VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Implementace online karetní hry pomocí technologií HTML5 a Websocket

Implementation of Online Card Game Using the Technologies HTML5 and Websocket

2012 Radim Vavřík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Radim Vavřík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Implementace online karetní hry pomocí technologií HTML5 a Websocket
Implementation of Online Card Game Using the Technologies HTML5 and Websocket

Zásady pro vypracování:

V současné době existuje spousta karetních a deskových her. Některé z těchto her se stanou natolik populární, že pak vznikají jejich počítačové verze. Cílem práce bude vytvořit online webovou hru BANG! pomocí HTML5 a Websocket.

Během vypracování se řiďte následujícími pokyny:
1. Seznamte se a popište aktuální návrh standardu HTML5 a souvisejících technologií, potřebných pro řešení práce.
2. Vytvořte analýzu této hry s ohledem na možnosti hraní více hráčů po síti přes klasický prohlížeč.
3. Naimplementujte tuto hru jako klient-server ve zvoleném jazyce.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Lumír Návrat**

Datum zadání: 18.11.2011
Datum odevzdání: 04.05.2012

VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA * FAKULTA elektrotechniky a informatiky * OSTRAVA

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 4. května 2012

Radim Vavřík

Na tomto místě bych chtěl velmi poděkovat vedoucímu práce Ing. Lumírovi Návratovi za odbornou pomoc a vstřícnost při zpracování této práce. Děkuji také všem, kteří mi pomohli s testováním, či mě jinak podporovali.

Abstrakt

Hlavním cílem této bakalářské práce je seznámení se s technologiemi nově vznikajícího standardu HTML5, který by měl přinést vylepšené postupy pro tvorbu a prezentaci moderního webového obsahu a poskytnout nové možnosti pro komplexní webové aplikace. Při této příležitosti bylo mým úkolem navrhnout a implementovat online webovou karetní hru BANG! pro více hráčů dle modelu klient–server s využitím WebSockets a dalších HTML5 technologií.

Klíčová slova: BANG!, online hry, více hráčů, HTML5, WebSocket

Abstract

The main aim of this bachelor thesis is to get familiar with the technologies of newly emerging HTML5 standard, which should bring improved methods for development and presentation of modern web content, and should provide new features for complex web applications. On this occasion it was my job to design and implement multiplayer online web card game BANG! according to the model client–server using HTML5 WebSockets and other technologies.

Keywords: BANG!, online games, multiplayer, HTML5, WebSocket

Seznam použitých zkratek a symbolů

AJAX	–	Asynchronous JavaScript and XML
API	–	Application Programming Interface
BLOB	–	Binary Large Object
CSS	–	Cascading Style Sheets
DDA	–	Digital Differential Analyzer
GUI	–	Graphical User Interface
HTML	–	HyperText Markup Language
HTTP	–	HyperText Transfer Protocol
IIS	–	Internet Information Services
JSON	–	JavaScript Object Notation
SHA-1	–	Secure Hash Algorithm
TCP	–	Transmission Control Protocol
UA	–	User Agent
UCS	–	Universal Character Set
URI	–	Uniform Resource Identifier
URL	–	Uniform Resource Locator
UTF-8	–	UCS Transformation Format – 8-bit
WHATWG	–	Web Hypertext Application Technology Working Group
W3C	–	World Wide Web Consorcium
XHTML	–	Extensible Hyper Text Markup Language
XML	–	Extensible Markup Language
XOR	–	Exclusive or

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

V posledních letech jsme svědky velmi rychlé expanze počítačových a zejména webových technologií do téměř všech odvětví lidské činnosti a tím i jejich provázání s našimi každodenními životy. Tento dlouhodobý a stále stoupající trend způsobil, že některé dosavadní metody pro tvorbu webového obsahu přestávají dostačovat požadavkům na stále komplexnější a funkcionálně i graficky bohatší webové aplikace, pro jejichž vývoj se často stávají těžkopádnými. Dalším negativním jevem na poli webového vývoje se stala větší, či menší roztříštěnost implementací jednotlivých technologií, např. rozdílnost zpracování HTML/XHTML kódu webovými prohlížeči různých výrobců.

To byly mimo jiné jedny z hlavních důvodů pro vznik pracovní skupiny Web Hypertext Application Technology Working Group (WHATWG), která v roce 2004 začala formovat návrhy nové revize značkovacího jazyka HTML, jehož čtvrtá verze, dokončená na konci roku 1997, se takřka přestala vyvíjet. Skupina tak dala podnět k opětovnému intenzivnímu vývoji nových technologií, které by vyhovovaly požadavkům na tvorbu moderního webového obsahu [3].

Mým úkolem bylo za pomoci některých z těchto technologií vytvořit online webovou verzi hry BANG! a tím v praxi ověřit způsob a míru jejich využitelnosti. Pro účel hraní více hráčů po síti se vybízelo užití HTML5 technologie WebSocket, které bude věnována samostatná podkapitola. Stejným způsobem bude věnována pozornost také HTML5 elementu Canvas, který jsem použil k vykreslování hrací plochy a všech grafických prvků ve hře.

Text práce čtenáře postupně seznámí se samotnou hrou BANG! a jejími specifiky. Následovat bude popis jednotlivých technologií, které byly k tvorbě hry použity. Na závěr bude čtenář seznámen s návrhem a popisem implementace klientské i serverové části aplikace.

2 BANG!

BANG! je společenská karetní hra, svým pojetím zasazená do prostředí westernového divokého západu. Jejím autorem je italský herní návrhář Emiliano Sciarra a vydána byla roku 2002. Od té doby se stala velmi oblíbenou v mnoha zemích a získala také několik mezinárodních ocenění [1].

2.1 Pravidla

Základní oficiální verze hry je určena pro 4 – 7 hráčů s doporučeným věkem od 8 let a časová náročnost je odhadována na cca 30 minut. Obsahuje celkem 103 hracích karet rozdělených do tří odlišných kategorií, dle svého způsobu použití.

První z kategorií jsou karty postav, které po svém vylosování určují roli a tím i taktiku a cíl každého z hráčů. Postava šerifa se s podporou svých pomocníků snaží vypořádat s bandity, jež usilují o jeho život. Celou situaci komplikuje nevyzpytatelná postava odpadlíka. Jejím cílem je zůstat poslední přeživší, vyzvat šerifa k závěrečnému duelu, vyhrát a nahradit jej v jeho dosavadní pozici.

Další skupinu karet tvoří charaktery postav. BANG! jich obsahuje 16 typů, každý z nich přiřazuje postavě jméno, často odvozené od skutečné historické osobnosti divokého západu, a speciální vlastnost, která hráče určitým způsobem zvýhodňuje. Například Calamity Janet může zaměňovat použití karet Bang! a Vedle! (viz níže).

Poslední a nejobsáhlejší kategorií jsou karty, jejichž zahráním se hráči snaží zdokonalit schopnosti svých postav, nebo uškodit protihráčům, a tím dosáhnout vítězství. Tyto karty mohou mít okamžitý účinek a jsou ihned odhozeny (Indiáni, Salón, Pivo...), nebo jsou hráčem vyloženy a jejich efekt je dlouhodobý (Mustang, Barel, Winchester...).

2.2 Průběh hry

Hra začíná vylosováním karet postav (rolí), jež zůstávají, kromě role šerifa, v anonymitě. Ta spolu s různorodostí cílů postav přispívá k celkové hratelnosti tím, že ponechává hráče v nejistotě a je na jejich strategii jak rychle a zda-li vůbec odhalí záměry svých protihráčů.

Druhým krokem je losování charakterů postav, které, jak již bylo uvedeno výše, přidělují postavám jedinečnou speciální vlastnost. Tyto karty jsou na rozdíl od karet postav vždy veřejné všem hráčům. Ruby zbylých nevylosovaných karet charakterů, vyobrazující 5 nábojnic, jsou v průběhu hry skrývány či odkrývány, čímž signalizují aktuální počet zbývajících životů daného hráče.

Následuje postupné rozdání zamíchaných hracích karet všem hráčům a to v takovém počtu, kolik zobrazují miniatury nábojnic na líci vylosované karty charakteru hráče. Výjimku tvoří opět postava šerifa, která obdrží jednu hrací kartu navíc. Hráč s touto postavou také vlastní hru započíná.

Průběh hry je tvořen koly, v nichž postupně každý hráč provede svůj tah, skládající se z těchto tří částí:

- líznutí dvou karet z lízacího balíčku (zbylé nerozdané hrací karty),

- zahrání libovolného počtu karet, které má hráč v rukou či vyloženy na hrací ploše,
- odhození přebývajících karet do odhazovacího balíčku (hráč může mít v rukou pouze tolik karet, kolik mu zbývá životů).

Hráči se v tazích po směru hodinových ručiček střídají a svým počínáním eliminují, dokud není hra ukončena splněním jedné z následujících podmínek:

- je zabit odpadlík i všichni bandité, vyhrává šerif se svými pomocníky,
- je zabit šerif a přežil pouze odpadlík, vyhrává odpadlík,
- je zabit šerif, vyhrávají bandité.

Toto byl velmi stručný nástin pravidel a popis průběhu hry, jejichž podrobný výklad v anglickém jazyce je k nalezení v oficiálních pravidlech [2].

2.3 Rozšíření

BANG! se postupem času stal natolik úspěšným, že začaly vznikat jeho oficiální i neoficiální rozšíření, pravidla pro turnaje a úpravy pravidel pro specifické situace (např. hra pro 2 hráče).

Mezi oficiální rozšíření patří:

- Bang! The Bullet!,
- Dodge City,
- A Fistful of Cards,
- Face Off,
- Gold Rush,
- High Noon,
- Wild West Show,

mezi neoficiální pak:

- El Dorado,
- Death Mesa,
- Robbers' Roost,
- Valley of Shadows.

Tato rozšíření povětšinou přidávají k základní verzi hry nové charaktery postav a jejich speciální vlastnosti, nebo doplňují výčet herních karet o další, se zcela novými způsoby užití, a tím rozšiřují možnosti a zvyšují hratelnost.

3 HTML5

Tato kapitola seznámí čtenáře se základními informacemi o jazyku HTML, jeho stručnou historií, odlišnostmi návrhu 5. verze od dosavadních standardů i náhledem do blízké budoucnosti.

3.1 Historie

Značkovací jazyk HTML byl navržen v roce 1990 za účelem tvorby dokumentů a jejich následného publikování v podobě webových stránek. Od té doby prošel mnoha revizemi, rozšířeními a úpravami na základě stále rostoucích požadavků. Tím, že se web postupně rozvíjel, bylo potřeba stanovit určité standardy, jako výchozí pilíře pro jednotlivé implementace. Návrh a vývoj těchto standardů probíhá pod záštitou mezinárodní organizace World Wide Web Consorcium (W3C), která definuje klíčové části systému WWW, a tím se stará o jeho provoz a dlouhodobý technologický růst.

