UNIVERZITA PARDUBICE
DOPRAVNÍ FAKULTA JANA PERNERA

ONLINE REŽIM PRO MOBILNÍ HRU S EKOLOGICKÝM MOTIVEM

Bc. Jakub Rykr

Diplomová práce
2013

Univerzita Pardubice
Dopravní fakulta Jana Pernera
Akademický rok: **2012/2013**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jakub Rykr**
Osobní číslo:	**D11862**
Studijní program:	**N3708 Dopravní inženýrství a spoje**
Studijní obor:	**Aplikovaná informatika v dopravě**
Název tématu:	**Online režim pro mobilní hru s ekologickým motivem**
Zadávající katedra:	**Katedra informatiky v dopravě**

Zásady pro vypracování:

Vytvoření pravidel pro online režim.
Návrh a vytvoření databáze.
Naprogramování služeb pro komunikaci s databází.
Realizace využívá technologií Microsoft XNA, WCF.
Tématicky je hra zasazena do reálného světa a motiv představuje ekologickou stopu.

Rozsah grafických prací:

Rozsah pracovní zprávy: **40 normostran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **REED, Aaron. Learning XNA 4.0. 1. ed. Sebastopol, CA: O'Reilly. ISBN 14-493-9462-0.**
2. **GARCIA-MOLINA, Hector. Database Systems: The Complete Book. 1. ed. New Jersey: Prentice Hall, 2002, 1119 s. ISBN 01-303-1995-3.**
3. **CONOLLY, Thomas, Carolyn E BEGG a Richard HOLOWCZAK. Mistrovství - databáze: profesionální průvodce tvorbou efektivních databází. Vyd. 1. Brno: Computer Press, 2002, 1119 s. ISBN 978-80-251-2328-7.**
4. **LÖWY, Juval. Programming WCF services. 3rd ed. Sebastopol, 2010, xxx, 875 s. ISBN 978-0-596-80548-7.**

Vedoucí diplomové práce: **Ing. Karel Greiner, Ph.D.**
Katedra informatiky v dopravě

Datum zadání diplomové práce: **6. prosince 2012**

Termín odevzdání diplomové práce: **23. května 2013**

L.S.

prof. Ing. Bohumil Culek, CSc.
děkan

doc. Ing. Josef Volek, CSc.
vedoucí katedry

V Pardubicích dne 6. prosince 2012

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou, nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 23. 05. 2013

Bc. Jakub Rykr

Poděkování

Autor by rád poděkoval Ing. Karlu Greinerovi, Ph.D. za vedení této práce. Dále by chtěl poděkovat Bc. Tomáši Kadaňkovi a Bc. Janu Tlučhořovi za spolupráci při vytváření společné aplikace.

ANOTACE

Práce je věnována vytvoření online režimu ve hře s tématem ekologické stopy.

KLÍČOVÁ SLOVA

programování, cloud, C#, Microsoft SQL, databáze, Microsoft WCF, webové služby, Windows Azure, ekologická stopa, online hra

TITLE

Online mode for mobile game with ecological theme

ANNOTATION

The work is dedicated to the creation of online mode in the game with the theme of ecological footprint.

KEYWORDS

programming, cloud, C#, Microsoft SQL, database, Microsoft WCF, web services, Windows Azure, ekological footprint, online game

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

SEZNAM POUŽITÝCH POJMŮ A ZKRATEK

NET Framework– knihovna od firmy Microsoft
ABC – adress, binding, contract (adresa, binding, kontrakt)
ACID – atomicity, consistency, isolation, durability (atomicita, konzistence, izolace, trvalost)
ADO .NET – knihovna pro práci s daty
API – aplication programming interface (programátorské rozhraní aplikace)
APM – asynchronous programming model (asynchronní programový model)
COM – component object model (komponentový objektový model)
CPU – central processing unit (centrální procesní jednotka)
CZK – česká měna
DirectX – knihovna pro práci s grafikou
DLL – dynamic link library (dynamicky připojitelná knihovna)
DU – database unit (databázová jednotka)
EAP – event based asynchronous pattern (událostní asynchronní model)
EGT – entity group transaction (transakce pro skupinu entit)
ES – ekologická stopa
GB – gigabyte
GHA – global hectare (globální hektar)
HTTP – hypertext transport protocol (hypertextový transportní protokol)
HW – hardware
IAAS – infrastructure as a service (infrastruktura jako služba)
ID – identifier (identifikátor)
IDL – interface definition language (definiční jazyk rozhraní)
IIS – Internet Information Services (Internet Informace Služby)
IPC – interprocess communication (meziprocesová komunikace)
KB – kilobyte
LIB – static library (statická knihovna)
LINQ – Language Integrated Query (jazykově integrované dotazování)
MSMQ – Microsoft Message Queue (Microsoft fronta zpráv)
MSSQL– Microsoft Server SQL
NF – normální forma
OSI – Open Systems Interconnection (Propojení otevřených systémů)
PAAS – platform as a service (platforma jako služba)
podproces – vlákno běžící v rámci procesu operačního systému
RAM – random access memory (paměť s náhodným přístupem)
REST – Restful services (Restful služby)
RFC – Request for comments (žádost o komentáře)
RPC – remote procedure call (vzdálené volání procedury)
SAAS – software as a service (software jako služba)
SOA – service oriented architecture (servisně orientovaná architektura)
SOAP – Simple Object Access Protocol (Jednoduchý protokol objektového přístupu)
SQL – Structured query language (strukturovaný dotazovací jazyk)
TB – terabyte
TCP – transmission control protocol (protokol kontroly přenosu)
trial – zkušební období
UDP – User datagram protocol (protokol uživatelských datagramů)

UML – Unified modelling language (unifikovaný modelovací jazyk)
URI – Uniform resource identifier (jednotný identifikátor zdrojů)
URL – Uniform resource locator (jednotný lokátor zdrojů)
USD – americký dolar
VM – virtual machine (virtuální stroj)
WAS – Windows Process Activation Server (Windows Aktivační Server Procesů)
WSDL – Web service description language (jazyk popisující webovou službu)
XML – Extensible Markup Language (značkovací jazyk)
XSD – XML schema definition (definice schematu XML)

Úvod

Cílem této práce je vytvoření podpory pro online režim ve strategické hře s tématem ekologické stopy. Hru tvoří ještě další dvě tvůrčí části, a to grafická část a část herního jádra. Těmto se věnují jiné dvě práce.

Teoretická část obsahuje stručný popis herního designu, což je oblast, která je při počátku samotného vývoje her. Dále obsahuje popis hry a klíčové prvky, kterými by měla být zajímavá. Popisuje aplikační framework XNA, který je použit k tvorbě mobilní hry. Pojednává o webových službách obecně. O tom, k čemu jsou a jak fungují. Podává základní informace k programování pomocí frameworku WCF, který je v práci použit k vytvoření služeb. Také obsahuje některé informace o databázích.

Praktická část popisuje online režim z hlediska herních pravidel. Popisuje konkrétní použití cloudové platformy Windows Azure. Shrnuje koncepčně navržení způsobu komunikace v online režimu. Obsahuje samotné informace o návrhu jednotlivých částí zajišťujících online režim. Věnuje se vytvoření komponenty XNA pro zobrazení světa ve hře. Poslední kapitola pak tvoří závěr a zhodnocení práce.

1 Herní design

Herní design je součástí tvorby hry. Jedná se o vymýšlení herních principů a herních cílů tak, aby hra byla zajímavá. Hlavním cílem je zaujmout hráče. Toho lze dosáhnout tím, že je stavěn před různé výzvy. Výzvy jsou jednou ze základních herních mechanik. Je potřeba se snažit, aby výzvy nebyly příliš jednoduché, nebo naopak, aby jich nebylo moc. Např. jednoduchá výzva může být "Sniž ekologickou stopu o x jednotek". Téma ekologické stopy je popsáno v části 2.1.1. Hra by tedy měla být založena na určité výzvě a odměně, která sama o sobě musí být dostatečně zajímavá a pro hráče přitažlivá. Hlavní výzvou může být například ovládnutí světa. Pro hráče se otevírá cesta, jíž se postupně dostává přes dílčí výzvy k tomu, aby dosáhl splnění té hlavní.

Hráče lze povzbudit k dosažení specifických cílů pomocí působení motivace. Ve hře mohou být dvě formy motivace – odměny zastupující pozitivní motivaci a tresty zastupující negativní motivaci. Následuje souhrn některých odměn a trestů.

1. Odměny

   - Pochvala – základní typ odměny. Hra může reagovat na splnění nějaké výzvy například hláškou chválící hráče (např. „Dobrá práce.“).

   - Skóre – používá se pro hodnocení hráče napříč celou hrou. Nasbírané skóre lze využít ke srovnávání hráčů a vytvoření nové výzvy – dosažení nejlepšího skóre.

   - Objevování – sem patří objevování nových částí hry či zpřístupňování nových funkcionalit – např. postupem do dalšího levelu.

   - Možnost vyjádřit se – vnést svou osobnost do hry. Hráč může mít například možnost vytvářet si a upravovat herního hrdinu.

   - Síla a moc – nějakým způsobem ovládnout herní prostředí nebo jeho část. Zde může záležet na různých prioritách hráčů.

   - Zdroje – herní prostředky, které nějakým způsobem může hráč využívat a s jejichž rostoucím množstvím je například silnější.

- Dokončení – největší odměnu hráč očekává ve chvíli, kdy splní všechny klíčové výzvy a dokončí hru.

2. Tresty

   - Zostuzení – opak odměny typu Pochvala. Hra reaguje hláškami oznamujícími prohru nebo ztrátu.

   - Ztráta bodů skóre.

   - Ztráta zdrojů – jedním z nejčastěji využívaných způsobů potrestání hráče je připravit ho o zdroje, jejichž prostřednictvím dosahuje moci. Mohou to být například virtuální peníze, munice apod.

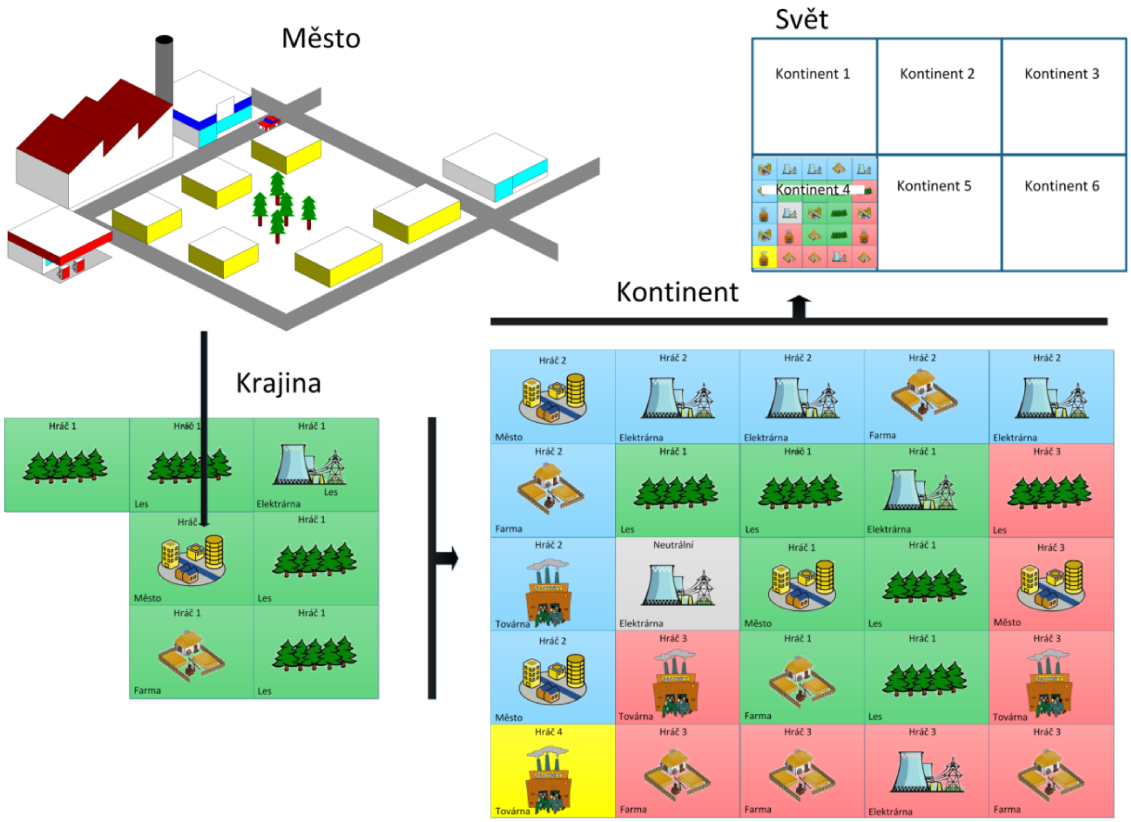
   - Oslabení – podle druhu hry způsobit hráči změny, které vedou k jeho oslabení v herním prostředí, například ztráta dovedností získaných během hry.

2 Popis hry

Hra je pojmenována The IMPrint. Je to online hra, která má za cíl poukázat na problémy vyspělých zemí s ekologickým dopadem jejich chování. Její tvorba je rozdělena do tří částí, z nichž jedna je popsána v této práci, a další dvě části zpracovávají jiní členové vývojového týmu. Tým tvoří kromě autora této práce Bc. Tomáš Kadaňka, který má na starost herní jádro, a Bc. Jan Tlučhoř, který má na starost herní grafiku.

Žánrově hra spadá do kategorie budovatelská simulační hra. Do této kategorie spadají hry, které se snaží zapojit realitu světa do hry. Dávají hráči možnost něco řídit a utvářet a důsledky, které toto hráčovo řízení má, jsou podobné důsledkům v reálném světě. Jako simulační hra se tedy snaží o jistou míru reálnosti, ale vždy je nutné udělat určitý kompromis.

Herní svět IMPrintu je zasazen do současnosti. Designově je hra směrována do kreslené podoby. Na herní svět lze nahlížet ve dvou pohledech (viz Obrázek 1). Ve hře bude jeden pohled zobrazující ve 3D grafice prostor města. Druhý pohled bude zobrazovat krajinu světa ve 2D grafice.



Obrázek 1: Koncept herních pohledů

Jádro hry tvoří budování a řízení města. Hráč bude město ovládat v pozici starosty. Při rozvíjení města je hráč konfrontován s potřebou získávání různých zdrojů nutných ke stavění budov, s okolními spoluhráči a s dalšími prvky. Jeho důležitým cílem by mělo být stabilizovat město do podoby, kdy je ekonomicky prosperující. V rámci tohoto snažení je hráč dále konfrontován s pojmem ekologická stopa. Ekologická stopa má ve hře různé projevy. (viz kapitola 2.1.1)

Cílová hráčská skupina této hry jsou majitelé chytrých mobilních telefonů s operačním systémem Windows Phone.

2.1 Klíčové vlastnosti hry

2.1.1 Herní téma ekologická stopa

Ekologická stopa je měřítkem toho, kolik produktivní země a vody potřebuje jedinec, město, stát nebo lidstvo k zajištění svých potřeb zdrojů a k zneškodnění odpadů, které jsou

vyprodukovány při využívání současných technologií a vzhledem k současnému stavu poznání. [2]

Ekologická stopa je vyjádřena v plošném měřítku tzv. globálními hektary(gha). Jeden globální hektar měří globální průměrnou produktivitu všech biologicky produktivních ploch za daný rok [14]. Jako biologicky produktivní plochy jsou označovány plochy souše a vodních ekosystémů, které podporují biologickou aktivitu a které využívá člověk.

Ekologická stopa má celosvětově vzestupný trend. To znamená nadužívání. V roce 2008 byla celková biokapacita Země stanovena na 12 mld. gha, neboli 1,8 gha na osobu. Ale ekologická stopa člověka byla 18,2 mld gha, neboli 2,7 gha na osobu [21]. Z těchto čísel lze spočítat, že celosvětově by lidé potřebovali k životu 1,5 (2,7/1,8) planety.

Průměrná ekologická stopa v České republice byla v roce 2012 5,27 globálních hektarů na osobu [37]. Jak bylo uvedeno, je celosvětově k dispozici pouhých 1,8 biologicky produktivních hektarů na osobu. Znamená to, že průměrný občan České republiky potřebuje k zajištění svých potřeb 2,93 (5,27/1,8) planety. Naše spotřeba tak výrazně překračuje regenerační schopnost Země.

Jeden ze spoluautorů ES, Kanaďan William Rees, ekologickou stopu přibližuje takto: "Kolik plochy (země a vodních ekosystémů) je třeba k souvislému zajišťování všech zdrojů, které potřebuji ke svému současnému životnímu stylu a k zneškodnění všech odpadů, které při tom produkuji?" Dále uvádí metaforu pro pochopení podstaty ekologické stopy: "Představte si ekonomiku jako velké zvíře. Otázka, kterou si musíme položit, zní, jak velkou pastvinu potřebujeme, abychom uživili toto zvíře?". [38]

EKOLOGICKÁ STOPA VE HŘE

Hráč je tedy ve hře postaven před výzvu udržovat svou ekologickou stopu. Bude mít k dispozici města s budovami. Bude se rozhodovat o tom, co za budovy postaví a kde je postaví. Dále činí další strategická rozhodnutí tak, aby se jeho ekologická stopa nezvyšovala.

Projevem velikosti hráčovy ekologické stopy je, že se zvyšující se ekologickou stopou je mu odebíráno stavební místo pro stavbu dalších budov. Je to pro hráče zpětná vazba, kdy poznává, že aby mohl stavět, musí se starat, aby ekologická stopa byla nižší.

V samotné hře je ekologická stopa považována za surovinu. Kromě toho zde existují i další druhy surovin. V tabulce 1 je uvedeno, jak se konkrétně ve hře s ekologickou stopou zachází.

NÁZEV	ZKRATKA	POPIS
Eko City	ec	Ekologická stopa města. Získá se součtem všech Eko House.
Eko House	eh	Ekologická stopa chování obyvatel jednoho domu. Vypočítá se dle chování obyvatel domu. Tuto hodnotu ovlivňuje, co obyvatelé konzumují, čím cestují atd. Tato stopa je ovlivňována i okolními činiteli (Eko Factory), jako jsou autobusová zastávka, popeláři apod.
Eko Space	es	Podává ekologickou stopu jinak. Neurčuje ekologickou stopu, ale celkový prostor v daném městě.
Eko Factory	ef	Ekologická stopa, kterou vytváří podniky pro svou výrobu apod. Tato hodnota může být kladná (např. popeláři), ale i záporná (např. továrny). Eko faktory vytvářejí i krajinné budovy, anebo spojenci či nepřátelé.

Tabulka 1: Ekologická stopa ve hře

Eko Space (ekologický prostor) se týká právě toho, jak je prostor provázán s hodnotou ekologické stopy. Je to přímo ve vztahu s tvrzením, že velikost ekologické stopy lze srovnávat například s požadavkem na 1,5 Země.

Celkově je tedy hráčova možnost výstavby po stránce prostoru omezena podmínkou $Eko\,Space \geqslant \sum (Eko\,House\,všech\,budov) + \sum (Eko\,Factory\,všech\,podniků)$. Pokud podmínka platí, hráč může z prostorového hlediska stavět, jinak nemůže.

