajaxMaximum', {"i": typ, "sklad": sklad}, function (data) {
            max = data;
            if (poc > 0) {
                max = parseInt(poc) + parseInt(max);
            }
            $("#" + id).slider("option", "max", max);
            $("#" + id + "_m").text(max);
        });
    }
});
```

Obrázek 32 - Ukázka kódu, odesílajícího data k úpravě maxima slideru (Zdroj: Vlastní)

7.11. Systém útoků

I v této hře najdeme systém útoků. Jedná se o prvek, který přináší soutěživost a rivalitu mezi hráče, bez něj by tato hra fungovat nemohla. Ovšem i když se jedná o systém útoků, není stejný jako všechny ostatní, které můžeme najít běžně na internetu. Je rozdílný svým návrhem, kdy silnější hráč nemůže jen tak pro radost masakrovat své slabší spoluhráče. To je věc, která se ve většině hrách vůbec neřeší a spoléhá se na to, že hráč zainvestuje nemalé peníze, aby se mohl silnějším hráčům vůbec vyrovnat. Já jsem proto zavedl pro každého hráče skóre, díky kterému bude možné jednoduše hráče porovnávat.

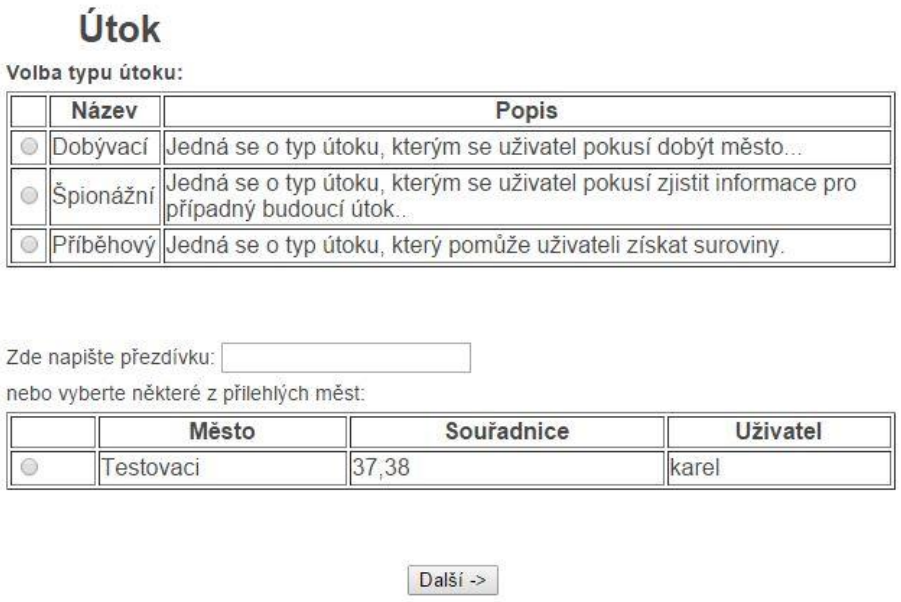
Útoky jsou rozděleny do tří typů. Dobývací, kde hráč pouze útočí na soupeřovu armádu. Špionážním útokem může zjistit protivníkovu sílu. Posledním typem je příběhový útok, díky kterému je možné získat suroviny. Tento útok je však omezen na jeden útok denně.

Poměrně zajímavou funkcí je vliv počasí na útočící armádu. Jedná se o něco, co nikdo do hry ještě neimplementoval a přináší to tak poměrně zajímavý prvek nahodilosti. Počasí bude negativně ovlivňovat morálku útočících jednotek, a tak i sílu

jejich útoků. Počasí bude naprosto reálné, protože bude stahováno prostřednictvím API (Aplication Programing Interface) meteorologického ústavu.

Prozatím je počasí sledováno pouze v jednom místě, do budoucna se tu nabízí rozšíření o lokalizování hráčovy IP adresy a kontrolování tak počasí přímo v místě jeho pobytu. Uživatel tak bude nucen rozmýšlet útoky, protože i se silnější armádou se díky nepříznivému vlivu počasí dá prohrát nad slabším soupeřem.

