Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Návrh a implementace miniher pro hru Space Traffic

Plzeň 2016 Bc. Martin Hron

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 12. května 2016

Bc. Martin Hron

Poděkování

Tímto bych chtěl poděkovat vedoucímu práce Ing. Petru Vaněčkovi, Ph.D. za cenné rady, připomínky a odborné vedení této práce.

Dále děkuji Bc. Veronice Švecové a Bc. Janu Kotalíkovi za vedení projektu Space Traffic a také za možnost realizace této diplomové práce.

Nakonec bych chtěl poděkovat rodině za podporu a umožnění studia na vysoké škole.

Abstract

This diploma thesis deals with the issues of the minigames design and implementation into the Space Traffic web game. This game is developed at the Department of Computer Science and Engineering University of West Bohemia in Pilsen. The project aims to increase the interest of high school students in studying at the department and also to present student work.

The main aim is to design and implement an interface for integration of the minigames into the game. The next aim is to implement at least two minigames and integrate them into the project.

The thesis describes issues of the web games, Space Traffic project and principles of SOA. The next part focuses on the design of the interface parts including their implementation and the implementation details of both minigames. The last part is dedicated to testing and the evaluation of results.

Abstrakt

Tato diplomová práce se zabývá problematikou návrhu a implementace miniher do webové hry Space Traffic. Tato hra je vyvíjena na Katedře informatiky a výpočetní techniky Západočeské univerzity v Plzni. Cílem projektu je zvýšení zájmu studentů středních škol o studium na katedře a možnost prezentace studentské činnosti.

Hlavním cílem práce je navrhnout a implementovat rozhraní umožňující integraci miniher do této hry. Dalším cílem je implementovat alespoň dvě minihry a začlenit je do projektu.

Práce popisuje problematiku webových her, projekt Space Traffic a principy SOA. Další část práce je zaměřena na návrh jednotlivých částí rozhraní včetně jejich realizace a na implementační detaily obou miniher. Poslední část je věnována testování a zhodnocení dosažených výsledků.

Obsah

1 Úvod

Tato diplomová práce se zabývá problematikou návrhu a implementace miniher do hry Space Traffic vyvíjené na Katedře informatiky a výpočetní techniky Fakulty aplikovaných věd Západočeské univerzity v Plzni. Jedná se o výukovou webovou hru v prostředí vesmíru, kde je hlavním motivem obchodování s možností využití programovatelných prvků. Hlavním cílem tohoto projektu je zvýšení zájmu studentů středních škol o studium na katedře a zároveň má tato aplikace sloužit k prezentaci studentské činnosti.

Hlavním cílem práce je navrhnout a implementovat rozhraní, které bude umožňovat integraci miniher do Space Trafficu. Toto rozhraní musí být navrženo v souladu s herním designem a architekturou aplikace a mělo by umožňovat začlenění co možná největšího množství her různých žánrů. Dalším cílem je implementace alespoň dvou miniher a jejich integrace do projektu jako příklad použití navrženého rozhraní. Vytvořené hry a jejich integrace do Space Trafficu musí být následně důkladně otestovány.

Práce nejprve popisuje problematiku webových her, jejich stručnou historii a také rozdělení dle různých kategorií. Dále pak charakterizuje společné herní prvky a definuje pojem minihry. V další části je popsán projekt Space Traffic. Rozebírá se zde historie a současnost projektu a také architektura aplikace. Následně je nastíněna problematika servisně orientované architektury (SOA) a její principy. V této kapitole je pak ještě vysvětleno, co to jsou webové služby a technologie Windows Communication Foundation.

Další část práce se věnuje samotnému návrhu rozhraní. Nejprve jsou specifikovány odlišnosti oproti hernímu designu. Dále jsou nastíněny možnosti spouštění her, odměňování hráčů a integrace miniher. Na základě těchto možností jsou pak diskutována zvolená řešení. Následující část práce popisuje implementační detaily všech částí rozhraní. Je zde charakterizována například úprava databáze, hlavní prvek rozhraní nebo životní cyklus her. Implementační část následně ještě vysvětluje jednotlivé detaily realizace obou miniher včetně popisu činnosti.

Poslední část práce se věnuje testování nejen všech součástí rozhraní, ale také jednotlivých miniher a jejich začlenění do projektu. V závěru jsou pak zhodnoceny dosažené výsledky a možná další rozšíření.

2 Webové hry

V této kapitole se budeme zabývat především webovými hrami a herními prvky. Nejprve bude vysvětleno co je webová hra a následně se podíváme na stručnou historii video her a webových her. Dále pak stručně rozebereme současné technologie těchto her a následně bude probráno jak a dle jakých kritérií se mohou hry dělit. Na závěr si přiblížíme herní prvky společné pro všechny hry a také co to vlastně jsou minihry.

2.1 Co jsou webové hry

Počítačovou hru můžeme označit jako webovou, pokud se k jejímu ovládání využívá webový prohlížeč. Tyto hry bývají postaveny na standardních webových technologiích, případně využívají různé pluginy. Jejich hlavní výhodou je velká dostupnost z většiny moderních webových prohlížečů a také přenositelnost. Další výhodou je často i nenáročnost na hardwarové požadavky a také to, že není nutné instalovat velké množství (mnohdy i objemných) souborů. Nevýhodou pak může být omezená funkcionalita závislá na možnostech prohlížeče. [Bro(2016)]

2.2 Historie

2.2.1 Historie videoher

První milník v oblasti počítačových her se podle [Tis(2011)] datuje k roku 1952. V tomto roce Alexander S. Douglas vytvořil v rámci své disertační práce na Cambridgeské univerzitě první hru, která využívala grafický displej. Hra na motivy piškvorek zvaná *OXO* běžela na počítači EDSAC[1].

V roce 1958 pak byla vytvořena hra zvaná *Tennis for Two*, která jako zobrazovací zařízení využívala osciloskop. Jejím autorem byl americký fyzik

[1]EDSAC neboli Electronic Delay Storage Automatic Calculator byl první v praxi využitelný počítač s uloženým programem postavený na základě John Von Neumannova dokumentu First Draft of a Report on the EDVAC. [Eds(2016)]

Wiliam Higinbotham. Jako další milník se pak dá považovat rok 1962, kdy student MIT Stephen Russell vytvořil hru s názvem *Spacewars*. O rok později se tato hra stala první rozšířenou a dostupnou hrou a následně jí Ken Thompson přepisoval pro operační systém Multics. [Tis(2011)]

Dalším milník v historii počítačových her je vývoj *Computer Space* (odnož *Spacewars*) jako arkádové hry Nolana Bushnella a Teda Dabneye v roce 1971. Tito lidé pak o rok později zakládají firmu Atari a vyvíjí první komerčně úspěšnou hru s názvem *PONG* [Tis(2011)]. V následujících letech se pak začala objevovat různá herní zařízení (konzole) a hry se tak začínaly objevovat i v domácnostech. Na počátku 80. let pak dosáhly arkádové hry svého vrcholu a s nástupem osobních počítačů se herní průmysl (především z důvodu nízkých nákladů vydavatelů) začal zaměřovat na tyto platformy. [His(2016)]

2.2.2 Historie webových her

Historie webových her je úzce spjata s vývojem webových technologií. První hry vznikaly na přelomu let 1995-1996 a byly postaveny především na DHTML a JavaScriptu. Právě v roce 1996 vyvinul Mehul Patel strategii nazvanou *Earth: 2025*. V této strategii bylo hlavním úkolem vybudovat stát v postapokalyptickém světě.

Nejpopulárnější se, ale v roce 1999 stala strategická hra *Utopia*, která se hraje prakticky dodnes. V tomto roce vznikly ještě další dvě stále běžící hry a to fotbalový manažer *Hattrick* a MMORPG[2] *RuneScape*. Obě hry ještě v dnešní době hrají stovky tisíc uživatelů po celém světě. [Puk(2009)]

S nástupem Flashe resp. Flash Playeru začaly webové hry nabírat další rozměr. Díky Flashy bylo možné ve hrách využívat audio a video a hry tak začínaly působit na emoční stránku hráče. Tato technologie také umožňovala vytvářet graficky náročnější prvky a zejména díky své jednoduchosti se zapříčinila o velký rozvoj webových her. [Puk(2009)]

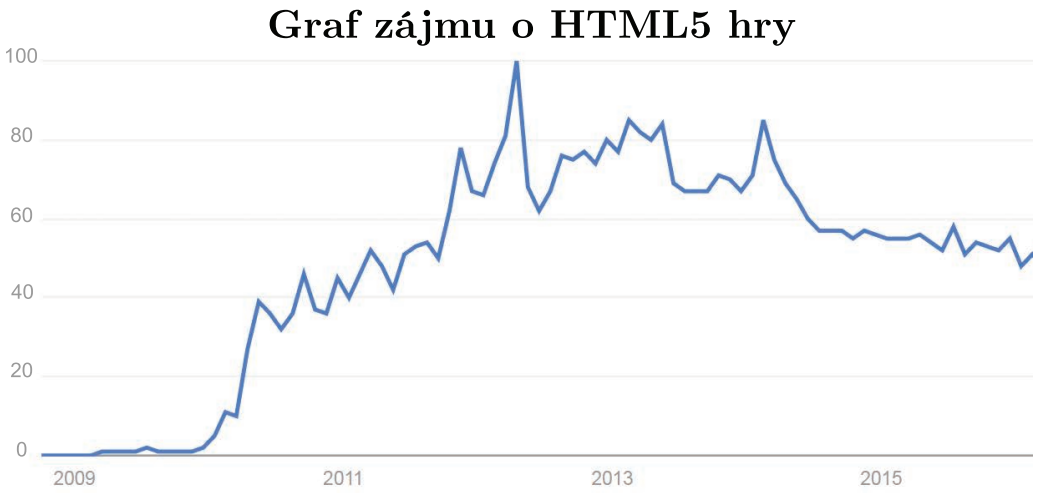
[2]Více viz kapitola 2.4.2.

2.3 Současné technologie webových her

V současné době se začínají webové hry stále více přesouvat z technologií jako Adobe Flash, Java či Unity do nativních webových technologií. Mezi tyto technologie můžeme zařadit **HTML5**, **CSS3** a **JavaScript**. Hry vytvořené v těchto technologiích se pak nazývají jako **HTML5 hry**. Při tvorbě **HTML5 her** se pak pro vykreslování často využívají elementy jako **canvas**, **SVG** či javascriptové API **WebGL**[3]. Především i díky těmto prvkům se webové hry čím dál tím více přibližují klasickým počítačovým hrám.

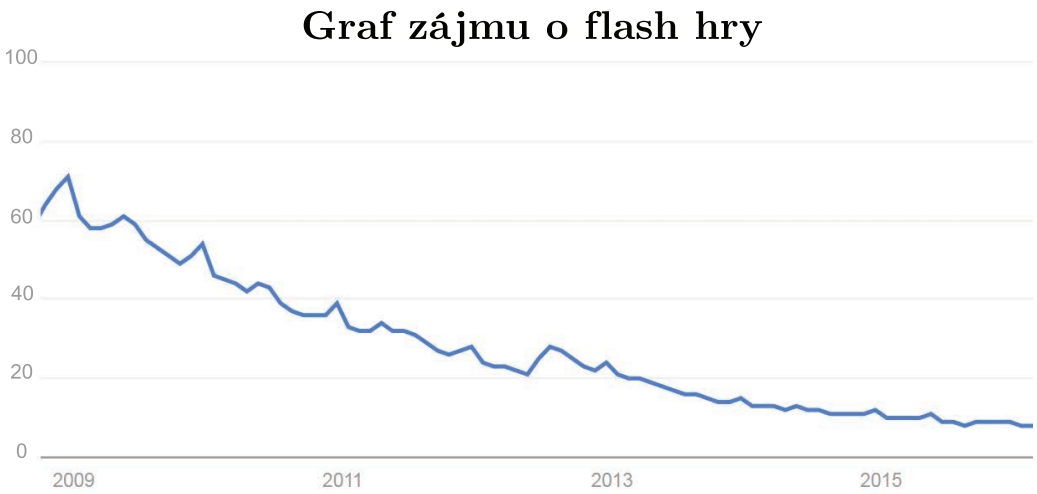
Tento trend způsobují zcela jistě dvě zásadní věci. První z nich je rozvoj internetu na mobilní zařízení a také snaha přizpůsobovat webové stránky pro tato zařízení. Druhou pak je, že aktuálně nejrozšířenější technologie Flash (dle [Ara(2009)] měl v roce 2009 zastoupení 99%) přestává být podporována v prohlížečích jak na mobilech, tak na běžných počítačích. Zde ho dokonce někteří vývojáři prohlížečů začínají blokovat [Mic(2015)].

Zájem o **HTML5 hry** a „nezájem“ o hry dělané ve flashy dokazují grafy na obrázcích 2.1 a 2.2. Tyto grafy znázorňují poměr vyhledávání daného výrazu Googlem vůči všem vyhledáváním od roku 2009 do současnosti. Z grafů můžeme vyčíst, že zájem o flash hry neustále klesá. Oproti tomu zájem o HTML5 hry od roku 2009 poměrně hodně narostl a to i přes to, že v letošní a loňském roce mírně opadl.



Obrázek 2.1: Graf zájmu o HTML5 hry. [Htm(2016)]

[3]WebGL neboli Web Graphics Library je API umožňující rendrovat 2D a především 3D grafiku v prohlížeči bez podpory různých pluginů.



Obrázek 2.2: Graf zájmu o flash hry. [Fla(2016)]

2.4 Rozdělení her

Hry můžeme dělit podle různých kritérií. Nejpoužívanějšími kritérii pak bývá dělení podle **žánru**, podle **počtu hráčů**, podle **herní platformy** a také podle **využití internetu** (online/offline). Jedna hra se pak může prolínat několika odvětvími a mohou tak vznikat zcela unikátní hry a herní žánry. V následujících podkapitolách si podrobněji popíšeme rozdělení her podle žánrů a také podle počtu hráčů.

2.4.1 Rozdělení dle žánru

Herních žánrů je dnes díky kombinaci několika základních žánrů velké množství a proto by ani nebylo možné všechny popsat. Vyjmenujeme si proto tedy pouze ty základní, resp. ty, které se nejvíce používají. Podle [Fai(2001)] jsou to tyto:

- **Akční hry** – Tento druh her je jeden z nejrozšířenějších a také jedním z prvních žánrů. Tyto hry obvykle obsahují násilí a střelbu. Často se také dělí na další podžánry.
  - *Klasické* – Povětšinou hry s vesmírným tématem např. Spacewars.
  - *Bludišťovky* – Tento druh her se dnes již příliš neobjevuje, patří sem např. Bomberman nebo Pacman.

  - *Shoot'em up* – V tomto žánru jde především o to sestřelit co nejvíce nepřátel. Zástupci jsou např. Space Invaders nebo Raptor.
  - *Plošinovky* – Tento žánr přináší spoustu skákání a lezení po různých plošinách a v současné době nabírá na popularitě hlavně na mobilních zařízeních. Patří sem legendární Super Mario Bros nebo Dangerous Dave.
  - *Bojovky* – V bojovkách je hlavním cílem umlátit protihráče pomocí bojového umění. Tyto hry se z počátku hodně objevovaly na starých automatech a patří sem hry jako Tekken či Mortal Kombat.
  - *First-person shooter* – Střílečka z pohledu první osoby, často vyžaduje rychlé reakce a práci s myší. Zástupců je velké množství, např. Wolfenstein 3D nebo DOOM.
  - *Third-person shooter* – Střílečka z pohledu třetí osoby, dříve často nazývaná jako „tomb raidrovka“, umožňuje větší volnost postavy a lepší interakci s prostředím. Nejznámějšími zástupci jsou Tomb Raider nebo Mafie.

- **Sportovní hry** – Do této oblasti spadají všechny hry, ve kterých je hlavním námětem jakýkoliv sport. Mohou být zpracovány buď jako simulace nebo jako arkády[4]. Do této kategorie mohou spadat i některé závodní hry. Zástupce tvoří hry ze série FIFA či NHL nebo i Tennis for Two zmíněný v kapitole 2.2.1.

- **Role playing games** – Role playing game, zkráceně RPG se dá volně přeložit jako hra na hrdiny. Hlavním námětem tohoto žánru je postava (hlavní hrdina), do které se hráč vžívá, prochází s ním příběhovou linií hry a vylepšuje jeho schopnosti. Těmto hrám se dříve také přezdívalo „Diablovky“ podle hry Diablo ztvárňující tento žánr. Do této kategorie dále patří např. Zaklínač nebo i nově chystaná česká hra Kingdom Come: Deliverance.

- **Strategické hry** – Hry tohoto typu nejvíce cílí na hráčovo analytické myšlení a schopnost plánování. Stejně jako akční hry si strategie rozdělíme na dva nejdůležitější podžánry.

  - *Tahové* – V těchto strategiích se, jak napovídá název, postupuje hrou po jednotlivých tazích či kolech. Při hře pro více hráčů se

[4] Arkádové hry na rozdíl od simulace ve většině případů změkčují či dokonce popírají fyzikální zákony.

pak jednotliví hráči střídají po kolech. Do této kategorie patří např. Heroes of Might and Magic či Civilization.

- *Real-timové* – V tomto žánru hra neprobíhá po tazích, ale veškeré úkony jsou prováděny v reálném čase. Jsou zde tedy vyžadována rychlá rozhodnutí, kde každé z nich může naprosto zvrátit vývoj hry. Patří sem především legendární hry ze sérií Warcraft, Starcraft nebo Age of Empires.

- **Adventury** – Adventury mají za cíl provést hráče nějakým příběhem. Hráč může v průběhu vyprávění příběhu vykonávat různé úkoly, řešit všelijaké hádanky a především může na základě svých rozhodnutí příběh ovlivňovat. Do tohoto žánru patří české série Polda a Horké Léto nebo poměrně nová hororová adventura Until Dawn.

- **Simulátory** – Simulátory se co možná nejvěrnějším způsobem snaží napodobit chování nějakého systému nebo činnosti z reálného světa. Simulátory mohou obsahovat spoustu podžánrů, například z pohledu dopravních prostředků, obchodování, života či jiného odvětví. Zařadit sem můžeme např. sérii The Sims, SimCity nebo Euro Truck Simulator.

- **Ostatní** – Ostatních žánrů může být spousta a mezi ty nejčastější budou jistě patřit logické hry, hudební a taneční hry, puzzle hry[5] či reklamní nebo erotické hry.

2.4.2 Rozdělení podle počtu hráčů

Základním dělením her podle počtu hráčů je rozdělení na hry **pro jednoho hráče** tzv. singleplayer a na hry **pro více hráčů** tzv. multiplayer. Jedna hra pak může obsahovat jak singleplayer, tak multiplayer. Multiplayer se dá dále rozdělit na **hry hrané na jednom zařízení** a na **hry hrané po síti či online**.

Multiplayer na jednom zařízení je možné realizovat třemi způsoby. Ten první a pravděpodobně nejčastější bývá většinou realizován **společnou obrazovkou** pro všechny hráče a **využitím více ovládacích zařízení nebo případným rozdělením klávesnice**. Druhým způsobem je tzv. **Split Screen** neboli rozdělená obrazovka, kde každý z hráčů má přidělenou část obrazovky, na které se zobrazuje jeho aktuální stav hry. Ovládaní je možné vyřešit obdobným způsobem jako u první varianty.

[5]Hry, kde je hlavním úkolem něco spojit či poskládat.

Třetí variantou může být tzv. **Hotseat**. Tento způsob se nejčastěji využívá u tahových her a jeho princip spočívá v prosté výměně hráčů u herního zařízení po každém kole. Nejtypičtější hrou pro tuto variantu je Heroes of Might and Magic.

Hry pro více hráčů hrané online jsou takové hry, které pro své hraní potřebují připojení k internetu. Podle [Onl(2016)] se tyto hry dělí do následujících kategorií:

MUD (Multi-User Dungeon) Hry typu MUD jsou více méně multiplayerová RPG. Zásadní rozdíl od ostatních her je zde v uživatelském rozhraní. Takto označované hry totiž využívají textového rozhraní a pro komunikaci se využívá telnet nebo specializovaný klient. Do této kategorie spadá např. MUD1, který položil základy těmto hrám nebo česká hra Prahy [Mud(2016)].

MMOG (Massively-Multiplayer Online Game) Hry spadající do této kategorie se vyznačují velkým (masivním) počtem hráčů připojených k jedné hře (k jednomu virtuálnímu světu). Hráči se zde pak mohou seskupovat do menších skupinek či větší spolků a plnit úkoly, které by jako jednotlivci nezvládli. V podstatě se dá říci, že se jedná o obdoba MUDu, kde se na místo textového rozhraní využívá grafického. Tato kategorie se dá mísit s různými herními žánry a mezi ty nejčastější patří tyto:

- **MMORPG (Massively-Multiplayer Online Role Playing Game)** – Tento žánr se dá jednoduše popsat jako hra na hrdiny pro více hráčů. Ze všech podžánrů v MMOG kategorii se právě tento nevíce podobá MUDu. Zástupcem je zde např. World of Warcraft nebo Lineage.
- **MMORTS (Massively-Multiplayer Online Real-Time Strategy)** – Tento žánr popisuje mix real-timové strategie a MMOG hry. Po většinou se jedná o různé obchodní, budovatelské nebo válečné strategie, kde velice důležitou roli hraje ekonomika. Tyto hry bývají nejčastěji realizovány jako webové a patří sem například Travian nebo Space Traffic (více viz sekce 3).
- **MMOFPS/MMOTPS (Massively-Multiplayer Online First/ Third Person Shooter)** – Tyto žánry jsou mix MMOG a střílečeк z pohledu první nebo třetí osoby. Tyto hry v poslední době začínají

nabírat na popularitě a stávají se čím dál tím více oblíbenějšími. Mezi zástupce patří např. World of Tanks nebo PlanetSide.

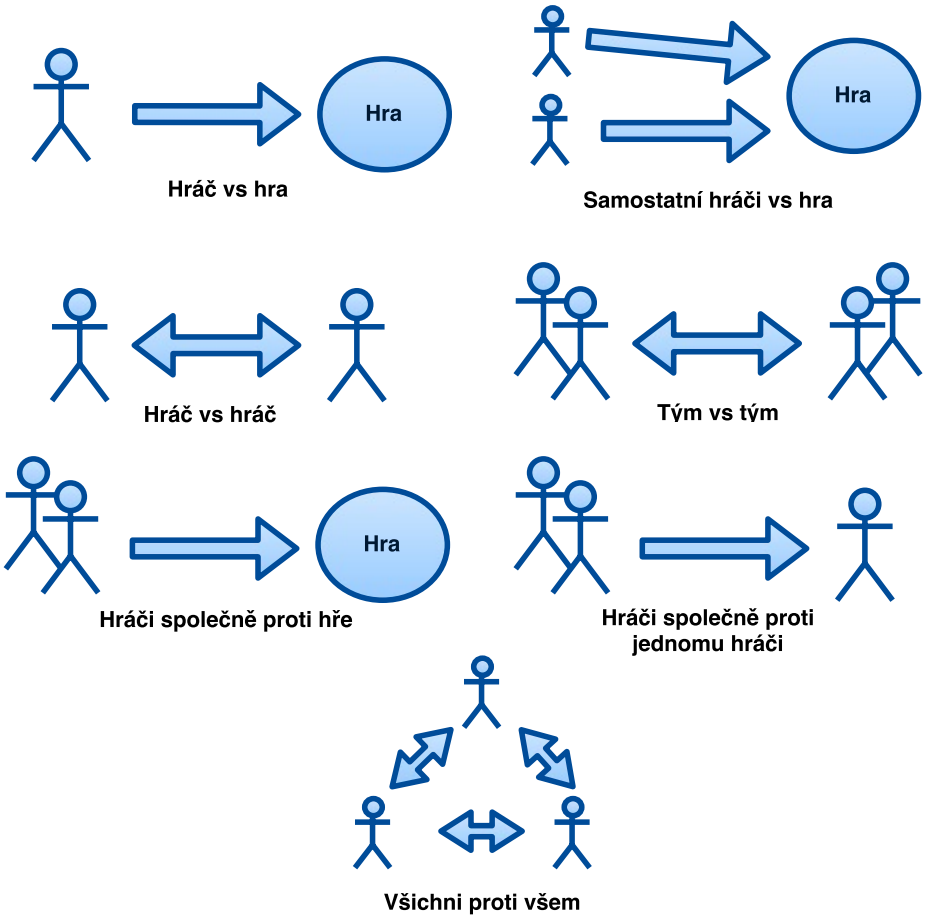
2.5 Společné herní prvky

V kapitole 2.4 jsme si rozebrali jak se hry mohou dělit a především jaká specifika přináší různé herní žánry. Některé žánry se mohou prolínat s jinými a některé jsou naopak hodně odlišné. Co když, ale budeme potřebovat určit společné prvky všech her? Dají se najít alespoň nějaké elementy, které mají všechny hry? Zcela jistě ano. Dle [Ful(2008)] můžeme rozdělit tyto prvky na dvě kategorie a to **formální elementy** a **dramatické elementy**. V následujících podkapitolách si stručně charakterizujeme tyto kategorie.

2.5.1 Formální elementy

Formální elementy představují základní strukturu každé hry a vztahy mezi nimi pak vytváří samotnou hru. Na základě [Ful(2008)] patří do této kategorie tyto prvky:

- **Hráči** – Nejdůležitějším prvkem hry jsou samotní hráči. Vzhledem k formálním elementům můžeme na hráče zacílit následujícími čtyřmi aspekty:
  - *Pozvánka ke hře* – Vůbec nejdůležitější aspekt hry. Pozváním ke hře je potřeba zacílit na hráče tak, aby byl ochoten začít hrát.
  - *Počet hráčů* – Počet hráčů může být pevný nebo proměnlivý a na základě počtu hráčů se pak mohou měnit různé další aspekty hry.
  - *Role hráčů* – Každý hráč může plnit určitou roli v dané hře a každá role pak může obsahovat určitá specifika.
  - *Interakce s hráči* – Hráči spolu mohou interagovat pomocí několika vzorů. Tyto vzory jsou znázorněny na obrázku 2.3.