V roce 2007 toto konsorcium vydalo 4. a poslední verzi HTML (roku 2009 byly dokončeny opravy a vydány jako standard HTML 4.01). Poté mělo v plánu zavést další standard v podobě nově vyvinutého jazyka XHTML, který upravoval zápis dosavadního HTML tak, aby splňoval specifické podmínky pro XML dokumenty, a který se měl stát nástupcem HTML.

Tento záměr byl, jak se později ukázalo, slepou uličkou v hlavním proudu webového vývoje. Jak již bylo uvedeno v kapitole 1, v roce 2004 byla některými pracovníky společností Apple, Mozilla Foundation a Opera Software založena skupina WHATWG, která v reakci na plány W3C začala pracovat na návrhu specifikace nové verze jazyka HTML. Ten by měl splňovat současné požadavky na tvorbu moderního webového obsahu, zejména webových aplikací a zároveň být zpětně kompatibilní se všemi svými předchozími verzemi.

V roce 2007 projevilo W3C zájem o spolupráci na tvorbě nového HTML standardu a vytvořilo vlastní pracovní skupinu, která začala kooperovat s organizací WHATWG. Výsledkem jsou dva mírně se lišící, neustále doplňované a aktualizované návrhy specifikací HTML5, jejichž vývoj je mimo jiné založen na emailové zpětné vazbě odborné veřejnosti a vývojářů, ať už tvůrců webového obsahu, či autorů implementací webových prohlížečů. Jejich návrhy a připomínky zpracovává zvolený editor, jímž se stal Ian Hickson.

V současné době je HTML5 ve stádiu funkčního pracovního návrhu, jehož klíčové části jsou již plně podporovány posledními verzemi všech rozšířených prohlížečů (Internet Explorer, Mozilla Firefox, Google Chrome, Safari, Opera). Specifikace bude podle autorů považována za dokončenou ve chvíli, kdy budou existovat dvě její kompletní implementace. Datum předpokládaného dokončení vývoje HTML5 není známo, dříve však bylo vydání poslední konečné verze obsahující všechny opravy editorem odhadováno na rok 2022. [4].

3.2 Nové vlastnosti

Specifikace HTML5 přichází s mnoha novými rysy, které mají umožnit vytvářet funkcionálně bohatší webový obsah a přitom zachovat co nejpřehlednější zápis kódu. Mezi tyto rysy patří velké množství zcela nových, nebo upravených elementů a jejich atributů, způsob jejich zápisu v dokumentu a v neposlední řadě množství nových či upravených API např. pro:

- ovládání videa/audia,
- validaci formulářů,
- offline aplikace,
- drag & drop,
- časovaná zpětná volání,
- canvas,
- skripty na pozadí,
- klientské úložiště,
- webSocket,
- přenos dat ze strany serveru

a mnoho dalších. O některých vlastnostech se zmíním níže, vybraným budu věnovat samostatné podkapitoly.

3.2.1 Struktura

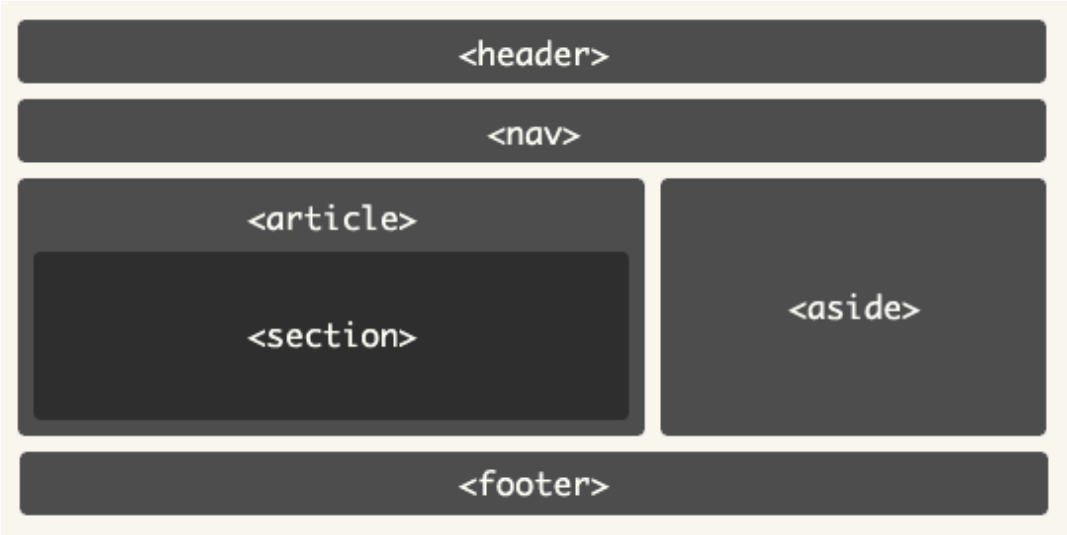
První výraznou změnou je zavedení nových strukturálních tagů, které mají za úkol vnést do kódu sémantiku. S tou již současné vyhledávače pracují a měly by přinést efektivnější a přesnější výsledky vyhledávání. Dalším kladným efektem je eliminace velkého počtu elementů div, a tím zlepšení čitelnosti kódu. Příklad je znázorněn na obrázku 1, kterému odpovídá zdrojový kód ve výpisu 1.

```
<!DOCTYPE html>
<html>
  <body>
    <header>...</header>
    <nav>...</nav>
    < article >
      <section>
      ...
      </section>
    </ article >
```

```
        <aside>...</aside>
        <footer> ... </footer>
    </body>
</html>
```

Výpis 1: Struktura dokumentu v HTML5

Za povšimnutí také stojí značně zjednodušený zápis DOCTYPE oproti předchozím zvyklostem.



Obrázek 1: Struktura dokumentu v HTML5 převzato [5]

3.2.2 Video, audio

Dalším velkým přínosem je zavedení zcela nových elementů video a audio. Ty jsou určeny, jak je z jejich názvu patrné, k umístění videa resp. zvukového záznamu na webovou stránku, bez použití pluginů výrobců třetích stran (Flash Player, QuickTime, Windows Media...). To v dosavadních verzích HTML nebylo možné. Nevýhodou tohoto standardu zůstává prozatím nevyřešená otázka rozdílnosti kodeků pro přehrávání videa resp. audia, používaných jednotlivými výrobci prohlížečů. Tento problém je pro správné přehrávání ve všech majoritních prohlížečích řešen uvedením několika video/audio souborů v potřebných formátech pomocí elementu source.

3.2.3 Web Forms 2.0

Webové formuláře doznávají příchodem HTML5 rozsáhlých vylepšení. Umožňují totiž využívat sémantických prvků a lépe tak zpracovávat data vkládaná uživatelem.

Největší změny se týkají elementu input. Jeho atribut type, který doposud mohl nabývat klasických hodnot jako text, password, checkbox, radio, submit, select, či textarea, byl rozšířen o nové praktické typy:

- tel (telefonní číslo),
- search,
- url,
- email,
- datetime,
- date,
- month,
- week,
- time,
- datetime−local (využívá časové zóny),
- number,
- range,
- color.

Tyto typy byly zavedeny, aby zjednodušily uživateli zadávání hodnot – prohlížeč může např. nabídnout kalendář pro výběr data, nebo seznam e-mailových adres. V neposlední řadě také poskytují automatickou validaci vkládaných hodnot dle svého určení (např. typ number příjme pouze ciferné hodnoty, typ email zase jen e-mailovou adresu v platném formátu atd.). Formuláře se tak stávají přehlednějšími, jednoduššími a zlepšují odezvu mezi uživateli a servery zpracovávajícími data.

Poznámka 3.1 Některá mobilní zařízení s virtuální klávesnicí nabízí uživateli podle typu vstupního pole také odpovídající typ klávesnice (číselnou pro typ email, textovou pro typ textarea apod.).

Element input ale nabízí i další zajímavé a užitečné atributy:

- autofocus – umožňuje nastavit element tak, aby byl ihned po načtení stránky označen a připraven k vkládání dat,
- placeholder – dovoluje zobrazit krátkou textovou nápovědu přímo v elementu, která po jeho označení zmizí,
- required – nedovolí uživateli odeslat formulář, dokud nevyplní požadované hodnoty, jejichž vstupní pole navíc graficky zvýrazní.

4 Použité technologie

V této kapitole budou blíže popsány jednotlivé technologie, které jsem v praktické části práce použil, nebo se kterými jsem se setkal.

4.1 WebSocket

Web byl od svého počátku postaven na modelu request – response (požadavek – odpověď), který má své nesporné výhody. Pro některé interaktivní aplikace je ale důležitá dynamická komunikace se serverem.

Prvním masově rozšířeným řešením se stal AJAX, který umožňuje odesílání požadavku a zpracování odpovědi bez nutnosti opětovného načtení celé stránky. Ten však pro potřebu stále čerstvých dat neřeší nutnost neustále se doptávat serveru na aktualizace (ať už interakcí uživatele, nebo tzv. poolingem).

Tento problém měla vyřešit technologie Comet, která umožňuje obousměrný přenos dat s využitím dvou paralelních HTTP spojení. Tato metoda se ale ukázala jako neefektivní. Docházelo k časovým prodlevám a nepřesnostem v komunikaci, zároveň byla náročná na provoz i vývoj, a proto se od ní postupem času upustilo.

Novou generací obousměrné (plnně duplexní) komunikace a součástí balíku HTML5 je technologie WebSocket. Ta umožňuje vytvořit a trvale uchovat bezpečné spojení mezi klientskou webovou aplikací využívající WebSocket API [6] a serverem implementujícím WebSocket protokol popsaný ve své specifikaci [7]. Výhodou WebSockets je také rychlost, daná řádově menším množstvím přenášených režijních, "neužitečných"dat (jednotky bajtů) oproti hlavičkám HTTP požadavků a odpovědí (stovky bajtů).

4.1.1 Podpora prohlížeči

V době psaní této práce je aktuální navrhovaný standard WebSocket majoritními prohlížeči podporován následovně:

- Google Chrome – od verze 16,
- Mozilla Firefox – od verze 11,
- Safari – pouze v nočních sestaveních,
- Internet Explorer – nepodporováno,
- Opera – nepodporováno.

Poznámka 4.1 Internet Explorer 10 bude podporovat navrhovaný standard WebSocket, Internet Explorer 9 podporuje po instalaci pluginu návrh protokolu WebSocket HyBi 09 a Opera od verze 11 má podporu návrhu protokolu WebSocket HyBi 00 standardně zakázánu.

4.1.2 WebSocket API

WebSocket API je rozhraní, kterého mohou tvůrci webových aplikací využít k navázání trvalého spojení se serverem a následně odesílat či přijímat zprávy.