Všechny tři útoky se skládají ze tří hlavních kroků. Prvním krokem je výběr útoku a výběr města, na které bude hráč útočit. Město buď může vybrat z předem vygenerované nabídky nejbližších měst. Tento seznam je získán na základě souřadnic města. Nebo uživatel může zadat jméno uživatele a vybrat některé z jeho měst. Zde je malá výjimka pro příběhový útok. Po zvolení možnosti „Příběhový útok“ je funkce zvolení města zablokována a uživatel tak nevolí město, na které bude útočit. Tento formulář je zobrazen na následujícím obrázku.



Po odeslání formuláře je uživateli zobrazen přehled jeho armády. Zde uživatel zvolí počet a typy jednotek, které pošle do souboje s protivníkem. Bylo nutno ošetřit vstupy, aby uživatel nemohl zadat jiné počty jednotek než vlastní, případně aby nezadal nějaké nedovolené hodnoty.

Poslední krok je nejdůležitějším a zároveň nejsložitějším na celém soubojovém systému. Zároveň je pro každý typ útoku jiný.

7.11.1. Dobývací útok

Nejdříve jsou načteny vlastnosti jednotek, aby bylo možné s nimi dále pracovat. Dále jsou zjištěny parametry síly a obrany jednotek jak pro útočníka, tak pro obránce. Útočícímu hráči je síla a obrana počítána z jednotek poslaných na souboj. Zatímco obránce automaticky bojuje se všemi jednotkami, které jsou ve městě k dispozici.

Zvláštním případem je situace, kdy obránce nedisponuje žádnými jednotkami. Útočník tak ve skutečnosti neví, jaká síla proti němu bude stát, tedy pokud vhodně nevyužije špionážní útok.

Dalším krokem je porovnáním síly a obrany. Porovnává se síla útočníka proti obraně obránce. Můžou nám tak nastat tři výsledky. Vítězství neboli situace, kdy útočník disponuje větší silou než je obránce schopen ubránit. Prohra, kdy je situace opačná než v předchozím případě. Posledním případem je remíza, kdy jsou síly obou stran vyrovnány. Na základě vzniklé situace je počítán koeficient mrtvých jednotek.

Výpočet koeficientu pro mrtvé jednotky je zobrazen na následujícím obrázku.

```
private function vypocetKoefMrtvych($vlast1, $vlast2) {
        $koefMrtvych = 0;
        if ($vlast2 == 0) {
            return 1;
        }
        $koefMrtvych = ($vlast1 / $vlast2) * (sqrt($vlast1 / $vlast2));
        return $koefMrtvych;
    }
```

Obrázek 34 - Výpočet koeficientu mrtvých jednotek
(Zdroj: http://forum.divokekmeny.cz/showthread.php?9514-V%C3%BDpo%C4%8Detn%C3%AD-vzorec)

Je potřeba zajistit, aby první vlastnost ($vlast1) byla větší než druhá vlastnost ($vlast2). Pokud je druhá vlastnost rovna nule, je potřeba zajistit, aby byla nenulovaná kvůli nemožnosti dělení nulou. Tato situace nastává, když jeden z uživatelů nemá jednotky. Proto se výsledná síla rovná nule. Samotný koeficient je pak součinem poměrů sil mezi sebou a odmocniny z onoho poměru. Tímto koeficientem jsou pak vynásobeny jednotky vítězné strany. Vznikne nám tak pro každý typ jednotky počet zabitých jednotek. Strana, která souboj prohrála, automaticky přichází o veškeré jednotky.

Výsledek je následně zaznamenán do databáze včetně informace o zúčastněných jednotkách. Jsou zaznamenány jednotky útočníka, jeho ztráty a ztráty obránce. Obránci je odeslána informační zpráva o průběhu souboje. Poté je útočník přesměrován na zobrazení bitevní zprávy, kde se dozví výsledek souboje.

7.11.2. Špionážní útok

Špionážní útok je v mnohém velice podobný jako dobývací útok. Využívá i obdobný vzorec, který určuje, zdali byl útok úspěšný či nikoliv. V tomto útoku řeším zatím pouze dva stavy, které mohou nastat. Buď byl špionážní útok úspěšný či nikoliv. Pokud je útok úspěšný útočník se dozví informace o stavu protivníkovy armády, zatímco protivník o ničem neví. Tento útok se mu nezobrazí ve výpisu a ani mu nepřijde žádná zpráva o útoku. Zatímco při opačném výsledku je útočníkův špion zabit a obránce je o útoku plně informován. Zde se nabízí možnost k dalšímu rozšíření v podobě částečného odhalení soupeřových jednotek a zároveň částečnému zabití špionů.