Obrázek 2.3: Vzory interakce s hráči. [Ful(2008)]

- **Cíle** – Cíle mají za úkol motivovat hráče a zároveň udávají ráz hry. Ideální cíle by měly být dosažitelné a vyzývavě náročné (přílišná náročnost odrazuje). Dle [Ful(2008)] se dají rozdělit na:
  - *Dobytí* – Hlavním úkolem je dobít nebo zničit něco soupeři (např. Quake).
  - *„Honička“* – Hráči mají za úkol něco nebo někoho (jiného hráče) doběhnout (desková hra Fox & Geese).
  - *Závod* – Zde je cíl poměrně zřejmý jako příklad je možné uvést např. herní sérii Need For Speed.
  - *Záchrana/útěk* – Zde je cílem zachránit nebo vysvobodit jednotky či předměty vlastněné soupeřem (Super Mario Bros).
  - *Uspořádání* – Úkolem je vytvořit určité uspořádání v dané konfiguraci (např. Tetris).

  - *Zakázaná akce* – Hlavním úkolem u zakázané akce je provádět nepovolené kroky k porušení pravidel. V počítačových hrách se objevují jen zřídka a patří sem např. Twister.
  - *Konstrukce* – Tato kategorie si bere za úkol postavit, udržovat nebo spravovat nějaké objekty (SimCity).
  - *Průzkum* – Úkolem bývá prozkoumat dosud neobjevenou herní oblast (Colossal Cave Adventure).
  - *Řešení* – Cílem je vyřešit daný problém dříve než soupeř. Většinou se jedná o různé puzzle hry např. Connect Four.
  - *Přelstění* – Zde mají hráči získat a využít znalosti k obelstění protihráče (desková hra Trivial Pursuit).

- **Procedury** – Procedury jsou různé herní akce nebo metody, na základě kterých hráči dosahují cílů. Každá procedura více méně popisuje co, kde, jak a kdy je možno udělat. U deskových her bývají popsány v pravidlech a počítačové hry je specifikují buď v manuálech případně je upřesňují v průběhu hry. Procedury by měly popisovat následující akce:
  - *Spuštění*
  - *Postup*
  - *Ukončení*
  - *Zvláštní akce*

- **Pravidla** – Pravidla jsou jedním ze základních stavebních kamenů herního systému. Pravidla určují vše, co hráč může nebo nemůže ve hře udělat. V počítačových hrách by měla být pravidla co nejvíce intuitivní a protože bývají často složitá, hlídá je hra případně herní server. Hra by měla mít definována:
  - Pravidla pro objekty a jejich podmínky
  - Pravidla pro omezení činností hráče
  - Podmíněná pravidla

- **Zdroje** – Zdroje můžeme definovat jako objekty, které přináší hráči nějakou hodnotu při dosahování cílů. Hodnotu těchto objektů určuje především jejich užitečnost a také míra nedostatku ve hře. Dle [Ful(2008)] se zdroje dělí do těchto kategorií:

  - *Životy*
  - *Jednotky* – Např. obyvatelé města nebo jednotky armády.
  - *Zdraví*
  - *Měna*
  - *Akce* – Např. bonusové tahy nebo kouzla.
  - *Power-upy* – Zvyšují některé atributy hráče (často bývají také nazývané jako artefakty).
  - *Inventář*
  - *Speciální terén* – Terén na mapě, ze kterého se dá něco získat.
  - *Čas*

- **Konflikt** – Konflikt se rodí prostřednictvím procedur a pravidel a má za úkol zabraňovat hráči v přímém dosažení cíle. Konflikt může být tvořen:
  - *Překážkami* – Mohou být fyzické (kráter uprostřed arény) nebo psychické (chybějící indicie k řešení hádanky).
  - *Protivníky*
  - *Dilematy* – Může se jedna o různé volby nebo strategická rozhodnutí, které mají pozdější následky.

- **Hranice** – Hranice hry určují co ještě je a co již není hra. Hranice mohou být:
  - *Fyzické* – Určují například kde končí a začíná mapa.
  - *Emocionální* – Oddělují virtuální realitu od skutečného života.

- **Výsledek** – Aby si hra udržela hráčovu pozornost měl by být výsledek pokud možno vždy nejistý. Výsledky pak většinou bývají *měřitelné* (např. body) a *nevyrovnané* (např. nemusí vyhrát vždy celý tým, ale pouze jeden z týmu). Podmínky vítězství pak bývají mnohdy odlišné od hráčských cílů.

2.5.2 Dramatické elementy

Dramatické elementy působí zejména na emoční stránku hráče a přináší tak do hry lepší herní zážitek. Dle [Ful(2008)] do dramatických elementů patří:

- **Příběh** – Příběh hry by stejně jako výsledek měl být nejistý (alespoň na začátku hry) a jejím průběhem by pak hráč měl tuto nejistotu rozluštit.
- **Výzva** – Výzvy jsou úzce spjaty s hráčskými dovednostmi. Jak hráč postupuje hrou, zlepšují se i jeho dovednosti. Aby hra neztrácela dramatičnost, měla by se s postupem hry zvyšovat i obtížnost výzev.
- **Premisa** – Premisa je jedním ze základních dramatických elementů, který by měl stručně nastínit o co ve hře (případně některých částech hry) jde. Může jít například o intro, které vysvětluje, kde a kdy se příběh odehrává, o co ve hře půjde, jaké jsou cíle a také okolo jakých postav se příběh odehrává.
- **Postavy** – Postavy by měly často působit na hráčovu psychickou stránku a měly by zrcadlit jeho pocity, přání a touhy. Hlavní postavy se pak nazývají protagonisté a jejich protivníci antagonisté.

2.6 Minihry

Tato práce s zabývá implementací miniher, ale co vlastně tento pojem znamená? Minihry bychom mohli definovat jako krátkou hru, která bývá obsažena jako součást hlavní hry nebo nějakého softwaru (ve výjimečných případech i hardwaru) [Min(2016)]. Většina miniher nějakým způsobem (např. motivem) zapadá do kontextu hlavní hry a často do ní bývají zakomponovány tak, aby bylo možno pokračovat až po jejich dokončení. Po dokončení minihry většinou dostane hráč nějakou odměnu v podobě zdrojů[6]. Tyto aspekty však nemusejí být vždy nutným pravidlem.

Tento typ her si klade za cíl vnést do hlavní hry určitou změnu či rozptýlení hráče od monotónního hraní. Někteří vydavatelé pak uvolňují minihry i samostatně a zdarma za účelem propagace hlavní hry. Jako příklad hry s těmito prvky můžeme uvést např. GTA 5, kde jich je celá řada (golf, šipky, tenis, ...) nebo českou hru Polda 3 (čára, prší, ...), kde je pro další postup ve hře nutné minihry vyhrát.

[6]Co jsou to zdroje je popsáno v kapitole 2.5.1.

3 Space Traffic

V této kapitole si přiblížíme co je to Space Traffic, popíšeme si jeho historii a také zhodnotíme současný stav. Na konci kapitoly pak podrobněji zanalyzujeme současnou architekturu projektu včetně jejích technologií.

3.1 Co je to Space Traffic

Space Traffic je výuková webová MMORTS[1] hra s vesmírnými motivy. V této hře je hlavním tématem obchodování a je zde možné využívat programování k ovládání hry. Tento prvek by měl umožňovat hráči např. naprogramovat cestu vesmírné lodě skrze galaxii nebo možnost zautomatizovat některé zdlouhavé a opakované postupy pomocí uložených procedur. [Vog(2016)]

Tato hra je vyvíjena na Katedře informatiky a výpočetní techniky (KIV) při Západočeské univerzitě v Plzni (ZČU) a projekt je veden jako ryze studentský. Díky tomu si studenti mohou ověřit své znalosti a dovednosti získané při studiu a mohou nabýt cenné zkušenosti v oblasti softwarového vývoje a vedení projektů. [Vog(2016)]

Katedra od projektu očekává, že zvýší zájem studentů středních škol o studium na této katedře. Dalším záměrem je také možnost prezentace činnosti svých žáků na veřejnosti. Garantem projektu je doc. Ing. Přemysl Brada, MSc., Ph.D. a mentorem pak je Ing. Petr Vaněček, Ph.D. V současné době projekt vedou studenti Bc. Veronika Švecová a Bc. Jan Kotalík.

3.2 Historie

Podle [Ste(2012)] se historie projektu se datuje k roku 2009, kdy díky iniciativě studenta Zbyňka Neuderta tento projekt vznikl. Ten chtěl původně pokračovat ve své dřívější webové hře Zoo City, což mu, ale nebylo umožněno a proto vytvořil základní návrh Space Trafficu. Ten se pak s týmem několika studentů pokusil realizovat, ovšem z nedostatku času vytvořil především návrh hry

[1] Co to je MMORTS je vysvětleno v kapitole 2.4.2.

a také funkční prototyp postavený na technologiích HTML, PHP a JavaScript.

V roce 2010 pak projekt převzal Richard Kocman a ukázalo se, že řešení vytvořené Zbyňkem Neudertem nemá dostatečné parametry k tomu, aby na něm mohl pokračovat další tým. Jak je popsáno v [Koc(2012)], neuchopitelná architektura, špatná rozšiřitelnost a nedostatečná dokumentace přiměly studenta Kocmana opustit původní řešení a vytvořit projekt znovu na základě PHP frameworku Nette. V následujícím roce pak byl projekt nuceně přerušen z důvodu vážných rodinných důvodů vedoucího projektu.

V roce 2011 převzali projekt studenti Petr Vogl a Martin Štěpánek. Jak popisuje Martin Štěpánek ve své diplomové práci [Ste(2012)], došlo v tomto roce k restartu projektu. Předchozí práce posloužily jako prototypy (zejména pro uživatelské rozhraní) a také jako inspirace. Zároveň došlo ke změně technologie z PHP na .NET a projekt byl postaven nad MVC architekturou[2]. Na projektu se kromě obou diplomantů začali podílet i samotní studenti v rámci semestrálních prací z různých předmětů na KIV. I díky tomu pak bylo možné vytvořit funkční jádro aplikace a kvalitní game design včetně dokumentace.

V roce 2012 byl projekt převzat Janem Dyrczykem a Pavlem Bořík em. Jak uvádí autor [Bor(2013)], díky dobrému základu hry bylo možno začít zajišťovat kvalitu softwaru a pokračovat ve vývoji hry dle game designu. Dále byla také vytvořena projektová wiki a do projektu byl integrován jeden z hlavních prvků hry a to možnost ovládání aplikace pomocí programovacího jazyka Starship Basic[3]. Zároveň se pokračovalo v rozvoji aplikace v podobě semestrálních, bakalářských a diplomových prací studentů KIV.

3.3 Současnost

V současné době vedou projekt studenti Veronika Švecová a Jan Kotalík, kteří projekt přebrali po předchozích vedoucích. Z důvodu nepříliš dobré kontroly kvality softwaru předchozími vedoucími byly nuceni projekt vrátit o několik verzí zpět do provozuschopného a rozšiřitelného stavu. Dle slov vedoucích byl tento krok nevyhnutelný, protože se aplikace chovala jako „domeček z karet“.

[2]Co to je MVC je popsáno v kapitole 3.4.

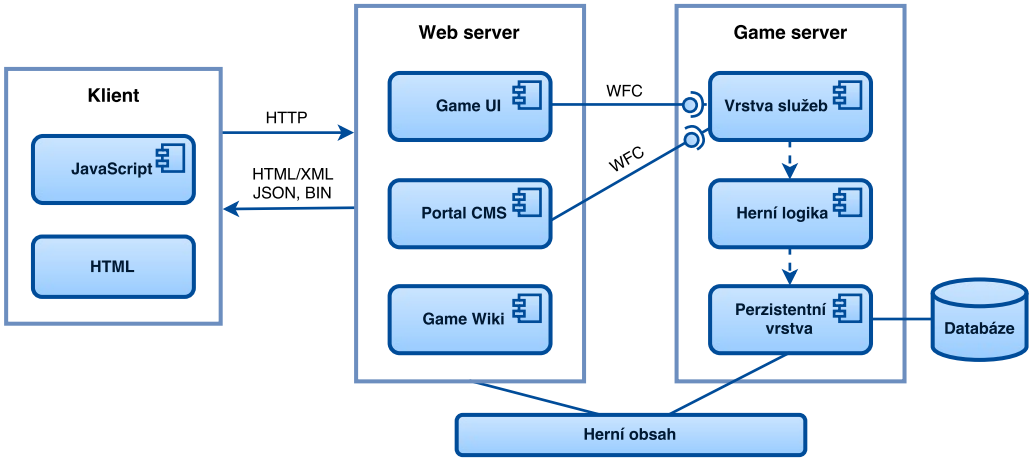
[3]Jazyk navržený studenty výhradně pro tuto aplikaci.

V loňském roce bylo do jádra aplikace implementováno obchodování, plánování a také testování nejdůležitějších součástí projektu. V rámci bakalářské práce pak Pavel Fakan vytvořil generátor hvězdných systémů. V aktuálním roce probíhá vývoj přihlašování a registrace hráčů. V rámci diplomových prací pak Jan Kotalík vytváří nové uživatelské rozhraní a Veronika Švecová navrhuje pravidla a postupy pro řízení tohoto projektu. Na základě této práce pak bude možné do hry přidávat minihry vytvořené dalšími studenty. Díky loňským a letošním projektům se tak aplikace znovu blíží k první hratelné verzi.

3.4 Architektura aplikace

Jak již bylo zmíněno v kapitole 3.2 od roku 2011 je Space Traffic vyvíjen pod platformou **Microsoft .NET Framework verze 4**. Tato platforma je jednou z nejrozšířenější aplikačních vývojových platforem pro operační systém Microsoft Windows. Dle [Vir(2002)] se skládá ze dvou základní částí a to **běhového systému** (Common Language Runtime) a **základních knihovních tříd** (Basic Class Library).

Architektura aplikace je postavena na základě architektonického vzoru **MVC**. Tato architektura od sebe odděluje datovou vrstvu (Model), uživatelské rozhraní (View) a aplikační logiku (Controller) tak, aby změny v jakékoliv z vrstev měly co nejmenší vliv na ostatní vrstvy. Diagram architektury aplikace je znázorněn na obrázku 3.1. Z obrázku můžeme vidět, že hra se skládá z pěti modulů. Mezi tyto moduly patří **klient**, **web server**, **game server**, **databáze** a **herní obsah**. Tyto moduly si stručně popíšeme v následujících podkapitolách.



Obrázek 3.1: Diagram architektury aplikace. [Sta(2015)]

3.4.1 Klient

Klientská část aplikace je tvořena v **HTML5** a **JavaScriptu**. Pro formátování je využívána knihovna *dotless*[4] a pro snazší práci s JavaScriptem je používáno sady knihoven *jQuery*[5] (konkrétně *jQuery*, *jQueryUI* a *jQuerySVG*). Pro komunikaci mezi klientem a webovým serverem je používán protokol **HTTP**[6] s využitím formátů **HTML**, **XML**, **JSON** a dalších formátů pro přenos multimediálního obsahu. Pro asynchronní komunikaci a aktualizaci stránek se pak využívá technologie *AJAX*[7] a nově také ajaxový systém zpráv vytvořený Janem Kotalíkem. Klientská část je primárně optimalizována pro moderní webové prohlížeče Firefox, Google Chrome a Opera. [Stc(2015)]

3.4.2 Web server

Webový server obsahuje tři webové aplikace, které jsou na sobě vzájemně nezávislé. Patří sem hlavně uživatelské rozhraní (**Game UI**), které je postavené na frameworku *ASP.NET MVC 3*[8] a slouží především pro komunikaci mezi herním serverem a klientem. Dále je zde herní portál (**Portal CMS**), jehož

[4]dotless je implementace preprocesoru LESS v C#.

[5]jQuery je populární open source knihovna usnadňující práci s JavaScriptem.

[6]HTTP neboli Hypertext Transfer Protokol je základní aplikační protokol pro výměnu hypertextových dokumentů na webu.

[7]AJAX neboli Asynchronní JavaScript a XML je webová technologie pro výměnu dat mezi klientem a serverem bez nutnosti znovunačítání stránky.

[8]ASP.NET MVC je součást platformy ASP.NET implementující MVC vzor a umožňuje tak jednoduše vytvářet aplikace postavené na této architektuře.

primárním úkolem je zajišťovat webovou prezentaci hry a celého projektu. Poslední součást web serveru je herní wiki (**Game Wiki**), která slouží jako nápověda ke hře. [Stw(2015)]

3.4.3 Game server

Herní server tvoří jádro celého systému hry a je implementován v jazyce C#. Game server se skládá ze tří částí. První částí je **vrstva služeb**. Tato vrstva obsahuje služby poskytované herním serverem, sloužící především ke komunikaci mezi game serverem a uživatelským rozhraním. Tato komunikace je realizována pomocí technologie *WCF*[9]. Prostřednictvím této vrstvy je pak možné komunikovat s modulem pro **herní logiku**. Herní logika realizuje simulaci celého herního světa. Poslední vrstvou je **perzistentní vrstva**, která zajišťuje perzistenci herních objektů a přístup k databázi. Tato vrstva je implementována pomocí *ADO.NET Entity Frameworku*[10] na základě návrhového vzoru DAO[11]. [Stg(2015)]

3.4.4 Databáze

Databáze je reprezentována systémem řízení a báze dat (SŘBD) *Microsoft SQL Server*. Relační model databáze můžeme vidět buď v příloze A nebo v [Vlč(2015)].

3.4.5 Herní obsah

Herní obsah reprezentuje úložiště statický datových souborů, které obsahují konfiguraci herního světa (definice map hvězdných systémů), druhy zboží a jejich specifika, druhy lodí a jejich vlastnosti a jiné další digitální zdroje. Tato data mohou být předána přímo klientovi a je možné u nich vyžívat cachování. [Sts(2015)]

[9] WCF bude detailněji popsáno v kapitole 4.4.

[10] ADO.NET Entity Framework je framework pro objektově-relační mapování.

[11] Návrhový vzor DAO neboli Data Access Object je ve své podstatě objekt, který poskytuje abstraktní rozhraní pro přístup k databázovým entitám.

4 SOA

V této kapitole si nejprve vysvětlíme co je to servisně orientovaná architektura neboli SOA a proč není tento pojem jednoznačně definován. Následně si pak vysvětlíme základní principy SOA. Dále pak v krátkosti rozebereme co jsou to webové služby a protože jsou webové služby ve Space Trafficu realizovány pomocí WCF, uvedeme si o jakou technologii se jedná a jaké hlavní vlastnosti má.

4.1 Co je to SOA

SOA neboli servisně orientovaná architektura představuje otevřenou, rozšiřitelnou a komponovatelnou architekturu orientovanou na služby. Tyto služby jsou autonomní a různými způsoby spolu vzájemně komunikují. Základem SOA při řešení určitého problému je rozdělení řešení na soubor menších vzájemně spolupracujících součástí a každá z těchto částí pak řeší určitou část problému. [Erl(2009)]

Definic SOA je dnes velké množství a není možné uvádět pouze jednu přesnou definici, protože se jednotlivé definice mohou poměrně hodně lišit. Z toho důvodu pak můžeme říci, že servisně orientovaná architektura je spíš koncept návrhu systému než technologie či dokonce standard. Jak velké mohou tyto odlišnosti být dokazují následující dvě definice.

První, poměrně jednoduchá, dle [Bhu(2011)] říká, že SOA je ve své podstatě sbírka služeb, které spolu vzájemně komunikují. Komunikace zahrnuje buď posílání jednoduchý dat nebo zahrnuje dvě či více služeb, které koordinují nějakou aktivitu.

Oproti tomu mnohem komplexnější definici uvádí [Ser(2016)]. Ta říká, že se jedná architektonický přístup, který podporuje integraci business úloh do souboru propojených a opakovaně volaných služeb. Zároveň podporuje inovaci díky rychlé, snadné a levné adaptaci informačních systémů na změny požadované trhem. Dále pak napomáhá zvyšovat flexibilitu byznys procesů a posiluje IT infrastrukturu a znovupoužitelnost stávajících investic do IT díky propojování informačních zdrojů a různorodých aplikací.

4.2 Principy SOA

Protože neexistují žádné standardy, které by charakterizovaly složení servisně orientované architektury, uvádějí různí autoři lehce odlišné principy. Dle [Erl(2009)] jsou principy SOA následující:

- **Standardizovaný kontrakt služeb** – Služby by měly zachovávat dohodu o komunikaci a proto by měl být kontrakt služby jasně definovaný.
- **Volná vazba** – Tento princip říká do jaké míry je služba závislá na znalosti jiné komponenty (případně systému). Vztahy mezi jednotlivými komponentami by tak měly být co nejmenší resp. volnost vazby by měla být co největší.
- **Abstrakce** – Principem abstrakce je pokud možno co nejvíce odstínit kontrakt služby od implementačních detailů. Díky abstrakci je umožněna lepší granularita sytému a zároveň je usnadněna správa a úprava systému.
- **Znovupoužitelnost** – Služby by měly být navrhovány tak, aby bylo možné jejich využití i v jiném systému nebo projektu. Tím se zajistí maximální využití dané služby.
- **Nezávislost** – Aby byla služba nezávislá (autonomní), měla by obsahovat vnitřní logiku nezávislou na správě zdrojů. Služba by měla být nezávislá na stavech a změnách ostatních entit a také na komunikačních protokolech ostatních systémů (aplikací). Díky nezávislosti je podpořena znovupoužitelnost.
- **Bezstavovost** – Pokud chceme zajistit znovupoužitelnost je potřeba také zajistit bezstavovost. Bezstavovost se zajistí minimalizací uchovávaných informací o stavu komponenty. Občas se může zdát tento princip trochu zavádějící, protože v průběhu vykonávání požadavku prochází služba různými stavy. Po odeslání informace by se, ale měla nacházet původním („neutrálním“) stavu.
- **Identifikovatelnost (Viditelnost)** – Služba by měla být popsána nějakými metadaty, pomocí kterých pak může být identifikována a používána. Pro popis služby v SOA a následnou identifikovatelnost se používá jazyk WSDL[1].

[1] Co je to WSDL je popsáno v kapitole 4.3.

- **Skládání** – Pokud jednotlivé služby řeší malé a snadno řešitelné problémy, je možné tyto služby poskládat a uspořádat tak, aby bylo možné vyřešit rozsáhlý problém. Na základě tohoto principu může služba využít volání jiných služeb (komponent) a řešit tak libovolný problém.

4.3 Webové služby

Webové služby, anglicky označované jako Web services jsou jednou z nejčastějších realizací služeb v servisně orientované architektuře. Webová služba je aplikační (business) komponenta, která klientům poskytuje opětovně použitelnou užitečnou funkčnost. Na tuto komponentu je možné pohlížet jako na komponentu s globální dostupností. Touto dostupností je myšlena možnost přístupu z kteréhokoliv místa na světě prostřednictvím internetu a přístupových práv. [Sha(2010)]

Webové služby využívají několika technologických standardů. Pro popis je používán jazyk WSDL (Web Service Description Language). Tento jazyk využívá pro popis přenositelný formát XML. Pro odesílání požadavků a příjímání odpovědí od webových služeb je pak využíváno protokolu SOAP (Simple Object Access Protocol). Ten slouží k výměně zpráv v XML formátu a je postavený nad HTTP (již zmíněný v kapitole 3.4.1). Na základě těchto technologií jsou webové služby platformně nezávislé (zajišťují tzv. interoperabilitu). [Sha(2010)]

4.4 Windows Communication Foundation

Microsoft Windows Communication Foundation (WCF) je součást Microsoft .NET Frameworku, která představuje API a běhové prostředí pro tvorbu servisně orientovaných aplikací. Účelem WCF je umožnit vytváření aplikací, které jsou co možná nejvíce nezávislé na mechanismu pro spojování služeb a aplikací. Tato knihovna také umožňuje udržovat zpětnou kompatibilitu s mnoha staršími technologiemi pro webové služby. [Sha(2010)]

Windows Communication Foundation mají jistě spoustu výhod a proto si vyjmenujeme jen ty nejdůležitější. Dle [Wha(2016)] jsou hlavní vlastnosti WCF následující:

- **Servisní orientace** – Jak již bylo uvedeno výše, WCF umožňuje vytvářet servisně orientované aplikace. To je umožněno především díky používání WS standardů. Tyto standardy definují volné vazby mezi službami a tím je zajištěno, že každý klient, který splňuje nezbytný kontrakt se může ke službě připojit a to bez ohledu na platformu.

- **Interoperabilita** – Interoperabilita je schopnost komunikace dvou různých systémů. Stejně jako v předchozím bodě je tato vlastnost umožněna na základě používání WS standardů.

- **Různorodost vzorů pro zasílání zpráv** – Zprávy je možné vyměňovat pomocí mnoha vzorů. Nejčastěji používaným vzorem je request/response (žádost/odpověď), kdy klient vyšle žádost a server ji zpracuje a odešle odpověď.

- **Metadata služeb** – WCF také umožňují publikovat metadata o službách pomocí různých standardů např. WSDL či XML schéma. Ta pak bývají publikována buď pomocí HTTP nebo pomocí Service Metadata Exchange[2].

- **Datové kontrakty** – Protože je tato knihovna postavena na .NET Frameworku je možné poměrně jednoduše definovat datový kontrakt služby. Nejjednodušším řešením je vytvoření třídy obsahující property (vlastnosti), na základě kterých je pak specifikován datový kontrakt.

- **Bezpečnost** – Bezpečnost je umožněna prostřednictvím šifrování zpráv dle standardů SSL[3] nebo WS-SecureConversation[4].

- **Různorodost transportních protokolů a kódování** – WCF umožňuje přenos na různých transportních protokolech a s různým kódováním. Nejpoužívanější je pak posílání SOAP zpráv přes HTTP protokol. Dále je možné také použít TCP, pojmenované roury nebo i vlastní transportní protokol.

- **Podpora spolehlivého přenosu** – WFC podporuje spolehlivou výměnu zpráv s vyžitím front.

