VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

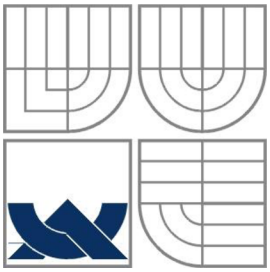
KOMPLEXNÍ WEBOVÝ HERNÍ SYSTÉM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

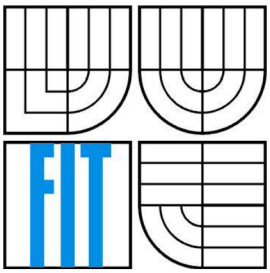
AUTOR PRÁCE
AUTHOR

Bc. VOJTĚCH KOLÁČEK

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

KOMPLEXNÍ WEBOVÝ HERNÍ SYSTÉM

COMPLEX WEB GAME SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. VOJTĚCH KOLÁČEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ZBYNĚK KŘIVKA, PhD.

BRNO 2011

Abstrakt

Tato práce se zabývá analýzou technologií pro moderní webová uživatelská rozhraní a návrhem a implementací komplexního webového systému s důrazem na ergonomické a robustní grafické uživatelské rozhraní. U technologií pro webová uživatelská rozhraní (Ajax, Adobe Flex a Microsoft Silverlight) jsou provedeny srovnávací testy, které testují propustnost, rychlost reakce, rychlost zpracování a zobrazení dat jednotlivých technologií. Webový systém je navržen a implementován jako Ajax strategická hra, která na klientské straně využívá JavaScriptové knihovny YUI. Na serverové straně poskytuje klientovi služby webový server implementovaný v jazyce PHP a komunikační server implementovaný v jazyce C++ umožňuje komunikaci s klientem v reálném čase pomocí techniky long polling. Na implementovaném systému jsou provedeny laboratorní testy propustnosti a rychlosti odezvy uživatelského rozhraní, které prověřují výkonnost a kvalitu systému. Na závěr jsou tyto testy vyhodnoceny a jsou zhodnoceny výhody, nevýhody a nedostatky implementace včetně návrhů na vylepšení.

Abstract

This Master's Thesis deals with the analysis of technologies for modern web-based user interfaces, and with the design and implementation of a complex web system with focus on an ergonomic and robust graphical user interface. A set of comparison tests is performed on technologies for web-based user interfaces (Ajax, Adobe Flex, and Microsoft Silverlight); throughput, response time, processing and rendering time of each technology is tested. The web system is designed and implemented as an Ajax strategy game which utilizes the YUI JavaScript library on the client side. The server side comprises of a web server implemented in PHP which offers services to the client, and a message server implemented in C++ that facilitates communication with the client in real-time using HTTP long polling. Tests measuring throughput and response time of the user interface analyze the performance and quality of the implemented system. In the end the tests are analyzed, and the advantages, disadvantages, and flaws of the implementation are evaluated.

Klíčová slova

Ajax, informační systém, C++, MySQL, PHP, RIA, YUI

Keywords

Ajax, information system, C++, MySQL, PHP, RIA, YUI

Citace

Koláček Vojtěch: Komplexní webový herní systém, diplomová práce, Brno, FIT VUT v Brně, 2011

Komplexní webový herní systém

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Zbyňka Křivky, PhD.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Vojtěch Koláček
20. 5. 2011

Poděkování

Rád bych poděkoval Ing. Zbyňku Křivkovi, PhD. za odbornou pomoc a vedení při této práci.

Obsah

1 Úvod

V posledních letech dochází stále častěji k využívání webových aplikací (běžících ve webovém prohlížeči) namísto desktopových aplikací (běžících přímo jako proces v operačním systému). Uživatelské rozhraní tvořené statickým HTML kódem stále častěji přestává stačit náročným požadavkům webových aplikací, a proto bývá nahrazováno novými technologiemi, jako jsou Ajax, Adobe Flash nebo Flex a Microsoft Silverlight. Každá z těchto technologií však má své silné a slabé stránky, které je třeba při rozhodování o použití jednotlivých technologií brát v úvahu.

V první části této práce se soustředím na technologie pro moderní webová uživatelská rozhraní hlavně z pohledu jejich využití v systémech s náročnými požadavky na propustnost a rychlost reakce. Schopnosti, rychlost a výkonnost těchto technologií jsou kvantifikovány pomocí sady srovnávacích testů. Dále popíši trendy moderních webových her a navrhnu komplexní webový systém s důrazem na ergonomické a robustní grafické uživatelské rozhraní. Systém poté v diplomové práci implementuji, provedu testy propustnosti a rychlosti odezvy uživatelského rozhraní a testy vyhodnotím. Na závěr zhodnotím implementaci systému a navrhnu další vylepšení.

Ve druhé kapitole se seznámíme výhodami, nevýhodami, technologiemi a požadavky na moderní webová uživatelská rozhraní. V další kapitole srovnáme nejpoužívanější technologie pro tvorbu moderních uživatelských rozhraní (Ajax, Adobe Flex a Microsoft Silverlight) z hlediska propustnosti, rychlosti reakce, rychlosti zpracování a zobrazení dat. Podle výsledků jsou identifikovány výhody a nevýhody nasazení testovaných technologií.

V další kapitole jsou komentovány technologie moderních webových her a s pomocí zkušeností z předchozích kapitol je navržen komplexní webový systém v podobě strategické hry. Dále je popsán princip hry a je navržena základní struktura komponent, ze kterých se systém skládá. Blíže je popsána klientská část včetně grafického uživatelského rozhraní a serverová část skládající se z webového, databázového a komunikačního serveru. Nakonec je popsán protokol zajišťující komunikaci v reálném čase mezi klientem a komunikačním serverem.

Pátá kapitola popisuje implementaci celého systému včetně detailního popisu uživatelského rozhraní, mezivrstvy, webového a komunikačního serveru. V šesté kapitole je testováno uživatelské rozhraní, a to z hlediska rychlosti odezvy (rychlost zpracování obsahu oken a rychlost překreslení oken) a propustnosti. Testy jsou vyhodnoceny a na jejich základě jsou implementovány některé optimalizace zrychlující odezvu grafického uživatelského rozhraní.

Na závěr je celý systém zhodnocen, jsou nalezena jeho silná a slabá místa a jsou navržena případná vylepšení.

2 Moderní webová uživatelská rozhraní

Webové aplikace přinášejí na rozdíl od desktopových mnoho výhod, jako je:

- Odstranění nutnosti instalace – aplikace je přístupná z jakéhokoli počítače připojeného k internetu, který disponuje vhodným prohlížečem, případně pluginem.
- Jednodušší správa a aktualizace – desktopové aplikace musejí být nainstalovány na každém počítači zvlášť a musejí se pravidelně aktualizovat. Dochází proto k přítomnosti velkého množství různých verzí. Internetové aplikace se naproti tomu instalují a aktualizují pouze na serverech provozovatele, což vede k jednotné verzi pro všechny uživatele.
- Centralizovaná správa dat umožňuje snadnější zálohy a při správném nastavení i větší bezpečnost.

Nástup rozsáhlejších webových aplikací ale také znamená větší nároky na uživatelské rozhraní. Pomocí statických HTML stránek nelze dosáhnout interaktivity a komplexnosti uživatelského rozhraní, jak ho známe z desktopových aplikací, dochází proto k vývoji technologií, které takové rozhraní poskytují. V současnosti patří mezi nejpoužívanější nástroje určené pro vývoj webových uživatelských rozhraní knihovny založené na metodách Ajax, framework Flex postavený nad pluginem Adobe Flash a plugin Microsoft Silverlight.

2.1 Ajax

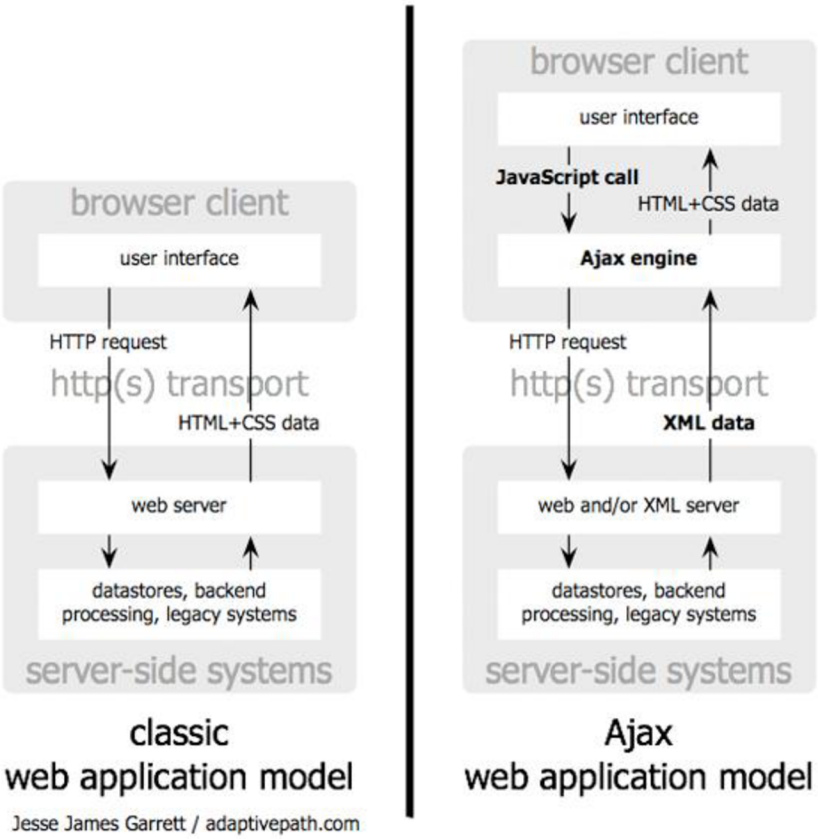
Ajax je zkratka pro Asynchronous JavaScript and XML (asynchronní JavaScript a XML) a označuje širokou škálu webových technologií sloužících k implementaci aplikací, které umožňují získávání dat ze serveru a dynamickou změnu stránky bez nutnosti znovunačtení celé stránky.

Autor termínu Ajax ve svém článku [1] vysvětluje, že Ajax využívá následující technologie:

- (X)HTML a CSS pro prezentaci dat
- DOM (Document Object Model) pro dynamické zobrazování a interakci
- XML a XSLT pro výměnu respektive manipulaci dat
- `XMLHttpRequest` pro asynchronní komunikaci se serverem
- JavaScript ke spojení všech těchto technologií.

Základem techniky Ajax je využití JavaScriptového objektu `XMLHttpRequest` k získání dat ze serveru na pozadí. Kvůli tomu není nutné aktualizovat celou stránku, ale jenom část, kterou je nutné změnit pomocí dat získaných ze serveru. I když je přímo v názvu objektu formát dat XML, je možné použít libovolný formát dat jako je například JSON [2].

Rozdíl mezi klasickou webovou aplikací a aplikací využívající modelu Ajax demonstruje Obr. 2.1.



Obr. 2.1 – klasický model vs. Ajax model [1]

Za účelem skrytí nekompatibilit implementací DOMu, CSS a JavaScriptu v jednotlivých prohlížečích vzniklo velké množství knihoven zapouzdřujících funkcionalitu pro vývoj Ajax aplikací. Patří mezi ně jQuery, YUI, Ext JS, Dojo, MooTools a jiné. Důležitou částí každé knihovny je prezentační vrstva, která spravuje uživatelské rozhraní na základě dat získaných ze serveru. V případě jQuery je to jQuery UI, naproti tomu YUI, Ext JS a Dojo obsahují GUI komponenty (widgety) přímo jako součást knihovny.

2.2 Adobe Flex

Adobe Flex je framework pro tvorbu RIA (Rich Internet Applications) postavený na platformě Adobe Flash, která je dostupná jako plugin do prohlížeče (Adobe Flash Player) pro operační systémy Windows, Mac OS X, Linux, Solaris, Windows CE, Android, Symbian, Palm OS a jiné. Podle [3] má Adobe Flash plugin nainstalováno 96,5% prohlížečů. Aktuální verze Adobe Flex je 4.1. Aplikace využívající Flex mají k dispozici:

- MXML, značkovací jazyk postavený na XML, který slouží ke tvorbě GUI
- Podmnožinu CSS (kaskádových stylů) pro popis zobrazení GUI komponent
- ActionScript, nativní jazyk platformy Adobe Flash postavený na standardu ECMAScript

- Velké množství komponent pro uživatelské rozhraní (`List`, `Scroller`, `TextBox`, `DataGrid`), komunikaci se serverem (`HTTPService`, `WebService`, `RemoteObject`), data binding, validaci a formátování dat, přechody a efekty

Od verze 3 je Flex SDK k dispozici pod licencí Mozilla Public License. Aktuální verze 4 přináší například plnou podporu pro Adobe Flash Player 10, vylepšený textový engine (Text Layout Framework) a novou sadu GUI komponent (Spark).

Oficiální IDE, Adobe Flash Builder 4 postavené na projektu Eclipse, není součástí otevřeného SDK; framework Flex ale díky své otevřenosti lze používat i v jiných IDE jako je FlashDevelop nebo Eclipse.

Platforma Adobe Flex také poskytuje vylepšený workflow pro designéry – do Flash Builder IDE lze jednoduše importovat zdroje z ostatních produktů firmy Adobe jako je Adobe Flash, Adobe Fireworks nebo Adobe Photoshop.

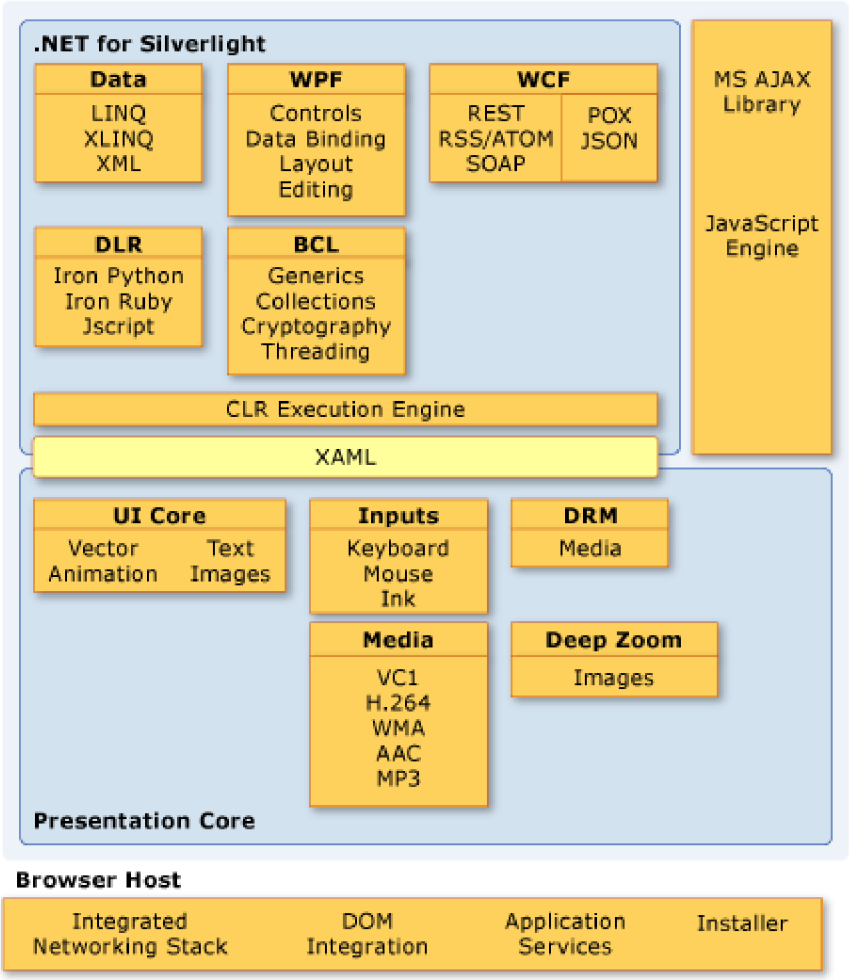
2.3 Microsoft Silverlight

Microsoft Silverlight je podobně jako Adobe Flex framework pro tvorbu RIA. Běhové prostředí (Run-time Environment) pro Silverlight je k dispozici jako plugin do prohlížeče v operačních systémech Windows, Mac OS X a Linux. Podle [3] má Microsoft Silverlight plugin nainstalováno 55,37% prohlížečů. Aktuální verze Microsoft Silverlight je 4.0.

Architektura technologie Silverlight je podobná frameworku Adobe Flex a obsahuje:

- Core Presentation Framework (CPF), množinu komponent a služeb pro UI a interakci s uživatelem včetně odlehčeného GUI pro použití ve webových aplikacích, komponent pro uživatelský vstup, data binding a zobrazování vektorové a rastrové grafiky, textu, animací a videa. Součástí je XAML, značkovací jazyk postavený na XML sloužící k definici GUI (obdoba MXML v Adobe Flex) [4]
- .NET Framework for Silverlight, podmnožinu .NET Frameworku. Programátor tedy může zvolit pro vývoj některý z jazyků jako jsou C#, Visual Basic, Iron Python, nebo Iron Ruby.

Diagram architektury Microsoft Silverlight 4 je popsán na obrázku 2.3. Mezi výhody oproti Adobe Flex patří možnost použití stejného kódu na serveru i klientovi (.NET), což snižuje nároky na programátory. Primární IDE k vývoji Silverlight aplikací je Visual Studio, což opět u .NET programátorů snižuje čas na učení nové technologie.



Obr. 2.3 – Architektura Microsoft Silverlight 4 [4]

3 Porovnání technologií pro moderní webová uživatelská rozhraní

Moderní webové aplikace mají stále větší nároky na propustnost, rychlost reakce, rychlost zpracování a zobrazení dat. V této kapitole se budeme věnovat testům jednotlivých technologií zaměřených právě na tyto vlastnosti. Každý experiment je desetkrát zopakován a jako výsledek je brán průměr těchto deseti experimentů.

3.1 Porovnávané technologie

Technologie porovnávané v jednotlivých testech jsou Adobe Flex 3/4 (Flash Player 10), Microsoft Silverlight 3 a Ajax. Protože Ajax není přímo technologie ale metodika, pro implementaci Ajax testů byla kvůli svému velkému tržnímu podílu [5] vybrána knihovna jQuery.

Všechny následující testy byly provedeny na počítači s CPU Intel Core i5 M430 @ 2,27GHz, 8GB DDR3 1066 SO-DIMM, OS Windows 7 Professional 64bit (sady testů 1 a 2) případně OS Ubuntu Linux 10.10 (sada testů 3).

3.2 Sada testů 1: rychlost zpracování a prezentace dat

V této sadě testů je porovnávána rychlost zpracování dat (HTML, XML nebo JSON) a vykreslení tabulky jednotlivými technologiemi. Jako zdroj dat slouží tabulky o 5 sloupcích a 50 000 řádcích, jejichž data jsou identická ve všech třech formátech (HTML, XML a JSON). Aby se minimalizoval vliv přenosového média na výsledný čas, jsou všechny soubory včetně datových hostovány na lokálním serveru.