Eko House je průměr kvantifikovaného ekologického chování všech obyvatel jednoho domu. Ve hře může nastat situace, že se nepřímo zvýší celková hodnota Eko House tak, že již postavené budovy zabírají větší prostor, než jim povoluje podmínka pro Eko Space. V takovém případě pak jsou některé budovy automaticky deaktivovány. Deaktivuje se minimální počet budov produkující Eko House tak, aby byla podmínka opět splněna.

Ve hře se nachází kromě ekologické stopy další suroviny, za které si hráč kupuje nebo vylepšuje budovy . Za suroviny může hráč budovy stavět a vylepšovat. Budovy také pro svůj chod požadují jistý příjem surovin. Suroviny ve hře lze rozdělit do sedmi kategorií:

- ekologická stopa (Eko City, Eko House, Eko Space, Eko Factory),
- peníze,
- energie,
- potraviny,
- pracovní pozice,
- výrobky,
- odpady.

BUDOVY

Budovy jsou jádrem hry. Většina herní doby se bude týkat právě práce s budovami. Pomocí budov hráč ovládá hru, získává peníze. Na druhou stranu mu také vznikají potřeby na jejich provoz.

Ve hře se nacházejí městské budovy v těchto kategoriích:

- Řídící budovy – budova tohoto typu může být obsažena maximálně jednou v každém městě. Patří sem například radnice, ambasáda.
- Obytné budovy – tyto budovy jsou jedny z nejdůležitějších. Lze jich mít libovolný počet (samozřejmě s ohledem na prostorové možnosti). Z chování lidí žijících v těchto budovách se vypočítává ekologická stopa. Patří sem například klasický dům, činžovní dům, luxusní dům.
- Ekologické budovy – tyto budovy se snaží působit šetrně na životní prostředí nebo se snaží řešit problémy chování obyvatel města. Patří sem například zahrádka, popeláři, park.
- Logistické budovy – tyto budovy zajišťují suroviny pro město, ale také mohou mít kladný vliv na ekologii města (například zavedením městské dopravy). Patří sem třeba obchod s potravinami, depo pro městskou dopravu, autobusová zastávka.

2.1.2 Mobilní platforma – Windows Phone

Každým rokem roste výkon mobilních zařízení. Na současných zařízeních není problém hrát náročné 3D hry, používat internet a dělat další aktivity dříve spíše vlastní stolním počítačům.

V dnešní době je tedy vhodné znát možnosti programování aplikací pro různé platformy – mobilní, desktopové.

Pro hru byla zvolena mobilní platforma. Konkrétně je použit operační systém Windows Phone verze 7.5. Další mobilní platformy jsou například Android či Apple iOS.

NĚCO MÁLO Z HISTORIE OPERAČNÍHO SYSTÉMU WINDOWS PHONE

Windows Phone byl představen v roce 2010 jako nástupce operačního systému Windows Mobile. Ve stejném roce přišla také první zařízení s tímto operačním systémem. Systém má nové uživatelské rozhraní využívající dlaždicového uspořádání. V roce 2011 byl představen Windows Phone 7.5 (kódové označní Mango), který zahrnoval asi 200 aktualizací k předchozímu systému [5]. Nynější verze tohoto operačního systému je Windows Phone 8.

2.1.3 Online hraní

Hráči hry se budou moci setkávat v online internetovém prostředí. Budou moci zjišťovat informace o jiných hráčích. Budou se vzájemně ovlivňovat a spolupracovat.

2.1.4 2D/3D zobrazovací režim

Herní prostředí se pro hráče zobrazuje ve 2D nebo 3D režimu. Když se hráč nachází v rámci globální mapy, tak je ve 2D režimu. Pokud se nachází ve městě, tak je ve 3D režimu. Pohled ve městě lze natáčet, přibližovat a i jinak s ním manipulovat.

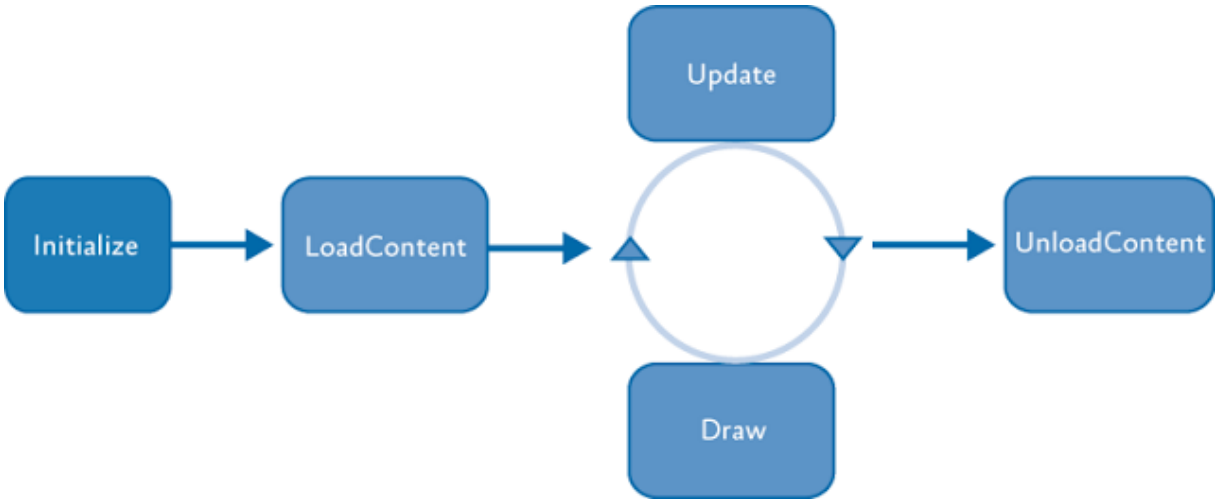
3 XNA

V této části jsou popsány některé základní pojmy programování s pomocí XNA.

XNA je framework od Microsoftu pro vývoj her na desktopovou a také na mobilní platformu. Jedná se o nadstavbu nad knihovnou DirectX. DirectX zapouzdřuje operace pro 2D/3D grafiku – matematické operace, vykreslování apod. XNA má pro vývojáře jednodušší API.

3.1 Použití XNA

Následuje popis některých důležitých principů a pojmů. XNA hra probíhá v herní smyčce. Herní smyčkou se nazývá koloběh metod `Update` a `Draw` třídy `Game`. Instance třídy `Game` je vytvořena při spuštění hry. V operaci `Game.Update` se aktualizuje herní prostředí a v `Game.Draw` se vykresluje na obrazovku (viz Obrázek 2). Průběh herní smyčky probíhá v čase, který lze získat z hodnot vlastností třídy `GameTime`.



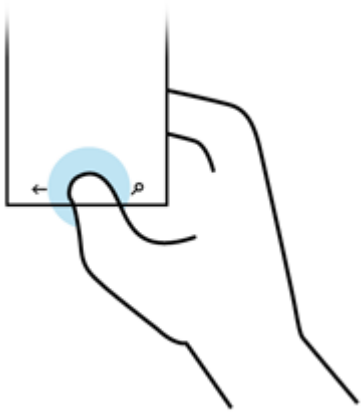
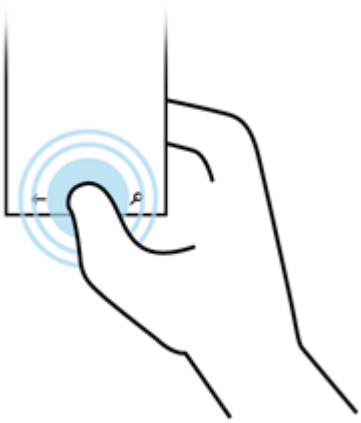
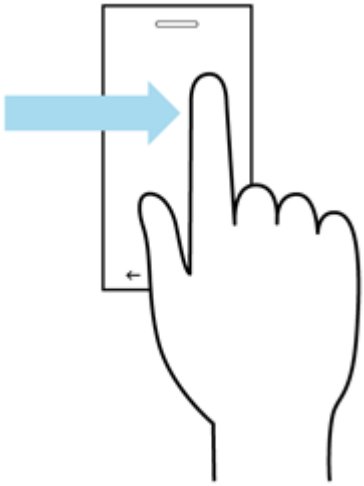
Obrázek 2: Herní smyčka [46]

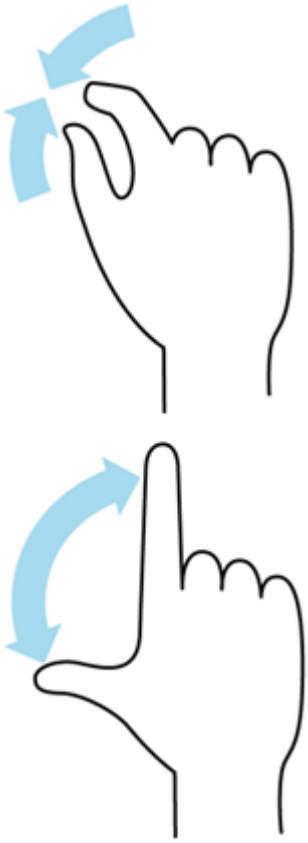
V XNA lze s výhodou využívat herní komponenty. Tyto komponenty jsou potomky třídy `GameComponent` nebo `DrawableComponent` a obsahují metodu `Update`, resp. metody `Update` a `Draw`. Pomocí komponent lze do hry komponovat různá menu, dialogy a další herní komponenty.

Načítání obsahu je realizováno pomocí content pipeline. Ve zkratce se jedná o manažer načítání obsahu. Nabízí zjednodušené načítání obvyklých datových zdrojů – obrázků, 3D modelů, zvuků. Každá komponenta má přístup k tomuto manažeru pomocí instance `Game`.

3.2 XNA pro Windows Phone

XNA existuje pro platformy Windows Desktop a Windows Phone. Pro platformu Windows Phone přibývá oblast ovládání mobilního zařízení dotykem a gesty. Windows Phone rozlišuje několik základních druhů dotyků a gest. Jedná se o například o drag, tap, flick (viz Obrázek 3).

KLEPNUTÍ  Jednou klepněte na položku.	*POKLEPÁNÍ*  Rychle za sebou dvakrát klepněte na položku.	*KLEPNUTÍ A PŘIDRŽENÍ*  Klepněte a nechte prst chvíli položený na displeji.
STISKNUTÍ  Jednou stiskněte hardwarové tlačítko.	*STISKNUTÍ A PŘIDRŽENÍ*  Stiskněte hardwarové tlačítko a nechejte na něm chvíli položený prst.	*POSUNUTÍ*  Položte prst na displej a při pohybu ho na něm stále držte.

PŘESUNUTÍ ZOBRAZENÍ  Rychle přejeďte prstem ve směru, ve kterém se má obrazovka pohybovat.	*STAŽENÍ A ROZTAŽENÍ PRSTŮ*  Položte dva prsty na displej a potom je spojte, nebo naopak roztáhněte.

Obrázek 3: Dotykové ovládání Windows Phone [45]

4 Cloud technologie

V této kapitole je popsána obecná teorie cloud computingu. Tento model byl zvolen jako poskytovatel dat a služeb hernímu online režimu.

4.1 Cloud computing

Cloud computing je model, který umožňuje všudypřítomně, pohodlně a na požádání přistupovat přes síť ke sdruženému konfigurovatelnému zdroji výpočetních prostředků (například serverů, úložišť, aplikací, služeb), které mohou být rychle použity a zase vráceny s minimální režijní snahou. Tento model je složen z pěti základních charakteristik, tří modelů služeb, a čtyř modelů nasazení. [7]

Cloud computing je založen na internetovém computingu. Sdílené výpočetní prostředky (hardware, software, data) jsou poskytovány počítačovým zařízením na požádání. Přechod ke cloud computingu je přirozený vývoj, který se základá na potřebě sdílení dat, distribuovaných aplikacích a vysokém výpočetním výkonu pro obsluhu mnoha klientů. Jedná se o vývoj směrem k virtualizaci, SOA architektuře (Service-Oriented architecture). Není zde potřeba řešit technické detaily hardwarové a softwarové infrastruktury. Tyto záležitosti jsou přenechány poskytovateli cloudového řešení.

4.1.1 Virtualizace

V datacentrech Microsoftu se vyskytuje obrovská výpočetní kapacita. Virtualizace je technologie, která dokáže maximálně využít tuto kapacitu. Virtualizace dokáže na jednom fyzickém stroji provozovat několik virtuálních strojů. Uživatelské aplikace totiž neběží přímo na hardwaru těchto datacenter, nýbrž je mezi nimi ještě jedna vrstva – virtualizační vrstva. Mechanismus virtualizace umožňuje přesouvání zátěže, protože virtuální stroj lze snadno přesunout, kdežto hardware už tak snadno přesunout nelze.

4.1.2 Základní charakteristiky

Základní charakteristiky cloud computingu jsou [7]:

- Samobslužný princip – zákazník si může samostatně opatřovat výpočetní zdroje dle své potřeby bez zásahu pracovníka technické podpory poskytovatele.

- Široká přístupnost – ke službám je přistupováno standardními mechanismy, což podporuje přístup mnoha různých klientů – mobilních zařízení, pracovních stanic apod.

- Sdružování zdrojů – výpočetní zdroje poskytovatele jsou sdružovány (anglicky resource pool) a dynamicky přidělovány za účelem obsluhy mnoha klientů podle velikosti jejich aktuálních požadavků. Klient většinou nezná přesné umístění zdrojů kromě širšího určení např. státu, regionu, datacentra.

- Rychlá elasticita – zdroje mohou být použity a zase vráceny, v některých případech automaticky, aby bylo možno pokrýt dynamické výpočetní nároky. Zákazníkovi se mohou zdroje jevit jako nekonečné a dostupné kdykoli.

- Měřená služba – cloudové systémy automaticky kontrolují a optimalizují používání zdrojů. Činí tak za využívání mechanismů měření podle konkrétního druhu služby (např. služby úložiště). Takto změřené používání zdrojů je poté přístupné jak klientovi, tak poskytovateli.

4.1.3 Modely služby

Cloud může zákazníkovi nabízet několik způsobů služeb [7]:

- Infrastructure as a service (IaaS) – zákazníkovi je poskytnuto pronajmutí výpočetních zdrojů, kde zákazník má kontrolu nad operačním systémem a softwarem, který je instalován.

- Software as a service (SaaS) – poskytovatel nabízí službu jako softwarovou aplikaci běžící na cloudovém řešení. Zákazník má možnost určité konfigurace. Může to být například poštovní služba. Aplikace je dostupná různým druhům klientů např. přes webový prohlížeč, nebo programové rozhraní.

- Platform as a service (PaaS) – zákazníkovi je poskytnuta možnost vytvářet vlastní aplikace využívající cloud. Při tomto přístupu zákazník využívá softwarové knihovny a jiné nástroje podporované platformou. Zákazník se nestará o správu nebo konfiguraci infrastruktury. Poskytovatel platformy spravuje vše od síťové konektivity až k prostředí pro běh aplikací (např. Microsoft .NET platforma). Vývojáři se poté mohou soustředit

pouze na své aplikace. Zákazník má kontrolu nad svými aplikacemi, které nahrává, a případně omezenou možnost konfigurace prostředí.

4.1.4 Modely nasazení

Cloud lze provozovat následujícími způsoby [7]:

- Privátní cloud – cloudové řešení je pouze pro výlučné použití nějaké organizace.
- Komunitní cloud – cloudové řešení je používáno specifickou komunitou zákazníků, kteří jsou členy organizací se společným zájmem.
- Veřejný cloud – cloudové řešení je určeno pro použití široké veřejnosti.
- Hybridní cloud – cloudové řešení se skládá z několika různých modelů nasazení.

4.2 Škálovatelnost

Škálovatelnost aplikace je měřítkem toho, kolik uživatelů může aplikace efektivně obsloužit v daný čas.[33] Můžeme například říci, že aplikace škáluje do 1000 uživatelů. Poté je bod, kdy už aplikace nezvládá obsluhovat další uživatele. Je to limit škálovatelnosti aplikace. Limit je dosažen, když už aplikace využila všechny dostupné výpočetní zdroje (hardware). Výpočetní zdroje zahrnují CPU, RAM, disk, kapacitu sítě a další. Škálování je tedy o alokaci zdrojů aplikace a jejich spravování aplikací tak, aby se minimalizovalo soupeření o zdroje.

Škálovatelnost aplikace může být rozšířena dvěma přístupy, vertikálním a horizontálním škálováním. Škálování se bude týkat uzlů, což jsou jednotky, jež mají hardwarové zdroje, které aplikace potřebuje ke svému běhu (výpočetní uzel, datový uzel). Můžeme tedy škálovat "nahoru" vertikálně, tj. zvýšením HW prostředků dostupných jednotlivým uzlům, nebo "ven" horizontálně, tj. přidávat další uzly. Vertikální přístup může zahrnovat dobu, kdy je aplikace nedostupná (anglicky downtime). Tyto přístupy nejsou vzájemně se vylučující.

Při horizontálním škálování je možné určit jednotku škálování, což jsou zdroje, které lze nějakým způsobem předem určit pro danou zátěž a které se přidělují najednou jako jedna jednotka.

Jedním z hlavních plusů používání cloudových technologií je možnost využít horizontální škálování (anglicky scalability) aplikace. Znamená to přizpůsobování se zátěži vytvářené klienty používajícími danou cloudovou aplikaci v daný čas.

4.3 Platforma Windows Azure

Platforma Windows Azure je veřejné cloudové řešení firmy Microsoft. Je to platforma ve smyslu PaaS (viz kapitola 4.1.3) [39]. Je kombinací hardware a software. Nachází se v datacentrech na různých geografických lokacích. Poskytuje řešení zajišťující základní charakteristiky cloud computingu.

Následují některé komponenty platformy [39]:

- Výpočet – poskytuje prostředí pro běh programů (web role, worker role, virtual machine)
- Úložiště – poskytuje možnosti ukládání dat (table storage, blob storage, queue storage, drive storage)
- Databáze – relační databáze jako služba (SQL Database)
- Service Bus – slouží k bezpečnému doručování zpráv mezi distribuovanými aplikacemi.
- Content delivery network – umožňuje přemístění dat blíže ke klientům.
- Caching – nabízí cachovací mechanismus, který je v paměti a distribuovaný. Umožňuje zrychlení aplikace.
- Virtuální síť – poskytuje různé možnosti propojení aplikací pomocí virtuální sítě.

Některé z uvedených komponent jsou dále popsány.

4.3.1 Výpočetní možnosti na Windows Azure

Ve Windows Azure se jako výpočetní jednotky používají tzv. role. Role odpovídá určité oblasti funkcí, pro které je určena. V rámci rolí je spouštěn určitý programový kód. Dále jsou uvedeny základní role:

- Web role – tato role se spouští na Internet Information Services (IIS) webovém serveru. Může sloužit například jako přístupový bod pro uživatele webových služeb.

- Worker role – tato role je nezávislá na vstupu uživatele. Stále běží a může provádět nějaké dlouho běžící operace. Popřípadě ji lze využít ve spolupráci s web rolí, kdy z web role jsou předávány požadavky, které zpracovává worker role (Queueing princip).

- Virtual machine role – umožňuje nahrát vlastní Windows Server operační systém.

Každá z těchto rolí je spuštěna v tzv. virtuálním stroji (VM). Existují různé velikosti VM. Velikost VM určuje v podstatě, jaký dostupný výkon má daný virtuální stroj.

INSTANCE ROLE

Při konkrétním použití role je vytvářena její instance. Je zde paralela mezi třídou a instancí v objektovém programování. Platí, že jedna role běží na jednom virtuálním stroji. Vytvářením více těchto instancí lze využívat horizontálního škálování (viz kapitola 4.2).

