České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Adam Lysák**

Studijní program: Otevřená informatika
Obor: Softwarové inženýrství

Název tématu: **Systém pro vzdálený monitoring a konfiguraci škálovatelných služeb**

Pokyny pro vypracování:

Navrhněte a implementujte systém pro vzdálený monitoring horizontálně škálovatelných služeb. V rámci monitoringu navrhněte různé metriky a klasifikátory pro určování aktuálního stavu jednotlivých služeb.
Počítejte s dynamicky se měnícím stavem jednotlivých služeb a přibývajícím či ubývajícím počtem instancí. Zaměřte se zejména na služby běžící na platformě .NET, zaměřte se však také na budoucí rozšiřitelnost systému na ostatní platformy.
V rámci systému umožněte změnu konfigurace jednotlivých služeb za běhu. Pro systém implementujte webové rozhraní, které bude visualizovat různé metriky jednotlivých instancí služeb. Dále implementujte klienta pro libovolnou mobilní platformu, která bude obsahovat zjednodušenou formu ovládání jednotlivých služeb a zjednodušenou visualizaci hlavních monitorovaných metrik.

Seznam odborné literatury:

[1] Flanders, Jon, 2008, RESTful .NET. "O'Reilly Media, Inc." ISBN-13:978-0-596-51920-9
[2] Nagel, Christian et el, 2014, Professional C# 5.0 and .NET 4.5.1. John Wiley & Sons. ISBN-13:978-1118833032
[3] Troelsen, Andrew, 2012, Pro C# 5.0 and the .NET 4.5 Framework. Apress. ISBN-13: 978-1430242338
[4] Abbott, Martin L. and Fisher, Michael T., 2011, Scalability Rules. Addison-Wesley Professional. ISBN-13:978-0321753885
[5] Introduction to Windows Communication Foundation, URL: https://msdn.microsoft.com/en-us/library/dd936243.aspx

Vedoucí: Ing. Martin Mudra

Platnost zadání: do konce letního semestru 2016/2017

[redacted]
doc. Ing. Filip Železný, Ph.D.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
LS

[redacted]
prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 23. 11. 2015

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů

Diplomová práce

Systém pro vzdálený monitoring a konfiguraci škálovatelných služeb

Bc. Adam Lysák

Vedoucí práce: Ing. Martin Mudra

5. ledna 2017

Poděkování

Děkuji svému vedoucímu práce Ing. Martinu Mudrovi za všechny poskytnuté rady a připomínky.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 5. ledna 2017 .....

Abstract

This diploma thesis deals with design and imlementation of system for remote monitoring and configuring scalable services. This work focuses on problematics of horizontal scalable services, which are nowadays frequently used in cloud environment. Horizontal scalable services are services in which may exist a very large number of parallel instances running during same time and their number may very dynamically change over time. Main goal is to create reliable system for monitoring and configuring such services, also create libraries for integration into services and client application for user management.

Abstrakt

Tato diplomová práce se zabývá návrhem a implementací systému pro vzdálený monitoring a konfiguraci škálovatelných služeb. Práce se zaměřuje na problematiku monitoringu horizontálně škálovatelných služeb, které se v dnešní době hojně vyskytují především v cloudovém prostředí. Jedná se o služby, u kterých může existovat velké množství paralelně běžících instancí a jejich počet se může v čase dynamicky měnit. Cílem je vytvořit serverový systém pro monitoring takovýchto služeb, knihovny pro integraci do monitorovaných služeb a klientské aplikace pro uživatelskou správu.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V poslední době se velmi rozmáhá využívání webových cloudových služeb. Především je velmi časté nasazení vlastních služeb do prostředí jako Microsoft Azure, Google AppEngine, Amazon Web Services, apod. Hlavním důvodem může být široká nabídka poskytovatelů také a náklady, která mohou být nižší než při nasazení služby na vlastním hardwaru.

Z nasazení služby do cloudového prostředí plyne řada výhod. Prvním je dynamické přiřazení prostředků na základně aktuálního vytížení služby. V případě, že služba není využívána žádným klientem, pak běží s minimální zdroji (např.: v jedné instanci na jednom procesoru) nebo je dokonce zcela pozastavena do doby, než na ni vznikne nějaký požadavek od klienta. Naopak v případě, kdy služba musí být ve špičkách poskytována maximální počtu klientů, je nutné ji přiřadit patřičné prostředky, aby byla schopna všechny požadavky od klientů obsloužit.