U technologií Adobe Flex a Microsoft Silverlight jsou pro zobrazení tabulky použity nativní komponenty `DataGrid`, u technologie Ajax je použit jQuery plugin DataTables [6] poskytující podobnou funkcionalitu jako komponenty `DataGrid` u Flex a Silverlight (sekce 3.2.1) a značka `table` (sekce 3.2.2 a 3.2.3).

U každého testu se měří doba, kterou zabere načtení dat a jejich parsování, a doba naplnění a vykreslení tabulky, a také celková doba zpracování.

3.2.1 Ajax - jQuery DataTables

V tomto testu jsou načtena data ve formátu JSON do pluginu DataTables pomocí `XMLHttpRequest`.

Jak lze vidět z tabulky 3.1, doba stažení a parsování formátu JSON je akceptovatelná, oproti tomu doba naplnění a vykreslení je poměrně dlouhá (kolem 6000 milisekund); při reálném použití by však bylo možné tuto dobu značně zkrátit například vhodným stránkováním [7].

Experiment	Stažení a parsování [ms]	Vykreslení tabulky [ms]	Celkem [ms]
1	540	5653	6193
2	549	5869	6418
3	734	5929	6663
4	548	6096	6644
5	700	6162	6862
6	532	6031	6563
7	660	5494	6154
8	363	5692	6055
9	364	5628	5992
10	385	5718	6103
Průměr	**538**	**5827**	**6365**

Tab. 3.1: Výsledky testu Ajax - jQuery DataTables

3.2.2 Ajax – XML

U tohoto testu jsou data načtena ve formátu XML pomocí `XMLHttpRequest`, je z nich vygenerována tabulka pomocí značky `table` a tato tabulka je vložena do stránky pomocí `innerHTML` [8].

Oproti formátu JSON se zde projevuje značně delší doba parsování XML dokumentu (o 345 ms, což je zpomalení o 50% oproti JSON). Ta je pravděpodobně zapříčiněna jednoduchostí formátu JSON a tím, že jsou výstupem parseru základní objekty typu řetězec, číslo nebo pole, zatímco výstup XML je relativně složitý DOM Document objekt.

Doba naplnění a vykreslení tabulky je oproti minulému testu přibližně dvakrát kratší (viz tabulka 3.2), ale za cenu snížené funkcionality (například nemožnost řazení a filtrování).

Experiment	Stažení a parsování [ms]	Vykreslení tabulky [ms]	Celkem [ms]
1	1019	3770	4789
2	847	3347	4194
3	843	2765	3608

4	1013	3617	4630
5	870	3372	4242
6	826	3575	4401
7	828	3372	4200
8	838	3792	4630
9	815	3319	4134
10	934	3853	4787
Průměr	**883**	**3478**	**4361**

Tab. 3.2: Výsledky testu Ajax - XML

3.2.3 Ajax – HTML

V tomto testu jsou data načtena pomocí `XMLHttpRequest`, a protože obsahují přímo HTML kód tabulky identický s kódem generovaným v minulém testu, jedinou operací je vložení kódu do stránky pomocí `innerHTML`.

Protože nedochází k žádnému parsování dat, snížila se doba stažení dat na 224 ms a vykreslení tabulky na 2334 ms. Z Ajax metod je tato varianta nejrychlejší, avšak na úkor funkcionality (například nemožnost řazení a filtrování dat oproti jQuery DataTables) a zvýšené zátěže serveru (je nutné předgenerování HTML dat na serveru).

Experiment	Stažení a parsování [ms]	Vykreslení tabulky [ms]	Celkem [ms]
1	204	2440	2644
2	253	2356	2609
3	266	2333	2599
4	259	2314	2573
5	228	2259	2487
6	204	2247	2451
7	149	2425	2574
8	214	2352	2566
9	181	2311	2492
10	277	2299	2576
Průměr	**224**	**2334**	**2558**

Tab. 3.3: Výsledky testu Ajax - HTML

3.2.4 Adobe Flex

U tohoto testu dochází k načtení XML dat pomocí komponenty `HTTPService`, která následně rozparsovaná data předá komponentě `DataGrid`, která je zobrazí.

Nejvíce času u tohoto testu zabere stažení a parsování XML dat (průměrně 892 ms), což je výsledek velmi podobný testu v tabulce 3.2 (883 ms). Samotné naplnění a vykreslení tabulky poté zabere pouhých 327 ms.

Experiment	Stažení a parsování [ms]	Vykreslení tabulky [ms]	Celkem [ms]
1	922	360	1282
2	1167	327	1494
3	835	317	1152
4	864	324	1188
5	795	328	1123
6	1000	290	1290
7	831	341	1172
8	779	318	1097
9	903	350	1253
10	822	312	1134
Průměr	**892**	**327**	**1219**

Tab. 3.4: Výsledky testu Adobe Flex

3.2.5 Microsoft Silverlight

V tomto testu jsou data ve formátu XML načtena pomocí komponenty `WebClient`, uložena do interní reprezentace (třída `Item`) a zobrazena v komponentě `DataGrid`.

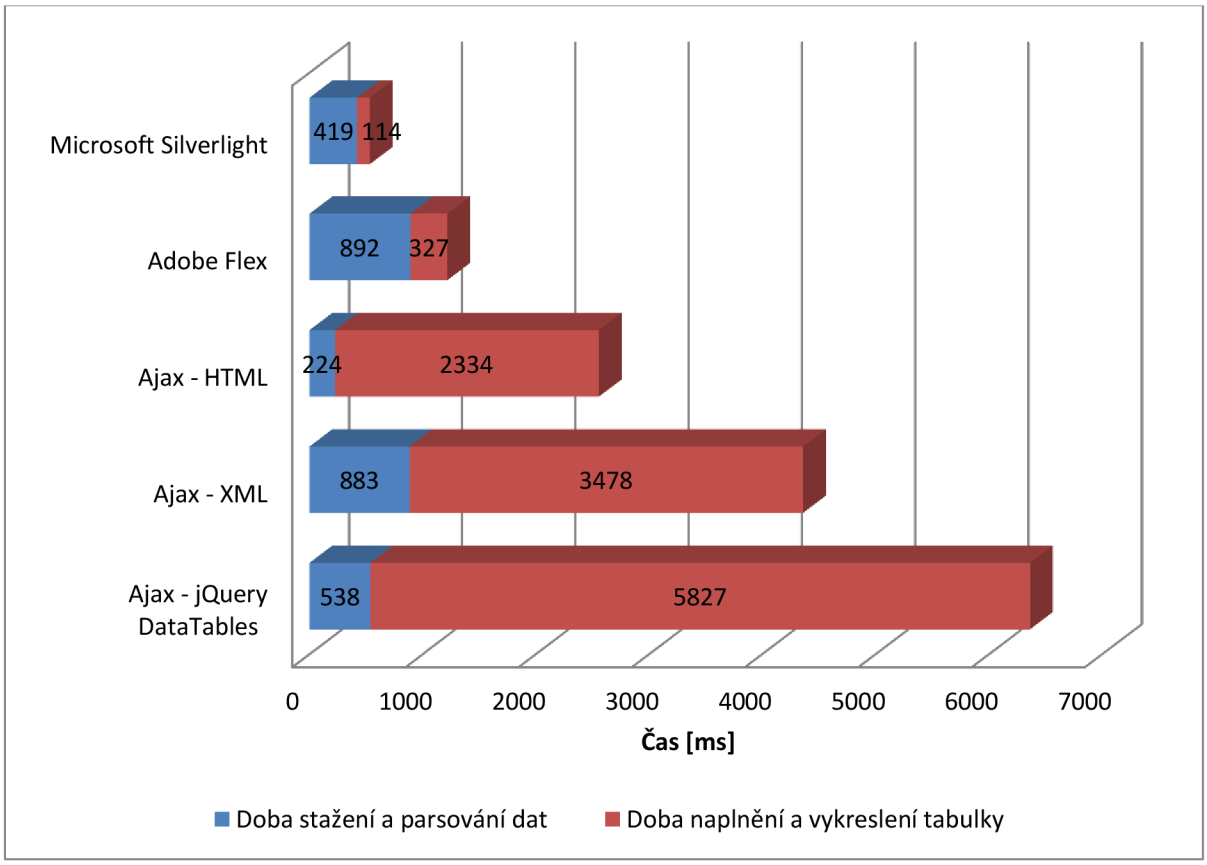
Čas stažení a parsování XML dat je v tomto případě nejkratší ze všech testů (pouze 419 ms), stejně jako vykreslení tabulky (114 ms) a celkový čas provedení (533 ms).

Experiment	Stažení a parsování [ms]	Vykreslení tabulky [ms]	Celkem [ms]
1	504	101	605
2	409	167	576
3	403	100	503
4	432	117	549
5	415	101	516
6	408	117	525
7	429	100	529

8	393	111	504
9	389	108	497
10	406	115	521
Průměr	**419**	**114**	**533**

Tab. 3.5: Výsledky testu Microsoft Silverlight

3.2.6 Souhrn výsledků všech testů



Graf 3.1: Graf průměrných výsledků testů rychlost zpracování a prezentace dat

3.2.7 Zhodnocení

V této kategorii jasně vede technologie Microsoft Silverlight s nejkratší dobou parsování XML dat, vykreslení tabulky i celkovou dobou zpracování. Technologie Adobe Flex dosáhla také relativně dobrých výsledků s druhou nejkratší dobou zpracování.

Ajax technologie dosahují minimálně dvojnásobné celkové doby než technologie Adobe Flex, přičemž nejdelší část tvoří naplnění a vykreslení tabulky. Tuto dobu by však bylo možné snížit vhodnou implementací stránkování.

3.3 Sada testů 2: rychlost vykreslování grafických elementů

U této sady testů je porovnávána rychlost vykreslování grafických elementů. Moderní webová uživatelská rozhraní musí být schopna vykreslovat velké množství elementů uživatelského rozhraní, vektorové i rastrové grafiky a textů; musí také zvládat plynulé animace a průhlednost.

3.3.1 GUIMark

Všechna tato kritéria testuje sada srovnávacích testů GUIMark, kterou vytvořil Sean Christmann [9]. Testy jsou implementovány pro HTML, Flex, Silverlight a Javu (knihovna Swing) a graficky jsou téměř identické, takže poskytují velmi přesné porovnání.

Na obrázku 3.1 je ukázka obrazovky GUIMark s očíslovanými elementy. Jednotlivé elementy testují výše uvedené vlastnosti:

1. Pohybující se obrázek testuje plynulost pohybu rastrové grafiky
2. Obdélníkové elementy plynule zvětšují a zmenšují svou velikost. Zde se testuje „9 Slice Scaling“ [10]
3. Textové pole – zde se testuje uspořádání a vykreslení textu
4. Animované hvězdy testují průhlednost a animaci vektorové grafiky

Obr. 3.1: Obrazovka GUIMark

Testy jsou navrženy a implementovány tak, aby co nejvíce zatížily vykreslovací řetězce jednotlivých technologií a tím porovnaly jejich relativní výkonnost. Při návrhu bylo zohledněno několik bodů, které mají za úkol zajistit, aby výstupem testů byla smysluplná a užitečná data: [11]

- Náročnost testů má být tak vysoká, aby maximální počet snímků za sekundu byl co nejmenší. Čím menší počet snímků za sekundu test zobrazí, tím větší vizuální rozdíl mezi jednotlivými technologiemi bude možno pozorovat. Většina komerčních LCD monitorů také pracuje na obnovovací frekvenci 60 Hz, takže větší počet snímků za sekundu nemá pro uživatele smysl.
- Animace mají mít absolutní periodu nezávislou na počtu snímků za sekundu. Tím se docílí toho, že při větších hodnotách FPS (frames per second) jsou animace plynulejší a opět lze lépe pozorovat vizuální rozdíly při přechodech.
- Testy by mělo být velmi těžké optimalizovat. Jednotlivé efekty byly zvoleny tak, aby každou technologii co nejvíce vytížily.
- Rychlost použitých programovacích jazyků má mít velmi malý vliv na výsledek testu.
- Vykreslovací pipeline má být co nejvíce zatížena. V současném stavu, kdy není vykreslování hardwarově akcelerováno, by mělo být CPU co nejvíce vytíženo při FPS menším než 60.

3.3.2 Testované technologie

Pomocí testu GUIMark byly testovány technologie HTML (V prohlížečích Microsoft Internet Explorer 8 v módu Standards mode (jádro Trident 4.0), Mozilla Firefox 3.6 a Google Chrome 8.0), Adobe Flex 3 a Microsoft Silverlight 3.

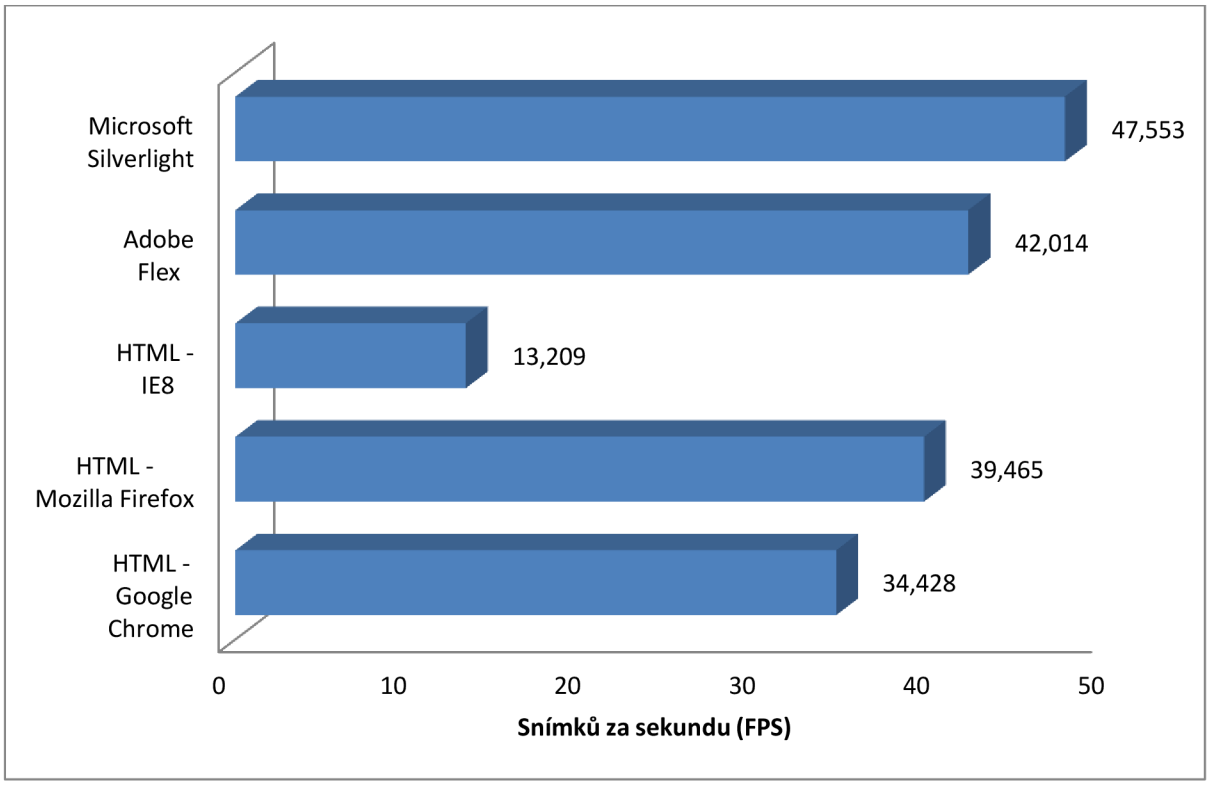
3.3.3 Výsledky testů

V tabulce 3.6 a grafu 3.2 jsou shrnuty průměrné hodnoty snímků za sekundu, kterých jednotlivé technologie v testu GUIMark dosáhly. Výsledky, kterých dosáhly technologie Microsoft Silverlight a Adobe Flex a prohlížeče Google Chrome 8 a Mozilla Firefox 3.6, jsou na podobné, pro uživatele dostatečně plynulé úrovni.

Experiment	HTML - Google Chrome [FPS]	HTML - Mozilla Firefox [FPS]	HTML - IE8 [FPS]	Adobe Flex [FPS]	Microsoft Silverlight [FPS]
1	34,78	41,28	13,52	41,75	44,22
2	34,24	38,93	13,18	41,88	49,63
3	34,05	39,37	12,88	41,48	48,41
4	34,30	38,99	12,78	41,33	47,97
5	34,31	36,82	13,13	41,64	48,03

6	34,18	41,97	13,44	41,65	46,31
7	34,78	39,52	13,27	42,41	46,57
8	34,97	41,28	13,38	42,90	47,93
9	34,47	38,84	13,47	42,22	48,69
10	34,20	37,65	13,04	42,88	47,77
Průměr	**34,43**	**39,47**	**13,21**	**42,01**	**47,55**

Tab. 3.6: Výsledky testu rychlosti vykreslování ve snímcích za sekundu (FPS)



Graf 3.2: Graf průměrných výsledků testů rychlosti vykreslování grafických elementů

Výsledky prohlížeče Microsoft Internet Explorer 8 dosáhly značně horších výsledků a animace jsou na testovacím stroji viditelně trhané. To je do jisté míry možné připsat relativnímu stáří zmíněného prohlížeče; nová verze (Microsoft Internet Explorer 9 [12]) má oznámenu hardwarovou akceleraci vykreslování.

3.3.4 Zhodnocení

I v této sadě testů je absolutním vítězem technologie Microsoft Silverlight s průměrnými 47,5 FPS, těsně následovaná technologií Adobe Flex s průměrnými 42,0 FPS a prohlížeči Mozilla Firefox 3.6 s průměrnými 39,5 FPS a Google Chrome 8 s průměrnými 34,4 FPS. Prohlížeče Microsoft Internet Explorer 8 dosáhl výsledků horších, ale v další verzi by se měl ostatním technologiím vyrovnat.

V této sadě testů dosáhla technologie Ajax jako celek průměrně horších výsledků než technologie Microsoft Silverlight a Adobe Flex, při implementaci graficky velmi náročných aplikací je proto potřeba na tuto skutečnost brát zřetel.

3.4 Sada testů 3: rychlost reakce a prostupnost spojení se serverem

Jednou z nejdůležitějších vlastností RIA frameworků je interaktivita. Aplikace potřebují rychle a spolehlivě odesílat a hlavně přijímat data od serverů, a to často v reálném čase. V této sekci nejdříve rozebereme techniky spojení se serverem v reálném čase, a poté provedeme výkonnostní testy jednotlivých technik.