Navázání spojení je velmi jednoduché, jak ukazuje výpis zdrojového kódu 2. Konstruktor WebSocket přijímá povinný parametr url vycházející z HTTP URL ve formátu:

- `"ws:" "//" host [ ":" port ] path [ "?" query ]`

nebo pro zabezpečené spojení:

- `"wss:" "//" host [ ":" port ] path [ "?" query ]`

Tento řetězec specifikuje schéma (ws/wss), název a adresu serveru (host), na který se bude UA (User Agent - součást webového prohlížeče, software jednající na základě požadavků uživatele) snažit připojit. Může rovněž obsahovat port, na kterém server naslouchá, a cestu s dotazem (path, query). Pokud není port zadán, je jako výchozí použit port 80 (port 443 v případě zabezpečeného spojení). Parametr protocols je nepovinný, pokud je zadán, musí se jednat o řetězec, nebo pole řetězců, vyjadřujících název subprotokolů, jimiž se uživatel hodlá připojit.

```
var url = "ws://example.com:8181/test";
var protocols = "chat.example.com";
var ws = new WebSocket(uri, protocols);
```

Výpis 2: Vytvoření WebSocket spojení

Informace o aktuálním stavu spojení je uchovávána v atributu readyState a může nabývat těchto hodnot:

- CONNECTING (0) – navazování spojení ještě nebylo dokončeno,
- OPEN (1) – spojení je úspěšně navázáno,
- CLOSING (2) – spojení se právě ukončuje,
- CLOSED (3) – spojení bylo uzavřeno, nebo nemůže být navázáno.

Rozhraní také nabízí sadu metod (použití ve výpisu 3) volaných v případě, že nastane odpovídající událost:

- open – volána ihned po úspěšném navázání spojení (viz podkapitola 4.1.3),
- message – oznamuje příchod nové zprávy ze serveru; parametr obsahuje atribut data, jehož hodnotou jsou samotná data zprávy (textová, BLOB, nebo ArrayBuffer),
- error – vyvolána v případě že nastane nějaká chyba spojení,

- close – signalizuje uzavření spojení, ať už ze strany UA, nebo ze strany serveru.

```
ws.onopen = function() {
    console.log("Spojeni␣uspesne␣navazano.");
};

ws.onmessage = function(event)
    // cteni prichozich dat
    console.log("Zprava␣prijata:␣" + event.data);
};

ws.onerror = function () {
    console.log("Nastala␣chyba␣WebSocket.");
};

ws.onclose = function() {
    console.log("Spojeni␣uzavreno.");
};
```

Výpis 3: Reakce na WebSocket události

Dalšími prvky WebSocket API jsou atributy bufferedAmount a binaryType.

První z nich reprezentuje množství dat (v bajtech) odeslaných do fronty metodou send(data) (viz níže), které ale ještě nebyly vypuštěny do sítě. Tento pomocný atribut je užitečný např. pro monitorování propustnosti zpráv, nebo pro řízení frekvence jejich odesílání.

Druhým jmenovaným je řetězec vyjadřující způsob, jakým budou zpracována příchozí binární data. Výchozí hodnotou je "blob" [8], jenž značí surový řetězec dat. Pokud je binaryType nastaven na hodnotu "arraybuffer", data jsou reprezentováná jako typ ArrayBuffer, což je pole binárních dat, které je možné číst ve zvoleném formátu [9].

V neposlední řadě rozhraní nabízí metodu send(data) sloužící pro odesílání dat na server. Tato metoda přijímá povinný parametr data, kterým může být textový řetězec v kódování UTF-8, binární řetězec BLOB, nebo objekt typu ArrayBuffer. Použití je patrné z výpisu 4.

```
try {
    var message = "Ahoj␣servere,␣tady␣klient";
    ws.send(message);
}
catch(e) {
    if (e == "InvalidStateError")
        console.log("Spojeni␣se␣stale␣navazuje.");
    else if (e == "SyntaxError")
        console.log("Retezec␣obsahuje␣neplatne␣znaky.");
    else
        console.log("Pri␣odesilani␣nastala␣chyba.");
}
```

Výpis 4: Odesílání zpráv na WebSocket server

Spojení je samozřejmě možné ukončit a to metodou close(), která navíc může přijímat 2 volitelné parametry. Prvním z nich je code označující kód důvodu uzavření (např. 1000 – normální ukončení spojení) a druhým řetězec reason, který obsahuje slovní popis příčiny uzavření (viz výpis 5).

```
try {
    var reason = "Standardni ukonceni spojeni";
    var code = 1000;
    ws.close(code, reason);
}
catch(e) {
    if (e == "InvalidAccessError")
        console.log("Neplatny kod uzavreni spojeni.");
    else if (e == "SyntaxError")
        console.log("Retezec obsahuje neplatne znaky, nebo je delsi nez 123 bajtu.");
}
```

Výpis 5: Ukončení WebSocket spojení

4.1.3 WebSocket Protokol

Tento protokol detailně popisuje, co by měly obsahovat jednotlivé implementace klientské i serverové strany. Je založen na principu jednoho TCP spojení pro komunikaci v obou směrech. Té musí předcházet tzv. „podání rukou“ (handshake) mezi oběma stranami, čímž dojde k vytvoření trvalého spojení. Klientův požadavek může vypadat tak, jak je uvedeno ve výpisu 6. Odpověď serveru pak bude podobná výpisu 7.

```
GET /test HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat.example.com, superchat
Sec-WebSocket-Version: 13
```

Výpis 6: Část handshake pocházející od klienta

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat.example.com
```

Výpis 7: Část handshake ze strany serveru

Poznámka 4.2 Stojí za zmínku, že první řádky obou stran handshake odpovídají hlavičkám HTTP požadavku resp. odpovědi.

Obě součásti handshake obsahují několik hlaviček parametrů spojení a k nim odpovídajících hodnot. Část handshake zaslaná klientem se skládá z těchto prvků:

- GET – HTTP GET požadavek včetně dotazované URI,
- Host – adresa serveru,
- Upgrade – musí obsahovat řetězec "websocket",
- Connection – musí obsahovat řetězec "Upgrade",
- Sec-WebSocket-Key – klientský bezpečnostní klíč,
- Sec-WebSocket-Version – pro současně navrhovaný standard je to verze 13,
- Origin – zdrojová adresa klienta, volitelný atribut,
- Sec-WebSocket-Protocol – seznam subprotokolů.

Ke klientskému handshake mohou být ještě připojeny atributy s informacemi o rozšířeních, popř. cookies.

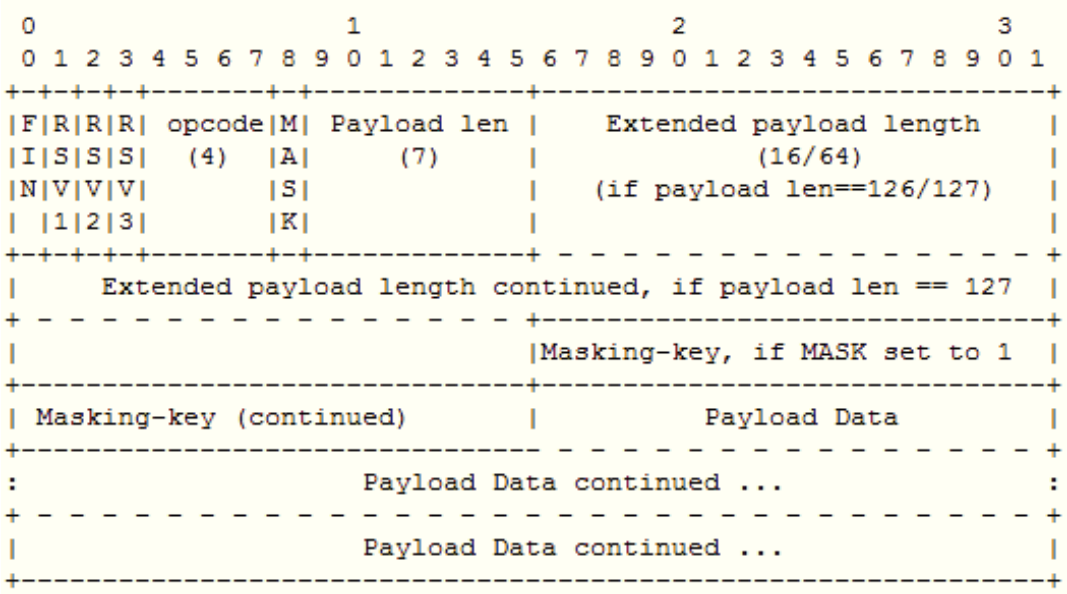
Server musí v rámci procesu autentizace nového připojení vytvořit serverový bezpečnostní klíč. Ten se skládá z klientského klíče (Base64-kódovaná 16 bajtová hodnota např. dGhlIHNhbXBsZSBub25jZQ==), ke kterému je připojen řetězec "258EAFA5-E914-47DA-95CA-C5AB0DC85B11". SHA-1 hash tohoto spojeného řetězce je následně Base64-zakódován a výsledek (z předchozích hodnot by vyšlo "s3pPLMBiTxaQ9kYGzzhZRbK+xOo=") je použit jako hodnota serverového klíče v atributu Sec−WebSocket−Accept.

Pokud je autentizace dokončena s kladným výsledkem, server odešle klientovi svou část handshake (viz výpis 7). Ta je tvořena následujícími atributy:

- "HTTP/1.1 101 Switching Protocols"– HTTP odpověď s kódem o úspěchu,
- Upgrade – musí obsahovat řetězec "websocket",
- Connection – musí obsahovat řetězec "Upgrade",
- Sec-WebSocket-Accept – serverový bezpečnostní klíč,
- Sec-WebSocket-Protocol – subprotokol použitý pro spojení.

Po úspěšném dokončení handshake a návázání spojení může začít samotný přenos dat. Ten je prováděn pomocí posloupnosti datových rámců. Tyto rámce jsou tvořeny jednotlivými bity, nebo skupinami bitů (znázorněnými na obrázku 2), které mají následující význam:

- FIN (1 bit) – stanovuje, zda se jedná o poslední rámec jedné zprávy;
- RSV1, RSV2, RSV3 (po 1 bitu) - bity vyhrazené pro potřeby rozšíření;



Obrázek 2: Formát datového WebSocket rámce

- opcode (4 bity) – operační kód rámce, jednotlivé hodnoty definují jeho význam, např. 0x0 značí, že rámec tvoří spolu s jedním, nebo více předchozími rámci jednu zprávu. 0x1 označuje rámec s textovými daty, naopak 0x2 s binárními. Hodnota 0x8 zase říká, že po tomto rámci bude spojení ukončeno;
- MASK (1 bit) – určuje, zda jsou data rámce maskována maskovacím klíčem;
- Payload len (7, 7 + 16, nebo 7 + 64 bitů) – pokud je velikost dat, přenášených rámcem v rozmezí 0 – 125 bajtů, Payload len vyjadřuje tuto jejich délku. Pokud je dat méně než 65536 bajtů, Payload len je nastaveno na hodnotu 126 a množství dat vyjadřují následující 2 bajty (Extended payload length). Pokud je délka dat větší, Payload len je nastaveno na hodnotu 127 a skutečnou délku dat udává následujících 8 bajtů;
- Masking-key (0, nebo 4 bajty) – Maskovací klíč je klientem generované 32 bitové náhodné (ne pseudo-náhodné) číslo, sloužící k šifrování dat posílaných od klienta na server. Data zpráv posílaných ze serveru klientovi maskována být nesmí – v tom případě není klíč definován. Maskování i odmaskování je prováděno podle algoritmu ve výpisu 8 pomocí funkce XOR. Bajt transformovaných dat i je roven výsledku funkce XOR bajtu zdrojových dat i s bajtem maskovacího klíče j, kde j se rovná i modulo 4;
- Payload Data – vlastní užitečná data zprávy (včetně případných rozšíření).