Škálování prostředků v cloudovém prostředí je řešeno zvyšováním nebo snižováním počtů instancí služby. Instancí se myslí běžící služba se všemi svými knihovnami a závislostmi, přičemž jednotlivé instance služby jsou mezi sebou zcela nezávislé a mohou také běžet na zcela různých serverech nebo umístění. Rozložení zátěže mezi jednotlivé instance následně prování cloudové prostředí přes tzv. load balancer. Takovéto služby nazýváme horizontálně škálovatelné.

Výhodou je, že zákazník platí pouze za prostředky, které jeho služba využije. Tedy v případě, že služba není využívána nebo běží s minimálními zdroji, tak je cena minimální a naopak v případě kdy služba běží ve špičce. Kdyby zákazník provozoval službu na vlastním hardwaru, mohl by mu zůstávat potenciální nevyužitý hardware v situacích kdy by nebyla služba využívaná. V případě cloudového prostředí zákazník přesně platí za prostředky, které využije.

Zajímavou disciplínou je monitorování a správa takovýchto cloudových služeb. U této disciplíny je nutné klást důraz právě dynamičnost prostředí, kde může ve velmi krátkém časovém horizontu vznikat nebo zanikat velké množství instancí služby. Je tedy nutné zajistit sběr a propagaci všech údajů z těchto běžících instancí, aby bylo možné provádět jejich správu.

Cíl práce

Cílem této práce je navrhnout a implementovat systém, který umožní monitorovat a konfigurovat tzv. horizontálně škálovatelné služby. Systém bude počítat s tím, že monitorované služby mohou běžet ve více instancích a tyto instance mohou v průběhu vznikat a zanikat.

Součástí řešení bude serverová část, která bude centrální webovou službou pro monitorování a konfiguraci. Bude také vytvořen klientský framework pro snadnou integraci do monitorovatelných služeb. Pro možnost interakce uživatele se systémem bude vytvořena webová aplikace a klienti pro dnes nejpoužívanější mobilní platformy Windows, Android a iOS.

Kapitola 1

Analýza

Součástí této kapitoly je definice měřitelný metrik. Dále následuje specifikace všech funkčních a nefunkčních požadavků na systém. Následně je v kapitole obsažena analýza existujících řešení, které jsou funkčně podobné navrhovanému řešení. Nakonec je provedena analýza technologií použitelných při návrhu a implementaci.

1.1 Měřitelné metriky

Jednou z funkcionalit systémů má být monitoring horizontálně škálovatelných služeb. Pod pojmem monitoring si lze, představit měření sady různých metrik (hodnot) služby, přičemž služby mohou běžet ve více instancích na různých běhových prostředích nebo fyzických zařízeních. Z toho plyne nutnost měřit všechny metriky pro každou instanci služby zvlášť. Měřené metriky se mohou týkat jak stavu samotných služeb, tak běhového prostředí (serveru), pod kterým instance běží. Mezi hlavní metriky především patří:

Vytížení CPU Vytížení CPU běhové prostředí, na kterém běží monitorovaná instance služby.

Využití paměti Měření využití fyzické nebo virtuální paměti běhové prostředí, na kterém běží monitorovaná instance služby.

Diskové I/O Měření využití pevných disků běhové prostředí, na kterém běží monitorovaná instance služby.

Síťové I/O Měření využití síťového datového provozu běhové prostředí, na kterém běží monitorovaná instance služby.

Počet zpracovaných požadavků Počet síťových požadavků, které instance služby zpracovala v časovém období.

Doba odpovědi Doba trvání zpracování síťového požadavku instance služby.

1.1.0.1 Agregované metriky

Zároveň by systém měl poskytovat možnost pracovat s ucelenými metrikami jednotlivých služeb. Vzhledem k faktu, že metriky jsou monitorovány pouze nad samotnými instancemi, je nutné v systému zajistit mechanizmus pro přepočet ze všech naměřených dat metrik instancí na ucelenou metriku služby. Tento přepočet je často řešen jako průměr z hodnot metrik všech instancí za pevný časový interval.

