Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Katedra informačních technologií

Studijní program: Aplikovaná informatika

Obor: Informační systémy a technologie

Tvorba mapové aplikace pro sledování polohy v Cloud – serverová část Windows Azure

DIPLOMOVÁ PRÁCE

Student : Bc. Vojtěch Růžička

Vedoucí : Ing. Tomáš Bruckner, Ph.D.

Oponent : Ing. Radek Skoupý

2012

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a že jsem uvedl všechny použité prameny a literaturu, ze kterých jsem čerpal.

V Praze dne 20. 4. 2012 .....

podpis

Poděkování

Rád bych zde poděkoval vedoucímu práce Ing. Tomáši Brucknerovi, Ph.D. za odborné vedení, rady a pomoc při tvorbě této práce.

Dále bych chtěl poděkovat Ing. Filipu Řehoříkovi ze společnosti Microsoft za pomoc v raných fázích práce a umožnění bezplatného přístupu k Microsoft Windows Azure.

V neposlední řadě bych rád poděkoval Ing. Zbyňku Poulíčkovi ze společnosti GINA Software za konzultace, poskytnutí GPS dat a především vizi, ze které vzešla aplikace ve své finální podobě.

Abstrakt

Hlavním cílem práce je představit platformu Windows Azure a její možnosti a limity, a to především co se týče spolupráce s mobilními klienty na platformě Windows Phone 7. To je demonstrováno na vývoji aplikace pro trasování a navigaci obtížným terénem určené pro vozíčkáře a cyklisty. Poznatky nabyté návrhem a implementací této aplikace jsou zobecněny, aby mohly posloužit dalším vývojářům na platformě Windows Azure. Samotná aplikace je vyvíjena s ohledem na užitečnost pro definované cílové skupiny. Tato diplomová práce je zaměřena na serverovou část na platformě Windows Azure. Klientská část aplikace na platformě Windows Phone 7 není její součástí a je zpracována Bc. Vítem Čurdou v jeho diplomové práci s názvem *Tvorba mapové aplikace pro sledování polohy v Cloudu - klientská část Windows Phone 7*.

Klíčová slova

Cloud Computing, Windows Azure, SQL Azure, Windows Phone 7, mapa, trasování, plánování trasy

Abstract

The main objective of this thesis is to introduce the Windows Azure platform, its possibilities and limits, particularly with regard to interaction with mobile clients running on the Windows Phone 7 platform. This is demonstrated by the development of an application designed to ease navigation through difficult terrain. This application is primarily designed to help cyclists and people with handicap restricting their movement. The knowledge acquired through design and implementation of this application is generalized to be useful for other developers using Windows Azure. The application is developed with focus on the usefulness for the defined target groups. This thesis is focused on the server side of the application running on the Windows Azure platform. The client side of the application running on the Windows Phone 7 platform is not covered in this thesis and is described in detail by Bc. Vít Čurda in his thesis *Development of map application for tracking in cloud - Client side on Windows Phone 7*.

Keywords

Cloud Computing, Windows Azure, SQL Azure, Windows Phone 7, tracking, route planning

Obsah

1 Úvod

V poslední době zaznamenala mobilní zařízení obrovský vzestup. Chytré mobilní telefony jsou již samozřejmostí a nové tablety se velmi rozšířily a téměř vytlačily ještě nedávno tak populární netbooky. Neustále roste výkon mobilních zařízení, množství aplikací pro ně určených a jejich komplexita. Přesto je výhodnější, aby tyto aplikace spolupracovaly se službami na straně serveru a určitou část zodpovědnosti na něj delegovaly. Server se pak často stará o složitější výpočty, samotné ukládání dat, jejich zpracování, distribuci ostatním klientům atd. Na straně serveru je v poslední době stále populárnější cloudové řešení. To má oproti standardnímu on-premise řešení řadu výhod jako je snadná škálovatelnost, odolnost vůči selhání a zátěži a mnoho dalších. Spolupráce mobilního klienta a serverové části na cloudové platformě je velmi výhodná a vhodně kombinuje přednosti mobilních zařízení a silné cloudové infrastruktury. Navíc v nedávné době společnost Microsoft představila nové technologie pro cloudovou i mobilní část. Jedná se o Windows Phone 7 a Windows Azure. Vzhledem k tomu, že se jedná o technologie jedné společnosti, jejich kombinace má tu výhodu, že výsledné řešení bude používat prostředky jedné společné platformy a bude lépe integrované a homogenní. Uvedení těchto nových technologií je tedy ideální příležitostí k vypracování aplikace, která bude na straně serveru implementována na platformě Windows Azure a na straně klienta na platformě Windows Phone 7. Takováto aplikace poslouží nejen sama o sobě jako nástroj, ale především poslouží jako ilustrace obou nových technologií, možností jejich provázání a spolupráce, jejich limitů a podobně. Vzhledem k tomu, že se jedná o technologie nové, není zatím k dispozici dostatečné množství zdrojů o nich pojednávajících, natož pak o takovéto formě spolupráce a integraci obou částí. Práce tedy poslouží jako vhodný zdroj informací. Výhodou také je, že na příkladu konkrétní aplikace a jejího vývoje jsou získané znalosti a ponaučení mnohem zřetelnější a konkrétnější, než kdyby se jednalo o pouhé teoretické zhodnocení spolupráce obou technologií. Získané poznatky budou zobecněny, aby je bylo možno aplikovat při vývoji ostatních aplikací.

Vyvíjená aplikace by měla být také prospěšná sama o sobě, tedy nejen jako prostředek k demonstraci určitých principů. Měla by efektivně využít přednosti mobilního zařízení a stejně tak i serverové části v cloudu. Jako téma aplikace nakonec bylo zvoleno sledování polohy a zaznamenávání trasy klientem s tím, že tyto údaje získané jednotlivými klienty pak pomohou k navigaci obtížným terénem dalším uživatelům. Aplikace je zaměřena na cílové skupiny, kterým nejvíce pomůže možnost naplánovat trasu obtížným a nepřehledným terénem, tedy osobám se sníženou schopností pohybu (především vozíčkářům) a cyklistům. Toto zaměření aplikace bylo vybráno s ohledem na skutečnou užitečnost aplikace s přihlédnutím k tomu, že žádné uspokojivé řešení zabývající se danou problematikou zatím neexistuje.

Serverová a klientská část jsou založeny na rozdílných technologiích a z hlediska vývoje se jedná o natolik rozdílné oblasti, že je vhodné, aby byly zpracovány zvlášť. Na každou z těchto částí je tedy zaměřena jedna diplomová práce. Moje práce se zabývá částí serverovou, diplomová práce Bc. Víta Čurdy (*Tvorba mapové aplikace pro sledování polohy v Cloudu - klientská část Windows Phone 7*) se zabývá částí klientskou. Vzhledem k úzkému propojení klientské a serverové části bylo nutné navzájem spolupracovat a koordinovat vývoj. Kromě toho probíhala spolupráce s dalšími dvěma subjekty, a to společnostmi Microsoft a GINA Software.

1.1 Cíle

Prvním cílem práce je prozkoumat možnosti a potenciální problémy platformy Windows Azure a její spojení s platformou Windows Phone 7 a poskytnout tak podklady, doporučení a varování vývojářům zabývající se touto problematikou. Práce je zaměřena konkrétně na model - Windows Azure služby na straně serveru a tenkého klienta se systémem WP7 na straně klientské.

Druhým cílem je vytvořit serverovou část aplikace, která bude sloužit primárně cyklistům a osobám se sníženou schopností pohybu k lepšímu plánování trasy nepřehledným terénem (především městem). Tato aplikace bude spuštěna na platformě Windows Azure a bude schopna komunikace s různými typy klientů, především pak s mobilními zařízeními se systémem WP7 a s webovými klienty na technologii Silverlight. Cílem je vyvinout takovou aplikaci, která bude vyhovovat požadavkům blíže definovaným v kapitole *4.1 Požadavky na aplikaci*.

1.2 Struktura

Na začátku práce je popsán výchozí stav – motivace vedoucí k vytvoření práce v její současné formě, popis ostatních prací na dané téma a partnerů, se kterými probíhala spolupráce na projektu. Další kapitolu tvoří teoretický základ ke Cloud Computingu a především k platformě Windows Azure. Následující kapitola se zabývá analýzou návrhu aplikace. Zde jsou rozebrány požadavky na aplikaci, dále pak případy užití, návrh architektury a další části související se samotným návrhem aplikace. Po analýze následuje kapitola *Implementace*. Ta se zabývá formami spolupráce při tvorbě aplikace a především zajímavostmi a problémy, které bylo nutno při implementaci řešit. Kapitola *Testování a provoz* popisuje post-implementační fáze životního cyklu aplikace, způsoby testování a nasazení. Následně jsou předloženy náměty k rozšíření celého projektu a informace pro potenciální pokračovatele. Na samém konci práce jsou vyhodnoceny cíle dle předem definovaných metrik.

1.3 Přínosy

Prvním přínosem je průzkum platformy Windows Azure a její interakce s klientem na WP7. Jedná se o nové technologie, které se velmi rychle vyvíjejí a u kterých není stále ještě k dispozici dostatek pramenů na dané téma. Přínosem je tedy zdroj informací, doporučení a varování pro vývojáře, kteří se rozhodnou tvořit nebo migrovat aplikace na platformu Azure. Práce přináší větší přínos pro vývojáře zabývající se konkrétně spojením Windows Phone 7 a Windows Azure.

Další přínosy plynou z veřejné dostupnosti aplikace, která usnadňuje plánování a výběr vhodné trasy cyklistům a vozíčkářům. Celkově se jedná o poměrně velkou cílovou skupinu a v současné době není volně a zdarma k dispozici vhodný nástroj, který by tento problém řešil.

2 Výchozí stav

Cílem této kapitoly je nastínit situaci v době zahájení práce, tedy vysvětlit, co mě vedlo k výběru daného tématu a formy, dále zmapovat ostatní práce na dané téma a představit jednotlivé účastníky projektu.

2.1 Motivace

Cílem této kapitoly je objasnit motivaci a pohnutky vedoucí k tvorbě této práce v dané podobě.

Primárním cílem bylo vybrat si takové téma, které bude řešit aktuální a reálný problém. Řešení tohoto problému by mělo být společensky přínosné a mělo by přinášet konkrétní praktický užitek. Dále pro mě bylo důležité spolupracovat na projektu se skutečnou firmou z oboru. Projekt by měl mít technologické zaměření zahrnující také vývoj aplikace. Použité technologie by měly být pokud možno nové, o kterých je zatím k dispozici malé množství pramenů, takže práce bude mít podstatný informační přínos.

V této práci se mi podařilo splnit všechny tyto požadavky.

2.2 Výchozí stav v dané oblasti

Žádná práce na určité téma není izolovaná a samostatná, ale je třeba ji chápat v kontextu ostatních prací z oboru a celkové úrovně poznání a zpracování daného tématu. Znalost těchto skutečností je nutná při rozhodování se o výběru tématu práce. Ta by neměla být nadbytečná, ale měla by se zabývat oblastmi, které v současné době nejsou uspokojivě zpracovány a kde tvorba takové práce bude přínosem pro danou oblast. Cílem této kapitoly je zmapovat současné práce na téma Windows Azure, Windows Azure + WP7 a navigace obtížným terénem. Na základě nabytých informací pak může být učiněno rozhodnutí, jak koncipovat práci, aby byla co nejprospěšnější v kontextu prací existujících, a zda má vůbec smysl se danou problematikou zabývat. Hodnocení jsem rozdělil do dvou oblastí. První je technologické hledisko. Druhá oblast se zabývá obecně podobným typem aplikací.

Informace byly čerpány především prostřednictvím elektronických zdrojů CIKS VŠE[1], především meta vyhledavače Serials Solutions[2], který v současné době zahrnuje 53 různých zdrojů. Ne všechny jsou však z oblasti ICT. Jako hlavní zdroj informací z oboru ICT sloužil ACM Digital Library[3], také zahrnutý v Serials Solutions. Jako sekundární zdroj informací, především pro neakademické práce (jako různé návody, dokumentace apod.), bylo použito standardní Google vyhledávání.

[1] http://www.vse.cz/zdroje/
[2] http://www.vse.cz/zdroje/o/133
[3] http://www.vse.cz/zdroje/o/85

2.2.1 Technologie

Windows Azure je poměrně nová technologie. Přesto lze nalézt nezanedbatelné množství informací. Při bližším průzkumu však situace rozhodně není tak příznivá. Velká část informací jsou případové studie a rozbory zabývající se vhodností Windows Azure pro business, jeho návratnost nebo problematiku migrace existujících aplikací. Další velkou skupinu tvoří prohlášení o vydání nových funkcí nebo plány do budoucnosti. Co se týče samotného vývoje a popisu platformy, zdrojů je výrazně méně. Velkým problémem je, že od doby svého uvedení do současnosti prošla platforma Windows Azure řadou změn. A to nejen co se týče přidání nových funkcí, ale často i radikálních změn funkcí existujících. Přitom tyto změny probíhají s vysokou frekvencí. Pokud je tedy zkoumán určitý informační pramen, neexistuje jistota, že obsažené informace jsou stále aktuální, případně mohou být zcela irelevantní. Je tedy potřeba provést další velmi důkladný průzkum a prostudovat ostatní prameny na dané téma a snažit se najít ten co možná nejnovější. Přitom u mnohých pramenů (často včetně i těch oficiálních od společnosti Microsoft) není možné zjistit přesně (často ani vůbec) datum jejich vzniku. Zastaralost informací je hlavním problémem knih o Windows Azure. Jejich tvorba zabere určitý čas a v okamžiku jejich vydání už jsou leckteré informace neaktuální nebo neúplné. Samotný čtenář se ke knize dostane ještě později. Přesto jsou tyto knihy dobrým úvodem do problematiky (a lze je doporučit jako primární zdroj pro úplné nováčky v oboru Windows Azure), neboť jsou v nich nastíněny jednotlivé komponenty a možnosti Windows Azure. Z knih lze doporučit např. Azure in Action [1] nebo Windows Azure Step By Step [2]. Detailnější a aktuálnější informace je pak ale třeba získat z jiných zdrojů nebo alespoň ověřit jejich aktuálnost. Jako další zdroje lze doporučit Microsoft Developer Network pro Windows Azure[4] a také Windows Azure Training Kit[5] poskytující množství příkladů a tutoriálů. Tato práce bude přínosná z hlediska toho, že ve své teoretické části zachytí „snímek“, v jakém stavu se celkově nachází platforma Windows Azure v jeden určitý moment. Poskytne tak spolehlivý zdroj informací k určitému datu bez nutnosti slučovat informace ze zdrojů s různým datem původu, u kterých nelze s jistotou určit aktuálnost informací.

K dispozici jsou sice zdroje, které nastíní možnosti jednotlivých komponent Windows Azure a základní manipulaci s nimi, ale postrádal jsem informace o samotném návrhu aplikace, možnosti různých přístupů k architektuře, výhody a nevýhody použití jednotlivých mechanismů a jejich propojení. Dále pak porovnání alternativ a jejich slabých a silných stránek, rovněž tak hodnocení specifik vývoje pro cloudovou platformu. Bude přínosem, pokud práce bude obsahovat také vysvětlení návrhu architektury aplikace a důvody volby takového řešení, dále vysvětlení výběru určitých mechanismů, porovnání s jejich alternativami a celkové zobecnění pro návrh takovýchto aplikací.

[4] http://msdn.microsoft.com/en-us/library/dd163896.aspx
[5] http://www.microsoft.com/download/en/details.aspx?id=8396

Co se týče spojení platformy Windows Azure a Windows Phone 7, počet zdrojů je ještě menší. Pravděpodobně zde hraje roli to, že WP7 je nová platforma, a také to, že mezi ostatními mobilními platformami je zatím velmi málo rozšířená[6]. V této oblasti poskytne práce největší informační přínos, protože se zaměřuje právě na interakci obou platforem, přičemž současné zdroje zatím nejsou vyhovující.

2.2.2 Aplikace pro navigaci obtížným terénem

Snaha poskytnout vozíčkářům (a nejen jim) nástroje k efektivnějšímu rozhodování při navigaci obtížným terénem (především městem) není rozhodně nová ani ojedinělá. Často se však jedná „pouze“ o standardní mapy v tištěné podobě, které jsou doplněny informacemi o bariérách přístupu [3]. Takové řešení není příliš praktické. Informace nelze aktualizovat, a proto rychle zastarávají. Oproti elektronickým verzím, kde lze (při drobném zjednodušení) počítat pouze s náklady fixními, se musí počítat s variabilními náklady na tvorbu mapy pro každého uživatele. Celkově se jedná o neflexibilní řešení. Elektronická verze má výhodu v jednoduché rozšiřitelnosti, snadnější distribuci, možnosti aktualizovat snadno informace samotnými členy komunity a mnohé další. Přesto elektronických nástrojů k dispozici příliš mnoho není. Někdy se jedná pouze o statickou digitalizaci standardních map bez možnosti editace. Pokud existují interaktivní aplikace, lze hodnotit přístupnost pouze pomocí bodů, nikoliv tras. Z tuzemských nástrojů můžeme jmenovat např. aplikaci vyvíjenou v rámci bakalářské práce Martina Biskupiče – *Interaktívna mapa prístupnosti pre hendikepovaných* [4]. Ze zahraničních aplikací lze jmenovat např. Planat[7]. Velkou nevýhodou existujících nástrojů je, že jsou určeny pro standardní webové prohlížeče provozované na desktopech a noteboocích. Uživatel může plánovat trasu z domova, ale ne už operativně přímo v terénu. Některé služby (např. již zmiňovaný Planat), mají verzi svého webu pro mobilní zařízení, ale stále se nejedná o příliš použitelné řešení. Nepodařilo se mi nalézt nástroj, který by poskytoval vlastního klienta pro komplexnější a komfortnější plánování přímo v terénu. V této oblasti je tedy velký prostor pro přínos, a to především poskytnutím samostatného klienta, který dokáže zajistit řadu operací sám bez nutnosti konstantního přístupu k serveru a poskytne kromě jednotlivých bodů možnost ukládat i celé trasy.

[6] Dle společnosti Gartner byl prodej mobilních zařízení s OS společnosti Microsoft v třetím čtvrtletí roku 2011 pouhých 1,5% celkového objemu. [34]
[7] www.planat.com

2.3 Partneři

Cílem této kapitoly je představit jednotlivé subjekty, které participovaly na projektu, a nastínit rozsah a průběh spolupráce s nimi.

2.3.1 Microsoft s.r.o.

http://www.microsoft.cz

IČO: 47123737

Sídlo: BB Centrum, budova Alfa, Vyskočilova 1461/2a, Praha 4, PSČ 140 00

Microsoft s.r.o. (dále jen Microsoft) je českou pobočkou americké společnosti Microsoft. Protože dlouhodobě pracuji na platformě .NET a hodlám se této oblasti věnovat i nadále, mou první volbou při hledání partnerské firmy pro diplomovou práci byla právě společnost Microsoft, která spolupráci se studenty podporuje různými formami. Příkladem může být projekt DreamSpark[8]. Díky tomuto programu mají studenti Fakulty Informatiky a Statistiky přístup zdarma k mnoha softwarovým produktům společnosti Microsoft pro nekomerční použití. Bez některých z nich by tvorba této práce nebyla možná (Microsoft Visual Studio 2010 Ultimate Edition – Integrated Development Environment pro .NET), další poskytly kvalitní nástroje pro tvorbu (Microsoft Office Visio 2010, Microsoft Office Onenote 2010, Microsoft Windows 7). Jedním z podobných iniciativ Microsoftu je i projekt Microsoft Students To Business, zprostředkovávající propojení studentů VŠ a společností z oboru. Součástí této iniciativy je i nabídka témat závěrečných prací[9]. Téma, které mě zaujalo a které jsem si vybral, bylo „Windows Azure“.

Jako kontaktní osoba programu Students To Business byl uveden Ing. Filip Řehořík. V rámci konzultace se ukázalo, že téma diplomové práce není spojeno s konkrétním projektem, ale záleží přímo na studentovi, jak téma pojme. Microsoft pak v celé věci vystupuje jako jakýsi mentor a prostředník, tedy poskytne potřebné konzultace, pomůže s přesným definováním projektu a zprostředkuje kontakt s vybranou firmou z okruhu svých partnerů, která se zabývá danou problematikou nebo která poptává projekt na dané téma. Microsoft dále může poskytnout přístup k hardware a software, pokud je tento potřebný k práci na projektu.

Při konzultacích s Ing. Řehoříkem byl také přítomen další zájemce o spolupráci s Microsoftem na diplomové práci Bc. Vít Čurda. Ten měl zájem o téma „Aplikace pro Windows Phone 7“. Při podrobnějším průzkumu a snaze o definici jednotlivých prací se ukázalo, že tato témata se velmi vhodně doplňují, aniž by se přitom překrývala. Bylo tedy vhodné pojmout oba projekty jakožto

[8] Donedávna známý pod názvem Microsoft Developer Network Academic Alliance (MSDNAA)
[9] Aktuální seznam dostupný na http://www.s2bp.cz/Thesis.aspx

navzájem propojené, tedy jako tvorbu aplikace se serverovou částí na Windows Azure a klientskou částí na Windows Phone 7.

Ing. Řehořík také zprostředkoval kontakt se společností GINA Software (viz. 2.3.2). Další spolupráce pak probíhala především s touto firmou.

V době raných fází příprav této práce byl Windows Azure natolik novou technologií, že nebyl možný veřejný přístup pro účely tvorby aplikace pro mou diplomovou práci. Ing. Řehořík mi tedy tento přístup zprostředkoval, a to bezplatně. Bez tohoto přístupu by nemohla být tato práce vůbec realizována[10] a bezplatným přístupem bylo ušetřeno značné množství finančních prostředků. Umožnění bezplatného přístupu k Windows Azure bylo vůbec největším přínosem plynoucím ze spolupráce s Microsoftem.

2.3.2 GINA Software s.r.o.

http://www.ginasoftware.cz

IČO: 29254191

Sídlo: Brno, U Vodárny 3032/2a, PSČ 616 00

GINA Software je společnost zabývající se vývojem mapového a trasovacího software. Jejím hlavním produktem je GINA System. Ten představuje interaktivní mapový software pro mobilní zařízení umožňující navigaci v náročném terénu, koordinaci týmů a efektivní výměnu geografických informací [5]. Tento systém byl úspěšně použit například ke koordinaci záchranných prací po zemětřeseních na Haiti v lednu roku 2010. Společnost GINA Software sama začínala jako studentský projekt v programu Microsoft Students To Business.

Po získání kontaktu na GINA Software následovaly osobní schůzky, které měly za cíl definoval další směr spolupráce a výslednou podobu projektu. Hlavní kontaktní osobou pro účely naší spolupráce byl Ing. Zdeněk Poulíček, spoluzakladatel a současný CEO společnosti GINA Software.

Podoba GINA System v březnu 2011 byla následující. Serverová část komunikující přes vlastní protokol založený na SOAP, zajišťující zpracování dat, tvorbu mapových podkladů aj., Klientská část nasazená na mobilních zařízeních se systémem Windows Mobile 6.5. Tato podoba však už nebyla pro současné potřeby dostačující. Zastaralá platforma Windows Mobile měla řadu omezení a serverová část často nedokázala poskytnout požadovanou spolehlivost a škálovatelnost. Vzhledem k tomu, že se jedná o aplikaci sloužící záchranářům v případě živelních pohrom, byly rozdíly mezi klidovým nasazením a krizovým provozem propastné. Přitom bylo nutné mít schopnost mezi oběma režimy

[10] V době dokončení této práce (4/2012) je již platforma Windows Azure volně k dispozici. Výhoda bezplatného přístupu však stále přetrvává.

plynule přecházet ve velmi krátkém čase a zachovat původní spolehlivost, stabilitu a dostupnost. Pro tyto potřeby výborně vyhovuje právě cloudové řešení typu Windows Azure. GINA Software tedy měla v plánu přechod na tyto vhodnější technologie – klienty na Windows Phone 7 a server na Windows Azure. Původní představa firmy byla taková, že bychom provedli konverzi na WP7 a migraci na Windows Azure. Toto řešení však nebylo z hlediska potřeb diplomové práce přijatelné. Jednalo by se pouze o portaci hotového projektu na jinou platformu. Přidaná hodnota práce, její přínos pro společnost a výzkumná část by tedy nebyly dostatečné.

Nakonec byla tato možnost zavržena. Po dalším jednání přišly ze strany GINA Softwarepodnty, které ve svém důsledku určikysoučasnou podobu aplikace. Tedy Software pro zaznamenávání tras cyklistů a vozíčkářů a plánování jejich trasy. Oproti předchozí variantě se jedná o tvorbu vlastní aplikace od počátku, s větším přínosem a přidanou hodnotou.

Přínos ze spolupráce se společností GINA Software měl být především v jejích zkušenostech s aplikacemi podobného typu postavených na stejném modelu. Tedy implementace mapové části a zajištění komunikace mezi klientem a serverem. S použitím nových technologií se ukázalo, že odlišnosti od jejich softwaru jsou už příliš velké, než abychom mohli využít nějaké výraznější zkušenosti z jejich implementace. Komunikaci na vlastním protokolu na bázi SOAP jsme zavrhli a vydali se cestou .NET Windows Communication Foundation. Mobilní klient na WP7 na bázi Silverlightu je taktéž technologicky zcela odlišný oproti Windows Mobile 6.5. S Windows Azure neměla GINA žádné zkušenosti.

Hlavní konzultace pak tedy spočívaly v používání geometrických a geografických typů v databázové vrstvě aplikace. Dále nám bylo poskytnuto obrovské množství GPS Dat, konkrétně trasy záchranářů na Haiti. Oproti původním plánům byla spolupráce s GINA Software méně intenzivní, přesto by bez ní aplikace neexistovala v takové formě, v jaké se nachází dnes.

2.3.3 Bc. Vít Čurda

Aplikace jako celek je rozdělena do dvou samostatných částí, z nichž každá je postavena na zcela odlišné technologii. Serverová část je postavena na platformě Windows Azure a zpracovával jsem ji já. Klientská část je postavena na platformě Windows Phone 7 a zpracovával ji Vít Čurda ve své samostatné diplomové práci.

Obě části jsou zcela samostatné a byly zpracovány zvlášť. Přesto bylo třeba často spolupracovat. Jednalo se především o globální definici aplikace, požadavků na ni a její funkčnosti. Dále pak definici rozhraní a komunikace mezi oběma částmi. Rozdělení zodpovědností a rozsah spolupráce je blíže

definován v kapitole 5.1. Průběh spolupráce, co se týče metodiky vývoje a koordinace spolupráce, je blíže popsán v kapitole 5.2.

Samotná spolupráce probíhala zcela bez problémů. Důležitou roli zde hrálo striktní oddělení obou oblastí a sfér působnosti. Každý měl pevně definované povinnosti a zodpovědnost a ty dodržoval. V oblastech, které bylo třeba definovat nebo zpracovat společně (jako například komunikace a definice rozhraní mezi klientskou a serverovou částí), jsme vždy dokázali bezproblémově dojít ke konečnému výsledku.

3 Teoretický základ

3.1 Cloud Computing

Cloud Computing je v poslední době moderním pojmem. Existuje celá řada definic, z nichž mnohé jsou poměrně neurčité. Mezi často citované patří definice National Institute of Standards and Technology ministerstva obchodu USA.

„Cloud Computing je model, který umožňuje všudypřítomný, pohodlný síťový přístup na požádání k sdílenému fondu konfigurovatelných ICT zdrojů (např. sítě, servery, úložiště, aplikace a služby), které mohou být rychle poskytnuty a opět uvolněny s minimálním úsilím nebo interakcí ze strany poskytovatele služby.“ [6]

Dle NIST má cloudový model pět základních vlastností [6]:

- **Samoobslužný systém na požádání** – Zákazník může samostatně měnit parametry poskytovaných služeb, jako je výpočetní výkon a úložná kapacita, a to i automaticky bez nutnosti lidské interakce mezi poskytovatelem a zákazníkem.
- **Síťový přístup** - Služby jsou dostupné přes síť prostřednictvím standardních mechanismů umožňujících použití heterogenních tlustých či tenkých klientů (např. mobilní telefony, tablety, notebooky, desktopy).
- **Sdílení zdrojů** – ICT zdroje poskytovatele jsou sdíleny tak, aby mohly sloužit většímu počtu zákazníků současně. Různé fyzické a virtuální zdroje jsou na základě poptávky dynamicky přiřazovány a odebírány. Zákazník přitom obecně nemá představu ani kontrolu nad přesným fyzickým umístěním těchto zdrojů. Může mít ale možnost určit umístění těchto zdrojů na vyšší úrovni abstrakce (např. geografické umístění – stát nebo datové centrum). Příkladem takto sdílených zdrojů může být úložiště, výpočetní výkon nebo operační paměť.
- **Rychlá škálovatelnost** – Zdroje mohou být konkrétnímu zákazníkovi dynamicky přidělovány a odebírány tak, aby bylo vyhověno okamžité poptávce. V některých případech je to možné i automaticky. Z pohledu zákazníka se zdroje zdají být neomezené (i když zákazník může nastavit určitý strop) a míra jejich poskytování může být kdykoliv upravena.
- **Měření poskytovaných služeb** – Cloudové systémy automaticky řídí a monitorují úroveň poskytovaných zdrojů. Zákazník pak má detailní přehled o rozsahu poskytovaných služeb a hradí také pouze zdroje skutečně spotřebované.