Špionážní útok se tak stává nástrojem, kterým může útočník využít ke zjištění podstatných informací o soupeřově armádě před samotným útokem.

7.11.3. Příběhový útok/úkol

Posledním útokem, který webová aplikace nabízí, je příběhový útok. Jedná se o speciální typ útoku, kdy hráč neutočí proti živému protivníkovi, ale bojuje proti předem vytvořenému imaginárnímu nepříteli.

V případě volby pro příběhový útok dojde k načtení příběhových úkolů a seznamu již splněných úkolů. Tyto seznamy se porovnají a úkoly jsou následně vyfiltrovány, uživatel totiž bude mít možnost splnit úkol pouze jednou. Seznam příběhových úkolů je následně promíchán tak, aby byl jeden úkol náhodně zvolen a zobrazen uživateli. Ten se na základě popisu úkolu rozhodne, kolik jednotek pošle. Uživatel dopředu nikdy netuší, jakou silou musí zaútočit, aby úkol splnil. V případě výhry jsou uživateli odečteny jednotky dle koeficientu uvedeného výše. Dále mu je přičtena odměna za splnění úkolu. V databázi je uložen záznam o splnění úkolu. Úspěšné splnění úkolu znamená, že se tento úkol již uživateli nezobrazí. Úkoly jsou vázány na město, proto uživatel, který má více měst, může jeden úkol splnit vícekrát (s každým městem jednou).

Z důvodu poměrně vysoké odměny a jiných bonusů za splnění úkolu je tento útok omezen. Je možné provést pouze jeden útok tohoto typu denně.

8. Optimalizace

Tato část práce se bude věnovat zejména optimalizaci aplikace jako takové. Porovná pár rozdílů, pokud budete používat čisté PHP oproti PHP frameworku. A hlavně jaké jsou možné způsoby optimalizaci aplikace, aby byla rychlejší, stabilnější a méně náročná pro server.

Při optimalizaci webových aplikací se zapomíná na drobné, přesto ale podstatné věci, které nám mohou ušetřit mnoho starostí. Optimalizovaný web je většinou rychlý web. Z praxe je zřejmé, že uživatelé se raději vrací na web, který má bleskovou odezvu a nemusí dlouho čekat na provedení všech jejich požadavků. Při správné optimalizaci webu by se mělo dbát i na správné pořadí načítání prvků. Nejdříve by měly být načteny CSS a pak teprve JavaScript. Pokud budeme nahrávat soubory střídavě (CSS, JS, CSS, JS …), aplikace bude mít velké načítací prodlevy. (37)

Optimalizace webové aplikace byla shrnuta do několika bodů tak, jak bylo postupováno při optimalizaci této webové aplikace.

8.1. Optimalizace JavaScriptu

Základním kamenem pro optimalizaci JavaScriptu je psát co nejjednodušší, minimalizovaný kód. Je nanejvýše vhodné se vyvarovat častého opakování například výpočtů, případně hledání toho samého objektu v dokumentu. Pokud tedy v algoritmu využíváme kód, který můžeme vidět níže, je to špatně.

- $(this).text($(this).text() + 'nějaký text');

V tomto případě dochází k vytvoření JQuery objektu 2x a pokaždé je to jeden a ten samý. Proto je vhodné kód poupravit tak, jak to vidíme níže.

- var e = $(this);
  e.text(e.text() + 'nějaký text');

Toto řešení vytvoří JQuery objekt pouze jednou a zavede na něj referenci v podobě proměnné. Pak se na tento objekt jen a pouze odkazujeme. To šetří jak čas, tak i serverové prostředky, vynaložené na práci s JavaScriptem.

Další poměrně důležitou věcí je používat co možná nejminimalizovanější kód. Je dobré využít různých nástrojů, které jsou plně k dispozici zadarmo na internetu a to i bez nutnosti instalace. Tyto nástroje nám kód optimalizují (nahradí dlouhé názvy proměnných za krátké), odstraní bílé znaky (mezery) a případně i komentáře. Pro vývoj aplikace je pohodlnější mít dvě verze jednu minimalizovanou pro ostré použití a druhou v původním přehledném formátu pro další vývoj.