1.1.0.2 Aplikační logy

Dnešní aplikace během svého běhu generují množství záznamů o událostech (logy). Tyto záznamy mohou být pro monitorování stavu služeb velmi podstatné a proto by bylo vhodné je v systému sbírat.

1.2 Analýza požadavků

V této části práce je uveden seznam funkčním a nefunkčních požadavků navrhovaného systému. Požadavky se týkají jak serverové části, tak i klientského frameworku a klientských aplikací.

1.2.1 Funkční požadavky

Na systém jsou kladeny tyto funkční požadavky:

- Systém musí být schopen monitorovat vybrané metriky nad instancemi služeb.
- Systém musí být schopen měřit uživatelem definované vlastní metriky nad instancemi služeb.
- Systém musí být schopen výpočtu agregovaných metrik jednotlivých služeb.
- Systém musí být schopen ukládat logy monitorovaných služeb.
- Systém musí být schopen vzdálené změny konfigurace monitorovaných služeb.

- Systém musí být schopen řídit (startovat, zastavovat) monitorované služby a jejich instance.
- Systém bude obsahovat klientský framework pro snadnou integraci do monitorovaných služeb.
- Klientské mobilní aplikace musí umět vizualizovat a reprezentovat naměřené metriky a logy.
- Klientská webová aplikace musí být schopna vizualizovat a reprezentovat naměřené metriky a logy. Zároveň bude také spravovat vzdálenou konfiguraci monitorovaných služeb.

1.2.2 Nefunkční požadavky

Na systém jsou kladeny tyto nefunkční požadavky:

- Serverová část musí být horizontálně škálovatelná, aby byl schopna obsluhovat větší množství monitorovaných služeb a klientských aplikací.
- Systém by měl být navržen s vícevrstvou architekturou.
- Systém by měl být jednoduše konfigurovatelný.
- Serverová část by měla být schopna pracovat s různými relačních databázemi.
- Klientské mobilní aplikace poběží na mobilních platformách Android, iOS, Windows.
- Klientské aplikace by měly být jednoduše uživatelsky použitelné.

1.2.3 Negativní vymezení

V rámci této práce nebude řešeno zabezpečení systému nebo autorizace uživatelů. Neboli veškerá komunikace bude probíhat nezabezpečeně. Zároveň v systému nebude implementována žádná logika pro podporu více uživatelských účtů, takže systém bude operovat nad tzv. jedním účtem a nebude vyžadovat žádnou autorizaci tohoto uživatele.

1.3 Existující řešení

V současné době na trhu existuje celá řada produktů umožňující určitou formu správy a monitoringu horizontálně škálovatelných služeb. Existujícími systémy, které se nejvíce blížili stanoveným požadavkům, jsou Microsost Azure Monitoring, Amazon CloudWatch, Nagios, PRTG Network Monitor. Služby Microsoft Azure Monitoring a Amazon CloudWatch nabízí velice podobnou funkcionalitu. Bohužel tyto služby nelze využít k monitorování jiných cloudových platforem.

1.3.1 Microsoft Azure Monitoring

Microsoft Azure je cloudová platforma vyvíjená společností Microsoft umožňující nasazení webových služeb a aplikací. Na Microsoft Azure lze provozovat tyto typy aplikací[4]:

Web Role Možnost provozovat webové služby a aplikace běžící v Microsoft IIS (Internet Information Services).

Woker Role Možnost provozovat asynchronní dlouhotrvající úlohy nezávislé na uživatelském vstupu.

SQL Database Možnost provozovat databáze běžící pod Microsoft SQL Server.