3.1.1 Přístupy k poskytování služeb

Na základě toho, jak velkou část poskytované služby zajišťuje a řídí zákazník a jak velkou poskytovatel, můžeme dle NIST rozlišit tři modely poskytování cloudových služeb, které jsou blíže popsány níže. Jedná se o modely Infrastructure as a Service (IaaS), Platform as a Service (PaaS) a Software as a Service. Rozdělení zodpovědnosti mezi poskytovatelem a zákazníkem v jednotlivých modelech ukazuje *Obrázek 1 - Porovnání IaaS, PaaS, SaaS a modelu on-premise*. Modře jsou označeny oblasti, za které nese zodpovědnost zákazník, šedivě pak ty, za které odpovídá poskytovatel služby.

3.1.2 Infrastructure as a Service

V tomto modelu jsou zákazníkovi poskytovány základní výpočetní zdroje (Virtualization, Servers, Storage, Networking). Zákazník je pak schopen nasadit operační systém a jakýkoliv software. Zákazník neřídí základní cloudovou infrastrukturu, ale má pod kontrolou operační systém, úložiště a nasazené aplikace. [6]

Příkladem poskytovatelů služeb v modelu IaaS může být například Amazon nebo Rackspace.

3.1.3 Platform as a Service

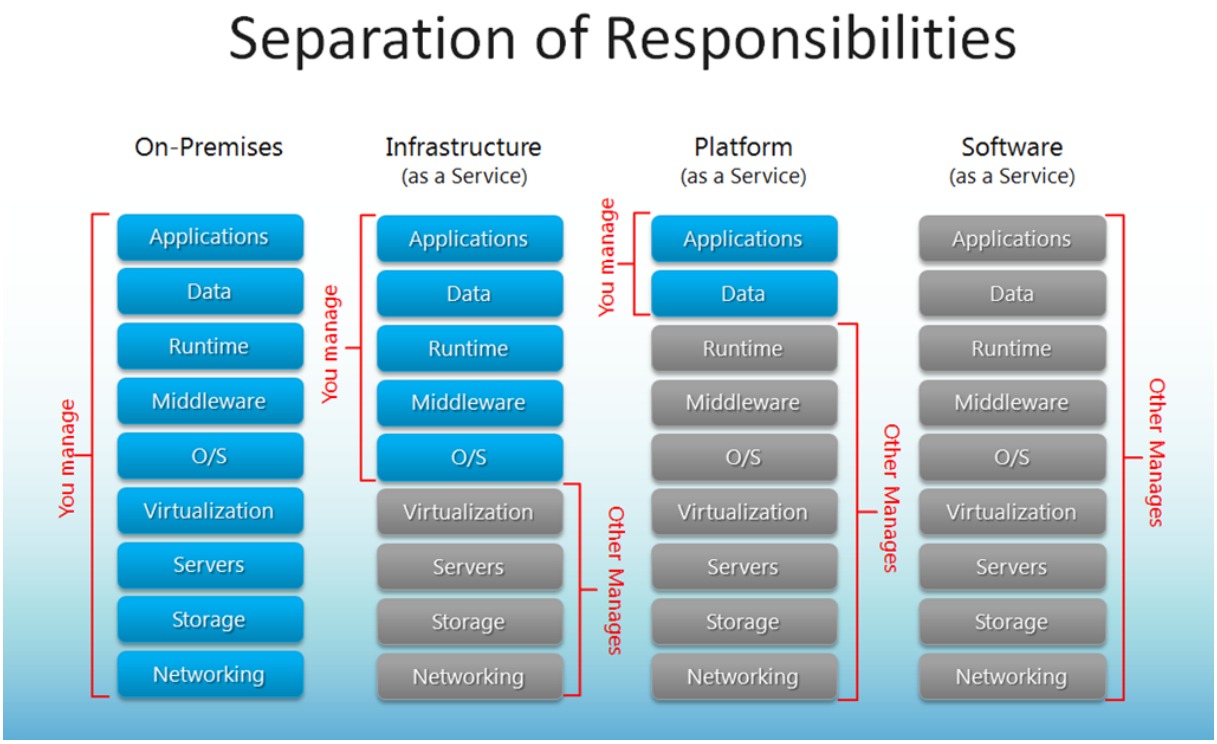
V modelu PaaS jsou poskytovány všechny zdroje, které jsou poskytovány v modelu IaaS, tedy hlavně základní hardwarová infrastruktura. K tomu je navíc poskytován operační systém, middleware a runtime prostředí. Zákazník se o tyto oblasti nemusí starat, ale může nasadit pouze takové aplikace, které jsou vytvořeny pomocí programovacích jazyků, knihoven, služeb a nástrojů podporovaných poskytovatelem služby. Zákazník má kontrolu nad nasazenými aplikacemi a jejich daty, popřípadě má možnost do určité míry konfigurovat nastavení aplikačního prostředí. [6]

Řešením typu PaaS je například právě Windows Azure (výjimkou je VM Role - viz 3.2.1- která spíše patří do kategorie IaaS).

3.1.4 Software as a Service

V tomto modelu jsou zákazníkovi poskytovány nejen všechny části z modelů IaaS a PaaS, ale i samotné aplikace s jejich daty. Zákazník tedy získá od poskytovatele aplikaci nasazenou na cloudové infrastruktuře. Aplikace jsou přístupné prostřednictvím různých klientských zařízení, a to buď prostřednictvím tenkého, nebo tlustého klienta. Klient nenese odpovědnost za cloudovou infrastrukturu, operační systémy ani aplikace s možnou výjimkou určité customizace poskytovaných aplikací. [6]

Příkladem Software as a Service jsou například Google Apps nebo Office 356.

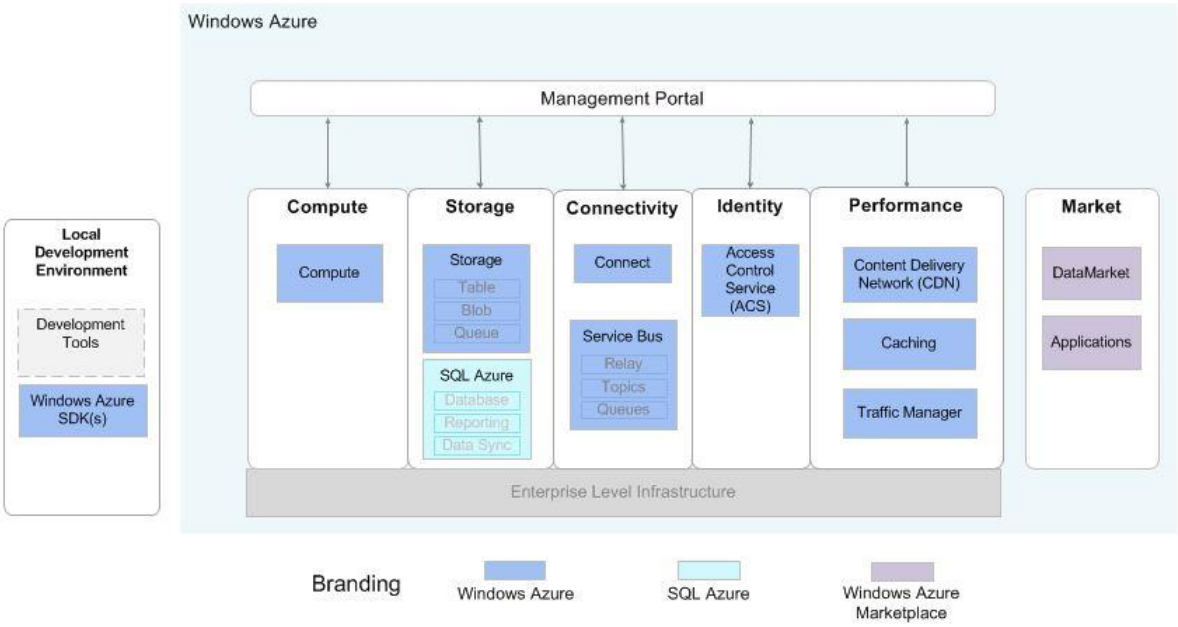


Obrázek 1 - Porovnání IaaS, PaaS, SaaS a modelu on-premise [7]

3.2 Windows Azure

Windows Azure lze chápat jako operační systém vytvořený pro cloud. [2] Tak, jako u každého operačního systému, je jeho účelem poskytnout abstraktní vrstvu nad fyzickým hardwarem a množinu služeb, které mohou aplikace využívat. Na rozdíl od tradičních operačních systémů, které abstrahují nad jedním strojem s jeho CPU, pevnými disky, periferiemi nebo grafickou kartou, Windows Azure abstrahuje na úrovni množiny serverů. Poskytuje tak platformu k tvorbě a nasazení služeb a aplikací ve zcela virtuálním prostředí. Zákazník tedy neurčuje, zda bude služba instalována na serveru A nebo B nebo zda bude nasazena na disk C nebo D. Stejně jako tradiční operační systémy, Windows Azure poskytuje rozhraní pro ukládání a manipulaci s daty. Ani zde se však neurčuje konkrétní server nebo jeho pevný disk. Výběr optimálního úložiště, replikaci a zálohu dat, to vše zajišťuje Windows Azure. Uživatel určuje pouze logické, nikoliv fyzické uspořádání a umístění dat. Obdobné je to i s potřebnými zdroji. Uživatel může vyčíslit nároky aplikace na zdroje logickou cestou, např. množství operační paměti, velikost úložiště, počet jader, ale o přidělení konkrétních fyzických prostředků, které tyto zdroje zajistí, se již nemusí starat. Systém sám zajišťuje rozložení aplikace na vhodné servery v závislosti na aktuální zátěži. Sám zabezpečuje, aby byly jednotlivé servery rozmístěny s odlišným geografickým umístěním (např. z důvodu velkých přírodních katastrof). Pokud je některá služba nepředpokládaně ukončena, proběhne automatický pokus o její restart. Pokud se zjistí závada na samotném serveru, je tento odpojen a instance služby, která na něm běžela, je přesunuta na jiný server.

Samotná platforma Windows Azure je logicky rozdělena do několika částí (viz *Obrázek 2 - Struktura Windows Azure Platform*). Jedná se především o „Compute“, „Storage“, „Connectivity“, „Identity“, „Performance“ a „Market“. Tyto části budou blíže popsány v následujících kapitolách. Část „Local Development Environment“ zahrnuje nástroje pro vývoj aplikací pro platformu Windows Azure. Jedná se především o SDK (Software Development Kit), který je k vývoji aplikací nezbytný a v současné době dostupný ve verzi 1.6. Ve verzi pro .NET se integruje přímo do Visual Studia 2010 a poskytuje cenné nástroje, jako například emulátor Azure Compute a Azure Storage (viz dále). „Management Portal“ poskytuje webové rozhraní pro administraci nasazených aplikací, uživatelských účtů, správu databáze Azure SQL a další. Většinu akcí dostupných prostřednictvím Management Portalu lze provádět i pomocí poskytovaných API. [8]



Obrázek 2 - Struktura Windows Azure Platform [9]

3.2.1 Compute

Windows Azure Compute umožňuje samotný běh aplikací v cloudu. Aplikace jsou rozděleny do jednotlivých komponent, které se nazývají role. Role jsou relativně samostatné celky, které mají poměrně úzkou specializaci. Každá role pak definuje rozhraní pomocí tzv. end-pointů (HTTP, HTTPS, TCP), které může sloužit ke komunikaci mezi jednotlivými rolemi a mezi vzdálenými klienty. Každá role pak může fungovat v určitém počtu instancí. Vyšší počet instancí zajišťuje větší dostupnost aplikace a odolnost. Pokud některá instance zaznamená potíže a selže, požadavky jejích klientů jsou automaticky přesměrovány na ostatní instance. Přitom je zajišťován load balancing, tedy zátěž je dle potřeby rozložena na jednotlivé instance. Větší počet instancí je také výhodou při aktualizaci služby na nejnovější verzi. Postupně jsou pak vypínány jednotlivé instance, které jsou aktualizovány, ale mezitím je vždy stále aktivní určitý počet instancí verze původní, takže je zajištěn hladký přechod na nejnovější verzi, a to bez jakéhokoliv výpadku v dostupnosti služby [1]. Aktualizaci lze provést dvěma způsoby. Pokud se jedná pouze o menší změny v konfiguraci služby, lze pouze nahrát nový konfigurační XML soubor. Pokud se jedná o větší změny v konfiguraci (např. změna end -pointů) nebo samotný zásah do kódu aplikace, je nutné znovu nahrát celou aplikaci.

Každá instance role má definovanou velikost. Přesněji řečeno – každá instance běží právě na jednom virtuálním stroji a každý virtuální stroj má právě jednu instanci. U těchto virtuálních strojů můžeme určovat jejich velikost. Ta v tomto případě znamená množství přidělených prostředků, které smí instance sama spotřebovávat. Konkrétně se jedná o počet jader CPU, velikost operační paměti a

velikost přiděleného diskového místa. Velikost instance se liší od velmi malé (XS) až po velmi velkou (XL). Přehled velikostí rolí a přidělených prostředků ukazuje *Tabulka 1 - Porovnání velikostí virtuálních strojů Windows Azure Compute.*

Tabulka 1 - Porovnání velikostí virtuálních strojů Windows Azure Compute [10] [11]

Velikost virtuálního stroje	Počet jader CPU	Operační paměť	Velikost lokálního úložiště[11]	Alokovaná šířka pásma	Cena za hodinu
XS	1 sdílené	768 MB	19480 MB	5 Mbps	$ 0,02
S	1	1,75 GB	229400 MB	100 Mbps	$ 0,12
M	2	3,5 GB	500760 MB	200 Mbps	$ 0,24
L	4	7 GB	1023000 MB	400 Mbps	$ 0,48
XL	8	14 GB	2087960 MB	800 Mbps	$ 0,96

Web Role

Web role je určena pro front-end webové aplikace a webové služby, které jsou nasazeny na webovém serveru IIS7.5. Nejčastěji se jedná o aplikace ASP.NET nebo služby Windows Communication Foundation. Principiálně lze ale hostovat nejen .NET aplikace, ale i další - například PHP nebo Java aplikace (Microsoft poskytuje příslušné SDK).

Worker Role

Worker role je určena pro back-end aplikace. Může zpracovávat asynchronní úlohu s dlouhou dobou zpracování nezávisle na uživatelském vstupu. Služba má tři metody - OnStart(), OnStop() a Run (). OnStart() slouží k inicializaci a přípravě zdrojů. V metodě Run() je samotná činnost, za kterou je role zodpovědná. Často se jedná o nekonečný cyklus, ukončovaný nějakou vnitřní podmínkou. Metoda OnStop() slouží k finalizaci role a uvolnění zdrojů. Poté je role ukončena. Pro komunikaci mezi Web Role a Worker role slouží často Azure Queue (viz Windows Azure Storage dále). Z webové role přijde určitý požadavek na zpracování, který je uložen do fronty. Worker role si pak po zpracování každého požadavku odebere nový z fronty. Často jsou tímto způsobem zpracovávány požadavky, které jsou časově náročné, ale není třeba je zpracovávat synchronně, tedy uživatel nemusí čekat na zpracování požadavku.

Oddělení front-endu a back-endu aplikace je vhodné z hlediska škálovatelnosti a možnosti přidělit každému zvlášť odpovídající prostředky. Například webová role pro příjem uživatelských požadavků

[11] 6144 MB je rezervováno pro systémové soubory. Hodnoty platí pro web role a worker role. [9]

bude nasazena v pěti instancích velikosti Medium, kdežto náročný back-end bude mít osm instancí velikosti Large. Stejně tak lze rozdělit např. jen samotný front-end do několika nezávislých rolí, z nichž každá bude mít různou náročnost na zdroje a dostupnost.

VM Role

VM role je výjimkou z modelu PaaS, do kterého patří zbytek Windows Azure. VM role je spíše blíže modelu IaaS. V tomto případě má zákazník zodpovědnost i za operační systém. Je ale limitován systémem Windows Server 2008 R2 (Standard / Enterprise). Zákazník musí poskytnout obraz disku s vlastní instalací tohoto operačního systému. Cílem VM role je tedy usnadnit přechod existujících on-premise aplikací na Windows Azure. Jedná se spíše o kompromisní dočasné řešení, které s sebou nese řadu rizik a neguje řadu výhod PaaS řešení. [12]

3.2.2 Storage

Storage je jednou z nejdůležitějších částí Windows Azure Platform. V cloudovém prostředí jsou na data a manipulaci s nimi kladeny zcela jiné požadavky než v klasickém pojetí jednoho stroje a jeho fyzických úložišť. V cloudu musí být abstrahováno od fyzického uložení dat a to musí být nahrazeno logickým uložením. Pokud mají být aplikace škálovatelné a odolné proti selhání, je nutné zajistit replikaci dat mezi jednotlivými instancemi v reálném čase. Data musí být k dispozici minimálně na několika různých místech, nejlépe i s rozdílnou geografickou polohou. Není garantováno, že dva po sobě následující požadavky téhož klienta budou obslouženy pomocí téže instance (přestože ve většině případů je tento přístup preferován), přitom každá instance musí mít okamžitě dostupná nejaktuálnější data. Ty musí být konzistentní a nemůže se stát, že jeden klient zapíše data, druhý je záhy přečte a získá ještě zastaralé neaktualizované informace. Je tedy nutné přizpůsobit práci s daty podmínkám cloudu.

Do Storage můžeme zařadit dvě hlavní části. První je Windows Azure Storage, což je systém pro ukládání dat speciálně přizpůsobený cloudovému prostředí. Ten se dále dělí na jednotlivé části – Blob, Table, Queue, Drive. Každá z nich má své výhody a nevýhody a je vhodná pro jiný typ dat a užití. Druhou velkou částí je SQL Azure, což je relační databáze přizpůsobená pro cloud. Poslední částí, kterou uvádím spíše jako doplněk a do kontrastu s předchozími částmi, je Local Storage, který není vhodný jako úložiště pro cloud a jsou na něm dobře vidět nevýhody tradičního přístupu k ukládání dat mimo cloud.

Local Storage

Local Storage je lokální úložiště každé instance. Lze ji přirovnat k tradičnímu pevnému disku, i když se jedná o logický, nikoliv fyzický zdroj. Každá instance má přidělenou maximální možnou velikost Local Storage (viz *Tabulka 1 - Porovnání velikostí virtuálních strojů Windows Azure Compute*) V Local

Storage lze vytvářet soubory a složky a dále s nimi manipulovat. Local Storage se nachází na stejném virtuálním stroji jako rodičovská instance role. U Windows Azure Storage tomu tak není. Z toho plynou výhody i nevýhody. Přístupová doba k datům je nesrovnatelně rychlejší. Protože se ale jedná o lokální data instance, neprobíhá žádná synchronizace s ostatními instancemi. K datům uloženým v Local Storage má přístup jen rodičovská instance. Navíc se nejedná o perzistentní úložiště. V případě upgradu nebo selhání instance nebo serveru jsou data ztracena. Užití Local Storage se tedy nabízí jako úložiště dočasných souborů a cachování vzdálených dat.

Windows Azure Storage

Windows Azure Storage je systém pro ukládání dat speciálně přizpůsobený cloudovému prostředí. Má vysokou škálovatelnost. Z tohoto hlediska je vhodnější než SQL Azure, kde je tato škálovatelnost nižší [13]. Nicméně SQL Azure obsahuje mnoho pokročilých funkcí relačních databází. Windows Azure Storage poskytuje neomezenou úložnou kapacitu (každý ze Storage účtů ale může mít maximálně 100 TB) oproti 150 GB [14] u SQL Azure. V závislosti na prioritách a požadavcích je pak v určitých situacích vhodnější použití SQL Azure a jindy je vhodnější použití Windows Azure Storage.

Blob

Zkratka Blob znamená Binary Large Object. Blob Storage je jakousi cloudovou alternativou souborového systému. Základní úložnou jednotkou je Blob, který můžeme chápat jako ekvivalent souboru. Blob nemůže existovat samostatně, ale musí být umístěn do tzv. kontejneru. Kontejnery slouží pro logickou organizaci Blobů a také k usnadnění manipulace s nimi. Kontejner lze procházet, nastavovat pro něj přístupová práva atd. Kontejner lze chápat jako alternativu ke klasickým složkám. Avšak ani kontejnery nemohou existovat samostatně. Každý kontejner patří do určitého Storage Accountu. Ten sdružuje nejen objekty typu Blob, ale i Table nebo Queue. Jedna Azure Subscription může mít i více Storage Accountů. Každá nasazená aplikace pak může využívat jeden nebo více těchto účtů nebo může být naopak využíván pouze jeden účet všemi aplikacemi.

Blob Storage umožňuje vytvářet dva typy Blobů. Page Blob a Block Blob. Typ je deklarován při tvorbě Blobu a nelze jej později změnit. Tyto dva typy se liší režimem přístupu k datům. Block Blob je určen pro sekvenční přístup. Skládá se z jednotlivých bloků, jejichž maximální velikost je 4MB, celková velikost Block Blobu pak činí 200 GB. Page Blob je určen k přístupu typu random access a je rozdělen do stránek po 512 bytech. Jeho maximální velikost pak činí 1 TB. [15]

Každý Blob a kontejner obsahuje určitá metadata – např. velikost obsahu, jeho kódování, jazyk, kontrolní součet MD5 atd. Kromě toho má každý Blob a kontejner definován mód přístupu. Ten nabývá jedné ze tří následujících hodnot [16]:

- **Plný veřejný přístup pro čtení -** Data daného kontejneru nebo Blobu jsou přístupná anonymním požadavkům. Takto mohou být procházeny Bloby v rámci daného kontejneru, nikoliv však kontejnery v rámci nadřazeného Storage Accountu.
- **Plný veřejný přístup ke čtení pouze pro Bloby -** Data Blobu v daném kontejneru jsou k dispozici anonymním požadavkům, ale nejsou k dispozici data kontejneru. Nelze procházet Bloby v daném kontejneru.
- **Bez veřejného přístupu -** Data kontejnerů a Blobů mohou být čtena pouze majitelem příslušného Storage Accountu.

K Blobům a kontejnerům se přistupuje pomocí REST API [17]. Každý objekt má definovanou unikátní adresu URI, která ho identifikuje. K objektům se pak přistupuje pomocí standardních HTTP 1.1 požadavků. Výhodou je, že tyto API mohou být volány nejen zevnitř cloudu, ale i vnějšími aplikacemi. Aplikace mimo cloud tak mohou přistupovat ke cloudovým datům a dokonce je možné vytvářet hybridní aplikace, které sice nejsou nasazeny na platformě Windows Azure, ale celá jejich datová vrstva je v cloudu. .NET Framework poskytuje řadu obalových tříd pro Blob Storage REST API a umožňuje tak jednodušší manipulaci s Bloby a kontejnery.

Table

Blob Storage je ideálním řešením pro ukládání nestrukturovaných dat v cloudu. V okamžiku, kdy je třeba pracovat s daty strukturovanými, je nutné zvolit jiné řešení. Mimo cloud by se nabízela relační databáze. Použití klasické relační databáze v cloudu je ale problematické, a to především kvůli obtížné škálovatelnosti takového řešení. I když je možné tradiční relační databáze do určité míry škálovat, toto řešení je velmi nákladné. Z tohoto důvodu Windows Azure nabízí Table Storage – mechanismus pro práci se strukturovanými daty v cloudu. Nejedná se však o relační databázi upravenou pro cloud – tou je naopak SQL Azure, která je vhodná, pokud je nutné použít některých pokročilých funkcí relačních databází. Menší komplexnost Table Storage je vyvážena nižší cenou, větší kapacitou a větší škálovatelností.

Na rozdíl od Blob Storage jsou data organizována do tabulek namísto kontejnerů. Každá tabulka musí mít rodičovský Storage Account. Jednotlivé položky v tabulkách se nazývají entity. Každá entita musí mít minimálně následující vlastnosti: TimeStamp, PartitionKey a RowKey [1]. TimeStamp je read-only hodnota data a času, kdy entita byla vložena do tabulky a je automaticky nastavována serverem. PartitionKey a RowKey dohromady tvoří unikátní identifikátor entity. PartitionKey slouží k definici partition, do které entita spadá. Partition je množina entit se stejným PartitionKey. Rozdělení tabulky na jednotlivé partitions je výhodné z hlediska škálovatelnosti a výkonu. Entity náležející ke stejné partition mají garantováno, že se nachází na stejném fyzickém serveru a přístupová doba bude

minimální. Při potřebě rozšířit tabulku stačí přidat novou partition – rozložením dat na více serverů bude zachován požadovaný výkon [1]. Vlastnosti entity (Properties) jsou definovány jako dvojice jméno-hodnota. V relačních databázích jsou vlastnosti definovány pro celou tabulku, jakožto její sloupce. Table Storage má definovány vlastnosti na úrovni entity, nikoliv tabulky. Každá entita v tabulce tedy může mít libovolný počet různých vlastností. V jedné tabulce můžeme teoreticky uchovávat registrační informace zákazníků a stejně tak produkty z našeho e-shopu. Přesto je výhodné, aby každá tabulka sdružovala pouze příbuzný druh entit (např. pouze produkt nebo pouze zákazníka). Table tedy nezaručuje přítomnost určitých vlastností u entity ani typovou integritu těchto vlastností. Také neudržuje závislosti a vazby mezi jednotlivými tabulkami. Díky tomu je zachována velká škálovatelnost. Komunikace s Table Storage probíhá také pomocí REST API jako u Blob Storage.

Queue

Azure Queue Storage je úložiště, do kterého lze ukládat zprávy. Zpráva zpravidla reprezentuje určitý požadavek na budoucí akci. Jeden aktér (instance role, aplikace) vždy uloží zprávu do Queue (tvůrce zprávy) a jiný aktér si pak zprávu ve vhodný čas vyzvedne a zpracuje ji (konzument zprávy). Tento model se nejčastěji používá pro komunikaci mezi Web a Worker rolemi, ale vzhledem k poskytovaným REST API může s Queue manipulovat i aplikace mimo cloud [2]. Výhodou tohoto mechanismu je, že minimalizuje závislost mezi tvůrcem a konzumentem zprávy. Aktéři na obou stranách nemusí mít o sobě navzájem žádnou znalost. Tvůrce zprávy ji jen vloží do Queue a už se nestará o to, kým a kdy bude zpráva zpracována. Konzument zprávy ji zpracuje až v okamžiku, kdy to pro něho bude výhodné. Počet konzumentů a tvůrců zpráv, kteří pracují s jednou Queue, se může dynamicky měnit, aniž by se navzájem ovlivňovali. To je výhodné také pro škálovatelnost, kdy lze takto velmi jednoduše přidávat a odebírat nové instance role (ať už jako tvůrce nebo konzumenta). Vhodné je použít Queue i jako bufferovací mechanismus. Při náhlém nárůstu zátěže jsou požadavky ukládány do fronty a postupně zpracovávány namísto toho, aby byly přímo odmítnuty. Pokud určitá část systému selže nebo má odstávku, Queue může stále přijímat a ukládat požadavky, i když konzumenti jsou v tu chvíli mimo provoz a zpracují je později. Výhodou také je, že konzumenti a tvůrci zpráv mohou být implementováni na jiné technologii a přesto mohou pohodlně komunikovat prostřednictvím fronty – např. front end na PHP a back end na .NET.

Každá Message musí být přiřazena určité Queue. Přitom každá Queue musí spadat pod určitý Storage Account (stejně jako Blob a Table). Zprávy v Queue nejsou plně perzistentní, mají nastavitelnou dobu životnosti, která je maximálně jeden týden. Vzhledem ke cloudovému prostředí není garantováno pořadí výběru zpráv. Nelze tedy počítat s tím, že Queue bude fungovat jako fronta typu FIFO.