Obecně je zastáván názor, že pro načítání dalších knihoven je nejlepší využívat CDN (Central Delivery Network) serverů. Jedná se o servery, které distribuují obsah místo vašeho serveru. Mimo jiné se využívají k uložení obsahu stránek knihoven do cache paměti, ze které je načtení stránky mnohem rychlejší. Pro načítání JQuery knihovny je tento systém ještě opodstatněnější vzhledem k faktu, že i prohlížeč uchovává ve své mezipaměti informace o různých podpůrných knihovnách. Pokud tedy zjistí, že onu JQuery knihovnu z tohoto CDN serveru již uloženou v mezipaměti máte, pak není potřeba ji znovu stahovat. Pokud by ale byla ta samá knihovna uložena na vašem serveru, tak ji prohlížeč musí znovu stáhnout. Tím se šetří zejména náklady na datový provoz a stránky se načítají rychleji. (39)

8.2. Optimalizace CSS

U kaskádových stylů platí podobná pravidla jako u JavaScriptu. Je nutné dbát na co nejčistší kód, u kterého je následně vhodné provést minimalizaci pomocí odstranění komentářů a prázdných znaků.

U CSS by se mělo zamezit načítání dalších CSS pomocí klauzule @import. Pokud je v CSS umístěna klauzule @import, je vytvořen nový požadavek na server (stažení nového CSS souboru), který zastaví zpracování všech ostatních. Web tedy čeká na načtení importovaných CSS, což prodlouží načítání webu. Společnost Google doporučuje CSS soubory, pokud se tedy nejedná o kritická CSS, načítat asynchronně, neboli načítat je až po načtení webu samotného. Také je výrazně doporučeno vyhýbat se obecným selektorům (*).

8.3. Optimalizace obrázků

Optimalizací obrázků na webu zajistíte snížení obsahu přenášených dat. Vždy je vhodné při tvorbě webu přemýšlet, na co bude daný obrázek určen. Pokud se bude jednat o statickou grafiku, která je součástí designu webu, je vhodně využít formáty „gif“ či „png“, které jsou k tomu určené. Je také dobré přemýšlet nad vhodnou kompresí, aby byly grafické prvky co nejmenší.

Další chybou, která se často objevuje, je ta, že se používají velké obrázky ve formátu „jpeg“ a následně jsou zmenšovány pomocí HTML atributů. Toto je situace, která by se nikdy neměla stát. Atributy pro obrázky by ale měly být vždy vyplněny, aby při načítání neskákal obraz.

Pokud je na webu použita nějaká drobná grafika, které se často opakuje, je vhodné využít CSS Sprites. To znamená, že se veškeré grafické prvky sloučí do jednoho obrázku a je z něj vytvořena jakási mapa. Na načtení všech grafických prvků tak zabere pouhý jeden dotaz, což ušetří čas potřebný k vykreslení stránky.

8.4. Cache a komprimace

Nyní máme naší webovou aplikaci plně optimalizovanou, ale stále je možné její výkon vylepšit. Často se jako jeden z optimalizačních prvků používá právě cache neboli uložení do mezipaměti. V tomto případě mluvíme o uložení do mezipaměti webového prohlížeče. To znamená, že pokud uživatel načítá naší stránku podruhé, je vysoká pravděpodobnost, že se mu stránka načte několikanásobně rychleji. Tato metoda poskytuje nejen výhodu rychlejšího načtení webové stránky uživateli, ale i nám v podobě menšího počtu HTTP požadavků a sníženého počtu přenesených dat.

Celý princip spočívá v tom, že server a prohlížeč kontrolují HTTP hlavičky požadavků. Ověřují datum platnosti cache a kontrolní součet. Ten pokud se změní je cache zahozena a stránka (soubor) je stažen znovu.

Celá situace se dá sledovat pomocí stavových kódů, které najdeme v hlavičce. Pokud budou všechny hlavičky platné, odpověď pro data bude stav „304 Not Modified“ a data se načtou z mezipaměti. Pokud by se ale soubor změnil a některá z hlaviček by se vyhodnotila jako neplatná, odpovědí bude stav „200 OK“ a soubor se opět stáhne ze serveru. (40)

Ukázka stavových kódů při načítání webové aplikace je na následujícím obrázku.