- **Odolnost zpráv vůči ztrátě** – Odolnost zpráv vůči ztrátě znamená, že je možné použít zprávy, které přežily výpadek spojení. V případě

[2]Service Metadata Exchange je protokol pro výměnu metadat o webových službách.

[3]Secure Sockets Layer je vrstva mezi transportní a aplikační vrstvou, která poskytuje šifrovanou komunikaci.

[4]WS-SecureConversation je specifikace pro webové služby, který umožňuje bezpečné sdílení obsahu.

výpadku spojení je část zprávy uložena a při obnovení spojení je možné navázat na část, kde došlo k přerušení.

- **Transakce** – V knihovně je umožněna podpora transakcí na základě transakčních modelů podporovaných .NET frameworkem.
- **AJAX a REST** – WCF také poskytuje podporu pro čistá XML (nezabalená v SOAP obálce) a také non-XML formáty jako JSON[5]. Zároveň je možné nakonfigurovat WCF pro zpracovávání REST požadavků[6].
- **Rozšiřitelnost** – WCF je možné dle potřeby rozšiřovat o další komponenty.

[5]JSON neboli JavaScript Object Notation je datový formát nezávislý na platformě používaný především v JavaScriptu.

[6]REST neboli Representational State Transfer je architektura rozhraní použitelného pro jednotný a snadný přístup ke zdrojům (např. datům).

5 Návrh rozhraní pro minihry

V této kapitole si nejprve na základě game designu specifikujeme jaké charakteristické vlastnosti by měly mít minihry. Následně si nastíníme jaké možnosti spouštění je možné využít a navrhneme vhodné řešení. Dále si ke zvolenému spouštění navrhneme typy akcí a podmínek na jejichž základě bude možné hry spouštět. Zároveň si specifikujeme i možné druhy odměn.

V další části této kapitoly si rozebereme proč je dobré mít popsané hry a co by takový popis měl obsahovat. Následně si rozebereme jednotlivé možnosti integrace v závislosti na architektuře projektu. Ke konci této kapitoly vysvětlíme zvolený způsob integrace a jak bude tato možnost principiálně fungovat.

5.1 Specifikace oproti game designu

V této části si specifikujeme co z game designu by bylo dobré použít pro minihry, co nikoliv a jaké změny by bylo dobré učinit. V [Stm(2015)] je uvedeno, že minihry by měly:

- Poskytovat odlišný herní zážitek od hlavní hry.
- Odměňovat hráče nad rámec běžné hry.
- Vnést do hry odlišnou hratelnost.

Tyto body není potřeba nijak měnit. V kapitole 2.6 uvádíme, že prvek miniher má za cíl vnést do hry určité rozptýlení od monotónního hraní, což tyto věty potvrzují. Zároveň by tato doporučení měla být rozšířena o následující:

- **Minihra by měla hráče hlavně pobavit, nikoli znechutit.** – Tento bod je vůbec nejtěžší a ne vždy je možné každé hráče nadchnout a potěšit.
- **Dosažení cíle by mělo trvat úměrně dlouho vzhledem k povaze hry.** – V případě hraní např. šachů se dá vzhledem k povaze

hry očekávat, že dosažení cíle bude trvat delší dobu. Pokud ovšem hrajeme nějaký kvíz, neměl by obsahovat příliš otázek, abychom hráče „neotrávili“.

- **Odměna by neměla přinášet příliš velkou výhodu.** – Pokud bude odměna přinášet velkou výhodu (např. hráč dostane nejmodernější loď), budou ostatní hráči značně znevýhodněni a sám hráč pak kvůli tomu může ztratit zájem o hraní hlavní hry.
- **Obtížnost by se měla vztahovat k povaze a výši odměny.** – Za náročnější hru např. vyřešení nějaké složitějšího logického problému by měla být hodnotnější odměna než za vyhrání např. puzzlí.
- **Hráč by neměl být za nesplnění hry penalizován.** – Penalizace za nedosažení cílů nebo předčasné ukončení by mohla hráče odradit od hraní jak miniher, tak hlavní hry.
- **Ovládání by mělo být jednoduché, intuitivní a komfortní.** – Ovládání by nemělo být nikterak složité a nekomfortní např. hraní hada ovládaného pomocí myši moc zábavné nebude.
- **Pravidla hry by měla být co nejvíce intuitivní a vždy vysvětlena.** – V případě známých her např. šachů není potřeba pravidla hry zmiňovat. U her s rozsáhlejšími nebo upravenými pravidly (např. různé karetní variace) je vhodné tato pravidla na začátku hry vždy zmínit.

V game designu se dále uvádí, že minihry by měly mít edukativní charakter. Toto doporučení má vzhledem k jednomu z cílů hry (výukový charakter) zcela logické opodstatnění. Toto doporučení je potřeba brát mírně s rezervou a ne vždy a za každou cenu do hry vnést výukové prvky. V tomto případě by se pak mohlo stávat, že edukativní prvek bude vytržen z kontextu hry nebo hra po čase začne ztrácet na zábavnosti a odradí hráče od hraní celé hry.

V dokumentu jsou dále uváděny druhy miniher (klasické, optimalizační a cutscény). Vzhledem k zadání práce, kdy má být možno do hry začlenit co možná největší množství her různých druhů a žánrů, je při návrhu počítáno jak s těmito druhy, tak i s dalšími možnými druhy tak, aby byla různorodost her co nejširší.

Game design dále uvádí druhy odměn za minihry a také formu integrace. Oba tyto body jsou detailněji popsány v dalších kapitolách.

5.2 Spouštění miniher

Aby bylo možné minihry vůbec hrát, je nejprve nutné navrhnout jakým způsobem se budou vůbec spouštět. Při návrhu byla brána v úvahu spouštění na základně **kontraktů**, na základně **akcí a podmínek** a **volně z menu hry**. Tyto možnosti budou rozebrány v následujících podkapitolách.

5.2.1 Spouštění dle kontraktů

V původním návrhu hry je počítáno se spouštění na základě kontraktů. Kontrakty mají představovat různé časově omezené mise. Tyto mise lze získávat na planetách a za jejich plnění či neplnění hráč dostane buď odměnu nebo pokutu. Protože v současné verzi hry nejsou kontrakty realizovány, bylo by velice obtížné na základě nich navrhovat jak hru spouštět. Z tohoto důvodu byl tento způsob při navrhování zavrhnut.

5.2.2 Spouštění dle akcí a podmínek

Další možností jak by bylo možné hru spouštět je na základě akcí a podmínek. Vzhledem k tomu, že si game server uchovává informace o aktuálním stavu hráčů a herního světa by bylo možné tuto variantu poměrně jednoduše realizovat. Pokud by tedy hráč vykonal nějakou akci, ke které by byly přiřazeny příslušné minihry a splňoval by některou z podmínek pro spuštění dané hry, bylo by mu nabídnuto jestli si chce příslušnou hru zahrát nebo ne.

Toto spouštění si můžeme představit následujícím způsobem. Hráč má zájem o koupi lodi. V uživatelském rozhraní tedy vyvolá akci *Koupení lodi*. K této akci bude na serveru přiřazena minihra Poker s podmínkou, kde hráč má *aktuální množství peněz menší na 10000 kreditů*. Game server vyhodnotí, že hráč splňuje podmínky pro hraní hry a nabídne mu jestli chce danou hru hrát.

5.2.3 Spouštění volně z menu

Poslední možností spouštění je volné spouštění z menu hry. Tato možnost by mohla být realizována např. nějakým seznamem všech her, kde by si hráč

zvolil příslušnou minihru a následně ji spustil. Výhodou této realizace by byla eventualita rozšíření spouštění her obdobným způsobem z CMS portálu. V případě externích aplikací by pak mohla být výhodná možnost spuštění minihry bez nutnost spouštění hlavní hry. Značnou nevýhodou by pak byla nutnost odebrání odměn za vyhranou hru. Kdyby odměny zůstaly mohlo by docházet k neustálému obohacování hráče.

5.2.4 Zvolené řešení

První možnost byla při návrhu vzhledem aktuální fázi projektu zavrhnuta. Se zbylými dvěma možnostmi je pak v návrhu počítáno. Spouštění na základě **akcí a podmínek** je však prioritní, protože by mohlo do hry přinášet daleko větší požitek a také moment překvapení.

Se spouštění **volně z menu** je počítáno pouze v případě, že by po dosažení cíle hry nebyl hráč odměněn. Tato varianta je uvažována především kvůli hře pro více hráčů, protože u druhé možnosti by mohlo docházet k poměrně velkým časovým prodlevám u vyvolání stejných akcí a podmínek pro stejné hráče.

V kapitole 2.6 je uvedeno, že některé hry dovolují hráči pokračovat až po splnění dané minihry. Tato možnost byla vzhledem k architektuře hry zavrhnuta. Prioritní je dokončení aktuálně prováděné akce a následně je pak možné zahrát si minihru.

5.3 Akce a podmínky spouštění

Pro zvolený typ spouštění miniher, tedy **akce a podmínky** je potřeba specifikovat jak typy akcí, tak podmínky. Protože projekt v průběhu návrhu obsahuje pouze několik málo akcí a není uvedeno jaké další akce jsou plánovány v dalších fázích vývoje, uvedeme si pouze již realizované akce. Mezi ty patří:

- **Nákup zboží**
- **Prodej zboží**
- **Nákup lodi**

- **Vzlet lodi**
- **Přistání lodi**
- **Nakládání nákladu**
- **Vykládání nákladu**
- **Průlet červí dírou**
- **Oprava lodi**
- **Tankování paliva**

Jednotlivé položky v tomto případě není potřeba nijak komentovat, jejich význam je patrný z názvu. K akcím je dále nutné specifikovat také podmínky spouštění. V případě podmínek je možné specifikovat alespoň jejich typy. Tyto typy jsou navržené na základě některých částí game designu a také na základě aktuálního stavu projektu. Patří sem:

- **Žádná podmínka** – Hra se spustí vždy na danou akci bez ohledu na příslušnou podmínku.
- **Level hráče** – Podmínka může být realizována na aktuální level, případně menší nebo větší než aktuální.
- **Pořadí hráče v žebříčku** – Tento typ podmínky se vztahuje k aktuálnímu pořadí hráče v žebříčku.
- **Kredity (Měna)** – Spuštění hry dle počtu kreditů hráče.
- **Achievementy** – Dle tohoto typu podmínky se hra spustí pokud hráč objevil, případně neobjevil určitý achievement či jejich skupinu.
- **Aktuální planeta hráče** – Hra se spustí pokud se hráč nachází na příslušné planetě.
- **Aktuální hvězdný systém hráče** – Obdobně jako u planet se u tohoto typu spustí minihra pokud se hráč nachází v příslušné hvězdném systému.
- **Obchodník** – Podmínka pro nakupování u určitého obchodníka.
- **Továrna** – Dle tohoto typu dojde ke spuštění hry pokud hráč provádí určitou operaci v příslušném typu továrny.

- **Zboží** – Spuštění na základě nákupu či prodeje určitého druhu zboží.
- **Loď** – Tento typ podmínky může být realizován buď nákupem nebo ovládáním příslušného typu lodi.

5.4 Odměňování

Dle game designu se v původním návrh miniher počítá pouze s odměnami jako jsou peníze, materiál či achievementy. Na základě toho byly při návrhu uvažovány další odměny, které by bylo možné v další verzích hry použít[1]. Při návrhu byly uvažovány následující odměny:

- **Zboží**
- **Kredity (Měna)**
- **Achievementy**
- **Zkušenosti**
- **Pořadí v hráčském žebříčku** – Touto odměnou je myšleno připsání bodů do hráčského žebříčku.
- **Loď**
- **Část lodi**
- **Pozemek**

Všechny odměny snad kromě **Pořadí v hráčském žebříčku** není potřeba nijak komentovat, jejich význam je patrný z názvu. Pro každý typ (kromě **Achievementů** a **Lodí**[2]) by se mělo specifikovat v jaké výši bude daná odměna. Zároveň by se také pro každý typ (kromě **Kreditů**, **Zkušeností** a **Pořadí v hráčském žebříčku**) měl specifikovat konkrétní druh dané odměny.

[1]Tím, že by bylo možné tyto odměny použít v dalších verzích je myšleno, že se s danými prvky počítá dle game designu v budoucích fázích implementace projektu.

[2]Nepředpokládá se, že by hráč dostal jako odměnu více jak jednu loď.

5.5 Popis miniher

Vzhledem k architektuře projektu (kapitola 3.4) by bylo vhodné navrhnout strukturu, která by popisovala minihry. Díky popisu bude možné publikovat informace o jednotlivých hrách v herní wiki, případně CMS portálu a zároveň by tento krok mohl dopomoci ke snazší integraci her do aplikace. Pro popis miniher pak byly, i na základě předchozích dvou kapitol navrženy následující charakteristiky:

- **Název hry** – Každá hra by měla mít nějaký název a v ideálním případě by měl být tento název unikátní v rámci všech her.
- **Počet hráčů** – Počet hráčů je jedním ze základních prvků hry, jak je uvedeno v kapitole 2.5.1.
- **Popis hry**[3] – Popis by měl nastínit o čem daná hra je, jaká jsou pravidla hry a také co je cílem. V případě potřeby mohou být do popisu uvedeny ještě další vlastnosti minihry.
- **Popis ovládání** – Důležitým prvkem při popisu hry je také ovládání. U každé minihry je proto nutné uvést jak se ovládá a případně jaké klávesové zkratky či jiná specifika se ve hře dají použít.
- **Seznam akcí** – Na základě kapitoly 5.3 by měl popis obsahovat i seznam jednotlivých akcí na jejichž základě pak bude možné hru spouštět.
- **Seznam podmínek** – Stejně jako v předchozím případě je pro správné spouštění nutné uvést i podmínky spouštění.
- **Typ odměny** – Protože by každá hry měla obsahovat odměnu je nezbytné uvést na základě kapitoly 5.4 i typ odměny jakou hráč dostane po dosažení cíle.
- **Velikost odměny** – Zároveň s typem odměny musí být specifikována velikost odměny, aby hráč věděl jak hodnotnou cenu může získat.

Tyto základní prvky pro popis každé hry by se dalo ještě rozšířit v závislosti na možnostech integrace. V případě, že budeme uvažovat možnost

[3]Pokud se v následujících částech práce bude hovořit o popisu hry, je tím myšlen komplexní popis celé hry. Tedy všechny vlastnosti a nikoli pouze tato charakteristika.

komunikace prostřednictvím webové služby a tedy realizaci klienta jako externí aplikaci, přibyly by do popisu atributy s **URL adresou pro stažení klienta** a také atribut indikace jestli hra **využívá externího klienta** nebo nevyužívá.

5.6 Možnosti integrace

Jednou ze základních otázek při návrhu rozhraní pro minihry je jakými způsoby je vůbec možné hry integrovat do stávající aplikace vzhledem k architektuře projektu. I když je možné integraci provést spousty způsoby, byly při návrhu uvažovány 3 možnosti.

První z možností je integrace **pouze do web serveru** s využitím pluginů či klientského javascriptu. Druhou možností je integrace do **game serveru** s částečným využitím první varianty. Poslední možností je pak realizace externího klienta a integrace do **herního serveru s využitím komunikace přes webové služby**.

5.6.1 Integrace do web serveru

První možností je integrace logiky miniher pouze do web serveru a zobrazování by pak probíhalo buď pomocí nějakých pluginů (např. Flashe, Silverlightu či Java appletů) nebo pomocí JavaScriptu. Komunikace s webovým serverem by pak mohla probíhat buď pomocí HTTP requestů nebo pomocí ajaxových volání. Tato možnost by se dala ještě modifikovat přesunutím části logiky přímo do klientské části.

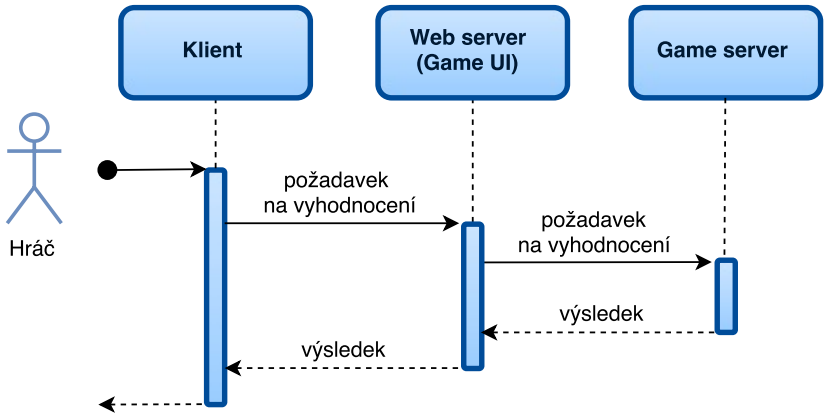
Výhodou tohoto řešení je značné odlehčení herního serveru. Tím, že by herní logiku zajišťoval pouze web server by došlo jednak ke **snížení zatížení game serveru**, tak i ke **zmenšení komunikace** mezi těmito dvěma částmi aplikace. V případě modifikace by došlo i ke snížení zatížení webového serveru.

Značnou nevýhodou pak může být především fakt, že by herní server neměl informaci o aktuálním stavu hráče resp. několika hráčů při multiplayeru. Tím bychom se připravili o možnosti **spouštění her na základě různých akcí a podmínek** (např. hráč nakupuje a nemá peníze) a také o **jednoduchou možnost odměnění hráče**. Zároveň bychom se připravili o možnost

komunikace přes webové služby game serveru pro případná další rozšíření a také by nebylo možné **použít informace o hrách například na CMS porálu**.

5.6.2 Integrace do game serveru a web serveru

Druhou možností, která byla brána v úvahu je integrace do game serveru s využitím první varianty. Tato možnost spočívá v zakomponování herní logiky miniher do herního serveru a využití webového serveru jako zprostředkovatele komunikace s klientem. Klient by pak pro zobrazování a komunikaci využíval stejné technologie jako v předchozí možnosti. Komunikace by vypadala jako na obrázku 5.1.



Obrázek 5.1: Diagram znázornění komunikace pro druhou možnost integrace.

Výhody tohoto řešení jsou přesným opakem nevýhod předchozího způsobu integrace. Tedy bylo by umožněno **spouštět minihry na různé akce a podmínky**, **jednoduché odměnění hráče**, možnost **publikovat informace o minihrách** na ostatních modulech web serveru a také možnost **komunikace na základě webové služby** pro případná rozšíření. Zároveň by také bylo možné **rozložit zatížení** mezi oběma hlavními moduly.

Do nevýhod můžeme zařadit **zvýšení komunikace** v rámci modulů aplikace a také **nutnost upravovat všechny hlavní části architektury při integraci nové hry**.

5.6.3 Integrace do game serveru

Posledním zvažovaným způsobem jak minihry integrovat je možnost zakomponování herní logiky do game serveru s využitím webových služeb pro komunikaci s klientem. Klientská část by pak mohla být realizována jako desktopová či mobilní aplikace, které by mohly být realizovány jako tenký klient. Díky komunikaci přes webové služby by pak technologie klienta mohla být prakticky libovolná.

Výhodou zde může být možnost realizovat klienta jako **mobilní aplikaci**, což je v současné době poměrně populární záležitost. Další výhodou je pak možnost implementace klienta v **libovolné technologii** podporující komunikaci skrze webové služby. Další výhody mohou plynout z předchozího řešení.

Hlavní nevýhodou tohoto řešení je především **problematické spouštění** z hlavní hry a také použití **jiných než nativních webových technologií** pro realizaci klienta. To může vést k problémům s přenositelností klientů. Dále pak může nastat problém s **velikostí přenášených dat** u mobilních aplikací.

5.7 Zvolené řešení integrace

Na základě předchozí kapitoly, kde jsou rozebrány možnosti integrace vzhledem k architektuře aplikace, byla jako vhodný způsob integrace vybrána kombinace druhé a třetí možnosti. Díky tomu bude možné do aplikace integrovat hry postavené na libovolné technologii, komunikující skrze webové služby a také hry ve webových technologiích[4] komunikujících přes web server s game serverem.

5.7.1 Odůvodnění zvoleného řešení

Toto řešení bylo vybráno hned z několika důvodů. Prvním důvodem je, že je možné **rozdělit aplikační logiku hry prakticky na jakékoliv místo**. To znamená, že může být logika jak na game serveru či web serveru, tak i na

[4]Webovými technologiemi je myšleno především použití ASP.NET, HTML5 a JavaScript, případně využití pluginů jako Flash, Silverlight či Java applety.

klientovi nebo v externí aplikaci. Na herním serveru by se určovalo pouze o jakou hru se jedná a v jakém stavu se právě nachází. Zároveň je možno herní logiku rozdělit či zduplikovat.

Díky tomu je možné **rozložit zatížení serveru** a zároveň je zde umožněna poměrně velká volnost pro návrh her. Tento způsob řešení také dává **prostor k realizaci velkého množství různých her**, které by mohly být jinak těžko realizovatelné (eventuálně by mohl vzniknout problém s některou s technologií).

Pro příklad můžeme uvést real-time střílečku z pohledu první osoby realizovanou jako síťovou hru pro více hráčů. V tomto případě by zatížení serveru bylo pravděpodobně neúnosné a komunikace mezi hráči by mohla váznou na odezvě serveru. Vzhledem k popisované možnosti by si tak hráči resp. klienti mohli mezi sebou zvolit jednoho, který bude realizovat server pro ostatní. Herní logiku by pak zajišťoval tento server a v případě potřeby by komunikoval s herním serverem. Ostatní klienti by pak komunikovali výhradně s ním namísto s game serverem.

Dalším důvodem k volbě tohoto řešení je pak možnost **vytvoření minihry jako mobilní aplikace**. Tato varianta je zvažována i v game designu a protože stále dochází k nárůstu popularity mobilních aplikací, byla by poměrně velká škoda ztratit tuto možnost nevhodným řešením integrace. Posledním důvodem je čistě akademický zájem. Vzhledem k tomu, že je možné vytvářet hry v různých technologiích a jedinou nutnou podmínkou je hru popsat, je možné **zadávat hry jako semestrální práce z různých předmětů** vyučovaných na KIV.

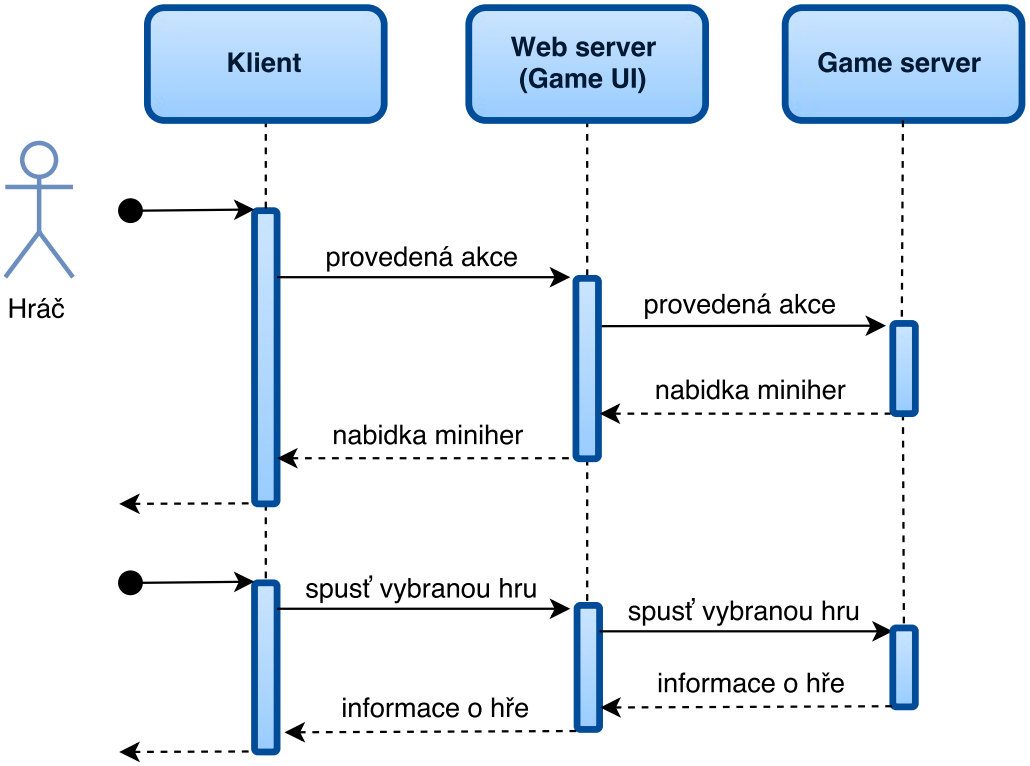
5.7.2 Princip činnosti zvoleného řešení

Zvolené řešení bude postavené na základě popisu hry uvedeného v kapitole 5.5. Díky němu bude mít game server informace o jakou hru se jedná, bude možné jednoduše odměňovat hráče a také bude umožněno spouštění na základě akcí a podmínek. Nejprve si tedy rozebereme spouštění hry.

Spouštění hry ilustruje obrázek 5.2. To začíná u provedení hráčské akce v hlavní hře např. hráč kupuje ovoce. Vykonáním této akce se odešle request na webový server, který v daném kontroleru přepošle dotaz pro vyhodnocení dané akce pro příslušného hráče na herní server. Zde dojde k vyhodnocení podmínek vzhledem k akci a aktuálnímu stavu hráče. V případě, že jsou

splněny podmínky pro hry spouštěné na danou akci, jsou opačnou cestou poslány příslušné informace o minihrách, které by si mohl hráč zahrát.

Pokud si pak hráč jednu z nich vybere, jsou opět přes web server poslány informace o spuštění dané hry na game server. Zde dojde k vytvoření objektu minihry a opačnou cestou jsou odeslány informace o vytvořené hře. Pokud se jedná o externí minihru může být hráči zobrazeno id hry a odkaz ke stažení klienta. V případě, že se jedná o webovou minihru může být rovnou spuštěna. O těchto činnostech rozhodne webový server na základě poskytnutých informací.



Obrázek 5.2: Diagram znázornění komunikace pro zvolený způsob integrace.