Drive

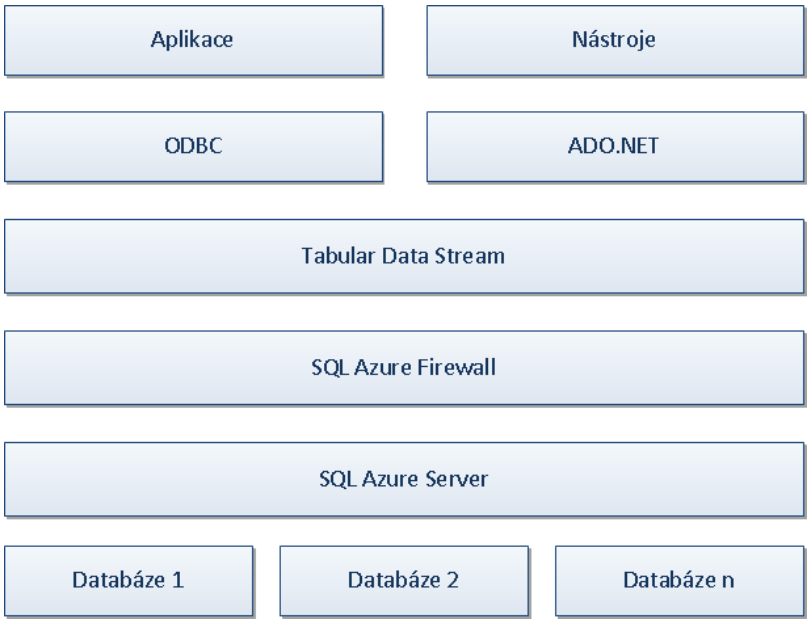
Jedním z problémů při migraci existující aplikace na Windows Azure je nutnost změnit způsob, jakým aplikace manipuluje s daty. Mnoho existujících aplikací používá jako úložiště pro určitou část svých dat tradiční pevný disk a používá NTFS API. K usnadnění migrace těchto aplikací byl vytvořen Azure Drive. Ten představuje perzistentní úložiště, vystupující jako NTFS jednotka. Je implementován jako Windows Azure Page Blob (viz. Blob Storage výše), který obsahuje obraz virtuálního pevného disku (VHD) formátovaného systémem souborů NTFS. Ten lze uploadovat pomocí standardních Blob API. Poté je konkrétní Blob deklarován jako disková jednotka. Velikost je omezena na minimum 16 MB a maximum 1TB. Virtuální stroj přitom dokáže pracovat až s 16 disky. Azure Drive je vhodné používat pouze jako dočasné řešení při migraci existující aplikace, ostatní mechanismy (Table, Blob, SQL Azure) jsou pro použití v cloudu vhodnější. [18]

SQL Azure

Jak bylo uvedeno v kapitolách výše, Azure Table Storage poskytuje velmi efektivní a škálovatelnou cestu, jak pracovat se strukturovanými daty. Nicméně se jedná o řešení, které je velmi vzdálené tradičnímu přístupu relačních databází, které mnoho on-premise řešení používá. Při snaze o migraci takového řešení na Windows Azure je nejjednodušší použít dosavadní technologii a přístup pro práci s daty nebo dokonce data samotná. SQL Azure toto umožňuje s malou nebo dokonce žádnou modifikací kódu aplikace. SQL Azure je relační databáze modifikovaná pro cloud. Neznamená to ale, že je přístupná pouze z cloudu. Jsou tedy možné dva modely. Celo-cloudový model uvažuje službu nasazenou v cloudu a využívající SQL Azure databázi. Hybridní model pak znamená SQL Azure databázi využívanou on-premise aplikací. SQL Azure vychází z Microsoft SQL Serveru a využívá jako dotazovacího jazyka T-SQL.

Oproti Windows Azure Storage nenabízí Azure SQL neomezenou úložnou kapacitu. K dispozici jsou dvě varianty. Web Edition je možné zvolit v konfiguraci 1 GB nebo 5 GB. Business Edition je nabízena v konfiguracích 10, 20, 30, 40, 50, 100 a 150 GB. Kromě kapacity se obě edice ničím neliší.

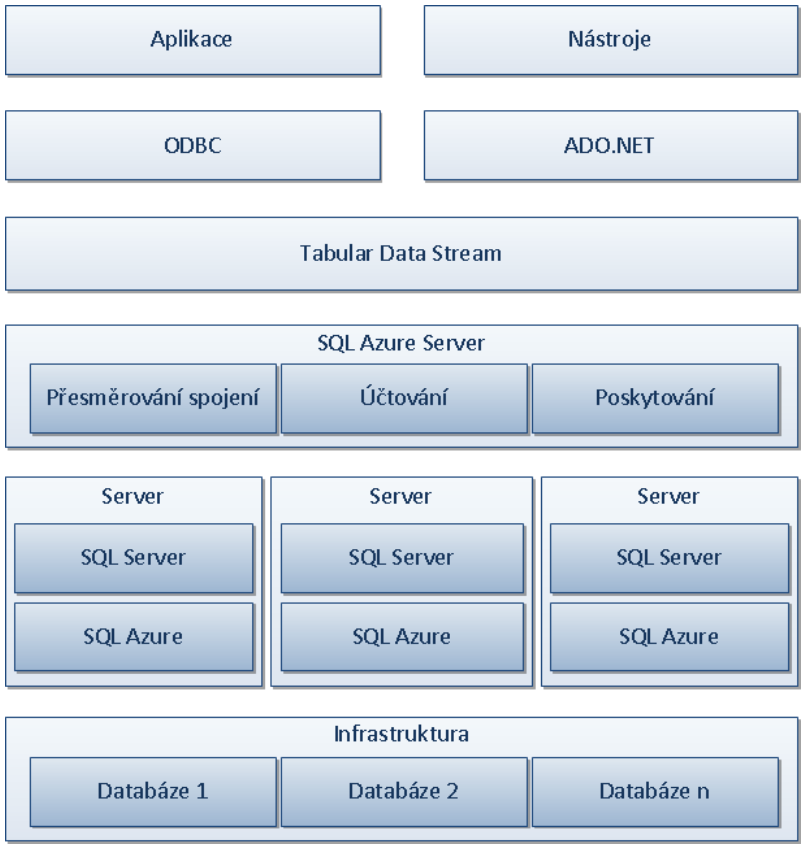
Strukturu SQL Azure z logického hlediska zachycuje *Obrázek 3 - SQL Azure - Logická struktura*. Komunikace s klienty probíhá přes protokol TDS (Tabular Data Stream) a prochází Azure SQL Firewallem. Ten lze konfigurovat přes Management Portal (webové rozhraní pro správu Windows Azure) nebo prostřednictvím REST API. Pak už následuje přímo SQL Azure server. Nejedná se však o fyzický SQL server, ale o virtuální server. Ten poskytuje mezivrstvu abstrakce, která odstiňuje klienty od skutečné back-endové infrastruktury.



Obrázek 3 - SQL Azure - Logická struktura [1]

.

Z fyzického hlediska (viz *Obrázek 4 - SQL Azure - Fyzická struktura*) SQL Azure server přesměrovává požadavky klientů na skutečné SQL servery, které existují jako řada replik. Jedna z replik je vždy stanovena jako primární a na ni jsou směrovány veškeré požadavky. Pokud z jakéhokoliv důvodu selže, je vyřazena z fondu replik a jedna z ostatních replik je ustanovena jako primární a přebírá její místo. Mezitím je nově vytvořena náhrada repliky, jež selhala. SQL Azure server může přesouvat jednotlivé repliky mezi fyzickými servery a zajistit tak optimální dostupnost. Samotná nejnižší vrstva (Infrastructure) obsahuje síťové a úložné prostředky, zajišťující bezpečné uložení a optimální replikaci fyzických dat.



Obrázek 4 - SQL Azure - Fyzická struktura [1]

3.2.3 Connectivity

Windows Azure Connect a Service Bus

Windows Azure Connect a Service Bus jsou dvě samostatné služby, přesto jsem se je rozhodl popsat v jedné kapitole. Důvodem je, že se zdánlivě velmi podobají svými cíli, i když každá je implementována jiným způsobem a používá se pro jiné scénáře. Obecně se jedná o propojení služeb a zdrojů mezi sebou, především pak mezi cloudem a on-premise. Bez použití Connect nebo Service Bus je toto propojení obtížné, a to především díky problémům plynoucím z použití firewallů a Network Adress Translation. Rozdíly mezi oběma službami jsou následující [19]:

Windows Azure Service Bus je messaging service operující v aplikační vrstvě. Vystupuje v roli prostředníka mezi klientem a cílem, ke kterému se potřebuje připojit. Služba si registruje své end-pointy u Service Bus. Ta je zařadí do svého katalogu a zpřístupní je pro budoucí klienty. Klient si vyžádá spojení se službou od Service Bus. Ta vytvoří spojení a od této chvíle funguje jako prostředník pro předávání zpráv mezi službou a klientem. Příkladem použití může být propojení WCF služby na Windows Azure a WCF služby provozované on-premise.

Windows Azure Connect virtualizovaná síťová služba, běžící ne na aplikační, ale na síťové vrstvě. Používá se pro hybridní Windows Azure Aplikace (část aplikace je v cloudu, část on-premise). Azure Connect lze přirovnat k virtuální privátní síti (VPN) mezi instancemi Azure rolí a on-premise stroji. Typickým příkladem použití je zajištění přístupu z Windows Azure instance role do on-premise zdrojů v podnikové síti (SQL servery, souborové systémy,…). Možnost takto napojit on-premise zdroje usnadňuje migraci existujících aplikací na platformu Windows Azure.

3.2.4 Identity

Access Control Service

ACS je postavena na modelu claim-based identity. Uživateli je přiřazen token, který obsahuje informace o jeho vydavateli, a dále pak jednotlivé claims. Oproti tomu, co název napovídá, se nejedná o informace autorizační (jaké akce uživatel smí a nesmí provádět), ale autentifikační (určení identity). Každý claim pak obsahuje určitou informaci o vlastníkovi tokenu. Např. jméno, email, věk, pohlaví atd. Token tedy deklaruje autentifikaci, na samotné aplikaci pak je, aby zajistila na základě těchto údajů autorizaci. Každý token je podepsán jeho vydavatelem. Aplikace pak nemusí mít vlastní mechanismy pro autentifikaci – registraci, nastavení a změnu hesla, vyplnění osobních údajů atd.

V modelu tedy vystupují tři základní aktéři:

- **Uživatel** – Prokazuje se bezpečnostním tokenem.
- **Poskytovatel Identity** – Vydává uživateli token, za který se zaručuje. Např. Facebook účet, Google Account nebo Live ID.
- **Aplikace** – Identifikuje uživatele na základě tokenu poskytovatele, kterému věří a s kterým umí spolupracovat.

Tento model sice má výhody, ale má i řadu nedostatků. Pokud chce aplikace podporovat více poskytovatelů identity a jejich tokenů, musí je implementovat zvlášť, protože každý z nich poskytuje odlišné informace a tokeny mají odlišný formát. Přidání nového poskytovatele pak znamená citelný zásah do samotného kódu aplikace. Je tedy vhodné přidat jednoho prostředníka, který by přijímal tokeny mnoha poskytovatelů, zpracoval je a vydal jediný token v unifikovaném formátu. Aplikaci by pak nezajímalo, zda se uživatel přihlásil prostřednictvím Google účtu nebo Live ID. Aplikace by také byla odstíněna od přidávání a odebírání podporovaných poskytovatelů a tyto změny by se jí nijak nedotkly. Takovýto prostředník se nazývá Federation Provider a Access Control Service je jeho konkrétní implementací pro Windows Azure [20].

3.2.5 Performance

Content Delivery Network

CDN poskytuje mechanismus pro zvýšení dostupnosti velkého objemu dat koncovým uživatelům. Koncový uživatel může k požadovanému obsahu přistupovat rychleji, poskytovatel obsahu zaznamená sníženou zátěž svých serverů, která se přesune na síť Content Delivery Network. Bez tohoto řešení by služba potenciálně nebyla schopná obsloužit takové množství požadavků s dostatečnou rychlostí nebo by nadměrná zátěž dokonce vedla k její nedostupnosti.

CDN pracuje na bázi fyzického rozmístění obsahu do mnoha uzlů. Uživatel je pak obsloužen geograficky nejbližším uzlem. Vždy, když je obdržen požadavek na obsah z CDN, služba zkontroluje, zda je tento dostupný na nejbližším uzlu. Pokud ne, je obsah získán přímo z Blob Storage a zároveň uložen i na nejbližší uzel. V současné době existuje 24 různých uzlů po celém světě [21]. Azure CDN se používá pro distribuci obsahu z Blob Storage. Je vhodný pro statický obsah, u vysoce dynamického obsahu je jeho použití neekonomické a z hlediska výkonu spíše kontraproduktivní [22]. Typickým příkladem použití je např. distribuce aktualizací nebo multimediálního obsahu. Microsoft sám používá tuto platformu pro distribuci Windows Update a mapových dlaždic Bing Maps.

Caching

Windows Azure Caching Service poskytuje mechanismus, jak zrychlit a zefektivnit práci s často používanými daty ukládáním do mezipaměti. Oproti lokálnímu ukládání na disk (Local Storage) je toto řešení rychlejší a škálovatelné. Jedná se vlastně o distribuovanou cache. Cachování ale neprobíhá automaticky, je nutné zdroje explicitně vložit a číst z cache pomocí poskytnutých API.

Možnosti a výhody Caching Service [23]:

- **Snadná škálovatelnost** – Jednoduchá změna velikosti prostřednictvím konfiguračního souboru.
- **Flexibilita** – Lze ukládat mnoho typů objektů (XML, binární data,…).
- **Snadná konfigurace** - Lze akcelerovat webové aplikace bez zásahu do kódu pouze změnou konfiguračního souboru.
- **Řízení přístupu** - Lze zabezpečit přístup k cache pomocí Access Control Service.

Použití je vhodné především pro [23]:

- **Read-only data** – U těchto dat je garantováno, že se nebudou měnit, a lze je tedy vždy bezpečně získávat přímo z cache. Příkladem může být seznam států.
- **Aplikační zdroje** – Data, která jsou platná pro celou aplikaci, nikoliv konkrétní instanci nebo session.
- **Data uživatelské session** – Často čtená a měněná. Je zde velký prostor pro zvýšení výkonu. Například data uživatelského nákupního košíku mohou být udržována v paměti a až při dokončení transakce uložena do perzistentního úložiště.

Traffic Manager

Komponenta Windows Azure Traffic Manager umožňuje distribuovat zátěž (load balancing) mezi dvě a více služeb v jednom nebo více datových centrech. Díky tomu je zvýšen výkon a dostupnost těchto služeb. Traffic Manager se každých třicet vteřin dotazuje na dostupnost služby. Pokud se mu třikrát za sebou nepodaří získat odpověď 200 OK, bude považovat službu za nedostupnou a odstraní ji z fondu spravovaných služeb.

Traffic Manager může pracovat ve třech různých režimech [24]:

- **Failover** – Všechna zátěž je směřována na jednu službu. Pokud je identifikována jako nedostupná, jsou všechna spojení přesměrována na službu následující. Protože detekce selhání trvá 90 vteřin (viz výše) a minutu či dvě změna DNS záznamů, služba bude při selhání několik minut nedostupná.
- **Performance** – V tomto režimu jsou příchozí požadavky vždy směrovány na službu s aktuálně nejnižší odezvou.
- **Round Robin** – Příchozí požadavky jsou rovnoměrně směrovány na všechny služby ve fondu, aby se zajistila stejnoměrná zátěž.

3.2.6 Market

Windows Azure Marketplace nabízí platformu pro prodej a distribuci cloudových služeb modelem SaaS [25]. Zajišťuje přitom všechny potřebné transakce, takže nemusí dojít k přímému kontaktu dodavatele s odběratelem. Microsoft přitom garantuje určitou úroveň kvality tím, že nabízené aplikace musí být předem schváleny a certifikovány.

Druhou, neméně důležitou, částí na Marketplace je obchod daty. Zákazníci zde mohou nakupovat potřebná data nebo je naopak nabízet. K pořízeným datům pak mohou přistupovat pomocí REST API nebo Open Data Protokolu. Marketplace je dostupný na adrese https://datamarket.azure.com/.

4 Analýza

Cílem kapitoly je na globální úrovni analyzovat problematiku aplikace a navrhnout řešení. U aplikace budou definovány požadavky, její architektura a mnohé další parametry, které je třeba určit před samotnou implementací.

4.1 Požadavky na aplikaci

Cílem této kapitoly je definovat požadavky, které aplikace musí splňovat. Ke každému požadavku je pak definována metrika, podle které se bude zjišťovat, zda byl požadavek splněn či nikoliv.

Nejdříve je identifikován problém, poté je stanoveno poslání aplikace jakožto cesta k řešení tohoto problému. V dalším kroku jsou již definovány konkrétní požadavky, které musí být v souladu s posláním. Ty jsou děleny na funkční, kvalitativní a technologické.

4.1.1 Problém

Pohyb městem pro osoby s omezenou možností pohybu (pro potřeby práce uvažujeme dvě základní skupiny – cyklisty a vozíčkáře) je obtížný z důvodu častého výskytu nepředpokládaných překážek a komplikací. Některé tyto překážky mohou zamýšlený přesun zcela znemožnit nebo jej alespoň velmi zkomplikovat. Plánování trasy je tedy nezbytností. Přitom jedinec nemá sám k dispozici dostatečné množství informací, aby se mohl racionálně rozhodnout. Neexistují vhodné nástroje pro takovéto plánování, a to především přímo v terénu.

4.1.2 Poslání

Poslání lze v tomto kontextu chápat jako globální určení smyslu existence aplikace a její zaměření na obecné úrovni. Oproti dále definovaným cílům a požadavkům nemá poslání konkrétní metriky a lze obtížněji určovat stupeň jeho naplnění. Jedná se tedy spíše o jakési hrubé vytyčení dlouhodobého směru, na jehož základě jsou dále určeny (a jemuž jsou podřízeny) konkrétní požadavky a parametry.

Posláním této aplikace je poskytovat cílovým skupinám (cyklisté a osoby se sníženou schopností pohybu) nástroje mapování a analýzu pohybu a na základě nasbíraných dat umožnit efektivní plánování trasy. Důraz je kladen na samotnou komunitu a interakci mezi jejími jednotlivými členy. Z dat nasbíraných jedním členem komunity mají užitek všichni ostatní.

4.1.3 Požadavky funkční

V této kapitole definuji cílové skupiny, na které je aplikace zaměřena. Dále je zde uvedena motivace jednotlivých skupin k použití aplikace a jejich požadavky na funkčnost.[12] Z požadavků jednotlivých skupin je pak vytvořen výsledný seznam funkčních požadavků.

Požadavky uživatelské – cyklisté sportovní a rekreační

Z hlediska cílové skupiny cyklistů budeme uvažovat dvě hlavní skupiny uživatelů, a to z hlediska motivace, která je vede k jízdě.

První skupinou jsou rekreační a profesionální cyklisté, kteří používají jízdní kolo jakožto prostředek ke sportu a zábavě. Jejich pohnutky používat software pro cyklisty pramení především z touhy zaznamenávat si svoje trasy a výsledky a sdílet je s přáteli. Software je pro ně jakýsi záznamník, v němž mají přehled svých tras, který si mohou kdykoliv prohlédnout. Mají přehled o tom, jakou vzdálenost už zvládli celkem ujet, jaká místa již navštívili, a mohou si udělat obrázek o tom, jak se jejich výkonnost mění z dlouhodobého hlediska. Sdílení s přáteli pak funguje jako motivátor, kdy se cyklista může pochlubit svými úspěchy nebo doporučit přátelům trasu, kterou stojí za to absolvovat. Tuto skupinu cyklistů lze považovat za velmi početnou, nejpočetnější ze všech cílových skupin. Není to ale hlavní cílová skupina, na kterou je program zaměřen. A to především proto, že na trhu již existují profesionální a kvalitní produkty na ni zaměřené, kterým by bylo velmi obtížné konkurovat a tato aplikace si to ani neklade za cíl. Přesto je tato skupina důležitá. Vzhledem k její početnosti je vhodná jakožto součást uživatelské základny, která poskytne značnou část potřebných dat pro funkci programu, zpětnou vazbu k jeho vývoji a rozšíří povědomí o programu mezi širší okruh uživatelů. Proto je vhodné, aby i členové této skupiny shledali aplikaci užitečnou a alespoň ji vyzkoušeli.

Mezi hlavní požadavky této skupiny patří:

- Možnost sledovat svou trasu při jízdě a zaznamenávat ji.
- Možnost zjišťovat informace o zaznamenaných trasách jako je vzdálenost, trajektorie, výškový profil, čas, počáteční a cílové destinace atd.
- Možnost navazovat přátelství s ostatními vybranými členy komunity.
- Možnost sdílet trasy s přáteli a možnost vidět trasy přátel.
- Možnost vidět aktuální pozici přátel na mapě.
- Možnost upozorňovat na význačná místa na mapě (cykloservisy, občerstvení u cyklostezek, přírodní body zájmu,…).

[12] V práci není blíže popsána metodika získávání a zpracování uživatelských požadavků, protože to není předmětem práce a nemá to přímý vliv na naplnění hlavního cíle.

Požadavky uživatelské – cyklisté používající jízdní kolo jako dopravní prostředek

Druhou skupinou jsou cyklisté, kteří pravidelně používají jízdní kolo jako prostředek dopravy městem (nebo jiným složitým a nepředvídatelným terénem). Tato skupina je sice méně početná než skupina sportovních a rekreačních cyklistů uvedená výše, ale z hlediska zacílení aplikace má větší prioritu. Pro tuto skupinu budou sice užitečné funkce poptávané skupinou první, ale mnohem větší důraz bude klást na usnadnění rozhodování při plánování cesty městem. Aplikace zde má také mnohem větší prostor pro vlastní přínos, protože zatím nejsou k dispozici vhodné nástroje, které by se zabývaly touto problematikou. Z tohoto důvodu mají požadavky této skupiny větší prioritu než požadavky první skupiny cyklistů. Pokud dojde ke konfliktu požadavků těchto dvou skupin a požadavky budou neslučitelné, mají přednost požadavky skupiny druhé. V zájmu této skupiny je také sledování trasy a její ukládání, i když motivace je zcela jiná než u skupiny první. Na rozdíl od osobního přehledu nebo sdílení s přáteli je mapování důležité pro samotné budoucí plánování trasy. Pokud totiž nebude nasbíráno dostatečné množství dat, nebude mít uživatel dostatečné podklady pro kvalitní rozhodování při plánování trasy. Trasy zaznamenané jednotlivými uživateli jsou z tohoto hlediska nutné pro správné fungování aplikace.

Mezi hlavní požadavky druhé skupiny cyklistů patří:

- Možnost upozorňovat na význačná místa na mapě, a to především ta, která znamenají potenciální nebezpeční, komplikace nebo omezení pohybu.
- Možnost sledovat svou polohu při jízdě a zaznamenávat trasu.
- Možnost ohodnotit svou trasu stupněm obtížnosti.
- Možnost použít při plánování trasy a body zájmu ostatních uživatelů (nejen přátel).
- Možnost plánovat přímo v terénu.

Požadavky uživatelské – vozíčkáři

Do této skupiny patří obecně osoby s tělesným handicapem, jehož důsledkem je omezená možnost pohybu. Pro zjednodušení je skupina dále nazývána jako „vozíčkáři“, ale můžeme do ní například zařadit téměř kohokoliv, kdo má zvýšenou citlivost na bariéry pohybu, tedy například osoby s dětskými kočárky a dalšími objemnými náklady. Jedná se o jednu ze dvou hlavních cílových skupin. Druhou cílovou skupinou jsou cyklisté pravidelně cestující městem nebo jiným obtížným terénem. Priorita těchto dvou skupin je totožná. Jakékoliv požadavky, které jsou v konfliktu, musí být vyřešeny ve prospěch obou stran. A to buď vhodným kompromisem, nebo rozdílnou implementací problému v závislosti na tom, do jaké skupiny patří aktuální uživatel.

Tato cílová skupina je označována jako jedna z hlavních z toho důvodu, že je zde očekáván největší potenciální přínos pro její členy. V současné době neexistují vhodné a efektivní nástroje pro plánování trasy obtížným terénem. Přitom poptávka po takovémto řešení existuje.

Motivace a požadavky této skupiny jsou v zásadě velmi podobné jako u druhé hlavní cílové skupiny.

Mezi hlavní uživatelské požadavky skupiny patří:

- Možnost upozorňovat na význačná místa na mapě, a to především ta, která znamenají potenciální nebezpeční, komplikace nebo omezení pohybu.
- Možnost sledovat svou trasu při jízdě a zaznamenávat ji.
- Možnost ohodnotit svou trasu stupněm obtížnosti.
- Možnost použít při plánování trasy a body zájmu ostatních uživatelů (nejen přátel).
- Možnost plánovat přímo v terénu.
- Jednoduchost rozhraní pro ovládání aplikace, aby menší tělesný handicap neznemožňoval práci s aplikací.

Sjednocení uživatelských požadavků

Jak bylo výše popsáno, aplikace je zaměřena na tři cílové skupiny, z nichž dvě jsou považovány za prioritní. Mohlo by se zdát, že se jedná o skupiny poměrně rozdílné a tedy i jejich požadavky budou špatně slučitelné. Přestože motivace každé ze skupin se liší a důvody požadování určitých funkcí jsou různé, samotné požadavky jsou však velmi podobné. Nehledě na rozdílnost těchto pohnutek může tedy poskytnout aplikace funkce, které budou vyhovovat požadavkům všech cílových skupin.

Výsledný seznam požadavků, který vznikl sjednocením požadavků jednotlivých cílových skupin, je následující:

Tabulka 2 - uživatelské funkční požadavky na aplikaci

Identifikátor	Název požadavku Metrika
UP 1	**Možnost upozorňovat na význačná místa na mapě a odlišit je dle typu.**
	1. Uživatel může označit místo na mapě jakožto bod zájmu. 2. Bod zájmu musí mít jméno a volitelně vysvětlující popis. 3. Uživatel vybírá kategorii bodu zájmu - na výběr jsou alespoň kategorie „Nebezpečí“, “Varování“, „Obecná informace“ a „Doporučení“.
UP 2	**Možnost sledovat svou polohu při jízdě a zaznamenávat trasu.**
	1. Aplikace poskytuje informaci o aktuální uživatelově poloze. 2. Uživatel může vynutit začátek a konec zaznamenávání trasy.
UP 3	**Možnost ohodnotit své trasy stupněm obtížnosti.**
	1. Uživatel může svou ukončenou trasu ohodnotit známkou z předem definované stupnice a vyjádřit tak názor na její obtížnost.
UP 4	**Možnost navázat přátelství s ostatními členy komunity.**
	1. Uživatel může vyhledávat mezi ostatními uživateli aplikace. 2. Uživatel může navázat přátelství s jiným uživatelem. 3. Přátelství musí být potvrzeno druhou stranou.
UP 5	**Možnost procházet trasy vlastní a trasy přátel.**
	1. Uživatel může zobrazit seznam svých dosavadních tras. 2. Uživatel může zobrazit detaily o konkrétní trase. 3. Uživatel může prohlížet trasy náležející konkrétnímu příteli.
UP 6	**Možnost sledovat pozici přátel na mapě.**
	1. Uživatel vidí poslední známou pozici svých přátel na mapě. 2. Pozice nemusí být aktualizována v reálném čase, ale v libovolném intervalu (nejméně však jednou za přítelovo zaznamenání jedné trasy).
UP 7	**Možnost plánovat trasu i na základě dat nasbíraných ostatními členy komunity.**
	1. Uživatel vidí body zájmu vytvořené ostatními uživateli. 2. Uživatel může komentovat cizí body zájmu. 3. Uživatel si může zobrazit i cizí trasy ve své blízkosti. Tyto jsou anonymizovány a nejsou spojitelné s konkrétním uživatelem.
UP 8	**Možnost plánovat trasu přímo v terénu.**
	1. Uživatel si může vyhledat body zájmu v blízkosti své aktuální pozice

Identifikátor	Název požadavku Metrika
	v souladu s UP 7. 2. Uživatel si může vyhledat trasy v blízkosti své aktuální pozice v souladu s UP 7.
UP 9	**Jednoduché ovládání.**
	1. Snaha o minimalizaci komplexnosti uživatelského rozhraní. Tento požadavek nemá tvrdou metriku splnění, jedná se pouze o doporučení, na které bude brán maximální možný zřetel, pokud to nebude v konfliktu s ostatními cíli. Tento požadavek je pro potřeby posouzení splnění funkčních požadavků aplikace automaticky považován za splněný.

Metrika

Oblast funkčních požadavků je z hlediska naplnění cílů aplikace považována za splněnou, pokud budou v aplikaci implementovány výše uvedené funkční požadavky (*Sjednocení uživatelských požadavků*). Pokud bude nedodržen alespoň jeden požadavek, tato oblast funkčních požadavků se bude považovat za nesplněnou.

4.1.4 Požadavky kvalitativní

Samotná implementace všech funkčních požadavků není zárukou uživatelsky použitelné aplikace. Pokud bude aplikace chybová, nestabilní nebo často nedostupná, znemožní to uživatelům efektivní práci a hrozí, že aplikaci zcela opustí. Požadavky kvalitativní mají za úkol zajistit použitelnost aplikace pro cílového uživatele tím, že stanoví určité standardy týkající se kvality zpracování, spolehlivosti a dostupnosti. Požadavky kvalitativní jsou považovány za splněné, pokud jsou splněny ve všech následujících oblastech, a to za použití příslušných níže uvedených metrik.

Stabilita

Stabilitou aplikace je v tomto případě míněna její odolnost proti pádu následkem chyby při běhu programu. Cílem je minimalizovat počet takovýchto pádů. V případě vzniku chyby by aplikace měla vytvořit kontrolovanou výjimku. Tyto výjimky by měly být vždy zachyceny a měly by být zpracovány. Uživatel by měl být upozorněn na chybu. Pokud se nejedná o chybu kritickou, nemělo by dojít ke zbytečnému ukončení aplikace. Pokud se o chybu kritickou jedná a není jiná možnost, měla by být aplikace řádně a kontrolovaně ukončena tak, aby byla zanechána ve stabilním a konzistentním stavu a byly uvolněny všechny zdroje.