Obrázek 35 - Ukázka Cache-ování stránek prohlížečem (Zdroj: Vlastní)

Aby byly hlavičky správně modifikovány a byly vynuceny k uložení do mezipaměti, je nutné mít povoleny moduly „mod_header“ a „mod_expires“ na webovém serveru Apache2. Pak už stačí umístit vhodný kód do souboru „.htaccess“.

```
<IfModule mod_expires.c>
 ### aktivace modulu mod_expires
 ExpiresActive On

 <FilesMatch "\.(ico|pdf|flv|jpg|jpeg|png|gif|js|css|swf)$">
 Header set Cache-Control "max-age= 2592000, public"
 </FilesMatch>
</IfModule>
```

Obrázek 36 - Ukázka kódu pro aktivaci prohlížečové cache (Zdroj: Vlastní)

Výše uvedený kód zajistí, že všechna statická data, např. obrázky, JavaScriptové a CSS soubory budou uloženy po dobu 30 dní v mezipaměti.

Další možností pro snížení velikosti přenášených dat je metoda komprimování přenášených dat. Komprimace obsahu se projeví zejména ve snížení obsahu přenášených dat. Daná stránka je pak po této operaci menší i o několik desítek KB. Metoda GZip je efektivní zejména při komprimaci textový souborů (HTML, JS, TXT, JSON, AJAX, …), pro špatně optimalizované obrázky je neúčinná. Zapnutí komprese na serveru je opět otázkou správného nastavení Apache. Je nutné mít aktivní modul „mod_deflate“. Poté již stačí aktivovat kompresi dat v souboru „.htaccess“.

```
<IfModule mod_deflate.c>
 AddOutputFilterByType DEFLATE text/html text/plain text/xml text/css application/x-javascript
                               text/javascript application/javascript application/json
</IfModule>
```

Obrázek 37 - Ukázka kódu pro komprimaci dat (Zdroj: Vlastní)

Výše uvedený příklad nám zajistí komprimaci obsahu pro konkrétní typy souborů.

8.5. Optimalizace SQL

Ve výše uvedených podkapitolách jsme vyřešili největší chyby týkající se načítání webové aplikace a také snížení nákladů na její načtení. Optimalizace SQL je tu uvedena také, ale na tuto optimalizaci by mělo být přihlíženo už při samotném vytváření databáze a posléze i dotazu. Pokud budeme mít sebe komprimovanější webovou aplikaci, která je schopná se uživateli načíst v rámci několika jednotek milisekund, ale obsah aplikace nám generuje SQL dotaz z databáze, který je špatně napsaný, pak se načítání stránky může prodloužit i o stovky procent. Proto je nutné psát SQL dotazy jednoduše a účinně. To má za následek snížení nákladů na server.

„*Existují obecná pravidla pro psaní dotazů tak aby byl dotaz co nejefektivnější:*

- *Vyjmenovat sloupce*
- *Používat co nejméně klauzule LIKE, IN, NOT IN*
- *Vhodně využívat klauzuli LIMIT*
- *Na začátek dávat obecnější podmínky*
- *Vybírat vhodné pořadí pro spojení*
- *Používat HINTy a Indexy*“ (42)

Psaní SQL dotazů v Yii má specifický tvar. Na následujícím obrázku vidíme, jak vypadá klasický SQL dotaz v čistém PHP při použití metody PDO.

```
$DB = new PDO("mysql:host=localhost;dbname=testDB", "root", "heslo" );
  $query = $DB->prepare("SELECT * FROM `uzivatel` WHERE id = ? ");
  $query->bindParam(1, $id, PDO::PARAM_INT);
  $query->execute();
  $vysledek = $query->fetch();
```

Obrázek 38 - Ukázka SQL dotazu pomocí PDO Zdroj: Vlastní

Zatímco práce s databází probíhá v Yii úplně jiným způsobem. Práce je jednodušší a není potřeba ošetřovat vstupy, protože to řeší Yii za nás. Ten samý dotaz s tím samým výsledkem nám poskytne následující kód.

```
//Pokud je id primárním klíčem pro tabulku
$model = Uzivatel::model()->findByPk($id);
$vysledek = $model->attributes;

//Pokud není id primárním klíčem pro tabulku
$model = Uzivatel::model()->findByAttributes(array('id' => $id));
$vysledek = $model->attributes;
```

Obrázek 39 - Ukázka SQL dotazu v Yii (Zdroj: Vlastní)

Jak je možné na ukázce vidět, je jasné, že psaní dotazů s pomocí frameworku Yii je rychlejší a jednodušší než v běžném čistém PHP.