```
for (int i = 0; i < source.Length; i++)
{
    result[i] = (byte)(source[i] ^ mask[i % 4]);
}
```

Výpis 8: Algoritmus maskování/odmaskování dat

Data přenášená pomocí zpráv mohou být textová, nebo binární. Jaký typ dat nese konkrétní zpráva je patrné z hodnoty operačního kódu opcode popsaného výše.

Výhodnou vlastností přenosu dat přes WebSockets je také možnost přenést libovolně dlouhou zprávu. To je umožněno tzv. fragmentací, kdy je zpráva rozdělena do potřebného množství rámců. Typ přenášených dat musí být pro všechny rámce zprávy stejný a je specifikován prvním rámcem. Všechny další rámce musí mít hodnotu opcode nastavenu na 0x0. Poslední rámec zprávy musí mít navíc nastaven bit FIN na hodnotu 1.

Protokol umožňuje posílání speciálních tzv. řídících rámců (Control frames). Prozatím jsou definovány pouze 3 typy těchto rámců a to Ukončovací rámec, Ping rámec a Pong rámec.

Operační kód opcode ukončovacího rámce je nastaven na hodnotu 0x8. Tento rámec signalizuje, že spojení bude uzavřeno. Může nést data uvozená 2 bajtovým číselným kódem, následovaným textovým zdůvodněním ukončení spojení. Pokud jedna ze stran spojení obdrží teno rámec, smí už odeslat pouze jedinou zprávu, a to rovněž ukončovací rámec jako potvrzení ukončení spojení.

Ping rámec má operační kód opcode nastaven na 0x9 a může obsahovat libovolná data. Pokud jedna ze stran obdrží tento Ping rámec, musí neprodleně odpovědět Pong rámcem s operačním kódem 0xA a daty shodnými s těmi, které obdržela v Ping rámci. Tento mechanismus může sloužit mimo jiné k ověření dostupnosti druhé strany spojení.

Podrobné infromace se všemi detaily může čtenář nalézt ve specifikaci WebSocket protokolu [7].

V současnosti rovněž existuje několik komerčních i volně šířených implementací WebSocket serveru v mnoha programovacích jazycích a platformách (např. Java, Ruby, Python, C++, .NET, Node.js)

4.2 Canvas

Canvas, nebo-li kreslicí plátno, je nový element jazyka HTML5. Umožňuje vykreslovat libovolné grafické prvky nebo obrázky na danou kreslicí plochu pomocí JavaScriptu.

Ke canvasu bylo vytvořeno kompletní API dostupné v kapitole *The canvas element* HTML5 specifikace [3]. Obsahuje atributy width a height, které udávají velikost plochy, na kterou je možné kreslit, transformační metody toDataURL() a toBlob() pro převod obrazu plátna na URL-řetězec resp. Blob objekt a především metodu getContext(), která vrací kontext elementu, poskytující metody pro samotnou práci s plátnem a kreslicími nástroji.

Poznámka 4.3 Rozhraní canvasu nabízí všemi současnými prohlížeči podporovaný 2D kontext, který bude v této kapitole dále popisován. Mimo něj je k dispozici také 3D kontext založený na WebGL API (použito např. v Google Maps nebo Google Body), který

je v době psaní této práce stále ve vývoji, a tudíž prozatím není plně podporován všemi prohlížeči.

Rozhraní 2D kontextu nabízí metody pro správu stavu kontextu (jednotlivé sady nastavení), práci s čarami, textem, obrázky, kreslení lineárních cest, křivek, základních geometrických tvarů, transformace pomocí matice (škálování, rotace, přemístění), nastavení obrysů, výplní, kolizních oblastí (např. pro události myši), manipulaci s jednotlivými pixely, kompozici grafických objektů (skládání, odečítání), nebo práci s barvami, průhledností a stíny.

Na výpisu 9 je prezentován jednoduchý příklad práce s canvasem, konkrétně vykreslení modrého obdélníku.

```
<!DOCTYPE html>
<html>
  <body>
    <canvas id="example" width="400" height="200">
      Alternativni text.
    </canvas>
    <script type="text/javascript">
      var canvas = document.getElementById("example");
      var context = canvas.getContext("2d");
      context.fillStyle = "#0000FF";
      context.fillRect(20, 20, 100, 50); // x, y, sirka, vyska
    </script>
  </body>
</html>
```

Výpis 9: Použití HTML5 elementu Canvas

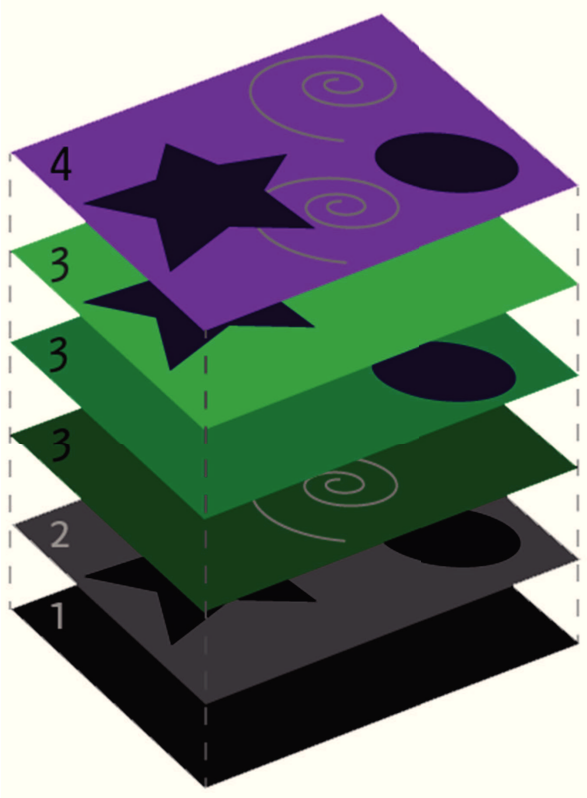
4.2.1 KineticJS

KineticJS je JavaScriptová knihovna pro práci s HTML5 canvasem. Je usilovně vyvíjena Ericem Rowellem a velmi často aktualizována (i několik verzí týdně). Nabízí jednoduché a intuitivní API pro efektivní tvorbu na sobě nezávislých dynamických grafických objektů s množstvím nastavitelných vlastností, metod a událostí.

Tato knihovna poskytuje vysoký grafický výkon i pro mnoho (tisíce) objektů díky systému vrstev. Ten je zapouzdřen do scény (objekt Stage) umístěné v libovolném kontejnerovém HTML elementu (např. div). Scéna je složena vždy z několika (minimálně 3) vrstev, jež jsou v základu tvořeny vlastním canvas elementem a mají svoji specifickou funkci. Rozložení vrstev ve scéně je patrné z obrázku 3.

Nejníže (1) se nalézá vyrovnávací (Buffer) vrstva, která je skrytá, uživateli neviditelná a slouží k detekci událostí na úrovni pixelů, ke kompozici objektů a vrstev, nebo např. k serializaci celé scény.

Další vrstvou v pořadí (2) je druhá skrytá vrstva a to tzv. zákulisí (Backstage). Tato vrstva obsahuje neviditelné obrazy objektů, detekuje jejich hrany, dráhy pohybu, kolize a vyvolává příslušné události. Tím, že jsou objekty neviditelné, není potřeba při každé



Obrázek 3: KineticJS systém vrstev [10]

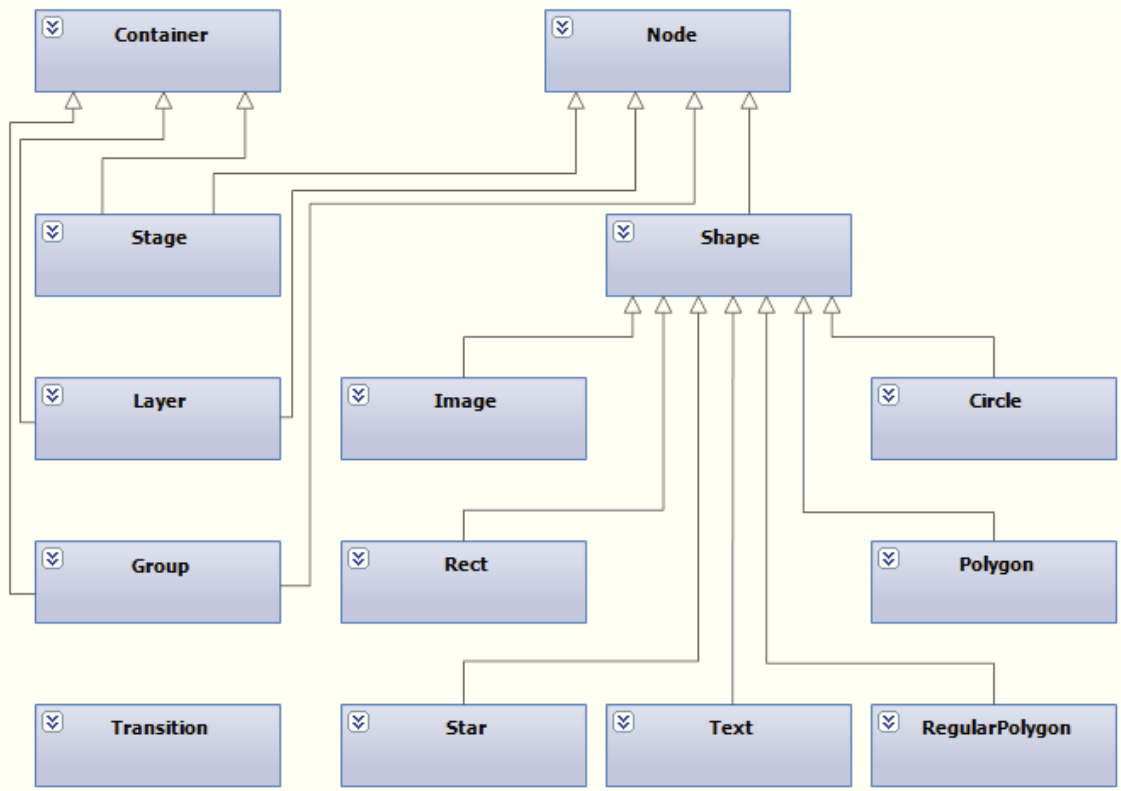
vyvolané události znovu vykreslovat celý canvas, což nese velký podíl na výkonnosti knihovny.

Nad těmito vrstvami se nachází libovolné množství uživatelem definovaných viditelných vrstev (3). Na ty je možné vykreslovat jakékoliv grafické objekty, nebo skupiny objektů. Jejich rozložení do vrstev je klíčové pro plynulost dynamicky se měnící scény s velkým počtem těchto objektů. Pro dosažení nejvyššího výkonu je doporučováno umístit do jedné vrstvy všechny statické objekty, u kterých se nepředpokládají žádné transformace (změny velikosti, tvaru, barvy apod.). V dalších vrstvách by měly být seskupeny dynamické objekty, které jsou transformovány současně. Speciální vrstvu je doporučeno vytvořit pro animace objektů – před jejím spuštěním je objekt přesunut do této animační vrstvy, v ní se provede celá animace a po jejím skončení je objekt opět přemístěn do své původní vrstvy. Takovýchto animačních vrstev lze samozřejmě dle potřeby vytvořit více. Výhoda opět spočívá v tom, že je s každým animačním rámcem překreslován pouze animovaný objekt, zatímco zbylá grafika zůstává nezměněna.

Poznámka 4.4 Z testování vyplývá, že pro velký počet objektů je potřeba nalézt optimální poměr mezi počtem vrstev (každý canvas element vyžaduje určitou režii systémových prostředků) a počtem dynamických objektů v jednotlivých vrstvách. Například

pro 10000 nezávislých pohyblivých objektů přináší rozdělení do 10 vrstev, každá s 1000 objekty, lepší výkon, než 5 vrstev po 2000 objektech, nebo 20 vrstev po 500 objektech.

Výsledný obraz celé scény (4) je dán sloučením všech viditelných, uživatelem definovaných vrstev s ohledem na jejich pořadí a na pořadí objektů v jednotlivých vrstvách (tzv. z index). Tím je umožněno vzájemné prostorové překrývání objektů a vytvoření plastického efektu scény.



Obrázek 4: Třídní diagram knihovny KineticJS v3.9.3

Ke knihovně KineticJS je poskytováno komplexní API popisující sadu tříd, které reprezentují jednotlivé prvky knihovny. Třídní diagram je znázorněn na obrázku 4. Základními „stavebními kameny“ jsou třídy Node a Container. Ty poskytují obecné prostředky pro práci s grafickými objekty, např. funkce Drag & Drop, pozicování, rotace, škálování, transformace, viditelnost, řazení, nastavení jména, hierarchie, vrstev, os, reakcí na události, práci s průhledností, selektory apod. Tyto třídy jsou rozšířeny o další specifické metody, čímž tvoří odvozené třídy Group (reprezentuje množinu seskupených objektů), Layer (představuje danou vrstvu) a Stage (vyjadřuje scénu). Z třídy Node je pak navíc odvozena abstraktní třída Shape reprezentující obecný grafický objekt, kterému je možno kromě vlastností uzlu nastavit např. výplň, obrys, nebo serializovat a deserializovat data. Shape je pak výchozí třídou (generalizací) pro třídy Circle (kruh), Image (obrázek), Polygon, Rect

(obdélník), RegularPolygon (pravidelný polygon), Star (hvězda) a Text. Tyto třídy představují konkrétní grafické prvky a poskytují specializované metody a vlastnosti dle svého určení. Knihovna nově poskytuje také statickou třídu Transition, která umožňuje transformace a plynulé přechody mezi hodnotami vlastností daného grafického objektu (např. plynulé zvětšování/zmenšování, rotace, přesun apod.).

Na výpisu 10 je prezentován jednoduchý příklad použití knihovny KineticJS. Jde o vykreslení zeleného obdélníku s černým obrysem, který se po najetí kurzoru myši plynule zvětší na dvojnásobnou velikost.