3.4.1 Techniky spojení v reálném čase technologiemi Flash a Silverlight

Nejčastěji používaným nástrojem pro spojení v reálném čase se serverem je v technologiích Adobe Flex i Microsoft Silverlight protokol TCP. V Adobe Flex jsou TCP sokety implementovány v třídách `flash.net.Socket` a `flash.net.XMLSocket` [13] a v Microsoft Silverlight v třídě `System.Net.Sockets.Socket` [14].

V obou případech je možnost iniciovat spojení se serverem omezena na stejnou doménu, ze které je aplikace načtena. Připojit se na jinou doménu je možné pouze po načtení cross-domain policy file ze serveru, na který se připojení iniciuje [15]. Důvodem je předcházení některým útokům, které by mohl provádět nepřátelský kód na klientských zařízeních (Denial of Service, DNS Rebinding Attack, nebo Reverse Tunnel Attack) [16].

3.4.2 Techniky spojení v reálném čase v technologii Ajax

Spojení se serverem iniciované klientem (typ Pull) je implementováno JavaScriptovým objektem `XMLHttpRequest`. Pokud je však potřeba posílat data ze serveru klientovi (typ Push), případně udržovat trvalé obousměrné spojení, nenabízí současné standardy žádnou přímou možnost. V HTML5 tento problém pravděpodobně vyřeší technologie WebSocket [17], do finální specifikace a rozšíření do prohlížečů je však třeba použít techniky, které jsou dostupné nyní.

V současnosti používané techniky typu Push se označují pod souhrnným názvem Comet [18]. Existuje mnoho variant využívajících skrytých iframů nebo `XMLHttpRequest` s vlastností content type `multipart/x-mixed-replace`, žádná z těchto technik však nefunguje spolehlivě ve všech nejpoužívanějších prohlížečích.

Nejpoužívanější technikou se proto v posledních letech stalo tzv. long polling, které funguje ve všech prohlížečích podporujících `XMLHttpRequest`. Tato technika je používána v mnoha webových aplikacích vyžadujících spojení v reálném čase se serverem jako je Facebook, Google Docs, nebo Meebo. Principem této techniky je

1. Klient iniciuje asynchronní připojení na server pomocí `XMLHttpRequest`
2. Server čeká, dokud nevyvstane potřeba poslat klientovi data (čekání může trvat i v řádu minut). Klient mezitím pořád čeká na odpověď.
3. Jakmile klient obdrží data od serveru a zpracuje je, okamžitě otevře další `XMLHttpRequest` připojení k serveru, které čeká na další data od serveru.

Tímto způsobem je vždy jedno připojení k dispozici, aby mohl server okamžitě poslat data klientovi. Komunikace od klienta k serveru je realizována dalším `XMLHttpRequest` dotazem.

3.4.3 Testy TCP a long polling spojení

V následujících sekcích provedeme test rychlosti odezvy TCP a long polling spojení a test propustnosti a rychlosti reakce long polling serverů. Ve všech testech slouží jako referenční server neblokující TCP server implementovaný v jazyce Python pomocí rozhraní `epoll`

3.4.4 Test rychlosti odezvy TCP spojení

První test v této sadě testů má za cíl změřit rychlost odezvy TCP spojení realizovaného pomocí třídy `XMLSocket` ve frameworku Adobe Flex. Klient měří dobu od odeslání zprávy do návratu hodnoty ze serveru, který se nachází na lokální stanici.

Experiment	Roundtrip [ms]
1	2
2	2
3	3
4	2
5	4
6	3
7	2
8	2
9	3
10	2
Průměr	**3**

Tab. 3.6: Výsledky testu rychlosti odezvy TCP spojení ve frameworku Adobe Flex

Jak lze vidět z tabulky 3.6, TCP spojení z platformy Adobe Flex je velmi rychlé a dostačující i pro služby s vysokými požadavky na malou latenci.

3.4.5 Test rychlosti odezvy long polling spojení

Tento test má za cíl změřit rychlost odezvy techniky long polling realizovanou v prohlížeči Google Chrome 8 pomocí `XMLHttpRequest`. Klient nejprve otevře long polling spojení pomocí `XMLHttpRequest`, a poté měří dobu od odeslání další zprávy pomocí `XMLHttpRequest` do návratu hodnoty ze serveru pomocí long polling spojení. Server se opět nachází na lokální stanici.

Experiment	Roundtrip [ms]
1	11
2	13
3	13
4	9
5	12
6	15
7	11
8	8
9	13
10	12
Průměr	**12**

Tab. 3.7: Výsledky testu rychlosti odezvy long polling spojení

Z tabulky 3.7 lze vidět, že i při použití long pollingu je rychlost odezvy relativně malá.

3.4.6 Test propustnosti a rychlosti reakce long polling serverů

U tohoto testu se zaměříme na rychlost reakce a propustnost dvou implementací long polling serverů v jazyce Python.

Existuje mnoho knihoven implementujících long polling server (twisted, Orbited, CherryPy, Jetty, web.py, Meteor, Tornado), drtivá většina z nich je napsána v jazyce Python. Pro srovnání výkonu byl vybrán server postavený na knihovně Tornado, která je podle testu [19] jedna z nejrychlejších. Jako referenční server slouží jednoduchý server v jazyce Python využívající rozhraní `epoll`.

3.4.6.1 Test propustnosti

Test propustnosti je proveden pomocí nástroje Apache Benchmark [20] s parametry `ab -n 100000 -c 500 http://localhost:8888/`. Odešle se 100 000 HTTP požadavků pomocí 500 konkurentních spojení a měří se čas, za který server všechny požadavky vyřídí.

Experiment	Tornado [s]	Epoll [s]
1	35,7	12,2
2	41,3	12,6
3	38,0	12,3
4	39,8	13,1
5	36,9	12,6
6	35,3	12,8
7	38,6	11,9
8	40,0	12,3
9	37,2	12,9
10	36,8	12,5
Průměr	**38,0**	**12,5**

Tab. 3.8: Výsledky testu propustnosti long polling serverů

V tomto testu jasně zvítězil jednoduchý epoll server s přibližně třetinovou dobou vyřízení požadavků oproti Tornado serveru.

3.4.6.2 Test odezvy

Odezva je testována stejným způsobem jako v sekci 3.4.4, ale zároveň je zavolán Apache Benchmark se stejnými parametry jako v sekci 3.4.5.1, aby byl simulován vytížený server.

Experiment	Tornado [ms]	Epoll [ms]
1	107	14
2	127	13
3	117	15
4	121	14
5	109	15
6	125	15
7	112	14
8	129	15
9	111	12

10	131	13
Průměr	**119**	**14**

Tab. 3.8: Výsledky testu latence long polling serverů

I v tomto testu je jasným vítězem jednoduchý epoll server, který bez problémů zvládá vyřizovat 500 konkurentních požadavků s minimální latencí.

3.4.7 Zhodnocení

Latence i prostupnost spojení v reálném čase se serverem je jak u pluginových technologií (Adobe Flex a Microsoft Silverlight), tak u Ajax řešení velmi dobrá. Je však třeba mít na paměti, že při použití long pollingu není garantováno pořadí doručení zpráv, takže pro implementaci spojení typu stream je nutné implementovat vlastní protokol s využitím např. ACK čísel, čímž se oproti TCP soketům zvyšuje složitost implementace.

Implementace long polling serveru pomocí knihovny Tornado se ukázala sice jednoduchá a rychlá, ale se značně menším výkonem než řešení implementované přímo nad epoll. Pro servery, kde je vyžadován velký počet konkurentních uživatelů, je třeba zvážit implementaci v některém z více nízkoúrovňových jazyků, jako jsou C nebo C++.

3.5 Zhodnocení

Z dat získaných v sadách testů 1 až 3 můžeme konstatovat, že všechny tři technologie (Ajax, Adobe Flex a Microsoft Silverlight) jsou využitelné v systémech s náročnými požadavky na rychlost zpracování a zobrazení dat, rychlost reakce a propustnost. Pluginové technologie Adobe Flex a Microsoft Silverlight dosáhly v prvních dvou testech vyššího výkonu, to je však vykoupeno nutností instalace proprietárního pluginu do prohlížeče, který ne vždy je k dispozici na všech architekturách či platformách.

Při výběru vhodné technologie je proto třeba zvážit, zda je mírně vyšší výkon pluginových řešení dostatečně důležitý na to, aby vyvážil nutnost instalace pluginu v případě nízké rozšířenosti (hlavně technologie Microsoft Silverlight), případně nedostupnost pluginu na některých zařízeních (hlavně mobilních, např. iPhone, iPod Touch, iPad).

4 Návrh webového systému

Tato kapitola se zabývá analýzou technické stránky moderních webových her a specifikací a návrhem webového systému, který bude dále implementován.

4.1 Technologie moderních webových her

Technologie, které se nejčastěji využívají v současných webových hrách, jsou Adobe Flex/Flash a Ajax.V této sekci rozebereme některé z novějších webových her využívající tyto technologie a zmíníme trendy v obou oblastech.

4.1.1 Webové hry na technologiích Ajax

Jedny z prvních webových her, které začaly využívat Ajax, byly strategické a taktické hry. Technologie Ajax byly většinou používány pro načtení HTML kódu částí stránky, které bylo třeba obnovit, téměř veškerá logika byla i nadále výlučně na straně serveru. Mezi zástupce takových her patří například Orion's Belt nebo Nile Online.

V současnosti využívají Ajaxových technologií většinou strategické a taktické hry, u kterých nejsou vysoké požadavky na animaci (ta je často implementována pomocí snímků ve formátu PNG animovaných JavaScriptem), ale kde je potřeba implementovat komplexní grafické uživatelské rozhraní. Herní plán je většinou upravován přímo na klientské straně podle interakce uživatele nebo dat ze serveru, nejčastěji ve formátu JSON.

Oproti tomu panely, okna i jiné prvky uživatelského rozhraní bývají načítány ze serveru jak již ve formě vygenerovaného HTML kódu, tak v datovém formátu (většinou JSON nebo XML). Aktualizace dat ze serveru stále probíhá většinou ve formě častých `XMLHttpRequest` dotazů na server. Začíná se však prosazovat i long polling.

Mezi nejmodernější hry využívající Ajaxu patří Lord of Ultima, Travians, nebo Kingdoms of Camelot.

4.1.2 Webové hry na platformě Adobe Flash

Technologie Flash bývá nasazována nejčastěji tam, kde je potřeba velký grafický výkon s velkým množstvím vykreslovaných objektů a animací, případně vektorová grafika. Využívají ji nejčastěji akční hry, stavitelské hry, ale i některé strategie.

Spojení se serverem je v případě akčních her a jiných real-timových aplikací (např. chat) nejčastěji realizováno pomocí TCP soketů. U her nevyžadujících aktualizace v reálném času nebo

pouze reagujících na uživatelské akce se používají HTTP požadavky s daty ve formátech AMF [21] nebo XML.

Platformy Flash využívají mimo jiné dvě nejhranější webové hry současnosti, CityVille a FarmVille, kromě toho například strategická hra Evony a akční hry Seafight nebo Dark Orbit.

4.1.3 Ajax a HTML5

Díky zvyšování výkonu prohlížečů, rozšiřování nových technologií z HTML5 (hlavně element Canvas) a využití techniky long pollingu se začínají objevovat první knihovny, které jsou schopny v prohlížeči realizovat hry náročné na grafiku, konkurentnost i rychlost odezvy, které byly donedávna doménou Adobe Flash. Příkladem je real-time izometrický engine od firmy Dextrose [22]. Dá se čekat, že v tomto směru bude docházet k intenzivnímu vývoji, hlavně kvůli rychle se zvyšujícímu počtu chytrých telefonů připojených na internet, které většinou nemají k dispozici Flash Player.

4.2 Neformální specifikace

Cílem této práce je návrh a vytvoření komplexního webového systému s důrazem na ergonomické a robustní grafické uživatelské rozhraní. Po průzkumu problematiky popsaném dříve v této práci byla za tento systém vybrána strategická webová hra.

Motivace výběru je následující:

- Strategická hra vyžaduje díky své komplexnosti kvalitní návrh grafického uživatelského rozhraní.
- Technologické řešení moderních strategických webových her má slabší místa, která se tato práce bude snažit vyřešit. Příkladem může být například použití techniky long polling místo použití techniky typu Pull, nebo návrh robustnějšího uživatelského rozhraní.

Jako implementační technologie byl vybrán Ajax, a to z následujících důvodů:

- Technologie Ajax je vhodnější pro implementaci komplexního uživatelského rozhraní.
- Webový systém nevyžaduje velmi graficky náročné operace jako je velké množství animací, proto je možno použít Ajax jakožto multiplatformní řešení nezávislé na pluginu v prohlížeči. Systém tedy lze spouštět i na mobilních zařízeních, která podporují JavaScript a DOM.

4.3 Principy hry

Hra nazvaná Twilight Age se řadí mezi strategické tahové hry. Je zasazena v prostředí Kuiperova pásu [23] 22. století a je rozdělena na kola, kde každé kolo trvá 24 reálných hodin. Na konci kola

proběhne výpočet všech akcí naplánovaných v daném kole a začíná kolo nové. V každém kole mají hráči k dispozici 20 tahů, během kterých provádí posádka mateřské lodi akce.

Každý hráč ovládá mateřskou loď, jejíž typ patří jedné z pěti frakcí:

1. Korporace
2. Vesmírné jednotky
3. Konfederace
4. Mars
5. Koperník

Hráč může každý den plánovat pohyb mateřské lodi po vesmíru, přiřazovat posádku do jednotlivých zaměstnání, vysílat bojové či civilní mise a komunikovat s ostatními hráči.

4.3.1 Suroviny

Hráči mají k dispozici následující suroviny:

- Artefakty - jsou nejdůležitější surovinou ve hře; lze je získat jen těžbou asteroidů a bojem
- Kredity - základní měna, získávají se každé kolo podle toho, v jaké zóně se mateřská loď nachází, z dolů, které dokončily těžbu, z rozbořených nepřátelských dolů a z nepřátelských mateřských lodí, které hráč vyraboval
- Jídlo - vyrábí farmáři, každý člen posádky zkonzumuje za tah 1 jednotku jídla. Při nedostatku jídla lidé opouští mateřskou loď. S jídlem lze obchodovat.
- Komponenty - základní součásti lodí, vyrábí je dělníci. Jsou potřeba pro výrobce, kteří z nich vyrábí lodě. S komponenty lze obchodovat.
- Datadisky - vyrábí je inženýři, spolu s výzkumnými body jsou potřeba k výzkumu technologií vědci. S datadisky lze obchodovat.
- Výzkumné body - hráč při každém výpočtu nového kola získá ze svých artefaktů výzkumné body. Počet výzkumných bodů je stejný, jako počet artefaktů v držení hráče, a může se pohybovat od 600 do 1500, z toho bonusový přírůstek z dolů nemůže nikdy přesáhnout 200. Výzkumné body lze spolu s datadisky použít na výzkum technologií.
- Body vylepšení - od celkového počtu (investovaných i volných) výzkumných bodů se odvíjí množství bodů vylepšení, které může hráč investovat do vylepšení svých lodí.
- Žetony - hráč dostane každé kolo 1000 žetonů, které může použít v Galaktickém kasinu ke hře, nebo za ně může přímo nakoupit vylepšení v obchodu kasina.

4.3.2 Mateřská loď

Maximální počet členů posádky je 10 000 a každý člen může být přiřazen do jednoho ze zaměstnání:

1. Vojáci – tvoří posádku lodí, které bez nich nemohou vzlétnout
2. Rekruti – každý tah se 4% rekrutů vycvičí na vojáky

3. Farmáři – vyrábí jídlo, které posádka spotřebovává každý tah
4. Dělníci – vyrábí komponenty, ze kterých se staví bojové lodě
5. Inženýři – vyrábí datadisky, které potřebují vědci
6. Výrobci – mechanici, kteří z komponent vyrobených dělníky montují bojové lodě
7. Vědci – pomocí datadisků a výzkumných bodů vynalézají nové technologie
8. Instruktoři – cvičí agenty, kteří se používají v obchodních válkách
9. Asistenti – zvyšují efektivitu práce členů posádky, které podporují

Poté, co jsou nováčkové přiřazeni na své místo, se postupně zacvičují. Tento proces trvá několik tahů, každý tah se zacvičí 40% nezacvičených lidí. Nezacvičení členové posádky nic neprodukují.

V mateřské lodi je možné postavit 16 000 sekcí. V sekcích pracují členové posádky zařazení do některého zaměstnání; pokud není dostatek sekcí, nelze do zaměstnání přidat více lidí.

V hangáru se nacházejí všechny bojové i civilní lodě. Lodě lze rozložit zpět na součástky, ale s účinností pouze 50%. V hangáru lze také lodě přidat do výrobní fronty. Na lodích ve výrobní frontě pracují výrobci, kteří každý tah z komponent sestaví určitý počet lodí. Konstrukční body jsou potřebné na stavbu lodí ve frontě. Fronta má kapacitu 5 příkazů.

Kromě čtyř lodí společných pro všechny frakce může každá frakce vyrábět tři až pět lodí unikátních pro danou frakci.

V mateřské lodi lze také cvičit agenty rezidenty, tajné agenty nebo speciální agenty. Agenti nepatří mezi zaměstnání, do kterých lze rozdělit posádku, ale jde o zcela nezávislé jednotky, které jsou používány při obchodních válkách.

Vědci mohou vynalézat nové technologie. Každá frakce má svůj technologický strom organizovaný v řádcích po sedmi sloupcích, přičemž k odemknutí dalšího řádku je třeba vyzkoumat technologie v určité hodnotě. Technologie vylepšují různé aspekty mateřské lodi, bojových lodí, nebo odemykají nové možnosti ve hře jako je přístup do Obchodní sítě (více viz sekce 4.3.5).

4.3.3 Mapa

Mapa 2D herního pole slouží k získání informací o situaci na herním plánu a k interakci s okolím. Mezi objekty nacházející se na herním plánu jsou asteroidy, struktury, mateřské lodě, NPC mateřské lodě (Non-player character, lodě ovládané počítačem), plánované akce, výsledky akcí a poznámky gildy. Herní mapa je rozdělena na zóny typu 0 až 3, které se liší v tom, kolik kreditů v ní hráči obdrží, a jak výnosné asteroidy se v ní nachází. Výnosy jsou největší v zóně 3 a s nižším číslem zóny klesají. V zóně 0 nelze bojovat.

Stavba dolů na asteroidech umožňuje zvýšení příjmu a nalezení artefaktů. Po stavbě důl tři kola těží suroviny a je možné na něj útočit. Existuje 5 druhů asteroidů, které se liší výtěžností; více výtěžné asteroidy se nacházejí ve vyšších zónách.

Na mapě se také mohou nacházet generátorové lodě (vyráběné frakcí Mars) nebo základny zvyšující obranu nebo útok, snižující ztráty nebo zvyšují účinek lodí vojenské policie (vyráběné frakcí Koperník).