Při optimalizaci webové aplikace se neobejdete bez vhodného nástroje, díky kterému uvidíte rychlostní rozdíl před a po optimalizaci. Každý prohlížeč disponuje vlastním prostředím pro kontrolu obsahu. Například webový prohlížeč Google Chrome disponuje režimem pro vývojáře, ve kterém najdete veškeré potřebné informace. Oproti tomu Mozzila Firefox ve svém obchodě nabízí, volně ke stažení, legendární plugin Firebug, který se vyrovnává onomu zmíněnému režimu pro vývojáře pro Google Chrome. Dalším skvělým pomocníkem jsou pak nástroje od Googlu. Při optimalizaci webu nejvíce využijete nástroje pro vývojáře dostupné z adresy: https://developers.google.com/speed, které doplňují funkci vestavěnou do prohlížeče.

9. Závěr

Na začátku práce byla myšlenka vytvoření textové webové hry, která nebude trpět dnešními herními neduhy. Byly stanoveny jak funkční tak non-funkční požadavky, které hru jako takovou definovaly. Zásadní podmínkou byla ta, že aplikace musí být vystavena na MVC frameworku. V první části, teoretické, je popsán samotný princip MVC, jeho výhody a nevýhody. Poté byl zvolen konkrétní framework, pomocí kterého bude aplikace postavena. V této části je popsáno, jak jsem framework vybíral a na základě jakých parametrů jsem se rozhodl pro vítězný Yii Framework.

Dále navazovala třetí praktická, část práce, ve které ukazuji vývoj celé aplikace. Nejprve je popsán návrh aplikace, který vychází právě z funkčních požadavků. Během

vytváření aplikace se ukázaly značné výhody práce s tímto frameworkem, který usnadňoval práci a zvyšoval bezpečnost oproti srovnání s čistým PHP. Většinu věcí řeší přímo Yii, a proto tak nebylo nutné vytvářet vše od úplného začátku. Vzhledem k množství věcí, které je potřeba spravovat v administračním módu, se ukázal Gii jako nepostradatelný nástroj. Jeho možnosti vytvoření CRUD modelu byly místy až neuvěřitelné a práci opravdu velice usnadňoval.

9.1. Shrnutí

Hlavním cílem této práce bylo vybrat, dle zadaných kritérií, z nepřeberného množství frameworků jeden a pomocí něho vytvořit jednoduchou webovou hru. Tato hra měla být oproštěna od dnešních herních neduhů, které číhají v každé nové on-line hře.

Při volbě toho „nejlepšího“ frameworku jsem narazil zejména na to, že většina jich je výborná a záleží pouze na preferencích programátora. Proto jsem se rozhodl vybrat, v Čechách poměrně neznámý, Yii Framework a rozšířit tak obzory zájemců o další prostředek k vytváření lepších a bezpečnějších webových aplikací. Z tohoto důvodu jsou veškeré zdrojové kódy této aplikace dostupné na přiloženém CD, tak aby zájemci o tento framework mohli nahlédnout na jeho možnosti a vlastnosti.

Co se týče splnění hlavního cíle, dle mého názoru byl z velké části splněn. Ovšem narazil jsem při vývoji na velice omezené možnosti free hostingových služeb, které nedokázali poskytnout ty pravé prostředky pro plnou funkčnost této aplikace. Při samotném programování této webové aplikace mě napadlo mnoho dalších možných rozšíření, takže je možné, že vývoj této hry bude nejspíše i nadále pokračovat.