```
window.onload = function() {
    // vytvoreni sceny
    var stage = new Kinetic.Stage({
        container: "MyStage", // nazev HTML5 elementu
        width: 578,
        height: 200
    });

    // vytvoreni vrstvy
    var layer = new Kinetic.Layer();

    // vytvoreni grafickeho objektu
    var rectangle = new Kinetic.Rect({
        x: stage.getWidth() / 2,
        y: stage.getHeight() / 2,
        width: 100,
        height: 50,
        fill : 'green',
        stroke: 'black',
        strokeWidth: 4,
        centerOffset: {
            x: 50,
            y: 25
        }
    });

    // vytvoreni reakce na udalost
    rectangle.on("mouseover", function() {
        // transformace meritka
        this.transitionTo ({
            scale: {
                x: 2,
                y: 2
            },
            duration: 0.5,
            easing: "ease-in"
        });
    });

    // umisteni grafickeho objektu na vrstvu
    layer.add(rectangle);
```

```
    // umisteni vrstvy na na scenu
    stage.add(layer);
};
```

Výpis 10: Použití knihovny KineticJS

Kompletní dokumentace API se nachází na webu knihovny [10].

4.3 JSON

JavaScript Object Notation neboli JSON je formát pro zápis objektových dat v textové podobě, díky níž je jazykově i platformově nezávislý. Standardně je používáno kódování UTF-8. Výhodou tohoto formátu je jednoduchý zápis čitelný pro člověka a zároveň dobře zpracovatelný počítačem. Nejčastější použití nalézá v serializaci dat nebo jejich výměně mezi serverem a webovou aplikací. Tato data jsou přenášena jako objekt, pole, nebo jejich kombinace.

Objekt je realizován kolekcí párů klíč : hodnota, kde klíč je řetězec s názvem položky. Hodnotou může být libovolný řetězec, číslo, boolean, null, nebo další vnořený objekt, resp. pole. Pokud objekt obsahuje 2 a více párů, jsou odděleny čárkou. Celý objekt je uzavřen do složených závorek.

Pole ohraničují hranaté závorky, mezi nimiž se nalézají hodnoty oddělené čárkami. Hodnotou může opět být řetězec, číslo, boolean, null, objekt, nebo pole.

JSON je podporován všemi moderními prohlížeči. Vytváření JSON objektu a jeho zpětné parsování v JavaScriptu je vidět na výpisu 11.

```
function Message(type, value, data) {
    this.Type = type;
    this.Value = value;
    this.Data = data;
}
// vytvoreni nove zpravy
var recipients = new Array("Petr", "Pavel");
var message = new Message("greeting", "Zdravi", recipients);

// prevedeni objektu na JSON
// {"Type":"greeting", "Value":"Zdravi", "Data":["Petr", "Pavel"]}
var jsonString = JSON.stringify(message);