Z hlediska stability je tedy cílem minimalizovat počet pádů aplikace. Kontrolované ukončení není pro potřeby metriky tohoto požadavku považováno za pád. Přesto některým pádům nelze zabránit (např. způsobeným nepředpokládaným chováním hardwaru nebo OS mobilního zařízení). Marketplace pro WP7 poskytuje informace o počtu pádů a o počtu stažení aplikace. Pády přitom nelze přímo přiřadit k jednotlivým instalacím a ani nelze změřit, jak dlouho byla instalace aktivní, než došlo k pádu. Nebude tedy uvažováno časové hledisko, ale pouze počet stažení a počet pádů. Aplikace bude považována za stabilní, pokud na každé stažení bude připadat nejvýše jeden pád. Poměr počtu pádů a počtu stažení bude tedy menší nebo roven jedné.

Dostupnost Aplikace

Dostupnost je v tomto případě uvažována pro serverovou část aplikace. Tato musí být k dispozici v dostatečné míře. Cílem je poskytnout vnější dostupnost alespoň 98%. Skutečnou míru dostupnosti lze vypočítat jako podíl času, kdy byla aplikace za měřené období skutečně dostupná, a celkového času za dané období. Pro průkaznost hodnot bude vyžadováno, aby délka období, ke kterému se dostupnost vztahuje, byla alespoň jeden měsíc (30 dní). Není požadováno měření skutečné dostupnosti, za dostačující se považuje, pokud poskytovatel zaručí poskytnutí služby se stejnou nebo vyšší dostupností než oněch 98%.

Dostupnost Dat

Vzhledem k tomu, že v závislosti na implementaci bude pravděpodobně datové úložiště odděleno od samotné aplikační části, je nutné ověřit jeho dostupnost zvlášť. Pokud sice aplikace na straně serveru poběží bezchybně, ale její datové úložiště bude nedostupné, je to totéž, jako by byla nedostupná celá aplikace. Tento požadavek lze považovat za splněný, pokud bude dosažena dostupnost datového úložiště alespoň 98%. Pro potřeby tohoto požadavku bude chápána dostupnost jako podíl času, kdy bylo datové úložiště za měřené období skutečně dostupné, a celkového času za dané období. Opět budeme požadovat délku vztažného období 30 dní. Postačuje pouze garance poskytovatele úložiště, není třeba provádět skutečné měření.

4.1.5 Požadavky technologické

Platforma

Práce je již od začátku koncipována jako aplikace na platformě Windows Azure. Zásadní podmínka, která musí být za jakýchkoliv okolností splněna, je tedy ta, že aplikace musí být vyvinuta pro platformu Windows Azure.

4.2 Případy užití

Cílem této kapitoly je představit aplikaci z vnějšího pohledu a zachytit jednotlivé aktéry v systému a na globální úrovni jejich interakce se systémem. Zjednodušeně řečeno, kdo v systému vystupuje a jakou interakci musí provést, aby naplnil svůj určitý cíl.

Toto bude demonstrováno pomocí diagramu případů užití (Use Case Diagram) dle notace UML 2.4.1 [26] skupiny Object Management Group. Tato specifikace popisuje diagram případů užití takto:

Případy užití jsou prostředkem pro specifikaci vyžadovaného užití systému. Obvykle jsou použity k zachycení požadavků systému, tedy toho, co se od systému očekává, že bude vykonávat. Klíčovými koncepty spojenými s případy užití jsou aktéři, případy užití a subjekt. Subjekt je systém, ke kterému se případy užití vztahují. Uživatelé a jakékoliv další systémy, které mohou být v interakci se subjektem, jsou reprezentováni jako aktéři. Aktéři vždy znázorňují entity mimo systém. Požadované chování subjektu je určeno jedním nebo více případy užití, které jsou definovány vzhledem k požadavkům aktérů. [26]

Poté, co byly definovány cílové uživatelské skupiny a jejich požadavky, je třeba tyto zohlednit v diagramu případů užití. Nejdříve budou definováni aktéři a systém. Poté bude představen diagram případů užití. Každý příklad bude následně detailněji popsán. Diagram poslouží jako přehledný pohled na systém z hlediska uživatele a jeho požadavků a přitom abstrahuje od vnitřního uspořádání a fungování systému. Pro větší přehlednost je diagram rozdělen do dvou částí, které jsou na sobě nezávislé.

První část zahrnuje případy užití související s uživatelem, jeho účtem a jeho přáteli. Patří sem např. registrace uživatele, přihlášení nebo žádost o přátelství. Případy užití v této části jsou označovány identifikátorem PU_UP_XX. PU značí případ užití, UP značí skupinu *Účet a přátelé* a XX je číslo případu užití (pro jednociferná čísla je první X nahrazeno nulou).

Druhá část zahrnuje případy užití spadající do kategorie tras, bodů zájmu a plánování. Patří sem např. založení nového bodu zájmu, komentování existujícího, přidání nové trasy nebo plánování trasy nové. Případy užití v této části jsou označovány identifikátorem PU_POIT_XX. Význam PU a X je totožný jako u první skupiny. POIT značí skupinu *Body zájmu a trasy*, kde POI reprezentuje body zájmu a T trasy.

V každé části je nejdříve uveden diagram případů užití. Následně jsou uvedené scénáře detailně popsány. Standardní průběh případu užití je naznačen v části *Hlavní scénář*. Jedná se vždy o sekvenci po sobě jdoucích kroků. Aby se mohlo přistoupit ke kroku následujícímu, je nutné úspěšně dokončit krok předchozí. Pokud má nějaký krok podpoložky (a, b, c,…), jedná se o navzájem se vylučující alternativy, nikoliv po sobě následující kroky.

4.2.1 Systém

Logicky je aplikace rozdělena na dvě oddělené části, klientskou na mobilním zařízení a serverovou na Windows Azure. Obě tyto části by bylo možné dále dělit. Z hlediska systému v diagramu případů užití však budeme považovat aplikaci za jeden monolitický celek (tedy nebudeme uvažovat rozdělení na serverovou a klientskou část). Z pohledu uživatele a jeho požadavků je totiž takovéto dělení zanedbatelné. Uživatel pracuje s aplikací a nemá (a ani nechce mít) tušení o struktuře aplikace, jejích vrstvách a rozdělení. Pro uživatele je systém atomický a uživatel na něj má požadavky jako na celek. Z tohoto diagramu pak tedy vyplývají požadavky jak pro klientskou, tak i pro serverovou část.

4.2.2 Aktéři

Z hlediska funkčních požadavků na systém byly definovány tři cílové skupiny uživatelů – dvě pro cyklisty a jedna pro osoby se sníženou schopností pohybu. Sjednocením požadavků těchto skupin vznikl jednotný seznam požadavků na funkčnost. Funkce v něm definované však budou dostupné všem uživatelům bez ohledu na to, do jaké skupiny uživatel patří. Nepředpokládá se tedy diferenciace funkcí a možností aplikace podle typu uživatele. Pro potřeby use case diagramu tedy není nutné rozlišovat uživatele na základě jeho příslušnosti k cílové skupině, ale pouze podle toho, jaký typ klienta používá. V současné době jsou implementovány dva typy klientů – mobilní a webový. Pro potřeby use case diagramu budou z uživatelského hlediska rozlišováni dva typy aktérů – mobilní klient a webový klient. Některé případy užití jsou dostupné pouze pro jednoho aktéra, většina je pak společných. Tyto rozdíly ve funkčnosti jsou dány především možnostmi a silnými a slabými stránkami platformy, na které jsou jednotliví klienti implementováni.

4.2.3 Případy užití pro oblast „Uživatelský účet a přátelé“ (PU_UP)

Obrázek 5 - Diagram případů užití pro oblast PU_UP [autor]

Tabulka 3 - Případ užití PU_UP_01

ID a název	PU_UP_01 Vytvořit účet
Popis	Uživatel vyplní požadované údaje a vytvoří si účet, se kterým se poté pokaždé přihlašuje.
Aktéři	Uživatel webového klienta, Uživatel mobilního klienta
Podmínky	-
Hlavní scénář	1. Uživatel zvolí akci „vytvořit účet“. 2. Uživatel vyplní požadované uživatelské jméno, heslo, typ účtu (cyklista/vozíčkář) a svou adresu pro elektronickou poštu. 3. Uživatel potvrdí zadané údaje. 4. Systém zkontroluje správnost údajů (viz výjimky). Pokud nastane výjimka, uživatel musí provést opravu údajů, u kterých je indikována chyba (návrat ke kroku 2). 5. Systém uloží zadané údaje a přihlásí uživatele (koncový stav tohoto scénáře je totožný s PU_UP_02).
Výjimky	Uživatelské jméno již existuje. Nebyly vyplněny všechny povinné údaje. Vyplněné údaje nesplňují podmínky (délka řetězce u jména a hesla).

Tabulka 4 - Případ užití PU_UP_02

ID a název	PU_UP_02 Přihlásit se
Popis	Uživatel se přihlásí s předem vytvořeným účtem (PU_UP_01) do systému.
Aktéři	Uživatel webového klienta, Uživatel mobilního klienta
Podmínky	PU_UP_01 Úspěšně dokončen kdykoliv dříve.
Hlavní scénář	1. Uživatel vyplní své přihlašovací jméno. 2. Uživatel vyplní své heslo. 3. Uživatel potvrdí zadané údaje. 4. Systém zkontroluje správnost údajů (viz výjimky). Pokud nastane výjimka, uživatel je upozorněn na příčinu a dle typu výjimky se vrací ke kroku 1 nebo 2. 5. Systém přihlásí uživatele.
Výjimky	Uživatelské jméno není registrováno. Neplatné heslo.

Tabulka 5 - Případ užití PU_UP_03

ID a název	PU_UP_03 Nastavit fotografii
Popis	Uživatel použije integrovaný fotoaparát v klientském zařízení, aby svému účtu přiřadil fotografii.
Aktéři	Uživatel mobilního klienta
Podmínky	PU_UP_01 Úspěšně dokončen kdykoliv dříve. PU_UP_02 Úspěšně dokončen na aktuální instanci klienta kdykoliv dříve.
Hlavní scénář	1. Uživatel zvolí v nastavení akci „Nastavit fotografii“. 2. Klientské zařízení se přepne do režimu fotografování. Pokud se to nepodaří (viz výjimky), scénář končí neúspěchem. 3. Uživatel zabere vybraný objekt a zvolí akci „Vyfotografovat“. 4. Uživateli se zobrazí pořízený snímek. a. Uživatel není spokojen. Návrat ke kroku 3. b. Uživatel je spokojen a potvrdí fotografii. 5. Fotografie je uložena a přiřazena uživatelskému účtu.
Výjimky	Klientské zařízení není vybaveno fotoaparátem.

Tabulka 6 - Případ užití PU_UP_04

ID a název	PU_UP_04 Změnit heslo
Popis	Uživatel změní dosavadní heslo ke svému účtu za nové.
Aktéři	Uživatel webového klienta, Uživatel mobilního klienta
Podmínky	PU_UP_01 Úspěšně dokončen kdykoliv dříve. PU_UP_02 Úspěšně dokončen na aktuální instanci klienta kdykoliv dříve.
Hlavní scénář	1. Uživatel v nastavení zvolí akci „Změna hesla“. 2. Uživatel vyplní nové heslo. 3. Systém ověří správnost údajů. Pokud je heslo nevyhovující (viz výjimky), uživateli je indikována příčina chyby. Návrat ke kroku 2. 4. Heslo je změněno na novou hodnotu.
Výjimky	Heslo nesplňuje požadavky (minimální počet znaků). Zamýšlené nové heslo je shodné s heslem původním.

Tabulka 7 - Případ užití PU_UP_05

ID a název	PU_UP_05 Žádat o přátelství
Popis	Uživatel vyhledá jiného uživatele a pošle mu žádost o navázání přátelství.
Aktéři	Uživatel webového klienta, Uživatel mobilního klienta
Podmínky	PU_UP_01 Úspěšně dokončen kdykoliv dříve. PU_UP_02 Úspěšně dokončen na aktuální instanci klienta kdykoliv dříve.
Hlavní scénář	1. Uživatel v sekci „Přátelé“ zvolí možnost „Přidat přítele“. 2. Uživatel zadá neprázdný řetězec do vyhledávacího pole. 3. Systém prohledá všechny uživatele (kromě aktuálního). Vybere ty, u nichž uživatelské jméno nebo adresa odpovídá hledanému výrazu nebo ho obsahuje jako svou součást. 4. Uživateli se zobrazí výsledky dotazu včetně základních informací o jednotlivých uživatelích. a. Uživatel nalezl, koho hledal. Pokračuje krokem 5. b. Uživatel nenalezl toho, koho hledal, a chce hledat znovu. Vrací se na krok 2. c. Uživatel nenalezl, koho hledal, a nepřeje si hledat dále. Scénář selhal. 5. Uživatel označí vybraného přítele a vybere akci „Poslat žádost o přátelství“.
Výjimky	Nenalezen cílový uživatel vyhovující zadání. Cílový uživatel je již přítelem aktuálního uživatele.

Tabulka 8 - Případ užití PU_UP_06

ID a název	PU_UP_06 Přijmout přátelství
Popis	Uživatel potvrdí žádost o přátelství a tím toto přátelství vstoupí v platnost.
Aktéři	Uživatel webového klienta, Uživatel mobilního klienta
Podmínky	PU_UP_01 Úspěšně dokončen kdykoliv dříve. PU_UP_02 Úspěšně dokončen na aktuální instanci klienta kdykoliv dříve. PU_UP_05 Úspěšně dokončen a žádost o přátelství byla odeslána aktuálnímu uživateli z tohoto scénáře.
Hlavní scénář	1. Uživateli se zobrazí notifikace, že je žádán o potvrzení přátelství. Je informován, kdo je žadatel a o jeho základních informacích. 2. Uživatel zareaguje na žádost: a. Ignoruje ji. Scénář selhal. Případ užití lze kdykoliv opakovat, dokud nenastane PU_UP_06, krok 2a vztahující se k téže žádosti. b. Odmítne ji. Scénář selhal a nelze ho znovu opakovat. c. Přijme ji. Pokračuje se krokem 3. 3. Oba zúčastnění uživatelé jsou od tohoto okamžiku považováni za přátele.
Výjimky	Uživatel žádost odmítne nebo ignoruje.

Tabulka 9 - Případ užití PU_UP_07

ID a název	PU_UP_07 Zrušit přátelství
Popis	Uživatel zruší přátelství s jedním ze svých dosavadních přátel.
Aktéři	Uživatel webového klienta, Uživatel mobilního klienta
Podmínky	PU_UP_01 Úspěšně dokončen kdykoliv dříve. PU_UP_02 Úspěšně dokončen na aktuální instanci klienta kdykoliv dříve. PU_UP_08 Právě úspěšně skončil a mezitím nebyl zahájen žádný jiný scénář, ani nebyla ukončena aplikace.
Hlavní scénář	1. Uživatel vybere přítele, se kterým se již nechce dále přátelit. 2. Uživatel zvolí akci „Zrušit přátelství“. 3. Zobrazí se potvrzovací dialog, zda opravdu zrušit přátelství. 4. Uživatel v dialogu zvolí a. „Ano“. Scénář pokračuje krokem 5. b. „Ne“. Selhání scénáře. 5. Přátelská vazba mezi oběma uživateli je odstraněna. Uživatelé se už navzájem neuvidí v seznamu přátel. Přátelství lze později opět navázat dle PU_UP_05/PU_UP_06.
Výjimky	-

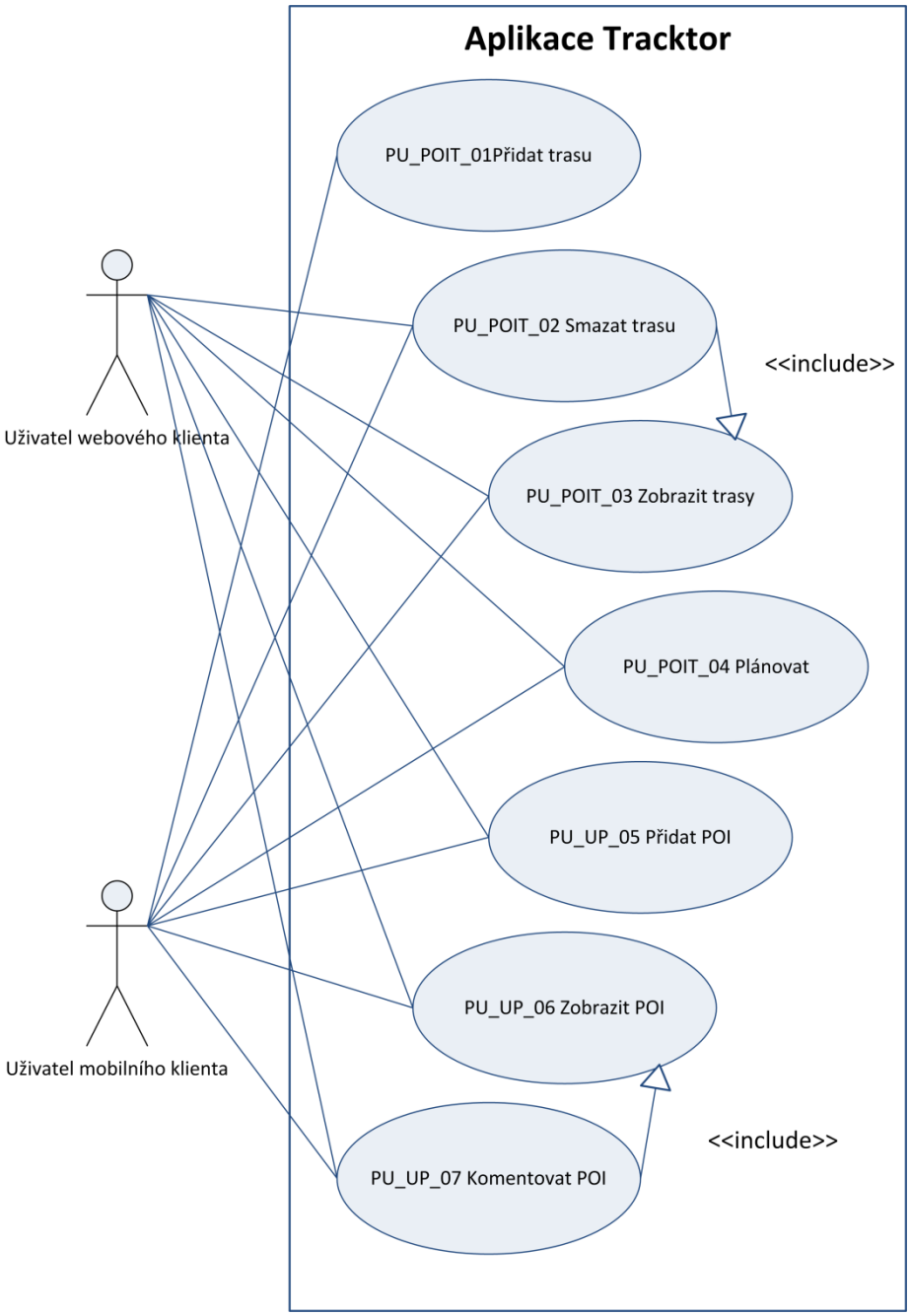
Tabulka 10 - Případ užití PU_UP_08

ID a název	PU_UP_08 Prohlížet přátele
Popis	Uživatel zobrazí seznam svých přátel.
Aktéři	Uživatel webového klienta, Uživatel mobilního klienta
Podmínky	PU_UP_01 Úspěšně dokončen kdykoliv dříve. PU_UP_02 Úspěšně dokončen na aktuální instanci klienta kdykoliv dříve.
Hlavní scénář	1. Uživatel zvolí sekci „Přátelé“. 2. Pokud má uživatel nevyřízené žádosti o přátelství (PU_UP_05), zobrazí se notifikace. Pokud na ni uživatel zareaguje, scénář selhal a místo toho pokračuje scénář PU_UP_06. 3. Pokud uživatel nemá žádné přátele, scénář selhal. 4. Uživateli se zobrazí seznam přátel. Jsou zobrazeny pouze obecné informace, detaily lze zobrazit vždy jen pro jednoho přítele najednou (viz PU_UP_09).
Výjimky	Uživatel nemá žádné přátele. Uživatel zareaguje na notifikaci žádosti o přátelství a pokračuje scénářem PU_UP_06.

Tabulka 11 - Případ užití PU_UP_09

ID a název	PU_UP_09 Zobrazit detaily přítele
Popis	Uživatel si prohlédne detailní informace o jednom ze svých přátel.
Aktéři	Uživatel webového klienta, Uživatel mobilního klienta
Podmínky	PU_UP_01 Úspěšně dokončen kdykoliv dříve. PU_UP_02 Úspěšně dokončen na aktuální instanci klienta kdykoliv dříve. PU_UP_08 Právě úspěšně skončil a mezitím nebyl zahájen žádný jiný scénář, ani nebyla ukončena aplikace.
Hlavní scénář	1. Uživatel vybere ze seznamu přítele, u kterého si přeje zobrazit detaily. 2. Místo dosavadního seznamu přátel se zobrazí detailní informace o vybraném příteli – jméno, email, seznam tras, poslední známá poloha.
Výjimky	-

4.2.4 Případy užití pro oblast „Body zájmu a trasy“ (PU_POIT)



Obrázek 6 - Diagram případů užití pro skupinu PU_POIT [autor]

Tabulka 12 - Případ užití PU_POIT_01

ID a název	PU_POIT_01 Přidat trasu
Popis	Uživatel zaznamenává svoji trajektorii a po dokončení ji uloží mezi své trasy.
Aktéři	Uživatel mobilního klienta
Podmínky	PU_UP_01 Úspěšně dokončen kdykoliv dříve. PU_UP_02 Úspěšně dokončen na aktuální instanci klienta kdykoliv dříve.
Hlavní scénář	1. Uživatel zvolí možnost „Zahájit sledování trasy“ 2. Uživatel se pohybuje a systém zaznamenává jeho trajektorii. 3. Uživatel zvolí možnost „Zastavit sledování trasy“ 4. Když je trasa zastavena, má uživatel následující možnosti a. Pokračovat ve sledování trasy. Scénář pokračuje bodem 5. b. Zrušit sledování trasy. Scénář selhal. c. Uložit trasu. Scénář pokračuje bodem 5. 5. Uživatel zadá hodnocení obtížnosti trasy. 6. Trasa je uložena do seznamu uživatelových tras. Pokud je trasa příliš dlouhá pro přenos, k uložení nedojde. Uživatel je upozorněn. Scénář selhal. 7. Systém dopočítá a uloží sekundární data o trase (vzdálenost, náročnost atd.)
Výjimky	Uživatel nemá na klientském zařízení zapnutý/funkční GPS lokátor. Trasa má příliš mnoho bodů (při běžném provozu je pravděpodobnost překročení limitu mizivá).

Tabulka 13 - Případ užití PU_POIT_02

ID a název	PU_POIT_02 Smazat trasu
Popis	Uživatel odstraní jednu ze svých vlastních tras.
Aktéři	Uživatel webového klienta, Uživatel mobilního klienta
Podmínky	PU_UP_01 Úspěšně dokončen kdykoliv dříve. PU_UP_02 Úspěšně dokončen na aktuální instanci klienta kdykoliv dříve. PU_UP_03 Úspěšně dokončen, aniž by byl mezitím zahájen jiný scénář nebo ukončena aplikace.
Hlavní scénář	1. Uživatel zvolí ze seznamu trasu, kterou si přeje smazat. 2. Uživatel zvolí akci „Smazat trasu“. 3. Uživatel je dotázán, zda si opravdu přeje odstranit trasu. 4. Uživatel vybere požadovanou odpověď. a. Ano. Scénář pokračuje bodem 5. b. Ne. Scénář selhal. K odstranění nedojde. 5. Trasa je odstraněna ze seznamu uživatelových tras a zobrazený seznam je aktualizován.
Výjimky	Uživatel změní svůj záměr a v potvrzovacím dialogu zruší mazání.

Tabulka 14 - Případ užití PU_POIT_03

ID a název	PU_POIT_03 Zobrazit trasy
Popis	Uživatel si prohlédne seznam svých vlastních tras.
Aktéři	Uživatel webového klienta, Uživatel mobilního klienta
Podmínky	PU_UP_01 Úspěšně dokončen kdykoliv dříve. PU_UP_02 Úspěšně dokončen na aktuální instanci klienta kdykoliv dříve.
Hlavní scénář	1. Uživatel vybere sekci „Trasy“. 2. Systém se pokusí načíst trasy. a. Pokud uživatel žádné trasy nemá, zobrazí se prázdný seznam. Scénář končí. b. Uživateli se zobrazí seznam jeho vlastních tras. U každé z nich lze vidět detaily – vzdálenost, čas, počáteční a cílové místo. Scénář pokračuje bodem 3. 3. Uživatel si prohlíží seznam svých tras: a. Uživatel je spokojen. Scénář končí. b. Uživatel se rozhodne měnit, které z jeho tras jsou zobrazeny na mapě a které ne. Toho lze docílit změnou hodnoty zaškrtávacího políčka (checkbox) u každé trasy. Scénář pokračuje krokem 4. 4. Systém aktualizuje zobrazené mapy podle hodnot z kroku 3.
Výjimky	-

Tabulka 15 - Případ užití PU_POIT_04

ID a název	PU_POIT_04 Plánovat
Popis	Uživatel si nechá zobrazit trasy a POI v okolí, aby mohl plánovat svou trasu.
Aktéři	Uživatel webového klienta, Uživatel mobilního klienta
Podmínky	PU_UP_01 Úspěšně dokončen kdykoliv dříve. PU_UP_02 Úspěšně dokončen na aktuální instanci klienta kdykoliv dříve.
Hlavní scénář	1. Uživatel vybere volbu „Plánovat“. 2. Systém zjistí polohu místa, v jehož okolí chce uživatel plánovat trasu. a. Aktér: Uživatel webového klienta. Poloha je bod ve středu zobrazované části mapy. b. Aktér: Uživatel mobilního klienta. Poloha je rovna aktuální poloze zařízení s mobilním klientem. Pokud nelze aktuální polohu zjistit, scénář selhal. 3. Systém zjistí vzdálenost, do které má zobrazovat trasy a POI. Ta je závislá na míře přiblížení mapy (zoom). 4. Systém vrátí všechny trasy a POI, pro které platí, že jejich vzdálenost od daného bodu (krok 2) je menší nebo rovna vzdálenosti vypočítané v kroku 3. 5. Uživatel má zobrazeny všechny informace o bodech zájmu a trasách ve svém okolí, na základě kterých může učinit lepší rozhodnutí o volbě trasy.
Výjimky	V nastavené vzdálenosti nejsou žádné trasy ani POI. Aktér: Uživatel mobilního klienta – nelze zjistit aktuální polohu.

Tabulka 16 - Případ užití PU_POIT_05

ID a název	PU_POIT_05 Přidat POI
Popis	Uživatel vytvoří nový bod zájmu a vyplní o něm informace.
Aktéři	Uživatel webového klienta, Uživatel mobilního klienta
Podmínky	PU_UP_01 Úspěšně dokončen kdykoliv dříve. PU_UP_02 Úspěšně dokončen na aktuální instanci klienta kdykoliv dříve.
Hlavní scénář	1. Uživatel označí místo na mapě. 2. Zobrazí se dialog pro přidání POI. 3. Uživatel zadá jméno a popis POI a vybere jeho kategorii. Jméno a kategorie jsou povinné, popis volitelný. 4. Uživatel zadané údaje: a. Potvrdí. Scénář pokračuje bodem 5. b. Zamítne. Scénář selhal. 5. Systém zkontroluje, zda jsou vyplněny všechny povinné údaje. a. Ano. Scénář pokračuje bodem 6. b. Ne. Návrat ke kroku 3. 6. Systém uloží POI.
Výjimky	Nebyly vyplněny všechny požadované údaje (název, kategorie).