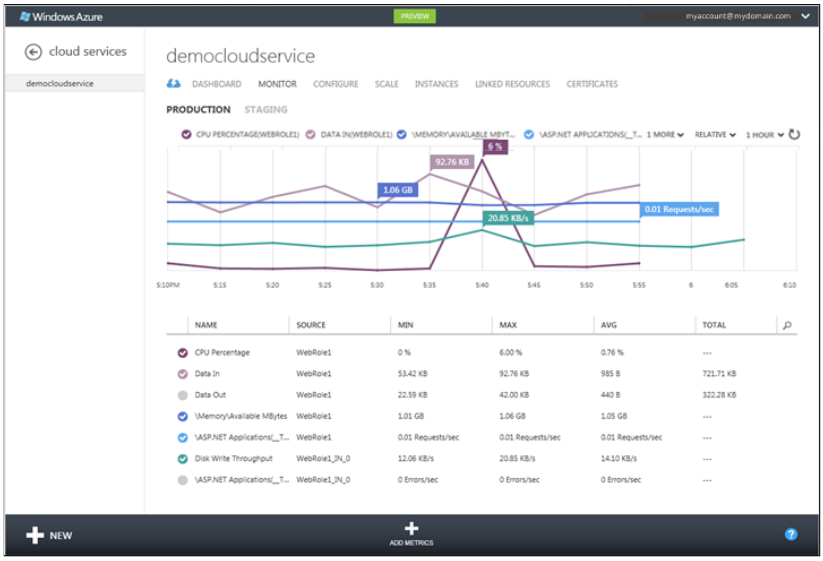
BLOB uložiště Možnost provozovat BLOB (Binary Large OBject) úložiště, což je speciální úložiště kde je možné uložit textová nebo binární data. Takto uložená data jsou pak dostupná přes speciální URL (Uniform Resource Locator) adresy.

Součástí Microsoft Azure je monitorovací systém nazývající se Microsoft Azure Monitoring. Lze v něm monitorovat služby běžící pouze v Microsoft Azure a sledovat velké množství metrik[14]. Mezi důležité metriky patří:

- Vytížení CPU
- Využití paměti
- Diskové I/O
- Síťové I/O
- Počet zpracovaných požadavků
- Doba odpovědi

Systém podporuje jak monitoring samotných služeb, tak i jednotlivých instancí služby. Navíc je možné do monitorovaných služeb na úrovni zdrojového kódu integrovat knihovnu Application Insights[17], pomocí níž je následně možné monitorovat vlastní metriky a události.

Nasazené služby je také možné v prostředí Microsoft Azure dále spravovat a konfigurovat.

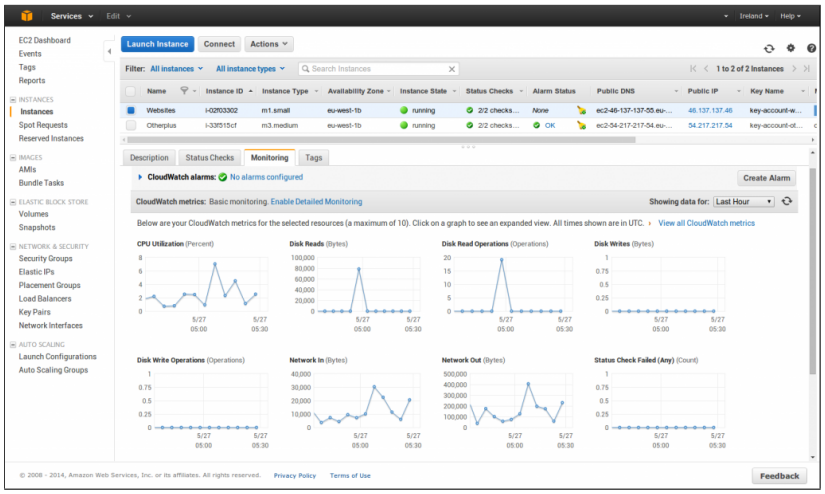


Obrázek 1.1: Ukázka hlavní obrazovky z prostředí Azure Monitoring[37]

1.3.2 Amazon CloudWatch

Amazon Cloudwatch[2] je systém vyvíjený společností Amazon zaměřující se pouze na monitoring služeb běžících pod Amazon Web Services. Systém je dostupný skrze webové rozhraní. Díky němu je možné monitorovat jednotlivé instance nasazených služeb. V systému lze sledovat předefinované metriky nebo definovat vlastní. Mezi důležité podporované metriky patří[3]:

- Vytížení CPU
- Využití paměti
- Diskové I/O
- Síťové I/O
- Počet zpracovaných požadavků