4.3.2 Možnosti ukládání dat na Windows Azure

Zde jsou popsány některé možnosti ukládání dat, které lze na platformě Windows Azure využít.

Data jsou do určité míry replikována tak, aby byla zajištěna jejich bezpečnost. Lze využít lokální redudance dat, kdy se data replikují v rámci jednoho datacentra či nějaké menší oblasti. Dále lze používat geograficky redudantní ukládání, kdy jsou data replikována mezi dvěma vzdálenými datacentry (vzdálenost 100 a více km). V každém z nich je poté ještě použita lokální redudance dat. [9]

Ve Windows Azure jsou k dispozici čtyři základní způsoby ukládání dat:

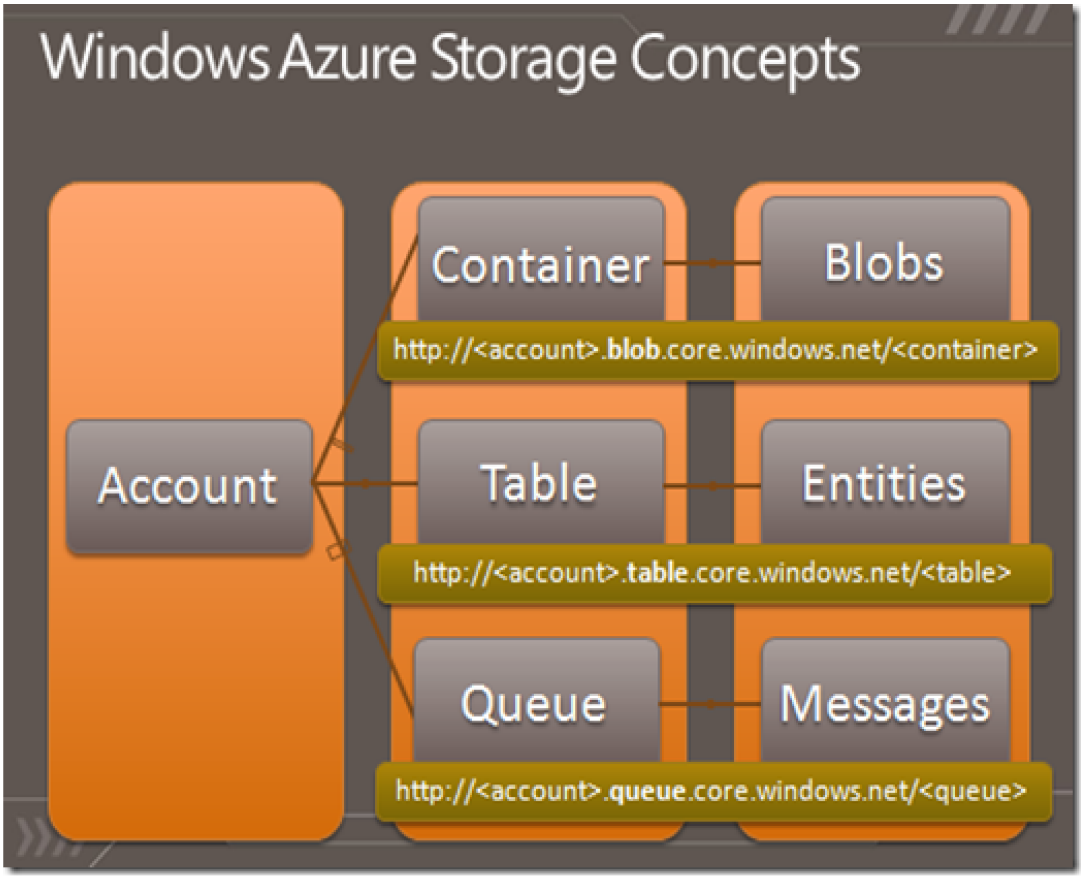
- Blob

- Table

- Queue

- SQL Azure

První tři způsoby vycházejí z podobného konceptu (viz Obrázek 4) a nazývají se Windows Azure Storage. K používání Windows Azure Storage je potřeba účet (storage account). Tento účet slouží k částečné identifikaci v rámci tohoto úložiště. Windows Azure Storage lze používat pomocí REST API (viz kapitola 5.2). Podstatnou vlastností je zde škálovatelnost. Jednak to zde představuje dynamickou změnu velikosti, jednak také přizpůsobení se zátěži (pomocí oddílů popsaných dále).



Obrázek 4: Koncept Windows Azure Storage [30]

WINDOWS AZURE TABLE STORAGE

Tabulka je množina entit, které obsahují množinu nějakých atributů. Entity mají dva povinné atributy – PartitionKey, RowKey. Atribut PartitionKey tvoří první klíčový atribut a RowKey druhý klíčový atribut. Systém používá atribut PartitionKey k distribuci zátěže (viz dále). RowKey jedinečně identifikuje záznam mezi záznamy se stejným PartitionKey. Kombinace atributů PartitionKey a RowKey tvoří jedinečnou identifikaci záznamu v tabulce.

Pro přibližnou představu jsou uvedena také nějaká čísla. Maximální velikost záznamu je 1 MB a maximální počet atributů na jeden záznam je 255 [30].

Operace nad tabulkami zahrnují vytváření / mazání tabulek, dotazování, vkládání / mazání záznamů. Tabulky mají také omezenou podporu pro transakční zpracování (pomocí EGT – Entity group transaction).

WINDOWS AZURE BLOB STORAGE

Bloby jsou velké binární objekty. Lze je využít k ukládání obrázků, videí nebo jakéhokoliv jiného druhu dat. Lze je využít například také k zálohování a samozřejmě ke sdílení dat.

Blob storage pracuje s bloby a kontejnery. Blob jsou data, která jsou identifikována nějakým klíčem. Kontejnery slouží pro ukládání blobů a také kontrolují přístup k blobům (veřejný, privátní).

WINDOWS AZURE QUEUES

Jedná se o spolehlivou metodu ukládání a doručování zpráv aplikacím. Zprávy jsou ukládány do front, které lze v aplikaci zpracovávat. Počet zpráv ve frontě je limitován pouze limitem účtu úložiště. Zpráva ve frontě má trvání omezené na týden a systém smazává starší zprávy. Zpráva má dále také omezenou velikost (v řádu KB).

KONCEPT ODDÍLŮ

Oddíl tvoří určitá data, která jsou manipulována společně. Do oddílu (partition) se data zahrnují, když mají stejný klíč oddílu (PartitionKey) (viz Tabulka 2). Oddíl vlastně tvoří jednotku distribuce (unit of distribution [28]). Záznamy, které odpovídají stejnému klíči oddílu, mohou být přeneseny na jiný méně vytížený uzel (anglicky load balancing). Každý typ objektu ve Windows Azure Storage má klíč oddílu [30].

Například ve Windows Azure Table Storage se jako PartitionKey používá kombinace názvu dané table (TableName) a atributu PartitionKey (různé záznamy v table jej mohou mít stejný). Záznamy v table odpovídající této stejné kombinaci tvoří oddíl. Z tabulky také například plyne, že jeden blob vlastně tvoří jeden oddíl.

TYP ÚLOŽIŠTĚ	KLÍČ ODDÍLU
Blobs	ContainerName + BlobName
Entities	TableName + PartitionKey
Messages	QueueName

Tabulka 2: Koncept oddílů Windows Azure Storage [30]

WINDOWS AZURE SQL DATABASE

SQL Database je relační databáze poskytovaná jako služba. Je založena na standardní Microsoft SQL databázi, ale je ochuzena o některé funkce.

SQL Database server a databáze jsou pouze virtuální objekty, které přímo nekorenspondují s fyzickým serverem a databází, ale poskytují pouze relační databázi jako službu.

4.3.3 Platební model Windows Azure

Existuje několik platebních plánů, jak lze používat službu Windows Azure. Je zde Pay-As-You-Go plán, kde uživatel platí pouze, co skutečně použije, a není nějak dopředu platebně vázán. Dále jsou předplacené služby. Je zde 6ti měsíční a 12ti měsíční plán. Platba u těchto dvou plánů startuje na minimální částce 500$/měsíc [40]. Výhodou těchto plánů jsou ale různé slevy, které poskytují. Při překročení nastavených limitů těchto služeb je pak použit plán Pay-As-You-Go.

5 Webové služby

Pojem služba v sobě nese informaci o tom, že se jedná o nějakou užitečnou funkci. V případě webové služby o softwarovou funkci. Webová služba v podstatě zajišťuje sdílení funkcionality, kterou poskytuje, více klientům. Mohou to být obchodní systémy a podobně. Webové služby používají protokol HTTP a některý formát výměny zpráv (např. XML). Mohou jej využívat jako transportní protokol, to je i případ SOAP/WSDL služeb, nebo jako vlastní aplikační protokol, který definuje množinu operací, které je možno provádět (REST služby).

5.1 HTTP

Http (Hypertext Transport Protocol) je protokol, který operuje na úrovni aplikační vrstvy OSI modelu. V roce 1990 vznikl World Wide Web a s ním tento protokol. Je definován v RFC 1945 a RFC 2616. Protokol je implementován ve dvou programech – klientský program a serverový program. Každý z těchto programů se nachází na jiném systému a komunikují spolu pomocí zasílání zpráv. Struktura zpráv a způsob jejich komunikace jsou předem definovány. Klientský program dnes běžně tvoří webový prohlížeč. Serverovou část tvoří webový server (například Apache Web Server, Microsoft Internet Information Server). Základní komunikace je vyžádání existující webové stránky klientem. Webová stránka je identifikována pomocí URL adresy a poté může server zaslat odpověď, ve které se bude nacházet obsah dané stránky. Pro přenos dat se používá protokol TCP (transportní vrstva OSI modelu), který slouží ke spolehlivému přenosu dat. Dále může být použit ještě UDP protokol. HTTP je bezstavový v tom smyslu, že server neuchovává informace o klientech a každý požadavek je znovu vyřizován, jako by byl nový. [34]

V rámci komunikace zde lze použít takzvaná stálá a dočasná spojení. Je možné, že HTTP klient a server společně budou komunikovat delší dobu. To znamená, že klient odešle více požadavků na server. Pokud se pro přenos používá TCP, může se v takovém případě využít stálé TCP spojení, kdy je několik požadavků klienta a odpovědí serveru komunikováno v rámci stejného TCP spojení. V případě dočasného spojení je pro každou dvojici požadavek klienta, odpověď serveru vytvořeno nové TCP spojení, s čímž jsou spojeny režijní náklady. Výhodou stálého spojení je tedy mj., že se šetří režijní náklady na vytváření spojení a také se zkracuje čas, ve kterém jsou přijímána data. HTTP používá jako výchozí hodnotu stálé TCP spojení.

5.2 API styly webových služeb

Návrh webových služeb lze podřídit různým stylům rozhraní (API), které služba může využívat. Realizace může být například provedena ve stylu, kde zprávy odpovídají běžnému volání funkcí, nebo pomocí speciálního formátu komunikačních zpráv.

RPC API [35] je API, kde se vyvolává vzdálená procedura (remote procedure call). Klient zasílá vzdálenému serveru zprávu, která identifikuje metodu, co se má zavolat. Také obsahuje případné parametry této metody. Jakmile se zpráva dostane na server, je odbavena – je zavolána příslušná procedura a jsou jí předány parametry. Klient je blokován, dokud se nevrátí výsledek volání.

Message API [35] je API, kde není přímá vazba na vzdálenou proceduru, ale je zde nějaká množina společných zpráv, které se při volání používají jako parametry. Tyto zprávy obsahují v nějaké formě informace o tom, co se má provést. Je zde tedy v podstatě ještě jedna vrstva mezi vlastním vyvoláním nějaké cílové procedury. Rozhraní služby je vytvořeno po navržení zpráv a má v podstatě metody pro příjem těchto zpráv. Tyto zprávy mohou zastupovat příkazy, události a jiné záležitosti.

Resource API [35] je API, které využívá samotného protokolu HTTP, respektive URI identifikátorů a standardních HTTP metod. Toto API není orientované jako procedurální API, nýbrž jako datové. Poskytuje přístup k datům. Realizace čtení, změn dat a podobných operací je poté nějakým způsobem uskutečňováno pomocí metod HTTP protokolu a identifikace zdrojů pomocí URI.

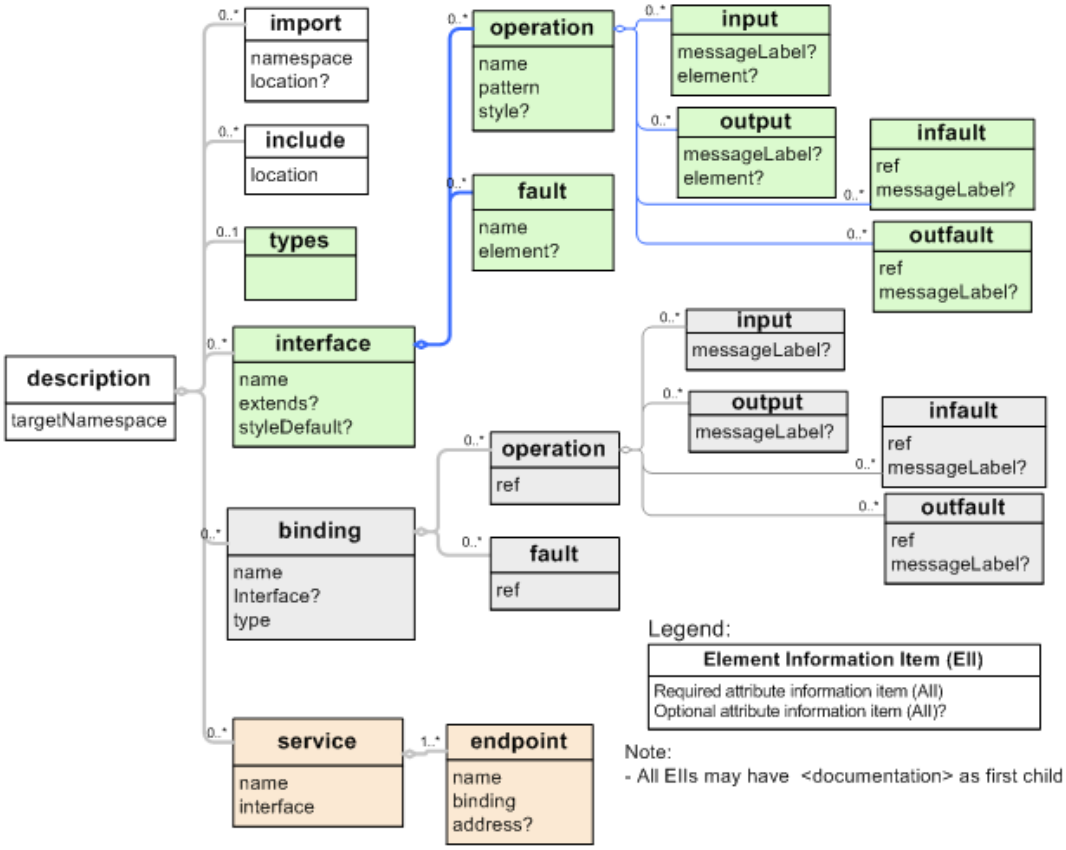
Při komunikaci s webovými službami vzniká latence neboli zpoždění. Je to dáno tím, že se komunikuje v rámci vzdálených a rozdílných bodů. Požadavky se musí serializovat jako proud bytů, přenést přes hranice procesu programu, musí být zachyceny serverovým procesem, deserializovány a namířeny k patřičnému zpracování. Podobně se poté děje při odesílání odpovědi. Z hlediska latence je žádoucí, aby API minimalizovalo počet nutných požadavků, které klient musí odeslat, aby vykonal to, co potřebuje.

5.3 WSDL

WSDL – Web Service Definition Language je jazyk, kterým se popisují definiční informace webových služeb. Jedná se o jazyk využívající specifický XML formát. Informace, které obsahuje, jsou strojově čitelné a nezávislé na konkrétním programovacím jazyku.[17]

XML schéma nebo také XML Schema Definition (XSD) popisuje strukturu XML dokumentu.[23] Účel XML schemat je definovat povolené stavební bloky XML dokumentu. Definuje například elementy a jejich atributy, které se mohou objevit v dokumentu. XML schéma podporuje datové typy. XML schéma umožňuje validovat XML dokument po obsahové stránce.

Každý WSDL dokument obsahuje element description, který slouží jako kontejner pro ostatní elementy. Dále obsahuje element types, kde jsou pomocí XML schématu definovány typy (zprávy). Dále je element interface, který definuje sadu abstraktních operací služby. Každá operace specifikuje typy svých jednotlivých parametrů. V rámci operací jsou také stanoveny typy chybových zpráv (element fault), které se používají. Dalším elementem je element binding. Účelem tohoto elementu je specifikovat, jak se mohou zprávy vyměňovat. Specifikuje konkrétní formát zpráv a přenosový protokol pro jednotlivé operace (například se může jednat o formát zpráv SOAP a protokol HTTP). Dalším elementem je element service. Ten specifikuje, kde je služba dostupná. Definuje jméno služby, pouze jeden interface a seznam koncových bodů (endpoints). Koncový bod obsahuje jméno, použitý binding a adresu URL, na které je možné ke službě přistupovat za použití daného druhu bindingu.[13] Na obrázku 5 je pro ukázku jednotlivých elementů zobrazenWSDL 2.0 Infoset.



Obrázek 5: WSDL 2.0 Infoset

Dále může WSDL dokument také obsahovat dokumentaci ke službě, která popisuje zamýšlený způsob použití a jiné detaily.

6 WCF

WCF (Windows Communications Foundation) je softwarový framework, který umožňuje vytváření služeb ve Windows. Jedná se o implementaci množiny průmyslových standardů pro vytváření služeb, zajištění komunikace služeb, typové konverze, management různých protokolů atd. [27]

6.1 SOA

SOA (Service Oriented Applications) je metoda psaní aplikací, která navazuje na snahy o oddělení závislostí (loose coupling) mezi systémy. Historicky bylo nejdříve funkční programování – to je například, jak se píší programy v jazyce C. Zde plní roli jednotky znovupoužití funkce. Funkce je vázána na data, která manipuluje, a pokud jsou to globální data, tak to použití omezuje. Poté navazovalo objektové programování. Objektové programování zapouzdřuje data a funkce do objektu. Mechanismus znovupoužívání je založen na třídě popsané zdrojovým kódem. To přinášelo problémy, protože se musely zpracovávat tyto zdrojové soubory při každé změně. Navíc nebylo možné používat v kombinaci více programovacích jazyků. Poté přišlo komponentové programování. Přineslo s sebou technologie jako je statická knihovna (lib), dynamická knihovna (dll), technologii COM (Component Object Model) a technologii .NET. Zde se nachází zaměnitelné komponenty, které mají definované binární rozhraní. Při tomto přístupu není sdílen zdrojový kód, ale dohoda na binárním rozhraní (IDL). To umožnilo načítat komponenty dynamicky za běhu aplikace a také používat různé zaměnitelné komponenty. Vznikly ale další problémy v rámci kooperace komponentově orientovaných systémů. V servisně orientované architektuře je služba, která slouží ve smyslu dřívějších komponent, ale je také oddělena technologická platforma. Službu mohou používat různorodí klienti. Interakce mezi klientem a službou je založena na výměně zpráv, která je zřízena na nějakém standardu. Služba nějakým způsobem deklaruje, jaké operace podporuje a jaké podmínky musí klienti splňovat, a pokud je nesplňují, nemohou službu použít. Tato deklarace je na úrovni služby a ne jejích operací. V servisně orientované aplikaci se spoléhá na standardní implementaci komunikace se službou. Neznamená to, že interoperabilita je možná pouze pomocí servisně orientované aplikace, ale servisní orientace poskytuje již mnoho připravených prostředků (implementace spolehlivé komunikace apod.), které lze využít k vytváření robustních aplikací, ve kterých se mohou vývojáři více věnovat samotnému jádru aplikace a implementaci jednotlivých obchodních požadavků.