Tabulka 17 - Případ užití PU_POIT_06

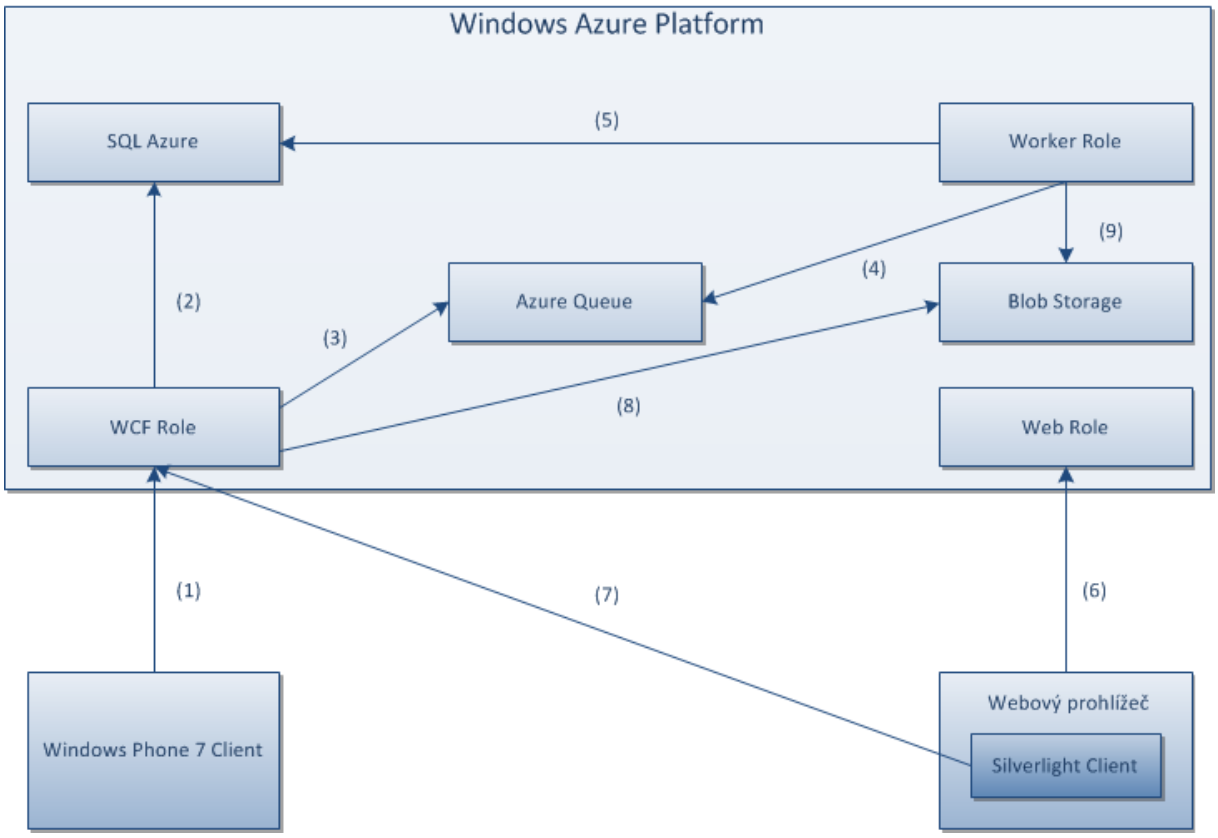
ID a název	PU_POIT_06 Zobrazit POI
Popis	Uživatel si nechá zobrazit body zájmu ve svém bezprostředním okolí.
Aktéři	Uživatel webového klienta, Uživatel mobilního klienta
Podmínky	PU_UP_01 Úspěšně dokončen kdykoliv dříve. PU_UP_02 Úspěšně dokončen na aktuální instanci klienta kdykoliv dříve.
Hlavní scénář	1. Uživatel zvolí akci „Zobrazit body zájmu“. 2. Systém zjistí polohu místa, v jehož okolí chce uživatel zobrazit body zájmu. a. Aktér: Uživatel webového klienta. Poloha je bod ve středu zobrazované části mapy. b. Aktér: Uživatel mobilního klienta. Poloha je rovna aktuální poloze zařízení s mobilním klientem. Pokud nelze aktuální polohu zjistit, scénář selhal. 3. Na mapě jsou zobrazeny body zájmu, které jsou v požadované vzdálenosti od daného bodu (z kroku 2). 4. Pokud v této vzdálenosti žádné body zájmu nejsou, scénář končí. 5. Uživatel vybere POI, u kterého chce zjistit podrobnosti nebo ho komentovat. Pokud žádný takový bod není, scénář končí. 6. Zobrazí se detaily vybraného POI. Jeho název, popis, kategorie a komentáře uživatelů (viz PU_POIT_07). 7. Uživatel si přeje: a. Komentovat daný bod zájmu. Pokračuje se scénářem PU_POIT_07. b. Ukončit prohlížení daného POI. Scénář končí.
Výjimky	V nastavení vzdálenosti nejsou žádné POI.

Tabulka 18 - Případ užití PU_POIT_07

ID a název	PU_POIT_07 Komentovat POI
Popis	Uživatel přidá komentář k existujícímu bodu zájmu.
Aktéři	Uživatel webového klienta, Uživatel mobilního klienta
Podmínky	PU_UP_01 Úspěšně dokončen kdykoliv dříve. PU_UP_02 Úspěšně dokončen na aktuální instanci klienta kdykoliv dříve. PU_POIT_06 Úspěšně dokončen, aniž by mezitím byl zahájen jiný scénář nebo ukončena aplikace.
Hlavní scénář	1. Uživatel u aktuálního POI vyplní text komentáře. 2. Uživatel zvolí akci „Odeslat komentář“. Volba není aktivní, pokud je text komentáře prázdný. 3. Komentář je uložen včetně informace o tom, kdo je autorem. Komentář je přiřazen k danému POI. 4. Kdykoliv nastane PU_POIT_06 u tohoto POI, prohlížející uživatelé uvidí všechny přidané komentáře.
Výjimky	-

4.3 Architektura Aplikace

Architekturu aplikace ukazuje *Obrázek 7 - Architektura aplikace*. Obecně se aplikace skládá z klientské a serverové části. Zpracování serverové části jsem prováděl já a je součástí této práce, klientskou pak Bc. Vít Čurda. Na obrázku je tato část ohraničena obdélníkem označeným jako „Windows Azure Platform“. Jednotlivé komponenty aplikace jsou označeny menšími obdélníky. Spojnice pak znázorňují interakci mezi komponentami. Každá spojnice je označena číslem, aby se na ni dalo v textu odkazovat. Šipka vede vždy od iniciátora komunikace k cíli komunikace. Iniciátor odešle požadavek, který cíl komunikace zpracuje a zpravidla na něj i odpoví (pokud byla odpověď vyžadována). Jednotlivé komponenty a jejich interakce jsou popsány níže.



Obrázek 7 - Architektura aplikace [autor]

WCF Role

Ve skutečnosti se jedná o Azure Web Role, ve které je hostovaná služba Windows Communication Foundation. Pro odlišení od druhé Web Role je v tomto textu označována jako WCF Role. Pro své klienty vystupuje jako standardní WCF služba a klienti vůbec nejsou ovlivněni tím, že služba je nasazena na platformě Windows Azure. Komunikace probíhá přes WCF a je totožná s komunikací s WCF službou provozovanou on-premise. Služba poskytuje rozhraní pro komunikaci (1, 7) – zpřístupňuje množinu metod. Jedná se o standardní metody ve smyslu objektově orientovaného paradigmatu. Tyto metody mohou tedy vracet určitou hodnotu a mohou jim být předány parametry.

Přitom parametry i návratové hodnoty mají určitý datový typ a je tedy zajištěna typová integrita přenesených dat. Systém podporuje primitivní datové typy a standardní typy .NET Frameworku (za předpokladu, že je konkrétní datový typ serializovatelný). Mimo to je v mnoha případech nutné pracovat s uživatelskými složenými datovými typy. Zde se jedná například o trasu, POI nebo uživatelský účet. Často se jedná o „obálky“ sdružující příbuzná data o jedné entitě (např. Účet a jeho uživatelské jméno, heslo, avatar,…), které se používají z toho důvodu, že metoda vrací maximálně jeden objekt. WCF služba pak definuje tzv. DataContract, což je vlastně definice třídy takovéhoto uživatelského objektu spolu s jejími atributy. Klient si pak přečte definici a po přijetí na jejím základě rekonstruuje objekt.

WCF Role je bránou, která obsluhuje veškeré uživatelské požadavky a odstiňuje klienty od back-end infrastruktury. Klient je tedy nezávislý na cloudové implementaci služby, na způsobu uložení dat atd. Změny v back-endu neznamenají nutnost zasahovat do kódu klienta. Kdyby se například změnil způsob nakládání s daty a místo SQL Azure by se implementovalo řešení závislé na Windows Azure Table Storage, klienta by to nijak neovlivnilo.

SQL Azure

Databáze SQL Azure představuje datové úložiště, ve kterém jsou uchovávána všechna perzistentní data – například informace o uživatelských účtech, trasách, bodech zájmu atd. K databázi nemají klienti přímý přístup, ale požadovaná data získávají zprostředkovaně voláním metod WCF Role (1). Přidávání a aktualizaci dat provádí jak WCF Role (2) (na základě uživatelských požadavků), tak Worker Role (5) (uložení zpracovaných uživatelských tras). Kromě uložení dat SQL Azure zajišťuje i operace nad geografickými daty, a to pomocí funkcí SQL Geography a SQL Geometry. Jedná se například o počítání délky tras a vzdálenosti mezi body.

Worker Role

Worker role slouží k asynchronnímu zpracování časově náročných požadavků. Jedná se především o zpracování tras nahraných na server. V okamžiku, kdy mobilní klient nahraje trasu na server pomocí WCF rozhraní (1), je trasa včetně základních dat uložena do databáze SQL Azure (2). Paralelně je také uložen požadavek na zpracování do Azure Queue (3). Tam se shromažďují požadavky jednotlivých tras, dokud si je postupně nevybere Worker Role (4) a nezpracuje je. Výsledky zpracování uloží do databáze SQL Azure (5). Protože velikost jedné zprávy v Message Queue je omezena, nejsou data ke zpracování uložena přímo ve zprávě, ale zpráva obsahuje pouze metadata určující, co a kde zpracovat. Samotná data jsou nejdříve WCF službou serializována do Blob Storage (8), odkud si je pak Worker role na základě zpracovávané zprávy opět vybere a deserializuje (9).

Samotné zpracování dat o trase spočívá ve vypočtení sekundárních dat na základě dat již známých. Jedná se například o spočítání výškového profilu, kumulativního převýšení a obtížnosti trasy. Pak je trasa rozdělena na mnoho segmentů, u kterých jsou také vypočítány sekundární údaje, ale tentokrát vždy pouze pro daný segment. Segmenty už nejsou asociovány s konkrétní trasou nebo účtem a slouží pro navigaci – uživatel pozná, jaké už absolvované trasy (resp. segmenty) jsou v jeho okolí, jakou mají náročnost, a může podle toho naplánovat vhodnou trasu. Segment jakožto podmnožina trasy je volen proto, že trasa je příliš dlouhá a informace se vztahují k příliš velké vzdálenosti, než aby byly relevantní pro lokální navigaci (ztratí se např. extrémní hodnoty v množství hodnot standardních). Celá trasa je rozdělena do malých částí, kdy každá má individuální hodnoty a ty se neztratí zprůměrováním pro celou trasu.

Azure Queue

Azure Queue je fronta, do které mohou různí aktéři vkládat zprávy a jiní je zase vybírat a zpracovávat. V tomto případě zde Queue vystupuje jako mezičlánek mezi WCF Role a Worker Role. WCF Role do fronty vkládá zprávy (3) a Worker Role si je vybírá (4). Výhodou tohoto přístupu je snížení závislosti mezi WCF Role a Worker Role. WCF Role se nestará o to, kým a kdy bude požadavek zpracován, pouze jej vloží do fronty a zříká se jakékoliv další zodpovědnosti. Worker Role je zase odstíněna od toho, kdo požadavek zadal. Lze libovolně měnit, přidávat a ubírat aktéry na obou stranách fronty, aniž by to ovlivnilo aktéry na opačné straně. Lze například rozšířit rodinu klientů, kteří mohou vkládat trasy do fronty, aniž by to jakkoliv ovlivnilo Worker Role. Stejně tak lze libovolně přidávat další zpracovatele zpráv z fronty na straně druhé, aniž by byli ovlivněni aktéři vkládající do fronty. To je výhodné také z hlediska škálovatelnosti, kdy lze libovolně navyšovat počet Worker Rolí a/nebo jejich instancí, aby se zvýšila kapacita zpracování požadavků z fronty. Další výhodou je odolnost proti zátěži. V okamžiku, kdy se náhle zvýší počet požadavků přibývajících do fronty, nedojde k přetížení a pádu zpracovatelů, protože fronta funguje jako zásobník, odkud jsou požadavky vybírány podle potřeb zpracovatelů, a nedojde tedy k jejich zahlcení. Stejně tak, pokud by všichni zpracovatelé byli mimo provoz, fronta stále přijímá požadavky a ukládá je ke zpracování až po dobu jednoho týdne. Velikost fronty je přitom omezena pouze velikostí nadřazeného Storage Accountu (100 TB). Velikost jedné zprávy je maximálně 64 Kb. Z toho důvodu neukládám data ke zpracování do fronty, ale pouze informaci, co a kde se má zpracovat. Samotná data jsou pak uložena v Blob Storage – viz níže.

Blob Storage

Protože maximální velikost zprávy v Queue je 64 Kb, nelze do ní ukládat větší objem dat, jako je například trasa, která může obsahovat desetitisíce bodů. Trasy ke zpracování je tedy nutné fyzicky ukládat jinde. K tomu je zde použit Blob Storage. Přestože trasa obsahuje strukturovaná data a mohlo by se zdát, že z toho důvodu je vhodnější použít Table Storage, není tomu tak. Data trasy obdržená

od klienta jsou automaticky rekonstruována do podoby objektu – instance určité třídy. Worker role sama pracuje s objekty tohoto typu. Kdyby se použilo Table Storage, musela by se instance složitě konvertovat na záznam v tabulce a poté, když si ji bude chtít Worker Role vyzvednout (9), by bylo nutné ji opačným procesem rekonstruovat. Je mnohem jednodušší a rychlejší serializovat instanci pomocí standardních metod .NET Frameworku na pole bytů a uložit ji jako Blob do Blob Storage. Poté se dá jednoduše vyjmout (9) a opět snadno deserializovat pomocí standardních mechanismů.

Web Role

Web role zde vystupuje spíše jako sekundární komponenta. Slouží k zobrazování statického obsahu souvisejícího s aplikací - jako například prohlášení o ochraně osobních údajů, informace o programu nebo nápověda. Hlavním posláním však je zprostředkovat uživatelům Silverlight Clienta. Uživatel se připojí na stránku, kde je hostovaný tento klient (6) a ten je automaticky stažen na klientský stroj. V tomto okamžiku přestává být Web Role aktivní. Silverlight Client běží v prohlížeči na straně klienta a jakékoliv další požadavky již nejdou přes Web Roli, nýbrž WCF Roli (7).

Webový prohlížeč

Webový prohlížeč slouží pouze ke komunikaci s Web Rolí (6), a to k zobrazování statického obsahu v ní hostovaném, ale především ke stažení Silverlight Clienta. Ten pak běží na straně klienta v prohlížeči a sám pak komunikuje s WCF Rolí (7), stejně jako mobilní klient.

Silverlight Client

Silverlight Client je vlastně portem Mobilního klienta do tradičního Silverlightu. Neobsahuje sice všechny jeho funkce (viz Případy užití), ale zase poskytuje pohodlný přístup z webového prohlížeče. Silverlight Client komunikuje s WCF Rolí (7) stejně jako mobilní klient (1) a z hlediska WCF role jsou tito klienti totožní.

Windows Phone 7 Client

Ačkoliv serverová část může obsloužit libovolného klienta schopného komunikovat přes WCF, tento mobilní klient je brán jako primární a serverová část je optimalizována pro efektivní spolupráci právě s ním. Tento klient komunikuje s WCF rolí (1) a jejím prostřednictvím manipuluje s daty na serveru. Komunikace mezi mobilním klientem a WCF Rolí probíhá asynchronně. Klient tedy pošle požadavek a nepozastaví svoji činnost, dokud požadavek není vyřízen. Namísto toho může pokračovat v práci. V okamžiku, kdy dojde ke konečnému vyřízení požadavku, vznikne událost splnění požadavku. Klient na ni může nebo nemusí zareagovat (resp. klient se může nebo nemusí přihlásit k jejímu odběru).

Shrnutí

Při návrhu aplikací pro Windows Azure je třeba dbát na to, aby řešení poskytovalo co možná nejvyšší míru škálovatelnosti a odolnosti proti selhání a byly tak využity hlavní výhody Cloudu, tedy to, proč je vůbec aplikace nasazena na platformu Windows Azure.

Vhodné je oddělit front-end a back-end. Front-end bývá implementován jako Web Role, back-end jako Worker Role. Jako mezičlánek snižující závislost a zvyšující odolnost řešení je vhodné zvolit Azure Queue. Pro ukládání dat je vhodné zvolit škálovatelné cloudové řešení, a to v závislosti na povaze dat a operací nad nimi. V tomto případě byla zvolena varianta cloudové databáze SQL Azure. Klient nemá přímý přístup k datům ani k Worker Roli a celá jeho interakce se systémem je zprostředkovaná Web Rolí, která hostuje WCF Službu. Tím je zajištěna ochrana dat a odstranění závislosti mezi klientem a back-endem a systémem uložení dat.

4.4 Výběr vhodného perzistentního úložiště dat

Každá aplikace pracuje s dvěma typy dat - perzistentními a dočasnými. V této kapitole nejsou uvažována dočasná data (cache, fronty,…), ale jen ta perzistentní. Cílem kapitoly je popsat požadavky aplikace na vhodné úložiště perzistentních dat v cloudu, porovnat jednotlivá řešení a vybrat to nejvhodnější. Kapitola mj. slouží i jako ukázka, jak postupovat obecně při volbě vhodného cloudového úložiště perzistentních dat a jaké jsou výhody a nevýhody jednotlivých řešení.

4.4.1 Charakteristika dat a požadavky

Aplikace pracuje se strukturovanými perzistentními daty. Jedná se o informace o uživatelských účtech, přátelství, trasách, bodech zájmu a navigačních segmentech tras. Tato data mají následující charakteristiky:

- Data jsou strukturovaná.
- Data jsou perzistentní. Je třeba uchovat je potenciálně po neomezenou dobu.
- Většina dat se v průběhu svého životního cyklu mění jen zřídka (např. heslo uživatele, avatar) a velká část dat je statická (například údaje o trase jsou zaznamenány v době jejího vytvoření a pak se nemění).
- Data jsou tvořena až za běhu aplikace. Není třeba importovat existující data nebo používat existující datové zdroje.
- Není třeba zachovat určitý mechanismus práce s daty daný případnou existencí původního on-premise řešení.
- Nad daty by mělo být možné vykonávat dotazy, třídit, vyhledávat, řadit atd.

4.4.2 Základní kritéria výběru

Data strukturovaná X nestrukturovaná

Podle povahy dat je vhodné zvolit i povahu řešení. Strukturovaná a nestrukturovaná data mají svá specifika. U strukturovaných dat bude kladen důraz na vyhledávání, třídění a dotazovací mechanismy nad daty, u nestrukturovaných bude více záležet na efektivní práci s větším objemem dat. Windows Azure nabízí dva hlavní systémy pro práci se strukturovanými daty. Jedná se o Azure Table Storage a SQL Azure. Pro nestrukturovaná data jsou k dispozici Azure Blob Storage a Azure Drive.

Toto rozdělení není konečné a neznamená to, že pro strukturovaná data nelze použít nestrukturovaný mechanismus a obráceně. Azure Drive například může sloužit k uložení strukturovaných dat ve formě XML souborů a SQL azure může ukládat Blob data. Pokud tedy v požadavcích figurují další významné faktory, je možné zvolit pod jejich tlakem i nestandardní řešení, ale ve většině případů bude platit, že je výhodnější pro strukturovaná data použít Azure Table

Storage nebo SQL Azure a pro nestrukturovaná Azure Blob Storage nebo Azure Drive. Příkladem volby nestandardního řešení může být snaha o rychlou migraci existující aplikace a zachování jejího stávajícího mechanismu pro práci s perzistentními daty.

Migrace on-premise řešení X nová aplikace napsaná pro cloud

Windows Azure nabízí pro perzistentní data velmi výkonné a vysoce škálovatelné mechanismy jako je Azure Table Storage a Azure Blob Storage. Protože jsou tato řešení optimalizována pro cloud, jsou velmi rozdílná od on-premise řešení a přechod většinou znamená značný zásah do existující aplikace. Z toho důvodu nabízí Azure mechanismy, které jsou sice méně výkonné a škálovatelné v prostředí cloudu, ale zato se velmi podobají rozšířeným on-premise mechanismům a usnadňují tak migraci existujících aplikací. Při migraci je pak nutné se rozhodnout, zda se vyplatí:

- Vynaložit okamžitě vysoké úsilí a implementovat efektivnější verzi (Table Storage, Blob Storage) ihned a vzdát se stávajícího modelu.
- Vynaložit řádově menší úsilí a implementovat kompromisní řešení (Azure Drive, SQL Azure) k urychlení migrace a později přejít na plnohodnotné řešení (Table Storage, Blob Storage).
- Vynaložit řádově menší úsilí a implementovat kompromisní řešení (Azure Drive, SQL Azure) a později už nepřecházet na plnohodnotné řešení, protože užitek z vyššího výkonu a škálovatelnosti by v tomto případě nepřevýšil náklady na implementaci.
- Data ponechat mimo cloud a do cloudu přesunout jen služby, nikoliv jejich data. Datové zdroje by pak byly zpřístupněny pomocí Azure Connect / service Bus.

V okamžiku, kdy je vyvíjena aplikace pro Windows Azure zcela od základů, je ve většině případů vhodnější použít Azure Blob Storage a Azure Table Storage. Pokud chceme migrovat existující on-premise aplikace, je možnost využít kompromisního řešení (Azure Drive, SQL Azure) k usnadnění migrace nebo (pokud to nebude příliš nákladné a pracné) upravit aplikaci na nový systém. Vždy je pak třeba porovnat užitek na straně jedné a vynaložené náklady a čas na straně druhé.

4.4.3 Výběr vhodného řešení

Azure Drive, Azure Blob Storage

Vzhledem k tomu, že se jedná o nově vyvíjenou aplikaci bez předchozích existujících dat nebo datových zdrojů a bez již implementovaného mechanismu práce s nimi, můžeme rovnou vyloučit Azure Drive jako nevhodné řešení, o kterém bychom uvažovali pouze v případě migrace existující aplikace na platformu Windows Azure. Zbývá tedy uvažovat pouze o Azure Blob Storage.

Stejně jako Azure Drive, tak i Azure Blob Storage je mechanismem pro ukládání a práci s nestrukturovanými daty. Vzhledem k tomu, že z výše uvedených požadavků vyplývá, že se bude pracovat s daty strukturovanými, není volba nestrukturovaného úložiště vhodná. Jedinou možností by bylo strukturovat samotné Bloby, tedy například, aby každý Blob byl XML souborem, a implementovat vlastní mechanismus práce s takto strukturovanými daty. Tato možnost je však velmi těžkopádná. Vzhledem k povaze uvažovaných dat nemá Azure Blob Storage žádné výhody oproti cloudovým úložištím strukturovaných dat – SQL Azure a Azure Table Storage. Není tedy vhodné jej použít a nebude dále uvažován jako možná alternativa.

Azure Table Storage X SQL Azure

Na otázku, zda je pro strukturovaná data lepší Azure Table Storage nebo SQL Azure, není jednoznačná odpověď. Každé řešení je vhodné pro jiná data a pro jiné scénáře použití. V následujícím textu budou oba mechanismy porovnány z různých hledisek a poté bude vybrán vhodnější z nich pro tuto konkrétní aplikaci.

Škálovatelnost

Škálovatelnost můžeme rozdělit na dvě části. První je škálovatelnost co se týče úložné kapacity, druhá co se týče výkonu. Zjednodušeně řečeno, systém musí být schopen vyhovět změnám požadavků na úložnou kapacitu a stejně tak i změnám ve frekvenci přístupu k datům. Přitom při nárůstu množství uložených dat nesmí docházet k významnějším propadům výkonu.

Azure Table Storage má omezenou kapacitu Storage Accountem, pod který náleží. Ten má maximální velikost 100 TB [1]. Storage Account potenciálně sdílí data s Blob Storage a Queue Storage. Pokud budeme uvažovat situaci, kdy Storage Account taková data neobsahuje, bude maximální kapacita Table Storage celých 100 TB. Přitom Azure Subscription může vlastnit více Storage Accountů. Vzhledem k tomu, že mezi jednotlivými tabulkami nejsou žádné relace jako v relační databázi, může být v každém Storage Accountu pouze jedna tabulka. Teoreticky tak nabízí Table Storage neomezené místo s jediným omezením 100 TB na jednu tabulku. SQL této kapacitě může jen stěží konkurovat. Aktuální maximální velikost jedné databáze je 150 GB. Přitom tomu není tak dávno, co limit byl 50 GB nebo dokonce jen 10 GB.

Co se týče výkonu, může Table Storage velmi snadno škálovat. Služba poskytuje mechanismus pro partitioning – každý záznam má atribut PartitionKey a množina záznamů se stejnou hodnotou tohoto atributu tvoří stejnou partition. Je garantováno, že záznamy v jedné partition se budou fyzicky nacházet na stejném stroji. Takto lze tabulky velmi účinně „rozmělňovat“ a do mnoha partitions nebo naopak partitions slučovat. U SQL Azure je tato výkonová škálovatelnost výrazně slabší.

Co se týče škálovatelnosti, rozhodně vítězí Table Storage. Oproti prvním verzím se však škálovatelnost SQL Azure výrazně zlepšuje a lze přepokládat, že tomu tak bude i nadále, tedy alespoň dokud se rozvoj nepřiblíží hranicím samotné technologie relační databáze a její implementace v cloudu. Horkou novinkou v tomto směru je nedávné vydání SQL Azure Federations , které poskytují nativní podporu a nástroje „Sharding-as-a-Service“ a umožňují zvýšit škálovatelnost SQL Azure databáze pomocí shardingu, tedy rozložení větší databáze na množství menších, provozovaných na samostatných strojích [27].

Relace mezi daty a jejich integrita

Na rozdíl od SQL Azure nenabízí Table Storage mechanismy pro definování relace mezi daty a vynucení referenční integrity. Data jedné tabulky tak nemohou odkazovat na data tabulky jiné tak, jak to umožňují relační databáze.

U SQL Azure a relačních databází sdružuje tabulka entity stejného typu (např. kniha) a definuje atributy (např. autor, nakladatelství,…) a jejich datové typy pro všechny záznamy v tabulce. Podle definice tabulky pak předem víme, jaké atributy bude entita mít, jakých jsou datových typů, případně můžeme mít garantováno, že jsou nenulové atd. Garance vlastností a struktury dat poskytuje mnoho výhod při manipulaci s nimi. U Table Storage tato garance chybí. Atributy entit nejsou deklarovány na úrovni tabulky, ale na úrovni každé entity zvlášť. V jedné tabulce mohou být teoreticky entity mnoha druhů. Předem nevíme, jaké budou mít entity vlastnosti a jaké jsou jejich datové typy. Stejně pojmenovaný atribut může mít u každé entity jiný datový typ. Je na samotné aplikaci, aby si zajistila integritu a jednotný formát příbuzných dat. To nemusí být až takový problém, pokud k úložišti přistupuje pouze jeden klient. V opačném případě nemůže být garantováno, že ostatní klienti ukládají do úložiště stejný druh dat se stejnou strukturou.

Přenositelnost dat

SQL Azure umožňuje snáze importovat existující data, pokud jsou již uložena v existující relační databázi. Stejně tak, pokud je třeba exportovat data do formátu, který bude použitelný mimo prostředí Windows Azure, vítězí SQL Azure. Také klienti mimo Windows Azure mohou snadno přistupovat k SQL Azure a mohou s ním pracovat jako s tradiční relační databází bez jakýchkoliv nových znalostí. U Table Storage je třeba naučit se a použít poskytované REST API.

Zpracování dat

Oproti SQL Azure neobsahuje Table Storage mechanismy pro manipulaci s daty, které jsou typické pro relační databáze (JOIN, GROUP BY, ORDER BY) a tyto musí být implementovány manuálně. Jednoduché dotazy nad Table Storage lze provádět pomocí LINQ (Language Integrated Query) [28],

nelze však použít dotazy, které využívají relační povahy dat. SQL Azure také poskytuje složitější metody práce s daty a jejich zpracování (komplexní dotazy, uložené procedury, transakce,…).

SQL Azure je tedy vhodnější než Table Storage, pokud je třeba provádět složitější dotazy nad daty a dále je zpracovávat nebo analyzovat. [29]

Z hlediska zaměření této aplikace stojí také za zmínku, že SQL Azure nabízí podporu pro ukládání a manipulaci se Spatial datovými typy (geografické a geometrické). Tedy umožňuje ukládat body na zemském povrchu a jejich množiny, definovat trasy, určovat jejich vzdálenosti a průniky a mnohé další možnosti.

Náklady

U Azure Table Storage je účtováno využité místo a počet transakcí (čtení/zápis). Využité místo se počítá pomocí denních průměrů po dobu jednoho měsíce. Pokud tedy první polovinu bylo využito 10 GB a druhou polovinu 0 GB, bude účtováno 5 GB. Ceny jsou následující [30]:

- 1GB: $ 0,125
- 10 000 Transakcí: $ 0,01

U SQL Azure se účtuje pouze velikost databáze, a to stejným mechanismem jako u Azure Storage. Cena za GB se přitom liší dle velikosti databáze. Účtování přehledně ukazuje následující tabulka:

Tabulka 19 - Náklady na SQL Azure [30]

Velikost databáze	Cena za měsíc [$]
0 až 100 MB	Paušálně 4,995
Více než 100 MB, až do 1 GB	Paušálně 9,99
Více než 1 GB, až do 10 GB	9,99 za první GB; 3,996 za každý další
Více než 10 GB, až do 50 GB	45,954 za prvních 10 GB; 1,998 za každý další
Více než 50 GB, až do 150 GB	125,874 za prvních 50 GB; 0,999 za každý další

Přestože cena za 1GB s přibývající velikostí databáze klesá, oproti nákladům na Azure Table Storage je stále řádově větší (a to v řádu tisíců procent). Porovnání nákladů na 1GB u Azure Table Storage a SQL Azure v závislosti na celkovém využitém místu ukazuje *Tabulka 20 - Srovnání nákladů na SQL Azure a Azure Table Storage*. Rozdíly jsou opravdu výrazné. Při velmi malé databázi do 100 MB je SQL Azure dražší 409,2 krát. U 10 GB je to 36,8 krát a při maximálním využití všech 150 GB (maximální velikost databáze) je to už „jen“ dvanáctkrát více. Na rozdíl od SQL Azure má Table Storage zpoplatněn i počet transakcí. Cena je poměrně nízká, činí jeden dolar za milion transakcí. Tento faktor může v případech, kdy počítáme s velmi velkým počtem transakcí, hovořit ve prospěch SQL Azure. Lze snadno vypočítat, že při využité kapacitě 150 GB je SQL azure výhodnější, pokud proběhne za měsíc více než 207,75 milionů transakcí, tedy přibližně 80 transakcí každou vteřinu. To je však optimální případ při užití 150

GB. Pokud bude vyžita menší kapacita, budou výsledky ještě výrazněji svědčit ve prospěch Azure Table Storage.