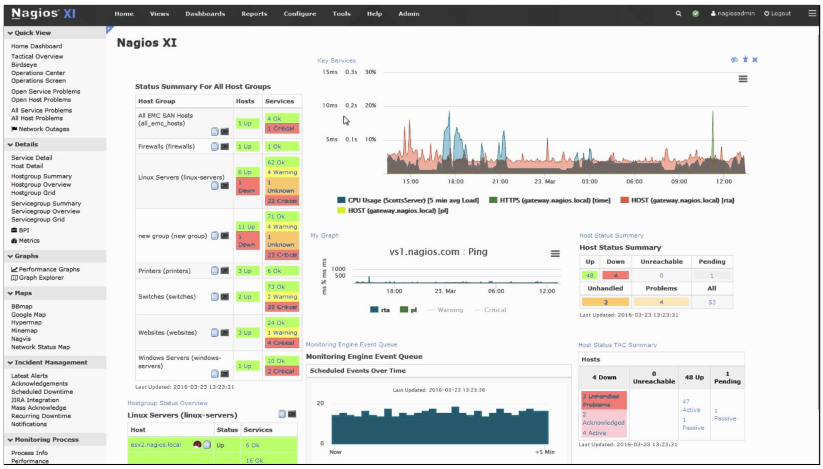
Obrázek 1.2: Ukázka hlavní obrazovky z prostředí Amazon CloudWatch[16]

1.3.3 Nagios

Nagios je sytém pro monitorování síťové infrastruktury (síťového hardwaru, serverů) a služeb. Základní Nagios Core je open source[25] a lze ho zdarma provozovat na většině Linuxových distribucí. Existuje několik dalších placených edicí Nagios Core přidávající další funkcionalitu. Zároveň také stejnojmenná společnost Nagios Enterprises provozuje vlastní placený produkt Nagios XI[26] v několika edicích, kterým přidává další funkcionalitu a podporu produktu. Samotný Nagios je modulární a lze do něj přidávat další funkcionalitu formou balíčků. Nagios se spíše zaměřuje na monitoring služeb a nelze v něm monitorovat jednotlivé instance. V systému Nagios jsou metriky reprezentovány pluginy. Existuje jejich rozsáhlá databáze[27]. Dostupnost jednotlivých metrik se také odvíjí od typu monitorované služby nebo zařízení. V databázi pluginů je možné najít tyto základní metriky:

- Vytížení CPU
- Využití paměti
- Diskové I/O
- Síťové I/O

V systému Nagios není možné monitorové služby spravovat a konfigurovat.

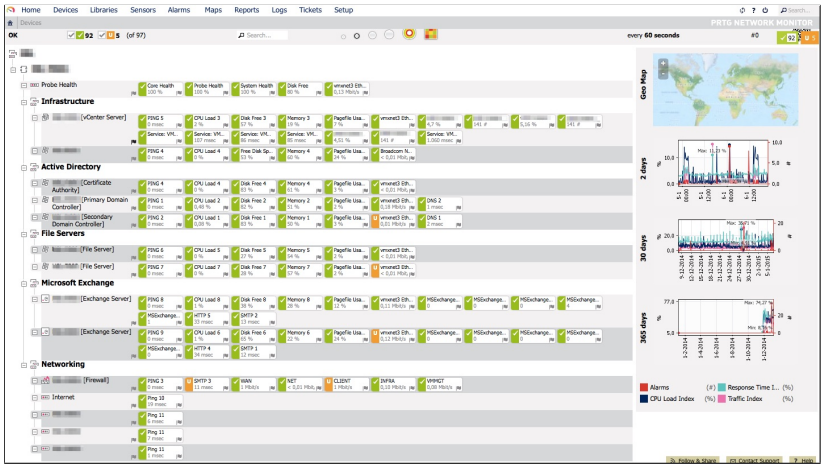


Obrázek 1.3: Ukázka hlavní obrazovky z prostředí Nagios XI[26]