V případě webové hry probíhá komunikace obdobně jak je znázorněno na obrázku 5.1. V případě externí hry bude komunikace probíhat prostřednictvím webových služeb. Pro oba případy bude potřeba **vytvořit ve vrstvě služeb novou WCF službu** pro minihry (`MinigameService`). Architektura aplikace tak zůstane prakticky nezměněna.

6 Implementace rozhraní

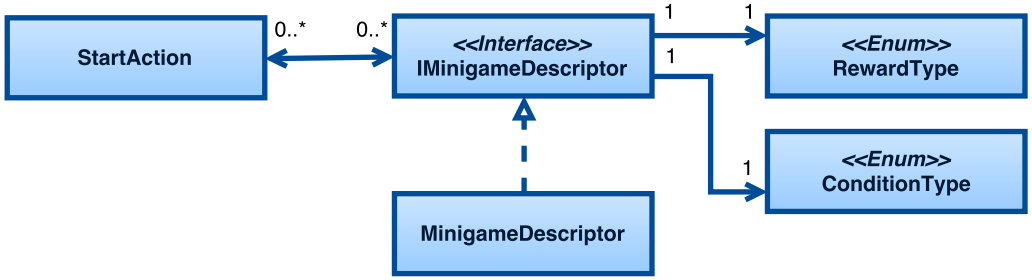
V této kapitole si popíšeme implementaci rozhraní pro minihry. Nejprve si rozebereme realizaci popisování her včetně rozšíření relačního modelu databáze a načítání popisů do hry. Dále přijdou na řadu objekty miniher a jejich životní cyklus. Následně si stručně popíšeme WCF služby pro minihry.

Dále pak detailně rozebereme nejdůležitější část rozhraní a to `MinigameManager`. Zde bude probrána manipulace s hrami, registrace popisů miniher, implementace kontrol a také řešení souběžného přístupu. Ke konci kapitoly bude popsána implementace rozhraní na webovém serveru včetně spouštění her z *GameUI*.

6.1 Popis minihry

První důležitou součástí rozhraní je implementace popisu hry. Popis hry slouží k uchovávání informací o jednotlivých minihrách, ke spouštění her na základě akcí a podmínek a také k jednoduchému odměňování hráče. Zároveň je možné díky němu vytvářet jednotlivé instance her (viz kapitola 6.2).

Popis hry je více méně realizován podle návrhu v kapitole 5.5 a je tvořen rozhraním `IMinigameDescriptor` resp. databázovou entitou `MinigameDescriptor`, která implementuje toto rozhraní. Všechny třídy a jejich vztahy reprezentující popis hry znázorňuje obrázek 6.1.



Obrázek 6.1: Digram tříd pro popis miniher.

Na obrázku můžeme vidět vztah mezi `MinigameDescriptorem` a dvěma enumy. První z nich (`ConditionType`) reprezentuje typ podmínky, na kterou se má popisovaná hra spustit. Druhý (`RewardType`) představuje typ odměny.

Oba výčtové typy obsahují příslušné typy specifikované v kapitolách 5.3 a 5.4. Vztah mezi třídou `StartAction`, která reprezentuje spouštěcí akce a `MinigameDescriptorem` bude podrobněji popsán v následující kapitole. Obě tyto třídy reprezentují databázové entity a jsou tak pro ně vytvořeny příslušné DAO objekty, kde je využíváno objektově relačního mapování.

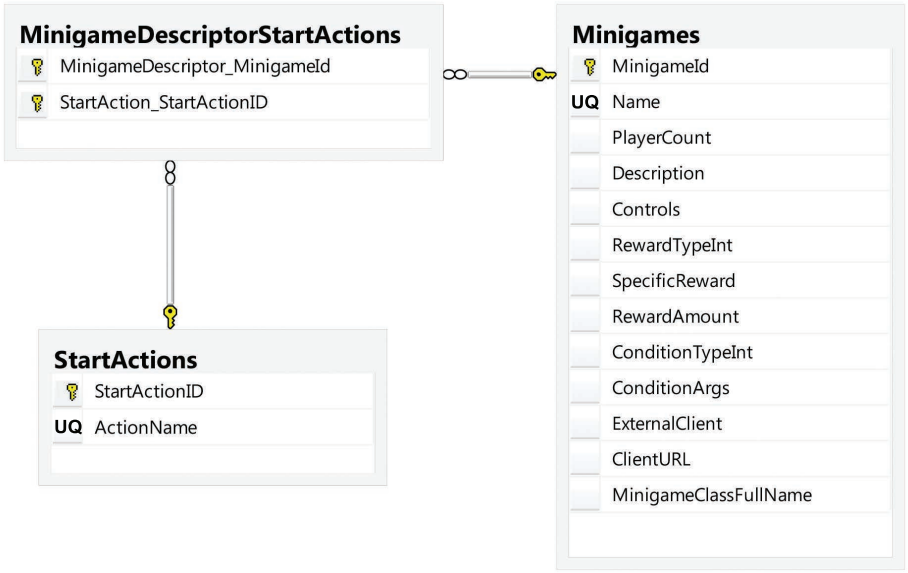
6.1.1 Relační model

Na obrázku 6.2 můžeme vidět relační model databáze pro popis her. Tento model více méně odpovídá UML diagramu z obrázku 6.1. Jednotlivé atributy v tabulce **MinigameDescriptor** odpovídají návrhu z kapitoly 5.5. Navíc jsou zde atributy *SpecificReward*, *ConditionArgs* a *MinigameClassFullName*.

První slouží ke konkretizaci dané odměny, druhý dospecifikovává podmínku spuštění hry a poslední atribut slouží k jednoduchému vytvoření instance minihry na základě názvu třídy[1]. Vztah M ku N mezi tabulkami **StartAction** a **MinigameDescriptor** je z toho důvodu, že jedna hra může být spouštěna na více akcí a zároveň jedna akce může mít přiřazeno více her. Zároveň je nutné podotknout, že záznamy v obou tabulkách jsou unikátní v rámci jejich názvu.

Kromě *SpecificReward*, *ConditionArgs* a *MinigameClassFullName* jsou všechny atributy ve všech tabulkách povinné. Sloupce se sufixy `Int` jsou vytvořeny z toho důvodu, že Entity Framework 4 nedokáže namapovat třídy výčtového typu do databáze a proto je zde provedena jejich konverze na celočíselný typ.

[1]Ve své podstatě se jedná o tzv. Assembly Qualified Name, který kromě názvu třídy obsahuje i název daného assembly. Důvodem použití tohoto názvu je možnost vytvoření instance hry, která je definována i v jiné komponentě než v game serveru.



Obrázek 6.2: Rozšíření relačního modelu databáze pro popis her.

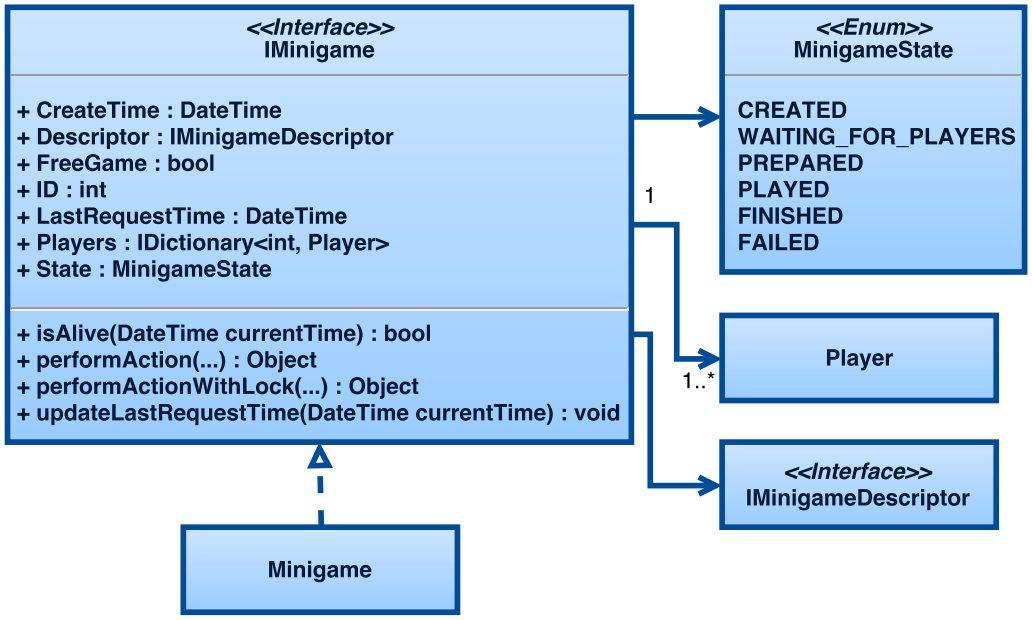
6.1.2 Načítání popisů

V současné době je nutné vytvořit spouštěcí akce a jednotlivé popisy her na začátku startu aplikace. To je zejména z toho důvodu, že pro výslednou práci není tato funkčnost klíčová a také proto, že na to vzhledem k časové náročnosti práce nezbyl čas. V budoucnu je tak možné realizovat načítání z XML souboru obdobně jako například hvězdné systémy. Případně by bylo možné realizovat načítání popisů her i za běhu aplikace.

6.2 Minihry

Druhým důležitým prvkem rozhraní jsou samotné třídy pro vytvoření instancí miniher. Všechny třídy, které by měly reprezentovat nějakou minihru musí implementovat rozhraní `IMinigame`. Toto rozhraní obsahuje signatury a kontrakty pro property a metody potřebné pro **nutnou** reprezentaci instance hry. Hierarchii tříd reprezentujících instance miniher můžeme vidět obrázku 6.3 v podobě UML diagramu tříd.

Jak můžeme vidět každá hra musí obsahovat *čas vytvoření* (`CreateTime`) a *čas posledního requestu* (`LastRequestTime`), *ID* (`ID`), *slovník hráčů* (`Players`) indexovaných podle jejich ID, *referenci na descriptor hry* (Descriptor),



Obrázek 6.3: Diagram tříd pro minihry.

stav hry (`State`) a také *atribut jestli byla hra vytvořena jako volná* (`FreeGame`). Dále musí obsahovat metody pro *kontrolu života hry* (`isAlive`), *provedení akce* (`performAction`) včetně akce se zámkem a také metodu pro *update času posledního requestu* (`updateLastRequestTime`).

Význam jednotlivých property a příslušných metod je popsán přímo v implementaci aplikace a není nutné je zde všechny popisovat. Uvedeme si pouze význam atributu `FreeGame` (*volná hra*). Volná hra se vyznačuje tím, že si ji hráč může zahrát kdykoliv bude chtít a nemusí čekat, než mu bude umožněno spuštění na základě nějaké akce a podmínky[2]. Z takto spuštěných her pak hráč nedostane žádnou odměnu, aby nedocházelo k neustálému obohacování hráče.

Jak bylo uvedeno v kapitole 5.7.1, je možné vytvářet hry s herní logikou přímo v game serveru nebo pouze vytvořit instanci, která bude procházet jednotlivými stavy a logika bude jinde. První variantu je možné realizovat buď již zmíněnou implementací rozhraní `IMinigame` nebo oděděním od třídy `Minigame`. Tato třída pak slouží také pro druhou variantu, kdy jsou v této struktuře uchovávány nejnutnější informace o dané hře.

[2]Spouštění volných her z hlavního menu hry bude umožněno až v dalších fázích vývoje projektu.

6.2.1 Životní cyklus hry

Jak jsme mohli vidět v předchozí kapitole, každá instance minihry by měla být v nějakém stavu. Tento stav popisuje property `State` která může nabývat následujících šesti stavů:

- **Vytvořená** (*CREATED*)
- **Připravená** (*PREPARED*)
- **Čekající na hráče** (*WAITING_FOR_PLAYERS*)
- **Hraná** (*PLAYED*)
- **Dokončená** (*FINISHED*)
- **Nedokončená** (*FAILED*)

Jednotlivé stavy jsou samovysvětlující a není potřeba je nijak zvlášť popisovat. Vývoj životního cyklu hry pak můžeme vidět na obrázku 6.4. Na obrázku je u každého stavu také metoda `MinigameManageru` (označená kurzívou), která minihru do tohoto stavu převede. Tyto metody budou podrobněji popsány v následující kapitole[3].

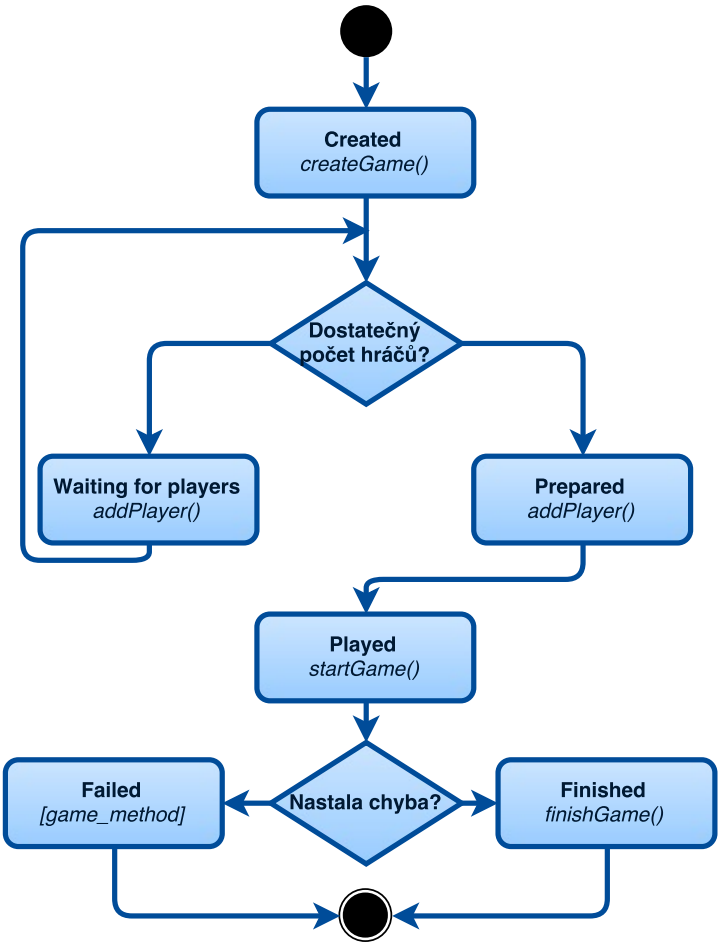
Aby herní server věděl jestli je minihra aktuální (tzv. živá) je nutné od odstartování hry[4], provádět periodické requesty. O kontrolu života hry se stará periodicky plánovaná akce `MinigameLifeControlAction`. Tato akce je poprvé naplánována až po odstartování hry (nikoli od vytvoření) a vždy jednou za minutu zkontroluje (na základě metody `isAlive`) jestli je příslušná instance živá. Pokud zjistí, že není, dojde k ukončení dané instance hry.

Protože zmíněná akce hlídá život hry až po jejím spuštění, běží na game serveru ještě akce `MinigameAllGamesLifeAction`. Ta je plánována každých 30 minut, kontroluje všechny existující instance miniher a hlídá, jestli některé vytvořené hry na serveru tzv. nevisí[5]. Akce vždy zkontroluje jestli hra od svého vytvoření existuje déle jak 30 minut a pokud ano, zkontroluje jestli daná instance žije. V případě, že nežije, dojde k jejímu ukončení.

[3]Označení *[game_method]* nereprezentuje metodu, ale představuje různé akce samotných her.

[4]Odstartováním hry je rozuměno převedení minihry do stavu *PLAYED*.

[5]Tímto výrazem je myšleno, že na příslušné instance existují reference, které nikdo nevyužívá (pravděpodobně už ani využívat nebude) a zabírají tak místo v paměti serveru.



Obrázek 6.4: Diagram životního cyklu minihry.

6.3 Minigame service

Pro komunikaci mezi game serverem a web serverem a také pro komunikaci s externími klienty byla implementována WCF služba `MinigameService`. Tato služba je popsána kontraktem `IMinigameService`, který využívá namespace **http://spacetraffic.kiv.zcu.cz/MinigameService**.

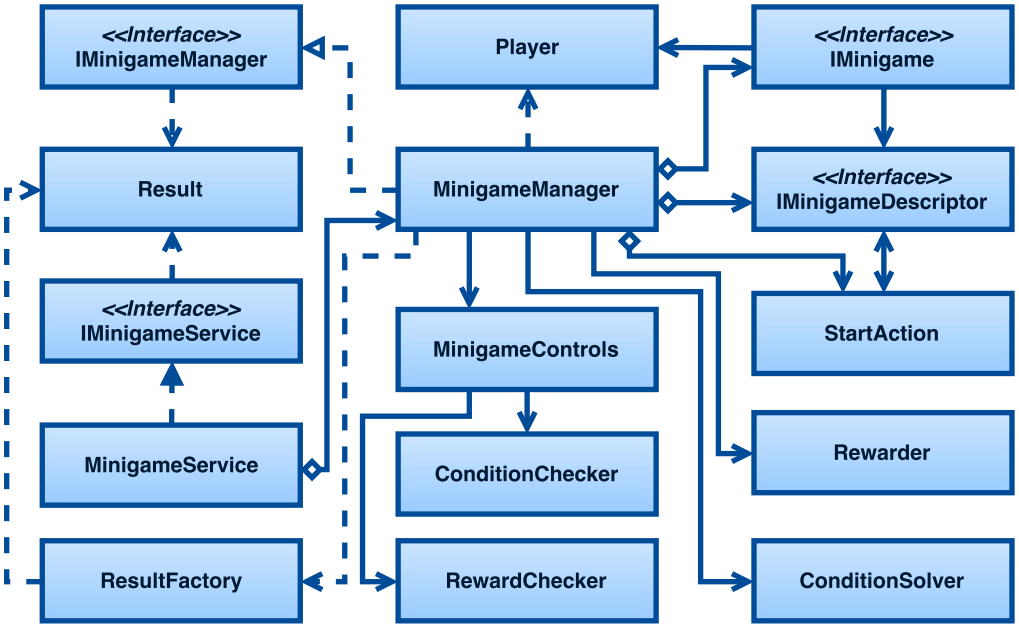
Pro tuto službu jsou realizovány dva koncové body. První využívá jako binding pojmenovanou rouru (Named Pipes), přes kterou komunikuje webový server s herním serverem. Druhý koncový bod má binding realizován pomocí HTTP, což umožňuje komunikaci s externími klienty skrze tento komunikační protokol.

`MinigameService` poskytuje služby pro spuštění a ukončení hry a také pro odměňování hráčů. Dále pak poskytuje možnost volání libovolné me-

tody nad danou instancí hry a také možnosti aktualizace času posledního requestu. Implementace jednotlivých metod jsou pak realizovány pomocí `MinigameManageru`, který bude podrobně rozebrán v následující kapitole.

6.4 Minigame manager

Nejdůležitější částí rozhraní je `MinigameManager`. Tato třída tvoří hlavní řídící prvek díky němuž je možné vytvářet, ukončovat a manipulovat s jednotlivými hrami. Manager také obsahuje prostředky k vyhodnocování podmínek pro spouštění her, odměňování hráčů, různé kontroly nebo např. generování ID miniher. Vztahy manageru s jednotlivými třídami rozhraní jsou znázorněny na obrázku 6.5.



Obrázek 6.5: Diagram tříd rozhraní pro minihry.

6.4.1 Spouštění a vytváření miniher

Jak můžeme vidět z UML diagramu na obrázku 6.5, `MinigameManger` je v agregaci s `IMinigameDescriptor` a se `StartAction`. To je mimo jiné z důvodu[6], že je v manageru slovník `minigamesByStartAction`. Tento slov-

[6]Hlavním důvodem agregace je, že příslušné instance těchto tříd vznikají a zanikají na základě metod manageru.

ník má jako klíčové položky názvy akcí a jako hodnoty instance spouštěcích akcí. Každá akce v tomto seznamu pak obsahuje seznam popisů her (`MinigameDescriptorů`), které je možné na příslušnou akci spustit.

Slovník byl zvolen z důvodu rychlého přístupu k akcím na základě názvu akce. Důvodem proč jsou akce a popisy miniher uchovávány oproti databázi i v manageru je rovněž z důvodu rychlejšího přístupu a tento slovník tak funguje jako read-only cache.

Samotné spouštění pak funguje na základě volání metod `getMinigame` a `getMinigameList`. Tyto metody nejprve podle názvu akce získají seznam příslušných `MinigameDescriptorů` a následně provedou vyhodnocení spouštěcích podmínek pro všechny hry z tohoto seznamu vzhledem k danému hráči. ID her, které je možné spustit, jsou pak vrácena jako návratová hodnota. Zároveň je možné využít obdobu těchto metod, které vrací instance `IMinigameDescriptoru` resp. jejich seznam.

Vyhodnocování podmínek pro jednotlivé hry na základě popisu minihry a ID hráče je realizováno prostřednictvím služebníka `ConditionSolver`. Na základě současného stavu projektu bylo možné implementovat vyhodnocování pouze na podmínky ohledně kreditů a úrovně hráče. Vyhodnocování pro ostatní typy podmínek bude možné realizovat až v dalších fázích vývoje aplikace.

Zároveň je zde umožněno poměrně hodně prostoru pro realizování různorodých podmínek na základě `ConditionArgs` z popisu hry. Zde je možné uvést prakticky cokoliv a pak pouze stačí v tomto služebníkovi doimplementovat metody pro vyhodnocování daných výrazů.

Vytváření her se pak provádí na základě volání metody `createGame`. Ta podle ID `MinigameDescriptoru` a příznaku jestli se má hra vytvořit jako *volná hra*[7], vytvoří instanci příslušné hry. Jaká instance se vytvoří je uvedeno v `MinigameClassFullName` atributu. Pokud je prázdný dojde k vytvoření instance třídy `Minigame`. Pokud ne, vytvoří se instance třídy definované v tomto atributu.

Při vytváření hry se pak vygeneruje ještě unikátní ID a příslušný objekt je pak vložen do slovníku aktivních her (`activeGames`). Tento slovník je indexován identifikátory miniher a byl opět zvolen z důvodu rychlého přístupu k minihrám. Takto vytvořená hra se pak nachází ve stavu *CREATED*.

[7]Co je volná hra je vysvětleno v kapitole 6.2.

6.4.2 Přidávání hráčů a začátek hry

Přidávání hráčů je umožněno voláním metody `addPlayer`. Ta na základě ID hry a ID hráče přidá do příslušné hry daného hráče. Pokud byl do hry přidán poslední hráč potřebný pro hraní hry, přejde hra do stavu *PREPARED*. V opačném případě je nastaven stav *WAITING_FOR_PLAYERS*. Před samotným přidáním hráčů do hry pak probíhají různé kontroly, které není potřeba popisovat. Důležité je zmínit, že je umožněno přidávat do hry hráče pouze pokud je ve stavu *CREATED* nebo *WAITING_FOR_PLAYERS*.

Začít hru je možné prostřednictvím volání metody `startGame`. Ta odstartuje hru podle daného identifikátoru, což představuje změnu stavu na *PLAYED* a naplánování akce `MinigameLifeControlAction` pro kontrolu života hry. Odstartování je dovoleno provést pouze za předpokladu, že bude hra ve stavu *PREPARED*. Od této chvíle je nutné periodicky provádět aktualizaci času posledního requestu a to nejdéle v rozmezí definovaném příslušnou hrou (pro `Minigame` a všechny její potomky, kde není definováno jinak je to 1 minuta). Pokud k aktualizaci nebude docházet, bude hra ukončena serverem, resp. již zmíněnou akcí. Po odstartování hry je také hráči nastaven atribut, že hraje minihru a také jakou hru hraje.

6.4.3 Ukončení hry a odměňování

Ukončení hry je možné realizovat voláním metody `endGame`, kdy dojde ke změně stavu hry na *FINISHED*. Ukončit hru je možné kdykoliv bez ohledu na předcházející stav. To je z toho důvodu, že tuto metodu využívají akce pro kontrolu života minihry a také proto, že díky jejímu volání je hráči odebrán atribut, že hraje hru.

Po ukončení hry lze provést odměnění hráče resp. hráčů. To se provádí voláním metody `rewardPlayer`. Odměňování je možné pouze v případě, že je hra ve stavu *FINISHED* a také pokud nebyla vytvořena jako *volná hra*. Samotné odměňování zajišťuje třída `Rewarder`, která je realizovaná jako služebník. Tato třída odmění hráče na základě popisu odměny v `MinigameDescriptoru`. Vzhledem k současnému stavu projektu bylo možné implementovat pouze odměňování kredity a zkušenostmi. Ostatní možnosti odměn bude možné realizovat až v pozdějších fázích vývoje.

Po odměnění je ještě nutné hru odstranit[8], aby nebyla zbytečně uchovávána ve slovníku aktivních her. Odstranění se provádí volání metody `removeGame` a je možné provést pouze pokud je hra ve stavu *FINISHED* nebo *FAILED*.

6.4.4 Provádění akcí

Nad jednotlivými instancemi miniher je možné provádět různé akce. To lze provést voláním metod `performAction` a `performActionWithLock`. Prostřednictvím těchto metod je na základě ID hry, názvu metody a jejích argumentů možné provádět libovolnou akci nad danou instancí minihry a v případě druhé metody je možné během vykonávání zamknout příslušnou hru (více v kapitole 6.4.7). Pokud akce obsahuje návratovou hodnotu, je tato hodnota zabalena do objektu `Result`.

Tato třída reprezentuje výsledek konkrétní akce. Obsahuje zejména status, tedy jak akce dopadla (**SUCCESS**/**FAILURE**), příslušnou zprávu a také návratovou hodnotu. K vytváření resultů slouží tovární třída `ResultFactory`. `Result` je také využíván u spousty ostatních metod `MinigameManageru`, aby bylo možné prostřednictvím služby sdělit jak daná akce dopadla.

Provádění různých akcí nad danou hrou je možné od vytvoření do odstranění hry. Hlavním důvodem proč tato možnost není omezena např. od odstartování do ukončení hry je možnost provádět různé inicializační a ukončovací akce, které je potřeba provést před odstartování resp. po ukončení minihry.