Tabulka 20 - Srovnání nákladů na SQL Azure a Azure Table Storage

Využité místo	Cena za 1 GB [$]		
	SQL Azure	Azure Table Service	Rozdíl
100 MB	51,15	0,125	409,20x
1 GB	9,99	0,125	79,92x
10 GB	4,60	0,125	36,80x
50 GB	2,52	0,125	20,16x
150 GB	1,51	0,125	12,08x

Co se týče ceny, je nutné vzít v úvahu očekávanou využitou kapacitu a očekávanou frekvenci přístupu k datům. Při malých a středních velikostech databáze bude rozdíl mezi náklady na SQL Azure a Azure Table Storage v řádu tisíců procent. Pokud nebudeme potřebovat velké množství úložné kapacity a počet transakcí nebude závratně vysoký, budou rozdíly v nákladech opravdu značné. V takových případech se už vyplatí zvažovat, jestli tyto náklady nepřeváží zisk z použití SQL Azure. Nejedná se přitom o jednorázovou platbu, ale periodické výdaje. Často se pak vyplatí vynaložit jednorázové úsilí a přizpůsobit aplikaci pro Azure Table Storage i za cenu jistých ústupků. Obecně lze říci, že při návrhu cloudových aplikací oproti on-premise mají rozhodnutí učiněná při návrhu (co se týče zvolených mechanismů ukládání dat, cachování, řízení přístupu, CDN,…) obrovský dopad na náklady aplikace, protože téměř všechny nabízené mechanismy jsou samostatně zpoplatněny jako služby a k požadovanému cíli se dá dopracovat užitím různě efektivních mechanismů s různými náklady. Výkonné a návrhově čisté řešení může být nepoužitelné z hlediska příliš vysokých nákladů, které nepřevýší užitek z takového řešení. To samozřejmě platí i u on-premise aplikací, v cloudu je však dopad takovýchto rozhodnutí mnohem větší.

4.4.4 Vyhodnocení

Azure Table Storage je obecně vhodné použít, pokud:

- Je prioritou minimalizovat cenu
- Je prioritou maximalizovat škálovatelnost
- Je třeba ukládat velmi velké množství dat (více než 150 GB)
- Není potřeba provádět složitější zpracování dat a složitější dotazy nad nimi
- Není třeba reprezentovat relace mezi daty

SQL Azure je vhodné použít, pokud:

- Je prioritou snadná migrace existujícího řešení založeného na relační databázi
- Stačí omezenější škálovatelnost
- Množství uložených dat je menší než 150 GB
- Je třeba znázornit vztahy mezi daty
- Je třeba garantovat strukturu dat
- Je třeba provádět složitější zpracování a dotazy nad daty

Je nutné podotknout, že u větších aplikací nestojí otázka, zda využít SQL Azure **nebo** Table Storage, ale **jak** použít každý z těchto mechanismů. Obecně je totiž nejvýhodnější u velkých aplikací použít oba mechanismy, každý pro třídu dat, u které je to vhodné.

Zvážil jsem výhody a nevýhody obou mechanismů v kontextu povahy dat této aplikace a jejích požadavků a vybral jsem SQL Azure. Hlavní důvody výběru jsou:

- SQL Azure poskytuje nástroje pro ukládání a pokročilé zpracování geografických a geometrických dat.
- Nepředpokládá se, že by objem dat překročil 150 GB.
- Počítá se s importem geografických dat od GINA Software uložených v relační databázi.
- Možnost definovat vztahy mezi daty bez implementace vlastních externích mechanismů.

4.5 Návrh asistence při plánování trasy

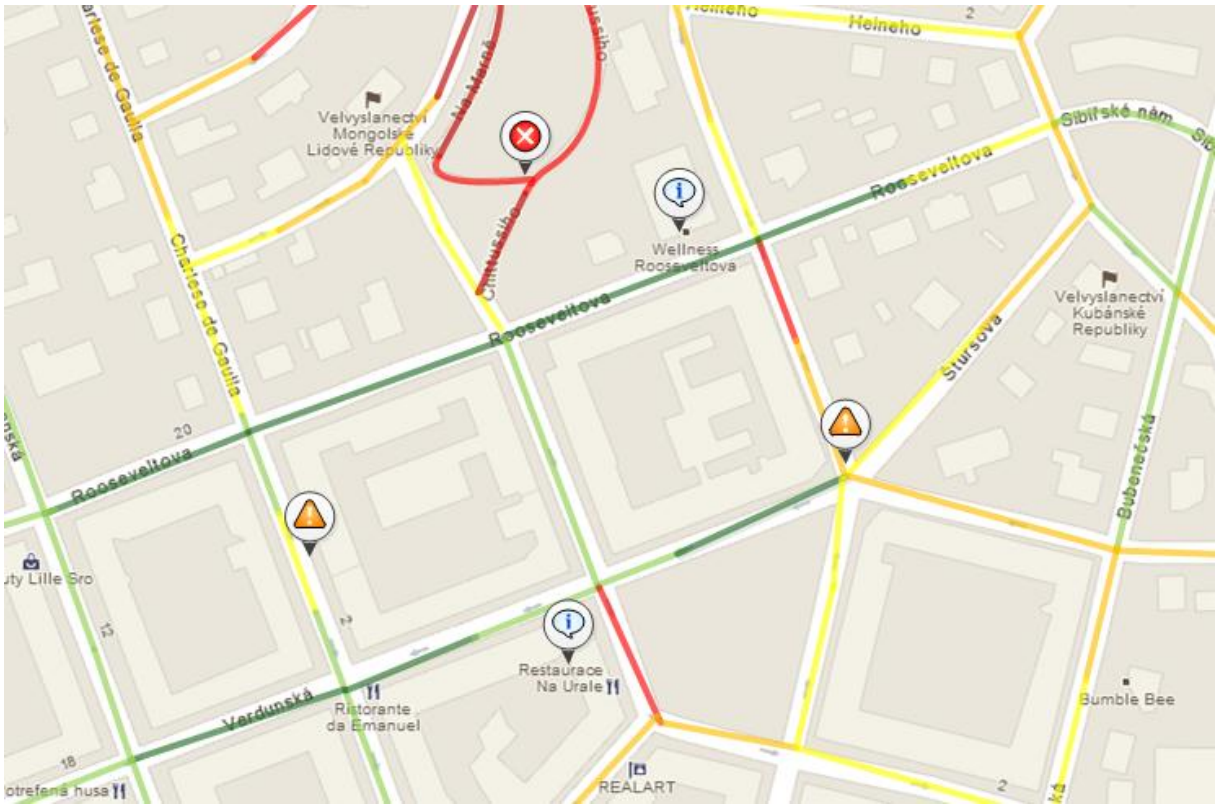
Jednou z hlavních funkcí aplikace je pomoc uživateli při navigaci nepřehledným a neznámým terénem. Nejedná se však o automatické naplánování konkrétní trasy z bodu A do bodu B. Takové řešení by vyžadovalo obrovské množství dat o průjezdnosti a obtížnosti terénu a jejich sběr by trval nepřijatelně dlouho. Z algoritmického hlediska je takové řešení nad rámec této práce a mimo oblast jejího zaměření. Aplikaci namísto plánování konkrétní trasy poskytne uživateli všechna dostupná navigační data pro jeho okolí. Tím je uživateli umožněno učinit efektivnější a racionálnější rozhodnutí o své trase. K tomu je použito dvou základních navigačních prvků – bodů zájmu (POI) a segmentů trasy.

POI označuje konkrétní bod na mapě. Má uložen název, popis a kategorii. Kategorie označuje, zda se jedná o doporučení, varování nebo například nebezpečí. Body zájmu mohou vytvářet uživatelé dle libosti a označovat tak místa, která stojí za pozornost. Uživatel pak s ohledem na blízké POI může pozměnit plán své trasy, aby se například vyhnul nesjízdným schodům nebo se po cestě občerstvil v restauraci s bezbariérovým přístupem nebo stojanem na jízdní kola.

Segmenty tras jsou malé úseky, které vznikají rozdělením skutečných tras, které uživatelé uloží do systému. Trasa sama o sobě je příliš dlouhá, než aby měla vypovídací hodnotu a údaje o ní byly přínosné. Pokud se však rozdělí na malé celky, každý takový segment pak má velkou vypovídací hodnotu, protože se vztahuje k malé oblasti. Každý segment je pak ohodnocen obtížností, která věrně odráží obtížnost pohybu po daném úseku. Obtížnost je přitom tvořena dvěma složkami. První určuje uživatel pro celou trasu a segmenty ji poté přebírají. Výhodou uživatelského hodnocení je, že dokáže zohlednit i parametry, které z GPS dat nelze vyčíst – například obtížnost povrchu. Druhá složka je obtížnost terénu. Ta je vypočítávána automaticky. Zde hraje roli především celkové převýšení vzhledem k délce segmentu a potom největší lokální převýšení (tedy je například zohledněn jeden krátký úsek velmi náročného stoupání na jinak rovinatém terénu). Segmenty jsou odlišeny barvou podle celkové obtížnosti (1-10) - postupně od odstínů zelené, přes oranžové a nakonec k červené.

Když si uživatel vyžádá navigační údaje, zobrazí se mu dostupné segmenty tras a POI v určitém okruhu od jeho polohy, který závisí na stupni přiblížení mapy. Na základě nich pak může sám učinit rozhodnutí o plánované trase.

Modelové zobrazení navigačních údajů ukazuje *Obrázek 8 - Návrh asistence při plánování trasy [autor].* Jedná se o ideální případ, kdy je nasbírán velký počet navigačních dat. Obtížnosti v jednotlivých úsecích a body zájmu na obrázku jsou pouze ilustrační a neodrážejí skutečnost.



Obrázek 8 - Návrh asistence při plánování trasy [autor]

5 Implementace

5.1 Rozdělení práce

Aplikace je rozdělena na klientskou a serverovou část, kdy každá z těchto částí je řešena samostatnou diplomovou prací. Serverovou část jsem zpracovával já, klientskou pak Bc. Vít Čurda. Nezanedbatelná část práce musela být vykonána společně (např. definice komunikačního rozhraní aj.), a to nejen na začátku, ale i v průběhu vývoje aplikace. Cílem této kapitoly je poskytnout přehled o tom, kdy bylo přikročeno ke spolupráci a kdy naopak práce probíhala zcela individuálně.

I. Fáze příprav

Tato fáze spočívala především v jednání s jednotlivými partnery projektu (Microsoft a GINA Software). Definovaly se hrubé obrysy projektu, způsob zapojení partnerů, globální požadavky a cíle atd. Všech těchto jednání jsem se účastnil i já i Bc. Vít Čurda, protože jednání probíhala na globální úrovni, kdy výsledky jednání ovlivňovaly aplikaci jako celek – tedy klientskou i serverovou část.

II. Analýza

Tato fáze vychází z výstupů fáze první. Byly definovány konkrétní funkce aplikace, uživatelské scénáře, způsob interakce serverové a klientské části. Opět se jedná o témata přesahující rozdělení na klient a server a bylo nutné je definovat a shodnout se na nich společně.

III. Návrh a implementace klientské a serverové části

Ve fázi, kdy se definovaly požadované vlastnosti aplikace jako celku a komunikační rozhraní, návrh a implementace klientské/serverové části již probíhal zcela samostatně. V průběhu vývoje se však stávalo, že se objevily nové požadavky na funkčnost a bylo třeba společně definovat rozšíření rozhraní, aby reflektovalo tyto změny.

IV. Testování

V průběhu testovací fáze se často ukázalo, že se aplikace chová nepředpokládaným způsobem nebo dochází k jejímu pádu. Ne vždy bylo jednoduché ihned zjistit, zda je příčina na straně serveru nebo klienta. V tom případě bylo nutné společně analyzovat situaci a najít příčinu. Opravu problému pak provedl už ten, v jehož části se problém nacházel.

5.2 Spolupráce

5.2.1 Metodika

Cílem kapitoly je popsat z metodologického hlediska vývoj aplikace a spolupráci. Ačkoliv se může zdát, že pro dvoučlenný tým není třeba dodržovat určitou metodiku, rozhodli jsme se přesto tak učinit a výsledky byly uspokojivé. Ze spolupráce bylo poté vyvozeno určité zobecnění. Tyto informace mohou být prospěšné i ostatním malým týmům.

Standardní rigidní metodiky se nezdají být pro tento typ práce vhodné. A to především s ohledem na svou velkou náročnost na provoz a malou flexibilitu. Pro velmi malý tým a menší projekty je nelze doporučit. Je třeba tedy zvolit agilní metodiku, která se dokáže lépe vypořádat s neustále se měnícími požadavky a distribuovanými týmy. Jejich provoz je také řádově méně administrativně náročný. Po úvaze jsme se rozhodli nevolit jednu konkrétní metodiku jako celek, ale sestavit vlastní jednoduchou metodiku s některými užitečnými prvky z existujících agilních metodik.

Mezi požadavky na metodiku patřila hlavně schopnost vypořádat se s neustále se měnícím zadáním. Dále pak aby mechanismy metodiky pokud možno zvyšovaly motivaci členů týmu k práci a snižovaly tendence k odkládání práce. Vynaložené úsilí by mělo být pokud možno rozděleno rovnoměrně po celou dobu práce na aplikaci a nikoliv být po většinu času minimální a až ke konci mnohonásobně zvýšené. Neméně důležitým požadavkem pak je minimalizace administrativy a úsilí věnovaného na provozování samotné metodiky. Vzhledem k tomu, že plánovaná doba vývoje byla minimálně jeden rok a předpokládalo se, že obě strany nebudou pracovat ve stejnou chvíli, je nutné, aby měl každý přehled o tom, v jakém stavu se nyní vývoj nachází a na čem konkrétně pracuje druhá strana. A když se po období nečinnosti znovu začne pracovat na projektu, bude zcela jasné, v jakém stavu se projekt právě nachází a jaké jsou další vyžadované akce.

Po vzoru metodiky SCRUM jsme stanovili, že vývoj bude probíhat v iteracích. Iterace je v tomto smyslu časový úsek 14 dní, který má stanoven určité cíle, které by se měly do jeho ukončení naplnit. Na konci iterace se vyhodnotí její průběh a na základě výsledků jsou přijata příslušná opatření do budoucna. Nesplněné úkoly jsou buď uzavřeny jako neaktuální nebo přesunuty do další iterace. Neustálé přenášení nesplněných úkolů do další iterace by mělo motivovat k jejich rychlému dokončení. Při vyhodnocení iterace se také stanoví cíle a plány pro iteraci následující. Důležitým pravidlem je, že iterace probíhají každých 14 dní nehledě na to, zda je práce právě pozastavena atd. V okamžiku, kdy by se iterační proces pozastavil, vznikaly by tendence k odkládání nového začátku práce na projektu. Z metodiky SCRUM jsme převzali také další prvek „daily stand-up“, kdy každý den na meetingu členové týmu ostatní informují o tom, co vypracovali od posledního meetingu, na čem budou pracovat nyní a jaké mají v této práci překážky. Tento princip jsme modifikovali pro

distribuované týmy. Report se nepřednášel na každodenním meetingu, ale tvořil se písemně vždy po dokončení určitého celku (spíše menšího). Po vzoru SCRUM jsme se také snažili vytvořit co nejrychleji fungující kostru, která bude moci být předvedena, a v dalších iteracích na ni nabalovat další funkce. Tím je umožněno mít neustále fungující produkt, který se kontinuálně zlepšuje. Výhodou je, že jsme mohli poměrně brzy vydat fungující verzi na Windows Marketplace nebo ji předvést zúčastněným stranám a mohli jsme obdržet cennou zpětnou vazbu v raných fázích vývoje – případné zásahy do aplikace pak jsou rychlejší a méně náročné než v pozdějších fázích.

Další princip byl zařazen po vzoru Lean Development – „Decide as late as possible“. Vzhledem k tomu, že neustále probíhala jednání s partnery projektu, objevovalo se mnoho nových požadavků a ty existující byly měněny. V této atmosféře nejistoty je výhodné odkládat rozhodnutí co nejdéle, protože je pravděpodobné, že rozhodnutí na dané téma budeme muset brzy učinit znovu na základě změny požadavků. Vždy, když bylo třeba vybrat si na konci iterace, na čem bude probíhat další práce, volila se oblast, u které již byla rozhodnutí učiněna a která je co nejméně závislá na ostatních částech a co nejvíce odolná oproti změně.

5.2.2 Nástroje

Assembla

Assembla je služba poskytující nástroje agilním distribuovaným týmům pro vývoj software (www.assembla.com). Služba nabízí mnoho volitelných modulů, které lze použít. Nejdůležitější pro tuto aplikaci byly především následující.

SVN

Integrované úložiště pro SubVersion zajišťující version control zdrojového kódu se všemi možnostmi SubVersion 1.6. Výhodou je např. možnost kdykoliv se vrátit k předchozí revizi, vytvářet samostatné větve vývoje atd. Tímto způsobem je centrálně zálohován všechen zdrojový kód včetně jeho změn v čase, což je velmi užitečné.

Scrum Stand-Up Tool

Tento nástroj umožňuje rychle a jednoduše vytvořit report o tom, na čem daný člen týmu dosud pracoval, na čem bude pracovat nyní a jaké má v tomto směru překážky. Ostatní členové týmu jsou automaticky notifikováni.

Tickets

Ticket slouží jako určitý požadavek na akci. Jedná se především o nahlašování bugů a požadování určité funkčnosti. Každý Ticket pak má svoji prioritu, zodpovědného člena týmu, termín splnění a další vlastnosti. Tickety pak lze dle libosti třídit, filtrovat, upravovat atd.

Schránka s Tickety pak slouží jako seznam akcí, které si vyžadují pozornost. Tickety jsme používali především k hlášení chyb a poukazování na nutné zásahy do aplikace nebo požadovanou funkčnost od druhé strany.

Milestones

Nástroj umožňující sdružovat Tickety a úkoly do jednotlivých milníků. Každý milník pak může mít určité datum splnění. Pro naše potřeby byly milníky použity k reprezentaci jednotlivých iterací a cílů/funkcí, které do nich patří.

Dropbox

Dropbox (www.dropbox.com) je služba pro sdílení a synchronizaci souborů zapojených počítačů skrze cloud. Dropbox byl použit k vzájemnému sdílení souborů. Jednalo se o soubory, které nejsou přímo součástí jednotlivých aplikací. Tyto byly sdíleny a uchovávány pomocí SVN repozitáře. Zbytek byl sdílen právě prostřednictvím Dropboxu. Jednalo se například o různé elektronické publikace na zpracovávané téma.

Skype

Skype byl hlavním komunikačním prostředkem a byl velmi často a velmi úspěšně využíván. Pro krátká sdělení byly využívány textové zprávy. Častěji pak audio hovory. Velmi užitečnou funkcí bylo sdílení obrazu plochy, například při testování.

OneNote

Pro sdílení textových dat jsme nepoužívali textové dokumenty (např. často používané Google docs), ale Microsoft OneNote. Kromě snadného sdílení a synchronizace byla velkou výhodou především možnost hierarchicky strukturovat informace. V takto uspořádaných informacích se velmi snadno vyhledává a přehledněji se s nimi pracuje. Oproti sekvenčnímu uložení v klasickém dokumentu je pak práce rychlejší a efektivnější. OneNote byl používán především v prvních fázích projektu (jednání s partnery, analýza a návrh), a to pro administrativu. Tedy zaznamenávání úkolů, klíčových informací, podnětů, nápadů, návrhů atd. Ve fázi implementace mnohé z těchto oblastí nahradily nástroje Assembly.

5.2.3 Zhodnocení

Dle mého názoru se použitá metodika osvědčila. Přestože v takto malém týmu a rozsahu práce by nebylo problémem žádnou metodiku nepoužít (a výsledek by se pravděpodobně příliš nelišil), jsem přesvědčen, že její použití bylo prospěšné. Jako velkou přednost spatřuji, že se metodika výborně hodí pro distribuované týmy a udržuje dobrý přehled o tom, na čem pracuje zbytek týmu. Také je velmi flexibilní a odolná proti změnám požadavků. Pro mou osobní potřebu však byla nejpřínosnější asi motivační síla metodiky. Automatické reporty o práci ostatních členů týmu jedince motivují k činnosti, neboť ví, že ostatní nezahálejí, a sám nechce být spatřován jako slabý článek. Iterace pak zajišťují pocit provinění, když se přesouvají nehotové úkoly do další iterace a nepozastavitelnost cyklu zajišťuje rovnoměrnější rozdělení práce.

Použité nástroje musím všechny hodnotit velmi pozitivně. Hlavně Assembla je velmi efektivní řešení pro agilní vývoj v distribuovaných týmech s celou řadou funkcí.

Celkově lze metodiku doporučit malým a středním týmům. V současné podobě je vhodná především pro distribuované týmy. Je vhodná v prostředí náchylnému ke změně požadavků nebo v takovém, kde se požadavky teprve formují v průběhu projektu, a mezitím je třeba již vykonávat práci. Metodika není vhodná pro větší týmy a příliš komplexní projekty.

5.3 Specifika a problémy vývoje pro Windows Azure

5.3.1 Zabezpečení komunikace s klientem

Volba způsobu zabezpečení

Kvalitní zabezpečení komunikace mezi klientem a serverem je důležité pro naprostou většinu aplikací. Přesto, že v aktuálním provedení není pro naši aplikaci takovéto zabezpečení nezbytné, rozhodli jsme se ho implementovat proto, abychom demonstrovali tuto možnost ostatním vývojářům a ulehčili jim práci, protože velká část z nich bude tento problém ve své aplikaci řešit. Úspěšná implementace trvala nakonec velmi dlouho, protože jsme narazili na řadu problémů a nedostatek zdrojů. Vždy se jednalo pouze o standardní WCF, nikoliv o variantu nasazenou ve Windows Azure. Navíc se ukázalo, že klient WP7 má oproti standardním klientům svá specifika. Proto jsem přesvědčen, že tato část ulehčí ostatním vývojářům řešení podobných problémů a hlavně mnoho času a je tedy velmi přínosná. Ve finální verzi aplikace nakonec šifrovaná komunikace není implementována. Důvodem byla potřeba ruční instalace certifikátů na klientovi a použití vlastního certifikátu namísto koupeného od certifikační autority. V zájmu zvýšení šance na získání certifikace pro Marketplace jsme tedy odstranili zabezpečenou komunikaci. V budoucnu však není problém podle níže uvedeného postupu zabezpečení opět nastavit.

Použití WCF pro komunikaci mezi klientem a serverem poskytuje možnost využít bezpečnostních mechanismů, které WCF nabízí. Pro zabezpečení komunikace můžeme použít jeden ze tří režimů služby [31]:

- **Transport Security** – Zabezpečení na úrovni přenosu, tedy například SSL pro HTTPS protokol. Veškerá komunikace probíhající kanálem musí být šifrovaná. Výhodou je větší výkon oproti Message Security.
- **Message Security** – Zabezpečení na úrovni posílané zprávy. Tělo zpráv je šifrováno. Toto řešení je nezávislé na transportní vrstvě a velmi flexibilní. Každá zpráva může obsahovat libovolné množství credentials pro autentifikaci odesílatele.
- **Transport With Message Credentials** – Kombinuje obě výše uvedené možnosti. Zabezpečený přenos a jeho vysoký výkon zajišťuje transportní vrstva a každá zpráva může být stále vybavena množstvím credentials.

Pro potřeby aplikace byla vybrána možnost Transport With Message Credentials kvůli vysokému výkonu, bezpečnosti a možnosti autentifikace. Při volbě této možnosti je nutné dále se rozhodnout, jaký typ autentifikace (credentials) bude použit. Na výběr je několik možností jako uživatelské jméno a heslo, certifikát, Windows autentifikace nebo bezpečnostní token. Byla vybrána varianta uživatelského jména a hesla, protože se jedná o jednoduchý a běžně užívaný způsob v podobném typu aplikací. Jako transportní mechanismus byl vybrán HTTPS protokol.

Tvorba certifikátů

K zajištění šifrované komunikace je třeba mít bezpečnostní certifikát. U komerčních aplikací by při nasazení aplikace byl použit certifikát vydaný ověřenou certifikační autoritou. Ve fázi vývoje je však vhodné použít vlastnoručně vydané testovací certifikáty. K jejich tvorbě se hodí například nástroj *makecert*, který je součástí instalace Visual Studio 2010. Postup tvorby je následující:

1. Vytvořit kořenový certifikát, který bude sloužit jako certifikát certifikační autority. Jedná se o certifikát podepsaný sám sebou (self-signed).

   ```
   makecert.exe -r -pe -n "CN=Diplomka Root CA,O=Diplomka,C=US" -sky
   signature -a sha1 -cy authority -sv DiplomkaRootCA.pvk
   DiplomkaRootCA.cer
   ```

2. Vytvořit certifikát pomocí kořenového certifikátu pro šifrování komunikace.

   ```
   makecert -pe -n "CN=diplomka.cloudapp.net,O=Diplomka,C=US" -a sha1 -
   sky exchange -eku 1.3.6.1.5.5.7.3.1 -ic DiplomkaRootCA.cer -iv
   DiplomkaRootCA.pvk -sp "Microsoft RSA SChannel Cryptographic
   Provider" -sy 12 -sv DiplomkaSSL.pvk DiplomkaSSL.cer
   ```

3. Zaheslovat certifikát pro šifrovanou komunikaci.

   ```
   pvk2pfx -pvk DiplomkaSSL.pvk -spc DiplomkaSSL.cer –pfx
   DiplomkaSSL.pfx -po password
   ```

Konfigurace služby

Nyní je třeba nasadit certifikáty na stroj, na kterém probíhá vývoj. Zde je třeba vědět, že standardní import certifikátu systémem Windows uloží certifikát do úložiště pro aktuálního uživatele, ale Visual Studio pracuje s úložišti pro aktuální počítač. Je tedy nutné certifikát přesunout do tohoto úložiště takto:

1. Spustit utilitu Microsoft Management Console - standardně C:\Windows\system32\mmc.exe nebo vyhledat „mmc“ v nabídce start.
2. Otevřít okno pro přidání modulů snap-in – CRL+M.
    a. Přidat modul snap-in pro aktuálního uživatele – Přidat modul Certifikáty z levého sloupce. V okně, které se otevře, vybrat možnost „Tento modul bude spravovat certifikáty pro: Můj uživatelský účet“.
    b. Přidat modul snap-in pro počítač – Přidat modul Certifikáty z levého sloupce. V okně, které se otevře, vybrat možnost „Tento modul bude spravovat certifikáty pro: Účet počítače“.
    c. Potvrdit a zavřít okno pro přidávání modulů snap-in (OK).
3. Pomocí metody drag-and-drop přetáhnout požadované certifikáty:
    a. Ze sekce Certifikáty - aktuální uživatel->Osobní->Certifikáty.
    b. Do sekce Certifikáty (místní)->Osobní->Certifikáty.

Po úspěšné instalaci je třeba ve Visual Studiu[13] u Azure projektu ve složce Roles označit konkrétní roli, která hostuje WCF službu. Z kontextového menu této položky je pak nutno vybrat Properties a nastavit:

1. V sekci Certificates – Přidat nový certifikát. Zachovat nastavení StoreLocation: Local Machine a StoreName: My. V sekci Thumbprint pak vybrat ze seznamu příslušný certifikát a Thumbprint se vygeneruje automaticky. Tyto akce provést pro oba vygenerované certifikáty.
2. V sekci Endpoints – U stávajícího endpointu změnit protokol na HTTPS a nastavit port (doporučeno 443). V sloupci SSL Certificate Name nastavit druhý vygenerovaný certifikát (pro šifrování komunikace).

Visual Studio automaticky upraví soubory ServiceConfiguration a ServiceDefinition. Pokud nepoužíváte Visual Studio, lze tyto XML soubory upravit ručně dle vzoru příslušných souborů v příloze práce (v sekci Zabezpečení komunikace).

Dále je třeba ručně upravit XML soubor Web.config v projektu, ve kterém je hostována WCF služba.