10. Zdroje

[1] PARC. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2015 [cit. 2014-11-12]. Dostupné z: http://en.wikipedia.org/wiki/PARC_%28company%29

[2] Smalltalk. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2015 [cit. 2014-11-12]. Dostupné z: http://cs.wikipedia.org/wiki/Smalltalk

[3] Model View Controller History. In: *Model View Controller History* [online]. 2014-12-26 [cit. 2014-11-13]. Dostupné z: http://c2.com/cgi/wiki?ModelViewControllerHistory

[4] Model-view-controller. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2015 [cit. 2015-11-15]. Dostupné z: http://cs.wikipedia.org/wiki/Model-view-controller

[5] MVC aplikace & presentery. In: *Nette* [online]. 2008-2015 [cit. 2014-11-15]. Dostupné z: http://doc.nette.org/cs/2.2/presenters

[6] Úvod do architektury MVC. *Úvod do architektury MVC* [online]. 2009-05-07, č. 1 [cit. 2014-11-15]. Dostupné z: http://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/

[7] BERNARD, Borek. Prezentační vzory z rodiny MVC. *Úvod do architektury MVC* [online]. 2009-05-11, č. 2 [cit. 2014-11-15]. Dostupné z: http://www.zdrojak.cz/clanky/prezentacni-vzory-zrodiny-mvc/

[8] Yii. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2015 [cit. 2014-11-20]. Dostupné z: http://en.wikipedia.org/wiki/Yii

[9] Nette Framework. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2015 [cit. 2014-12-10]. Dostupné z: http://cs.wikipedia.org/wiki/Nette_Framework

[10] *Nette features* [online]. 2008-2015 [cit. 2014-12-10]. Dostupné z: http://nette.org/cs/#toc-features

[11] Cross-site scripting. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2015 [cit. 2014-12-10]. Dostupné z: http://cs.wikipedia.org/wiki/Cross-site_scripting

[12] Cross-site request forgery. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2015 [cit. 2014-12-10]. Dostupné z: http://cs.wikipedia.org/wiki/Cross-site_request_forgery

[13] *Yii Security* [online]. 2008-2015 [cit. 2014-12-11]. Dostupné z: http://www.yiiframework.com/doc/guide/1.1/en/topics.security

[14] *Zabezpečení před zranitelnostmi* [online]. 2008-2015 [cit. 2014-12-11]. Dostupné z: http://doc.nette.org/cs/2.2/vulnerability-protection

[15] FOWLER, Martin. *Supervising Controller* [online]. 2006-06-19 [cit. 2014-12-17]. Dostupné z: http://martinfowler.com/eaaDev/SupervisingPresenter.html

[16] Is MVC and MVP supervising controller the same?. In: *Codeguru* [online]. 2011-10-23 [cit. 2014-12-17]. Dostupné z: http://forums.codeguru.com/showthread.php?517594-Is-MVC-and-MVP-supervising-controller-the-same

[17] 20 Best PHP Frameworks for Developers in 2014. DRUMELIS, VYTAUTAS. *Codegeekz.com* [online]. 2014-08-19 [cit. 2014-12-17]. Dostupné z: http://codegeekz.com/20-best-php-frameworks-developers-august-2014/

[18] Performance benchmark of popular PHP frameworks. KUJAWA, Lukasz. *Systemsarchitect.net* [online]. 2013-08-23 [cit. 2014-12-17]. Dostupné z: http://systemsarchitect.net/performance-benchmark-of-popular-php-frameworks/

[19] *About Yii* [online]. 2008-2015 [cit. 2015-01-07]. Dostupné z: http://www.yiiframework.com/about/

[20] Internacionalizace a lokalizace. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2015 [cit. 2015-01-08]. Dostupné z: http://cs.wikipedia.org/wiki/Internacionalizace_a_lokalizace

[21] *Features of Yii* [online]. 2008-2015 [cit. 2015-01-09]. Dostupné z: http://www.yiiframework.com/features/

[22] WINESETT, Jeffrey. *Agile web application development with Yii1.1 and PHP5: fast-track your web application development by harnessing the power of the Yii PHP framework*. Birmingham: Packt Publishing, 2010, s. 11. ISBN 978-1-847199-58-4.