4.3.4 Akce

Jednotlivé mateřské lodi mohou vysílat své lodi na mise různého druhu. V rámci gildy je možno koordinovat akce tak, že se jedné mise mohou zúčastnit lodi více hráčů (pokud je akce založena jako "společná"). Mezi akce patří:

- Vojenské útoky – tyto útoky jsou vždy cíleny na cizí mateřskou loď, důl, základnu nebo generátorovou loď ve volném prostoru. Brání se jim obrannými letkami, při úspěšném útoku (silnějším než obrana) je možno zničit až 15 000 štítů a odvézt suroviny a artefakty.
- Obranná letka brání lodě a struktury v oblasti před útokem.
- Konstrukční loď je možno vyslat k asteroidu za účelem vystavění dolu či do volného prostoru k vystavění podpůrné základny. K této lodi je možno přidat doprovodné lodě jako ochranu proti potenciální obranné letce, která hlídkuje v dané oblasti.
- Konstrukční loď také lze vyslat k opravě štítů poškozené lodi či struktury. Pokud je v tomto dni proveden proti tomuto objektu útok, který překoná obranu, účinnost oprav je pouze 50%.
- Akce lodí vojenské policie, infiltračních lodí a výzvědných lodí (frakce Koperník) – jsou to obrana vojenské policie, sabotáž letky, zaminování hangáru, odminování a výzvědná mise.
- Příprava obchodní války, obchodní války a obrana obchodní války. Pomocí obchodní války je možno jinému hráči odcizit některé suroviny.

4.3.5 Interakce a komunikace mezi hráči

Hráči se kvůli ochraně a spolupráci sdružují do gild. Gilda se skládá až z 10 hráčů, má vlastní fóra, a její členové mohou přímo spolupracovat při útoku i obraně. Aliance je společenství až 10 gild, které mají společné fórum a mohou spolupracovat na výhře celé hry.

Kterémukoliv hráči je možné poslat soukromou zprávu. Hráči mají k dispozici fóra, kde některé jsou soukromá (gildovní, alianční) a některá veřejná (veřejné fórum gild a aliancí, nábor do gild, diplomacie atd.).

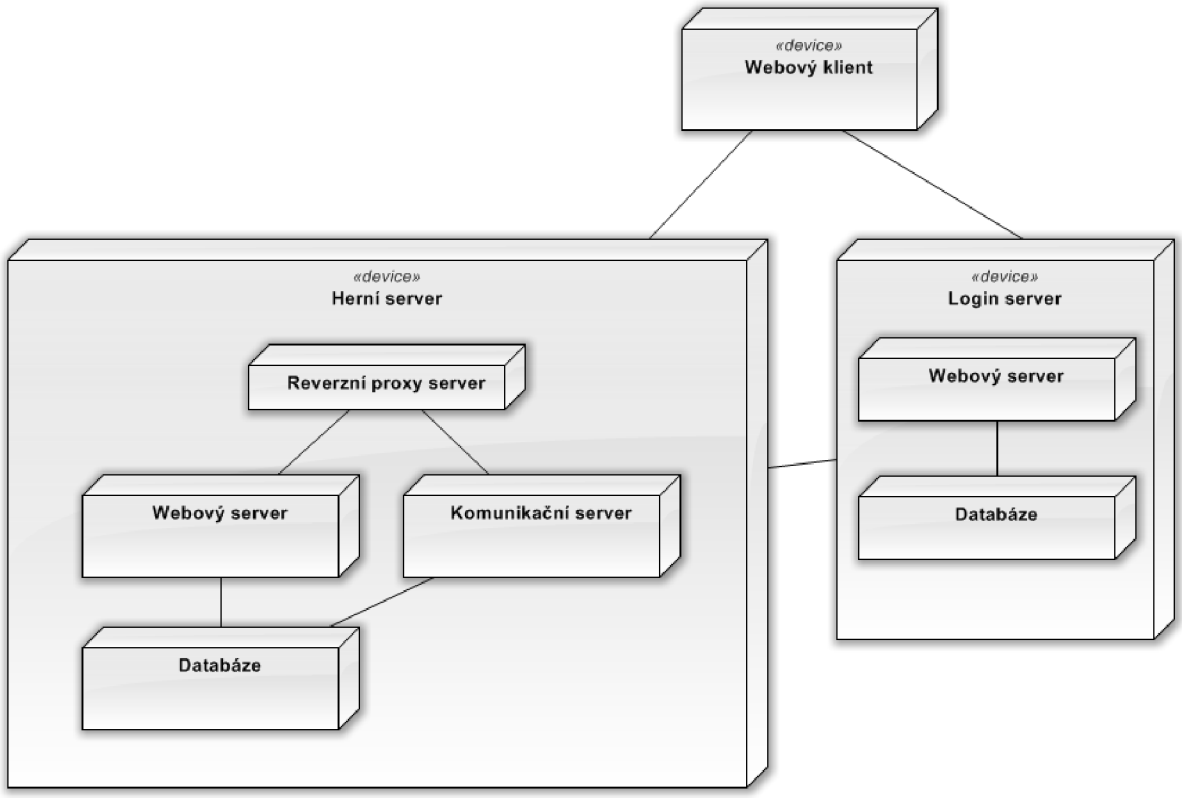
Hráč může přidat další hráče mezi přátele, potom lze vidět, kdy jsou online, a mohou navzájem komunikovat pomocí instantních zpráv (tzv. instant messaging). Pomocí instantních zpráv lze přátelům posílat v reálném čase zprávy. K dispozici je také gildovní, alianční a celoherní chat. Na zpestření jednodenního kola mohou hráči v Galaktickém kasinu hrát Texas hold ‘em poker.

Po vyzkoumání odpovídající technologie (Obchodní síť) je možno nakupovat a prodávat některé suroviny od ostatních hráčů, kteří se mohou nacházet kdekoliv ve hře.

4.4 Návrh

Návrh architektury systému (obrázek 4.1) vychází z požadavků daných pravidly hry, kde je jedna herní plocha omezena na přítomnost maximálně 10 000 hráčů. Vzniká proto potřeba zavést takzvaný login server, který má za úkol registrovat a přihlašovat hráče a přesměrovávat je na jednotlivé herní servery.

Real-time chat a instantní komunikace vyžadují použití long polling serveru pojmenovaného v diagramu „komunikační server“. Kvůli same origin policy [24] objektu `XMLHttpRequest` je nutné, aby požadavky od hráče přesměrovával reverzní proxy server [25] buď na herní webový server, nebo na komunikační server.



Obr. 4.1: Diagram nasazení systému

Navrhovaná softwarová architektura nemůže kvůli velkému využití Ajaxu přímo využít návrhový vzor typu Model-View-Controller nebo třívrstvou architekturu. Pro potřeby této práce proto byly adaptovány některé z návrhových vzorů pro Ajaxové aplikace uvedené v [26]. Klientská část je implementována jako jediná stránka, která po načtení veškerou interakci se serverem provádí pomocí `XMLHttpRequest` a všechny změny probíhají pomocí JavaScript skriptů přímo na stránce. Ve všech případech, kdy je na klientské straně dostatek dat, jsou akce (jako plánování útoků nebo změny zaměstnání posádky) prováděny přímo na klientské straně a informace o nich jsou zaslány serveru až po provedení. Tím sice dochází k duplikaci kódu na klientské a serverové straně, ale uživatelé mají k dispozici rychlé grafické rozhraní, které jen zřídka musí čekat na odpověď ze serveru.

4.5 Klientská část

Klientskou část systému běžící v prohlížeči lze rozdělit na dvě části, grafické uživatelské rozhraní a mezivrstvu, která se stará o herní logiku a komunikaci se serverem. Obě části využívají JavaScriptovou knihovnu Yahoo! YI Library (YUI), která byla vybrána z následujících důvodů:

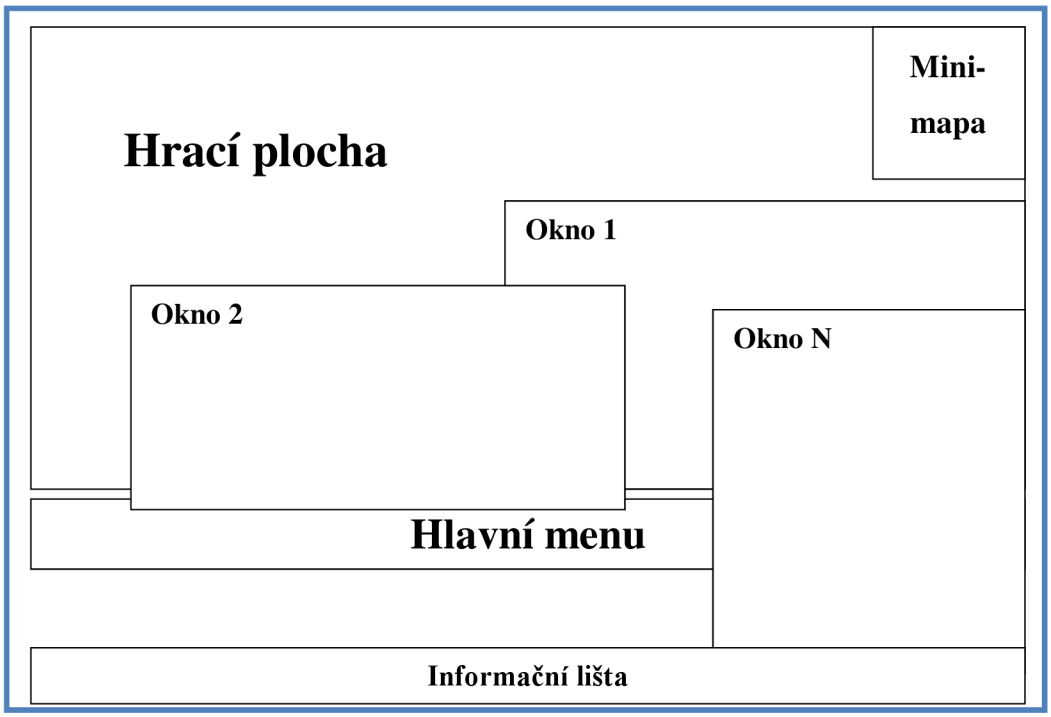
- Kvalitní zapouzdření rozdílů v implementacích JavaScriptu a DOM modelu v různých prohlížečích
- Velké množství komponent pro tvorbu uživatelského rozhraní
- Široká paleta utilit poskytující třídy pro manipulaci s daty, jednodušší práci s událostmi, animace, připojení a drag & drop

Velký přínos knihovny YUI leží kromě kvalitních utilit a podpory všech používanějších prohlížečů (Microsoft Internet Explorer 7 a 8, Mozilla Firefox 3, Google Chrome a Safari 5, Safari pro iOS a WebKit pro Android) také ve velké řadě komponent pojmenovaných widgety. Widgety nabízené knihovnou YUI jsou například tlačítka, kalendáře, tzv. kontejnery (widgety ohraničující obsah jako jsou `Module`, `Overlay`, `Tooltip`, `Panel`, nebo `Dialog`), tabulky, různé menu včetně kontextových, stránkovače, ukazatele průběhu (Progress Bars), WYSIWYG (What You See Is What You Get) editory textu, posuvníky (Sliders), panely (Tabs) a stromové struktury (komponenta `TreeView`).

U všech widgetů jde jednoduše změnit vzhled pomocí kaskádových stylů (CSS) a přihlásit se k odebírání událostí jako je posun, klik myši, stisknutí klávesy atd.

4.5.1 Grafické uživatelské rozhraní

Dobře navržené uživatelské rozhraní je jedním z nejdůležitějších aspektů systému, protože je potřeba uživateli jasným, srozumitelným a efektivním způsobem sdělit velké množství informací. Část informací je potřeba mít k dispozici jen někdy, ale například stav hracího pole nebo aktuální stav surovin je potřeba znát neustále. Z toho důvodu byla zvolena koncepce do jisté míry podobná organizaci oken a pracovní plochy v operačním systému typu Microsoft Windows (obrázek 4.2):



Obr. 4.2: Schéma rozložení GUI prvků

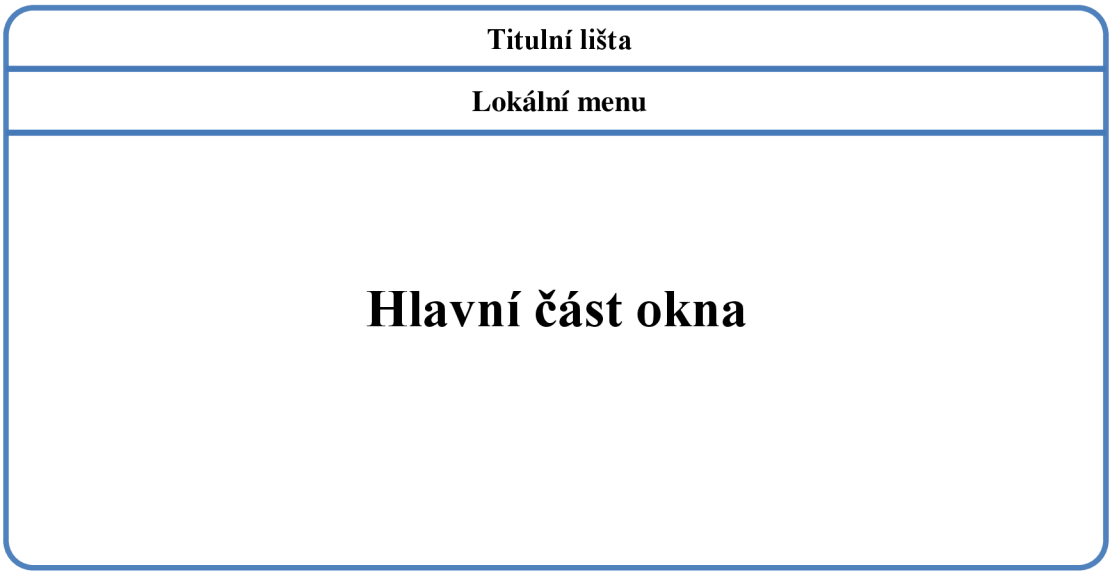
1. Na pozadí se nachází hrací plocha zobrazující polohu hráčů, asteroidů a ostatních objektů ve vesmíru. V pravém horním rohu se nachází tzv. minimapa, zmenšená verze hrací plochy, která poskytuje nejdůležitější globální informace a usnadňuje posun hrací plochy (kliknutí na minimapu posune hrací plochu na příslušné místo).
2. Na spodní části hrací plochy je hlavní menu s tlačítky, které otevírají okna s dalším obsahem, a pod ním informační lišta se stavy surovin a ovládáním chatů a instantní komunikace. Na pravé straně se nachází ikona, která skryje nebo zobrazí všechna otevřená okna.
3. Nad těmito prvky se otvírají nemodální okna (může jich být otevřen libovolný počet) s dalším obsahem, jako jsou informace o mateřské lodi, gildě, akcích, zprávách a další. Každé okno obsahuje tematicky blízký obsah, např. v jednom okně jsou zobrazena všechna fóra, v dalším nastavení a v dalším výsledky akcí.

Všechny prvky se tedy nacházejí v jediném okně prohlížeče, které není obnovováno; změny se zobrazují buď přímo na hrací ploše, nebo v jednom z oken. Okna jsou spravována správcem oken, který se stará o relativní pozice jednotlivých oken a správné překrytí dříve aktivovaných oken později aktivovanými. Při návrhu rozložení okenního systému bylo vycházeno z [27].

Samotné okno se skládá z následujících částí (obrázek 4.3):

1. Titulní lišta s názvem okna - potáhnutím myší za lištu lze pohybovat s celým oknem. V pravém rohu se nachází tlačítka pro zavření okna a také pro otevření okna s nápovědou pro aktuální okno. Tak může uživatel snadno vyhledat více informací, aniž by musel prohledávat manuál.

2. Lokální menu obsahuje odkazy na další stránky obsahu. V každém okně se nachází tematicky spřízněný obsah, příkladem odkazů v menu mohou být například Moje výsledky, Výsledky gildy a Výsledky aliance.
3. Hlavní část okna – zde se zobrazuje samotný obsah okna



Obr. 4.3: Schéma rozložení prvků v okně

Rozložení grafického uživatelského rozhraní je navrženo s ohledem na co nejvyšší uživatelský komfort. Nejvyužívanější část (hrací plocha) je vždy k dispozici na pozadí, stejně jako aktuální stav surovin a počet online přátel a členů gildy. Pomocí ikony na informační liště lze kdykoliv skrýt nebo zobrazit všechna otevřená okna, a tím je umožněn snadný přístup k hrací ploše i při velkém počtu otevřených oken.

Lze mít otevřených větší počet oken, takže může uživatel snadno využívat informací z jednoho okna v druhém, aniž by musel ukládat nebo opouštět rozdělanou práci. Příkladem může být hráč, který právě přidává do výrobní fronty infiltrační lodě, ale není si jistý, jestli velitel gildy opravdu doporučoval doplnit stavy tohoto typu lodí. V dalším okně otevře fórum gildy, zjistí, že je tomu opravdu tak, a přidání lodí dokončí.

4.5.2 Mezivrstva

Mezivrstva se stará o ošetření uživatelského vstupu, interaktivních akcí uživatele jako je posun hrací plochy a zobrazení / skrytí různých informací na ní, zakládání akcí nebo přesun posádky a komunikaci se serverem. Bližší informace o komunikačních protokolech následují v sekci 4.6.

4.6 Serverová část

Serverová část sestává z reverzního proxy serveru, webového serveru, databázového serveru a komunikačního serveru. Reverzní proxy server má jediný úkol, podle typu zprávy požadavky směruje buď na webový nebo komunikační server.

4.6.1 Webový server

Webový server kromě poskytování statických zdrojů jako je hlavní stránka, JavaScript skripty, CSS soubory, obrázky a jiné média, slouží výhradně jako rozhraní, které poskytuje služby pro klientskou část. Existují dva základní typy požadavků, na zjištění informací a na provedení akce.

- Zjištění informací – zde je typickým příkladem získání dat potřebných k zobrazení odeslaných soukromých zpráv pomocí odkazu `messages.php?sub=sent`. První část adresy `messages` označuje skupinu požadavků souvisejících s oknem "Zprávy" v uživatelském rozhraní, HTTP GET parametr `sent` označuje výpis odeslaných zpráv.
- Provedení akce – podobně jako v případě zjištění informací, i zde lze adresu rozdělit na část označující skupinu požadavků na změnu, a na část označující konkrétní změnu. Například odkaz `settings_change.php?type=password` označuje požadavek na úpravu nastavení účtu, konkrétně změnu hesla. Další parametry specifické pro každou konkrétní akci jsou zaslány pomocí metody HTTP GET nebo POST.

Mezi požadavky na zjištění informací patří hlavně zjišťování podrobností mateřských lodí, asteroidů, načtení aktuálního stavu herního plánu včetně plánovaných akcí a poznámek gildy, načtení obsahu oken nebo nové soukromé zprávy.