6.2 Hosting

WCF služba je hostována v nějakém procesu. Tento proces může být poskytnut několika způsoby [27]:

- IIS hostování – hostování v rámci Internet Information Services webového serveru (používá HTTP)
- Vlastní hostování – hostující proces je plně v režii vývojáře.
- WAS (Windows Activation Server)

Aplikační doména je logický kontejner pro několik .NET sestavení. Účelem aplikační domény je poskytnout izolaci. Aplikační doména vlastní v ní vytvořené objekty a objekty z jiné aplikační domény, k těmto mohou přistupovat jenom omezeně. [44]

Hostující .NET proces může obsahovat více aplikačních domén. Každá z těchto domén může mít 0 nebo více instancí hostujících WCF službu (instance typu `ServiceHost`[1]) a každá z těchto instancí má 0 nebo více kontextů. Kontext tvoří nejnižší jednotku provádění. Kontext je asociován s 0 nebo více konkrétními instancemi služby (instance tříd implementující kontrakt).[27]

6.3 Binding

Binding reprezentuje objekt, který nějakým způsobem popisuje způsob komunikace se službou (viz také kapitola 5.3), například to, jaký se použije transportní protokol. Služba deklaruje, jaký binding (může jich být i více) je schopna používat, a klient musí použít stejný. Ve WCF je k dispozici několik často používaných bindingů[27].

- basic binding
- TCP binding
- IPC binding
- web service binding (WS)

[1] Třídy vztahující se k WCF se nacházejí většinou v sestavení `System.ServiceModel.dll` ve jmenném prostoru `System.ServiceModel`.

- MSMQ binding

Některé z těchto bindingů jsou uplatnitelné jen, když je klient i server WCF.

6.4 Koncový bod služby (endpoint)

Každá služba má asociaci s nějakou adresou, bindingem a kontraktem služby. Této trojici se ve WCF říká koncový bod (viz také kapitola 5.3), a lze pro ni používat zkratku ABC. Každá služba může mít více koncových bodů. Adresy koncových bodů jsou jedinečné a každý z nich musí mít přesně jeden kontrakt.

6.5 Průběh volání

Při komunikaci ve WCF se používají proxy. Proxy se stará o nalezení služby (adresy) a vytváření spojení se službou. Dále zajišťuje odesílání požadavků klienta a příjem odpovědí. Proxy obsahuje metody, které odpovídají rozhraní služby. Proxy je odvozena od typu `ClientBase<T>`. Ke svému fungování potřebuje konfiguraci, která může být provedena prográmatorskými příkazy, nebo je získána z konfiguračního souboru (soubor ServiceReferences.clientConfig). Tato konfigurace specifikuje koncové body služeb.

V rámci komunikace se službou se provádí několik kroků. Volání z proxy projde přes sestavu tzv. kanálů (channel), jednou na straně klienta a podruhé na straně serveru. Tyto kanály plní různé funkce. Mohou být například zodpovědné za šifrování zprávy. Posledním kanálem na straně klienta a prvním na straně serveru je transportní kanál. Ten provádí vlastní odeslání zprávy. Na straně serveru předá poslední kanál zprávu tzv. dispatcheru. Ten poté provádí samotné volání na serveru.[27]

6.5.1 Vypršení volání

Každé volání služby, které uskuteční WCF klient, musí být ukončeno v časovém limitu. Tento limit lze konfigurovat v konfiguračním souboru, nebo v programovém kódu. Dodržení limitu hlídá proxy objekt. Pokud je limit překročen, volání je přerušeno a je vyvolána výjimka `TimeoutException`. Výchozí hodnota je 1 minuta.

6.6 Service kontrakty

Service kontrakt popisuje, co služba nabízí. Jedná se o obdobu rozhraní. Ve WCF je použit atribut `ServiceContract` z jmenného prostoru `System.ServiceModel`. Tímto

atributem se označí rozhraní, které je pro službu definováno. Jednotlivé operace rozhraní, které má služba nabízet, se označí atributem `OperationContract`.

6.7 Data kontrakty

Data kontrakty na rozdíl od service kontraktů popisují, jaké zprávy se v rámci služby vyměňují. Atributem `DataContract` je označen programový typ, který má sloužit jako zpráva. Dále se atributem `DataMember` označí atributy tohoto typu, které budou součástí této zprávy. Data kontrakty fungují na opt-in principu. To znamená, že lze specifikovat jen některé atributy, které se mají do datového kontraktu zahrnout.

Data kontrakty označují typ pro datovou serializaci, což využívá instance třídy `DataContractSerializer`.

V rámci komunikace musí mít komunikující strany lokální definici typu, která odpovídá schématu daného datového kontraktu.

6.8 Mód instancí služby

V rámci vykonávání volání služby mohou být klienti obslouženi nějakou instancí služby. WCF poskytuje tři způsoby, jak mohou být klienti obslouženi instancí služby[27]:

- instance služby pro každé volání (`PerCall`)
- instance služby se session (`PerSession`)
- singleton instance služby (`Single`).

Výchozí způsob obsluhy je `PerSession`. Management instancí se týká pouze hostující strany služeb a ne klientů.

Ke konfiguraci módu se používá atribut `ServiceBehavior`. Atribut definuje vlastnost `InstanceContextMode`, která má odpovídající hodnoty pro tři uvedené způsoby. Context ve slově `InstanceContextMode` souvisí s kontextem, jak byl popsán dříve (viz kapitola 6.2). Ve skutečnosti se totiž jedná o instance kontextu, kterých se mód týká.

Instance služby se session vyžaduje perzistentní HTTP spojení (viz kapitola 5.1).

6.9 Ošetření chyb

V rámci WCF vznikají při komunikaci tři druhy chyb – chyby komunikační, chyby plynoucí z použití neplatného proxy objektu, chyby vzniklé vykonáváním operace služby [27]. Chyby vzniklé z třetí uvedené možnosti se ke klientovi dostávají jako `FaultException`, a tím separují klienta od možnosti chyby nějak řešit (maskování chyb). Služba jako taková by měla být v řešení chyb autonomní a neměla by záviset na klientovi. `FaultException` také ve většině případů zneplatní proxy objekt.

WCF nabízí také možnost jistého odmaskování chyb použitím kontraktu chybových zpráv. Kontrakt udává specifické chybové údaje. Kontrakt je definován pomocí atributu `FaultContract` a je vázán ke konkrétní operaci služby. Použití chybového kontraktu je ve spojení s třídou `FaultException<T>`, kde typový parametr T reprezentuje právě informaci o chybě.

6.10 Konkurence

Klient může pomocí stejného objektu proxy uskutečnit více volání najednou. Může je uskutečnit v sérii za sebou v jednom podprocesu nebo v různých podprocesech souběžně. Na straně serveru mohou být volání přiřazena různým pracovním podprocesům a konkurenčně přistupovat k instanci služby. Závisí na konfiguraci služby, jak jsou volání synchronizována. Výsledný způsob zpracování je dán konfigurací kombinace módu instance, módu konkurence a druhu transportního spojení [27].

Jsou tři typy módu konkurence:

- `Single`
- `Multiple`
- `Reentrant`

Mód konkurence je vlastnost atributu `ServiceBehavior` a určuje, jestli a kdy je povolen konkurenční přístup k instanci. Například v případě módu vytváření instance pro každé volání a s perzistentním transportním spojením, mód `Single` určuje, že volání jsou obsloužena jedno po druhém, a mód `Multiple` určuje, že volání jsou vykonávána konkurenčně, jak

přijdou. V případě, že není perzistentní transportní spojení, nemá mód konkurence vliv, a volání jsou zpracována konkurenčně.

Výchozím způsobem zpracování je mód konkurence `Single` a mód instance `PerSession`.

6.11 Throttling

Throttling [27] je technika vyrovnávání zatížení služby. Cílem je, aby klienti byli včas obslouženi. Působení throttlingu se děje v rámci všech instancí daného typu služby. V rámci throttlingu lze stanovit např. maximální počet konkurenčních volání (výchozí počet je roven 16 krát počet procesorů (jader)).

7 Databáze

7.1 Transakce

Transakce je soubor změn v databázi, které se provedou podle určitých pravidel. Tato pravidla jsou atomicita, konzistence, izolace, trvalost. (zkratka ACID)

Atomicita znamená, že transakce je buď provedena jako celek, nebo není provedena vůbec. Transakce má k tomu příkazy Commit a Rollback.

Konzistence znamená, že transakce převádí systém z jednoho konzistentního stavu do dalšího konzistentního stavu. Transakce musí začít v konzistentním stavu a skončit konzistentním stavem. Jedná se o to, aby byla dodržena konzistence a správnost modelovaných dat. Každé omezení musí být platné.

Izolace znamená, že transakce je oddělená od efektů jiných transakcí. MSSQL používá dva modely založené na zamykání záznamů a další založený na verzi záznamu [24]. Dále v textu jsou popsány jednotlivé izolační úrovně.

Trvalost znamená, že transakce je po potvrzení trvalá. Je to zajištěno i vzhledem například k selháním systému.

V MSSQL existuje několik izolačních úrovní. Izolační úroveň stanovuje pravidla při konkurenčním čtení a zápisu. Čtení vyžaduje takzvaný sdílený zámek (anglicky shared lock). Zámek je kontrolní prostředek, který transakce získá k ochraně dat. MSSQL kontroluje zámky nad tabulkami, individuálními řádky a jinými částmi databáze [26]. Sdílený zámek může získat více transakcí současně. Na rozdíl od čtení zápis vyžaduje vždy exkluzivní zámek (exclusive lock) a zámek vždy trvá do ukončení transakce. Transakce nemůže získát exkluzivní zámek, pokud jiná transakce drží jakýkoli zámek na žádaný zdroj [25].

Ve výchozí instalaci MSSQL serveru je výchozí izolační úroveň READ COMMITED [25], což znamená, že lze číst pouze konzistentní (commited) data.

V SQL Databázi (Azure) je výchozí izolační úroveň READ COMMITED SNAPSHOT [25]. Zde se používá model verzování řádků. Při pokusu číst data, která jsou nějakou transakcí modifikována, je vrácena kopie posledního konzistentního stavu (snapshot) dostupného v čase

začátku příkazu, a tudíž zde klienti při čtení nejsou blokováni. Stále jsou však vystaveni možnosti změny při opakovaném čtení (nonrepetable reads) a možnosti fantomového čtení (phantom read) [25].

7.2 Indexy

Index je databázový objekt, který slouží ke zrychlení přístupu k záznamům a ke zrychlení vykonávání dotazů. Indexy v MSSQL jsou realizovány jako B-strom index [26]. B-strom je vyvážený k-cestný strom. MSSQL používá typ B-stromu zvaný B+-strom (B plus strom). Velikost indexu vychází z definice indexu, počtu a velikosti záznamů v tabulce, která se indexuje.

Index má dvě komponenty – úroveň listů a úroveň ne listů [26]. Úroveň ne listů slouží k navigaci k položkám indexu. Úroveň listů obsahuje nějaký údaj pro každý indexovaný záznam. Podle toho, co je uloženo v těchto komponentách se rozlišují indexy klastrované a neklastrované.

Klastrovaný index (clustered index) je index, ve kterém jsou všechna data tabulky součástí indexu. Klastrovaný index je provázaný s fyzickým uspořádáním záznamů, a proto tabulka může mít pouze jeden klastrovaný index. Jakmile je vytvořen, data jsou překopírována z tabulky a uložena podle klíče klastrovaného indexu.

Neklastrovaný index (nonclustered index) je index, který neobsahuje všechny datové sloupce tabulky, ale obsahuje pouze data definovaná indexem. Index je separovaný od bázových dat, na které se pouze odkazuje. Bude se odkazovat na bázovou tabulku, což může být jiný klastrovaný index nebo halda (struktura, ve které se ukládají data, když není klastrovaný index [26]). Podle toho, na co se index odkazuje, je pak odkazem klíč v indexu respektive rowid (identifikátor) záznamu pro haldu.

7.3 Normalizace databáze

Nejdříve jsou zde stručně shrnuty pojmy n-tice, relační typ, relační hodnota a relační proměnná.

N-tici tvoří n prvků, které jsou dány třemi složkami – jménem atributu, typem, hodnotou. Jméno atributu a typ tvoří jeden atribut n-tice. Kompletní množina těchto atributů vytváří hlavičku n-tice. Typ n-tice je určen touto hlavičkou. Nad atributy n-tice neexistuje žádné třídění, protože jsou definovány jako množina. Atributy jsou vždy odkazovány jménem. Každá n-tice

obsahuje právě jednu hodnotu (správného typu) pro každý atribut. Podmnožina n-tice tvoří jinou n-tici.

Relační hodnota má hlavičku a tělo. Tělo obsahuje n-tice, jejichž hlavička je shodná s hlavičkou relační hodnoty. Hlavička určuje relační typ. Relační hodnota splňuje některá pravidla. Jednak zde n-tice splňují to, co platí pro n-tice, a dále zde neexistuje žádné pořadí n-tic a žádné dvě n-tice nejsou stejné.

Relační proměnná je proměnná, která má nějaký relační typ. Všechny možné relační hodnoty, kterých může tato proměnná nabývat, jsou omezeny daným relačním typem. Definicí relační proměnné vzniká v systémovém katalogu záznam s údaji o této proměnné. Relační proměnná má kandidátní klíč a cizí klíče.[29]

Pro relační hodnoty bylo uvedeno, že žádné dvě n-tice nejsou stejné (jsou jedinečné). Nechť K je množina atributů relační proměnné R. Pak K je kandidátní klíč proměnné R pouze tehdy, když splňuje podmínku jedinečnosti a neredukovatelnosti (v tomto případě to znamená minimálnosti množiny K). Podmínka jedinečnosti stanoví, že v rámci hodnot relační proměnné R, nenabývají dvě různé n-tice stejných hodnot atributů pro dané K. Podmínka neredukovatelnosti pak znamená, že žádná podmnožina K nesplňuje podmínku jedinečnosti. [29]

Normalizace databáze je proces, při kterém se nahrazuje relační proměnná za jiné více vhodné (rozpadem relační proměnné za několik relačních proměnných). Normalizace je proveditelná i nazpět (např. ze 3NF do 2NF), a tím pádem je také bezztrátová nebo také zachovávající informaci. Bezztrátovost je vázána na dodržení funkčních závislostí mezi jednotlivými atributy. Normalizace vede k odstraňování datových duplicit. Existuje několik normálních forem. Každá vyšší normální forma splňuje tu předcházející nižší.

8 Koncepce online komunikace

V této kapitole je stručně popsáno, jak je navržena komunikace v souvislosti s úložištěm, službami a mobilní hrou. V další kapitole je poté návrh všech částí více rozebrán.

8.1 Část na Cloudu

8.1.1 Běh služeb

Systém online komunikace funguje na principu WCF služeb, které jsou spouštěny pomocí instance webové role. Web role je kód spouštěný a hostovaný na webovém serveru IIS (Internet Information Service), aby byla zajištěna webová komunikace pomocí HTTP.

Klienty tvoří jednotlivá mobilní zařízení, která mají nainstalovanou herní aplikaci. Aplikace zná komunikační body služeb (endpoints) a jejich kontrakty.

8.1.2 Použité technologie

ADO .NET ENTITY FRAMEWORK

Framework pro práci s relačním databázovým modelem v objektovém programovém prostředí. Použitá verze má číslo 5.0. Entity Framework pracuje na principu odpojených dat. Nejdříve se vytváří kontext, což je objekt, který slouží pro komunikaci s databází. Konkrétně je používaný kontext instance typu `DbContext`. Obsahuje kolekce typu `DbSet`, do kterých se nahrávají data z databáze jako instance tříd (tyto třídy odpovídají objektovému-relačnímu modelu databáze), nebo ve kterých může také opačně uživatel vytvářet tyto instance nebo je mazat a upravovat. Kontext všechny prováděné změny na instancích sleduje. Tyto změny dokáže převést do SQL jazyka a zaslat je databázi k vykonání. Odpojený princip značí to, že data jsou upravována v odpojeném režimu, kdy ještě nejsou změny prováděny v žádné transakci. Jakmile se ale kontextu řekne, aby uložil změny, vytvoří transakci, převede změny na SQL kód a provede jej v databázi. Při úspěchu se provede commit transakce, při neúspěchu rollback transakce.

Kontext funguje nejlépe, pokud se používá podle vzoru Unit of work[2]. Unit of work tvoří logickou jednotku vzájemně souvisejících operací pro provedení transakce. Při uplatňování tohoto vzoru se vždy vytváří nový objekt kontextu na úrovni logické jednotky, provedou se

[2] Více o Unit of work viz [36]

změny a uloží se v databázi. V případě, že se neuplatňuje tento vzor, a kontext tedy existuje delší dobu, či je sdílený, tak hrozí možné chyby. Jedním typem takové chyby je, že totiž kontext prováděné změny akumuluje, a pokud se provede změna, která ohrozí správné uložení do databáze např. z důvodu porušení pravidel referenční integrity, tak je potřeba takovou chybu opravit. To ohrožuje doposud provedené změny a může vyžadovat nápravu. V případě uplatňování principu Unit of work jsou takové chyby lépe identifikovatelné.

LINQ(LANGUAGE INTEGRATED QUERY)

V Entity frameworku lze pro získávání dat z databáze používat objektový model pomocí technologie LINQ to Entities. Jedná se o způsob integrování dotazovacího jazyka do programovacího jazyka. Aby bylo možné takto psané dotazy interpretovat pro databázi, jsou převedeny do SQL jazyka za pomocí výrazových stromů (expression tree).

8.2 Část na mobilní aplikaci

Mobilní aplikace tedy zná koncové body služeb. Dále sdílí WCF datové kontrakty, aby mohla přijímat a odesílat data. V rámci samotné aplikace jsou jednotlivé funkce rozvrstveny – vrstva komunikace se službami, vrstva ukládání dat, vrstva vyřizování herních operací. Veškerá komunikace se službami probíhá asynchronně za pomoci objektů proxy.

Je také použita knihovna Reactive extensions. Knihovna pomáhá při asychronním programování. Je úzce propojena s LINQ a funkcionálním přístupem k programování. Umožňuje srozumitelněji komponovat asynchronní operace. Jádrem knihovny jsou typy `IObserver` a `IObservable` (viz Obrázek 6). Jak vypovídají názvy, jedná se o aplikaci vzoru Observer. `IObserver` poskytuje metody pro informování od `IObservable`. `IObservable` obsahuje metody pro registraci `IObserveru`. `IObservable` má podobný význam jako typ `IEnumerable`, který slouží jako reprezentace sekvence nebo nějakého seznamu. Rozdíl je v tom, že `IObservable` nemusí mít data, která vrací, zrovna k dispozici. `IObservable` pracuje s budoucími daty a informuje o nich `IObserver` pomocí jeho tří metod. Tyto budoucí zdroje dat lze komponovat v dotazech LINQ. Reactive Extension poskytuje různé operátory pro práci s `IObservable`. `IEnumerable` a `IEnumerator` jsou ve vztahu k `IObservable` a `IObserver`. Vztah lze popsat termíny push a pull. Data z `IEnumerable` jsou získávána (pull) za pomoci `IEnumerator`. Data z `IObservable` jsou poskytována (push) jednotlivým `IObserver`. Metoda `Subscribe` má asynchronní povahu.