1.3.4 PRTG Network Monitor

PRTG Network Monitor stejně jako Nagios je především určen k monitorování síťové infrastruktury[34]. Veškeré monitorování je prováděno skrze standardní síťové protokoly, kterými jsou SNMP, WMI, PING, atd. PRTG Network Monitor je placený produkt, ale obsahuje i freeware verzi[33] s omezeným počtem monitorovaných metrik zároveň. PRTG Network Monitor je distribuován jako softwarové řešení. Každý zákazník si musí nasadit vlastní verzi PRTG serveru. V systému lze sledovat různé metriky[32] nebo také definovat vlastní. V PRTG se metriky nazývají sensory. Mezi základní podporované metriky patří:

- Vytížení CPU
- Využití paměti
- Diskové I/O
- Síťové I/O
- Doba odpovědi



Obrázek 1.4: Ukázka hlavní obrazovky z prostředí PRTG Network Monitor[36]

1.4 Frameworky pro vývoj mobilních aplikací

Jedním z požadavků je implementace klientů pro mobilní platformy Android, iOS a Windows. Existuje několik možností jak lze mobilní aplikace implementovat. Nepřímočařejší je implementace separátní aplikaci pro každou platformu s použitím vývojářským nástrojů a doporučení konkrétní platformy. Zde nastává problém kdy vývoj každé platformy (Android, iOS, Windows) probíhá v jiné platformě a programovacím jazyku. Pro některé z platforem je dokonce nutné používat i určený operační systém pro vývoj.

Android Vývoj probíhá v jazyce Java.

iOS Vývoj probíhá v jazyce Objective C nebo Swift a vývojářské nástroje jsou dostupné pouze pro operační systém macOS (nazývaný též OS X)[38].

Windows Vývoj probíhá na platformě .NET (jazyk C# nebo Visual Basic) přičemž vývojářské nástroje jsou dostupné pouze pro operační systém Microsoft Windows[21].

Každá z platforem navíc obsahuje zcela rozdílné mobilní API (Application Programming Interface). Tímto API se myslí sada knihoven, která je poskytnuta operačním systémem zařízení. Toto API následně aplikace využívá k interakci s operačním systémem a hardwarem mobilního zařízení.

Vývoj separátních aplikací pro každou platformu může být časově náročný. Zde se tedy nabízí otázka jestli neexistují další možnosti jak vyvinout mobilní aplikace pro všechny tři platformy nejlépe v jednom programovacím jazyku nebo i platformě. V současně době existuje celá řada frameworků, které přesně tento přístup k vývoji umožňují.

1.4.1 Apache Cordova

Apache Cordova je framework pro vývoj mobilní aplikací, který je založen na technologiích HTML, CSS a javascript. Výsledné aplikace vytvořené tímto frameworkem jsou v mobilní zařízení zobrazovány jako webové stránky uvnitř komponenty webového prohlížeče[7]. Přes speciální Javascript API, které Cordova poskytuje je následně možné využívat mobilní API a hardware mobilního zařízení (kamera, GPS, telefonní modul, atd.). Toto Javascriptové API je řešené formou pluginů, které jsou psány nativně pro každou mobilní platformu. Je tedy možné psát i vlastní pluginy. V současné době jsou podporovány platformy Android, iOS, Windows a další méně známe platformy. Framework je vyvíjen komunitou a je open source dostupný pod licencí Apache Licence 2.0.

Výhody:

- Vývoj v HTML, CSS, Javascript, což jsou známe a rozšířené technologie a tedy aplikace může vyvíjet webový vývojář.
- Jednotný kód pro všechny mobilní platformy.

Nevýhody:

- Použití webových technologií, při kterých aplikace musí běžet uvnitř webové prohlížeče zařízení a nevyužívá plný potenciál zařízení. Takové aplikace jsou mnohem pomalejší než aplikace psané nativním přístupem.
- Aplikační logika psaná v Javascriptu, který není ideálním jazykem pro vývoj velkých aplikací.

1.4.2 PhoneGap

PhoneGap je framework patřící společností Adobe Systems. PhoneGap je celý postaven nad frameworkem Apache Cordova[1]. V současné chvíli PhoneGap nenabízí žádnou další podstatnou funkcionalitu vůči Apache Cor-

doba. Framework je také open source dostupný pod licencí Apache Licence 2.0. Platí u něj stejné výhody i nevýhody jako u Apache Cordoba.