6.4.5 Registrace a deregistrace her

`MinigameManager` umožňuje mimo jiné registraci a deregistraci miniher. Tyto činnosti spočívají v uložení popisů her do databáze resp. odebrání z databáze. Tyto operace umožňují metody `registerMinigame` a `deregisterMinigame`. Po úspěšném uložení nebo odebrání příslušného popisu hry z databáze dojde ke znovunačtení slovníku `minigamesByStartAction`. Tento krok je vzhledem k současnému načítání her zbytečný. Implementován je zde z toho důvodu, že se do budoucna počítá s načítáním miniher za běhu serveru. V manageru

[8]Pokud se tak nestane, o odstranění se postará jedna z plánovaných akcí na kontrolu života.

jsou ještě implementovány metody pro přidání a odebrání spouštěcích akcí k příslušným popisům her, kde se rovněž počítá s tímto načítáním.

6.4.6 Kontroly

U všech metod `MinigameManageru` uvedených v předchozích pěti kapitolách se provádí různé kontroly. Tyto kontroly jsou realizovány prostřednictvím třídy `MinigameControls`. Pro kontrolu popisů her při registraci tato třída využívá služebníky `RewardChecker` (pro kontrolu odměn) a `ConditionChecker` (pro kontrolu podmínek spouštění). U obou služebníků bylo kvůli současnému stavu projektu možné realizovat pouze kontroly pro kredity a zkušenosti. Stejně jako u odměňování a řešení podmínek pro spouštění bude možné implementovat kontroly pro další typy až v dalších fázích vývoje aplikace.

6.4.7 Souběžný přístup

Z důvodu umožnění komunikace skrze webové služby je nutné řešit otázku souběžného přístupu k jednotlivým hrám. Pro řešení tohoto problému bylo možné zvolit dvě možnosti. První z nich je nakonfigurování WCF služeb tak, že budou realizovány pouze jedním vláknem pro všechny klienty. Tím může (hlavně při vyšší frekvenci komunikace více klientů) docházet ke značnému zpomalování herního serveru a nárůstu odezvy při komunikaci.

Druhým řešením je pak ošetření kritických sekcí v `MinigameManageru` a v případě provádění souběžných operací nad příslušnými instancemi miniher nechat rozhodnutí ošetření na jejich tvůrcích. Při tomto způsobu by nemělo docházet k razantnímu zpomalování game serveru ani velkému nárůstu odezvy při komunikaci.

Jednu z kritických sekcí manageru tvoří slovník `minigamesByStartAction`. Tato kritická sekce je ošetřena pomocí zámku nad přidruženým objektem v metodě používané pro přístup k tomuto slovníku. Druhým atributem, který je potřeba ošetřit je čítač pro generování identifikátorů miniher (`minigameCounter`). Ten je ošetřen stejným přístupem jako zmíněný slovník.

Poslední kritickou sekci v `MinigameManageru` tvoří slovník aktivních her (`activeMinigames`), který je realizován jako **ConcurrentDictionary**. Tato kolekce je navržena tak, aby ošetřovala souběžný přístup a také, aby při větší

frekvenci přidávání a odebírání položek nedocházelo k velkému zpomalování. Důvodem volby této implementace je předpoklad častého přidávání a odebírání miniher a na rozdíl od zamykání slovníku zámkem by mělo toto řešení být vhodnější. K operaci s touto kolekcí jsou pak implementovány příslušné metody.

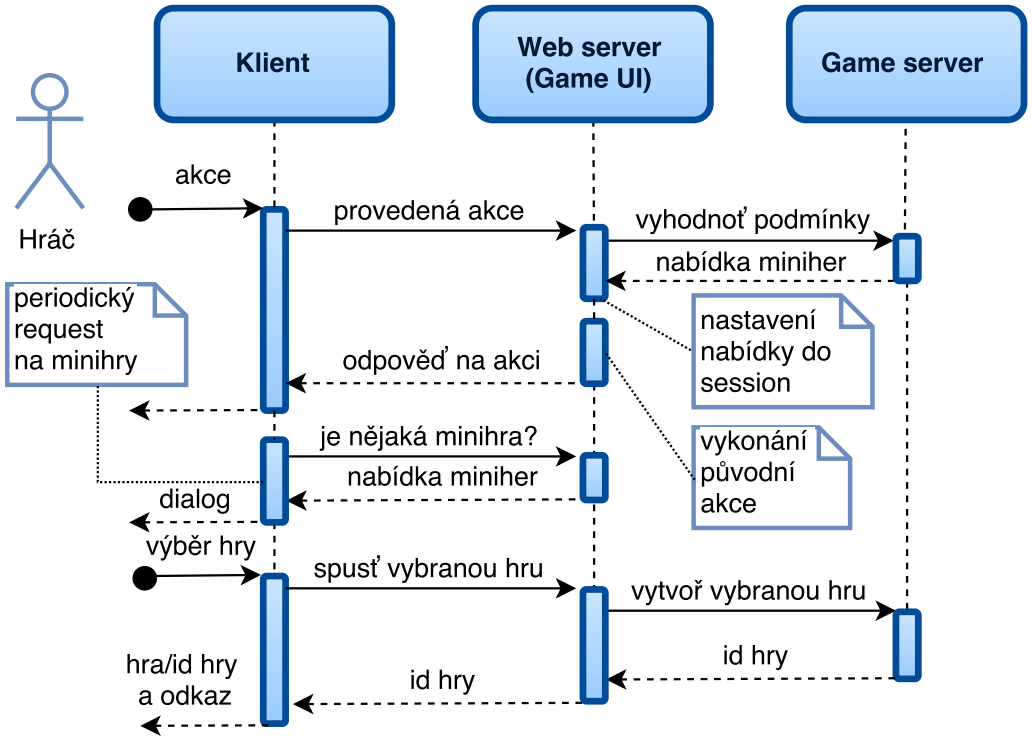
I když je přístup ke hře ošetřen, v případě samotné manipulace se hrou už tomu tak není. K tomuto účelu slouží metoda `performActionWithLock`, která nejprve uzamkne instanci nad příslušným objektem a pak provede danou akci. Kdy tuto metodu použít záleží na programátorovi příslušné hry.

6.5 Spouštění her z web serveru

V kapitole 6.4.1 bylo uvedeno jak probíhá spouštění her na herním serveru. V této kapitole si shrneme princip spouštění z pohledu webového serveru. Princip komunikace při spouštění her z můžeme vidět na obrázku 6.6. Tato činnosti probíhá následujícím způsobem.

Po vykonání hráčské akce se odešle request na webový server. Tam se v příslušném kontroleru zavolá metoda `AbstractControlleru`, která provede komunikaci s game serverem. Na základě této komunikace jsou hry, které je možné hrát nastaveny do session s názvem *minigame*. Následně se provede ještě původně volaná akce a vrátí se její výsledek klientovi. V klientovi běží javascript, který využívá ajaxový systém zpráv a periodicky se doptává jestli není možné hrát nějakou minihru. Pokud je v session nastavena nějaká hra, vyvolá dialogové okno.

V tomto okně si hráč může zvolit hru, kterou chce hrát. Jeho potvrzením dojde k odeslání požadavku (opět již zmíněným systémem) o spuštění dané hry na webový server a následně na game server. Ten vytvoří příslušnou instanci hry a vrátí přes web server její identifikátor. Javascript pak na základě ID a předchozích informací o možných minihrách vyvolá buď nové okno s hrou nebo dialog s informací o příslušném ID a možností stažení externího klienta. Jednotlivé části umožňující spouštění z web serveru budou popsány v následujících podkapitolách. Ukázku dialogů ze spouštění miniher můžeme vidět v příloze C.



Obrázek 6.6: Sekvenční diagram spouštění her z webového serveru.

6.5.1 Úprava AbstractControlleru

Do `AbstractControlleru` byly doimplementovány metody `evaluateMinigameByStartActionName` a `evaluateMinigameListByStartActionName`. Tyto metody zajišťují vyhodnocení spouštěcích podmínek na základě názvu akce a nastavují do session descriptory her, které je možné hrát. Od tohoto kontroleru dědí všechny herní kontrolery a pro realizaci vyhodnocování her na základě příslušných herních akcí je potřeba v dané akci zavolat jednu z těchto dvou metod.

6.5.2 Ajaxový systém zpráv

Ajaxový sytém zpráv slouží k jednoduché ajaxové komunikaci mezi klientem a web serverem. Tento systém byl implementován Janem Kotalíkem a slouží primárně k aktualizaci uživatelského rozhraní. Tato komponenta funguje jako fronta požadavků, která se vyprazdňuje v pravidelných intervalech tím, že se tyto požadavky odešlou hromadně na server pomocí AJAXu. Zde jsou pak jednotlivé požadavky obslouženy pomocí handlerů, tedy tříd, které

implementují rozhraní `IAjaxHandable`. Výsledek této obsluhy je pak vrácen zpět na server a zpracován pomocí callback metod.

Každý požadavek systému zpráv musí obsahovat *unikátní ID*, *název handleru*, *data*, která chceme poslat ke zpracování handleru, *callbackovou metodu* a volitelně také *interval opakování požadavku*. Každý handler pak kromě implementování již zmíněného rozhraní musí být v namespacu **SpaceTraffic.GameUi.Controllers.AjaxHandlers**.

6.5.3 Spouštění miniher z klienta

O spouštění miniher z klienta se stará javascript **MinigameStarter.js** a handler `MinigameStarter`. Javascriptový modul se na základě ajaxového systému zpráv doptává každé 2 vteřiny jestli neexistuje nějaká minihra, kterou by bylo možné hrát. Tyto požadavky handler zpracuje a pokud je v session *minigame* nastaven descriptor nějaké hry (případně seznam descriptorů) odešle je jako výsledek zpět na klienta a danou session smaže.

V klientovy je pak volán callback, který zajistí vyvolání dialogového okna realizovaného prostřednictvím jQueryUI. V tomto okně může hráč buď potvrdit (v případě seznamu her i vybrat příslušnou hru), že chce hrát hru, nebo odmítnout. Pokud dojde k potvrzení, je opět odeslán požadavek na server. Handler zprostředkuje vytvoření hry a vrátí její ID. V callbacku je pak v případě externí minihry vyvolán dialog, kde je uvedena URL na klienta minihry a také ID vytvořené hry.

Pokud se ale jedná o webovou minihru, je otevřeno nové okno, které je přesměrováno na URL adresu dané hry (adresa je definována v `MinigameDescriptoru` v atributu `ClientURL`). Od této chvíle si pak hra zajišťuje vše sama (přidávání hráčů do hry, start hry, periodické requesty apod.) a to i v případě externího klienta. V *GameUI* je pro minihry vyhrazen namespace **SpaceTraffic.GameUi.Areas.Minigame**. Zde je možné vytvořit příslušná view, kontrolery a modely a realizovat tak vlastní webovou minihru. V tomto namespacu jsou pak kontrolery namapovány na URI ve tvaru `http://<adresa_serveru>/Minigame/<controller>`.

6.5.4 Zamezení ovládání hry

Ve chvíli, kdy hráč hraje minihru, neměl by pokud možno ovládat hlavní hru. To zajišťuje javascriptový modul **PlayerIsPlayingMinigame.js** a ajaxový handler `PlayerIsPlayingMinigame`. Ten se stejně jako modul pro spouštění miniher každé 2 vteřiny doptává pomocí ajaxového systému zpráv, jestli aktuálně přihlášený hráč nehraje nějakou hru. Tyto requesty obsluhuje již zmíněný handler, který si příslušné informace zjistí z game serveru a předá je klientovi.

V případě, že klient zjistí, že hráč nějakou hru hraje zobrazí dialogové okno s příslušnou informací a možností zavřít toto okno. Pokud hráč toto okno zavře, dojde k ukončení a odstranění minihry z `MinigameManageru` a aktuálně rozehraná hra by na to měla příslušným způsobem zareagovat. Reakci zajišťuje programátor dané minihry a je jen na něm jestli hru nechá hráče dohrát a pak oznámí, že hra byla ukončena nebo tento fakt oznámí po zjištění periodickými requesty či na základě jiných požadavků.

Pokud klient zjistí, že hráč nehraje žádnou minihru a je otevřen zmíněný dialog, dojde k jeho zavření. Tím je zajištěno, že pokud hráč ukončí hru regulérním způsobem (dohrání, případně zavření okna minihry[9]) dojde k zavření tohoto dialogu automaticky.

[9]Ukončení a odebrání hry na základě zavření okna musí taktéž zajistit tvůrce minihry.

7 Implementace miniher

V této kapitole si rozebereme implementační detaily obou vytvořených miniher. Tyto hry slouží především jako příklad jak je možné hry do Space Trafficu implementovat na základě navrženého rozhraní. U každé hry si specifikujeme jak serverovou, tak klientskou část a obě části doplníme UML diagramy tříd. Dále si nastíníme princip činnosti každé hry a také jaké kontroly probíhají v obou hrách jak na herním serveru, tak v příslušných klientech. Ukázky obou miniher můžeme vidět v příloze D.

7.1 První minihra – Spaceship Cargo Finder

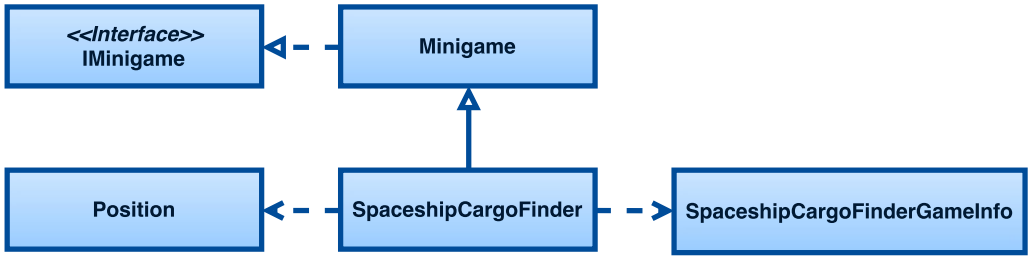
Tato hra s názvem **Spaceship Cargo Finder** není nic jiného než variace na legendární hru **Had**. Tato variace se od původní hry liší pouze tím, že hráč ovládá místo hada vesmírnou loď s vagóny nákladu a místo ovoce sbírá náklad uprostřed galaxie[1]. Tato hra byla zvolena, protože je poměrně oblíbená, zábavná a také jednoduchá na ovládání. Hra je implementována jako webová hra, kde herní logika probíhá v klientovi a game server v pravidelných intervalech provádí její kontroly. Ukázky ze hry je možné vidět v příloze D.1.

7.1.1 Implementace na game serveru

Hierarchii tříd pro část hry implementovanou na herním serveru můžeme vidět v UML diagramu na obrázku 7.1. Jak můžeme vidět třída `SpaceshipCargoFinder` je vytvořena oddděděním od třídy `Minigame`. Tato třída využívá třídu `Position`, která slouží jako přepravka pro uchovávání souřadnic a zároveň představuje jednotlivé objekty hry (tělo hada a jídlo).

Tato třída ještě využívá `SpaceshipCargoFinderGameInfo`, která slouží k přenosu popisu základních informací hry (výška a šířka herní plochy, velikost jednoho políčka hry, vítězné skóre, ...) na webový server přes *MinigameService*.

[1] V následujících podkapitolách bude hovořeno o hadovi a jídle, ale ve skutečnosti se jedná o loď a náklad.

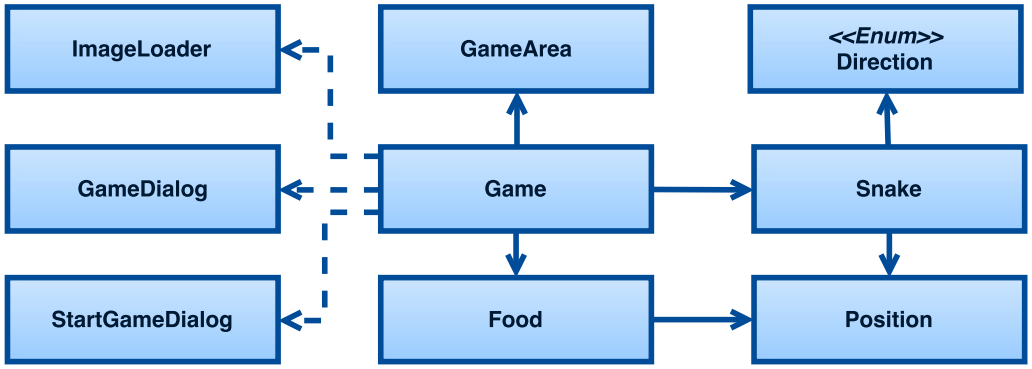


Obrázek 7.1: UML diagram tříd pro první hru na herním serveru.

7.1.2 Implementace na web serveru

Na webovém serveru je tato hra implementována ve dvou částech. První část tvoří kontroler `SpaceshipCargoFinderController` společně s jeho view a také ajaxové handlery `StartSpaceshipCargoFinder` a `PerformAction-SpaceshipCargoFinder`. Druhou část pak tvoří klientský javascript. Hierarchii tříd tohoto javascriptu můžeme vidět UML diagramu na obrázku 7.2.

Jak můžeme vidět, hlavním prvkem je třída `Game`. Tato třída obsahuje reference na ostatní prvky hry (`Snake`, `Food`, `GameArea`). Zároveň také využívá třídu `ImageLoader` k načítání obrázků a třídy `GameDialog` a `StartGameDialog` k informování hráče.



Obrázek 7.2: UML diagram tříd pro první hru v javascriptu.

7.1.3 Princip činnosti

Poté co se hráč rozhodne hrát hru a ta je následně spuštěna mechanismem popsaným v kapitole 6.5.3, dochází k přesměrování vytvořeného okna na kontroler `SpaceshipCargoFinderController`. Kontroler si nejprve vyžádá informace o hře a následně je předá do příslušného view. Ve view nejprve

dojde k nastavení potřebné velikosti okna a jeho vycentrování a následně dojde k inicializaci hry a zobrazení spouštěcího dialogu (`StartGameDialog`).

Inicializace hry spočívá ve vytvoření jednotlivých herních prvků, načtení obrázků `ImageLoaderem` a jejich vykreslení do canvasu. Poté co se hráč rozhodne z úvodního dialogu spustit hru, pošle se skrze systém zpráv (nezávislý na hlavní hře) požadavek ke spuštění hry. V handleru `StartSpaceshipCargoFinder` dojde k přidání hráče do hry a k jejímu odstartování. V následném callbacku je pak spuštěno odpočítávání startu hry a také se inicializuje odesílání 2 periodických requestů.

Prvním z requestů je odesílání updatu času posledního requestu každých 15 sekund, aby game server věděl, že hra stále žije. Druhým periodicky odesílaným požadavkem je kontrola kolize a pozice hada a to každých 10 sekund. Oba requesty zpracovává handler `PerformActionSpaceshipCargoFinder`.

Po startu hry je každých 80 ms prováděna herní smyčka. V této smyčce se nejprve provede vygenerování nové pozice hada, následně dojde ke kontrole kolizí s novou pozicí a posunu hada na novou pozici. Pak už je provedeno pouze vykreslení všech herních elementů. O kontrolách budeme podrobněji hovořit v následující kapitole. Při přesunu hada na novou pozici se kontroluje jestli na této pozici není jídlo. Pokud ano, dojde k prodloužení hada o jeden dílek, vygenerování nového jídla a přičtení skóre. V opačném případě je pouze posunut had na novou pozici.

Při přičítání skóre se zároveň odesílá požadavek o zvýšení skóre v instanci hry na herním serveru. Tento požadavek zpracovává stejný handler jako předchozí requesty. Pokud pak do callbacku dorazí zpráva, že skóre je vítězné, zobrazí se hráči dialog (`GameDialog`). Následně jsou systémem zpráv odeslány požadavky na odměnění hráče a ukončení hry. Oba požadavky zpracovává handler pro provádění akcí a v případě ukončení hry rovnou provádí i její odstranění.

7.1.4 Kontroly

Jak již bylo uvedeno, kontroly probíhají jak v klientovi, tak v herním serveru. Nejprve si probereme kontroly v game serveru. Zde slouží ke kontrole metoda `checkCollision` ve třídě `SpaceshipCargoFinder`. Tato metoda kontroluje na základě těla hada (reprezentovaného seznamem instancí třídy `Position`), jestli se hráč nepohybuje mimo herní oblast a také jestli se had nekříží. Pokud

je objeven některý z těchto prohřešků, je přes *MinigameService* a systém zpráv oznámen až ke klientovi a hra je ukončena.

Kontrola v klientovi je obdobná. V herní smyčce se vždy kontroluje jestli příští pozice hada není v kolizi jak s herní oblastí, tak se samotným hadem. Pokud je tato kolize objevena, je hra ukončena a hráči se zobrazí dialog o prohře (instance třídy `GameDialog`). V klientovi probíhá ještě jedna kontrola a to při generování nového jídla. Zde je dbáno na to, aby se nově vygenerovaný herní prvek nepřekrýval s hadem a aby nebyl vygenerován mimo herní oblast.

7.2 Druhá minihra – Logo Quiz

Tato minihra s názvem **Logo Quiz** je variace na různé kvízové hry, kde je hlavním úkolem uhádnout jaké logo je na obrázku. V této hře jde především o to, že hráč dostane 30 otázek, kde v každé otázce je neúplné logo nějaké světové značky a tři možnosti. Cílem hry je uhádnou alespoň 20 ze 30 otázek. Na každou otázku je časový limit 39 sekund, během kterých je nutné odpovědět, jinak hra končí.

Logo Quiz je realizován jako externí minihra, kde klient je implementován jako aplikace pro mobilní platformu Android (minimální verze Androidu je 4.4 – Kitkat). Herní logika je opět rozdělena jak na část serverovou, tak klientskou. Tento typ hry byl zvolen kvůli své popularitě, zábavnosti a také jednoduchosti.

Mobilní klient pro Android byl vybrán z důvodu popularity mobilních aplikací a protože Android je nejrozšířenější mobilní platformou. Zde je nutné ještě poznamenat, že ve hře jsou použity obrázky z [Log(2016)] a zvuky z [Rec(2016)]. Aplikace je v současné verzi přizpůsobena pro běh v emulátoru a hlavní hra musí běžet na localhostu. Ukázky ze hry je možné vidět v příloze D.2.

7.2.1 Implementace na game serveru

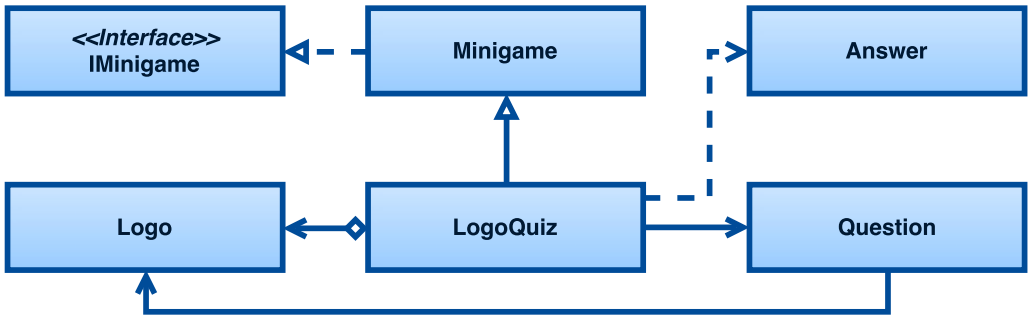
Diagram tříd části minihry implementované na herním serveru můžeme vidět na obrázku 7.3. Stejně jako u předchozí hry je třída `LogoQuiz` oděděna od třídy `Minigame`. Jak můžeme vidět, je tato třída v agregaci s třídou `Logo`. To

je z toho důvodů, že všechny instance třídy `Logo` jsou obsaženy ve statickém seznamu této třídy a jsou používány při generování otázek (`Question`). Pokud by pak instance třídy `LogoQuiz` zanikla, jednotlivá loga stále existují, ale sami o sobě nemají příliš velký význam.

`Logo` je reprezentováno názvem dané značky a názvem souboru s příslušným obrázkem na klientovi. Seznam všech log je načítán `MinigameManagerem` při spuštění serveru. Samotné načítání probíhá prostřednictvím `LogoQuizLoaderu` z XML souboru *logos.xml*. Struktura tohoto souboru je přizpůsobena třídě `Logo`, každý element loga tedy obsahuje stejné atributy jako tato třída. Ukázku tohoto souboru můžeme vidět v následujícím fragmentu kódu:

```
<?xml version="1.0" encoding="utf-8"?>
<logos>
  <logo>
    <name>Abarth</name>
    <file_name>abarth.png</file_name>
  </logo>
  <logo>
    <name>Absolut</name>
    <file_name>absolut.png</file_name>
  </logo>
  ...
</logos>
```

`LogoQuiz` dále využívá třídu `Question`, reprezentující jednotlivé otázky, které se skládají vždy ze správné odpovědi (loga) a dvou náhodných špatných odpovědí. Ke zpracování odpovědí je využívána přepravka `Answer`. Ta obsahuje vždy identifikátor otázky a zvolenou odpověď.



Obrázek 7.3: UML diagram tříd pro druhou hru v game serveru.

7.2.2 Implementace klienta pro android

Diagram tříd pro klientskou aplikaci můžeme vidět v příloze B. Základ tvoří třídy `LogInActivity` a `GameActivity`. Z těchto dvou tříd je ovládána celá aplikace. Nejprve je na základě `LogInActivity` provedeno přihlášení a start hry a následně je vytvořena `GameActivity`. Ta realizuje větší část herní logiky a řídí samotnou hru.

Pro komunikaci s herním serverem slouží třída `Sender`. Tato třída využívá ke komunikaci knihovnu **ksoap2**[2], která umožňuje jednoduchou komunikaci Androidu s webovými službami pomocí SOAP protokolu. Tato knihovna byla vybrána, protože je jednou z nejpoužívanějších pro komunikaci s webovými službami na Androidu a také, protože lze pomocí stránky [Wsd(2016)] vygenerovat API na základě WSDL popisu služby, jež komunikaci ještě více usnadňuje.