// prevedeni JSON na objekt
try {
    message = JSON.parse(jsonString);
    var text = message.Value + "␣" + message.Data[0]; // Zdravi Petr
} catch (e) {
    console.log("Data␣nejsou␣platnym␣formatem␣JSON.");
}
```

Výpis 11: Vytvoření a parsování JSON objektu

Kromě JavaScriptu je rovněž podporován ve většině ostatních jazyků, ať už nativně, nebo s využitím externích knihoven [11].

5 Analýza a implementace hry BANG!

První část této kapitoly bude věnována analýze karetní hry BANG! a specifikaci požadavků, odpovídajících zadání. Poté bude popsán způsob implementace klientské i serverové části řešení. Následně specifikuji způsob nasazení serverové i webové aplikace a vysvětlím ovládání samotné hry. Na závěr kapitoly budou shrnuta možná rozšíření a vylepšení.

5.1 Analýza

Požadavkem bylo vytvořit řešení založené na modelu server – klient s tím, že klientem bude klasický webový prohlížeč. Z prohlížeče je tedy potřeba se připojit na server specifikovaný svou adresou. Každá hra určená pro více hráčů musí nabízet dvě možnosti jak začít hrát – založit novou hru, nebo se připojit k již založené. V případě založení nové hry je potřeba stanovit, pro kolik hráčů má být založena. Aby si hráč mohl vybrat hru, ke které se připojí, je nutné zobrazit seznam založených her. Pro potřeby jednoduché identifikace bude každý hráč vystupovat pod unikátní přezdívkou a stejně tak musí mít každá hra jedinečný název. Ke hře je možné se připojit, pouze dokud není rozehrána. Z probíhající hry bude hráč moci kdykoli odejít, nebo se rovnou odpojít ze serveru. Pokud ze hry odejde poslední hráč, musí být hra ukončena a smazána. Hráčům jedné hry bude umožněno mezi sebou chatovat. Tyto funkční požadavky na klientskou část jsou graficky vyjádřeny diagramem případů užití na obrázku 5. Vybrané případy užití jsou popsány níže.

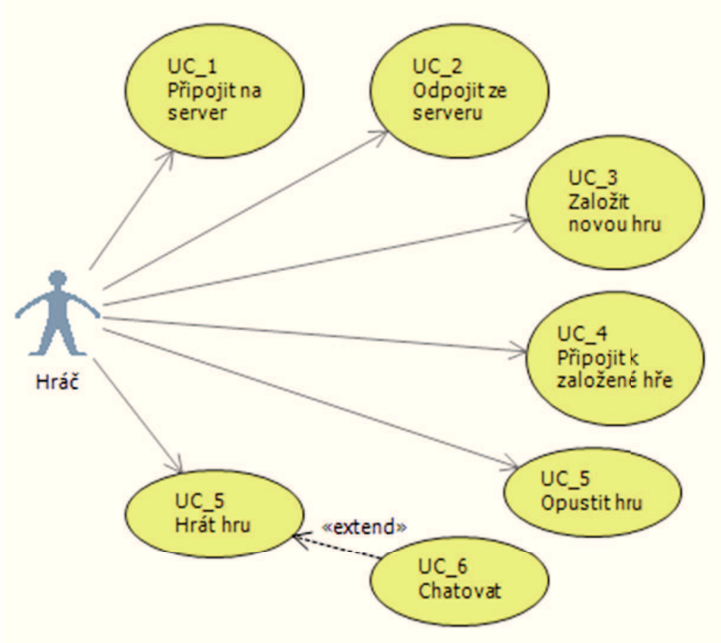
Případ užití: Připojit na server
ID: UC_1
Aktéři: hráč
Vstupní podmínky: Hráč má otevřenu stránku s hrou.
Tok událostí:

1. Hráč zadá přezdívku;
2. Hráč zadá adresu serveru;
3. Hráč stiskne tlačítko *Připojit k serveru*;
4. Systém se pokusí navázat spojení se serverem.

Alternativní tok:

1. Hráč zadá duplicitní přezdívku;
2. Systém vyzve hráče k zadání jiné přezdívky.

Případ užití: Založit novou hru
ID: UC_3



Obrázek 5: Diagram případů užití – klientská část

Aktéři: hráč
Vstupní podmínky: Hráč je připojen na serveru.
Tok událostí:

1. Hráč zadá název hry;
2. Hráč zadá počet hráčů;
3. Hráč stiskne tlačítko *Vytvořit hru*;
4. Systém vytvoří hru se zadanými parametry.

Alternativní tok:

1. Hráč zadá duplicitní název hry;
2. Systém vyzve hráče k zadání jiného názvu hry.

Případ užití: Připojit k založené hře
ID: UC_4
Aktéři: hráč
Vstupní podmínky: Hráč je připojen na serveru.
Tok událostí:

1. Hráč si vybere a označí požadovanou hru ze seznamu vytvořených her;

2. Hráč stiskne tlačítko *Připojit ke hře*;
3. Systém hráče připojí k požadované hře.

Alternativní tok:

1. Hráč označí hru ve stavu *Hraje se*;
2. Systém vyzve hráče k označení jiné hry.

Samotná hra bude implementována včetně všech pravidel (průběh hry, význam karet atd.), které byly popsány v kapitole 2, a které jsou detailně specifikovány v oficiální verzi [2].

Klient musí graficky zobrazovat celý průběh hry a umožnit interaktivně reagovat na všechny herní situace.

Server bude sloužit jako centrální uzel mezi připojenými hráči. Musí nést informace o všech hráčích i založených hrách. Dále musí umožňovat provoz několika her současně.

5.2 Návrh

Podle zpracované analýzy jsem vytvořil návrh rozdělení jednotlivých funkcí mezi klientskou a serverovou část. Komunikace mezi nimi bude založena na protokolu *WebSocket* a přenášená data budou ve formátu *JSON*. Obě technologie byly probrány v kapitole 4.

Za účelem sjednocení přenášených dat jsem navrhl třídu `Message`, která je implementována na straně klienta i serveru. Její konstruktor přijímá tři parametry, a to:

- type – řetězec sloužící k rozlišení účelu zprávy, může nabývat hodnot `"C"` (příkaz), `"R"` (požadavek), `"I"` (info), `"E"` (chyba), `"A"` (odpověď), nebo `"CH"(chat)`;
- value – řetězec reprezentující samotnou zprávu;
- data – pole řetězců vyjadřující dodatečné parametry nebo data.

Na základě požadavku, aby byl klientem webový prohlížeč a zároveň byly využity technologie HTML5, je zřejmé, že pro implementaci funkcionality na straně klienta bude použit JavaScript. Ten je už ze své podstaty mnohem pomalejší než kompilované [1] jazyky jako C# nebo Java, a proto bude z výkonnostních důvodů veškerá možná logika soustředěna do serverové části [12].

5.3 Klientská část

Klientskou částí je webová aplikace tvořená jednoduchou HTML5 stránkou, CSS, zdrojovými obrázky a především sadou skriptů v jazyce JavaScript. Jedná se o tzv. tenkého klienta, kdy je logika přesunuta na server.

[1] Termínu *kompilované* je užito pro zjednodušení.

Webová stránka obsahuje tři formuláře potřebné pro připojení k serveru, vytvoření nové hry a připojení se k již založené hře. Ty jsou podle potřeby pomocí JavaScriptové knihovny *JQuery* dynamicky skrývány nebo zobrazovány. Dále se na stránce nalézá div element, který slouží jako textová oznamocí oblast a během hry zároveň jako chatovací okno. Další div tvoří kontejner pro objekt scény Stage JavaScriptové grafické knihovny *KineticJS*, která byla podrobněji popsána v kapitole 4. Tento prvek reprezentuje vlastní hru – dovoluje vykreslovat grafiku, ovládací prvky a slouží rovněž jako interaktivní grafické uživatelské rozhraní (GUI). Kromě knihoven *JQuery* a *KineticJS* obsahuje aplikace skripty websocket, canvas a game.

Websocket

První ze skriptů, jak už název vypovídá, se stará o komunikaci se serverem skrz *WebSockets*. Je v něm využito *API WebSockets* popsané v kapitole 4.

Při požadavku na navázání spojení se serverem je vytvořena instance třídy WebSocket a v případě úspěšného spojení vyvolána událost onopen. Ta způsobí zviditělnění formulářů pro vytvoření hry a připojení k již vytvořené hře. Zároveň je od serveru přijata zpráva obsahující výpis existujících her. Tento seznam je spolu s informacemi o stavu hry (hraje se, čeká na hráče) a počtem aktuálně připojených hráčů ihned zobrazen ve formě tabulky.

Dále tento komunikační skript nabízí metody konverze JSON řetězce na typ Message a naopak. Tyto metody jsou postaveny na principu, který je uveden ve výpisu 11. K dalším patří sada pomocných dekódovacích metod, které slouží k parsování příchozích zpráv a volání konkrétních výkonných metod na základě typu a parametrů těchto zpráv.

Canvas

Druhý skript s názvem canvas slouží pro práci s grafickými objekty (především obrázky karet) a metodami knihovny *KineticJS*. Některé z metod canvasu nyní stručně představím.