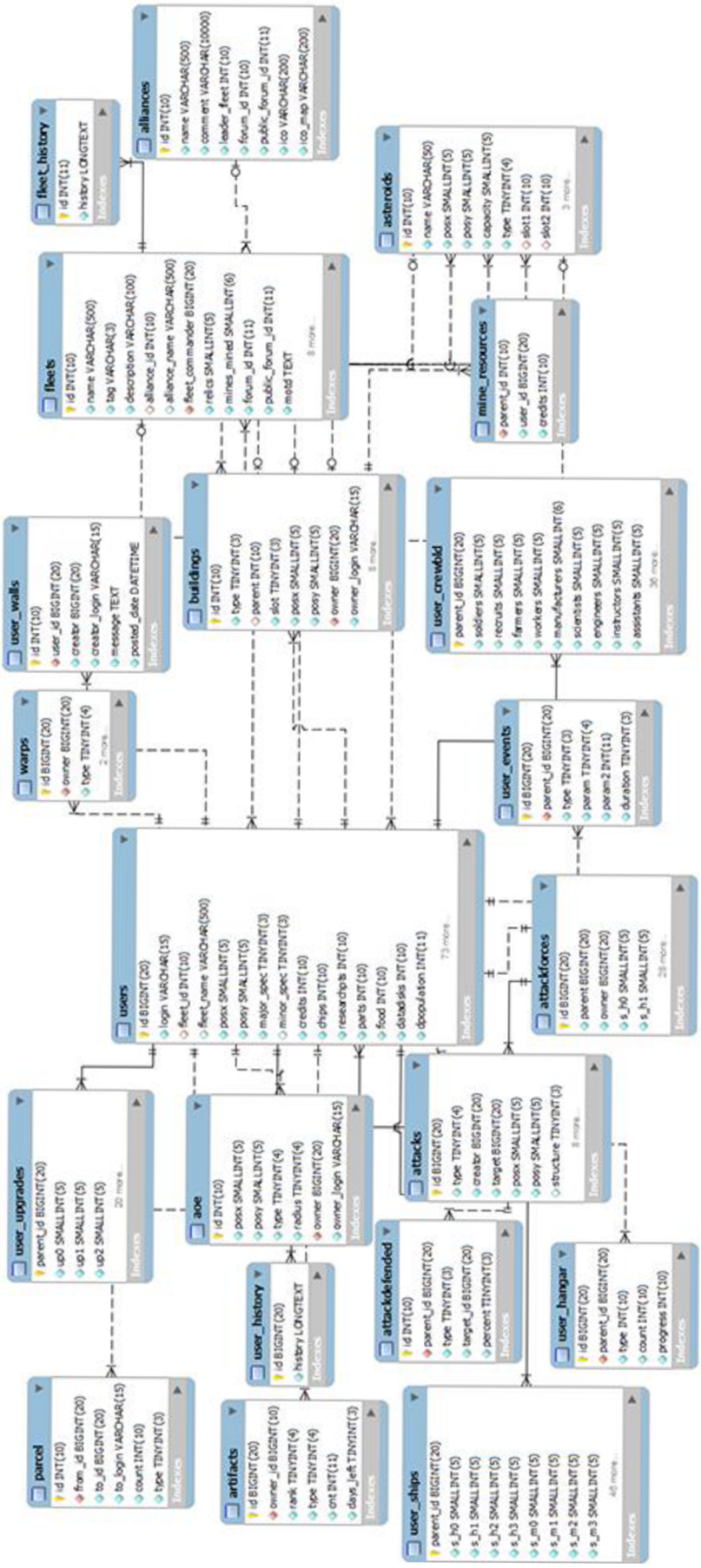
Mezi provedení akce patří hlavě tzv. business logika – validace a provedení akcí provedených na klientské straně jako je změna zaměstnání posádky, založení akce nebo přidání jednotek do akce, přidání/odebrání lodí do výrobní fronty, změna zkoumané technologie nebo výpočet nového tahu.

Webové rozhraní je implementováno pomocí jazyka PHP verze 5, který byl vybrán kvůli své rozšířenosti, velmi dobré dostupnosti dokumentace a velkému množství rozšiřujících knihoven.

4.6.2 Databázový server

Databáze obsahuje všechna persistentní data potřebná pro chod systému. Jako databázový server byl vybrán MySQL server kvůli své rozšířenosti a jednoduchosti použití, zvláště s jazykem PHP. Jako databázové úložiště dat (storage engine), který vyhovoval požadavkům (podpora transakcí a cizích klíčů) byl vybrán InnoDB.

ER diagram popisující nejdůležitější části databáze můžete naleznout na obrázku 4.4. S databází může komunikovat jak webový server, tak komunikační server.



Obr. 4.4: ER diagram obsahující nejdůležitější tabulky

4.6.3 Komunikační server

Komunikační server je jednoduchý HTTP server využívající techniku long polling, který zvládá udržovat až několik tisíc otevřených spojení a potenciálně vyřizovat podobný počet požadavků za sekundu. Z důvodu požadavků nízkého zatížení serveru byl zvolen implementační jazyk C++ s využitím Linuxového rozhraní `epoll`. Bližší informace o protokolu následují dále.

4.6.4 Long polling komunikace

Webový klient a komunikační server pro svou real-time komunikaci využívají metodu long polling. Průběh a aktéry tohoto protokolu popisuje obrázek 4.4. Části systému jsou

1. Klient – pomocí jednoduchého rozhraní odesílá požadavky a přijímá zprávy
2. Klientský adaptér – přijímá požadavky na odeslání dat od klienta, dekóduje příchozí data ze serveru a předává je klientovi
3. Serverový adaptér – dekóduje příchozí požadavky od klienta, předává je serveru, a odesílá zprávy klientovi
4. Server – přijímá požadavky od klienta a odesílá zpět zprávy

Adaptéry skrývají detaily implementace před klientem a serverem. Důvodem je jednoduchost přístupu k datům (jak na serveru, tak na klientovi stačí pouze funkce pro odeslání a přijmutí dat) a snížení komplikací při případné změně detailů protokolu.

Klient a server tedy komunikují jen skrze jednoduché rozhraní poskytované adaptérem; komunikace mezi adaptéry již probíhá metodou long polling:

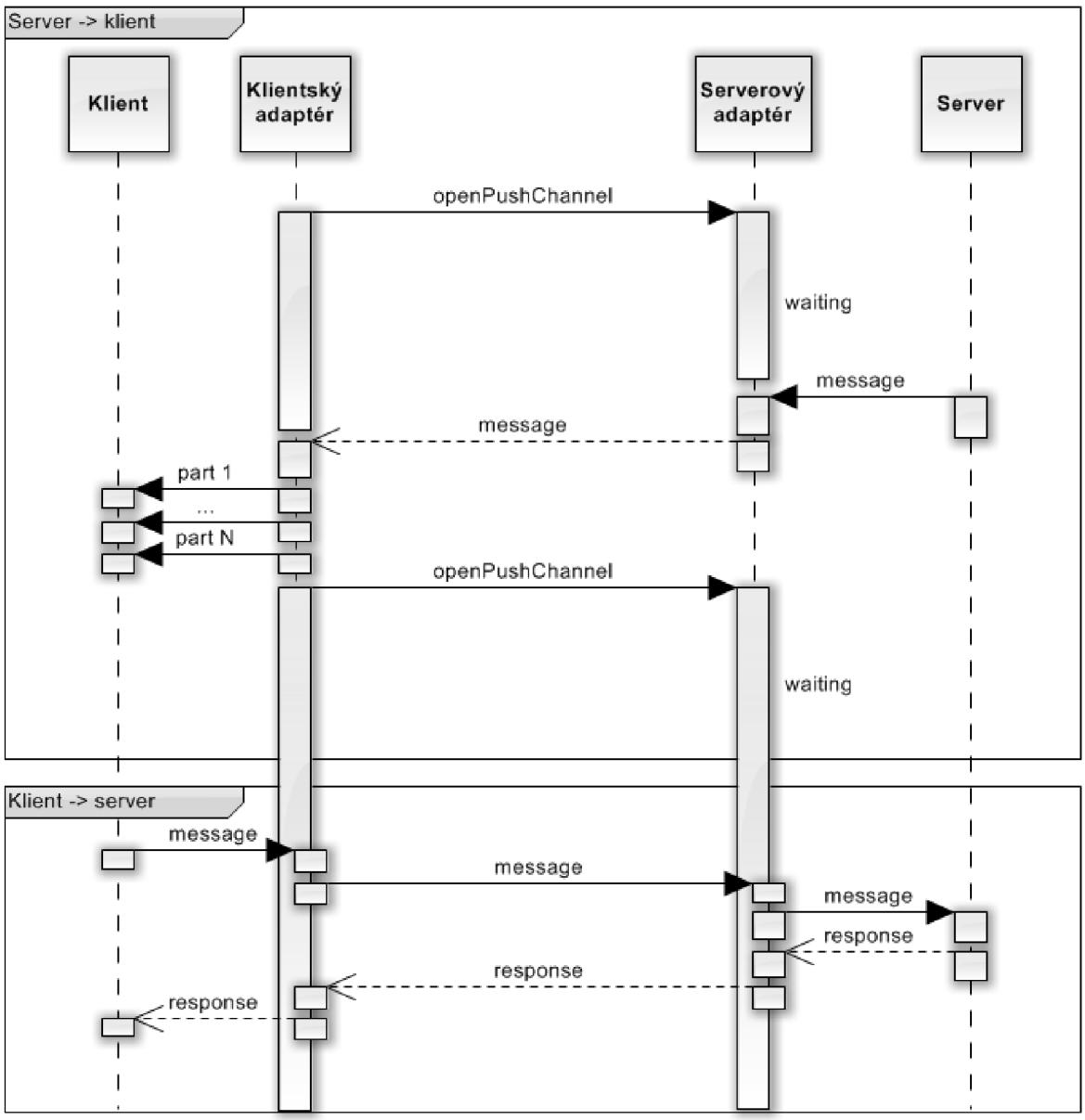
1. Klientský adaptér zašle asynchronní požadavek na server pomocí `XMLHttpRequest`, ale zatím nedostane odpověď
2. Serverový adaptér čeká, dokud server neindikuje potřebu poslat klientovi data, poté odešle klientskému adaptéru odpověď zakódovanou ve formátu JSON
3. Jakmile klientský adaptér obdrží data od serveru a zpracuje je, okamžitě otevře další `XMLHttpRequest` připojení k serveru, které zase jako v bodě 2 čeká na odpověď
4. Pokud klient potřebuje zaslat data serveru, zakóduje je pomocí metody GET v dalším `XMLHttpRequest` požadavku.

Protože je long polling protokol primárně určen k přenosu dat ze serveru ke klientovi, zprávy od klientského adaptéru serverovému adaptéru jsou enkódovány v HTTP GET `XMLHttpRequest` požadavku. Příkladem může být zpráva pro připojení do aliančního chatu:

`command.php?SID=XYZ&target=CHAT&parameter=ALLIANCE&action=join`

kde `SID` označuje session ID identifikující klienta, `target` označuje cílový modul, `parameter` označuje typ chatu a `action` označuje akci, kterou má server provést (přihlásit

uživatele). Odpovědí je JSON objekt ve formátu `{"result":"RES"}`, kde RES může nabývat hodnot OK (požadavek byl přijat) nebo ERR (při přijetí došlo k chybě).



Obr. 4.5: Long polling protokol

Zprávy od serverového adaptéru klientskému jsou enkódovány ve formátu JSON jako pole objektů. Typický formát objektu vypadá následovně:

```
{"protocol":"CHAT","type":"MESSAGE","fromid":"X","fromname":"Player","timestamp":"Y","message":"Message text"}
```

Klientský adaptér JSON data dekóduje, rozdělí pole objektů na jednotlivé položky, a ty předává klientovi jako příchozí zprávy.

5 Implementace

Tato kapitola se zabývá popisem implementace jednotlivých komponent systému, mezi něž patří klientská část, komunikační server a webový server.

5.1 Klientská část

Komponenty klientské části lze rozdělit na grafické uživatelské rozhraní a mezivrstvu, která se stará o uživatelský vstup nebo interakci s některými prvky grafického uživatelského rozhraní.

5.1.1 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní je implementováno podle specifikací popsaných v sekci 4.5.1. Základem systému je hrací plocha (oblast 1 v obrázku 5.1) obsahující všechny objekty jako jsou mateřské lodě hráčů, asteroidy, struktury NPC lodě, plošné efekty, plánované akce, výsledky akcí nebo poznámky gildy. Pomocí tlačítek vedle minimapy (oblast 2 v obrázku 5.1) lze zobrazovat a skrývat některé informace na ploše jako je čtvercová síť, výsledky akcí, plošné efekty nebo poznámky gildy. Pohyb mapy je realizován metodou táhni-a-pusť (drag-and-drop) pomocí YUI Drag & Drop Utility (`YAHOO.util.DD`).

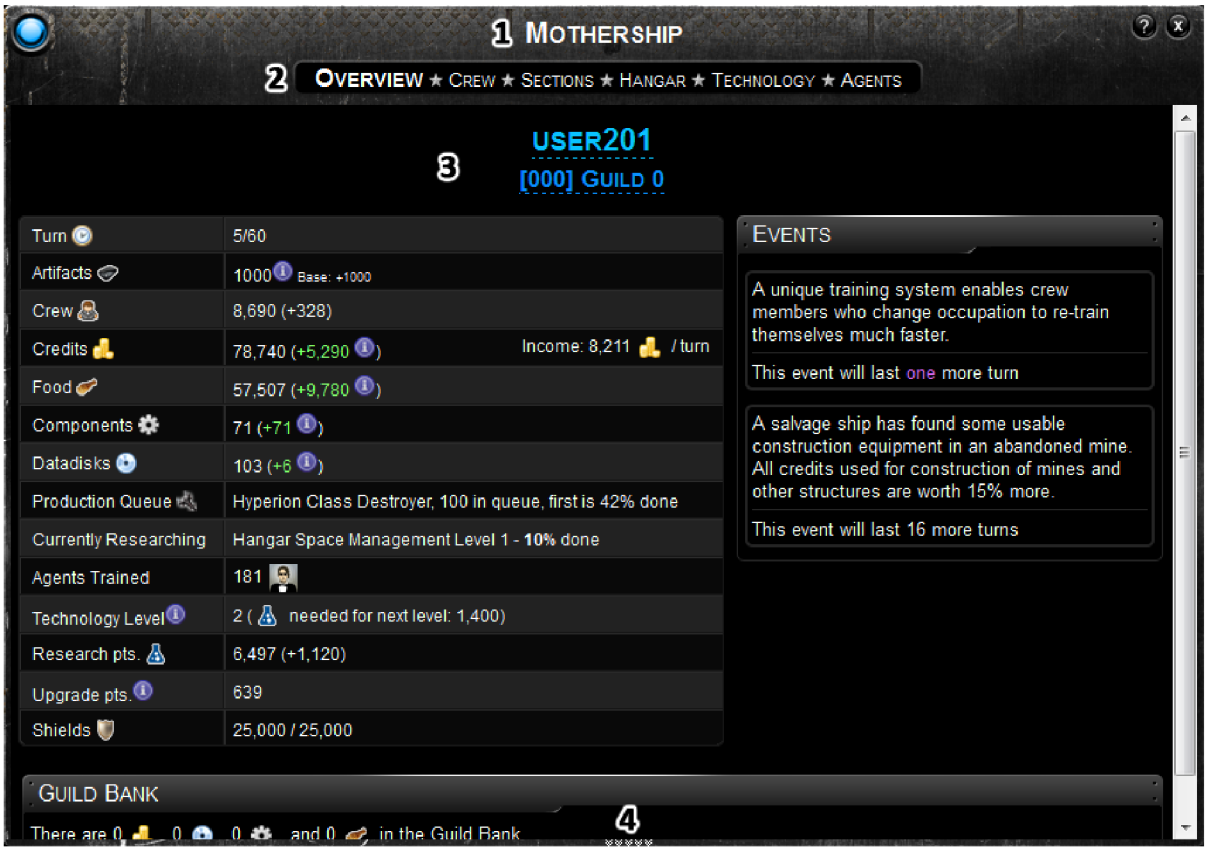
Obr. 5.1: Rozložení GUI prvků

Samotná minimapa obsahuje zmenšenou herní mapu, na které lze vidět aktuální polohu herní plochy a polohu hráče a členů gildy. Kliknutím na minimapu nebo potáhnutím (drag-and-drop) obdélníku znázorňujícího aktuální polohu hrací plochy dojde k přesunu herní plochy. I zde je použita utilita YUI Drag & Drop (`YAHOO.util.DD`). Dynamická část hrací plochy a minimapy je implementována v modulu `map.js`.

Hlavní menu (oblast 3 v obrázku 5.1) se skládá z ikon, kliknutím na něž se otevře v okně (obrázek 5.2) další obsah jako jsou informace o mateřské lodi, gildě, plánovaných i proběhlých akcích, nápověda, statistiky, nastavení nebo soukromé zprávy.

V levé části informační lišty (oblast 4 v obrázku 5.1) se nachází současný stav surovin – kreditů, jídla, komponent, datadisků, výzkumných bodů a žetonů. V pravé části lze vidět tlačítka s chaty a počtem online přátel a členů gildy, kliknutím na něž se otevře vyskakovací prvek (oblast 5 v obrázku 5.1) se seznamem hráčů nebo chatů. Kliknutím na položku seznamu se otevře okno instantní komunikace, pomocí kterého lze v reálném čase komunikovat s ostatními hráči.

Ikona nacházející se v pravém rohu informační lišty skryje všechna otevřená okna, takže je viditelná celá hrací plocha. V případě, že už jsou všechna okna skryta, způsobí kliknutí na tuto ikonu zpětné zobrazení všech skrytých oken. Jedná se o podobnou funkcionalitu jako je ikona „Zobrazit plochu“ v OS Windows.



Obr. 5.2: Rozložení prvků v okně

Velká část informací se zobrazuje v oknech, která se otevírají většinou z hlavního menu. Okno jsou postavena na widgetu `YAHOO.widget.Panel`, který poskytuje základní strukturu graficky upravenou pomocí kaskádových stylů. Tento widget také zajišťuje posun okna pomocí YUI Drag & Drop Utility (`YAHOO.util.DD`) navázané na titulní lištu.