Jak se později ukázalo, nebylo vygenerované API využitelné dle představ a přinášelo spoustu problémů. Nakonec byla vytvořena třída `Sender` a serializovatelné třídy `Result`, `Answer`, `Result` a `Question` reprezentující třídy popsané datovými kontrakty *MinigameService*, které toto API nahradily.

Protože v Android aplikacích není možné komunikovat s internetem z hlavního vlákna, byla vytvořena třída `MyAsyncTask`, která reprezentuje asynchronní task. Tato třída využívá jednak instance třídy `Sender` pro komunikaci, ale také instance tříd implementujících rozhraní `IPostAction`. Toto rozhraní definuje kontrakt pro třídy s callback metodami, které jsou volány vždy po dokončení asynchronního úkolu. Tyto třídy jsou vždy implementovány jako anonymní vnitřní třídy obou hlavních tříd aplikace[3].

Obě hlavní třídy ještě využívají `InternetBroadcastReceiver`, který zajišťuje kontrolu připojení k internetu a třídu `Result`, kterou vrací *MinigameService* jako výsledek většiny služeb. Přihlašovací aktivita ještě využívá třídu `MinigamePasswordHasher` pro šifrování hesla, kde je využíváno šifrovacího algoritmu AES.

Herní aktivita pak navíc využívá třídy `Sounds`, pro přehrávání zvuků, `MyCountDownTimer`, pro odpočet zbývajícího času a také třídu `AlertDialogFactory`. Ta obsahuje metody pro vytvoření předem definovaných `AlertDialogů`. Poslední třídou, kterou je nutné zmínit je `Answers`, která dědí od

[2]Knihovna byla použita z [Kso(2015)].

[3]V UML diagramu jsou vazby s těmito třídami znázorněny kolečkem s křížkem uprostřed.

kolekce `ArrayList<Answer>` a hlavní metodu, kterou obsahuje navíc oproti předkovi je metoda `getXml`.

Ta umožňuje převést kolekci na řetězec ve formátu XML. To je z toho důvodu, že již zmíněná knihovna nedokáže přenést kolekci instancí tříd popsaných v kontraktu *MinigameService* z klienta na server (opačným způsobem vše funguje). Tento problém byl vyřešen již zmíněným převodem kolekce na řetězec, který se na serveru rozparsuje XML parserem zpět na jednotlivé odpovědi.

7.2.3 Princip činnosti

Po vytvoření minihry z hlavní hry popsané v kapitole 6.5.3, je hráči zobrazeno dialogové okno s ID hry a také odkazem, kde si může stáhnout klienta hry. Po spuštění klienta na mobilním zařízení se spustí `LogInActivity` a hráč musí zadat *uživatelské jméno*, *heslo* a *ID hry*. Po potvrzení je heslo zašifrováno a odesláno společně s uživatelským jménem na server, kde dojde k jeho ověření. Následně je pak ještě odeslán požadavek na přidání hráče do hry a odstartování hry. V případě, že je hra v pořádku odstartována, je vytvořena instance `GameActivity`.

Ta nejprve zažádá server o vygenerování otázek. Na herním serveru se v třídě `LogoQuiz` vygeneruje seznam 30 unikátních otázek, které jsou uchovány v instanci hry a zároveň odeslány klientovi. Klient si tyto otázky uloží do svého lokálního seznamu a začne odesílat každých 25 sekund informace na server, že hra žije (update času posledního requestu). Následně pak dojde k zobrazení informačního dialogu o hře.

Po zavření dialogu dojde k nastavení první otázky a spuštění časovače. Poté má hráč 39 sekund na to, aby vybral jednu z odpovědí jinak automaticky prohrál. Po zvolení odpovědi dojde k jejímu uložení do seznamu odpovědí (instance třídy `Answers`) a přehrání příslušného zvuku podle toho jestli byla odpověď správná či nikoliv. Následně je nastavena další otázka ze seznamu a resetován časovač. Pokud takto hráč projde všech 30 otázek, je na server odeslán seznam odpovědí, resp. již zmíněný řetězec se seznamem odpovědí. Zde je zkontrolována správnost odpovědí a klientovi je odesláno jestli vyhrál či nikoli.

Po obdržení výsledku hry je buď odeslán požadavek na odmění hráče a následné ukončení hry nebo rovnou na ukončení[4]. Následně je hráči zobrazen dialog o výsledku hry a hráč může hru ukončit. V průběhu hry je např. v případě nutnosti hovoru hra pozastavena, ale periodické requesty se odesílají neustále, aby nedošlo k ukončení hry ze strany serveru.

7.2.4 Kontroly

Stejně jako v případě předchozí minihry, probíhají kontroly jak na serveru, tak v klientovi. Z důvodu šetření mobilních dat ale kontrola na serveru proběhne pouze jednou a to na konci hry. Zde se pak kontroluje správnost jednotlivých odpovědí odeslaných z klienta vůči vygenerovaným otázkám uložených v instanci hry na herním serveru.

Nejprve dochází ke kontrole odpovědí z hlediska jejich počtu a identifikátorů otázek. Kontroluje se počet přijatých odpovědí, jestli nejsou v identifikátorech odpovědí duplicity a jestli jsou jejich ID ve stejném rozsahu jako ID otázek (např. aby odpovědi neměly ID od 10 do 40 a otázky od 1 do 30). Dále se pak kontroluje samotná správnost odpovědí. Zde se vždy na základě ID otázky porovná zvolená odpověď se správnou odpovědí. Pokud je vyhodnoceno 20 a více správných odpovědí, je hráči oznámeno, že vyhrál.

Na klientovi je kontrolována správnost odpovědi vždy po zodpovězení příslušné otázky. Jestli hráč odpověděl správně či nikoli se dozví na základě přehrání příslušného zvuku. V klientovi pak probíhá ještě kontrola existence hry na herním serveru na základě periodických requestů (update času posledního requestu). To je z toho důvodu, kdyby se hráč rozhodl ukončit minihru z hlavního okna hry.

[4]Ukončením hry je v obou případech myšleno i její odstranění.

8 Testování

V této kapitole bude popsáno testování jednotlivých součástí vytvořených během realizace této práce. Nejprve bude popsáno testování rozhraní pro implementaci miniher na základě jeho dílčích částí. Následně pak bude popsáno testování obou miniher a jejich integrace do projektu. Testování probíhalo průběžně během implementace projektu a v případě objevení chyb nebo nedostatků byla provedena jejich oprava.

8.1 Testování rozhraní

Testování rozhraní bylo rozděleno na tři části. První částí bylo testování **popisů her** a to zejména operace s databází. Ve druhé části byl testován samotný **Minigame manager**, kde se ověřovaly všechny jeho metody. Jako poslední pak bylo otestováno **spouštění her** z *GameUI*.

8.1.1 Popis her

U popisů her byly testovány zejména operace s databází. Protože operace s databází jsou zajišťovány pomocí DAO objektů, byly k těmto objektům vytvořeny jednotkové testy. Vzhledem k tomu, že se popis her skládá ze dvou entit (`MinigameDescriptor` a `StartAction`), kde obě entity mají své DAO třídy (`MinigameDescriptorDAO` a `StartActionDAO`), byly vytvořeny jednotkové testy pro obě dvě DAO třídy.

Pro operace s `MinigameDescriptorem` byla vytvořena testovací třída `MinigameDescriptorDAOTest`. Tato třída testuje nejen vkládání, upravování a mazání příslušných objektů z databáze, ale také operace pro přidání a odebrání vazeb mezi popisem hry a spouštěcí akcí. V těchto testech jsou pak ještě ověřovány různé operace na získání příslušných objektů z databáze podle různých kritérií (např. název hry, ID) nebo seznamu všech popisů her.

Pro testy operací se spouštěcími akcemi byly vytvořeny testy `StartActionDAOTest`. V této testovací třídě jsou ověřovány operace pro vkládání, upravování a mazání spouštěcí akcí. Zároveň jsou ověřovány metody pro

získávání jednotlivých instancí dle různých kritérií a také metoda pro získání seznamu spouštěcích akcí včetně příslušných popisů her. V obou případech jsou tedy otestovány všechny operace, které poskytují příslušné DAO třídy.

8.1.2 Minigame manager

Vzhledem k tomu, že Minigame manager tvoří hlavní část celého rozhraní, byl testován jak jednotkově, tak byla otestována jeho integrace do game serveru. Pro jednotkové testy byla vytvořena třída `MinigameManagerTest`. Protože samotný manager potřebuje pro správnou funkčnost ještě instanci `GameServeru` a několika dalších managerů, museli být vytvořeny příslušné mock objekty. Sem patří `GameServerMock`, `GameManagerMock`, `PersitanceManagerMock` a `WorldManagerMock`. V těchto třídách pak byly implementovány jen nejnutnější metody potřebné pro správné otestování manageru.

Samotná testovací třída pak ověřuje všechny public metody poskytované testovanou třídou[1]. Protože všechny služby poskytované `MinigameService` jsou implementovány na základě volání metod tohoto manageru, je tak jednotkově otestována i tato WCF služba. Vzhledem k tomu, že Minigame manager nejčastěji operuje s instancemi třídy `Minigame` včetně jejích potomků je pro tuto třídu napsána testovací třída `MinigameTest`. Tyto testy pak pokrývají všechny public metody poskytované testovanou třídou.

Poslední částí, která byla testována byla integrace do game serveru. Toto testování probíhalo víceméně ručně, kdy byly po spuštění game serveru postupně volány jednotlivé metody manageru a byly zkoumány jak návratové hodnoty příslušných metod, tak změny vlastností příslušných objektů. Obdobný postup byl pak proveden i k otestování `MinigameService`, kdy byly jednotlivé metody volány z *GameUI*.

8.1.3 Spouštění

Spouštění je poslední částí rozhraní a proto ani to nesmí zůstat bez ověření. Pro možnost otestování spouštění her byl vytvořen testovací kontroler `MinigameTest` a na game serveru byly vytvořeny popisy her pro obě

[1]Jediné dvě metody, které nemohly být otestovány v této testovací třídě jsou `updateLastRequestTime` a `loadAssets`. Tyto metody, resp. části z těl těchto metody jsou testovány v `MinigameTest` a `LogoQuizTest`

vytvořené minihry spouštěné na akci **TestAction**. Obě hry mají stejnou spouštěcí podmínku a to více než 100 kreditů. Jako typ odměny byly rovněž zvoleny kredity. URL pro zavolání testovacího kontroleru pak měla podobu **http://<server>/Game/MinigameTest/GameRequest?list=true**. Použití tohoto odkazu umožňuje při spouštění vybírat z obou miniher. Pokud je v adrese provedena změna hodnoty parametru **list** na **false**, je možné spustit pouze první minihru.

Na základě tohoto odkazu pak bylo testováno zobrazování dialogu pro spuštění miniher. Dále pak bylo zkoumáno, jestli se po vybrání minihry vytvoří správná instance hry na game serveru a také jestli po tomto kroku nebude dialog vyskakovat vícekrát. V případě vybrání webové hry pak také bylo otestováno, jestli dojde k otevření okna s přesměrováním na příslušnou URL. V případě externí hry pak bylo ověřováno, jestli se zobrazí dialog se správným ID hry a jestli je z tohoto dialogu umožněno stáhnout externího klienta.

Po integraci obou miniher bylo ještě otestováno zobrazování dialogu pro zamezení hraní hlavní hry, jeho zavření po ukončení hry a také ukončení minihry z tohoto dialogu. Toto testování spočívalo ve spuštění hry prostřednictvím výše zmíněného testovacího odkazu a v jejím odstartování z příslušného klienta.

Po startu hry bylo ověřeno zobrazení příslušného dialogu v hlavní hře a následně jeho automatické zavření po ukončení hry. Po opětovném vyvolání tohoto dialogu došlo k ukončení aktuálně rozehrané minihry a bylo testováno, jestli došlo k ukončení a odstranění hry v Minigame manageru. Všechny tři možnosti byly testovány pro webovou i externí minihru.

8.2 Testování miniher

Testování obou miniher můžeme rozdělit na 3 části. První částí je testování herní logiky na serveru. Druhou částí pak je ověřování správné reakce na požadavky volané přes webové služby a to především u externí minihry. Poslední část pak patří testování samotných funkčních vlastností her. Kromě automatických jednotkových testů, probíhaly všechny ostatní testy ručně, kde po každé akci docházelo ke kontrole změn buď v game serveru nebo v příslušném klientovi.

8.2.1 Spaceship Cargo Finder

Pro otestování serverové části této hry byly napsány jednotkové testy. Tyto testy tvoří třída `SpaceshipCargoFinderTest`. Protože `SpaceshipCargoFinder` dědí od třídy `Minigame`, na kterou již byly jednotkové testy napsány, jsou v testovací třídě ověřovány pouze metody z potomka. Jelikož komunikace skrze webové služby již z web serveru byla testována, byla druhá část testování zaměřena na ověření komunikace mezi klientem a webovým serverem prostřednictvím ajaxového systému zpráv.

Zde se ověřovalo přidávání hráče do hry a její odstartování, periodické requesty na update času posledního requestu a kontroly kolize a přičítání skóre. Dále pak byly testovány požadavky na ukončení hry a odměnění hráče. Poslední částí testování bylo ověření funkčních vlastností hry. V tomto případě byly ověřovány následující funkční vlastnosti:

- Spuštění hry a odpočítávání.
- Ovládání lodi kurzorovými klávesami.
- Kolize s lodí a herní oblastí.
- Kontrola křížení lodi a jízdy mimo herní oblast.
- Sbírání nákladu resp. přičítání skóre a prodlužování lodi.
- Ukončení hry (po zavření okna, po výhře, po prohře, z hlavní hry, po ukončení života hry, po objevení podvodu z game serveru).
- Pauza hry a její obnovení.
- Odměnění po výhře.
- Zobrazování a ukončování herních dialogů.

8.2.2 Logo Quiz

Stejně jako v předchozí případě byla pro testování serverové části vytvořena třída `LogoQuizTest` s jednotkovými testy. Tato třída ověřuje všechny public metody třídy `LogoQuiz` včetně načítání log. V druhé části, tedy ověřování správné komunikace skrze webovou službu Minigame service, byly testovány

požadavky na ověření hráče a přidání do hry, odstartování hry a požadavek na vygenerování otázek. Dále pak byly ověřovány požadavky na periodické requesty updatu času posledního requestu a vyhodnocení hry. Jako poslední bylo testováno odměnění hráče a také ukončení hry.

V poslední části testování této minihry byly ověřovány její funkční vlastnosti. Testování probíhalo v emulátoru s verzí Androidu 5.1. Mezi ověřované vlastnosti patří:

- Šifrování hesla.
- Přihlašování a spuštění hry.
- Ovládání hry (odpovídání na otázky).
- Spouštění časovače.
- Přehrávání zvuků.
- Pauza hry a její obnovení.
- Odměnění po výhře.
- Ukončení hry (na tlačítko domu, po výhře, po prohře, z hlavní hry, po ukončení života hry, po vypršení času).
- Dialog na tlačítko zpět.
- Zobrazování a ukončování herních dialogů.
- Reakce na ztrátu připojení k internetu.

9 Závěr

Hlavním cílem této práce bylo navrhnou a implementovat rozhraní umožňující integraci miniher do webové hry Space Traffic. Rozhraní mělo umožňovat integraci co možná největšího množství her různých žánrů a mělo být navrženo v souladu s herním designem a architekturou aplikace. Následně měly být na základě tohoto rozhraní implementovány a integrovány alespoň dvě minihry, které měly být následně otestovány. Pro dosažení tohoto cíle byly stanoveny tři menší cíle.

Prvním z těchto cílů byla analýza současného stavu projektu, principů SOA a problematiky webových her. Do tohoto cíle bylo také zahrnuto navržení rozhraní pro integraci her. Na základě analýzy tak byly stanoveny možnosti spouštění her, jejich popisování a odměňování a především možnosti integrace. Následně pak byly diskutovány důvodu pro zvolená řešení a nastíněn možný princip činnosti zvoleného řešení.

Druhým cílem pak byla implementace rozhraní dle návrhu a integrace dvou miniher. Pro realizaci rozhraní pak bylo nutné upravit všechny hlavní části architektury aplikace. Protože je do aplikace možné zakomponovat jak webové hry, tak hry realizované externími klienty, byly implementovány obě varianty.

Posledním cílem pak bylo otestování jak vytvořeného rozhraní, tak samotných miniher včetně jejich integrace. Všechny tři součásti této práce byly testovány jak v průběhu jejich realizace, tak po dokončení projektu. Důležité součásti rozhraní jsou pak testovány i jednotkovými testy.

Na základě splnění těchto jednotlivých cílů byl splněn i hlavní cíl práce. Díky této práci je tak možné do Space Trafficu integrovat minihry různých herních žánrů. Pro integraci miniher není primárně nutná znalost technologie Microsoft .NET a je tak možné realizaci her zadávat jako semestrální práce z různých předmětů vyučovaných na KIV.

Práci je možné dále rozšířit o spouštění her volně z menu hry, o možnost přidávání her za běhu systému či načítání popisu her z XML souborů. V dalších fázích vývoje projektu pak bude nutné doimplementovat další možnosti vyhodnocování podmínek a odměňování.

Seznam zkratek

.NET „dotnet“ pochází z anglického slova network a představuje soubor technologií, které tvoří platformu.

ADO.NET ActiveX Data Objects .NET představuje soubor komponent používaných pro přístup k datům nebo k datovým službám v různých datových zdrojích (součást Microsoft .NET Frameworku).

AES Advanced Encryption Standard je standardizovaný symetrický šifrovací algoritmus.

AJAX Asynchronous JavaScript and XML je webová technologie pro výměnu dat mezi klientem a serverem bez nutnosti znovunačítání stránky.

API Application Programming Interface označuje soubor funkcí a knihoven, které tvoří rozhraní pro programování aplikací.

ASP.NET Active Server Pages .NET představuje framework pro tvorbu webových stránek (součást Microsoft .NET Frameworku).

CMS Content Management System je systém zajišťující správu dokumentů (často nazývaný jako redakční systém).

CSS Cascading Style Sheets označuje standardizovaný jazyk pro definování zobrazování elementů např. na webových stránkách.

DAO Data Access Object je ve své podstatě objekt, který poskytuje abstraktní rozhraní pro přístup k databázovým entitám.

DHTML Dynamic HTML představuje soubor několika technologií používaných k tvorbě interaktivních webových stránek.

EDSAC Electronic Delay Storage Automatic Calculator byl první v praxi využitelný počítač s uloženým programem postavený na základě John Von Neumannova dokumentu First Draft of a Report on the EDVAC.

FAV Fakulta aplikovaných věd.

FIFA Fédération Internationale de Football Association je organizace řídící světový fotbal, po které je pojmenována fotbalová herní série.

GTA Grand Theft Auto je populární herní série společnosti Rockstar Games z gangsterského prostředí.

HTML HyperText Markup Language představuje standardizovaný značkovací jazyk pro tvorbu webových stránek.

HTTP Hypertext Transfer Protokol je základní aplikační protokol pro výměnu hypertextových dokumentů na webu.

JSON JavaScript Object Notation je datový formát nezávislý na platformě využívaný především v JavaScriptu.

KIV Katedra informatiky a výpočetní techniky.

MIT Massachusetts Institute of Technology označuje soukromou výzkumnou univerzitu v americké státě Massachusetts.

MMOFPS Massively-Multiplayer Online First Person Shooter je označení online hry pro více hráčů s FPS motivy (podrobněji v kapitole 2.4.2).

MMOG Massively-Multiplayer Online Game je obecné označení online hry pro více hráčů (více podrobností v kapitole 2.4.2).

MMORPG Massively-Multiplayer Online Role Playing Game představuje onlinovou hru pro více hráčů s RPG motivy (podrobněji v kapitole 2.4.2).

MMORTS Massively-Multiplayer Online Real-Time Strategy označuje online hru pro více hráčů s RTS prvky (podrobněji v kapitole 2.4.2).

MMOTPS Massively-Multiplayer Online Third Person Shooter je označení online hry pro více hráčů s TPS motivy (podrobněji v kapitole 2.4.2).

MUD Multi-User Dungeon označuje počítačovou hru na hrdiny pro více hráčů ovládanou textovým uživatelským rozhraním.

MVC Model-view-controller představuje aplikační architekturu oddělující datovou vrstvu, uživatelské rozhraní a aplikační logiku realizovanou tak, aby změny v jakékoliv z vrstev měly co nejmenší vliv na ostatní vrstvy.

NHL Nation Hockey League je kanadsko-americká hokejová soutěž, po které je pojmenována hokejová herní série.

PHP Hypertext Preprocessor je programovací jazyk, určený k programování dynamických webových aplikací.

REST Representational State Transfer je architektura rozhraní použitelného pro jednotný a snadný přístup ke zdrojům (např. datům).

RPG Role Playing Game představuje herní žánr, tzv. hru na hrdiny (více informací v kapitole 2.4.1).

SOAP Simple Object Access Protocol je protokol pro výměnu zpráv založených na XML používaný pro komunikaci s webovými službami.

SOA Service Oriented Architecture označuje architekturu orientovanou na služby (více v kapitole 4).

SQL Structured Query Language je standardizovaný strukturovaný dotazovací jazyk používaný v databázových systémech.

SŘBD Systém řízení báze dat označuje počítačový systém zabezpečující definování struktury dat, možnost manipulace s daty, ochranu dat a také komunikaci systému s uživatelem.

SSL Secure Sockets Layer je vrstva mezi transportní a aplikační vrstvou, která poskytuje šifrovanou komunikaci.

SVG Scalable Vector Graphics označuje značkovací jazyk a formát souboru popisující dvojrozměrnou vektorovou grafiku.

TCP Transmission Control Protocol je spolehlivý spojovaný komunikační protokol transportní vrstvy.

UI User Interface je anglické označení uživatelského rozhraní.

UML Unified Modeling Language představuje jednotný modelovací jazyk, který slouží k návrhu, specifikaci či vizualizaci softwarových systémů při jejich vývoji.

URI Uniform Resource Identifier je standardizovaný textový řetězec sloužící k přesné identifikaci zdroje informací.

URL Uniform Resource Locator je podmnožina URI, která často používá schémata jako http, mailto a podobně.

WFC Windows Communication Foundation je součást Microsoft .NET Frameworku, která představuje API a běhové prostředí pro tvorbu servisně orientovaných aplikací (více podrobností v kapitole 4.4).

WSDL Web Service Description Language označuje jazyk založený na XML používaný k popisu webových služeb.

WS Web Services neboli webové služby.

WebGL Web Graphics Library je API umožňující rendrovat 2D a především 3D grafiku v prohlížeči bez podpory různých pluginů.

XML Extensible Markup Language je obecný standard značkovacího jazyka.

ZČU Západočeská univerzita v Plzni.

Literatura

[Ara(2009)] ARAH, Tom. 99% Flash Player Penetration: Too Good to be True? In: *Alphr* [online]. 2009 [cit. 2016-04-25]. Dostupné z: `http://www.alphr.com/blogs/2009/02/20/99-percent-flash-player-penetration`

[Bhu(2011)] BHUVANESWARI, N. Sudha a S. SUJATHA. *Integrating SOA and Web Service*. Aalborg: Rivers publishers, 2011. ISBN 978-879-2329-653.

[Bor(2013)] BOŘÍK, Pavel. *Zajištění kvality webové hry Space Traffic*. Plzeň, 2013. Diplomová práce. Západočeská univerzita v Plzni. Vedoucí práce Ing. Petr Vaněček, Ph.D.

[Bro(2016)] Browser game. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-04-24]. Dostupné z: `https://en.wikipedia.org/wiki/Browser_game`

[Eds(2016)] EDSAC. *Encyklopedia Britannica: School and library subscribers* [online]. 2016 [cit. 2016-04-25]. Dostupné z: `http://www.britannica.com/technology/EDSAC`

[Erl(2009)] ERL, Thomas. *SOA: servisně orientovaná architektura : kompletní průvodce*. Brno: Computer Press, 2009. Programování (Computer Press). ISBN 978-80-251-1886-3.

[Fai(2001)] FAIRCLOUGH, Chris, et al. *Research directions for AI in computer games*. Trinity College Dublin, Department of Computer Science, 2001.

[Fla(2016)] Flash Games. *Google Trends* [online]. 2016 [cit. 2016-04-25]. Dostupné z: `https://www.google.cz/trends/explore#q=flash%20games`

[Ful(2008)] FULLERTON, Tracy., Christopher SWAIN a Steven HOFFMAN. *Game design workshop: a playcentric approach to creating innovative games*. 2nd ed. Boston: Elsevier Morgan Kaufmann, 2008. ISBN 02-408-0974-2.

[His(2016)] History of video games. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-04-25]. Dostupné z: https://en.wikipedia.org/wiki/History_of_video_games

[Htm(2016)] HTML5 games. *Google Trends* [online]. 2016 [cit. 2016-04-25]. Dostupné z: https://www.google.cz/trends/explore#q=html5%20games

[Koc(2012)] KOCMAN, Richard. *Projekt implementace a provozu webové hry Space Traffic*. Plzeň, 2012. Diplomová práce. Západočeská univerzita v Plzni. Vedoucí práce doc. Ing. Přemysl Brada, MSc. PhD.

[Kso(2015)] *Ksoap2-android Project* [online]. 2015 [cit. 2016-04-30]. Dostupné z: http://simpligility.github.io/ksoap2-android/index.html

[Log(2016)] Logo Quiz Help. *Answers* [online]. 2016 [cit. 2016-05-29]. Dostupné z: http://logosquiz.info/cz/logoquiz/

[Mic(2015)] MICHL, Petr. Firefox začal automaticky blokovat Flash. Konec mu věští i ve Facebooku. Vzdá se ale tak lehce? In: *Marketing journal.cz* [online]. 2015 [cit. 2016-04-25]. Dostupné z: http://www.m-journal.cz/cs/aktuality/firefox-zacal-automaticky-blokovat-flash--konec-mu-vesti-i-ve-facebooku--vzda-se-ale-tak-lehce-_s288x11463.html

[Min(2016)] Minigame. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-04-25]. Dostupné z: https://en.wikipedia.org/wiki/Minigame

[Mud(2016)] MUD. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-04-25]. Dostupné z: https://en.wikipedia.org/wiki/MUD

[Onl(2016)] Online game. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-04-25]. Dostupné z: https://en.wikipedia.org/wiki/Online_game

[Puk(2009)] PUKLEK, Davor. History of Web Browser Games. In: *Ezine articles* [online]. 2009 [cit. 2016-04-25]. Dostupné z: http://ezinearticles.com/?History-of-Web-Browser-Games&id=2670093

[Rec(2016)] Recent Files. *The Ultimate Site* [online]. 2016 [cit. 2016-04-29]. Dostupné z: http://www.tus-wa.com/files/

[Ser(2016)] Service Oriented Architecture. *IBM.com* [online]. 2016 [cit. 2016-04-27]. Dostupné z: http://www-01.ibm.com/software/solutions/soa/glossary/

[Sha(2010)] SHARP, John. *Microsoft Visual C# 2010: krok za krokem*. Brno: Computer Press, 2010. Krok za krokem (Computer Press). ISBN 978-80-251-3147-3.