Jako první je z tohoto skriptu volána metoda loadImages, která načte všechny zdrojové obrázky do paměti. Po ní následuje inicializační metoda initStage, jenž inicializuje scénu a vytvoří 4 vykreslovací vrstvy – jednu pro pozadí hrací plochy (pozadí, středová elipsa, přezdívky hráčů), druhou pro ovládací prvky (křížek, šipky), třetí pro všechny statické karty na ploše a čtvrtou pro vykreslování animací karet (přesun, odhození atd.). Další důležitou metodou je drawCard, která vytváří a inicializuje novou instanci karty a přiřadí jí odpovídající obrázek a reakce na události (dle typu karty – role, charakter, herní atd.). Neméně důležitou je metoda moveCard. Ta se stará o animaci přesunů karet po hrací ploše. K výpočtu bodů trajektorie přesunu využívá algoritmu DDA (Digital Differential Analyzer). Ten spočívá v postupném přičítání konstantních přírůstků k oběma souřadnicím. Tuto trajektorii následně karta svým pohybem opíše (u všech klientů stejně).

Dalšími metody se starají např. o změnu měřítka karet, zvýraznění okraje karty při jejím označení nebo třeba vykreslují a označují jména hráčů a jejich ovládací prvky.

Game

Poslední skript se jmenuje game a kromě spouštění inicializačních metod po načtení stránky se stará kupříkladu o test podpory *WebSockets*, aktualizaci seznamu založených her nebo o reakce na potvrzení jednotlivých HTML formulářů.

5.4 Serverová část

Pro implementaci serverové části jsem si vybral jazyk C#, se kterým mám pravděpodobně nejvíce zkušeností. Server je vytvořen jako klasická .NET konzolová aplikace, která jako spouštěcí parametry přijímá IP adresu a číslo portu, na kterém má naslouchat příchozím klientům. Je navržen pro současný provoz libovolného množství her. Architektura serveru je patrná z třídního diagramu na obrázku 6.

Základním stavebním kamenem je třída BangServer, která obsahuje reference na kolekci všech připojených hráčů a všech vytvořených her. Tato řídící třída poskytuje obsloužení hráče na úrovni serveru. Nabízí metody pro vytvoření nové hry, připojení hráče k dané hře, nebo naopak odpojení z ní, dekóduje zprávy přicházející od daného hráče, umožňuje poslat výpis založených her všem připojeným a nebo poslat zprávu jen určitému hráči.

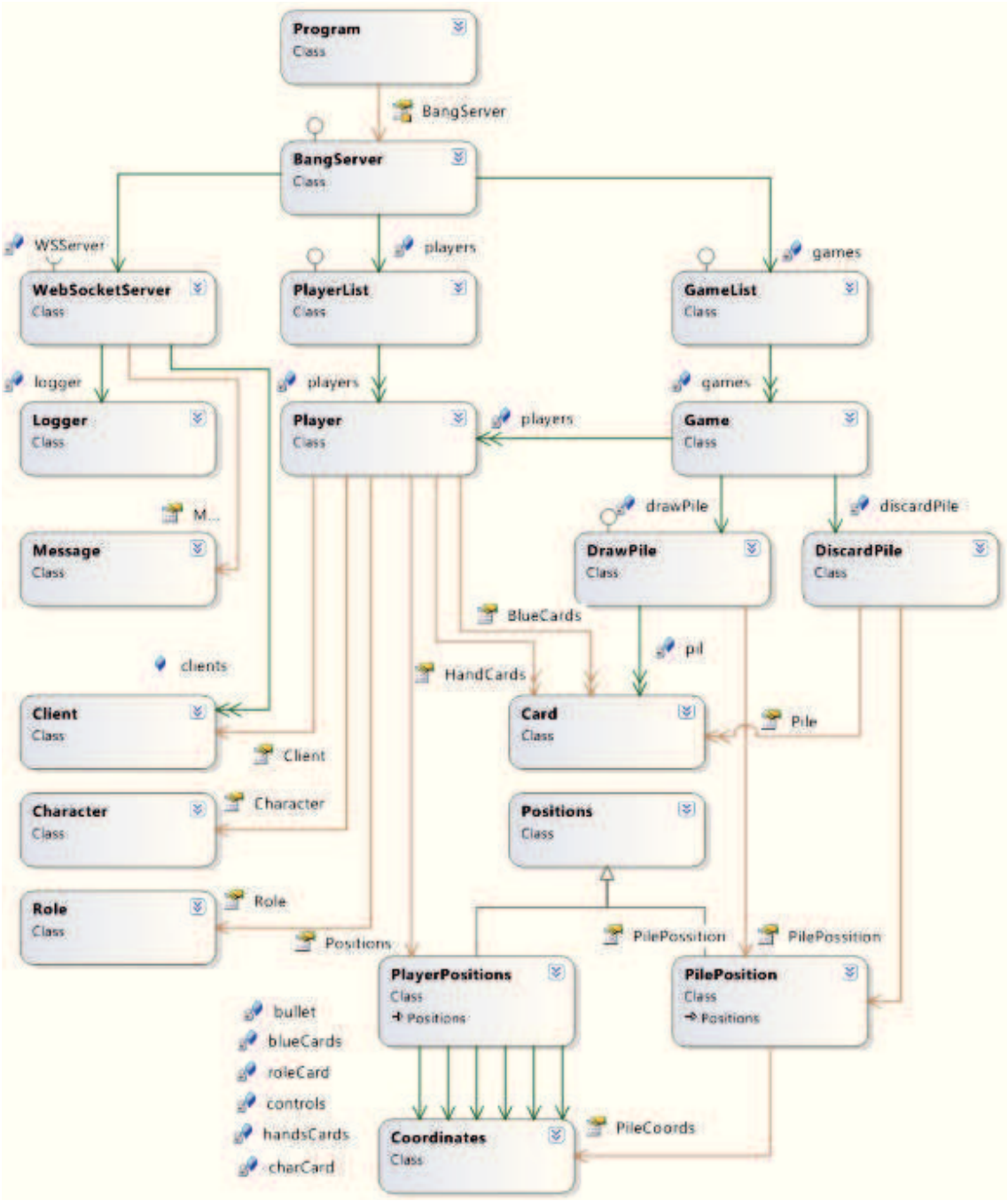
Všechny procesy spojené s navazováním spojení a komunikací s klientem jsou zapouzdřeny ve třídě WebSocketServer. Ta je z velké části implementací nejdůležitějších pasáží WebSocket protokolu [7]. Po své inicializaci naslouchá na zadané IP adrese a portu příchozím požadavkům na navázání spojení. Díky použití asynchronních verzí metod pro práci se sockety jsou prodlevy, v nichž by server nebyl schopen přijímat nová spojení, eliminovány na minimum.

V případě, že je signalizováno nové příchozí spojení, je vytvořen objekt třídy Client, kterému je předána reference na socket s tímto spojením. Následně je provedena extrakce klientské části a generování serverové části handshake (popsáno v kapitole 4). Průběh celého procesu je patrný ze sekvenčního diagramu na obrázku 7.

Třída WebSocketServer dále obsahuje metodu SendMessage, která přijímá jako parametr objekt typu Message, ten zkonvertuje na JSON řetězec, zakóduje do tvaru požadovaného pro WebSocket zprávu, kterou následně odešle. Metoda ReadMessage funguje analogicky.

Dalšími důležitými třídami jsou třídy Player a Game. Třída Player představuje aktuální stav hráče. Obsahuje kolekce typu Card pro reprezentaci karet, které má hráč v ruce, nebo vyloženy před sebou. Ve třídě rovněž najdeme odkazy na objekty typu Role a Character, které udávají, za jakou postavu a charakter hráč v dané hře vystupuje. V neposlední řadě je zde slovník shromažďující oprávnění hráče k různým úkonům. Skládá se z názvu oprávnění (klíč) a booleovské hodnoty.

Třída Game zastupuje jednu vytvořenou hru. Nalézá se v ní mimo jiné kolekce hráčů, kteří hru hrají, odkazy na objekty typu DrawPile (Lízací balíček) a DiscardPile (Odhazovací balíček) a 4 sady metod.



Obrázek 6: Třídní diagram BANG! – serverová část

První jsou metody inicializační. Ty obstarávají například zamíchání a rozdání karet, vytvoření instancí a nalosování postav a charakterů nebo počáteční nastavení karet s nábojnicemi (počty životů).

Druhou skupinou jsou metody řídící – vyhledávají a přidávají/odstraňují hráče, vyhledávají a lížou/vykládají/odhazují karty, střídají hráče na tahu apod. Klíčovou metodou je VerifyAction. Každý hráčův úkon (na straně klienta) je zaslán na server k validaci, a teprve pokud akce touto metodou projde, je klientovi zaslána kladná odpověď a úkon může být vykonán. Validace probíhá na základě jednotlivých oprávnění hráče proti naimplementovaným podmínkám, odpovídajícím pravidlům hry BANG!.

Třetí sadou jsou metody realizující funkci herních karet. Bang napadne vybraného hráče, Birra vyléčí život atd.

Do poslední kategorie patří metody pomocné, které umožňují přesun karty mezi stanovišti, zamíchat karty z odhazovacího balíčku a vytvořit nový lízací, označit hráče na tahu, nebo poslat jednomu, či více hráčům zprávu.

Pozice karet jsou vypočítávány ve formě souřadnic x, y a rotace a zapouzdřeny do objektů typu Coordinates. Hodnoty souřadnic jsou vraceny s připočítanou několikapixelovou odchylkou pro dojem rozházených karet. Z objektů typu Coordinates jsou složeny třídy PilePosition a PlayerPosition, které dědí konstanty a pomocné metody ke generování odchylek z třídy Position. PilePosition a PlayerPosition specifikují rozložení stanovišť jednotlivých hráčů na hrací ploše (stanovištěm je myšlena pozice karty postavy, role, karet v rukou, vyložených karet apod.).

5.5 Nasazení

Během vývoje byla klientská část nasazena dvěma způsoby a to pomocí:

- Vývojového serveru Microsoft Visual Studia 2010 – v prvních fázích vývoje,
- Lokálního IIS webového serveru – pro účely testování, další vývoj.

V prvním případě je potřeba k adrese přidat i port, na kterém byla webová stránka s hrou nasazena. Adresa může vypadat např. http://192.168.1.100:8080/BangWebApp/.

Při nasazení na IIS web server lze číslo portu vynechat např. http://192.168.1.100/BangWebApp/.

Pokud je projekt nasazován bez použití virtuálního adresáře BangWebApp, je pro úspěšné spuštění nutné provést velmi malou úpravu skriptu canvas.js. Instrukce k ní jsou popsány v souboru readme.txt.

Serverová část byla spouštěna jako klasická konzolová aplikace s parametry. Těmi je IP adresa a číslo portu, na kterém server naslouchá novým příchozím spojením. Výchozí parametry jsou nastaveny na adresu 127.0.0.1 (localhost) port 8181.

Testování probíhalo na jednom až čtyřech počítačích v rámci lokální sítě v průběhu celého vývoje.

5.6 Ovládání hry

Při navštívení stránky, na které se hra nachází, má uživatel možnost zadat adresu herního serveru, na který se chce připojit (nebo ponechat výchozí), a svou přezdívku, pod kterou bude na serveru i při samotné hře vystupovat. Po kliknutí na tlačítko *Připojit k serveru*

je ihned navázáno spojení, což je potvrzeno i informací v textovém okně. Tamtéž je v případě problému s navázáním spojení vypsána příčina neúspěchu.