Okna se skládají z několika základních částí (viz obrázek 5.2):

1. Titulní lišta – zde se nachází název okna, ikona pro zavření okna a ikona nápovědy. Kliknutí na ikonu nápovědy otevře v dalším okně prohlížeče novou stránku, kde se nachází relevantní nápověda k obsahu aktuálního okna. Ikona nápovědy je nepovinná a nemusí být přítomna na všech oknech.
2. Lokální menu – každé okno obsahuje informace z jednoho tématu, které má zpravidla několik podtémat. Lokální menu obsahuje seznam podtémat k danému tématu. Kliknutím na položku menu se ve stejném okně otevře dané podtéma. Například v okně Loď jsou podtémata Přehled, Posádka, Sekce, Hangár, Technologie a Agenti.
3. Hlavní část okna – zde se zobrazuje samotný obsah okna. Pokud obsah přesáhne výšku okna, objeví se vertikální posuvník.
4. Prvek pro změnu výšky – každé okno má určenu maximální výšku obsahu, většinou 500 pixelů. Potáhnutím prvku pro změnu výšky lze tuto výšku libovolně měnit, minimální hodnota však je vždy 250 pixelů. Maximální hodnota není omezena. Tento prvek je založen na widgetu `YAHOO.util.Resize`.

5.1.2 Mezivrstva

Mezivrstva je tvořena množstvím JavaScript modulů starajících se o interaktivní akce, zakládání akcí, přesun posádky nebo komunikaci se serverem.

Modul `window.js` zapouzdřuje přístup k jednotlivým oknům založeným na widgetu `YAHOO.widget.Panel`. Obsahuje také metody pro různá nastavení parametrů oken jako je šířka, titulek, pozice, obsah nebo lokální menu.

Modul `window_manager.js` implementuje správce oken, který se stará o relativní pozice jednotlivých oken, správné překrytí dříve aktivovaných oken později aktivovanými, zobrazení nebo skrytí okna podle názvu, nastavení obsahu okna přímo nebo přes XHR či skrytí nebo zobrazení všech oken. Každé okno je jednoznačně identifikované řetězcem.

Modul `ajax_handlers.js` obsahuje kód pro komunikaci se serverem. Kromě funkcí pro načtení a zpracování dat ze serveru implementuje třídu `UpdateManager`, která se stará o periodické obnovování mapy, zjišťování počtu nových soukromých zpráv nebo požadavků na přátelství.

V modulu `map.js` se nachází kód pro práci s mapou a minimapou jako je pohyb, zobrazení objektů nebo interakce s různými typy objektů. Stěžejní úlohou je zobrazování objektů na mapě při pohybu, protože naivní implementace (přidání všech objektů na mapě do DOM stromu) je příliš

pomalá i pro malý počet objektů. První optimalizace zobrazení objektů na mapě spočívá ve vykreslování pouze objektů v bezprostředním okolí aktuální pozice hrací plochy, a to vždy když dojde k pohybu plochy. I tato optimalizace však byla nedostatečná, druhá optimalizace proto rozděluje mapu na čtvercové oblastí a objekty k zobrazení se vybírají pouze z nejbližší oblasti, což celý proces dále zrychluje. Obě optimalizace mapy a jejich efektivitu blíže popisuje sekce 6.2.

Modul `mouseover.js` implementuje okno zobrazující informace o objektech na mapě, jako jsou hráči, asteroidy, struktury nebo NPC lodě.

Modul `attack.js` implementuje okno zobrazující akce, které může hráč provést. Mezi akce patří vojenské útoky a obrany, stavba struktur, oprava štítů, akce lodí vojenské policie, infiltračních a výzvědných lodí nebo obchodních válek. Toto okno dynamicky počítá, kolik lodí a kterého typu může hráč na akci poslat, zda má dostatek pilotů aj.

Modul `im.js` zprostředkovává spojení s komunikačním serverem, funguje tedy jako klientský adaptér long polling komunikačního protokolu (viz sekce 4.6.4). Obsahuje také třídy pro příjem a posílání instantních zpráv pro uživatele i chaty, a třídu implementující Texas hold 'em poker klienta.

Některé moduly zajišťují interaktivitu obsahu oken. Jde hlavně o přepočítávání hodnot, zobrazování nebo skrývání prvků stránky, což zrychluje odezvu na uživatelské podněty a omezuje zbytečnou komunikaci se serverem. Mezi tyto moduly patří `forums.js` pro fóra, `tech.js` pro technologie, `ships.js` pro hangár, nebo `crew.js` pro posádku.

Modul `tutorial.js` implementuje počáteční tutoriál, který vysvětluje uživateli základní pravidla hry.

Modul `common.js` obsahuje některé funkce z projektu *phpjs* [28] (dostupné pod MIT a GPLv2 licencí). Projekt *phpjs* implementuje nejčastěji používané funkce z jazyka PHP v JavaScriptu, což vede k rychlejšímu vývoji při potřebě duplikovat některý PHP kód také do klientské části.

5.2 Serverová část

Implementace serverové části sestává z programování PHP skriptů pro webový server, kompletní implementace komunikačního serveru a utility pro výpočet nového kola.

5.2.1 Webový server

PHP skripty dostupné přes protokol HTTP slouží jako rozhraní, které poskytuje služby pro klientskou část systému. Dle návrhu popsaného v sekci 4.6.1 byly implementovány PHP stránky obsluhující dva základní typy požadavků: na zjištění informací a na provedení akce.

Pro přístup k databázi je použita upravená verze třídy `database.class.php` [30] dostupná pod licencí GPLv3.

Požadavky na zjištění informací

Požadavky na zjištění informací lze rozdělit na dvě části: požadavky na obsah ve formátu HTML a požadavky na JSON data ze serveru. Stránky s výstupem ve formátu HTML poskytují výhradně obsah pro okna, zatímco JSON data jsou používána klientskou mezivrstvou ke generování informačních oken, případně aktualizaci mapy nebo informační lišty.

Stránky poskytující HTML kód pro obsah oken jsou organizovány podle následujícího klíče: každé okno tvoří jeden tematický celek, který je rozdělen na několik podoken přístupných z lokálního menu. Jednomu oknu přísluší vždy jedna hlavní stránka, která je volaná požadavkem ve tvaru `<název_okna>.php?sub=<podokno>`.

Hlavní stránka se stará hlavně o autentizaci uživatele, spojení s databází s případným načtením potřebných dat, výpis lokálního menu a vložením podstránek. Mezi hlavní stránky patří `actions.php`, `forums.php`, `friends.php`, `messages.php`, `mothership.php`, `politics.php`, `premium.php`, `results.php`, `settings.php`, `statistics.php` nebo `trade.php`.

Ke každé hlavní stránce patří několik podstránek, které se starají o generování dat k zobrazení jednotlivých podoken. Nejsou nikdy volány přímo, ale vždy vkládány do hlavní stránky v závislosti na hodnotě parametru `sub`. Například oknu Loď odpovídá stránka `mothership.php`, která v závislosti na parametru `sub` vkládá obsah podstránky `m_overview.php` pro podstránku Přehled, `m_crew.php` pro podstránku Posádka, `m_sections.php` pro stránku Sekce, `m_hangar.php` pro stránku Hangár, `m_technology.php` pro stránku Technologie a `m_agents.php` pro stránku Agenti.

Stránky poskytující JSON data vrací strukturovaná data, která nejsou přímo obsahem oken. Mezi takové stránky patří `map_show_asteroid.php` poskytující informace o asteroidu, budovách na něm postavených a jejich majitelích, `map_show_user.php` poskytující informace o uživateli, `get_alliance_from_name.php` vracející identifikační číslo aliance podle jména, nebo `get_update.php` vracející počet nových soukromých zpráv, žádostí o přátelství, informací o mapě aj.

Požadavky na provedení akce

Stránky obsluhující požadavky na provedení akce slouží pro uložení změn provedených na klientské straně, a to buď v oknech, nebo na mapě. Požadavky mají tvar `<název_akce>.php`, další parametry jsou předávány buď pomocí HTTP GET (v případě malého množství dat), nebo pomocí HTTP POST (v případě většího množství dat).

Všechny vstupní parametry jsou sanitizovány pomocí funkce `sanitize()` z modulu `sanitize.php`. Pro co nejvyšší bezpečnost jsou definovány 3 třídy sanitizace:

- Přetypování pomocí `(int)$variable`. Všechny číselné vstupní parametry jsou sanitizovány pomocí této třídy
- Překódování pomocí PHP funkce `htmlspecialchars()`. Tato funkce nahrazuje v textu některé znaky významné pro HTML, a tím brání vstupu jakéhokoliv HTML kódu. Dochází k překladům '&' na '&', '"' (dvojitá uvozovka) na '"', "'" (jednoduchá uvozovka) na ''', '<' na '<' a '>' na '>'. Pomocí této třídy jsou sanitizovány všechny vstupní parametry obsahující čistý text (bez HTML)
- Sanitizace HTML kódu pomocí nástroje *kses* [31] dostupného pod licencí GPLv2. Nástroj *kses* umožňuje nastavit množinu pravidel, které musí HTML tagy nebo jejich atributy ve vstupním textu splňovat, jinak jsou odfiltrovány. Tato třída sanitizace povoluje pouze HTML tagy `strong`, `div`, `em`, `ul`, `li`, `br`, `a`, `img` a `span`. Atribut `style` musí splňovat regulární výraz povolující CSS vlastnosti `font-size`, `text-decoration`, `color` a `background-color`. Pomocí této třídy jsou sanitizovány vstupní parametry obsahující HTML text.

Odpovědi stránek obsluhujících požadavky na provedení akce jsou tvořeny JSON daty v následujícím formátu:

Úspěch: `{"result": "OK", "message": <JSON objekt> (, "update": <JSON objekt>) }`

Chyba: `{"result": "ERR", "message": "Error message" }`

Ve zprávě o úspěchu má proměnná `result` hodnotu `OK` a proměnná `message` obsahuje JSON objekt s případnými daty, které se vrací jako výsledek akce. Proměnná `update` může obsahovat další data, která jsou transparentně zpracována klientskou mezivrstvou – může jít o aktuální stav surovin změněný akcí, nové objekty na mapě aj.

V chybové zprávě má proměnná `result` hodnotu `OK` a proměnná `message` obsahuje řetězec s popisem chyby.

Mezi stránky obsluhující požadavky na uložení změn provedených v oknech patří například `crew_change.php` ukládající podstránku Posádka stránky Loď, `friends_change.php` ukládající stránku Přátelé, `message_add.php` posílající zprávy ze stránky Zprávy, nebo `market.php` ukládající stránku Obchodní síť. Mezi stránky obsluhující požadavky na uložení změn provedených na mapě patří například `attack_add.php` zakládající akci nebo přidávající jednotky do už existující akce, nebo `note_post.php` přidávající novou poznámku gildy.

5.2.2 Komunikační server

Komunikační server je implementován v jazyce C++ pro OS Linux. Jde o HTTP 1.0 server implementující long polling protokol popsaný v sekci 4.6.4, který poskytuje služby instantních zpráv, chatů a hry Poker. Protože je potřeba udržovat stovky otevřených spojení, využívá server Linuxové rozhraní `epoll`, díky kterému je možné obsluhovat velké množství neblokujících soketů v jednom vlákně. Diagram tříd komunikačního serveru lze kvůli značné velikosti nalézt na DVD příloze v adresáři `diagram`.

Jádrem síťové komunikace je třída `EpollSrv`, která jednak obsluhuje sokety, druhak slouží k časování událostí. Tyto dvě funkce jsou spolu nutně spojeny, protože rozhraní `epoll` blokuje zpracování kódu, dokud nedojde na jednom ze sledovaných soketů k události nebo dokud nevyprší specifikovaný časový interval.

Události pro časovač jsou ve třídě `EpollSrv` přidávány metodou `addTimer` a odebírány metodou `removeTimer`, při volání události je zavolána metoda `onTimer` rozhraní `TimerListener`. Přidání soketu na monitorování čtení je realizováno metodou `addRead`, odebrání metodou `removeRead` a připravenost ke čtení dat je indikována voláním metody `readSelected` rozhraní `ReadListener`. Obdobně fungují metody `addWrite` a `removeWrite` pro monitorování čtení a `addExcept` a `removeExcept` pro monitorování chyb na soketu.

Dekódování long polling protokolu provádí třída `IOWorker`, jejíž instance je vytvořena pro každé příchozí spojení. Tato třída tedy funguje jako serverový adaptér, který dekódované požadavky předá třídě `IMHandler`, jedná-li se o instantní zprávy, `ChatHandler`, jedná-li se o chat, nebo `PokerHandler`, jedná-li se o poker. Tyto třídy poté vstup zpracují a případný výstup předají zpět třídě `IOWorker`, která je v nejkratší možné době předá klientovi.

Třída `UserInfo` uchovává informace o připojených uživatelích, obstarává autentizaci a poskytuje informace o příslušnosti uživatelů ke gildě a alianci, přátelích aj. Tato třída je využívána třídami `IMHandler`, `ChatHandler` a `PokerHandler`.

5.2.3 Výpočet nového kola

Každých 24 hodin je třeba vypočítat nové kolo. Utilita pro výpočet nového kola je napsána v jazyce C++ a využívá knihovny MySQL++ [29]. Nejprve jsou provedeny akce naplánované hráči, jako jsou vojenské útoky a obrany, oprava štítů, akce lodí vojenské policie, infiltračních a výzvědných lodí, stavba struktur, obchodní války nebo skoky mateřských lodí. Poté jsou vypočítány těžby dolů na asteroidech, jsou přidány nové asteroidy, odebrány vytěžené, a je inkrementován čítač dne.

6 Testy uživatelského rozhraní

V této kapitole se budeme věnovat testování uživatelského rozhraní webového systému z hlediska rychlosti odezvy a propustnosti. Každý experiment je desetkrát zopakován a jako výsledek je brán průměr těchto deseti experimentů. Experimenty byly prováděny těsně po sobě.

Všechny následující testy byly provedeny na počítači s CPU Intel Core i5 M430 @ 2,27GHz, 8GB DDR3 1066 SO-DIMM, OS Windows 7 Professional 64bit. Použity byly prohlížeče Microsoft Internet Explorer 8 v módu Standards mode (jádro Trident 4.0), Mozilla Firefox 3.6 a Google Chrome 8.0.

6.1 Rychlost odezvy uživatelského rozhraní

Tato sekce se zabývá testováním rychlosti odezvy uživatelského rozhraní. Budeme měřit jednak rychlost zpracování obsahu oken, druhak rychlost překreslení oken s různým počtem HTML elementů.

6.1.1 Sada testů 1: Rychlost zpracování obsahu oken

V této sadě testů je měřena rychlost naplnění oken HTML kódem, tedy zpracování HTML kódu a sestavení DOM stromu prohlížečem. V oknech může být zobrazeno potenciálně velké množství obsahu, je proto vhodné zajistit, aby rychlost zobrazení okna byla pro uživatele dostatečně vysoká. Za dostatečně rychlé zobrazení okna byla určena hodnota kratší než 300 ms, delší hodnota je považována za nedostatečnou.

Jako míra náročnosti byl zvolen počet HTML elementů, které sestávají z náhodné posloupnosti jedno- až trojciferných čísel a ikon ve formátu PNG s velikostí 15x15 pixelů. Experimenty byly měřeny s přesností na milisekundy, což je největší přesnost, kterou lze pomocí JavaScriptu dosáhnout. Měřen byl čas od získání dat ze serveru pomocí `XMLHttpRequest` po zpracování dat vložených do okna pomocí `innerHTML`.

Experiment	Google Chrome [ms]	Mozilla Firefox [ms]	IE8 [ms]
1	9	10	25
2	10	11	25
3	9	11	26
4	11	12	27
5	11	12	29
6	10	13	30
7	10	14	31
8	11	15	31
9	9	15	34

10	11	16	33
Průměr	**10,1**	**12,9**	**29,1**

Tab. 6.1: Rychlost zpracování HTML kódu s 50 elementy

Z tabulky 6.1 lze vidět, že zpracování kódu s 50 HTML elementy proběhne prakticky okamžitě ve všech testovaných prohlížečích a je dostatečně rychlé. Největší část doby otevření okna tedy bude tvořit načtení dat ze serveru.

U prohlížečů Mozilla Firefox a Internet Explorer lze vidět mírný růst doby zpracování s téměř každým novým experimentem. V prohlížeči Mozilla Firefox vzrostla během deseti experimentů doba zpracování o 6 ms (60%), v prohlížeči Internet Explorer o 8 ms (32%). Tento trend budeme zkoumat v dalších testech.

Experiment	Google Chrome [ms]	Mozilla Firefox [ms]	IE8 [ms]
1	40	33	95
2	37	35	97
3	39	37	101
4	40	39	107
5	39	40	109
6	41	45	115
7	41	47	121
8	43	48	127
9	41	52	129
10	39	54	131
Průměr	**40,0**	**43,0**	**113,2**

Tab. 6.2: Rychlost zpracování HTML kódu s 250 elementy

Zpracování kódu s 250 elementy (tabulka 6.2) trvalo oproti kódu s 50 elementy (5x více elementů) v prohlížeči Google Chrome 4x déle, v prohlížeči Mozilla Firefox 3,3x déle a v prohlížeči Internet Explorer 3,8x déle. Během 10 experimentů také vzrostla doba zpracování v prohlížeči Mozilla Firefox o 21 ms (64%) a v prohlížeči Internet Explorer o 36 ms (38%).

I v tomto testu ve všech prohlížečích trvá zpracování kódu relativně krátkou dobu, a i když už ji nelze zcela zanedbat, stále nemá výrazný vliv na odezvu uživatelského rozhraní a otevření okna je dostatečně rychlé.

Experiment	Google Chrome [ms]	Mozilla Firefox [ms]	IE8 [ms]
1	81	59	157
2	83	64	178
3	85	68	182
4	83	78	192
5	87	87	206
6	89	88	208

7	88	92	230
8	90	97	244
9	92	98	248
10	84	105	257
Průměr	**86,2**	**83,6**	**210,2**

Tab. 6.3: Rychlost zpracování HTML kódu s 500 elementy

Tabulka 6.3 ukazuje rychlost zpracování kódu s 500 elementy. Oproti kódu s 250 elementy (2x více elementů) trvalo zpracování v prohlížeči Google Chrome 2,2x déle, v prohlížeči Mozilla Firefox 1,9x déle a v prohlížeči Internet Explorer 1,9x déle. V průběhu 10 experimentů vzrostla doba zpracování v prohlížeči Mozilla Firefox o 46 ms (78%) a v prohlížeči Internet Explorer o 100 ms (64%). Stále je však doba zpracování obsahu oken ve všech prohlížečích dostatečně rychlá.

Průměrná doba zpracování v prohlížeči Internet Explorer (210 ms) je pozorovatelná uživatelem a při rychlé odezvě serveru může tvořit značnou část doby otevření okna.

Experiment	Google Chrome [ms]	Mozilla Firefox [ms]	IE8 [ms]
1	298	182	543
2	305	211	667
3	318	221	617
4	313	235	671
5	305	229	682
6	304	242	717
7	318	252	788
8	309	286	783
9	310	282	790
10	323	291	799
Průměr	**310,3**	**243,1**	**705,7**

Tab. 6.4: Rychlost zpracování HTML kódu s 1500 elementy

Zpracování kódu s 1500 elementy (viz tabulka 6.4) trvalo oproti kódu s 500 elementy (3x více elementů) v prohlížeči Google Chrome 3,6x déle, v prohlížeči Mozilla Firefox 2,9x déle a v prohlížeči Internet Explorer 3,6x déle. Během 10 experimentů vzrostla doba zpracování v prohlížeči Mozilla Firefox o 109 ms (60%) a v prohlížeči Internet Explorer o 256 ms (47%).