[Sta(2015)] ŠTĚPÁNEK, Martin, KOTALÍK, Jan (ed.). Architecture overview. In: *Space Traffic Development* [online]. 2015 [cit. 2016-04-26]. Dostupné z: http://spacetraffic.kiv.zcu.cz/code/tiki-index.php?page=Architecture%20overview

[Stc(2015)] ŠTĚPÁNEK, Martin, KOTALÍK, Jan (ed.). Client. In: *Space Traffic Development* [online]. 2015 [cit. 2016-04-26]. Dostupné z: http://spacetraffic.kiv.zcu.cz/code/tiki-index.php?page=Client

[Ste(2012)] ŠTĚPÁNEK, Martin. *Architektura a implementace webové hry pro více hráčů*. Plzeň, 2012. Diplomová práce. Západočeská univerzita v Plzni. Vedoucí práce doc. Ing. Přemysl Brada, MSc. PhD.

[Stg(2015)] ŠTĚPÁNEK, Martin. Game Server. In: *Space Traffic Development* [online]. 2015 [cit. 2016-04-26]. Dostupné z: http://spacetraffic.kiv.zcu.cz/code/tiki-index.php?page=Game%20Server

[Stm(2015)] ŠTĚPÁNEK, Martin, KOTALÍK, Jan (ed.). Minihry. In: *Space Traffic Development* [online]. 2015 [cit. 2016-04-28]. Dostupné z: http://spacetraffic.kiv.zcu.cz/code/tiki-index.php?page=Minihry

[Sts(2015)] ŠTĚPÁNEK, Martin, SOBĚHART, Michel (ed.). Assets. In: *Space Traffic Development* [online]. 2015 [cit. 2016-04-26]. Dostupné z: http://spacetraffic.kiv.zcu.cz/code/tiki-index.php?page=Assets

[Stw(2015)] ŠTĚPÁNEK, Martin, KOTALÍK, Jan (ed.). Web Server. In: *Space Traffic Development* [online]. 2015 [cit. 2016-04-26]. Dostupné z: http://spacetraffic.kiv.zcu.cz/code/tiki-index.php?page=Web%20Server

[Tis(2011)] TIŠNOVSKÝ, Pavel. Historie vývoje počítačových her (1.část - první milníky). In: *Root.cz* [online]. 2011 [cit. 2016-04-25]. Dostupné z: http://www.root.cz/clanky/historie-vyvoje-pocitacovych-her-1-cast-prvni-milniky/

[Vir(2002)] VIRIUS, Miroslav. *C# pro zelenáče*. Praha: Neocortex, 2002. Bestseller for all. ISBN 80-863-3011-7.

[Vlč(2015)] VLČEK, Lukáš, SOBĚHART, Michel (ed.). Persistence Layer. In: *Space Traffic Development* [online]. 2015 [cit. 2016-04-26]. Dostupné z: http://spacetraffic.kiv.zcu.cz/code/tiki-index.php?page=Persistence+Layer

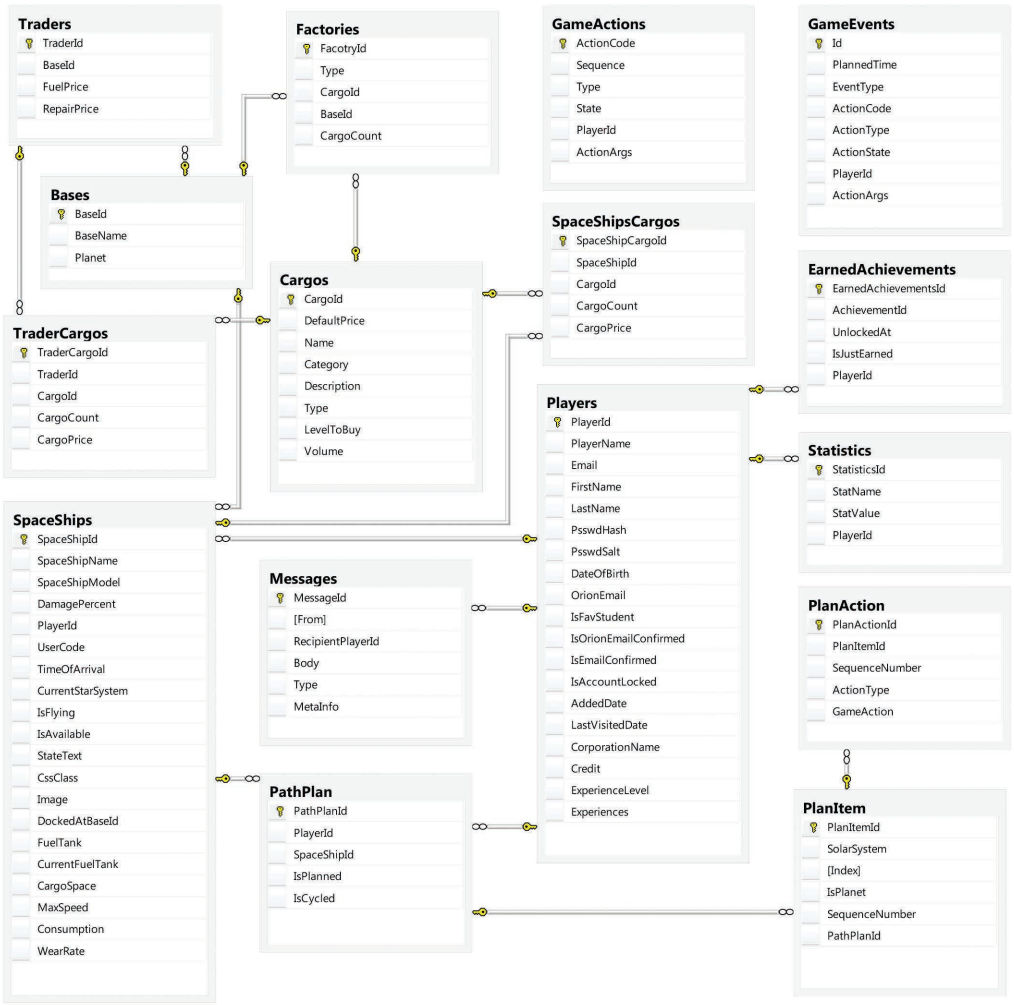
[Vog(2016)] VOGEL, Pert, ŠVECOVÁ, Veronika (ed.). O projektu. In: *Space Traffic Development* [online]. 2016 [cit. 2016-04-26]. Dostupné z: http://spacetraffic.kiv.zcu.cz/code/tiki-index.php?page=O+projektu

[Wha(2016)] What Is Windows Communication Foundation. *Microsoft Developer Network* [online]. 2016 [cit. 2016-04-27]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms731082.aspx

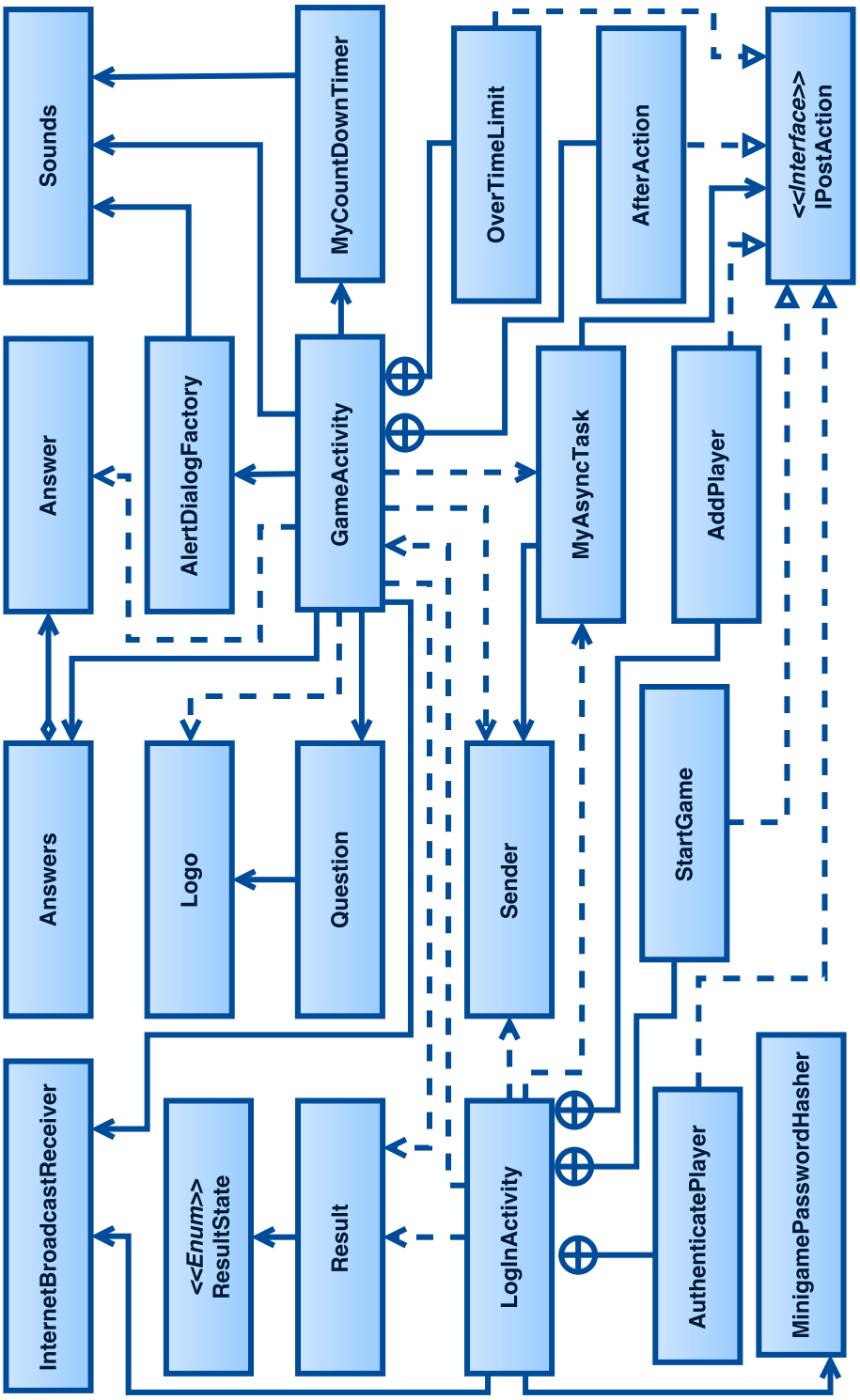
[Wsd(2016)] *Wsdl2Code* [online]. 2016 [cit. 2016-04-30]. Dostupné z: http://www.wsdl2code.com/pages/home.aspx

Seznam příloh

A Relační model databáze

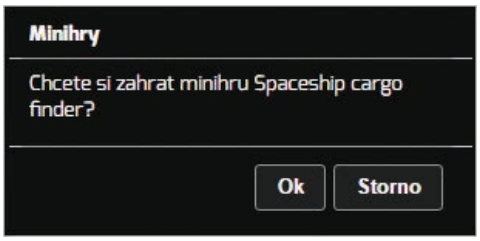


B Diagram tříd pro Logo Quiz

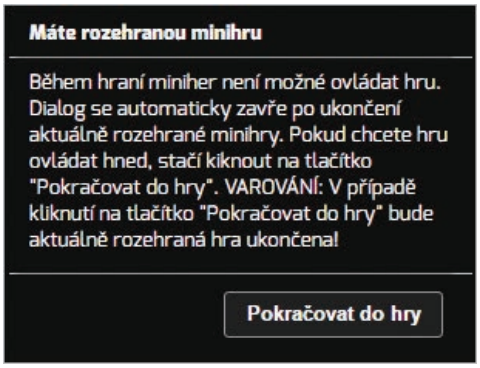


C Ukázka spouštění miniher

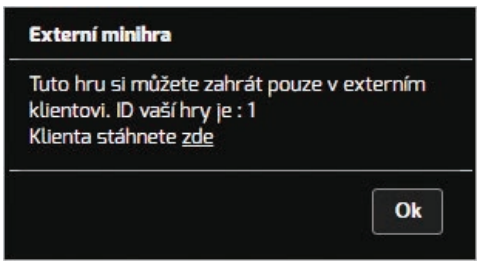
Obrázek C.1: Dialog pro spuštění miniher – výběr z několika her.



Obrázek C.2: Dialog pro spuštění miniher – jedna hra.



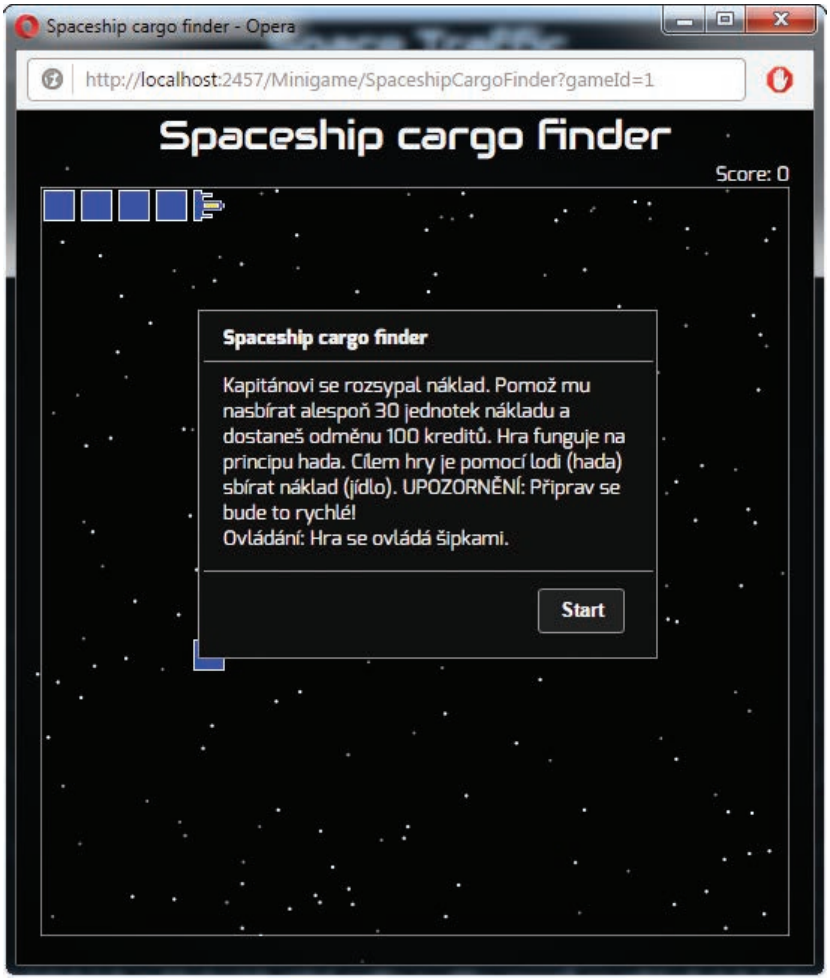
Obrázek C.3: Dialog v hlavní hře o rozehrané minihře.



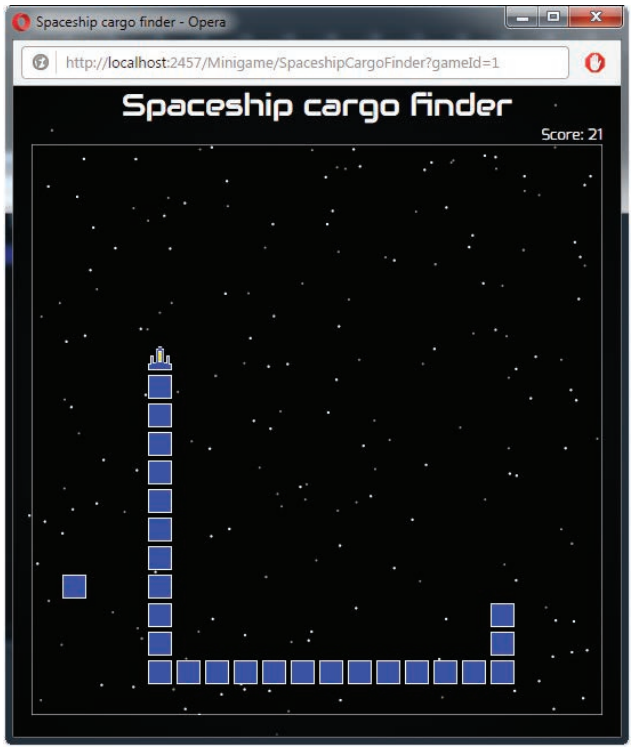
Obrázek C.4: Dialog s informacemi pro externí minihru.

D Ukázka miniher

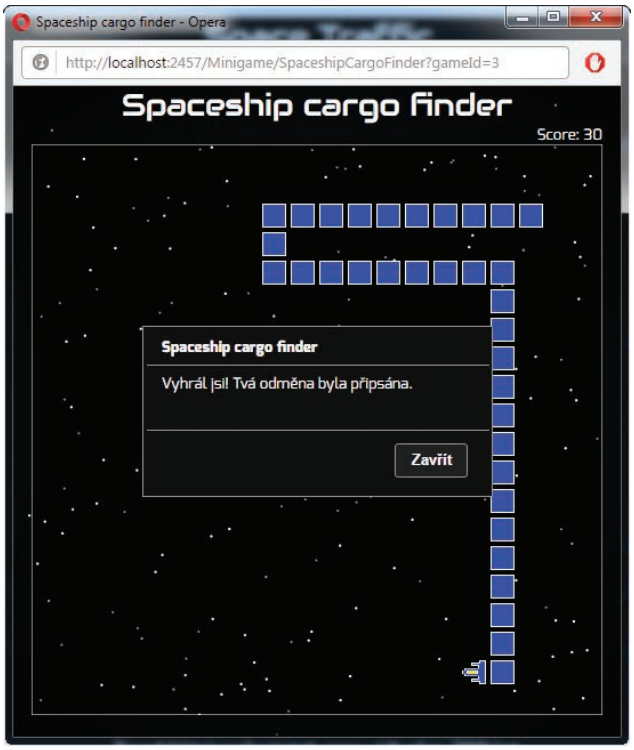
D.1 Spaceship Cargo Finder



Obrázek D.1: Úvodní spuštění.

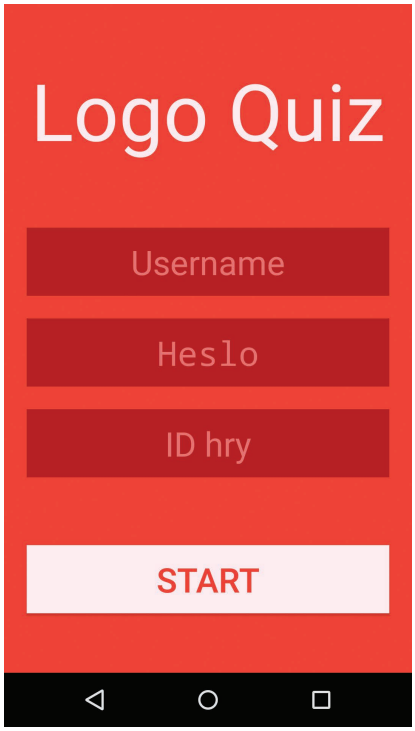


Obrázek D.2: Průběh hry.

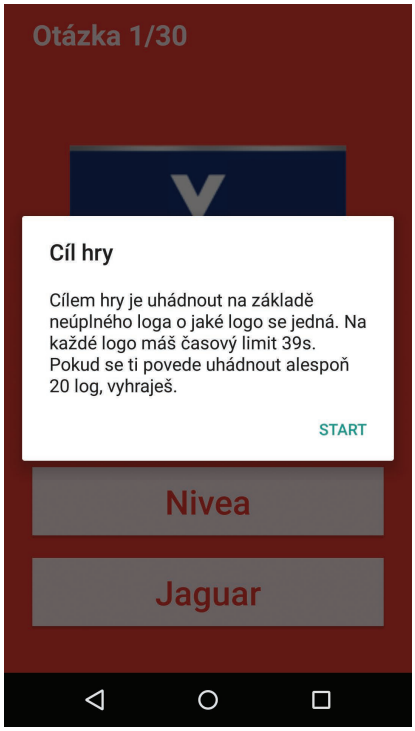


Obrázek D.3: Úspěšné dokončení hry.

D.2 Logo Quiz

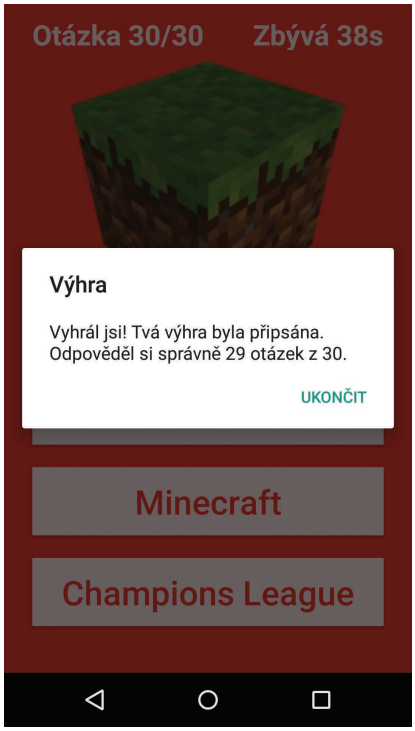


Obrázek D.4: Přihlášení.



Obrázek D.5: Začátek hry.

Obrázek D.6: Průběh hry.



Obrázek D.7: Vyhraná hra.

E Uživatelská dokumentace

E.1 Překlad a spuštění

E.1.1 Potřebné nástroje

Pro překlad a spuštění aplikace (včetně minihry *Logo Quiz*) je **nutné** mít nainstalované následující nástroje:

- Microsoft .NET Framework 4 nebo novější[1]
- Microsoft Visual Studio 2013 nebo novější, v edici Professional, Premium nebo Ultimate[2]
- Microsoft SQL Server 2012 nebo novější
- ASP.NET MVC 3
- Java SE JDK verze 1.8 nebo novější
- Android Studio 2.0 nebo novější

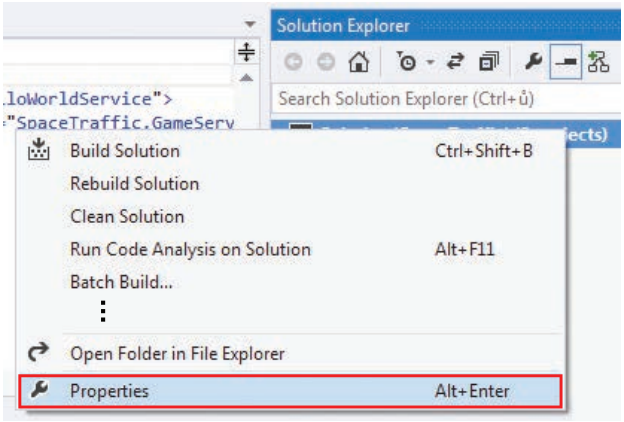
V případě SQL serveru může být instalace poněkud složitější a proto je potřeba postupovat dle obrázkového návodu uvedeného na stránce `http://spacetraffic.kiv.zcu.cz/code/tiki-index.php?page=Návod+na+instalaci+SQL+Serveru`.

E.1.2 Překlad a spuštění hlavní hry

Překlad a spuštění projektu se provádí pomocí Visual Studia (VS). Nejprve spustíme VS jako administrátor. Ve VS pak spustíme soubor `SpaceTraffic.sln` ze složky `Aplikace\SpaceTraffic`. Následně je potřeba v Properties daného Solution nastavit spouštění více projektů. Nejprve tedy klikneme pravým tlačítkem myši na **Solution 'SpaceTraffic'** v **Solution Exploreru** a následně vybereme **Properties** (obdobně jako na obrázku E.1).

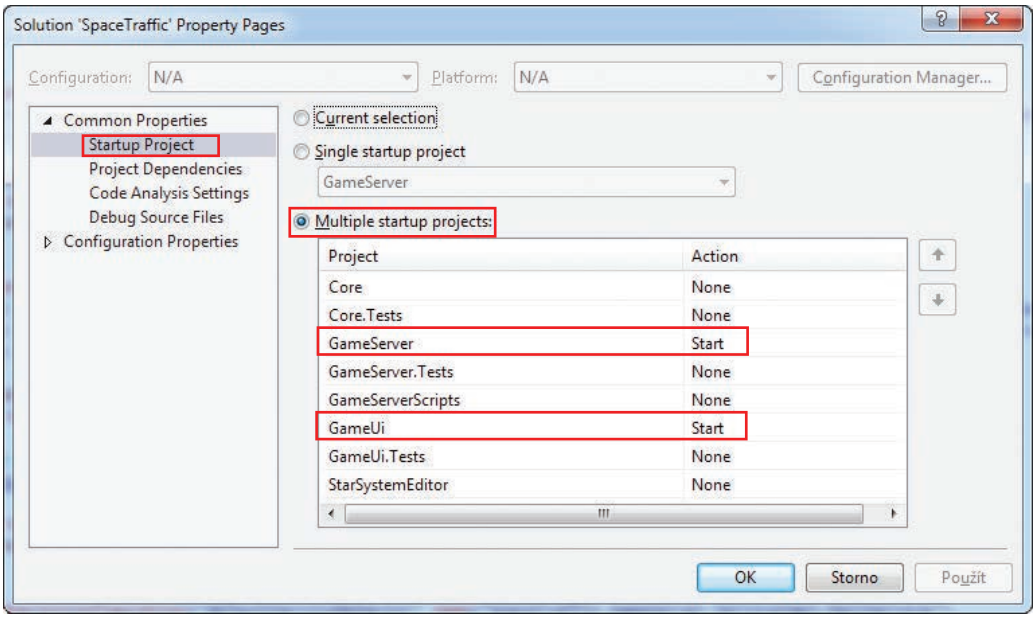
[1] Měl by se nainstalovat společně s Visual Studiem.