To znamená, že podproces, který ji volá, není blokován, dokud není oznámeno ukončení od `IObservable` (OnCompleted).

```
public interface IObservable<out T>
{
    IDisposable Subscribe(IObserver<T> observer);
}

public interface IObserver<in T>
{
    void OnCompleted();
    void OnError(Exception error);
    void OnNext(T value);
}
```

Obrázek 6: Základní typy v knihovně Reactive Extensions

9 Použití Windows Azure služeb

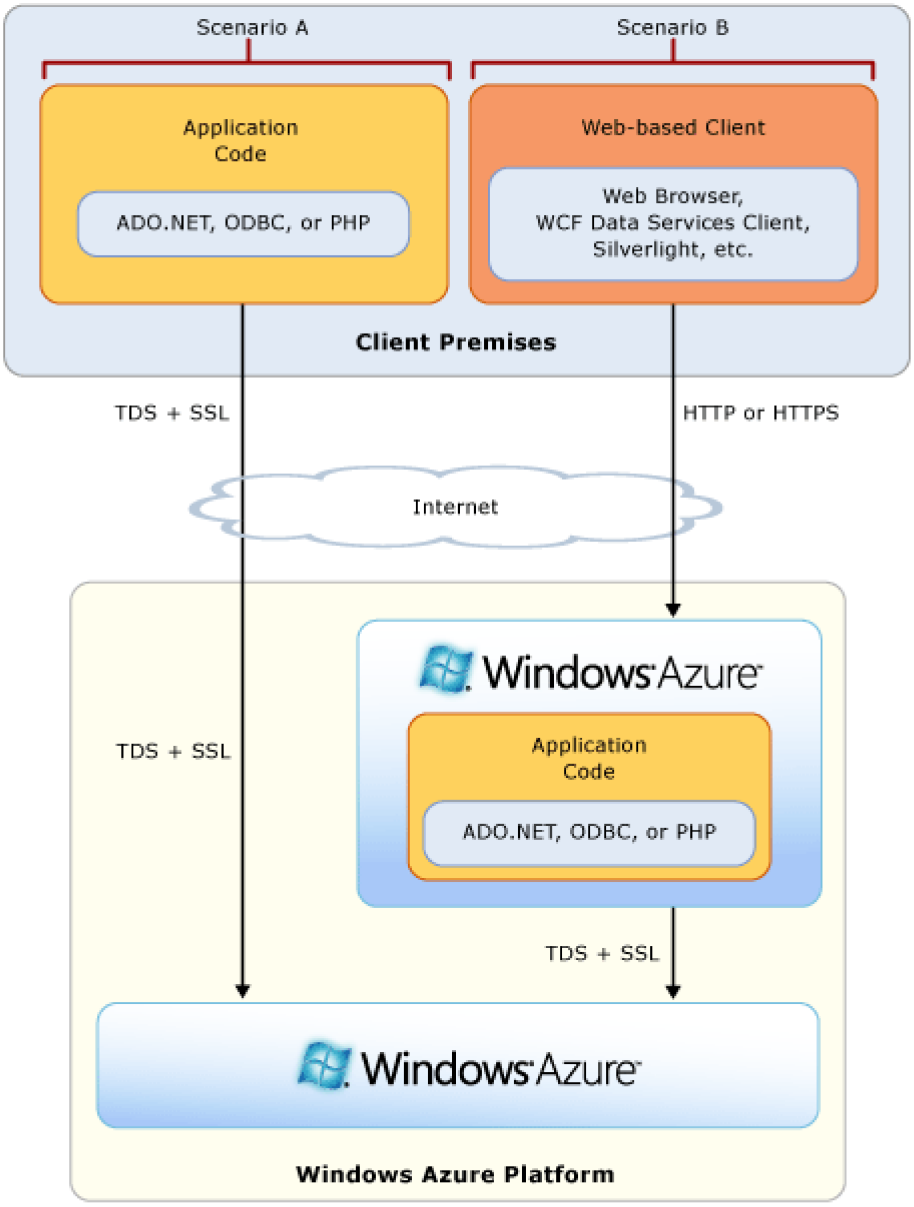
V této části je nastíněn přibližný výpočet předpokládaných nákladů na provoz online režimu z pohledu využívání Windows Azure. Náklady jsou uvedeny při využívání plánu Pay-As-You-Go. Dále jsou zde popsány kroky, kterými byly zprovozněny služby a provedená konfigurace. Náklady jsou vyjádřeny v dolarech a přepočtu na české koruny. Přepočet vychází z čtvrtletního průměrného kurzu koruny vůči dolaru v roce 2013, který je 19,375 CZK/USD [6].

Hlavní herní data budou ukládána do SQL databáze. K databázi lze přistupovat zevnitř cloudu prostřednictvím aplikací běžících v cloud službě. Také lze k databázi přistupovat zvnějšku přes internet (viz Obrázek 7).

Nastavení přístupu se provádí na úrovni SQL serveru pomocí pravidel pro firewall, která přístup omezují.

Windows Azure SQL Database služba je dostupná přes TCP port 1433 [10]. Musí být povolena odchozí komunikace přes tento port straně klienta. Z hlediska obecného přístupu různých klientů to není vhodné. Navíc by při přímém přístupu k databázi zvenčí vznikla vazba na určité technologie (např. ADO .NET). Přenos SQL dotazů přidává také dodatečné zpoždění ke komunikaci. Mimo jiné z uvedených důvodů je online režim postaven na komunikaci se službami běžícími na cloudu a komunikujícími uvnitř cloudu s databází.

Vnitřní přístup je implicitně povolen pro cloudové služby. Umožňuje to aplikacím v cloudu přistupovat k databázi.



Obrázek 7: Přístup k Windows Azure SQL Database

9.1 Výpočetní čas

V počátečním nastavení bude zprovozněna jedna instance webové role (viz kapitola 4.3.1), která bude sloužit jako hostitel WCF služeb (viz kapitola 6). WCF služby budou fungovat jako komunikační rozhraní pro klienty online režimu. Kontejner služby bude hoštěn na extra malém virtuálním stroji. Role poběží 24 h denně, takže výpočetní náklady na den budou 24*hodinová sazba. Tyto náklady jsou násobeny 31 krát a vyjdou náklady na měsíc. Tabulka 3 uvádí ceny za

výpočetní hodiny pro jednotlivé velikosti virtuálních strojů pro plán Pay-As-You-Go. Náklady za výpočetní čas vznikají, jakmile je aplikace nahrána na Azure, ať už je aplikace aktuálně uspaná nebo běží. Počítá se zde každá i jen započatá hodina (běžící více než 5 minut), takže je lepší nechat hodinu doběhnout do konce.

NÁZEV	VIRTUÁLNÍ JÁDRA	RAM	CENA ZA HODINU
Extra Small (A0)	Shared	768 MB	$0.02 (~$15/měsíc)
Small (A1)	1	1.75 GB	$0.08 (~$60/měsíc)
Medium (A2)	2	3.5 GB	$0.16 (~$119/měsíc)
Large (A3)	4	7 GB	$0.32 (~$238/měsíc)
Extra Large (A4)	8	14 GB	$0.64 (~$476/měsíc)

Tabulka 3: Standardní velikost virtuálních strojů (Pay-As-You-Go plán, při 744 hodin na měsíc).[31]

Z tabulky 3 lze vyčíst, že pro virtuální stroj Extra Small budou předpokládané výpočetní náklady rovny 15$/měsíc. To bude činit měsíčně 291 CZK.

9.2 Datový objem

V případě použití Windows Azure Storage se náklady odvíjejí na základě využití místa úložiště (Blob, Table, Queue) dostupného účtu a podle počtu transakcí – počtu zápisů a čtení z úložiště. Dále je rozlišena cena za geograficky redudantní a lokálně redudantní data.

Cena za transakce činí 0,01$ za 100,000 transakcí.

KAPACITA STORAGE	GEOGRAFICKY REDUDANTNÍ	LOKÁLNĚ REDUDANTNÍ
První 1 TB / měsíc	$0.095 za GB	$0.070 za GB
Dalších 49 TB / měsíc	$0.08 za GB	$0.065 za GB
Dalších 450 TB / měsíc	$0.07 za GB	$0.06 za GB
Dalších 500 TB / měsíc	$0.065 za GB	$0.055 za GB
Dalších 4,000 TB / měsíc	$0.06 za GB	$0.045 za GB
Dalších 4,000 TB / měsíc	$0.055 za GB	$0.037 za GB
Přes 9,000 TB / měsíc	Kontaktovat Azure	Kontaktovat Azure

Tabulka 4: Ceny Windows Azure Storage (při 744 hodin na měsíc, 1 TB = 1,024 GB) [9]

V aplikaci se prozatím s použitím Azure Storage nepočítá, a tudíž náklady zde by měly být nulové nebo minimální možné.

Následují náklady na SQL Database. SQL Database je dostupná ve dvou edicích. V edici Web a Bussiness. Web edice může mít až 5 GB velkou databázi a Bussiness edice podporuje až 150 GB databázi. Od velikosti se potom odvíjejí náklady. Konkrétní čísla měsíčních nákladů jsou uvedena v tabulce 5. Databáze je účtována podle počtu dní, po které existovala.

VELIKOST DATABÁZE	CENA ZA DATABÁZI ZA MĚSÍC (ZAPOČTE SE POMĚRNÁ DENNÍ ČÁST)	
0 až 100 MB	$4.995	
100 MB až 1 GB	$9.99	
1 GB až 10 GB	$9.99 za první GB	$3.996 za každý další GB
10 GB až 50 GB	$45.96 za prvních 10 GB	$1.996 za každý další GB
50 GB až 150 GB	$125.88 za prvních 50 GB	$0.999 za každý další GB

Tabulka 5: Ceny SQL Database [8]

Následují náklady na transfer dat. Transfer dat je přenos dat směrem dovnitř datacentra, směrem ven z datacentra nebo v rámci datacentra. V rámci transferu dat jsou účtovány pouze přenosy ven z datacentra. Přenosy dat v rámci jednoho datacentra a data, která jsou posílána do datacentra, jsou zdarma. Tabulka 6 uvádí přehled cen datových transferů. V tabulce Zóna 1 a Zóna 2 odpovídají skupinám sub-regionů. Sub-region je nejnižší jednotka, kterou lze vybrat pro geo-lokaci aplikačních dat.[8]

Zóně 1 odpovídají sub-regiony East US, West US, North Central US, South Central US, North Europe, Western Europe.[8]

Zóně 2 pak sub-regiony East Asia and Southeast Asia.[8]

ODCHOZÍ DATOVÉ TRANSFERY	ZÓNA 1	ZÓNA 2
Prvních 5 GB / měsíc	Zdarma	Zdarma
5 GB - 10 TB / měsíc	$0.12 za GB	$0.19 za GB
Dalších 40 TB / měsíc	$0.09 za GB	$0.15 za GB
Dalších 100 TB / měsíc	$0.07 za GB	$0.13 za GB
Dalších 350 TB / měsíc	$0.05 za GB	$0.12 za GB
Více než 500 TB / měsíc	Kontaktovat Azure	Kontaktovat Azure

Tabulka 6: Ceny za datové transfery v rámci Windows Azure datacenter [32] (při 744 hodin na měsíc, 1 TB = 1,024 GB)

Předpokládaný maximální počet uvažovaných uživatelů bude omezen velikostí databáze. Databáze je jako téměř vše na cloudu taktéž předmětem škálování. Je možné používat příkazu ke změně velikosti databáze (ALTER DATABASE). To je ale limitováno kapacitou dostupnou v rámci edice (web, bussiness). Dále lze využít techniku federalizace. Federalizace využívá tzv. sharding.

Sharding znamená, že z jedné databáze se data rozdělí do více databází (shardy), které mají stejné schéma jako originální databáze. Data jsou distribuována tak, že datový řádek je obsažen pouze v jednom shardu, a tudíž kombinace dat jednotlivých shardů vytvoří originální databázi. Soubor shardů je brán jako jedna logická databáze. [33]

Pro databáze bude použita web edice s počáteční velikostí 1 GB. Tedy bude měsíčně vytvářet náklady 9,99$, což činí 194 CZK/měsíc. Dále se bude nacházet v zóně 1 pro datové transféry.

Přepokládané transfery dat je možné jen velmi přibližně odhadnout. Vstupuje zde do výpočtu počet klientů služeb. Snažší je vyjít z nějakého spíše nedosažitelného maxima. Zde bude vycházeno z maximálního transferu v rámci jednotek TB. Konkrétně při 2 TB dat to činí náklady na transfery 245$, což je v přepočtu na CZK 4747 Kč.

Výsledné náklady na provoz online režimu jsou poté součet všech uvedených nákladů (viz Tabulka 7). Spočítané náklady jsou pouze předpokládané. Náklady vycházejí z plánu Pay-As-You-Go.

POPIS NÁKLADU	VÝŠE NÁKLADU NA MĚSÍC
Virtuální stroj pro instanci web role	291 CZK
SQL databáze	194 + 4747 = 4941 CZK
Celkem	5232 CZK

Tabulka 7: Výpočet nákladů na provoz online režimu

Pro srovnání vypočtených nákladů existuje pro Windows Azure kalkulačka nákladů [18]. Tato kalkulačka při uvedených vstupních datech vypočetla náklady $263,79/měsíc, což je v přepočtu 5111 CZK.

9.3 Fakturační účtování

V kapitole o charakteristikách cloud computingu bylo řečeno, že cloud je měřená služba. Na základě těchto měření může být poté zákazník fakturován. V rámci fakturace ve Windows Azure jsou používány určité přepočty pomocí specifikovaných jednotek. Pro výpočetní hodinu je jednotkou virtuální stroj velikosti Small (viz Tabulka 8). Zákazníkovi se poté účtuje počet jednotek, takže 1 Medium instance bude účtována jako 2 jednotky.

INSTANCE SLUŽBY (CLOUD SERVICES INSTANCE)	POČET HODIN (CLOCK HOURS)	PŘEPOČET ČASU NA SMALL INSTANCE
Extra Small (A0)	1	1/2 hodiny
Small (A1)	1	1 hodina
Medium (A2)	1	2 hodiny
Large (A3)	1	4 hodiny
Extra Large (A4)	1	8 hodin

Tabulka 8: Přepočet výpočetních hodin při fakturaci za služby Windows Azure [31]

Databáze je účtována pomocí databázových jednotek (DU). Jedna DU odpovídá 9,99$/měsíc [8].

VELIKOST DATABÁZE	DATABÁZOVÉ JEDNOTKY (KAŽDÁ DU = $9.99)
0 až 100 MB	0.5 DU
100 MB až 1 GB	1 DU
1 GB až 10 GB	1 DU za první GB (0.4 DU za každý další GB)
10 GB až 50 GB	4.6 DU za prvních 10 GB (0.2 DU za každý další GB)
50 GB až 150 GB	12.6 DU za prvních 50 GB (0.1 DU za každý další GB)

Tabulka 9: Přepočet velikosti databáze při fakturaci za služby Windows Azure [8]

Část vyučtování pro databázi je na obrázku 8. Zde bylo spotřebováno 0,209677 DU.

Billing Period	Name	Type	Resource	Region	SKU	Consumed	Included
201305(4/23/2013 - 5/22/2013	"SQL Database"	"Web Edition"	"Database Units"		"8YD-00001"	0.209677	1

Obrázek 8: Ukázka výměry databázových jednotek při fakturaci

Část vyučtování výpočetních hodin je na obrázku 9. Bylo vyúčtováno 60 hodin.

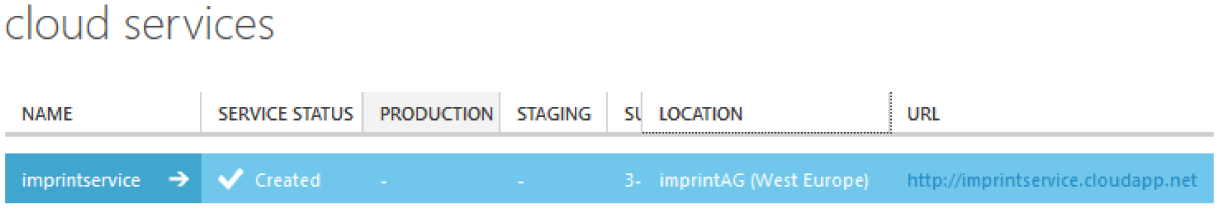
Billing Period	Name	Type	Resource	Region	SKU	Consumed	Included
201304(3/23/2013 - 4/22/2013	"Cloud Services"		"Compute Hours"		"7UD-00001'	60	750

Obrázek 9: Ukázka výměry výpočetních hodin při fakturaci

K vyúčtování samozřejmě patří také cena. V tomto případě byly ceny nulové, protože se jednalo o fakturaci v rámci zkušební doby (trial).

9.4 Vytvoření účtu

K používání Windows Azure bylo potřeba vytvořit účet. Tento účet je používán ke konfiguraci služeb a na fakturaci služeb poskytovatelem. V rámci tohoto účtu byly dále vytvořeny cloud služba, účet úložiště, SQL databáze.



Obrázek 10: Ukázka vytvořené cloud služby

Po vytvoření kontejneru je možné nahrát aplikaci. Aplikace je nahrána jako tři soubory. Jedním je samotný soubor s aplikací (aplikační balíček s koncovkou *.cspkg). Aplikační balíček je zkompilovaný program. Je to „obraz“, který může Windows Azure nahrát na virtuální stroj ke spuštění. Dále se poskytují dva konfigurační soubory – ServiceDefinition.csdef a ServiceConfiguration.cscfg. Jejich účelem je popsat model služby (service model [28]).

Soubor ServiceDefinition.csdef obsahuje výpočetní role (viz kapitola 4.3.1), které služba využívá, a jejich vlastnosti. Vlastnost může být například velikost virtuálního stroje. Soubor také určuje, jaké porty role využívá (například port 80 pro HTTP).

Soubor ServiceConfiguration.cscfg obsahuje konfiguraci počtu instancí jednotlivých rolí. Dále obsahuje hodnoty pro konfiguraci účtu úložiště (identifikační údaje) a další. Obrázek 11 ukazuje stručný příklad takové konfigurace, který říká, že má být vytvořena jedna instace

```
ServiceConfiguration.cscfg

<?xml version="1.0"?>
<ServiceConfiguration serviceName="Name" xmlns="">
  <Role name="WebRole1">
    <Instances count="1" />
      ...
  </Role>
</ServiceConfiguration>
```

Obrázek 11: Ukázka ServiceConfiguration.csfg

webové role, která má název WebRole1.