Průměrné doby zpracování v prohlížečích Google Chrome a Mozilla Firefox jsou pozorovatelné, ale stále dostatečně rychlé na to, aby měl uživatel dojem téměř okamžité odezvy. Doba zpracování v prohlížeči Google Chrome je však již na hranici dostatečné rychlosti. Průměrná doba zpracování v prohlížeči Internet Explorer (705 ms) již dostatečně krátká není a brání rychlé interakci.

Experiment	Google Chrome [ms]	Mozilla Firefox [ms]	IE8 [ms]
1	1300	639	1897

2	1221	682	1991
3	1220	779	2072
4	1363	857	2437
5	1250	885	2544
6	1291	934	2713
7	1313	960	2806
8	1289	1008	2942
9	1352	1052	3103
10	1368	1107	3233
Průměr	**1296,7**	**890,3**	**2573,8**

Tab. 6.5: Rychlost zpracování HTML kódu s 5000 elementy

Z tabulky 6.5 lze vidět rychlost zpracování kódu s 5000 elementy. Oproti kódu s 1500 elementy (3,3x více elementů) trvalo zpracování v prohlížeči Google Chrome 4,2x déle, v prohlížeči Mozilla Firefox 3,7x déle a v prohlížeči Internet Explorer 3,7x déle. V průběhu 10 experimentů vzrostla doba zpracování v prohlížeči Mozilla Firefox o 468 ms (73%) a v prohlížeči Internet Explorer o 1336 ms (70%).

U tohoto testu je průměrná doba zpracování nedostatečně dlouhá již ve všech prohlížečích a pro uživatele je nevyhovující.

Lze konstatovat, že doba zpracování s přibývajícími experimenty stoupá v prohlížečích Mozilla Firefox a Internet Explorer ve všech testech. Taktéž stoupá spotřeba paměti prohlížečů z počátečních 80 MB až k 160-200 MB RAM.

U prohlížeče Mozilla Firefox dochází během 10-30 sekund po provedení deseti experimentů k snížení alokované paměti až k původní hodnotě, přičemž při opětovném provedení experimentů opět dochází k růstu doby zpracování kódu od původní nízké hodnoty.

Prohlížeč Internet Explorer 8 paměť začne uvolňovat paměť po více než 30 sekundách po provedení všech deseti experimentů, ale ke snížení až k původní hodnotě nedojde, ustálí se kolem hodnoty 150 MB. Při opětovném provedení experimentů dochází opět k růstu doby zpracování, ne však od původní nejnižší hodnoty, ale od vyšší hodnoty odpovídající přibližně průměru všech deseti experimentů.

Z výsledků této sady testů můžeme vyvodit následující závěry:

- Časté otevírání oken v relativně krátkém časovém intervalu (desítky vteřin) způsobuje v prohlížečích Mozilla Firefox a Internet Explorer zpomalení doby reakce. I když rychlé otevírání oken není typické chování uživatele, je na tento fakt třeba brát zřetel.
- Pokud je obsah zobrazovaný v okně složitý (obsahuje tisíce HTML elementů), může jeho zpracování trvat i několik vteřin, během kterých prohlížeč nereaguje. Proto je vhodné vyhnout se používání takto velkých stránek a obsah rozdělit, například použitím stránkování.

- Největší rychlosti dosáhly prohlížeče Mozilla Firefox a Google Chrome, Internet Explorer byl oproti nim přibližně 3x pomalejší. Proto je vhodné při optimalizacích brát zřetel hlavně na Internet Explorer jakožto nejpomalejší prohlížeč.

6.1.2 Sada testů 2: Rychlost překreslení oken

Tato sada testů se zabývá měřením rychlosti vykreslení okna v závislosti na počtu HTML elementů, které tvoří obsah okna. Konkrétně zkoumáme, zda je při posunu okno překreslováno dostatečně plynule.

Za ideálně plynulý posun byla zvolena hodnota 25 snímků za sekundu (FPS, frames per second), což odpovídá hodnotě 40 milisekund na vykreslení snímku. Za dostatečně plynulý posun (působí trhaně, ale stále je použitelný) byla zvolena hodnota 10 FPS, což odpovídá hodnotě 100 milisekund na vykreslení snímku. Nižší hodnota FPS je považována za nedostatečnou.

Stejně jako v sadě testů 1 byly za obsah okna zvoleny HTML elementy, které sestávají z náhodné posloupnosti jedno- až trojciferných čísel a ikon ve formátu PNG s velikostí 15x15 pixelů. Experimenty byly měřeny s přesností na milisekundy a jako výsledek je brána průměrná hodnota FPS během 30 sekund pohybu okna.

Experiment	Google Chrome [FPS]	Mozilla Firefox [FPS]	IE8 [FPS]
1	47,8	35,5	24,2
2	48,8	36,1	23,5
3	46,3	35,7	23,4
4	49,3	33,3	22,9
5	49,0	35,2	23,6
6	48,3	35,5	23,6
7	44,4	34,8	23,1
8	48,8	35,3	22,9
9	50,8	35,0	24,3
10	44,6	35,1	23,8
Průměr	**47,8**	**35,2**	**23,5**

Tab. 6.6: Rychlost překreslování okna s 50 HTML elementy

V tabulce 6.6 vidíme, že posun okna s obsahem 50 HTML elementů je v prohlížečích Google Chrome a Mozilla Firefox ideálně plynulý (průměrná hodnota FPS je 47,8 respektive 35,2). V prohlížeči Internet Explorer dosahuje snímková frekvence průměrně 23,5 FPS, což sice nelze označit za ideálně plynulý posun, ale jde o hodnotu, která je této hranici velmi blízká. Celkově tedy lze v tomto testu rychlost překreslení oken označit za velmi vysokou ve všech testovaných prohlížečích.

Experiment	Google Chrome [FPS]	Mozilla Firefox [FPS]	IE8 [FPS]
1	36,1	23,9	15,2
2	35,6	23,6	15,4
3	35,7	23,5	15,3
4	35,6	24,2	15,2
5	36,0	24,3	15,4
6	36,5	23,9	15,6
7	35,8	24,2	15,2
8	37,2	25,1	15,2
9	37,5	23,7	15,0
10	36,4	23,0	15,3
Průměr	**36,2**	**23,9**	**15,3**

Tab. 6.7: Rychlost překreslování okna s 250 HTML elementy

Posun okna s 250 HTML elementy (viz tabulka 6.7) je v prohlížeči Google Chrome stále ideálně plynulý (průměrná hodnota 36,2 FPS) a v prohlížeči Mozilla Firefox na hranici ideální plynulosti (průměrně 23,9 FPS), v prohlížeči Internet Explorer se však začíná blížit k dolní hranici (průměrně 15,3 FPS). Rychlost překreslení oken je tedy ve všech prohlížečích uspokojivá, v Internet Exploreru už však je pohyb viditelně trhaný, což však stále nebrání pohodlné interakci.

Experiment	Google Chrome [FPS]	Mozilla Firefox [FPS]	IE8 [FPS]
1	28,1	16,6	10,4
2	27,5	16,6	10,6
3	28,7	16,8	10,4
4	28,2	16,4	10,2
5	27,8	16,9	10,5
6	28,0	16,8	10,3
7	27,8	16,7	10,5
8	27,6	16,3	10,3
9	27,1	15,9	10,2
10	28,1	16,5	10,3
Průměr	**27,9**	**16,5**	**10,4**

Tab. 6.8: Rychlost překreslování okna s 500 HTML elementy

Zatímco rychlost překreslování okna s 500 elementy (viz tabulka 6.8) je v prohlížeči Google Chrome stále ideálně plynulá (průměrně 27,9 FPS) a v prohlížeči Mozilla Firefox dostatečně plynulá (průměrně 16,5 FPS), v prohlížeči Internet Explorer už je na hranici plynulosti (10,4 FPS).

Experiment	Google Chrome [FPS]	Mozilla Firefox [FPS]	IE8 [FPS]
1	24,0	12,8	8,5
2	23,9	12,8	8,7
3	25,2	12,8	8,6

4	25,1	12,9	8,2
5	23,3	12,8	8,5
6	25,7	12,8	8,6
7	27,0	12,9	8,5
8	25,3	12,9	8,5
9	27,2	12,6	8,4
10	24,8	12,8	8,6
Průměr	**25,2**	**12,8**	**8,5**

Tab. 6.9: Rychlost překreslování okna s 1500 HTML elementy

V tabulce 6.9 popisující výsledky testů rychlosti překreslování okna s 1500 HTML tabulky je prohlížeč Google Chrome na hranici ideální plynulosti (průměrná hodnota 25,2 FPS), zatímco prohlížeč Mozilla Firefox je téměř na hranici dostatečné plynulosti (průměrně 12,8 FPS). Internet Explorer však již je s průměrnými 8,5 FPS za hranicí dostatečné plynulosti posunu.

Experiment	Google Chrome [FPS]	Mozilla Firefox [FPS]	IE8 [FPS]
1	23,7	7,7	6,4
2	25,2	7,7	6,4
3	26,4	7,5	6,4
4	23,5	7,6	6,4
5	25,2	7,5	6,3
6	24,7	7,6	6,4
7	24,3	7,7	6,4
8	25,3	7,7	6,3
9	25,0	7,5	6,3
10	24,0	7,6	6,3
Průměr	**24,7**	**7,6**	**6,4**

Tab. 6.10: Rychlost překreslování okna s 5000 HTML elementy

V testu rychlosti překreslování s 5000 HTML (tabulka 6.10) dokáže prohlížeč Google Chrome jako jediný vykreslovat obsah okna dostatečně rychle (průměrná hodnota 24,7 FPS). Prohlížeče Mozilla Firefox a Internet Explorer jsou nedostatečně plynulé s hodnotami 7,6 FPS respektive 6,4 FPS, pohyb okna je velmi trhaný a pro uživatele velmi špatně použitelný.

Prohlížeč Google Chrome dosáhl v této sadě testů nejlepších výsledků, velmi vysokou snímkovou frekvenci udržel až do posledního testu s 5000 HTML elementy (24,7 FPS). Prohlížeč Mozilla Firefox dokázal dostatečně plynule překreslovat okna až do obsahu 1500 HTML elementů (12,8 FPS), zatímco prohlížeč Internet Explorer dosáhl nejhorších výsledků s dostatečně plynulým vykreslováním pouze do 500 HTML elementů (10,4 FPS).

Z výsledků této sady testů můžeme vyvodit následující závěry:

- Ideálně plynulého překreslení oken ve všech prohlížečích jsme dosáhli pouze v prvním testu s oknem s 50 HTML elementy. Dostatečně plynulého překreslení dosáhly všechny prohlížeče až do hranice 500 HTML elementů.
- Nejhorších výsledků dosáhl prohlížeč Internet Explorer, je proto vhodné brát zřetel hlavně na jeho limity, tedy maximální obsah okna 500 HTML elementů pro dosažení dostatečně rychlého překreslování.

6.1.3 Zhodnocení

Rychlost odezvy uživatelského rozhraní je velmi dobrá ve všech testovaných prohlížečích, pokud se systém řídí omezeními zjištěnými v sadě testů 1 a 2 – tedy okna obsahují maximálně stovky HTML elementů. S klesajícím počtem HTML elementů stoupá jak rychlost otevření, tak plynulost překreslení oken. Z testovaných prohlížečů dosáhl nejlepších výsledků prohlížeč Google Chrome následovaný prohlížečem Mozilla Firefox, zatímco nejhorších výsledků dosáhl prohlížeč Internet Explorer.

6.2 Propustnost uživatelského rozhraní

Tato sekce se zabývá zkoumáním limitů uživatelského rozhraní z hlediska propustnosti, konkrétně jde o počet objektů, které lze zobrazit na mapě při zachování dostatečně rychlé snímkové frekvence při pohybu mapy. Stejně jako v sekci 6.1.2 je za ideálně plynulý posun zvolena hodnota 25 FPS, za dostatečně plynulý posun hodnota 10 FPS a nižší snímková frekvence hodnota je považována za nedostatečnou.

Ze specifikace vyplývá, že by měl systém podporovat počet objektů na mapě v řádu desítek tisíc.

6.2.1 Sada testů 1: Bez optimalizace

V této sadě testů je použito naivní řešení bez optimalizací, kdy jsou všechny objekty na mapě vloženy do DOM stromu. Pro každý prohlížeč byly testovány mapy s 500 a 5000 objekty.

Experiment	Google Chrome [FPS]	Mozilla Firefox [FPS]	IE8 [FPS]
1	41,6	22,5	12,1
2	40,5	23,8	12,4
3	41,8	22,7	11,8
4	40,2	23,3	12,1
5	41,1	22,9	12,5
6	40,5	22,3	12,1
7	41,7	23,0	12,1
8	41,4	22,3	12,3

9	40,5	22,8	11,9
10	41,0	23,3	12,8
Průměr	**41,0**	**22,9**	**12,2**

Tab. 6.11: Rychlost překreslování mapy s 500 objekty bez optimalizace

Rychlost vykreslování mapy s 500 objekty bez optimalizací (tabulka 6.11) je v prohlížeči Google Chrome ideálně plynulá (41,0 FPS), v prohlížeči Mozilla Firefox dostatečně plynulá (22,9 FPS) a v prohlížeči Internet Explorer na hranici dostatečné plynulosti (12,2 FPS).

Experiment	Google Chrome [FPS]	Mozilla Firefox [FPS]	IE8 [FPS]
1	15,8	8,0	2,1
2	17,0	8,3	2,0
3	16,4	7,9	2,0
4	16,8	8,2	2,1
5	16,3	8,2	2,1
6	16,4	8,1	2,0
7	16,9	8,5	2,1
8	16,7	8,3	2,1
9	17,0	8,3	2,0
10	16,6	7,8	2,2
Průměr	**16,6**	**8,2**	**2,1**

Tab. 6.12: Rychlost překreslování mapy s 5000 objekty bez optimalizace

Z tabulky 6.12 vidíme, že pro 5000 objektů bez optimalizace dosáhl dostatečné plynulosti pouze prohlížeč Google Chrome s 16,6 FPS. V prohlížečích Mozilla Firefox (8,2 FPS) a Internet Explorer (2,1 FPS) není pohyb dostatečně plynulý. Vzhledem k tomu, že by měl systém plynule vykreslovat desítky tisíc objektů, bylo potřeba přistoupit k optimalizacím.

6.2.2 Sada testů 2: První optimalizace

Protože naivní řešení (vložení všech objektů do DOM stromu) nedosahuje potřebného výkonu, bylo třeba přistoupit k optimalizacím. První optimalizace spočívá v tom, že se do DOM stromu vloží vždy jen objekty, které lze aktuálně vidět na obrazovce. Při každém posunu mapy je třeba zjistit, které objekty jsou aktuálně viditelné, a zobrazit je.

Tato sada testů měří výkonnost této optimalizace, a to ve dvou částech: dobu, kterou trvá výběr aktuálně zobrazených objektů, a dobu, kterou trvá jejich vložení do DOM podstromu a vykreslení. Poté jsou obě části sečteny a je vypočítána snímková frekvence. V prvním testu se měří výkonnost pro 5000 objektů, v druhém pro 50 000 objektů na mapě.

Experiment	Výběr objektů [ms]	Vykreslení [ms]	Celkem [ms]	Celkem [FPS]
1	0,9	2,3	3,2	312,5
2	0,8	2,0	2,8	357,1
3	1,0	2,1	3,1	322,6
4	0,9	2,0	2,9	344,8
5	0,9	2,2	3,1	322,6
6	1,0	2,1	3,1	322,6
7	0,9	2,2	3,1	322,6
8	1,0	2,0	3,0	333,3
9	1,0	2,1	3,1	322,6
10	0,8	2,0	2,8	357,1
Průměr	**0,9**	**2,1**	**3,0**	**331,1**

Tab. 6.13: Rychlost překreslování mapy s 5000 objekty s první optimalizací v prohlížeči Google Chrome

Experiment	Výběr objektů [ms]	Vykreslení [ms]	Celkem [ms]	Celkem [FPS]
1	4,0	1,3	5,3	188,7
2	4,3	1,5	5,8	172,4
3	4,4	1,5	5,9	169,5
4	4,2	1,4	5,6	178,6
5	4,1	1,5	5,6	178,6
6	4,3	1,2	5,5	181,8
7	4,1	1,5	5,6	178,6
8	4,0	1,3	5,3	188,7
9	4,3	1,5	5,8	172,4
10	4,2	1,5	5,7	175,4
Průměr	**4,2**	**1,4**	**5,6**	**178,3**

Tab. 6.14: Rychlost překreslování mapy s 5000 objekty s první optimalizací v prohlížeči Mozilla Firefox

Experiment	Výběr objektů [ms]	Vykreslení [ms]	Celkem [ms]	Celkem [FPS]
1	7,5	2,5	10,0	100,0
2	7,6	2,6	10,2	98,0
3	7,6	2,7	10,3	97,1
4	7,5	2,6	10,1	99,0
5	7,4	2,4	9,8	102,0
6	7,5	2,5	10,0	100,0
7	7,5	2,3	9,8	102,0
8	7,5	2,4	9,9	101,0

9	7,4	2,7	10,1	99,0
10	7,6	2,5	10,1	99,0
Průměr	**7,5**	**2,5**	**10,0**	**99,7**

Tab. 6.15: Rychlost překreslování mapy s 5000 objekty s první optimalizací v prohlížeči Internet Explorer

Z tabulek 6.13, 6.14 a 6.15 lze vidět, že překreslování mapy s 5000 objekty s využitím první optimalizace je velmi rychlé ve všech prohlížečích. V prohlížeči Google Chrome trvá výběr objektů méně času než vykreslení, v ostatních prohlížečích je to naopak.

Díky optimalizacím bylo v jeden okamžik viditelných průměrně 30 objektů.