[2] V případně Visual Studia 2015 některou ze srovnatelných edicí.



Obrázek E.1: Properties Solution 'SpaceTraffic'.

V následujícím okně pak vybereme v záložce **Startup Project** volbu **Multiple startup projects**. Následně u projektů `GameServer` a `GameUI` nastavíme volbu **Start**. Vše můžeme vidět na obrázku E.2.



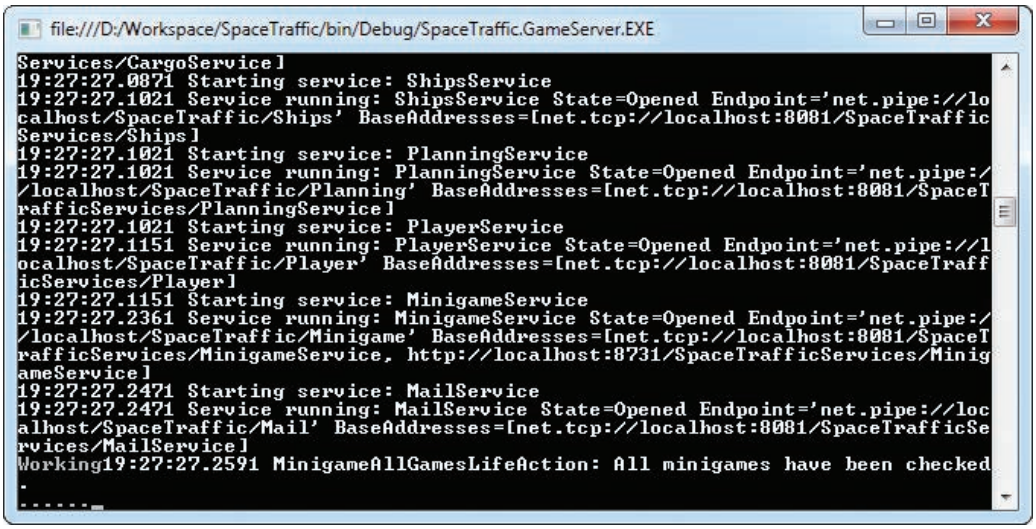
Obrázek E.2: Nastavení spouštění více projektů.

Takto nastavený projekt následně přeložíme. To se provádí v hlavním menu VS v záložce **Build** výběrem volby **Build solution** nebo klávesovou zkratkou `CTRL + SHIFT + B`. Po úspěšném přeložení provedeme spuštění projektu buď z menu hry kliknutím na **Start Debugging** v záložce **Debug** nebo klávesovou zkratkou `F5`. Následně se spustí konzole herního serveru a samotná aplikace ve výchozím prohlížeči.

Pro přihlášení je připraven testovací uživatel:

login: user
heslo: user

Před přihlášením je **nutné** počkat na spuštění jednotlivých WFC služeb. Jestli byly služby spuštěny se pozná podle konzole game serveru, kde bude na konci hláška `Working` a budou se vypisovat tečky (viz obrázek E.3). Případné možné problémy a jejich řešení je popsáno na stránce `http://spacetraffic.kiv.zcu.cz/code/tiki-index.php?page=Časté+chyby+a+jejich+řešení`.



Obrázek E.3: Konzole game serveru po spuštění WCF služeb.

E.1.3 Překlad a spuštění Logo Quizu

Nejprve spustíme Android Studio (AS). Na uvítací obrazovce vybereme **Open an existing project** a zvolíme cestu buď do složky `aplikace\LogoQuiz\src` a nebo do `aplikace\SpaceTraffic\AndroidMinigames\LogoQuiz`. Oba projekty jsou totožné, takže je jedno, který bude zvolen. Po spuštění projektu vybereme v hlavním menu AS položku **Build** a následně **Make Project**.

Po úspěšném překladu musíme ještě aplikaci spustit na emulátoru. To se provede kliknutím na položku **Run** v hlavním menu AS a následně na **Run 'app'**. V následujícím okně buď vybereme příslušný emulátor nebo ho přes tlačítko **Create new emulátor** s pomocí průvodce vytvoříme. Podmínkou pro emulátor je minimální verze Androidu 4.4 resp. API level 19.

E.2 Spouštění miniher

Pro spouštění miniher je připraven testovací kontroler. Pro jeho volání je potřeba v prohlížeči zavolat URL adresu:

`http://localhost:2457/Game/MinigameTest/GameRequest?list=true`

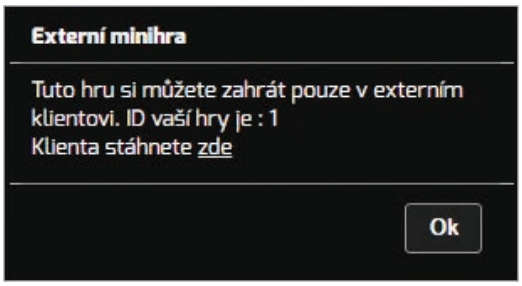
Na základě tohoto volání se zobrazí dialogové okno jako na obrázku E.4. Zde si pak můžeme vybrat minihru a potvrdit tlačítkem **Ok** nebo zrušit tlačítkem **Storno**. Pokud bude v uvedené adrese změněn parametr *list* na **false**, bude možné vybrat pouze jednu minihru (momentálně nakonfigurováno na *Spaceship Cargo Finder*). V případě spuštění *Spaceship Cargo Finder* se otevře nové okno, kde se postupuje pomocí instrukcí hry. V případě spuštění *Logo Quizu* se postupuje podle následující kapitoly.

Obrázek E.4: Dialog pro spouštění her.

E.2.1 Logo Quiz

V případě výběru minihry *Logo Quiz* (jako externí hry) bude zobrazen dialog jako na obrázku E.5. Kde je ID hry a také odkaz ke stažení externího klienta. Spuštění je pak možné dvěma způsoby. První je popsáno v kapitole E.1.3. Druhým způsobem je stáhnout si z uvedeného odkazu příslušný *apk* soubor a nainstalovat ho do emulátoru. To se provádí následujícím způsobem.

Ze složky, kde je nainstalované SDK androidu (např. `D:\Programy\Android\sdk`) spustíme program **AVD Manager.exe**. V něm pak vybereme příslušný emulátor (případně tlačítkem **Create** vytvoříme nový) a spustíme tlačítkem **Start**. Pro emulátor musí platit stejná pravidla jaká byla popsána v kapitole E.1.3. Po spuštění emulátoru si spustíme příkazový řádek ve složce `platform-tools`, která je v adresáři, kde máme nainstalované SDK androidu (např. `D:\Programy\Android\sdk\platform-tools`).

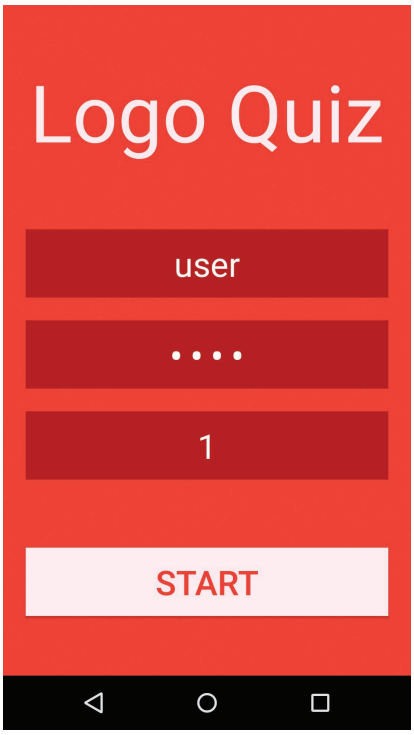


Obrázek E.5: Dialog po spuštění externí minihry.

Pro nainstalování *apk* souboru pak do konzole zadáme příkaz

```
adb install <path_to_apk>
```

kde `<path_to_apk>` je cesta k *apk* souboru[3]. Následně stačí v emulátoru spustit aplikaci **LogoQuiz**. V této aplikaci je potřeba vyplnit *login* a *heslo* testovacího uživatele (údaje z kapitoly E.1.2) a *ID hry* uvedené v dialogu po spuštění hry (obrázek E.5). V aplikaci jsou předpřipraveny údaje pro testovacího hráče a hru s ID 1, jak můžeme vidět na obrázku E.6.



Obrázek E.6: Úvodní obrazovka *Logo Quizu*.

[3]Tento způsob bude fungovat pouze pro jeden spuštěný emulátor, jinak je potřeba pomocí přepínače `-e` specifikovat název zařízení.

Pak už jen stačí spustit hru kliknutím na tlačítko **Start** a řídit se pokyny hry. Spustit hru oběma způsoby je možné nejdéle 30 minut od jejího vytvoření z hlavní hry. Zde je nutné ještě podotknout, že aplikace je přizpůsobena ke komunikaci pouze z emulátoru a není možné ji v aktuálním stavu použít ve fyzickém zařízení.

E.3 Integrace miniher

Pro integraci miniher je potřeba uvažovat rozdělení klienta na *webového* a *externího* a kontroly na *game serveru* a *mimo game server*. Pro integraci minihry je pak potřeba provést následující body:

- **Popsání hry a registrace**
- **Vytvoření třídy minihry** – Není nutné pokud bude herní logika (kontroly) probíhat v klientovi.
- **Přidání volání metod pro vyhodnocování podmínek do příslušných akcí v daných kontrolerech** – Není nutné pokud je hra spouštěna na akci, která již volání metody obsahuje.
- **Vytvoření klienta** – To spočívá především ve volání služeb (metod), tak aby hra procházela správně životním cyklem.

E.3.1 Popis hry a registrace

Každou hru je potřeba popsat a především specifikovat na jaké akce a podmínky je možné ji spouštět. Pro popis hry slouží třída `Minigame Descriptor`. Ukázka použití je ve třídě `GameServer` v regionu `minigame test data`. Všechny atribut kromě `SpecificReward`, `ConditionArgs` a `MinigameClass FullName` jsou povinné. V současné době je možné jako typ odměňování použít pouze kredity a zkušenosti a jako typ podmínky pro spuštění kredity a úroveň hráče. Pro případ použití jiného odměňování je potřeba rozšířit třídu `Rewarder` a pro jiné podmínky třídu `CoditionSolver`. Současně s tím i jejich kontroly `RewardChecker` a `ConditionChecker`.

V případě `SpecificReward` lze aktuálně nastavit pouze hodnotu **null**. Pro rozšíření je potřeba doimplementovat příslušné metody v již zmíněných

třídách. Do `ConditionArgs` lze v současné době uvést pouze číselnou hodnotu, která vyjadřuje minimální level nebo minimální množství peněz pro spuštění hry. Pro rozšíření platí stejná pravidla jako v předchozím případě.

Pokud je `ExternalClient` nastaven na **true**, uvádí se do `ClientURL` pouze název klienta. Tento klient pak musí být uložen v `GameUi\Content \minigames`. Pokud je však nastaven na **false**, uvádí se do tohoto atributu název kontroleru, který je po spuštění hry zavolán. `MinigameClassFullName` je buď **null**, pro použití instance `Minigame` jako třídy minihry na game serveru, nebo je uvedeno *AssemblyQualifiedName* příslušné třídy jejíž instance se má vytvořit (více viz další kapitola).

Popis hry je pak **nutné** ještě **zaregistrovat** pomocí `MinigameManageru` (ukázka rovněž v GameServeru). Pokud chceme použít jiné akce než ty, které jsou definovány (v současné době pouze `TestAction`) je potřeba je vložit do databáze ještě před registrací hry, která tyto akce bude využívat. Jak na to je ukázáno u testovací akce. Následně je v příslušném kontroleru a dané metodě potřeba zavolat metodu pro vyhodnocení miniher s názvem požadované akce (více viz kapitola E.3.3).

E.3.2 Vytvoření třídy minihry

Pokud chceme mít herní logiku na game serveru nebo alespoň její část, je potřeba vytvořit příslušnou třídu. Pro vytvoření třídy minihry je možné použít dva způsoby. Prvním ze způsobů je oddědění od třídy `Minigame`. Tím je zajištěno, že základní metody budou implementovány a bude stačit ve třídě potomka pouze doimplementovat potřebné atributy a metody.

Druhým způsobem je implementovat rozhraní `IMinigame`, kde je potřeba doimplementovat i základní metody. Pro vytvořenou třídu je pak nutné při popisu hry uvést do `MinigameClassFullName` její *AssemblyQualifiedName*. To je z toho důvodu, aby bylo možné vytvářet instance miniher i z jiných projektů (assembly) než z **GameServeru**.

E.3.3 Spouštění her

Pro umožnění spouštění her je potřeba v metodě daného kontroleru zavolat jednu z metod `evaluateMinigameByStartActionName` a `evaluateMinigame`

`ListByStartActionName`. Jako argument se pak uvádí název příslušné akce. První z metod slouží k umožnění nabídnout hráči hrát pouze jednu z her, která bude vyhodnocena jako spustitelná (první hra ze seznamu her pro danou akci, která je vyhodnocena jako spustitelná).

Druhá metoda pak dává hráči vybrat ze všech vyhodnocených her. O zbytek se pak postará rozhraní. Příklad je možné vidět v kontroleru `MinigameTestController` a metodě `GameRequest`. Pokud bude příslušná akce volání jedné z metod již obsahovat, není třeba tento krok provádět.

E.3.4 Webový klient

Pro vytvoření webového klienta je přepraven namespace `Areas.Minigame`, který má namapovánu routu na adresu:

<server>/Minigame/<controller>

kde **<server>** je adresa serveru a **<controller>** je příslušný kontroler dané minihry. Pokud tedy v tomto namespacu vytvoříme kontroler pro minihru a jeho view (model nemusí být potřeba) a příslušný kontroler uvedeme v popisu hry jako `ClientURL`, dojde po spuštění hry k otevření nového okna, které bude na daný kontroler přesměrováno. Od přesměrování na kontroler už si musí vše zajišťovat programátor sám.

První, co je potřeba zajistit, je změna velikosti okna pomocí JavaScriptu. Příklad je možné najít ve view *Spaceship Cargo Finder*. Druhým, co je potřeba udělat, je svázat událost zavření okna s ukončením a odebráním hry. Protože prohlížeče Firefox a Internet Explorer nedovolují vykonat asynchronní volání ajaxem na zavření okna, není možné použít ajaxový systém zpráv a musí být použit **synchronní** ajaxový request. Ukázku najdeme opět ve view, kde se provádí request na metodu kontroleru `EndGame`.

Ostatní věci už je možné zajišťovat libovolně. Pro komunikaci klienta s web serverem je možné využít ajaxový systém zpráv a pro komunikaci z webového na herní server pak v handlerech stačí využít `GSClient` a příslušné metody poskytované `MinigameService`. Použití systému zpráv je možné najít na adrese `http://spacetraffic.kiv.zcu.cz/code/tiki-index.php?page=AJAX`.

Jako příklad webové hry je implementován již zmíněný *Spaceship Cargo Finder*, kde jsou využity featury popsané v předchozím odstavci. Ta se skládá z již zmíněného controlleru a view a javascriptových tříd v `GameUi\JS\minigame\SpaceshipCargoFinder`. Více viz kapitola 7.1.

E.3.5 Externí klient

Externího klienta je možné realizovat prakticky v jakémkoliv jazyce, nutnou podmínkou je umožnění komunikace přes webové služby s game serverem. Komunikace je umožněna prostřednictvím SOAPu přes `MinigameService`. Pro externí klienty je tato služba dostupná na localhostu adrese `http://localhost:8080/SpaceTraffic/Minigame`[4]. Jaké jednotlivé kroky je potřeba provádět je popsáno v následující kapitole.

Nutné je v každé hře nejprve ověřit hráčovu identitu prostřednictvím volání metody `authenticatePlayerForMinigame`. Heslo je potřeba zašifrovat prostřednictvím algoritmu *AES*. Příklad jak šifrovat je možné najít v *Logo Quizu* ve třídě `MinigamePasswordHasher`, kde jsou i jednotlivé inicializační parametry hashování (`SALT`, `INIT_VECTOR`, `PDKBF2_PASSWORD`). Případně je možné nahlédnout i do třídy `MinigamePasswordHasher` v `Core` herního serveru.

Stejně jako u webového klienta je potřeba ošetřit ukončení hry na serveru, pokud dojde k ukončení klienta. Tato realizace je pak závislá na příslušné realizaci klienta a je tak zcela v intervenci programátora. Jak realizovat externího klienta v podobě Android aplikace můžeme vidět na hře *Logo Quiz*, jejíž projekt se nachází v `AndroidMinigames\LogoQuiz`. Zde je ke komunikaci použita knihovna **ksoap2**. Více o této minihře je v kapitole 7.2 a překlad a spuštění je v kapitole E.1.3.

Výsledného klienta je pak potřeba umístit do složky `GameUi\Content\minigames` a jeho název uvést v popisu hry do `ClientURL` včetně přípony. Zároveň je nutné v popisu hry nastavit atribut `ExternalClient` na **true**. V případě, že se bude jednat o jiného klienta než pro Android, je potřeba rozšířit metodu `DownloadGame` v controlleru `DefaultController` v minihrách o příslušný typ souboru (**Content-type**), aby bylo možné klienta stáhnout.

[4]Po stažení projektu z gitu je **nutné** odkomentovat end pointy pro http binding pro `MinigameService` v souboru **App.config** `GameServeru`.

E.3.6 Komunikace s game serverem

Pro komunikaci s game serverem je vytvořena `MinigameService`, jejíž služby zajišťuje `MinigameManger`. Z webového serveru zajišťuje komunikaci `GSClient`, který je součástí `AbstractControlleru`. Z externích klientů si komunikaci musí zajistit tvůrce klienta (více v předchozí kapitole). Důležité při komunikaci je dodržet životní cyklus hry znázorněný na obrázku 6.4. Jednotlivé kroky pro dodržení tohoto postupu jsou následující:

1. **Vytvoření hry** – Toto zajišťuje rozhraní a není potřeba se o něj starat.
2. **Inicializační akce** – Pokud je potřeba provést nějaké inicializace, je to možné provést voláním metody `performAction` (ukázka ve *Spaceship Cargo Finder* – volání `getGameInfo`). Tuto akci je možné provádět až do startu hry resp. i po něm, ale je vhodné ji využít na tomto místě. Tento krok ovšem není nutné provádět.
3. **Ověření hráče** – V případě externích miniher je potřeba ověřit hráče pomocí metody `authenticatePlayerForMinigame` jinak je možné tento krok přeskočit.
4. **Přidávání hráčů** – Přidávání hráčů se provádí metodou `addPlayer`. Pro více hráčů se pak provádí tento krok vícekrát.
5. **Start hry** – Start hry se provádí voláním metody `startGame` a je nutné ho provést nejpozději do 30 minut od vytvoření hry. To je ale spíše úkol hráče než programátora.
6. **Inicializace odesílání pravidelných requestů** – Od startu hry je potřeba nejpozději každou minutu odeslat request o tom, že hra žije a zároveň je tak možné zkontrolovat, jestli hráč hru neukončil prostřednictvím hlavní hry. K tomu se využívá metoda `checkMinigameLifeAndUpdate LastRequestTime` případně `updateLastRequestTime`, která ale kontrolu života hry neprovádí. Tyto requesty je pak nutné odesílat až do posledního kroku.
7. **Provádění akcí** – Během hry je možné provádět různé akce v instanci minihry na herním serveru prostřednictvím volání metody `perform Action`, případně `performActionWithLock`. Ta zajišťuje ošetření souběžného přístupu více hráčů k jedné instanci hry. V případě problémů s interoperabilitou u těchto metod, je možné doimplementovat si supportní metody do `MinigameService`, ze kterých pak budou tyto metody

volány (jako příklad slouží metoda `checkAnswersSupportMethod` pro *Logo Quiz*). Tento krok je možné libovolně opakovat. Pokud herní logika (kontroly) neprobíhá na serveru, je možné tento krok přeskočit.

8. **Ukončení hry** – Ukončení hry se provádí metodou `endGame`. Tento krok je možné využít už od vytvoření hry např. pro ukončení po zavření klienta a v tomto případě pak musí následovat i odstranění hry.

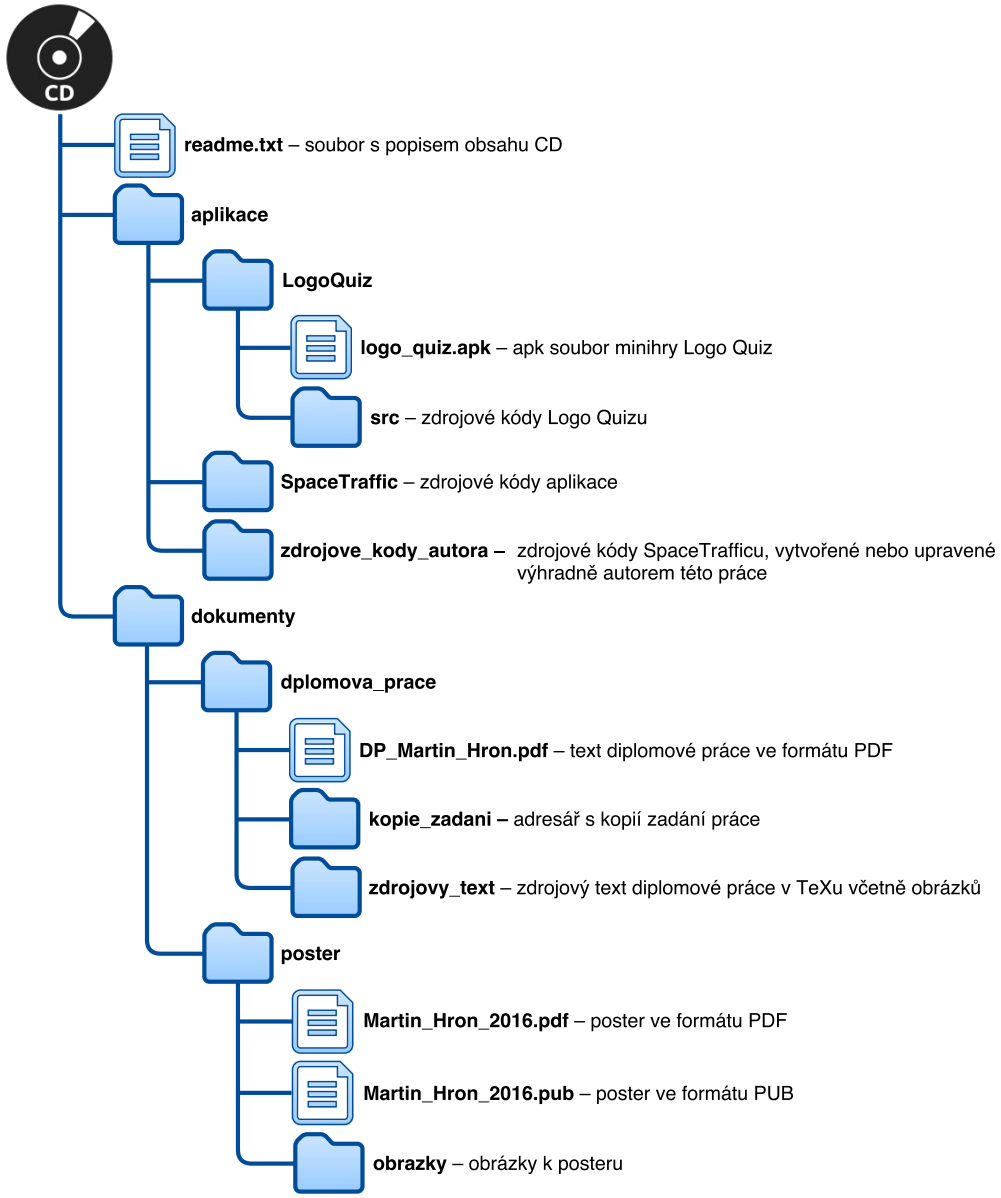
9. **Odměnění hráče** – Pokud hráč dosáhl cíleného výsledku je možné ho odměnit metodou `rewardPlayer`, případně více hráčů na základě volání metody `rewardPlayers`. Pokud hráč nevyhrál, je možné tento krok přeskočit.

10. **Ukončovací akce** – Ukončovací akce je možné provádět od ukončení hry až do odstranění a to prostřednictvím metody `performAction`. Stejně jako inicializační akce, není nutné tento krok provádět.

11. **Odstranění hry** – Odstranění hry se provádí metodou `removeGame`. Po tomto kroku už není potřeba odesílat periodické requesty.

Všechny metody `MinigameService` jsou popsány v rozhraní (kontraktu) `IMinigameService`. V případě inicializačních a deinicializačních akcí je **nutné** tyto akce doimplementovat do třídy, která bude představovat instance hry a ty se pak volají na základě `performAction`.

F Obsah CD


CD
readme.txt – soubor s popisem obsahu CD
aplikace
LogoQuiz
logo_quiz.apk – apk soubor minihry Logo Quiz
src – zdrojové kódy Logo Quizu
SpaceTraffic – zdrojové kódy aplikace
zdrojove_kody_autora – zdrojové kódy SpaceTrafficu, vytvořené nebo upravené výhradně autorem této práce
dokumenty
dplomova_prace
DP_Martin_Hron.pdf – text diplomové práce ve formátu PDF
kopie_zadani – adresář s kopií zadání práce
zdrojovy_text – zdrojový text diplomové práce v TeXu včetně obrázků
poster
Martin_Hron_2016.pdf – poster ve formátu PDF
Martin_Hron_2016.pub – poster ve formátu PUB
obrazky – obrázky k posteru