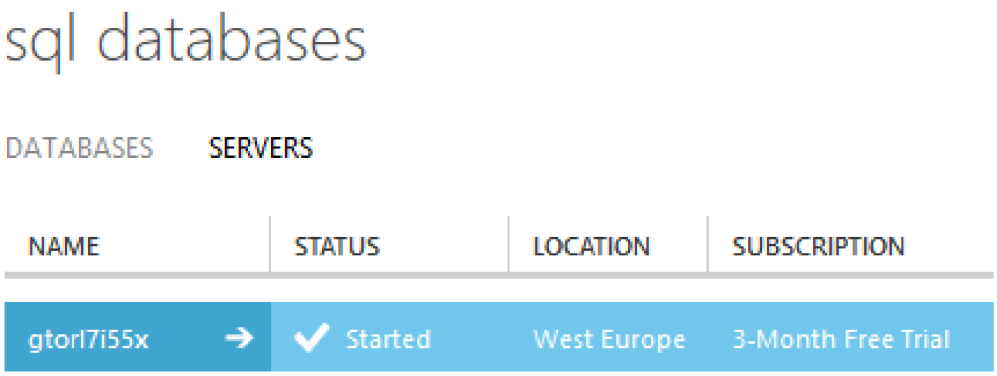
Konfigurační hodnoty v ServiceConfiguration.cscfg mohou být měněny za běhu, kdežto změny v ServiceDefinition.csdef nelze provést bez znovunahrání aplikačního balíčku. Z tohoto důvodu je vhodné umístit konfiguraci, která se mění v souboru ServiceConfiguration.cscfg.

V rámci testování aplikace lze využít ladicího prostředí. Aplikace může být nahrána do ladicího nebo produkčního prostředí.

9.5 Vytvoření SQL Database

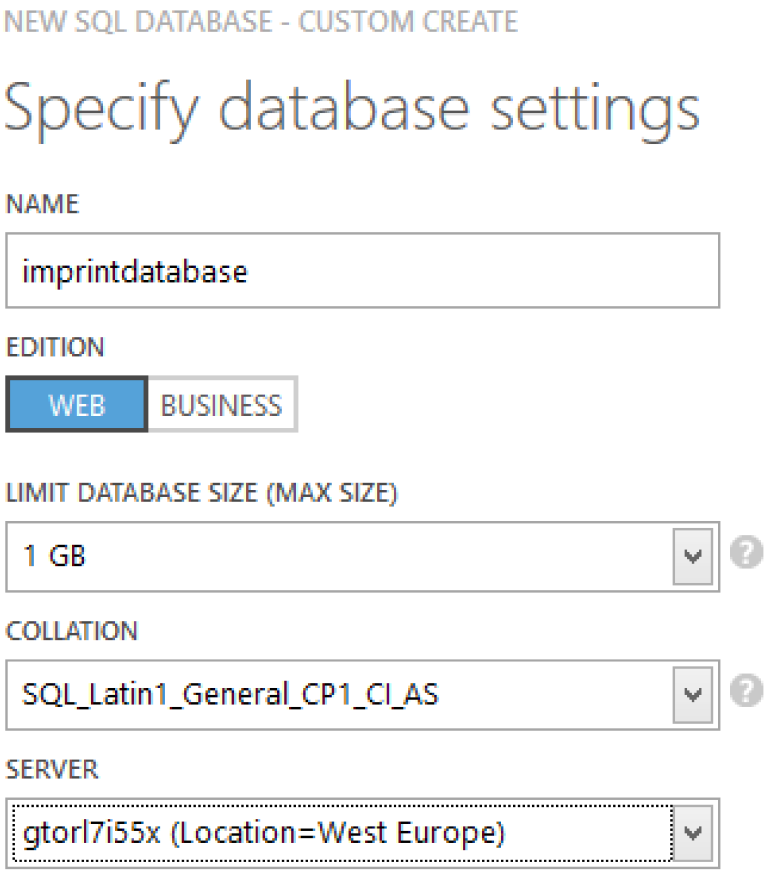
Nejdříve bylo potřeba vytvořit SQL Database server (viz Obrázek 12). SQL Database server tvoří logickou skupinu databází a slouží jako centrální bod jejich správy. Každý server má přidělenu vlastní unikátní adresu ve tvaru servername.database.windows.net.

Vytvoření lze učinit přes portál účtu, nebo pomocí REST API (viz kapitola 5.2). Při vytváření serveru je vyžadován údaj login, který bude použit jako hlavní účet k administraci serveru. Server může obsahovat až 150 databází (včetně databáze master). Při procesu je automaticky vytvořena databáze master. Ta slouží pro evidenci uživatelských účtů, které mají právo vytvářet a mazat databáze, nebo vytvářet nové účty. Při tvorbě databáze je potřeba být připojen právě k databázi master. [19]



Obrázek 12: Vytvoření SQL Database serveru (virtuální server)

Následuje vytvoření databáze (viz Obrázek 13) na serveru a pomocí skriptů (viz kapitola 11.4) SQL se vytvoří schéma databáze a data.



Obrázek 13: Vytvoření SQL Database

10 Pravidla online režimu

V této kapitole je popsáno, jaké jsou možnosti v online režimu hry a jaká jsou zde herní pravidla. Do online režimu patří tvorba hráčova profilu, vytváření hráčských aliancí a používání krajiny světa (například k vytváření hráčových měst).

10.1 Profil hráče

Součástí hry je vytvoření profilu hráče. Hráč musí zadat jedinečné jméno a bude mu vytvořen profil. Po vytvoření profilu může hráč začít hrát. V rámci profilu může zjistit následující informace:

- Hráčova města
- Aliance, kterých je členem
- Vybrané statistiky ekologické stopy (například min-max ekologická stopa budov)

10.2 Aliance

Hráči mohou během hry vstupovat do aliancí, což jsou seskupení několika hráčů. Z aliančního spojení plynou ve hře výhody i nevýhody. Hráči v alianci se vzájemně nepřímo ovliňují. To znamená, že akce jednoho hráče se mohou projevit na jiných hráčích, kteří jsou s daným hráčem v alianci.

10.2.1 Vytvoření aliance a členství

Vytvoření aliance probíhá tak, že zakládací hráč založí alianci s jím určeným jedinečným jménem pro alianci. Poté tento hráč může rozeslat pozvánky hráčům, které si přeje do aliance zahrnout. Hráči nemohou vstoupit do aliance bez pozvání od zakladatele, nebo hráče oprávněného zasílat pozvánky do dané aliance. Přijetím pozvání do aliance se hráč stává členem. Dále hráč může z aliance vystoupit, nebo z ní být vyřazen.

Okamžikem, kdy je aliance bez hráčů, se stává předmětem automatického zrušení.

10.2.2 Vliv aliance na členy

Jak bylo řečeno, v rámci aliance se hráči vzájemně ovlivňují pozitivně nebo negativně. Ve hře je stanoveno pravidlo pozitivního efektu na příjmy aliančních členů. To se stane tehdy, když

jeden hráč aliance postaví budovu, která všem pozitivně ovlivní příjem nějakých surovin. Je také stanoveno pravidlo negativního působení na alianční členy. To je, když je hráč aliance ovlivněn zvenčí nepřátelským hráčem.

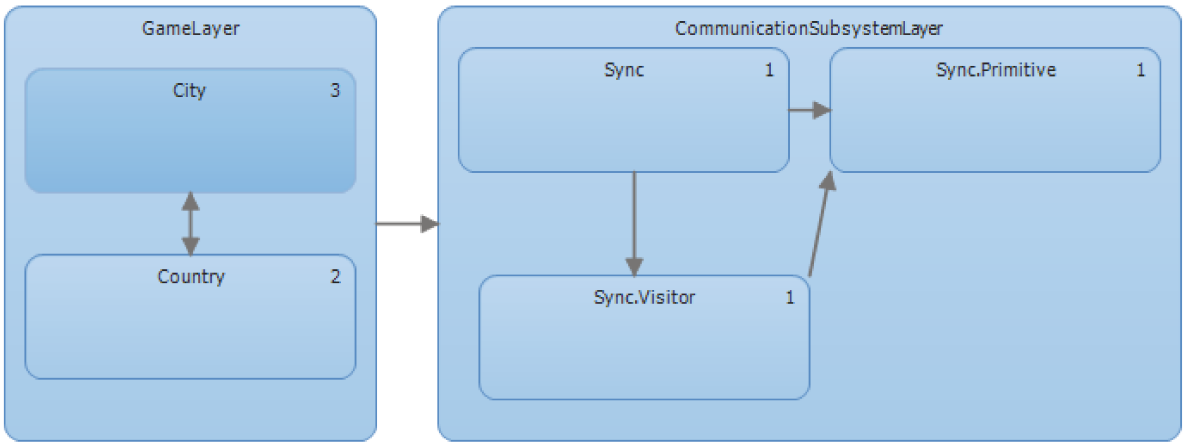
10.3 Krajina světa

V krajině světa se odehrává hra v rámci většího prostoru. Mimo tuto krajinu se hra odehrává také ve městech. Krajina světa je rozdělena na sektory. Hráči mají možnost založit město na dostupném sektoru. Hráči mohou na krajině zjišťovat informace o jednotlivých sektorech. Mohou tak zjistit, jakému hráči daný sektor patří nebo co se na sektoru nachází.

11 Návrh

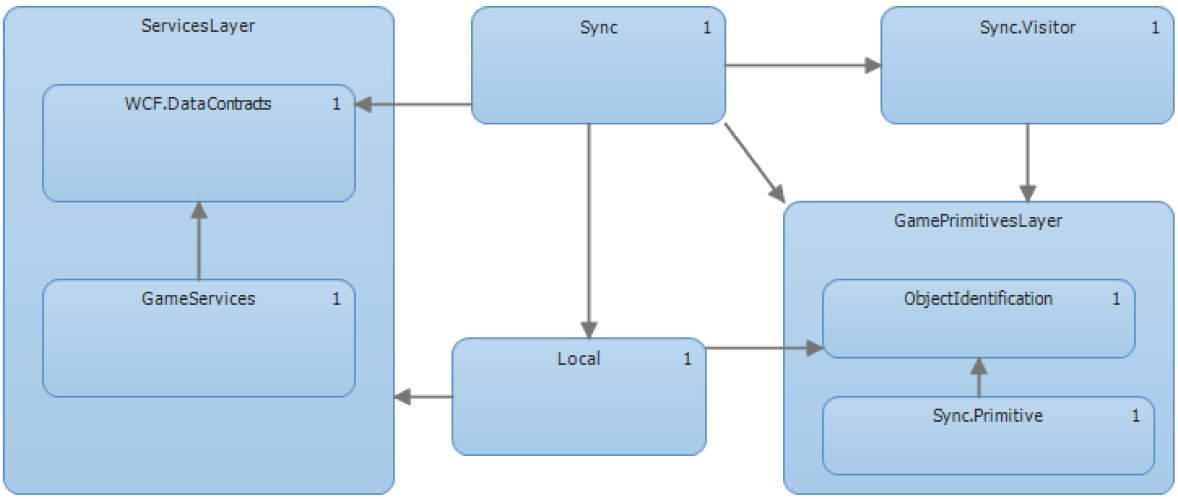
Návrh byl proveden na úrovni mobilní aplikace a na úrovni služeb.

Návrh v mobilní aplikaci byl rozdělen do vrstev (viz Obrázek 14, Obrázek 15), které separují závislosti. Výhodou je, že v prostředí **Visual Studio** je možné vrstvový model validovat.



Obrázek 14: Návrh vrstev pro komunikaci se službami v mobilní aplikaci

Vrstvy zachycují, jak by měly vypadat programové závislosti ve hře.



Obrázek 15: Návrh vrstev v rámci širší komunikace v mobilní aplikaci

V rámci návrhu byly navrženy následující knihovny tříd.

`GameServices` – obsahuje vygenerované proxy pro konečnou komunikaci se službami.

`WCF.DataContracts` – obsahuje datové kontrakty používané při komunikaci.

`Local` – vrstva nad `GameServices`. Poskytuje třídy, které umožňují zjednodušenou komunikaci se službami. Dále poskytuje doplňující funkce ukládání přijatých dat a ukládání dat k odeslání. Používá datové třídy z `WCF.DataContracts`.

`Sync` – vrstva nad `Local`. Převádí data mezi třídami z `WCF.DataContracts` a herními třídami.

`Sync.Primitive` – knihovna základních typů použitých dále ve hře, které jsou spjaty s daty v online komunikaci a jejichž data jsou v rámci knihovny `Sync` převáděna do tříd datových kontraktů.

`Sync.Factory` – obsahuje rozhraní podle návrhového vzoru Factory. Instance některých tříd jsou vytvářeny pomocí těchto rozhraní.

`Sync.Visitor` – obsahuje třídy navržené podle návrhového vzoru Visitor. Tento princip je využit pro třídy herních operací.

11.1 Proxy

Proxy zajišťují samotnou komunikaci se službou. Jsou vygenerovány nástrojem slsvcutil [41]. Tento nástroj vygeneruje třídu proxy s operacemi odpovídajícími voláním služby. Operace třídy lze využívat jednak podle asynchronního návrhového vzoru APM (Asynchronous programming model) a jednak také podle vzoru EAP (Event-based asynchronous pattern).

APM definuje pro každou operaci dvojici operací.

Jedna z dvojice je ve tvaru

`Begin<Operace>(<parametryOperace>, AsyncCallback, object),`

kde údaj ve špičatých závorkách bude název operace, a vrací typ `IAsyncResult`. `IAsyncResult` se nachází ve jmenném prostoru

`System.Runtime.Remoting.Messaging`. Parametr typu `AsyncCallback` je určen k specifikaci operace, která se zavolá při ukončení asychronní operace.

Druhá z dvojice operací má tvar

```
End<Operace>(<refOutParametry>, IAsyncResult)
```

a vrací návratový typ původní neasynchronní operace. Fungují tak, že operace `BeginOperace` zahájí asynchronní operaci a vrátí instanci typu `IAsyncResult`, v podstatě reprezentující běžící asynchronní operaci. V okamžiku volání `EndOperace` s parametrem, kterým musí být platná instance (platná je, pokud odpovídá instanci získané z předchozího volání `BeginOperace`) `IAsyncResult`, mohla být operace ukončena a žadatel dostává výsledek. Pokud nebyla ukončena, je žadatel blokován do ukončení operace. Případné vzniklé výjimky jsou vyvolány v podprocesu volajícím `EndOperace`.

EAP je založený na posílání událostí. Třída aplikující tento vzor definuje operace ve tvaru

```
<operace>Async(<parametryOperace>)
```

a k nim odpovídající události `On<operace>Completed`. Klient se poté musí před zavoláním operace přihlásit k odběru události a zavolat asynchronní metodu. Po dokončení asynchronní operace je informován pomocí vyvolání události.

11.2 Komunikační subsystém

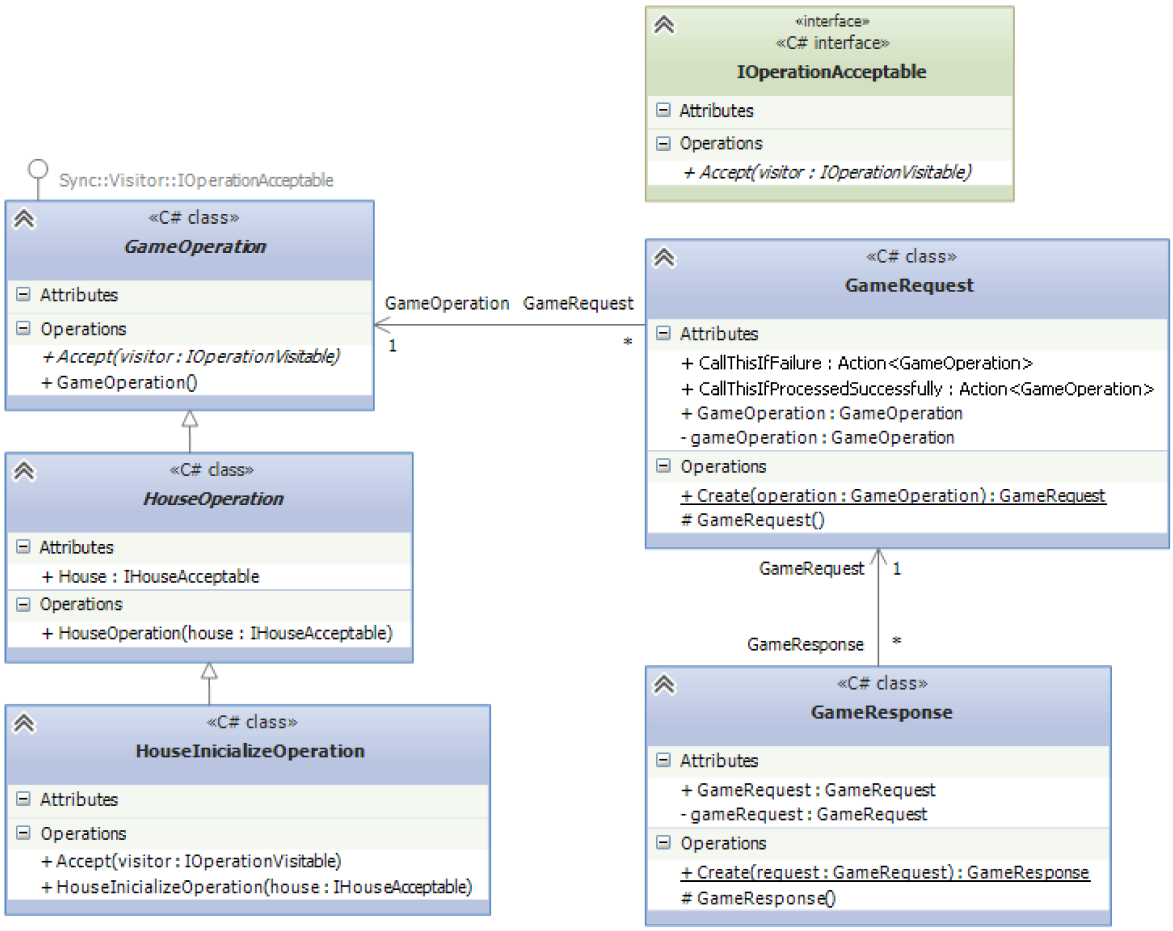
Komunikační subsystém (tvoří ho skupina spolupracujících tříd) v rámci aplikace je realizován pomocí herních operací (viz Obrázek 16). Každá herní operace má samopopisný (pro aplikaci) název (například HouseInitializeOperation) a nese data potřebná k jejímu provedení. Subsystém je schopen herní operace zpracovávat a informovat o jejich dokončení (viz Obrázek 17). Pomocí tohoto subsystému herní jádro (vrstva GameLayer) používá online režim. Herní jádro využívá pouze vrstvu CommunicationSubsystemLayer.

Následuje podrobnější popis fungování subsystému. V širším pojetí tvoří rozhraní subsystému třídy hracích operací a rozhraní `IServicePoint`. Subsystém využívá návrhového vzoru návštěvník (visitor [16]). Konkrétně se využívá dvojího volání k rozlišování konkrétního typu herní operace (jedná se o techniku double-dispatch [42]).