Experiment	Výběr objektů [ms]	Vykreslení [ms]	Celkem [ms]	Celkem [FPS]
1	17,4	7,8	25,2	39,7
2	17,1	7,7	24,8	40,3
3	17,7	7,6	25,3	39,5
4	16,9	7,7	24,6	40,7
5	17,0	7,8	24,8	40,3
6	17,4	7,7	25,1	39,8
7	17,4	7,8	25,2	39,7
8	17,2	7,6	24,8	40,3
9	17,0	7,6	24,6	40,7
10	16,8	7,8	24,6	40,7
Průměr	**17,2**	**7,7**	**24,9**	**40,2**

Tab. 6.16: Rychlost překreslování mapy s 50 000 objekty s první optimalizací v prohlížeči Google Chrome

Experiment	Výběr objektů [ms]	Vykreslení [ms]	Celkem [ms]	Celkem [FPS]
1	32,5	9,0	41,5	24,1
2	35,3	10,0	45,3	22,1
3	35,8	9,7	45,5	22,0
4	35,4	10,1	45,5	22,0
5	35,3	9,7	45,0	22,2
6	35,0	9,9	44,9	22,3
7	35,8	10,0	45,8	21,8
8	35,7	9,5	45,2	22,1

9	35,4	10,3	45,7	21,9
10	34,9	9,8	44,7	22,4
Průměr	**35,1**	**9,8**	**44,9**	**22,3**

Tab. 6.17: Rychlost překreslování mapy s 50 000 objekty s první optimalizací v prohlížeči Mozilla Firefox

Experiment	Výběr objektů [ms]	Vykreslení [ms]	Celkem [ms]	Celkem [FPS]
1	50,9	13,4	64,3	15,6
2	52,1	14,0	66,1	15,1
3	51,7	15,3	67,0	14,9
4	50,9	15,1	66,0	15,2
5	50,3	15,0	65,3	15,3
6	51,2	14,5	65,7	15,2
7	51,1	14,3	65,4	15,3
8	50,4	14,4	64,8	15,4
9	50,8	15,0	65,8	15,2
10	50,8	14,5	65,3	15,3
Průměr	**51,0**	**14,6**	**65,6**	**15,3**

Tab. 6.18: Rychlost překreslování mapy s 50 000 objekty s první optimalizací v prohlížeči Internet Explorer

Tabulky 6.16, 6.17 a 6.18 znázorňují výsledky rychlosti překreslování mapy s 50 000 objekty s využitím první optimalizace. V prohlížeči Google Chrome je překreslování velmi plynulé (průměrně 40,2 FPS), v prohlížečích Mozilla Firefox a Internet Explorer dostatečně plynulé (průměrně 22,3 respektive 15,3 FPS). Díky optimalizacím bylo v jeden okamžik viditelných průměrně 250 objektů.

V prohlížeči Google Chrome tvoří výběr objektů 69% času, v prohlížeči Mozilla Firefox 78% a v prohlížeči Internet Explorer 78%. Díky optimalizacím bylo v jeden okamžik viditelných průměrně 250 objektů, které bylo třeba vybrat z celkového počtu 50 000 objektů. To vysvětluje velký podíl výběru objektů na celkovém čase a ukazuje, kterým směrem se mají ubírat další optimalizace.

6.2.3 Sada testů 3: Druhá optimalizace

Druhá optimalizace vylepšuje první optimalizaci tím, že rozděluje mapu na čtvercové oblasti. Při vyhledávání objektů k zobrazení se potom prohledávají pouze aktuálně viditelné oblasti. Tato optimalizace by měla zrychlit výběr objektů až o 90%.

Sada testů 3 měří výkonnost druhé optimalizace, a to obdobně jako sada testů 2: dobu, kterou trvá výběr aktuálně zobrazených objektů, a dobu, kterou trvá jejich vložení do DOM podstromu a vykreslení. Poté jsou obě části sečteny a je vypočítána snímková frekvence. V prvním testu se měří výkonnost pro 50 000 objektů, v druhém pro 200 000 objektů na mapě.

Experiment	Výběr objektů [ms]	Vykreslení [ms]	Celkem [ms]	Celkem [FPS]
1	1,4	9,1	10,5	95,2
2	1,5	10,3	11,8	84,7
3	1,3	9,7	11,0	90,9
4	1,0	9,1	10,1	99,0
5	1,3	9,4	10,7	93,5
6	1,6	9,7	11,3	88,5
7	1,2	9,7	10,9	91,7
8	1,4	9,8	11,2	89,3
9	1,4	9,5	10,9	91,7
10	1,6	10,1	11,7	85,5
Průměr	**1,4**	**9,6**	**11,0**	**90,8**

Tab. 6.19: Rychlost překreslování mapy s 50 000 objekty s druhou optimalizací v prohlížeči Google Chrome

Experiment	Výběr objektů [ms]	Vykreslení [ms]	Celkem [ms]	Celkem [FPS]
1	4,1	6,8	10,9	91,7
2	4,4	7,7	12,1	82,6
3	4,9	8,0	12,9	77,5
4	4,6	8,1	12,7	78,7
5	4,6	7,7	12,3	81,3
6	4,6	7,4	12,0	83,3
7	4,8	7,9	12,7	78,7
8	4,6	8,0	12,6	79,4
9	4,8	9,4	14,2	70,4
10	4,9	8,5	13,4	74,6
Průměr	**4,6**	**8,0**	**12,6**	**79,5**

Tab. 6.20: Rychlost překreslování mapy s 50 000 objekty s druhou optimalizací v prohlížeči Mozilla Firefox

Experiment	Výběr objektů [ms]	Vykreslení [ms]	Celkem [ms]	Celkem [FPS]
1	9,0	14,3	23,3	42,9
2	9,4	14,6	24,0	41,7
3	9,3	14,6	23,9	41,8
4	9,3	14,5	23,8	42,0
5	9,4	13,9	23,3	42,9
6	9,2	14,6	23,8	42,0
7	9,6	15,5	25,1	39,8

8	9,3	15,2	24,5	40,8
9	9,4	15,8	25,2	39,7
10	9,3	15,7	25,0	40,0
Průměr	**9,3**	**14,9**	**24,2**	**41,3**

Tab. 6.21: Rychlost překreslování mapy s 50 000 objekty s druhou optimalizací v prohlížeči Internet Explorer

Z tabulek 6.19, 6.20 a 6.21 lze vidět, že překreslování mapy s 50 000 objekty s využitím druhé optimalizace je velmi rychlé ve všech prohlížečích. Ve všech prohlížečích také trvá výběr objektů kratší dobu než vykreslení: v prohlížeči Google Chrome je to průměrně 1,4 ms (92% zrychlení oproti první optimalizaci), v prohlížeči Mozilla Firefox je to průměrně 4,6 ms (87% zrychlení oproti první optimalizaci) a v prohlížeči Internet Explorer je to průměrně 9,3 ms (82% zrychlení oproti první optimalizaci).

Díky optimalizacím bylo v jeden okamžik viditelných průměrně 250 objektů.

Experiment	Výběr objektů [ms]	Vykreslení [ms]	Celkem [ms]	Celkem [FPS]
1	5,2	31,3	36,5	27,4
2	5,4	33,0	38,4	26,0
3	5,4	32,7	38,1	26,2
4	5,4	32,0	37,4	26,7
5	5,2	32,3	37,5	26,7
6	5,5	33,6	39,1	25,6
7	5,7	33,8	39,5	25,3
8	5,4	33,8	39,2	25,5
9	5,5	34,0	39,5	25,3
10	5,7	34,9	40,6	24,6
Průměr	**5,4**	**33,1**	**38,6**	**25,9**

Tab. 6.22: Rychlost překreslování mapy s 200 000 objekty s druhou optimalizací v prohlížeči Google Chrome

Experiment	Výběr objektů [ms]	Vykreslení [ms]	Celkem [ms]	Celkem [FPS]
1	19,7	29,6	49,3	20,3
2	20,1	30,4	50,5	19,8
3	20,3	29,7	50,0	20,0
4	19,6	30,4	50,0	20,0
5	19,4	30,7	50,1	20,0
6	18,8	29,9	48,7	20,5
7	20,4	30,8	51,2	19,5
8	19,1	30,0	49,1	20,4
9	21,1	31,8	52,9	18,9

10	19,2	31,6	50,8	19,7
Průměr	**19,8**	**30,5**	**50,3**	**19,9**

Tab. 6.23: Rychlost překreslování mapy s 200 000 objekty s druhou optimalizací v prohlížeči Mozilla Firefox

Experiment	Výběr objektů [ms]	Vykreslení [ms]	Celkem [ms]	Celkem [FPS]
1	36,1	50,3	86,4	11,6
2	35,7	49,5	85,2	11,7
3	34,7	49,4	84,1	11,9
4	35,2	52,8	88,0	11,4
5	34,8	51,3	86,1	11,6
6	36,0	52,5	88,5	11,3
7	37,2	51,3	88,5	11,3
8	35,6	52,7	88,3	11,3
9	35,7	52,9	88,6	11,3
10	35,8	52,8	88,6	11,3
Průměr	**35,7**	**51,6**	**87,2**	**11,5**

Tab. 6.24: Rychlost překreslování mapy s 200 000 objekty s druhou optimalizací v prohlížeči Internet Explorer

Plynulost pohybu mapy s 200 000 objekty s využitím druhé optimalizace (viz tabulky 6.22, 6.23 a 6.24) je na hranici ideální plynulosti (25,9 FPS) v prohlížeči Google Chrome, dostatečně plynulá (19,9 FPS) v prohlížeči Mozilla Firefox a na hranici dostatečné plynulosti (11,5 FPS) v prohlížeči Internet Explorer. Díky optimalizacím bylo v jeden okamžik viditelných průměrně 900 objektů.

V prohlížeči Google Chrome tvoří výběr objektů 14% času, v prohlížeči Mozilla Firefox 39% a v prohlížeči Internet Explorer 40%. Stále je tedy prostor pro další optimalizace, pro potřeby tohoto systému je však již současná úroveň optimalizace dostatečná.

6.2.4 Zhodnocení

Sada testů 1 prokázala, že naivní řešení bez optimalizací, kdy jsou všechny objekty na mapě vloženy do DOM stromu, je zcela nedostatečná při požadavcích na plynulost systému s počtem objektů na mapě v řádu desítek tisíc.

První optimalizace přinesla značné zrychlení a umožnila plynulé vykreslení až 50 000 objektů, výběr objektů však zabíral velkou část celkového času.

Druhá optimalizace zkracuje čas výběru objektů pomocí rozdělení mapy na čtvercové oblasti až o 92%. Díky tomu lze vykreslovat desítky tisíc objektů s velmi vysokou plynulostí (více než 40 FPS ve všech prohlížečích pro 50 000 objektů) a stovky tisíc objektů s dostatečnou plynulostí.

Z výsledků testů plyne, že je propustnost uživatelského rozhraní velmi dobrá ve všech prohlížečích. I když je současné řešení zcela dostatečné pro limity dané specifikací, stále je zde prostor pro další optimalizace, pokud by bylo potřeba zvýšit výkon systému.

7 Závěr

Výstupem této práce je implementace komplexního herního systému založeného na standardních technologiích (HTML, CSS, Ajax) včetně grafického uživatelského rozhraní optimalizovaného pro rychlost, propustnost, ergonomii a robustnost.

Systém je navržen tak, aby bylo uživateli možné přehledně sdělit velké množství informací, na které on poté může velmi rychle a pohodlně reagovat. Klientská část je proto implementována jako jediná stránka, kde jsou nejdůležitější informace (hrací plocha) uživateli vždy přístupné, a další informace se otevírají v oknech nad hrací plochou. Velké množství interaktivních akcí (jako je změna zaměstnání, útoků nebo přidávání lodí do akcí) probíhá přímo na klientské straně a informace jsou serveru zaslány až po provedení. Tím se minimalizuje nutnost častého načítání obsahu ze serveru a zvětšuje se rychlost odezvy uživatelského rozhraní.

Serverová část systému je tvořena hlavně webovým, komunikačním a databázovým serverem. PHP skripty dostupné pomocí webového serveru slouží jako rozhraní, které poskytuje služby pro klientskou část systému. Komunikační server implementovaný v jazyce C++ umožňuje komunikaci uživatelů v reálném čase pomocí long polling protokolu.

Implementovaný systém dosahuje v testech velmi dobré rychlosti odezvy uživatelského rozhraní, pokud řídí omezeními zjištěnými v sekci 6.1 – tedy okna obsahují maximálně stovky HTML elementů. S klesajícím počtem elementů v okně se rychlost odezvy dále zvyšuje. Propustnost uživatelského rozhraní v systému je po implementování optimalizací také velmi dobrá, a to i nad rámec specifikací daných systémem (desítky tisíc objektů na mapě).

Tento systém byl testován a je funkční v prohlížečích Google Chrome 8, Mozilla Firefox 3.6, Internet Explorer 8 a Safari 5.

Mezi slabší místa systému patří fakt, že klientská část běží delší dobu v jediné stránce v prohlížeči, která často mění DOM strom. To způsobuje v prohlížečích Mozilla Firefox a Internet Explorer jisté zpomalení (viz sekce 6.1.1). Během psaní této práce také byly vydány nové verze prohlížečů, Internet Explorer 9 a Mozilla Firefox 4, které nebyly dostatečně testovány a předběžné testy naznačují, že s nimi systém není plně kompatibilní.

Rychlost odezvy uživatelského rozhraní je dostatečná, pokud se dodrží dříve popsaná omezení. V případě, že by bylo potřeba zobrazovat v oknech větší počet elementů nebo větší počet objektů na mapě, bylo by třeba využít některé z rychlejších řešení jako je Adobe Flex nebo Microsoft Silverlight.

S vydáním prohlížečů Internet Explore 9 a Mozilla Firefox 4 se také rozšiřuje spektrum prohlížečů podporujících HTML5, takže by do budoucna bylo zajímavé implementovat náročnější části systému, jako je hrací plocha, pomocí tagu `<canvas>`, což by zároveň umožnilo například snadnější využití animací. Současné řešení hrací plochy by však mělo být možné dále optimalizovat,

například pomocí hierarchického dělení mapy na menší oblasti pro rychlejší výběr viditelných objektů.

Jako další vylepšení by bylo možné implementovat podporu historie v prohlížeči. Řetězec nacházející se za znakem # v adresním řádku je přístupný ke čtení i psaní pomocí JavaScript vlastnosti `location.hash` a jeho změny se ukládají v historii prohlížeče. Vhodným nastavováním a monitorováním změn této vlastnosti by bylo možné implementovat chování, kdy se lze k významným událostem vracet pomocí tlačítka Zpět v prohlížeči.

Literatura

[1] *Ajax: A New Approach to Web Applications* [online]. 2005-03-13. [cit. 2011-01-04]. Dostupný na URL: <http://www.adaptivepath.com/ideas/essays/archives/000385.php>

[2] *JSON* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://www.json.org/>

[3] *Rich Internet Application Market Share* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://www.statowl.com/custom_ria_market_penetration.php>

[4] *Silverlight Architecture* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://msdn.microsoft.com/en-us/library/bb404713%28v=VS,95%29.aspx>

[5] *Usage of javascript libraries for websites* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://w3techs.com/technologies/overview/javascript_library/all>

[6] *DataTables (table plug-in for jQuery)* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://www.datatables.net/>

[7] *DataTables server-side processing example* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://www.datatables.net/examples/data_sources/server_side.html>

[8] *element.innerHTML - MDC Doc Center* [online]. [cit. 2011-01-04]. Dostupný na URL: <https://developer.mozilla.org/en/DOM:element.innerHTML>

[9] *GUIMark* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://www.craftymind.com/guimark/>

[10] *Understanding Illustrator's 9-Slice Scaling* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://www.creativepro.com/article/understanding-illustrator-s-9-slice-scaling>

[11] *GUIMark* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://www.craftymind.com/guimark-inside/>

[12] *Benefits of GPU-powered HTML5* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://blogs.msdn.com/b/ie/archive/2010/04/09/benefits-of-gpu-powered-html5.aspx>

[13] *Socket connections* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://livedocs.adobe.com/flex/3/html/help.html?content=17_Networking_and_communications_5.html>

[14] *Socket Class (System.Net.Sockets)* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://msdn.microsoft.com/en-us/library/system.net.sockets.socket%28VS,95%29.aspx>

[15] *Adobe Flex 3 Help: Loading data* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://livedocs.adobe.com/flex/3/html/help.html?content=05B_Security_10.html>

[16] *Network Security Access Restrictions in Silverlight* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://msdn.microsoft.com/en-us/library/cc645032%28v=VS,95%29.aspx>

[17] *The WebSocket API* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://dev.w3.org/html5/websockets/>

[18] *Comet Programming: Using Ajax to Simulate Server Push* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://www.webreference.com/programming/javascript/rg28/ >

[19] *Tornado Web Server Documentation* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://www.tornadoweb.org/documentation#performance>

[20] *ab - Apache HTTP server benchmarking tool* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://httpd.apache.org/docs/2.0/programs/ab.html>

[21] *Action Message Format - AMF 3 Specification* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://opensource.adobe.com/wiki/download/attachments/1114283/amf3_spec_05_05_08.pdf>

[22] *Zynga Bets on HTML5 With Dextrose Acquisition* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://www.insidesocialgames.com/2010/09/24/zynga-acquires-dextrose-aves-engine-html5/>

[23] RUSSELL, C. T. (ed.): *New Horizons: Reconnaissance of the Pluto-Charon System and the Kuiper Belt*. Springer Science+Business Media, 2009. 404s. ISBN 978-0-387-89517-8

[24] *Browser Security Handbook, part 2* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://code.google.com/p/browsersec/wiki/Part2>

[25] *Proxy and reverse proxy servers* [online]. [cit. 2011-01-04]. Dostupný na URL: <http://en.kioskea.net/contents/lan/proxy.php3 >

[26] MAHEMOFF, M.: *Ajax Design Pattern*. O'Reilly Media, 2006. 655 s. ISBN 978-0-59-610180-0

[27] GALITZ, W. O.: *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques*. Wiley Publishing, 3. vydání, 2007. 888 s. ISBN 978-0-470-05342-3

[28] Use PHP functions in JavaScript [online]. [cit. 2011-04-17]. Dostupný na URL: < http://phpjs.org/>

[29] *MySQL++* [online]. [cit. 2011-04-17]. Dostupný na URL: <http://tangentsoft.net/mysql++/>

[30] *PHP MySQL wrapper* [online]. [cit. 2011-04-17]. Dostupný na URL: <http://www.ricocheting.com/code/php/mysql-database-class-wrapper>

[31] *kses - PHP HTML/XHTML filter* [online]. [cit. 2011-04-17]. Dostupný na URL: < http://sourceforge.net/projects/kses/ >

Seznam příloh

Příloha 1. Obsah přiloženého DVD

Obsah jednotlivých adresářů na přiloženém DVD:

- `server` – VMware virtuální stroj obsahující kompletní instalaci systému. Tento virtuální stroj je kompatibilní s volně šiřitelnou aplikací VMware Player verze 2.5.
- `navod` – Návod k použití virtuálního stroje.
- `php` – Zdrojové kódy PHP skriptů pro webserver.
- `sql` – SQL skripty pro inicializaci MySQL databáze.
- `app` – Zdrojové soubory komunikačního serveru a utility pro výpočet nového kola.
- `diagram` – Diagram tříd komunikačního serveru
- `testy` – Testy použité pro porovnání technologií pro moderní uživatelská rozhraní.
- `text` – Text diplomové práce.