Následně si může uživatel vybrat, zda chce vytvořit vlastní novou hru, nebo se připojit k již založené, která čeká na dostatečný počet hráčů. V prvním případě je nutné zadat název hry, zvolit pro kolik hráčů bude hra určena a stisknout tlačítko *Vytvořit hru*. Poté bude zobrazena prázdná hrací plocha a uživatel musí vyčkat, než se připojí zbývající hráči.

Pokud má uživatel v úmyslu přidat se k vytvořené hře, musí ji kliknutím označit v seznamu vytvořených her a stisknout *Připojit ke hře* (Na obrázku 8 v příloze). Následně je zobrazena prázdná hrací plocha. Byl-li tímto připojením dosažen potřebný počet hráčů, jsou zobrazeny stanoviště hráčů a započne rozdávání karet.

Hráč, který je na řadě, má přezdívku umístěnou po obvodu středové elipsy zvýrazněnu červeně.

Líznutí jedné, nebo více karet se docílí kliknutím na svrchní kartu lízacího balíčku, který se nachází v pravé polovině středové elipsy.

Zahrání karty je podle jejího typu dvojí. První typ karet (např. Pivo, Vedle atd.) je zahrán kliknutím na její plochu. Poté je vykonána její funkce. Zahrání druhého typu karet (Bang, Vězení apod.) spočívá v jejím označení prvním kliknutím myši (okraj karty se zvýrazní) a druhém kliknutí na kartu protihráče.

Všechny typy karet kromě karty s nábojnicemi je možné pro lepší rozpoznání dvojklikem zvětšit a stejným způsobem vrátit do původní velikosti.

Karta s nábojnicemi slouží k signalizaci aktuálního počtu životů hráče. Odebrání života se provádí jedím kliknutím na tuto kartu s nábojnicemi.

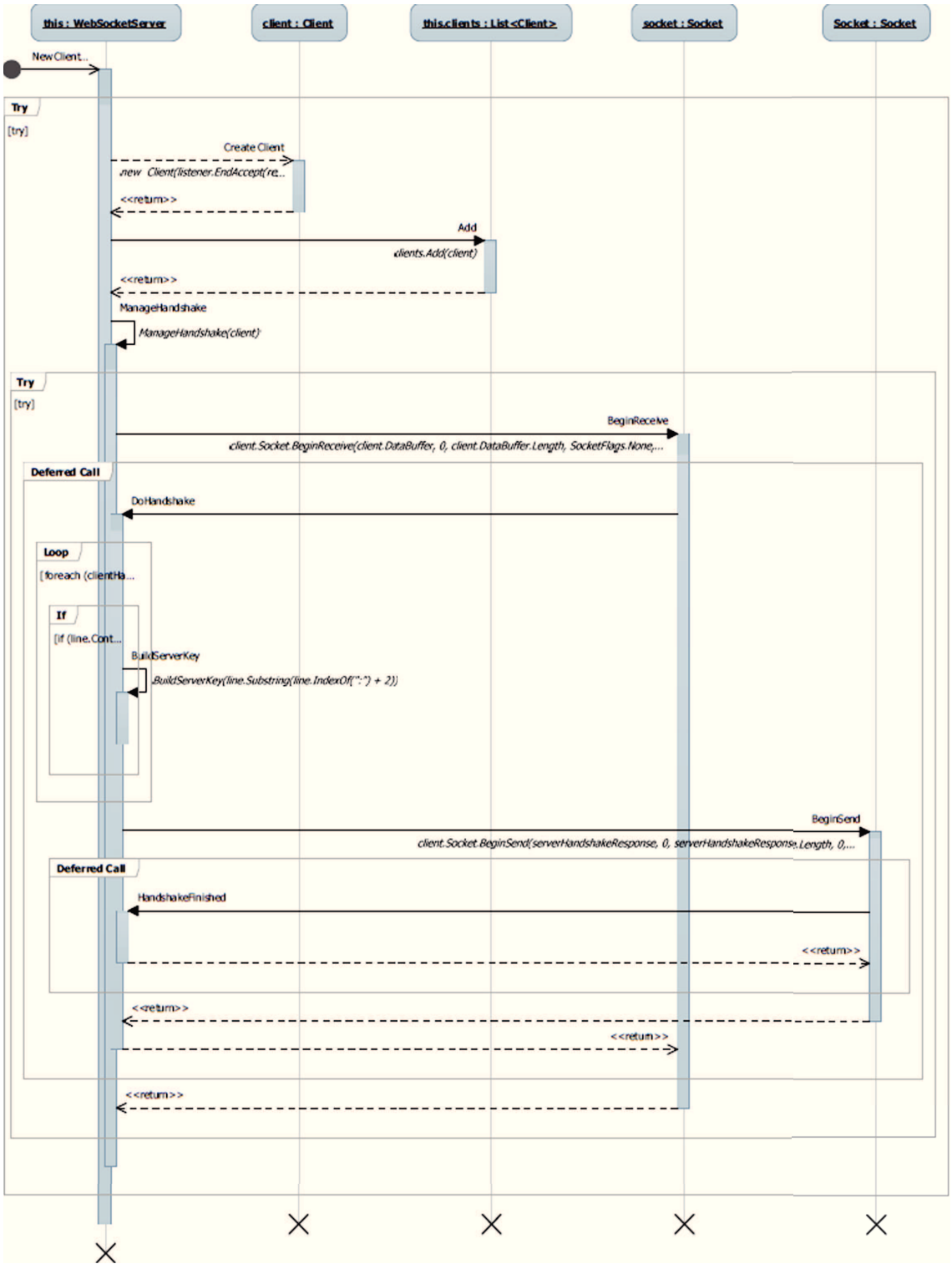
Pro případ nutnosti odhození přebytečných karet slouží značka červeného křížku nalevo od karty s nábojnicemi. Po kliknutí se křížek zvýrazní a hráč přejde do odhazovacího módu. Kdykoli v něm hráč klikne na svou kartu (v rukou), je tato karta odhozena do odhazovacího balíčku, který je umístěn v levé polovině středové elipsy. Ukončení tohoto módu je možné opětovným kliknutím na značku červeného křížku (zvýraznění zmizí), nebo ukončením hráčova kola (viz níže.).

Pokud chce hráč ukončit své kolo a jsou k tomu splněny všechny podmínky (např. počet zbývajících karet), musí kliknout na bílou značku znázorňující dvě půlkruhové šipky, umístěnou vlevo od karty s nábojnicemi. Poté je označena přezdívka následujícího hráče.

Průběh hry je možné sledovat v podobě krátkých informací, které jsou vypisovány v textovém poli pod hrací plochou.

Hru je možné kdykoli opustit kliknutím na tlačítko *Opustit hru*, nebo se zcela odpojit ze serveru tlačítkem *Odpojit*.

Obrázek 9 v příloze ilustruje průběh hry.



Obrázek 7: Sekvenční diagram – připojení nového klienta (zjednodušeno)

6 Závěr

Cílem teoretické části mé práce bylo seznámit se se současnou vývojovou verzí nového standardu HTML5 a vybrané části pak popsat. Po pečlivém prostudování jednotlivých specifikací, odborných článků i názorů z praxe jsem došel k následujícímu závěru.

Přestože je velké množství HTML5 technologií stále usilovně vyvíjeno, a podle očekávání ještě několik let vyvíjeno bude, většina z nich je již v současné době moderními prohlížeči podporována a tato podpora stále narůstá. Proto je možné tyto technologie při vývoji webových aplikací využít již nyní.

V rámci praktické části jsem měl za úkol analyzovat karetní hru BANG!, poté vytvořit návrh online webové verze této hry, která by měla umožnit hraní více hráčů přes webový prohlížeč, a nakonec podle tohoto návrhu s využitím některých HTML5 technologií hru implementovat. Nejlépe se z nich pro tento úkol hodilo využití nového HTML5 elementu *canvas* a technologie *WebSocket*. Element *canvas* prostřednictvím moderní JavaScriptové knihovny *KineticJS* poskytoval výborné výsledky pro vykreslování herní grafiky přímo do webové stránky. Neméně významným prvkem bylo využití obousměrné komunikace pomocí *WebSockets*, která do prostředí webu přináší potřebnou interaktivitu se serverem. Věřím, že tyto i mnohé další HTML5 technologie se budou nadále úspěšně rozvíjet a budou stále častěji využívány pro tvorbu funkcionálně bohatých webových aplikací.

Přestože je hra v současném stavu hratelná, chybí oproti originální verzi doimplementovat některá podrobná pravidla. Dalším prvkem, který by jistě stál za pozornost, je zlepšení grafického ztvárnění jak webu, na kterém je hra umístěna, tak i samotné hry. S ohledem na množství a rozmanitost oficiálních i neoficiálních rozšíření hry BANG! se pro budoucí vývoj nabízí právě jejich implementace a integrace do prozatím vytvořené základní verze.

Výrazným vylepšením funkcionality by pak byla tvorba databáze, ve které by mohly být ukládány herní účty registrovaných uživatelů nebo například vedena statistika odehraných her a výsledků jednotlivých hráčů.

Těmito úkoly nad rámec zadání této bakalářské práce se budu s největší pravděpodobností zabývat i po jejím dokončení. Hra by pak mohla získat další rozměr a možná i poskytnout zábavu nejen fanouškům stolní verze hry BANG!.

Nezanedbatelný význam pro mě měla práce v tom, že jsem nabyl nových cenných zkušeností s vývojem v prostředí webu a ověřil si tyto teoretické poznatky v praxi.

7 Reference

[1] BoardGameGeek. *Bang!* [online]. c2002, [cit. 2012-04-14]. Dostupné z: http://boardgamegeek.com/boardgame/3955/bang

[2] CORBELLI a VOICU. Bang rules. In: *BANG!* [online]. Fourth edition. I-06131 - Perugia - Italy: daVinci Editrice S.r.l., 2008 [cit. 2012-04-15]. Dostupné z: http://www.davincigames.net/bang/bang_rules.pdf

[3] WHATWG. *HTML* [online]. 2012, [cit. 2012-04-15]. Dostupné z: http://www.whatwg.org/specs/web-apps/current-work/multipage

[4] WHATWG. *FAQ* [online]. 2012, [cit. 2012-04-15]. Dostupné z: http://wiki.whatwg.org/wiki/FAQ

[5] HUNT, Lachlan. A Preview of HTML 5. *A List Apart Magazine* [online]. 4.12.2007 [cit. 2012-04-16]. ISSN 1534-0295. Dostupné z: http://www.alistapart.com/articles/previewofhtml5

[6] HICKSON, Ian. *The WebSocket API: Editor's Draft* [online]. 2.4.2012 [cit. 2012-04-16]. Dostupné z: http://dev.w3.org/html5/websockets/

[7] FETTE a MELNIKOV. *The WebSocket Protocol* [online]. Prosinec 2011, s. 71 [cit. 2012-04-16]. ISSN 2070-1721. Dostupné z: http://tools.ietf.org/html/rfc6455

[8] RANGANATHAN a SICKING. *File API* [online]. 29.3.2012 [cit. 2012-04-18]. Dostupné z: http://dev.w3.org/2006/webapi/FileAPI/

[9] HERMAN a RUSSELL. *Typed Array Specification* [online]. 2.4.2012 [cit. 2012-04-18]. Dostupné z: https://www.khronos.org/registry/typedarray/specs/latest

[10] ROWELL, Eric. *KineticJS* [online]. [cit. 2012-04-22]. Dostupné z: http://www.kineticjs.com

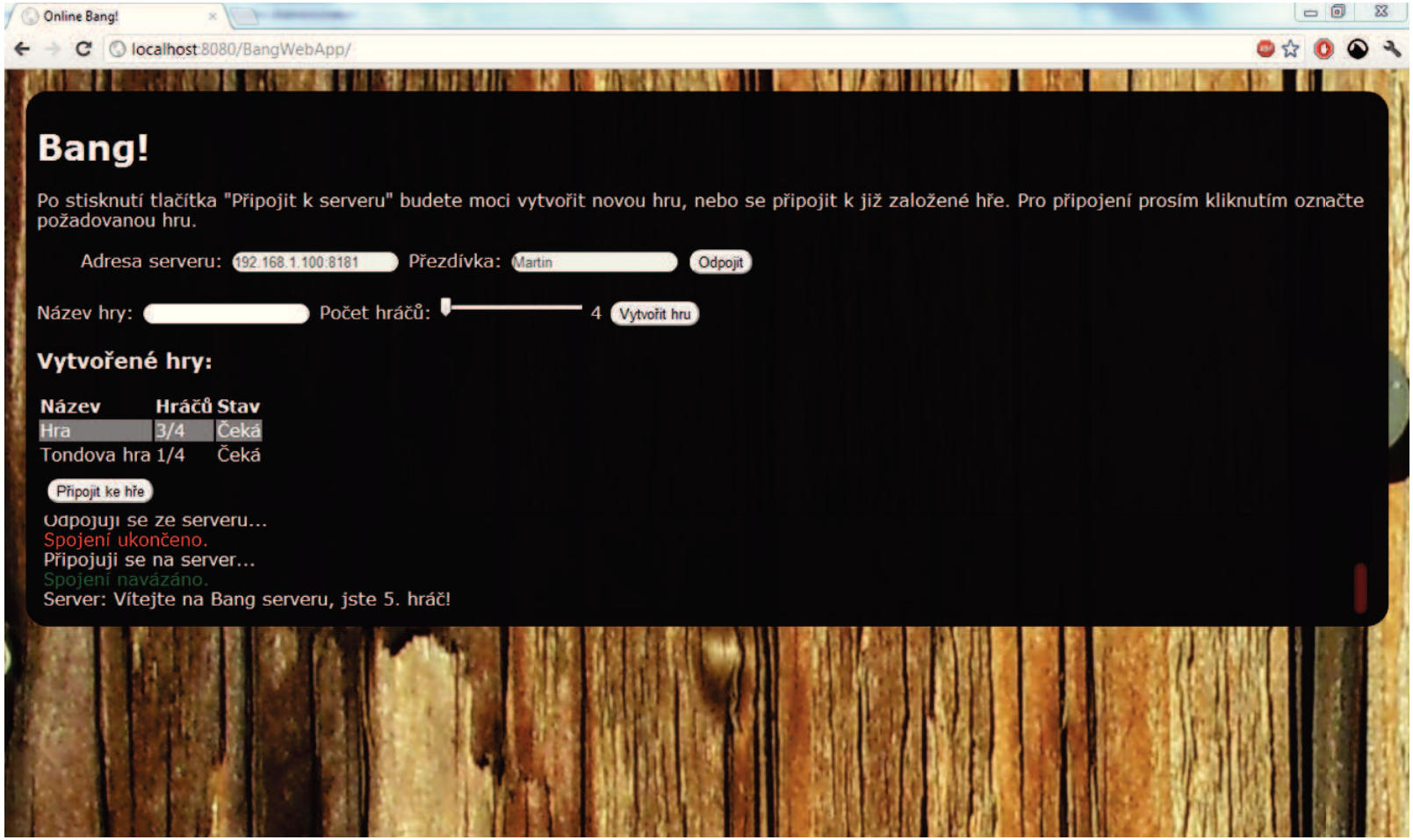
[11] CROCKFORD, D. *JSON* [online]. [cit. 2012-04-22]. Dostupné z: http://http://www.json.org

[12] RAPPL. Operation Performance Evaluation. *The Code Project* [online]. 18.1.2012 [cit. 2012-04-27]. Dostupné z: http://www.codeproject.com/Articles/304935/Operation-Performance-Evaluation

A Obsah CD

Na přiloženém CD se nalézá adresář Bang obsahující kompletní řešení vytvořené ve vývojovém prostředí Microsoft Visual Studio 2010.

B Snímky



Obrázek 8: Snímek z přípravy na hru

Obrázek 9: Snímek z probíhající hry