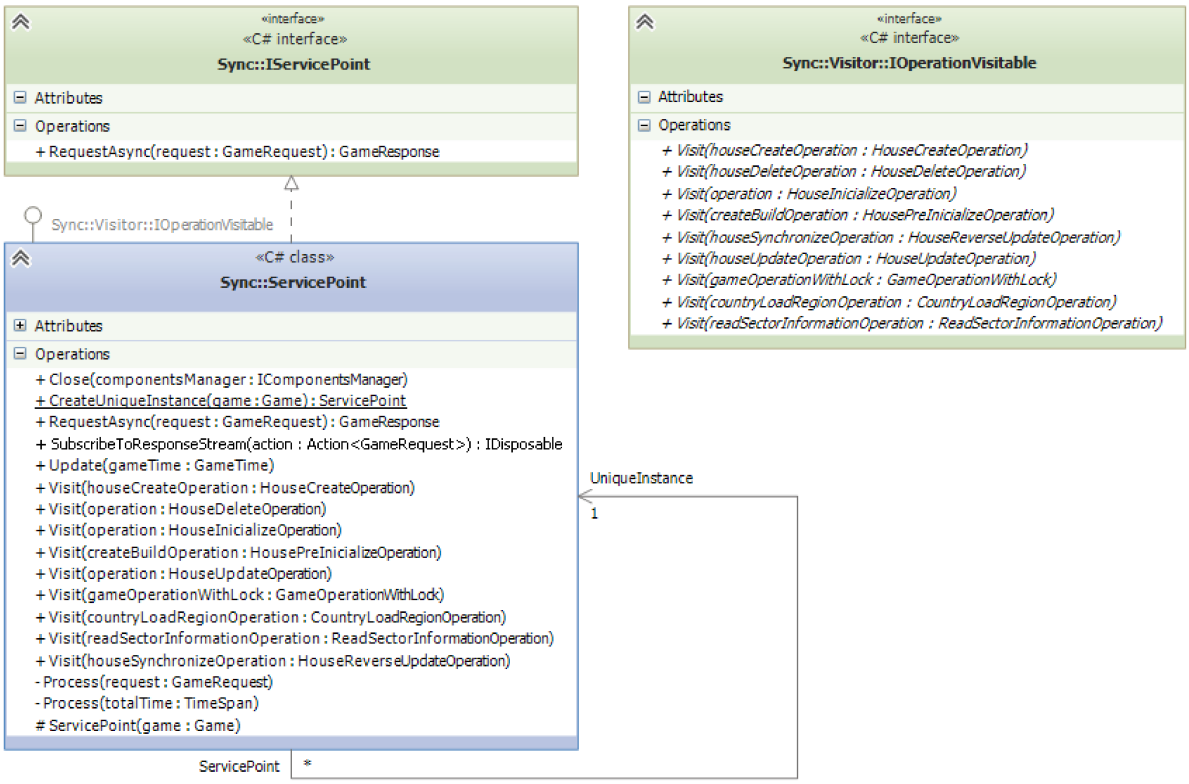
Hra využívá třídu `GameRequest`, do jejíž instance vloží instanci herní operace (třída `GameOperation`), která se má provést. Následně instanci `GameRequest` předá metodě `RequestAsync` rozhraní `IServicePoint` a tato realizuje kroky k vyřízení požadavku. Požadavky (`GameRequest`) jsou vyřizovány serializovaně jeden po druhém. Jakmile je požadavek vyřízen, je zavolán příslušný delegát typu `Action<GameOperation>` třídy `GameRequest`, pokud má přiřazenu hodnotu. Delegát má název `CallThisIfFailure` nebo `CallThisIfProcessedSuccessfully` podle toho, jestli se jedná o neúspěšné, resp. úspěšné vyřízení požadované operace.

Zpracování herní operace realizuje třída, která implementuje rozhraní `IOperationVisitable`. Herní operace (třída `GameOperation`) implementují rozhraní `IOperationAccetable`. Pomocí této dvojice rozhraní je realizován vzor Visitor.

Při vyřizování herní operace je volána metoda `Accept` instance této operace s argumentem instance `IOperationVisitable`. Uvnitř metody `Accept` je zavolána metoda `Visit` instance `IOperationVisitable` s argumentem instance konkrétního typu herní operace (využití klíčového slova `this`). Tím došlo k druhému volání, kde bylo možné určit konkrétní typ herní operace, a tím i konkrétní metodu rozhraní `IOperationVisitable`.



Obrázek 16: UML - koncept herních operací



Obrázek 17: UML - koncept zpracování herních operací

11.3 Návrh služeb

V této kapitole je uveden návrh služeb, které hra využívá při online komunikaci.

Rozhraní služeb je navrženo jednak na základě akcí, které mohou vykonávat hráči, jednak na základě potřeb synchronizace dat mezi hráči a úložištěm. Synchronizace je akce automaticky prováděná mobilní aplikací v určitém čase nebo časovém intervalu.

Bylo uvedeno, na jakém základě jsou akce vytvořeny. Navíc mohou být akce ještě nějak omezeny (například právem), ale protože vše bude řídit aplikace, není zde žádné omezení řešeno. Samotné rozhraní služeb je navržené dle RPC API (viz kapitola 5.2). Většina operací služeb má parametr, kterým je jméno hráče. Akce jsou:

- Získej město,
- Vytvoření aliance,
- Vytvoření hráče,
- Vytvořit město,
- Vlož budovy ve městě,

- Vymaž budovy ve městě,
- Synchronizuj budovy,
- Synchronizuj město,
- Pozvat hráče do aliance,
- Reagovat na pozvánku do aliance,
- Opustit alianci,
- Načíst sektory světa,
- Vyhledat město,
- Získávat informace o herním prostředí obecně,
- Obecné akce budov definované uvnitř herní aplikace (identifikované názvem a budovou, které se týkají)
- Ovlivnit příjem hráčů v alianci.

Na základě předcházejících informací byly navrženy služby pro čtyři oblasti:

- Služba CityService, kde probíhají operace na úrovni jednotek město a budovy, obecných akcí budov.

- Služba PlayersService, kde probíhají operace v rámci hráčů a aliancí.

- Služba WorldService, kde se získávají informace o herním světě a herním prostředí.

- Služba PlayersProfileService, kde lze získat informace o hráči.

Mimo to byla navržena služba KeyService, která poskytuje identifikaci novým objektům. Například před vložením budovy si klientská aplikace vyžádá identifikační klíč, který bude sloužit jako ID budovy.

Pro všechny služby byly vytvořeny nebo stanoveny WCF kontrakty služby, datové kontrakty, chybové kontrakty, konkurenční chování, mód instancí, koncové body. (viz kapitola 6 pro vysvětlení pojmů). Mód instance je PerSession a konkurenční chování je Single. Je uveden kontrakt pro službu WorldService (viz Obrázek 18) a dílčí datakontrakt této služby (viz Obrázek 19).

```
[ServiceContract(Namespace = "GameServices")]
public interface IWorldService
{
    [OperationContract]
    Sector LocateCity(string playerId, string cityId);
    [OperationContract]
    Sector[] LoadSectors(string playerId, RegionSpecification region);
    [OperationContract]
    void OccupySector(string playerId, Sector area);
    [OperationContract]
    void CreateCity(string playerId, string cityId, Sector sector);
}
```

Obrázek 18: Ukázka kontraktu služby

```
[DataContract]
public class RegionSpecification
{
    [DataMember]
    public int MinX { get; set; }
    [DataMember]
    public int MaxX { get; set; }
    [DataMember]
    public int MinY { get; set; }
    [DataMember]
    public int MaxY { get; set; }
}
```

Obrázek 19: Ukázka datakontraktu

V příloze 2 je možno si prohlédnout všechny ostatní kontrakty.

11.4 Návrh databáze

Úložiště bude řešeno pomocí relační databáze. Důvodem jsou výhody, které relační databáze poskytuje. Jsou to mimo jiné transakce, indexy, datová integrita, SQL jazyk.

V rámci návrhu databáze je řešena oblast navržení schématu databáze, databázových objektů (jako jsou například indexy) a konkurenčního přístupu.

11.4.1 Konkureční přístup

V rámci konkurenčního přístupu k databázi mohou vznikat situace, kdy uživatelé provádějí změny na stejných datech. Z toho poté mohou plynout různé datové konflikty. Entity Framework je postavený na odpojeném modelu dat a používá optimistický přístup k řešení datových konfliktů. Dokáže například využívat nějaký datový sloupec, který udává, o jakou jde změnu, a v případě konfliktu dává programátorovi možnost řešení. Pro datový sloupec lze například využít typ rowversion v MSSQL nebo nějaký časový typ (datum a čas).

V rámci online režimu vzniká problém pouze na úrovni zápisu. Při čtení není klient blokován a vše se řídí pravidly výchozí izolační úrovně Windows Azure SQL databáze. Při zápisu je klient blokován podle toho, zda je potřeba čekat na získání exkluzivního zámku. Takové případy zde určitě vzniknou a bude otázkou, jak rychle bude databáze zvládat vyřizovat požadavky.

V rámci zápisu jsou dále kontrolována časová razítka. Časová razítka jsou časy změny dat a jsou uvedena pouze v některých tabulkách. Pokud by časová změna byla starší než aktuálně uložená v databázi, tak se požadavek nemůže provést a operace se musí přerušit.

11.4.2 Schéma databáze

Návrh databáze vychází z několika funkčních požadavků.

- Hra bude obsahovat svět, který se bude skládat ze sektorů.
- Sektor může obsahovat krajinu nebo město.
- Město obsahuje budovy. Budovy generují příjem surovin (včetně ekologické stopy) a také odebírají zdroje. Město má také určité vlastní zdroje a také poskytuje některé suroviny.
- Hráč vykonává akce v herním světě – zakládá města, čte informace, vytváří a mění budovy, uzavírá aliance, posílá pozvánky do aliance, obecné akce budov.
- Hráč má herní level.

Na základě těchto požadavků lze stanovit několik datových entit do databáze a doplnit jim požadované atributy (viz Tabulka 10).

DATOVÁ ENTITA	ATRIBUTY
Sektory	typ sektoru, pozice
Města	vlastník, název, sektor, budovy, zdroje surovin, poskytování surovin
Budovy	typ budovy, popis, název, město, příjem surovin, spotřeba surovin, pozice budovy, level
Hráči	jméno, stát, aliance, města, budovy, level
Suroviny	název, popis, standardní jednotka
Aliance	jméno, hráči, zakladatel
Pozvánky	hráč zvoucí, hráč zvaný, platnost
Akce budov	název akce, budova, čas začátku platnosti, čas konce platnosti

Tabulka 10: Datové entity databáze

Kardinalita vztahu mezi dvěma tabulkami určuje, kolik záznamů v jedné tabulce je ve vztahu k záznamům druhé tabulky. Například vyjádření vztahu, že město se nachází na jednom sektoru, je dáno vztahem 1:1. Jeden záznam z tabulky Sektory je ve vztahu s jedním záznamem v tabulce Města. Tabulka 11 popisuje kardinalitu mezi uvedenými entitami.

VZTAH ENTIT	KARDINALITA VZTAHU
Města : Sektory	1:1
Města : Budovy	1:N
Města : Suroviny (zdroje, poskytování)	M:N
Budovy : Suroviny (zdroje, příjmy)	M:N
Hráči : Města	1:N
Hráči : Aliance	N:1
Hráči : Pozvánky	1:N
Hráči : Sektory	1:N
Akce budov : Budovy	M:N

Tabulka 11: Kardinalita vztahů databáze

Databázové tabulky byly dále rozděleny normalizací. Konkrétní podoba tabulek je v příloze 1.

CIZÍ A PRIMÁRNÍ KLÍČE

Pro každou tabulku byl stanoven primární klíč. Kde to bylo možné, byly klíče stanoveny za využití přirozeného klíče. Například primární identifikace města je provedena podle jeho jména. U některých tabulek jsou číselné primární klíče, u některých jsou složené primární klíče z kombinace cizích klíčů. V rámci MSSQL 2012 lze navíc využít také databázové sekvence, nebo sloupec identity pro automatické vytváření hodnoty klíče.

KOLACE DATABÁZE

Kolace se váže k textovým datům používaným v databázi. Je to sada pravidel, které určují, jak budou textová data řazena a porovnávána.

Databáze používá kolaci SQL_Latin1_General_CP1_CI_AS. Rozebráním názvu kolace se dostaneme k jednotlivým pravidlům. Název kolace je SQL_Latin1_General_CP1_CI_AS. Latin1 udává jazyky západní Evropy. CP1 (code page) udává kódovou stránku. CI (case-insensitive) udává, že se nerozlišují malá a velká písmena. AS (accent-sensitive) udává, že se rozlišuje akcent. Jiným příkladem názvu kolace je například Finnish_Swedish_CS_AS.

V rámci databáze je dále pro všechny textové atributy použit Unicode datový typ (Nvarchar, Nchar apod.)

REFERENČNÍ INTEGRITA

Pravidlo referenční integrity říká, že databáze nesmí obsahovat žádné cizí klíče, které nelze přiřadit k rodičovské tabulce [29]. Zajišťuje kontrolu používaných cizích klíčů u dceřinných tabulek. Jinými slovy omezuje hodnoty vkládané do dceřinných tabulek na existující hodnoty v rodičovské tabulce. V databázi lze dále definovat referenční akce cascade delete a cascade update. Tyto akce jsou prováděny automaticky databázovým systémem. Cascade delete lze využít, když dochází při mazání rodičovského záznamu k potřebě vymazat přidružené záznamy. Něco obdobného platí i pro pravidlo cascade update.

V databázi je stanoveno referenční pravidlo cascade delete pro vztahy uvedené v tabulce 12.

Města→Budovy
Města→Suroviny
Budovy→Suroviny.
Města → Sektory

Tabulka 12: Cascade delete v databázi

Pravidlo cascade update se nevyužívá.

INDEXY

Indexy mohou být v databázi vytvořeny automaticky nebo uživatelem.

V rámci databáze jsou nad primárními klíči automaticky vytvořeny unikátní klastrované indexy (pokud se nespecifikuje neklastrovaný index [43]). Doporučená pravidla pro klastrovací klíč jsou, že by měl být unikátní, úzký (počtem bytů), statický (nemění se) [26]. Tato pravidla se hlavně vztahují na situaci, kdy je nějaký neklastrovaný index závislý na klastrovaném. V interní reprezentaci neklastrovaných indexů se totiž duplikují klastrovací klíče, a proto by dodržování těchto pravidel vedlo k celkově menší velikosti indexu.

Neklastrované indexy lze využít k několika cílům, jednak jako rychlé nasměrování k datům, jednak k rychlému zodpovězení dotazu. Pokud neklastrovaný index obsahuje všechna data, která jsou součástí dotazu, potom není nutné již dále hledat a dotaz může být zodpovězen pomocí samotného indexu. Pokud všechna data k dispozici nemá, musí je vydolovat pomocí odkazu, který má k dispozici. Odkaz může vést na klastrovaný index nebo haldu.

V rámci databáze jsou pro často přistupované údaje použity neklastrované indexy. Je to například u tabulky budov, kde je neklastrovaný index nad cizím klíčem město tak, aby se rychle vyhledávaly budovy v daném městě.

TRIGGERY

Trigger je spouštěč nějaké akce v databázi. Triggery mají spouštěcí akce, které aktivují akci definovanou triggerem. Databáze obsahuje trigger aktivovaný při vymazání hráče. Jakmile dojde k vymazání hráče, je v akci triggeru zkontrolována aliance, ve které hráč byl, a je-li aliance prázdná, tak je smazána.

11.4.3 Vytvoření skriptů

Dále byly v rámci tvorby databáze vytvořeny vytvářecí a datové skripty, tak aby bylo maximálně zautomatizováno počáteční vytvoření databáze do provozuschopného stavu. Vytvářecí skript obsahuje SQL kód vytvářející schéma databáze a všech databázových objektů. Pak jsou vytvořeny datové skripty, které obsahují vlastní inicializační data tabulek.

12 XNA komponenty

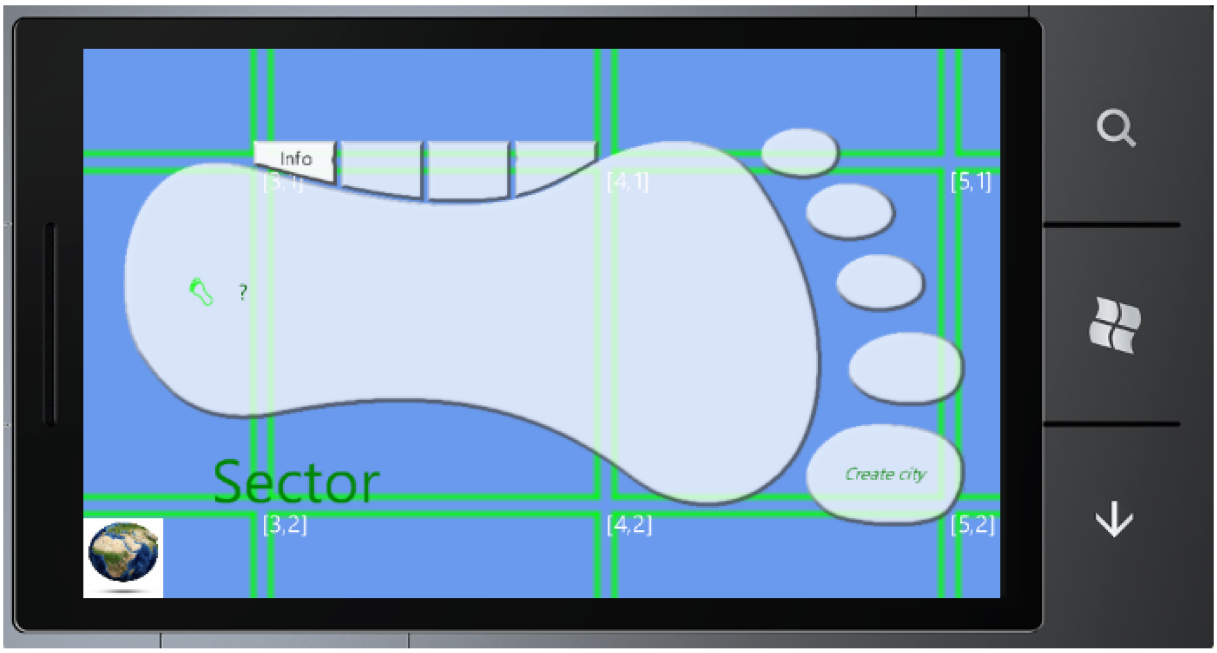
V této části je stručně popsána komponenta XNA, která realizuje krajinu světa, a dále to, jak hráč při hraní specifikuje textové údaje.

12.1 Krajina

V této části je popsána krajina světa. V rámci krajiny hráči objevují ostatní hráče a také zde zakládají svá města. Krajina je realizována jako komponenta XNA, která komunikuje prostřednictvím subsystému herních operací se službami k zajištění načítání daného obsahu. Obsah tvoří sektory světa a údaje o světě. Dále jsou zde také zakládána města.

Konkrétní ztvárnění mapy světa se děje pomocí obrázkové matice, v níž je obrazovka rozdělena na obdélníkové oblasti určité velikosti, v nichž je zobrazen obrázek. Obrázek odpovídá typu sektoru, který daná oblast reprezentuje. Po kliknutí do oblasti sektoru se hráči zobrazí dialogové okno, v němž se mu zobrazí informace o sektoru, a v němž může sektor obsadit novým městem, pokud již ale není sektor obsazen.

Součástí vytvořené komponenty je možnost cachování dat. Již načtené sektory se cachují do trvalé paměti telefonu. V případě žádosti o načtení sektorů, které jsou v této cache, není už nutné načítat je voláním služby. Jako cache je použit B+-strom [1], který tvoří dva soubory – index a data. Cache je vždy nově vytvořena při spuštění aplikace, takže jsou k dispozici nová data.