[13] Ověřeno pro Microsoft Visual Studio 2010 Ultimate SP1 a Azure SDK 1.6 (vývoj byl zahájen na starších verzích a v jeho průběhu byly vydávány aktualizace SDK; aplikace byla vždy po tomto vydání transformována na aktuální verzi). V novějších verzích Visual Studia a Azure SDK se může postup lišit.

```
<system.serviceModel>
   <services>
      <service name="MainWFCRole.MainService" behaviorConfiguration="sb1">
        <endpoint binding="basicHttpBinding" contract="MainWFCRole.IGEOService"
        bindingConfiguration="binding1"/>
        <endpoint binding="basicHttpBinding" contract="MainWFCRole.IAccountService"
        bindingConfiguration="binding1"/>
      </service>
    </services>
    <bindings>
      <basicHttpBinding>
        <binding name="binding1">
          <security mode="TransportWithMessageCredential">
            <message clientCredentialType="UserName"/>
          </security>
        </binding>
     </basicHttpBinding>
    </bindings>
</system.serviceModel>
```

V tagu service jsou definovány jednotlivé endpointy služby co se týče typu spojení a služby, která je na nich poskytována. BasicHttpBinding označuje typ komunikace a contract označuje rozhraní (konkrétní interface), které definuje metody (operation contract) a datové typy (data contract) služby. Behavior configuration je odkaz na konfiguraci chování, ta je definována v sekci *Validace uživatelského jména a hesla* níže. Binding configuration je odkaz na konfiguraci, která je definována níže v sekci bindings. Zde konkrétně tato konfigurace definuje mód zabezpečení (mode="TransportWithMessageCredential", viz vvýše) a metodu autentifikace odesílatele zprávy (pomocí jména a hesla - UserName).

Validace uživatelského jména a hesla

Pokud je jako metoda autentifikace použito uživatelské jméno a heslo, je nutné implementovat vlastní metodu ověření. Je nutné vytvořit třídu, která dědí od předka Systém.IdentityModel.Selectors.UserNamePasswordValidator a přepisuje jeho metodu public void validate(string userName, string password). V těle této metody probíhá samotná validace. Pokud metoda bude úspěšně vykonána, validace proběhla úspěšně. Pokud nastala v průběhu vykonávání těla metody výjimka, validace se považuje za neúspěšnou.

Dále je nutné upravit soubor Web.config WCF služby. Do části system.servicemodel->behaviors->serviceBehaviors přidat následující tag.

```
<behavior name="sb1">
       <serviceCredentials>
              <userNameAuthentication userNamePasswordValidationMode="Custom"
              customUserNamePasswordValidatorType="MainWFCRole.AccountValidator,MainWFC
              Role"/>
       </serviceCredentials>
       <serviceMetadata httpsGetEnabled="true"/>
</behavior>
```

MainWCFRole je jméno služby a AccountValidator jméno výše zmíněné třídy pro validaci jména a hesla. Pomocí jména *sb1* je tento behavior referencován z konfigurace v předešlé sekci (*Konfigurace služby*).

Příklad implementace třídy pro validaci uživatelského jména a hesla pomocí výše uvedeného mechanismu je uveden v příloze jako AccountValidator.cs.

Nasazení certifikátů na klienta a na server

Vygenerované certifikáty nestačí zohlednit v Azure projektu ve Visual Studiu, je třeba je také přímo nasadit pomocí Management Portálu Windows Azure:

1. Vyberte sekci Hosted Services, Storage Accounts & CDN.
2. Z menu v levé horní části obrazovky vyberte položku Hosted Services.
3. V hlavní části okna je seznam nasazených služeb, jejich rolí a jejich instancí. Každá služba má na kořenové úrovni složku Certificates. Po vybrání se v horní části obrazovky objeví velká ikona Add Certificate.
4. Po kliknutí na Add Certificate se objeví okno pro vybrání souboru certifikátu. Dále je nutné zadat heslo, kterým byl certifikát zaheslován. Poté je možné certifikát uploadovat.
5. Tyto akce je nutné zopakovat pro oba certifikáty.

U standardních klientů (např. WPF aplikace nebo standardní Silverlight klienti), kteří konzumují WCF službu, se nemusí certifikáty ručně nasazovat. Klient si je vyžádá od serveru sám a ten mu je poskytne. V případě klienta na Windows Phone 7 tomu tak není. Certifikáty je nutné předem nasadit na klienta. Ve verzi aplikace, která používala zabezpečenou komunikaci, bylo tedy nutné při prvním spuštění aktivovat průvodce, který poradil uživateli, jak a odkud stáhnout certifikáty a nasadit je. Teprve poté mohl klient začít komunikovat se serverem.

5.3.2 Změna výchozích limitů pro WCF zprávy

Při standardním užití aplikace při testování nebyly patrné žádné chyby. V okamžiku, kdy se však zaznamenala delší trasa než je obvyklé, nastaly problémy. Server začal odmítat zprávy. Jako důvod oznamoval limit maximální možné velikosti zprávy nebo maximální možný počet prvků pole. Větší zprávy tedy nešlo posílat, a to ani na server ani na klienta.

Každá WCF služba má nastaveny určité limity, které omezují přenášené zprávy. V konfiguračním souboru ale nejsou tyto limity zaznamenány. Je nutné ručně přidat atributy, které definují hodnoty těchto limitů, a nastavit konkrétní hodnoty. Pokud tak není učiněno, nabývají limity výchozích hodnot. Při nastavování je nutné znát tyto výchozí hodnoty, aby bylo možné nastavit číslo vyšší, ale ne zbytečně vysoké vzhledem k výchozí hodnotě. Jednotlivé limity, jejich význam a výchozí hodnoty jsou popsány v tabulce *Tabulka 21 - Limity zpráv WCF*. U některých parametrů je zmíněn jazyk XML. Je tomu tak proto, že ve výchozím nastavení jsou zprávy ve WCF posílány přes protokol SOAP právě v jazyce XML. Binární data jsou kódována v base64 (textová reprezentace binárních dat).

Tabulka 21 - Limity zpráv WCF [32]

Parametr	Význam	Výchozí hodnota
maxReceivedMessageSize	Kladné celé číslo určující maximální velikost zprávy v bytech, a to včetně hlaviček.	65536 (64 Kb)
readerQuotas/maxDepth	Kladné celé číslo, které určuje maximální počet úrovní pro vnořené XML elementy při čtení.	32
readerQuotas/maxStringContentLength	Kladné celé číslo, které určuje maximální počet znaků hodnoty každého XML elementu.	8192
readerQuotas/maxArrayLength	Kladné celé číslo, které určuje maximální délku pole bytů.	16384

V konfiguračním souboru Web.config je třeba přidat do části system.serviceModel->bindings nový binding (v aplikaci je používán basicHttpBinding) podle požadovaného typu přenosu. Umístění jednotlivých atributů a strukturu zápisu ukazuje následující příklad. Nastavení tímto způsoben je třeba provést na straně klienta i serveru.

```
<system.serviceModel>
  <bindings>
    <basicHttpBinding>
       <binding maxReceivedMessageSize="65536">
            <readerQuotas maxStringContentLength="8192" maxArrayLength="16384"
            maxDepth="32"/>
       </binding>
    </basicHttpBinding>
  </bindings>
</system.serviceModel>
```

Limity lze sice ručně měnit, ale přístup spočívající v doplnění několika nul za výchozí hodnoty rozhodně nelze doporučit. Přílišné navýšení hodnot má negativní vliv na škálovatelnost služby a na její bezpečnost. Taková služba je pak velmi snadnou obětí útoků typu Distributed Denial of Service, protože umožní klientům posílat velmi objemné zprávy, které se mnohem déle zpracovávají a více zahlcují spojení [32]. Navýšení limitů je tedy možné, ale pokud se má jednat o významnější zvýšení hodnot, je třeba jednat s rozmyslem. V takové situaci je nutné zvážit, zda zvýšené riziko stojí za užitek plynoucí z možnosti posílat objemnější zprávy. Pokud je posílání objemných zpráv nevyhnutelné a navýšení limitů by bylo příliš vysoké, je vhodné změnit strategii přenosu nebo dokonce architekturu aplikace. U menších a středních objemů dat se může například provádět segmentace obsahu do několika menších zpráv. U objemného obsahu (zvláště např. multimédií) je pak možné opustit model WCF služby jakožto jediného vstupního kanálu serverové části aplikace. Lze například zpřístupnit jeden veřejný kontejner Blob Storage a nechat klienta, ať obsah uploaduje pomocí datového proudu na server do tohoto kontejneru. Pomocí WCF služby pak klient informuje server, že uploadoval obsah, jaký je jeho typ, kde je možné ho nalézt a jakým způsobem má být zpracován. Server pak požadavek zařadí do fronty. Po úspěšném zpracování jsou pak data z Blob Storage odstraněna.

5.3.3 Diagnostika

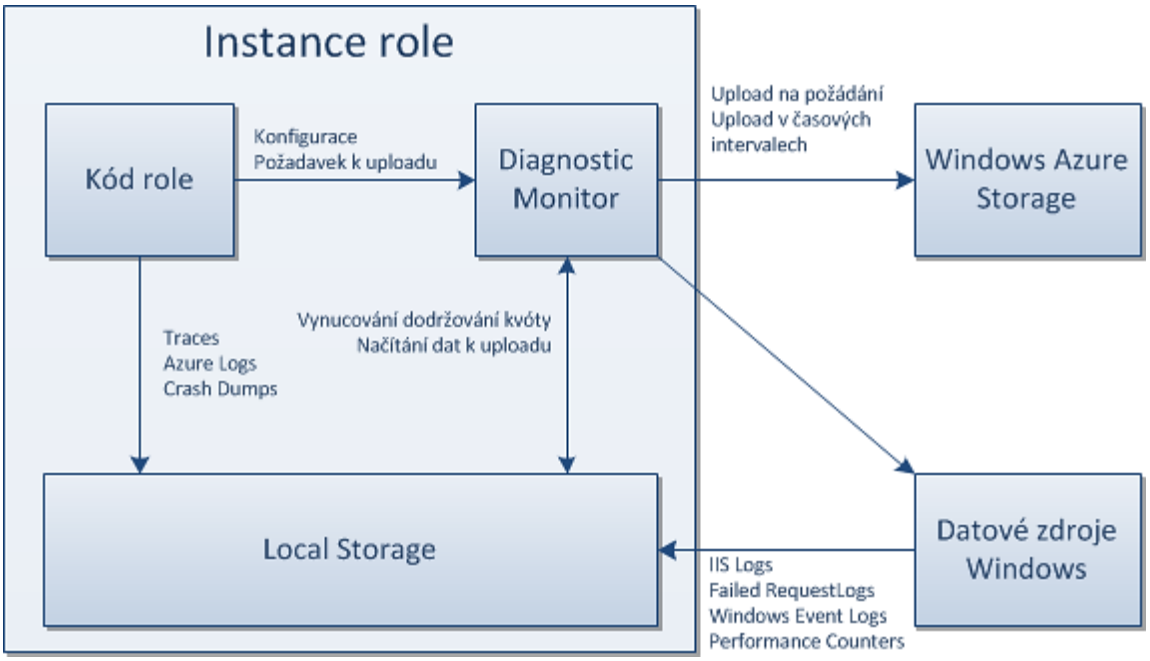
Windows Azure Diagnostics

Diagnostika aplikace je velmi důležitou oblastí. Umožňuje například včas zjišťovat problémy, ladit aplikace, měřit výkon a spotřebu zdrojů. Oproti on-premise řešení přináší diagnostika v cloudu mnohá úskalí. Nelze diagnostikovat jeden server, ke kterému je snadný přístup, ale aplikace se skládá z mnoha rolí a každá z nich je nasazena v mnoha instancích na mnoha fyzických strojcích. Je třeba mít k dispozici mechanismus, který dokáže pracovat v takovémto prostředí, a zároveň dokáže poskytnout spolehlivý a ucelený pohled na diagnostiku aplikace. Takovým mechanismem je Windows Azure Diagnostics (WAD). Ten lze použít například v následujících scénářích [33]:

- Měření výkonu
- Měření spotřebovávaných zdrojů
- Detekce problémů
- Debugging
- Rozhodování o škálování (změna počtu instancí a přidělených zdrojů)

Značnou výhodou WAD je skutečnost, že používá tradiční nástroje .NET Frameworku (např. System.Diagnostics). Vývojáři tedy mohou používat známé nástroje se známou syntaxí, což snižuje dobu učení při přechodu na platformu Windows Azure. Hlavní výhodu to ale představuje u migrace on-premise aplikací, u kterých je možné zachovat stávající systém sběru diagnostických údajů. Samotný sběr diagnostiky je zachován, ale další zpracování diagnostických dat je už specifické pro cloud.

Každá role je nasazena v určitém počtu instancí, který se v čase může měnit. Přitom každá instance musí sbírat diagnostické údaje zvlášť. Je ale nutné mít centralizovaný přístup k nasbíraným údajům. Princip fungování WAD ukazuje *Obrázek 9 - Windows Azure Diagnostics*.



Obrázek 9 - Windows Azure Diagnostics [33]

Instance role má spuštěný Diagnostic Monitor, který na základě své konfigurace sleduje chod instance a sbírá požadované informace. Ty ukládá do Local Storage, což je lokální úložiště dané instance, které není synchronizováno s ostatními instancemi. Informace jsou sbírány z dvou hlavních zdrojů. Prvním je chod samostatné instance – Trace a Debug záznamy v kódu, logy, výpisy po pádu. Druhým jsou datové zdroje Windows – IIS Logs, Failed Request Logs, Windows Event Logs a Performance Counters. Nasbírané informace uložené v Local Storage nejsou zcela perzistentní, při překročení kvóty nebo recyklace role jsou ztraceny. Local Storage je jen jakýmsi zásobníkem, odkud mohou být diagnostické informace v případě potřeby přesunuty do perzistentního Windows Azure Storage. Tam jsou dle svého typu uloženy ve formě Blobů nebo tabulek. Přesun do Azure Storage může být realizován dvěma způsoby, a to buď na požádání, nebo dle plánu v určitých časových intervalech. Přitom může být nastaven filtr pro uložení pouze určité podmnožiny dat, např. jen data označená jako kritická. Požadavek na získání dat z Local Storage může přijít od instance samotné, instance jiné role nebo i od aplikace mimo cloud – například WPF aplikace pro diagnostiku Azure služeb.

WAD poskytuje různé datové zdroje pro získávání diagnostických informací. Základní přehled těchto zdrojů ukazuje tabulka *Tabulka 22 - Přehled datových zdrojů Windows Azure Diagnostics*.

Tabulka 22 - Přehled datových zdrojů Windows Azure Diagnostics [34]

Datový zdroj	Výchozí nastavení	Popis	Podporované typy rolí
Windows Azure logs	Zapnuto	Záznamy Trace zpráv nasbírané WAD trace listenerem.	Web, Worker
IIS 7.0 logs	Zapnuto	Záznamy informací z IIS logů platné pouze pro Web role.	Web
WAD infrastructure logs	Zapnuto	Záznamy generované WAD infrastrukturou užitečné pro detekci problémů samotného Windows Azure Diagnostics.	Web, Worker
Failed Request logs	Vypnuto	Zaznamenává informace o neúspěšných požadavcích na IIS stránky nebo aplikace.	Web
Windows Event logs	Vypnuto	Zaznamenává události OS.	Web, Worker
Performance counters	Vypnuto	Zaznamenává informace o hodnotách zvolených indikátorů výkonu v čase.	Web, Worker
Crash dumps	Vypnuto	Zaznamenává informace o stavu OS a paměti v okamžiku pádu.	Web, Worker
Custom error logs	Vypnuto	Umožňuje ukládat vlastní informace o chybách.	Web, Worker

Práce s diagnostikou ve Windows Azure

Import diagnostického modulu

1. V souboru ServiceDefinition.csdef přidat import diagnostického modulu

```
<ServiceDefinition name="WindowsAzureProject10"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">
  <WebRole name="WebRole1" vmsize="Small">
    <Imports>
      <Import moduleName="Diagnostics" />
    </Imports>
  </WebRole>
</ServiceDefinition>
```

2. Projekt musí mít referenci na Microsoft.WindowsAzure.Diagnostics.
3. V souborech zdrojových kódů tříd, které pracují s diagnostikou, musí být přítomen using statement:

```
using Microsoft.WindowsAzure.Diagnostics;
```

Konfigurace diagnostiky

Jednotlivé typy diagnostických dat je třeba nakonfigurovat zvlášť. Níže je uveden základní postup pro hlavní typy.

Windows Azure Logs

Je nutné přidat do konfiguračního souboru následující tag, který aktivuje trace listener.

```
<system.diagnostics>
   <trace>
     <listeners>
       <add type="Microsoft.WindowsAzure.Diagnostics.DiagnosticMonitorTraceListener,
       Microsoft.WindowsAzure.Diagnostics, Version=1.0.0.0, Culture=neutral,
       PublicKeyToken=31bf3856ad364e35" name="AzureDiagnostics">
          <filter type="" />
       </add>
     </listeners>
   </trace>
</system.diagnostics>
```

Pro Web roli je konfiguračním souborem Web.config, pro Worker roli je to App.config.

Záznamy se pak zapisují pomocí standardních Trace funkcí .NET – např. Trace.TraceError, Trace.TraceInformation, Trace.TraceWarning nebo Trace.Assert.

Windows Event Logs

Příklad nastavení je uveden níže:

```
var config = DiagnosticMonitor.GetDefaultInitialConfiguration();
config.WindowsEventLog.DataSources.Add("System!*");
config.WindowsEventLog.DataSources.Add("Application!*");
DiagnosticMonitor.Start("StorageConnectionString", config);
```

Nejdříve je vytvořena instance konfigurace Diagnostic Monitoru. Poté jsou jí nastaveny datové zdroje, odkud má čerpat záznamy. Nakonec je spuštěn Diagnostic Monitor s výše nastavenou konfigurací. Parametr *StorageConnectionString* slouží k připojení k Azure Storage pro trvalé uložení diagnostických dat (viz dále).

Crash Dumps

Existují dva typy výpisů po pádu. První je částečný a druhý je úplný. Pro zapnutí sběru částečných výpisů je nutné zavolat metodu:

```
Microsoft.WindowsAzure.Diagnostics.CrashDumps.EnableCollection(false);
```

Pro úplný výpis se volá stejná metoda s parametrem true:

```
Microsoft.WindowsAzure.Diagnostics.CrashDumps.EnableCollection(true);
```

Performance Counters

Sledování indikátorů výkonu aktivuje následujícím způsobem:

```
var config = DiagnosticMonitor.GetDefaultInitialConfiguration();
var counter1 = new PerformanceCounterConfiguration();
counter1.CounterSpecifier = @"\Processor(_Total)\% Processor Time";
counter1.SampleRate = TimeSpan.FromSeconds(5);
DiagnosticMonitor.Start("StorageConnectionString", config);
```

Na prvním řádku se tvoří nová instance konfigurace. Dále je vytvořena instance nastavení konkrétního indikátoru. Poté je definováno, o jaký indikátor se jedná. Následně je určena frekvence sběru dat, zde konkrétně pět vteřin. Nakonec je spuštěn diagnostický monitor s výše definovanou konfigurací. Seznam dostupných indikátorů výkonu je dostupný na

http://msdn.microsoft.com/en-us/library/windowsazure/hh411520.aspx.

Trvalé uložení diagnostických dat

Nasbíraná diagnostická data jsou ukládána do Local Storage příslušné instance. Před prohlížením je nutné data ze všech instancí přenést do perzistentního úložiště Windows Azure Storage. To je možné buď na vyžádání, nebo ve stanovených časových intervalech. Nastavení přenosu v intervalech je následující:

1. Zaznamenat přihlašovací údaje k Storage Accountu do konfiguračního souboru (ServiceConfiguration.cscfg). Údaje by mohly být přímo v kódu role, ale takové řešení je neflexibilní. Při změně stačí pouze nahradit konfigurační soubor namísto toho, aby se zasahovalo do kódu role a ta se poté musela znovu nasadit.

   ```
   <ConfigurationSettings>
      <Setting name="StorageConnectionString"
      value="DefaultEndpointsProtocol=https;AccountName=[Jméno Storage
      Accountu];AccountKey=[Přístupový klíč ke Storage Accountu]" />
   </ConfigurationSettings>
   ```

2. Nastavení Diagnostic Monitoru.

```
DiagnosticMonitorConfiguration config =
DiagnosticMonitor.GetDefaultInitialConfiguration();
config.Logs.ScheduledTransferPeriod = TimeSpan.FromMinutes(10);
config.Logs.ScheduledTransferLogLevelFilter = LogLevel.Critical;
```

Nejdříve je vytvořena instance konfigurace Diagnostic Monitoru. V dalším kroku je nastaven časový interval, ve kterém mají být data z Local Storage přesunuta do Azure Storage. Na posledním řádku se nastavuje filtr pro přenášená data. V uvedeném případě budou přenesena pouze data s kritickou důležitostí. Je výhodné přenášet a ukládat pouze data nezbytná ke konkrétnímu úkonu a ne všechna dostupná data. U Azure Storage je zpoplatněno využité místo a stejně tak i počet transakcí čtení a zápisu. Za pozornost stojí, že výše uvedené nastavení ukládá pouze Azure Logs (config.Logs). Pro ostatní typy dat je třeba ukládání definovat zvlášť. Například:

```
config.PerformanceCounters.ScheduledTransferPeriod = TimeSpan.FromMinutes(10);
config.PerformanceCounters.ScheduledTransferPeriod = TimeSpan.FromMinutes (10);
config.WindowsEventLog.ScheduledTransferPeriod = TimeSpan.FromMinutes (10);
```

3. Spuštění Diagnostic Monitoru.

```
DiagnosticMonitor.Start("StorageConnectionString", config);
```

StorageConnectionString je jméno Connection Stringu, který je používán pro připojení k účtu Azure Storage a který je definován v kroku 1. Config je instance konfigurace Diagnostic Monitoru, která je definována v kroku 2.

Prohlížení diagnostických dat

K uloženým diagnostickým datům v Azure Storage je možné přistupovat mj. i pomocí specializovaných aplikací mimo prostředí Windows Azure (ZDROJ).

- Nejjednodušší možností je použít přímo Visual Studio, konkrétně nástroj Server Explorer[14].
- Azure Storage Explorer – nástroj zdarma ke stažení nabízející mnoho funkcí pro základní práci s úložišti Azure Storage.[15]
- Azure Diagnostics Manager – Specializovaný nástroj od společnosti Cerebrata přímo specializovaný na Azure Diagnostics nabízející mnoho pokročilých funkcí. Software je zpoplatněn.[16]

[14] Více informací je dostupných na http://msdn.microsoft.com/en-us/library/windowsazure/ff683677.aspx
[15] Dostupný z http://azurestorageexplorer.codeplex.com/
[16] K dispozici na http://www.cerebrata.com/Products/AzureDiagnosticsManager/Default.aspx

6 Testování a provoz

6.1 Testování

V průběhu celého období implementace aplikace probíhalo ruční testování. Správně odladit aplikaci v cloudu však není zcela jednoduché. Již od začátku jsme počítali s modelem, kdy co nejdříve vytvoříme funkční beta verzi, kterou poté nasadíme na Windows Phone Marketplace, kde bude volně a zdarma ke stažení. Uživatelé pak poskytnou zpětnou vazbu, budou posílat bug-reporty a aplikace sama bude generovat automatická hlášení o pádech. Na základě zpětné vazby a odhalených chyb implementujeme novou funkčnost, pozměníme stávající a opravíme chyby. Poté bude opět aktualizována aplikace na Marketplace. To bude probíhat v rychlých cyklech, takže vývoj bude pružně reagovat na požadavky komunity a bude výrazně větší šance, že se chyby nebo vady návrhu objeví dříve a bude jednodušší a rychlejší je odstranit. Původní plány dokonce uvažovaly aktivní spolupráci se zájmovými skupinami z řad cyklistů a vozíčkářů v Praze. Protože nebyly k dispozici mobilní telefony s WP7, GINA Software přislíbila zapůjčení speciálních GPS lokátorů, s jejichž pomocí by mohli zúčastnění trasovat bez mobilního klienta a rozšířit pak naši databázi o cenná data. Z této aktivní spolupráce s komunitou nakonec sešlo a osobně to vnímám jako promarnění velké příležitosti. Zvláště vzhledem k tomu, jak se nakonec komunita na WP7 Marketplace ukázala jako nepočetná a pasivní.

První nasazení aplikace na Marketplace proběhlo v poměrně rané fázi vývoje a jednalo se o closed-beta (nebo spíše alpha). Byla určena jen pro partnery projektu z řad Microsoftu a GINA Software především jako demonstrace aktuálního stavu a možnost k poskytnutí zpětné vazby. Vzhledem k tomu, že GINA neměla tou dobou k dispozici telefony s WP7, nepodařilo se nám tímto způsobem získat zpětnou vazbu. Ze strany Microsoftu proběhlo testování aplikace pouze v omezené míře.

Další fáze testování byla již otevřená beta na Windows Phone Marketplace. Tato fáze nenaplnila původní očekávání v mnoha směrech. Každá aplikace, která se uchází o umístění na Marketplace, musí projít certifikačním procesem Microsoftu, musí splňovat určitá formální kritéria a především nesmí obsahovat chyby. Výjimky musí být odchytávány a zpracovány a nikdy nesmí dojít k nekontrolovanému pádu aplikace.[17] Tento certifikační proces trvá týden. Při prvním pokusu byla aplikace odmítnuta. Byl poskytnut seznam některých formálních náležitostí, které nesplňuje, a také nastalo několik případů, při kterých došlo k pádu aplikace. Na druhý pokus byla aplikace přijata. Certifikační proces však celkově zkrátil dobu veřejné bety o celé dva týdny. Celkový počet stažení aplikace za testování byl velmi neuspokojivý. Většina uživatelů, kteří si aplikaci stáhli, ji přitom použili

[17] Více o certifikačním procesu a požadavcích na http://msdn.microsoft.com/en-us/library/hh184843(v=vs.92).aspx

velmi málo (soudě podle informací z databáze). Zpětnou vazbu pak nám neposkytl nikdo, což se při tak malém počtu stažení a aktivních uživatelů dalo očekávat. Přišli jsme tak o cenný zdroj informací. Pravděpodobně bylo vhodné aktivně navázat spolupráci s konkrétní skupinou z řad cyklistů nebo vozíčkářů, kteří by byli více motivováni ke spolupráci. Tento model však narážel na problém nutnosti mobilních zařízení se systémem WP7, která jsou velmi málo rozšířena. Microsoft přitom neměl dostatek zařízení k zapůjčení.

6.2 Provoz

Serverová část aplikace je v současné době nasazena jako WCF služba na adrese tracktor.cloudapp.net na portu 8080. Na stejné adrese na standardním portu 80 je nasazen webový klient, tedy port mobilního klienta do standardního Silverlightu.

Mobilní klient, tedy aplikace pro Windows Phone 7, je volně dostupný na Marketplace na adrese http://www.windowsphone.com/cs-CZ/apps/9b374f22-9fbe-4242-b5aa-7ee1014d9d64.

Aplikace bude ve veřejném provozu po dobu, dokud bude v platnosti dohoda s Microsoftem o financování poplatků za Windows Azure.

7 Náměty k rozšíření

Dle mého názoru má aplikace potenciál přinést užitek vozíčkářům a cyklistům a nabízí funkce, které v současné době běžné konkurenční aplikace neobsahují. Kdyby bylo zdokonalení aplikace věnováno další úsilí, tento potenciál by mohl ještě významně vzrůst. Hlavní užitek, který by mohly přinést třetí strany do projektu, je především jeho propagace a rozšíření mezi uživatele. Absenci povědomosti o aplikaci vnímám jako jednu z jejích největších slabin. Kromě propagace by rozšíření aplikace také mohla výrazně pomoci tvorba klientů pro alternativní platformy (především Android), protože podíl mobilních zařízení s OS WP7 je velmi neuspokojivý. Serverová část by přitom mohla být zachována v současném stavu. Aktuální webový klient by také měl být rozšířen a doplněn tím způsobem, aby mohl být používán jako plnohodnotná alternativa klienta mobilního. Například by velmi pomohla možnost manuálně zadat trasu. Další cestou rozšiřování by mohla být integrace se sociálními sítěmi. Velká část operací, které nyní probíhají ručně, by na ně mohly být delegovány. Nebylo by například nutné registrovat nový účet a zadávat všechny údaje, ale informace jako jméno, email nebo profilová fotografie by byly načteny automaticky. Systém by také snadno poznal, kteří přátelé ze sociálních sítí již aplikaci používají, a mohl by je velmi jednoduše přidat. Absolvované trasy by mohly být uveřejněny na profilu uživatele. Tato integrace by navíc přispěla k zvýšení povědomí o aplikaci prostřednictvím sociální sítí a zvýšila by počet uživatelů. Další možnost, která by výrazně zlepšila výkon aplikace, je generování mapových podkladů s nakreslenými trasami přímo na serveru. V současné době dostává klient trasy jako množiny bodů, a to klade poměrně vysoký nárok na jeho výkon při větším počtu tras. Vhodné by také bylo vytvořit speciální třídu uživatelů – administrátorů, kteří by měli neomezená práva na manipulaci s daty a mohli by např. mazat nevhodné/nerelevantní komentáře a odstraňovat následky „vandalství“ neukázněných uživatelů. Velmi přínosnou funkcí by bylo také na základě existujících tras v databázi, informacích o jejich náročnosti a jiných úskalích navrhnout trasu z aktuální pozice uživatele do cílové destinace. Jedná se však o poměrně složitou funkci, která navíc vyžaduje poměrně velké množství nasbíraných tras v dané lokalitě.