[23] HOLEC, MIROSLAV. MVC Routing. *Miroslavholec.cz* [online]. 2014-09-08 [cit. 2015-01-15]. Dostupné z: https://www.miroslavholec.cz/blog/mvc-routing-uvod

[24] Routování URL. *Nette.org* [online]. 2008-2015 [cit. 2015-01-15]. Dostupné z: http://doc.nette.org/cs/2.3/routing

[25] *Cool URL / SEO URL* [online]. 2012 [cit. 2015-01-15]. Dostupné z: http://www.seo-expert.cz/cool-url-seo-url

[26] *CUserIdentity* [online]. 2008-2015 [cit. 2015-02-04]. Dostupné z: http://www.yiiframework.com/doc/api/1.1/CUserIdentity

[27] Message-Digest algorithm. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2015 [cit. 2015-02-04]. Dostupné z: http://cs.wikipedia.org/wiki/Message-Digest_algorithm

[28] Kolize (informatika). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2015 [cit. 2015-02-04]. Dostupné z: http://cs.wikipedia.org/wiki/Kolize_(informatika)

[29] Zive.cz *Jak ukládat hesla do databáze* [online]. 2010-01-06 [cit. 2015-02-04]. Dostupné z: http://miho.blog.zive.cz/2010/01/jak-ukladat-hesla-do-databaze/

[30] Secure Hash Algorithm. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2015 [cit. 2015-02-10]. Dostupné z: http://cs.wikipedia.org/wiki/Secure_Hash_Algorithm

[31] Bcrypt. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2015 [cit. 2015-02-12]. Dostupné z: http://en.wikipedia.org/wiki/Bcrypt

[32] Usenix.org. PROVOS, Niels a David MAZIERES. *Bcrypt Algorithm* [online]. 2010-01-06 [cit. 1999-04-28]. Dostupné z: https://www.usenix.org/legacy/events/usenix99/provos/provos_html/node5.html

[33] How to manage a PHP application's users and passwords. PESLYAK, Alexander. *Php-security.org* [online]. 2005-05-26 [cit. 2015-02-14]. Dostupné z: http://php-security.org/2010/05/26/mops-submission-10-how-to-manage-a-php-applications-users-and-passwords/

[34] Několik poznámek k heslům. MALÝ, Martin. *Zdrojak.cz* [online]. 2011-07-20 [cit. 2015-02-14]. Dostupné z: http://www.zdrojak.cz/clanky/nekolik-poznamek-k-heslum/

[35] Features. *Fancybox.net* [online]. 2008-2015 [cit. 2015-02-15]. Dostupné z: http://fancybox.net/

[36] Cron. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2015 [cit. 2015-02-15]. Dostupné z: http://cs.wikipedia.org/wiki/Cron

[37] Jak můžeme zrychlit své webové aplikace?. MAŇÁK, Michal. *Manakmichal.cz* [online]. 2010-11-15 [cit. 2015-02-16]. Dostupné z: http://www.manakmichal.cz/blog/optimalizace/jak-muzeme-zrychlit-sve-webove-aplikace-4-dil/

[38] Optimalizácia JavaScript a jQuery kódu. AMBRUŠ, Martin. *Interval.cz* [online]. 2011-02-17 [cit. 2015-02-16]. Dostupné z: https://www.interval.cz/clanky/optimalizacia-javascript-a-jquery-kodu/

[39] OVH CDN. *Ovh.cz* [online]. 1999-2015 [cit. 2015-02-16]. Dostupné z: https://www.ovh.cz/cdn/vyhody.xml

[40] Jak můžeme zrychlit své webové aplikace?. In: MAŇÁK, Michal. *Manakmichal.cz* [online]. 2010-10-19 [cit. 2015-02-16]. Dostupné z: http://www.manakmichal.cz/blog/optimalizace/jak-muzeme-zrychlit-sve-webove-aplikace-2-dil/

[41] Kešovací návod. JANOVSKÝ, Dušan. *Jakpsatweb.cz* [online]. 2004 [cit. 2015-02-16]. Dostupné z: http://www.jakpsatweb.cz/clanky/caching-tutorial-czech-translation.html

[42] DUŠEK, Roman. *Optimalizace SQL dotazl*. 2010. Dostupné z: http://www.fi.muni.cz/~kripac/PV136/dusek/opt_sql_dot.pdf

[43] Výkonnostní optimalizace front-endu web integračního projektu. In: STANĚK, Zdeněk. *Web-integration.info* [online]. 2014-06-16 [cit. 2015-02-20]. Dostupné z: http://www.web-integration.info/cs/blog/vykonnostni-optimalizace-front-endu-web-integracniho-projektu/fef

[44] BSD licence. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-2015 [cit. 2015-01-05]. Dostupné z: http://cs.wikipedia.org/wiki/BSD_licence

[45] Session hijacking. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-2015 [cit. 2014-12-11]. Dostupné z: http://en.wikipedia.org/wiki/Session_hijacking

11. Seznam obrázků

12. Seznam příloh

Webová aplikace na nosiči CD