Obrázek 20: Ukázka dialogu Sector v krajině světa ve hře

12.2 Tvorba textových údajů

Obrázek 21: Ukázka vložení textových údajů

Hra potřebuje, aby hráč mohl zadávat určité údaje. Například při tvorbě hráče, vytváření aliance, pojmenovávání objektů apod.. K tomu je využita v XNA existující třída `Guide` nacházející se v sestavení `Microsoft.Xna.Framework.GamerServices.dll` a

jmenném prostoru `Microsoft.Xna.Framework.GamerServices`. Tato třída obsahuje mimo jiné metodu, pomocí které se hráči zobrazí klávesnice ke psaní.

13 Závěr a zhodnocení práce

V práci byly vytvořeny dvě části zajišťující online režim ve hře s tématem ekologické stopy. Byla vytvořena relační databáze zajišťující hlavní úložiště dat klientů – hráčů. Do databáze se ukládají veškeré herní informace potřebné k online hraní. Dále byly vytvořeny služby pro klienty, které zajišťují přenosy dat mezi mobilní hrou a úložištěm. Používá se přitom cloudová platforma Windows Azure. Databáze je poskytována v rámci této platformy jako služba.

Veškerou komunikaci se službami iniciuje mobilní aplikace. Vytvoření služeb bylo realizováno za pomoci technologie WCF. WCF technologie je použita jak na straně cloudu, tak na straně mobilní aplikace. Rozhraní služeb byla vytvořena pro podporu aliancí ve hře, hráčského profilu, možnosti používání herního světa a synchronizace herních dat. Byly zváženy ohledy na víceuživatelskou povahu aplikace. Hlavní úložiště je relační databáze podporující transakce a mající prostředky k zajištění integrity dat.

Byl uskutečněn návrh tříd na mobilní aplikaci, který podporuje komunikaci se službami. Komunikaci se službami realizují proxy objekty. Proxy byly vygenerovány speciálním nástrojem. Proxy komunikují se službami v asynchronním režimu, takže byla dále také vyřešena asynchronní komunikace v rámci aplikace. Aplikace tudíž odesílá a přijímá data asynchronně. Identifikace a autentizace klientů za účelem zajištění určité bezpečnosti, je realizována pouze na úrovni jména uživatele, které klient odesílá ve většině požadavků službě. V rámci použití Windows Azure byly také nastíněny možné peněžité náklady.

Dále byla vytvořena komponenta pro krajinu světa, v níž hráči mohou využívat takzvané sektory, zakládat město či zjišťovat informace o jiných hráčích a jejich městech.

POUŽITÉ ZDROJE

[1] B plus tree in C#, java and Python | Free Security & Utilities software downloads at SourceForge.net. *SourceForge - Download, Develop and Publish Free Open Source Software* [online]. Dice Holdings, Inc. [cit. 2013-05-24]. Dostupné z: http://sourceforge.net/projects/bplusdotnet/

[2] *EKOSTOPA SPOTŘEBY* [online]. Praha: Zelený kruh, 2009 [cit. 2013-05-24]. Dostupné z: http://www.zelenykruh.cz/dokumenty/letak-web.pdf

[3] Global hectare. *Wikipedia, the free encyclopedia* [online]. Wikipedia, The Free Encyclopedia., March 2013 [cit. 2013-05-24]. Dostupné z: http://en.wikipedia.org/w/index.php?title=Global_hectare&oldid=541477351

[4] Glossary. *Global Footprint Network :: HOME - Ecological Footprint - Ecological Sustainability* [online]. Global Footprint Network, 10/12/2012 [cit. 2013-05-24]. Dostupné z: http://www.footprintnetwork.org/en/index.php/GFN/page/glossary/

[5] History - Windows Phone Wiki Guide - IGN. *Video Games, Wikis, Cheats, Walkthroughs, Reviews, News & Videos - IGN* [online]. IGN Entertainment, Inc., September 2012 [cit. 2013-05-24]. Dostupné z: http://www.ign.com/wikis/windows-phone/History

[6] Kurzy devizového trhu – měsíční průměry - Česká národní banka. *Česká národní banka - Česká národní banka* [online]. Česká národní banka, 2013 [cit. 2013-05-24]. Dostupné z: http://www.cnb.cz/cs/financni_trhy/devizovy_trh/kurzy_devizoveho_trhu/prumerne_mena.jsp?mena=USD

[7] Peter Mell & Tim Gance, The NIST Definition of Cloud Computing (Final) (NIST Special Publication 800-145) (Sept. 2011)

[8] Pricing Details - SQL Database. *Windows Azure: Microsoft's Cloud Platform | Cloud Hosting | Cloud Services* [online]. Microsoft, 2013 [cit. 2013-05-24]. Dostupné z: https://www.windowsazure.com/en-us/pricing/details/sql-database/

[9] Pricing Details - Storage. *Windows Azure: Microsoft's Cloud Platform | Cloud Hosting | Cloud Services* [online]. Microsoft [cit. 2013-05-24]. Dostupné z: https://www.windowsazure.com/en-us/pricing/details/storage/

[10] Security Guidelines and Limitations (Windows Azure SQL Database). *Windows Azure: Microsoft's Cloud Platform | Cloud Hosting | Cloud Services* [online]. 2013 [cit. 2013-05-24]. Dostupné z: http://msdn.microsoft.com/en-us/library/windowsazure/ff394108.aspx

[11] Understand Your Bill for Windows Azure. *Windows Azure: Microsoft's Cloud Platform | Cloud Hosting | Cloud Services* [online]. Microsoft, 2013 [cit. 2013-05-24]. Dostupné z: http://www.windowsazure.com/en-us/support/understand-your-bill/

[12] Understanding Windows Azure Storage Billing – Bandwidth, Transactions, and Capacity - Windows Azure Storage Team Blog - Site Home - MSDN Blogs. CALDER, Brad. *Windows Azure Storage Team Blog - Site Home - MSDN Blogs* [online]. 8.7.2010 [cit. 2013-05-24]. Dostupné z: http://blogs.msdn.com/b/windowsazurestorage/archive/2010/07/09/understanding-windows-azure-storage-billing-bandwidth-transactions-and-capacity.aspx

[13] Web Services Description Language (WSDL) Version 2.0 Part 0: Primer. *World Wide Web Consortium (W3C)* [online]. W3C, 2007 [cit. 2013-05-24]. Dostupné z: http://www.w3.org/TR/wsdl20-primer/

[14] WIKIPEDIA CONTRIBUTORS. Global hectare. *Wikipedia, the free encyclopedia* [online]. Wikipedia, The Free Encyclopedia., 24 May 2013, 1 March 2013 [cit. 2013-05-24]. Dostupné z: http://en.wikipedia.org/wiki/Global_hectare

[15] WIKIPEDIA CONTRIBUTORS. SOAP. *Wikipedia, the free encyclopedia* [online]. Wikipedia, The Free Encyclopedia., May 2013 [cit. 2013-05-24]. Dostupné z: http://en.wikipedia.org/w/index.php?title=SOAP&oldid=555486267

[16] WIKIPEDIA CONTRIBUTORS. Visitor pattern. *Wikipedia, the free encyclopedia* [online]. Wikipedia, The Free Encyclopedia, May 2013 [cit. 2013-05-24]. Dostupné z: http://en.wikipedia.org/w/index.php?title=Visitor_pattern&oldid=553955458

[17] WIKIPEDIA CONTRIBUTORS. Web Services Description Language. *Wikipedia, the free encyclopedia* [online]. Wikipedia, The Free Encyclopedia., May 2013 [cit. 2013-05-24]. Dostupné z: http://en.wikipedia.org/w/index.php?title=Web_Services_Description_Language&oldid=556533830

[18] Windows Azure Pricing Calculator | Cloud Offers | Cloud Pricing. *Windows Azure: Microsoft's Cloud Platform | Cloud Hosting | Cloud Services* [online]. Microsoft, 2013 [cit.

2013-05-24]. Dostupné z: https://www.windowsazure.com/en-us/pricing/calculator/?scenario=full

[19] Windows Azure SQL Database Provisioning Model. *Windows Azure: Microsoft's Cloud Platform | Cloud Hosting | Cloud Services* [online]. Microsoft, 2013 [cit. 2013-05-24]. Dostupné z: http://msdn.microsoft.com/en-us/library/windowsazure/ee336227.aspx

[20] WWF - Human demand outstrips nature's supply. *WWF - WWF* [online]. WWF [cit. 2013-05-24]. Dostupné z: http://wwf.panda.org/about_our_earth/all_publications/living_planet_report/demands_on_our_planet/

[21] WWF. *Living Planet Report 2012* [online]. Gland, Switzerland: WWF International, May 2012 [cit. 2013-05-24]. ISBN 978-2-940443-37-6. Dostupné z: http://awsassets.panda.org/downloads/1_lpr_2012_online_full_size_single_pages_final_120516.pdf

[22] XML Information Set. *World Wide Web Consortium (W3C)* [online]. W3C, February 2004 [cit. 2013-05-24]. Dostupné z: http://www.w3.org/TR/xml-infoset/

[23] XML Schema Tutorial. *W3Schools Online Web Tutorials* [online]. [cit. 2013-05-24]. Dostupné z: http://www.w3schools.com/schema/

[24] NIELSEN, Paul W. *Microsoft sql server 2008 bible* [online]. 1st ed. Indianapolis, IN: Wiley Pub., Inc., 2009, p. cm. [cit. 2013-05-24]. ISBN 04-702-5704-0.

[25] BEN-GAN, Itzik. *Microsoft SQL Server 2012 T-SQL fundamentals* [online]. 1st ed. Farnham: O'Reilly [distributor], c2012, xxvi, 412 p. [cit. 2013-05-24]. ISBN 07-356-5814-5.

[26] DELANEY, Kalen. *Microsoft SQL server 2008 internals* [online]. 1st ed. Redmond, Wash.: Microsoft, c2009, xxvi, 754 p. [cit. 2013-05-24]. ISBN 07-356-2624-3.

[27] LÖWY, Juval. *Programming WCF services* [online]. 3rd ed. Sebastopol: O'Reilly, 2010, xxx, 875 s. [cit. 2013-05-24]. ISBN 978-0-596-80548-7.

[28] KRISHNAN, Sriram. *Programming Windows Azure* [online]. 1st ed. Sebastopol, CA: O'Reilly, c2010, xix, 345 p. [cit. 2013-05-24]. ISBN 978-059-6801-977.

[29] DATE, C. J. *An introduction to database systems* [online]. 8. ed., international ed. Boston, Mass. [u.a.]: Pearson, Addison-Wesley, 2004, xix, 345 p. [cit. 2013-05-24]. ISBN 03-211-8956-6.

[30] Windows Azure Storage Abstractions and their Scalability Targets - Windows Azure Storage Team Blog - Site Home - MSDN Blogs. CALDER, Brad. *Windows Azure Storage Team Blog - Site Home - MSDN Blogs* [online]. Microsoft Corporation, May 2010 [cit. 2013-05-24]. Dostupné z: http://blogs.msdn.com/b/windowsazurestorage/archive/2010/05/10/windows-azure-storage-abstractions-and-their-scalability-targets.aspx

[31] Pricing Details – Cloud Services. *Windows Azure: Microsoft's Cloud Platform | Cloud Hosting | Cloud Services* [online]. Microsoft [cit. 2013-05-24]. Dostupné z: https://www.windowsazure.com/en-us/pricing/details/cloud-services/

[32] Pricing Details – Data Transfers. *Windows Azure: Microsoft's Cloud Platform | Cloud Hosting | Cloud Services* [online]. Microsoft [cit. 2013-05-24]. Dostupné z: https://www.windowsazure.com/en-us/pricing/details/data-transfers/

[33] WIDER, Bill. *Cloud Architecture Patterns* [online]. First release. Sebastopol: O'Reilly Media, Inc., 2012 [cit. 2013-05-24]. ISBN 978-1-449-31977-9.

[34] KUROSE, James F a Keith W ROSS. *Computer networking: a top-down approach* [online]. 6th ed. Boston: Addison-Wesley, 2013, xxiv, 862 s. [cit. 2013-05-24]. ISBN 978-0-13-285620-1.

[35] DAIGNEAU, By Robert a Keith W ROSS. *Design patterns for domain services: solutions for the foundational elements of service oriented architectures* [online]. 6th ed. Boston, Mass: Addison-Wesley, 2010, xxiv, 862 s. [cit. 2013-05-24]. ISBN 978-032-1544-209.

[36] FOWLER, Martin. P of EAA: Unit of Work. *Martin Fowler* [online]. Martin Fowler [cit. 2013-05-24]. Dostupné z: http://www.martinfowler.com/eaaCatalog/unitOfWork.html

[37] Trends. *Global Footprint Network :: HOME - Ecological Footprint - Ecological Sustainability* [online]. Global Footprint Network, 5.8.2012 [cit. 2013-05-25]. Dostupné z: http://www.footprintnetwork.org/en/index.php/GFN/page/trends/czechrepublic/

[38] Ekologická stopa. *Hra o Zemi* [online]. Zelený kruh o.s., 2007 [cit. 2013-05-25]. Dostupné z: http://www.hraozemi.cz/ekostopa.html

[39] *Windows Azure Programming Patterns for Start-ups: A step-by-step guide to create easy solutions to build your business using Windows Azure services* [online]. 1. vyd. Birmingham: Packt Publishing, 2012 [cit. 2013-05-25]. ISBN 978-1-84968-560-3.

[40] Windows Azure Purchase Options: How to buy Windows Azure | Windows Azure Offers. *Windows Azure: Microsoft's Cloud Platform | Cloud Hosting | Cloud Services* [online]. Microsoft [cit. 2013-05-24]. Dostupné z: http://www.windowsazure.com/en-us/pricing/purchase-options/

[41] Using SLsvcUtil.exe to Access a Service. *MSDN – the Microsoft Developer Network* [online]. Microsoft, 2013 [cit. 2013-05-25]. Dostupné z: http://msdn.microsoft.com/en-us/library/cc197958%28v=vs.95%29.aspx

[42] WIKIPEDIA CONTRIBUTORS. Double dispatch. *Wikipedia, The Free Encyclopedia* [online]. Wikipedia, The Free Encyclopedia., April 2013 [cit. 2013-05-25]. Dostupné z: http://en.wikipedia.org/w/index.php?title=Double_dispatch&oldid=552340086

[43] Clustered Index Design Guidelines. *MSDN – the Microsoft Developer Network* [online]. Microsoft, 2013 [cit. 2013-05-25]. Dostupné z: http://msdn.microsoft.com/en-us/library/ms190639%28v=sql.105%29.aspx

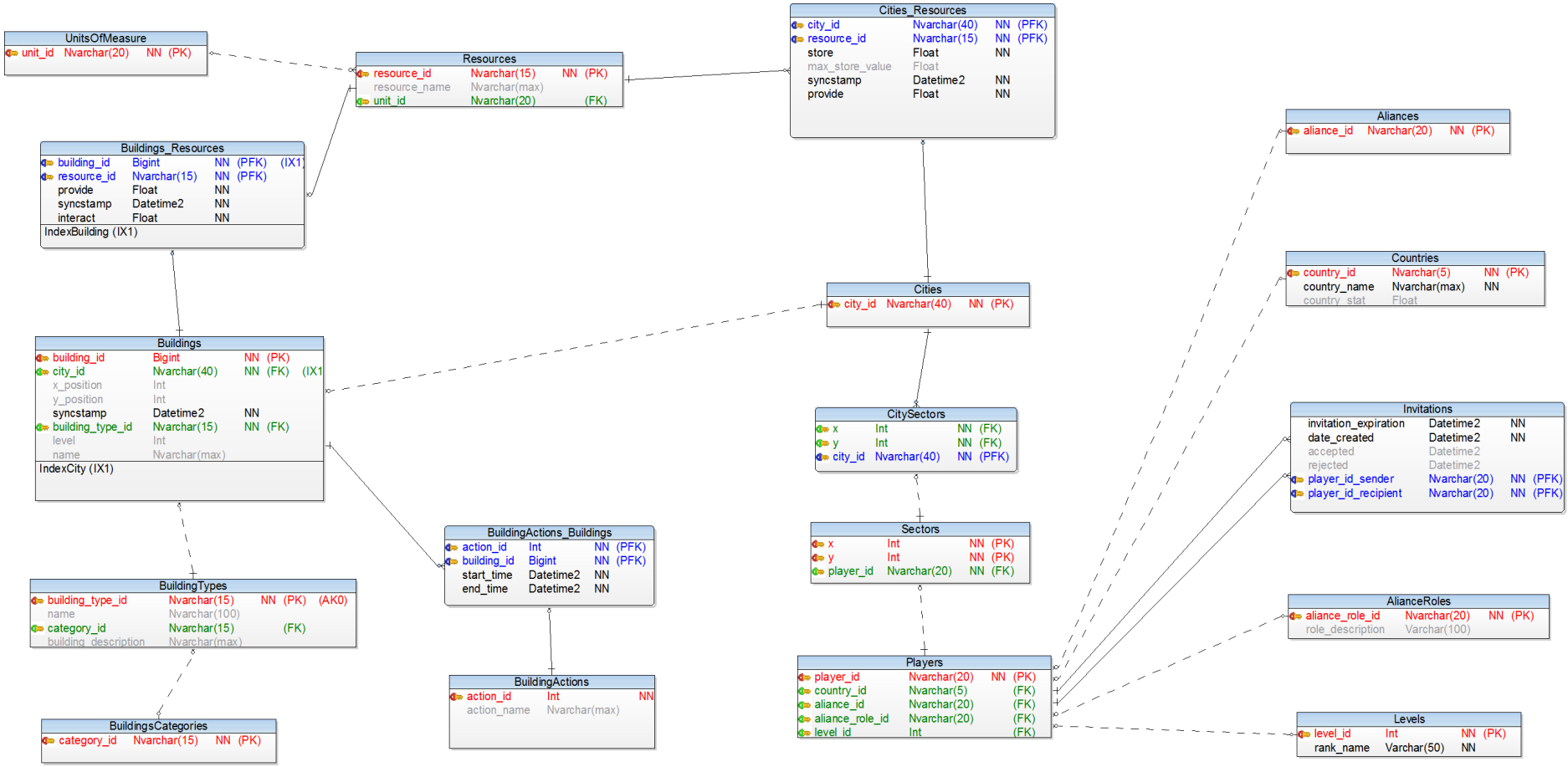
[44] RICHTER, Jeffrey. *CLR via c#* [online]. 4th ed. Redmond, WA: Microsoft Press, 2012, p. cm. [cit. 2013-05-26]. ISBN 978-073-5668-751.

[45] Gesta: přechod, posunutí a roztažení | Návody pro Windows Phone (Česká republika). *Smartphone stvořený přímo pro vás | Windows Phone (Česká republika)* [online]. Microsoft, 2013 [cit. 2013-05-29]. Dostupné z: http://www.windowsphone.com/cs-CZ/how-to/wp8/start/gestures-flick-pan-and-stretch

[46] I Love Shaders: Anatomy of a XNA project. ADRIAN-FLORIN VIȘAN. *I Love Shaders* [online]. April 2011 [cit. 2013-05-29]. Dostupné z: http://iloveshaders.blogspot.cz/2011/04/anatomy-of-xna-project.html

PŘÍLOHY

Příloha 1 - Databázové schéma



Příloha 2 – CD ROM

Obsah přiloženého CD ROM tvoří:

- Tato práce ve formátu PDF.
- Zdrojové kódy programu.