Vzhledem k časovým možnostem autorů aplikace není pravděpodobné, že vývoj bude pokračovat aktivní formou příliš dlouho. Očekáváme ještě kratší období, kdy doplníme aplikaci o některé menší funkce, ale poté již aktivní vývoj pravděpodobně skončí. Budeme velmi rádi, pokud nás kontaktují případní zájemci. Jistě se dohodneme na vhodné formě spolupráce. Pokud by se našla skupina, která by jako celek chtěla pokračovat v práci na projektu, bude vítána. S návrhy spolupráce se můžete obrátit přímo na mě (vojtech.ruz@gmail.com) nebo na Bc. Víta Čurdu (drcurda@gmail.com).

8 Naplnění cílů práce

8.1 Naplnění cílů práce

8.1.1 Cíl první – Průzkum platformy Windows Azure

Prvním cílem práce bylo:

„…prozkoumat možnosti a potenciální problémy platformy Windows Azure a její spojení s platformou Windows Phone 7 a poskytnout tak podklady, doporučení a varování vývojářům zabývající se touto problematikou.“

První částí práce, která přináší užitečné informace pro vývojáře na platformě Azure, je definována v kapitole *3 – Teoretický základ*. Tato kapitola obsahuje úvod do Cloud Computingu, ale hlavně popsání struktury platformy Windows Azure a jejích jednotlivých komponent. To je důležité pro získání základního přehledu o platformě. V dalším textu se pak odkazuje na pojmy v této kapitole definované a očekává se, že čtenář je s nimi již seznámen. Tato kapitola nabízí ucelený přehled o platformě a jejích možnostech k dubnu 2012 a pracuje s nejnovějšími dostupnými údaji. Čtenář tedy nemusí složitě zjišťovat, které zdroje jsou již zastaralé a neaktuální, vzhledem k tomu, že platforma Azure se neustále bouřlivě vyvíjí. Klíčové je především vysvětlení následujících oblastí:

- Azure Compute a porozumění rozdělení aplikace do jednotlivých rolí a jejich instancí.
- Azure Storage a porozumění jednotlivým mechanismům pro ukládání dat a práci s nimi v prostředí cloudu.
- Fungování Windows Azure Platform jako celku a role jednotlivých komponent.

Dalším přínosem je kapitola *4 – Analýza*. Zde je dobře patrné, jak na základě definovaných požadavků probíhá návrh aplikace, a to se všemi specifiky, které s sebou cloudové prostředí nese. Zde se jedná především o celkový návrh architektury, definici jednotlivých komponent aplikace a jejich interakce. Dále jsou vysvětleny důvody výběru jednotlivých komponent (jako například úložiště) a mechanismů společně s jejich alternativami a porovnány výhody a nevýhody jednotlivých řešení. Problém návrhu je tedy dostatečně zobecněn, aby mohl být prospěšný i pro vývoj jiných typů aplikací v cloudu s odlišnými požadavky a specifiky. Za hlavní konkrétní přínosy lze označit:

- Jak oddělit front-end a back-end tak, aby byla zachována maximální dostupnost služby a maximální škálovatelnost.
- Jak použít Azure Queue ke snížení závislosti mezi jednotlivými komponentami a k celkovému zvýšení škálovatelnosti a odolnosti aplikace.

- Výběr vhodných mechanismů k ukládání a manipulaci s daty na základě jejich předností, nedostatků a povahy dat.
- Celkové navržení aplikace a rozdělení do vhodných komponent a navržení komunikace mezi těmito komponentami.

Další kapitola přispívající k naplnění cíle je *5.3 Specifika a problémy vývoje pro Windows Azure*. Zde jsou specifika platformy nahlížena z hlediska implementačního a nikoliv návrhového jako v kapitole *Analýza*. Jedná se o popis specifických a problematických oblastí vývoje na platformě Windows Azure jako je kvalitní zabezpečení komunikace mezi klientem a serverem nebo ladění a logování aplikací v cloudu. Jedná se o běžné scénáře pro cloudové aplikace, které jsou přínosem bez ohledu na povahu a zaměření vyvíjené aplikace. Z konkrétních přínosů lze jmenovat především ukázku toho, jak:

- Měřit výkonové ukazatele aplikace
- Ladit aplikaci a diagnostikovat problémy
- Implementovat bezpečnou šifrovanou komunikaci na základě WCF
- Překonat výchozí limity zpráv

U podobných cílů je často těžké ověřit jejich naplnění, protože nelze definovat tvrdou metrikou, která by popisovala různé varianty a stupně naplnění. Osobně jsem přesvědčen, že cíl se podařilo naplnit a že jsem poskytl dostatek kvalitních informací, které mají potenciál pomoci vývojářům na platformě Windows Azure.

8.1.2 Cíl druhý – Tvorba aplikace

Prvním cílem práce bylo „vytvořit serverovou část aplikace, která bude sloužit primárně cyklistům a osobám se sníženou schopností pohybu k lepšímu plánování trasy (především městem)“ a „…bude vyhovovat požadavkům blíže definovaným v kapitole 4.1.“.

V kapitole 4.1 jsou definovány funkční, kvalitativní a technologické požadavky.

Požadavky Funkční

Všechny funkční požadavky byly naplněny. Podrobnosti ukazuje následující tabulka.

Tabulka 23 - Naplnění funkčních požadavků aplikace

ID	Název požadavku Metrika Naplnění požadavku
UP 1	**Možnost upozorňovat na význačná místa na mapě a odlišit je dle typu.**
	1. Uživatel může označit místo na mapě jakožto bod zájmu. 2. Bod zájmu musí mít jméno a volitelně vysvětlující popis. 3. Uživatel vybírá kategorii bodu zájmu - na výběr jsou alespoň kategorie „Nebezpečí“, “Varování“, „Obecná informace“ a „Doporučení“.
	SPLNĚNO. Uživatel může přidávat body zájmu na mapu, přičemž jsou splněny všechny výše uvedené požadavky.
UP 2	**Možnost sledovat svou polohu při jízdě a zaznamenávat trasu.**
	1. Aplikace poskytuje informaci o aktuální uživatelově podobě. 2. Uživatel může vynutit začátek a konec zaznamenávání trasy.
	SPLNĚNO. Uživatel vidí svou aktuální polohu a může zapínat/vypínat sledování trasy.
UP 3	**Možnost ohodnotit své trasy stupněm obtížnosti.**
	1. Uživatel může svou ukončenou trasu ohodnotit známkou z předem definované stupnice a vyjádřit tak názor o její obtížnosti.
	SPLNĚNO. Uživatel může svou trasu ohodnotit stupněm obtížnosti 1-10.
UP 4	**Možnost navázat přátelství s ostatními členy komunity.**
	1. Uživatel může vyhledávat mezi ostatními uživateli. 2. Uživatel může navázat přátelství s jiným uživatelem. 3. Přátelství musí být potvrzeno druhou stranou.

ID	Název požadavku Metrika Naplnění požadavku
	SPLNĚNO. Uživatelé se mohou vyhledávat navzájem a posílat si žádosti o přátelství. Pokud druhá strana přijme, je přátelství navázáno.
UP 5	**Možnost procházet trasy vlastní a trasy přátel.**
	1. Uživatel může zobrazit seznam svých dosavadních tras. 2. Uživatel může zobrazit detaily o konkrétní trase. 3. Uživatel může prohlížet trasy náležející konkrétnímu příteli.
	SPLNĚNO. Uživatel může procházet seznam svých tras a zobrazit jednotlivé trasy na mapě. Každá z tras má dostupné detaily. Uživatel může procházet přátele a u každého z nich vidí seznam jeho tras.
UP 6	**Možnost sledovat pozici přátel na mapě.**
	1. Uživatel vidí poslední známou pozici svých přátel na mapě. 2. Pozice nemusí být aktualizována v reálném čase, ale v libovolném intervalu (nejméně však jednou za přítelovo zaznamenání jedné trasy).
	SPLNĚNO. Systém uchovává poslední známou pozici každého uživatele na mapě. Tuto pak lze zobrazit na mapě pro všechny své přátele, kteří ji mají uloženu.
UP 7	**Možnost plánovat trasu i na základě dat nasbíraných ostatními členy komunity.**
	1. Uživatel vidí body zájmu vytvořené ostatními uživateli. 2. Uživatel může komentovat cizí body zájmu. 3. Uživatel si může zobrazit i cizí trasy v blízkosti. Tyto jsou anonymizovány a nejsou spojitelné s konkrétním uživatelem.
	SPLNĚNO. Body zájmu vidí všichni ostatní členové, kteří jsou dostatečně blízko k nim. Komentovat bod zájmu může přitom každý, nejen autor. Uživatel si může nechat zobrazit nabídky tras v okolí. Ty jsou sestavené na základě tras všech uživatelů a jsou rozděleny do krátkých segmentů, které mají větší vypovídací hodnotu.
UP 8	**Možnost plánovat trasu přímo v terénu.**
	1. Uživatel si může vyhledat body zájmu v blízkosti své aktuální pozice v souladu s UP 7. 2. Uživatel si může vyhledat trasy v blízkosti své aktuální pozice v souladu s UP 7.

ID	Název požadavku Metrika Naplnění požadavku
	SPLNĚNO. Uživatel má možnost zobrazit všechny body zájmu v okolí. Uživatel si může nechat zobrazit segmenty tras v okolí, které mu umožní lepší rozhodování při volbě trasy.
UP 9	**Jednoduché ovládání.**
	1. Snaha o minimalizaci komplexnosti uživatelského rozhraní. Tento požadavek nemá tvrdou metriku splnění, jedná se pouze o doporučení, na které bude brán maximální možný zřetel, pokud to nebude v konfliktu s ostatními cíli. Tento požadavek je pro potřeby posouzení splnění funkčních požadavků aplikace automaticky považován za splněný.
	SPLNĚNO. Definovaná metrika určuje, že požadavek je automaticky splněn.

Požadavky kvalitativní

Stabilita

Požadavek: Minimalizovat počet nekontrolovaných pádů aplikace.

Metrika: Počet pádů aplikace nesmí převýšit počet stažení aplikace za sledované období.

Naplnění: Ano.

Marketplace pro Windows Phone poskytuje statistiky o počtu stažení aplikace v čase a počtu pádů aplikace v čase. Do sledovaného období jsem neuvažoval periodu první certifikace aplikace, kdy byla certifikace odmítnuta a ještě obsahovala některé později opravené chyby. Sledované období je tedy až od skutečného udělení certifikace pro Marketplace. Sledování probíhalo od 27. 3. 2012 do 21. 4. 2012. Za tu dobu bylo zaznamenáno 12 stažení a 1 pád. Aplikaci tedy lze považovat za stabilní a cíl byl naplněn. Prodloužení sledovaného období by pravděpodobně navýšilo počet stažení zcela minimálně – 11 z 12 stažení proběhlo během prvních pěti dnů po uveřejnění. Pak byla pravděpodobně aplikace přesunuta z kategorie „Nové“ a tím se dramaticky snížila její viditelnost.

Dostupnost aplikace

Požadavek: Dostatečná vnější dostupnost serverové části aplikace.

Metrika: Dostupnost minimálně 98%. Měřeno jako podíl času, kdy je aplikace dostupná, a celkového času za období. Období měření musí být minimálně 30 dnů.

Naplnění: Ano.

Microsoft garantuje dostupnost aplikace ve svém SLA k Windows Azure, podkapitole Compute SLA takto: Garantovaná vnější konektivita alespoň 99,95% času. Časový úsek je 30 dní, měření probíhá v pětiminutových intervalech. Platí za podmínky, pokud jsou nasazeny alespoň dvě instance aplikace v rozdílných aktualizačních a chybových doménách. Tato podmínka je splněna. Dále je v 99,99% garantována automatická detekce zastavení procesu instance a okamžité provedení nápravné akce. [35]

Dostupnost dat

Požadavek: Datové úložiště používané serverovou částí aplikace a jeho obsah musí mít dostatečnou dostupnost.

Metrika: Dostupnost minimálně 98%. Měřeno jako podíl, kdy je aplikace dostupná, a celkového času za období. Období měření musí být minimálně 30 dnů.

Naplnění: Ano.

Microsoft garantuje dostupnost aplikace ve svém SLA k Windows Azure, podkapitole SQL Azure SLA takto: Garantována konektivita k databázi minimálně 99.99% v časovém úseku 30 dní, měřeno v pětiminutových intervalech. [36]

Požadavky Technologické

Platforma Windows Azure

Požadavek: Serverová část aplikace musí být implementována na platformě Windows Azure.

Metrika: Implementováno na Windows Azure ano/ne.

Naplnění: Ano. Řešení je implementováno na platformě Windows Azure.

Shrnutí

Všechny požadavky (uživatelské, kvalitativní, technologické) byly úspěšně naplněny a cíl lze tedy považovat za zcela splněný.

Závěr

V rámci práce se podařilo splnit oba hlavní cíle (detailní vyhodnocení v kapitole *8 Naplnění cílů práce*). Byly poskytnuty informace o platformě Windows Azure, její struktuře a mechanismech. V analytické části byly demonstrovány postupy při návrhu architektury aplikace a porovnání a výběru jednotlivých alternativ, pomocí kterých lze řešit daný problém. V implementační části byly nastíněny hlavní problémy, které se objevily při vývoji. Získané poznatky byly zobecněny a jsou tedy vhodné jako podklady pro tvorbu vývojářů na platformě Windows Azure, především pak pro ty, kteří zvolí model spolupráce s mobilním klientem na platformě Windows Phone 7.

Byla vyvinuta aplikace, která usnadňuje plánování trasy v obtížném terénu pro cyklisty a osoby se sníženou schopností pohybu. Aplikaci se podařilo implementovat takovým způsobem, aby byly naplněny všechny definované požadavky. Aplikace je k dispozici k volnému použití. Jakékoliv snahy o budoucí rozšíření třetími stranami jsou vítány.

Diplomová práce byla zpracovávána poměrně dlouhou dobu. V době zahájení prací existovalo velmi omezené množství informačních zdrojů a v ČR nebyl dostupný veřejný přístup k Windows Azure. Od té doby uplynul rok a půl a situace se samozřejmě změnila. Windows Azure je dostupný i pro veřejnost a jako nástupce systému WP 7 je ohlášen Windows 8. Samotná platforma Windows Azure za dobu tvorby práce prošla mnohými změnami, všechny jsou však v práci zachyceny a informace nelze považovat za zastaralé, neboť všechny jsou aktualizovány k začátku dubna 2012. V současné době stále neexistuje dostatek kvalitních a hlavně aktuálních zdrojů a jsem přesvědčen, že v této oblasti má práce co nabídnout. Především se jedná o samotný návrh aplikace pro cloud, volba optimálních nástrojů a mechanismů z platformy Windows Azure a demonstrace rozdílů mezi nimi, jejich silných a slabých stránek. Nejedná se však pouze o čistou teorii, vše je demonstrováno na vývoji skutečné cloudové aplikace s reálným použitím. Sám jsem se při tvorbě práce mnohému naučil a pevně věřím, že se mi podařilo co možná nejvíce z těchto poznatků předat dál prostřednictvím této práce. Také doufám, že se najde pokračovatel, který by se podílel na projektu a vhodně jej rozšířil, případně se ho ujal jako celku.

9 Zdroje

[1]. **Hay, Chris a Prince, H. Brian.** *Azure In Action.* Stamford : Manning , 2011. 9781935182481.

[2]. **Brunetti, Roberto.** *Windows Azure Step by Step.* Redmont : Microsoft Press, 2011. 978-0-7356-4972-9.

[3]. **Otrusinová, Jana.** Mapa přístupnosti města Brna pro vozíčkáře - teoretická východiska a vlastní řešení. [Diplomová práce]. Brno : Masarykova Univerzita, 2012.

[4]. **Biskupič, Martin.** Interaktívna mapa prístupnosti pre hendikepovaných. [Bakalářská práce]. Brno : Masarykova univerzita, 2011.

[5]. **GINA Software.** GINA System - O systému. *GINA System.* [Online] [Citace: 15. březen 2012.] http://www.ginasystem.cz/gina-system.htm.

[6]. **Mell, Peter a Grance, Timothy.** The NIST Definition of Cloud Computing. *National Institute of Standards and Technology.* [Online] NIST. [Citace: 20. březen 2012.] http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[7]. **Remde, Kevin.** SaaS, PaaS, and IaaS.. Oh my! ("Cloudy April" - Part 3). *Full of I.T.* [Online] 3. duben 2011. [Citace: 20. březen 2012.] http://blogs.technet.com/b/kevinremde/archive/2011/04/03/saas-paas-and-iaas-oh-my-quot-cloudy-april-quot-part-3.aspx.

[8]. **Microsoft.** API References for Windows Azure. *MSDN.* [Online] [Citace: 23. březen 2012.] http://msdn.microsoft.com/en-us/library/windowsazure/ff800682.aspx.

[9]. —. Windows Azure. *MSDN.* [Online] 12. říjen 2012. [Citace: 20. březen 2012.] http://msdn.microsoft.com/en-us/library/dd179367.aspx.

[10]. —. How to Configure Virtual Machine Sizes. *MSDN.* [Online] 7. září 2011. [Citace: 23. březen 2012.] http://msdn.microsoft.com/en-us/library/windowsazure/ee814754.aspx.

[11]. —. Windows Azure Compute. *MSDN.* [Online] [Citace: 23. březen 2012.] https://www.windowsazure.com/en-us/home/features/compute/#computeinstancesize.

[12]. —. Overview of the Windows Azure VM Role. *MSDN.* [Online] 8. březen 2011. [Citace: 23. březen 2012.] http://msdn.microsoft.com/en-us/library/windowsazure/gg433107.aspx.

[13]. **White, Jim.** Windows Azure Table Storage vs. Windows SQL Azure. *Intertech.* [Online] 1. červen 2010. [Citace: 23. březen 2012.] http://www.intertech.com/Blog/post/Windows-Azure-Table-Storage-vs-Windows-SQL-Azure.aspx.

[14]. **Microsoft.** SQL Azure. *Windows Azure.* [Online] [Citace: 23. březen 2012.] https://www.windowsazure.com/en-us/home/features/sql-azure/.

[15]. —. Understanding Block Blobs and Page Blobs. *MSDN.* [Online] 30. září 2011. [Citace: 24. březen 2012.] http://msdn.microsoft.com/en-us/library/windowsazure/ee691964.aspx.

[16]. —. Restricting Access to Containers and Blobs. *MSDN.* [Online] [Citace: 24. březen 2012.] http://msdn.microsoft.com/en-us/library/dd179354.aspx.

[17]. —. Blob Service REST API. *MSDN.* [Online] [Citace: 24. březen 2012.] http://msdn.microsoft.com/en-us/library/windowsazure/dd135733.aspx.

[18]. **Calder, Brad a Edwards, Andrew.** Windows Azure Drive. *Microsoft.* [Online] únor 2010. [Citace: 24. březen 2012.] http://go.microsoft.com/?linkid=9710117.

[19]. **Microsoft.** Windows Azure Service Bus & Windows Azure Connect: Compared & Contrasted. *Windows Azure Customer Advisory Team.* [Online] [Citace: 25. březen 2012.] http://windowsazurecat.com/2011/08/windows-azure-service-bus-connect-compared-and-contrasted/.

[20]. **Bertocci, Vittorio a Wegner, Wade.** Re-Introducing the Windows Azure Access Control Service. *MSDN Magazine.* [Online] prosinec 2010. [Citace: 25. březen 2012.] http://msdn.microsoft.com/en-us/magazine/gg490345.aspx.

[21]. **Microsoft.** Windows Azure CDN Node Locations. *MSDN.* [Online] 2. březen 2011. [Citace: 25. březen 2012.] http://msdn.microsoft.com/en-us/library/windowsazure/gg680302.aspx.

[22]. —. Overview of the Windows Azure CDN. *MSDN.* [Online] 8. březen 2011. [Citace: 25. březen 2012.] http://msdn.microsoft.com/en-us/library/ff919703.aspx.

[23]. —. Windows Azure Caching. *Windows Azure.* [Online] [Citace: 25. březen 2012.] https://www.windowsazure.com/en-us/home/features/caching/.

[24]. **Convective.** Windows Azure Traffic Manager. *Convective.* [Online] 15. duben 2011. [Citace: 25. březen 2012.] http://convective.wordpress.com/2011/04/15/windows-azure-traffic-manager/.

[25]. **Microsoft.** Learn About Windows Azure Marketplace. *Windows Azure Marketplace.* [Online] [Citace: 25. březen 2012.] https://datamarket.azure.com/about.

[26]. **Object Management Group.** UML Supersturcture. *UML 2.4.1.* [Online] srpen 2011. [Citace: 20. březen 2012.] http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF.

[27]. Database Sharding. *Code Futures.* [Online] [Citace: 26. březen 2012.] http://www.codefutures.com/database-sharding/.

[28]. **Kumar, Dhananjay.** SQL Azure with LINQ. *C# corner.* [Online] 10. duben 2011. [Citace: 26. březen 2012.] http://www.c-sharpcorner.com/uploadfile/dhananjaycoder/sql-azure-with-linq/.

[29]. **Fultz, Joseph.** SQL Azure and Windows Azure Table Storage. *MSDN Magazine.* [Online] listopad 2010. [Citace: 26. březen 2012.] http://msdn.microsoft.com/en-us/magazine/gg309178.aspx.

[30]. **Microsoft.** Windows Azure Pricing Overview. *Windows Azure.* [Online] [Citace: 26. březen 2012.] https://www.windowsazure.com/en-us/pricing/details/.

[31]. —. Programming WCF Security. *MSDN.* [Online] [Citace: 7. duben 2012.] http://msdn.microsoft.com/en-us/library/ms731925.aspx.

[32]. **Pialorsi, Paolo.** WCF configuration default limits, concurrency and scalability. *Bridge The Gap!* [Online] 23. březen 2008. [Citace: 18. duben 2012.] http://weblogs.asp.net/paolopia/archive/2008/03/23/wcf-configuration-default-limits-concurrency-and-scalability.aspx.

[33]. **Kerner, Matthew.** Windows Azure Monitoring, Logging, and Management APIs. *PDC10.* [Online] [Citace: 20. duben 2012.] http://ecn.channel9.msdn.com/o9/pdc09/ppt/SVC15.pptx.

[34]. **Microsoft.** Enabling Diagnostics in Windows Azure. *Windows Azure.* [Online] [Citace: 20. duben 2012.] https://www.windowsazure.com/en-us/develop/net/common-tasks/diagnostics/#step2.

[35]. —. Windows Azure Compute Service Level Agreement (SLA). *Microsoft Download Center.* [Online] 4. září 2010. [Citace: 22. březen 2012.] http://download.microsoft.com/download/0/E/E/0EE244BF-22CA-4180-ACF0-F2F40CAEE3D6/Windows%20Azure%20Compute%20SLA-Czech.doc.

[36]. —. SQL Azure Service Level Agreement (SLA). *Microsoft Download Center.* [Online] 11. říjen 2010. [Citace: 22. březen 2012.] http://download.microsoft.com/download/B/0/9/B09851E2-6177-4A62-83AB-3B591659CE1E/SQL%20Azure%20SLA-Czech.doc.

[37]. —. .NET Framework 4. *MSDN.* [Online] [Citace: 23. duben 2012.] http://msdn.microsoft.com/en-us/library/w0x726c2.aspx.

[38]. **Redkar, Tejaswi.** *Windows Azure Platform.* New York : Apress, 2010. 9781430235637.

[39]. **Abrahamsson, Pekka, a další, a další.** *Agile software development methods.* Espoo : VTT Publications, 2002. 951-38-6009-4.

[40]. **Blokdijk, Gerard.** *Service Level Agreement 100 Success Secrets: SLA, Service Level Agreements, Service Level Management and Much More.* místo neznámé : Emereo Publishing, 2008. 978-0980471618.

[41]. **Microsoft.** Microsoft Azure - SQL Azure. *Windows Azure.* [Online] [Citace: 23. duben 2012.] http://www.microsoft.com/cze/azure/sqlazure/.

[42]. —. Microsoft Azure - Hlavní stránka. *Windows Azure.* [Online] [Citace: 23. duben 2012.] http://www.microsoft.com/cze/azure/.

[43]. —. Windows Communication Foundation. *MSDN.* [Online] [Citace: 9. duben 2012.] http://msdn.microsoft.com/en-us/netframework/aa663324.

[44]. —. 7 Things to Know About Windows Azure Capacity. *TechNet.* [Online] [Citace: 23. březen 2012.] http://technet.microsoft.com/en-us/cloud/gg663909.

[45]. **Gartner.** Gartner Says Sales of Mobile Devices Grew 5.6 Percent in Third Quarter of 2011; Smartphone Sales Increased 42 Percent. *Gartner Newsroom.* [Online] 15. listopad 2011. [Citace: 4. duben 2012.] http://www.gartner.com/it/page.jsp?id=1848514.

10 Seznam obrázků

11 Seznam tabulek

12 Terminologický slovník

Termín	Zkratka	Význam
NET Framework	.NET	.NET Framework je platforma pro vývoj aplikací, která poskytuje nástroje pro tvorbu, nasazení a provoz desktopových, webových a mobilních aplikací a webových služeb. Skládá se ze dvou hlavních částí: Common Language Runtime (CLR), která poskytuje správu paměti a dalších systémových služeb a rozsáhlé knihovny tříd, která obsahuje osvědčený, znovupoužitelný kód pro všechny hlavní oblasti vývoje aplikací. [37]
Infrastructure as a Service	IaaS	Zákazník si kupuje infrastrukturu (virtualizace, HW, úložiště dat, síťová infrastruktura), na které pak sám nasazuje vlastní OS, middleware a aplikace spolu s jejich daty. [7]
On-Premise		Aplikace nebo služby nasazené a řízené podnikem v jeho vlastních datových centrech. [38]
Platform as a Service	PaaS	Zákazník je zodpovědný pouze za aplikace a jejich data. Vše ostatní (např. infrastruktura, OS nebo běhové prostředí) je zodpovědností poskytovatele služby. Příkladem PaaS je Windows Azure. [7]
Point Of Interest	POI	Bod zájmu. V kontextu této práce a aplikace se jedná o bod na mapě s pevně přiřazenou polohou, který má uživatele upozornit na určitý objekt hodný zájmu na daném místě, například překážku v cestě. [autor]
SCRUM		Agilní metodika vyvinutá pro systémové řízení vývoje. Empirický přístup, který klade důraz na flexibilitu, adaptabilitu a produktivitu. Nedefinuje žádné speciální techniky vývoje softwaru pro implementační fázi. Zaměřuje se na roli jednotlivých členů týmu pro dosažení maximální flexibility v neustále se měnícím prostředí. [39]

Segment (trasy)		Trasy jsou rozděleny na mnoho malých segmentů, které se chovají velmi podobně jako trasy, ale jsou výrazně kratší. Metadata jsou přepočítána. Segmenty jsou používány jako ukazatele náročnosti terénu v okolí. Jsou výhodnější než samotné trasy, protože jsou kratší a mají tedy větší vypovídací hodnotu v lokálním měřítku.
Service Level Agreement	SLA	Service Level Agreement (SLA) je definována jako kontrakt mezi dvěma stranami: Poskytovatelem služby a zákazníkem nebo mezi dvěma poskytovateli služeb. Formálně zachycuje úroveň poskytované služby. [40]
Software as a Service	SaaS	Zákazník si kupuje software jako službu. Je nabízena malá nebo žádná míra přizpůsobení. Zákazník nemá zodpovědnost za konfiguraci a aktualizaci aplikací, operační systém ani hardware. Například Office 365. [7]
SQL Azure		Microsoft SQL Azure Database je relační databázová služba pro prostředí cloudu, která vychází z produktu SQL Server. Zajišťuje vysoce dostupnou, škálovatelnou databázovou službu, která je na multitenantním principu hostována společností Microsoft v prostředí cloudu. [41]
Trasa		Trajektorie, po které se klient pohyboval, zatímco zaznamenával trasu. Skládá se z množiny po sobě jdoucích bodů. Každý bod má polohu, nadmořskou výšku a další údaje. Trasa dále obsahuje metadata jako délku, převýšení, datum a čas zahájení a ukončení atd. [autor]
Windows Azure		Windows Azure je operační systém pro služby typu „Cloud Computing“, který slouží jako prostředí pro vývoj, hostování a správu služeb na platformě Windows Azure. Windows Azure je postaven na operačním systému Windows Server 2008 64 bit. [42]

Windows Azure Platform		Platforma Windows Azure je platforma pro služby technologie Cloud Computing hostované v datacentrech společnosti Microsoft. Platforma Windows Azure poskytuje celou škálu funkcí pro vytváření aplikací, jejichž použití sahá od spotřebitelských webů až po nasazení v podnicích, a zahrnuje operační systém pro aplikace běžící v prostředí cloudu a sadu služeb pro vývojáře. [42]
Windows Communication Foundation	WCF	Část .NET Frameworku, která poskytuje jednotný programovací model pro rychlý vývoj aplikací orientovaných na služby, které komunikují přes web a podnikem. [43]
Windows Phone 7	WP7	Operační systém pro mobilní zařízení vyvinutý společností Microsoft. Nástupce OS Windows Mobile, s tímto systémem však není zpětně kompatibilní. [autor]

13 Přílohy

- Elektronické přílohy
    - Zdrojové kódy
    - Zabezpečení komunikace
        - AccountValidator.cs
        - Web.config
        - ServiceDefinition.csdef
        - ServiceConfiguration.csfg