1.4.3 Trigger.io

Trigger.io je další zástupce HTML/CSS/Javascript frameworku. Jedná se placený produkt, který neposkytuje žádnou verzi zdarma. Principiálně funguje stejně jako Apache Cordova nebo PhoneGap, ale na rozdíl od nich implementuje vlastní mechanizmus pro konverzi z HTML do mobilní aplikace[41]. Výhodou tohoto vlastního mechanizmu je až pětinásobné zrychlení[40] finálních mobilních aplikací vůči Apache Cordova. V současné době jsou podporovány platformy Android, iOS a Windows.

Výhody:

- Použití HTML/CSS/Javascriptu a z toho plynoucí stejné výhody jako u Apache Cordova.
- Finální mobilní aplikace jsou mnohonásobně rychlejší než u Apache Cordova.
- Jednotný kód pro všechny mobilní platformy.

Nevýhody:

- Použití HTML/CSS/Javascriptu a z toho plynoucí stejné nevýhody jako u Apache Cordova.
- Placený produkt.

1.4.4 React Native

React Native je framework od společnosti Facebook. Aplikace jsou vyvíjeny pouze v jazyku Javascript, ve kterém je napsána jak aplikační business logika, tak i uživatelské rozhraní. Framework obsahuje sadu vlastních grafických uživatelských komponent, které využívají nativní komponenty jednotlivých mobilních platforem[9]. Tento přístup umožňuje, že mobilní aplikace psané v React Native vypadají po stránce uživatelské rozhraní zcela totožně jako nativní aplikace. V současné době React Native podporuje pouze vývoj aplikací pro platformy iOS a Android.

Výhody:

- Použití Javascriptu, což je známá a rozšířená technologie.
- Výsledné aplikace jsou rychlejší než u frameworků postavených pouze nad HTML/CSS/Javascript.
- Jednotný kód pro všechny mobilní platformy.

Nevýhody:

- Stále znatelný výkonový propad vůči nativním aplikacím.
- Nepodporuje vývoj pro platformu Windows.

1.4.5 Xamarin

Xamarin je framework vyvíjen společností xamarin Incorporated[46], která je dceřinou společností Microsoft Corporation. Xamarin je dostupný v několika edicích, přičemž je dostupný i v edici Community, která je k dispozici zdarma[47] pod proprietární licencí Microsoft Software license[23].

Vývoj aplikací v Xamarinu probíhá na platformě Microsoft .NET Framework a v programovacím jazyce C#. V Xamarinu je možné k vývoji mobilní aplikace přistoupit dvěma způsoby, a to buď navrhnout separátní aplikaci pro každou platformu nebo použít tzv. Xamarin Forms.

1.4.5.1 Separátní aplikace pro každou platformu

Při použití toho přístupu je nutné vyvinout pro každou platformu separátní aplikaci. Struktura jednotlivých projektů a zdrojových kódů zůstává velmi podobná, jako kdyby byl použit nativní vývoj aplikace. Vývojář pouze místo originální programovací platformy a jazyka používá .NET Framework s jazykem C#. Návrh uživatelského rozhraní pro každou z platforem zůstává stejný jako v případě vývoje nativní aplikace. S využitím tohoto přístupu může vývojář využít plný potenciál každé z platforem.

Výhody:

- Ideální pro vývoj aplikací s rozsáhlou business logikou, kterou lze napsat pouze jednou pro všechny mobilní platformy.
- Lze využít plný potenciál každé platformy.
- Rychlost aplikací (nepatrně pomalejší než nativní aplikace).

Nevýhody:

- Nutné psát separátní aplikace pro každou platformu.

1.4.5.2 Xamarin Forms

Xamarin Forms je rozšíření stávajícího Xamarinu umožňující napsat jednu implementaci aplikace, která následně poběží na všech platformách[45]. Stačí tedy napsat celou aplikační logiku a zároveň jednou implementaci uživatelského rozhraní. Aplikace se při kompilaci následně pro každou z platforem automaticky přizpůsobí, aby vypadala podle design guidlines každé platformy. Xamarin Forms implementuje vlastní sadu uživatelských komponent, které nabízí proti nativnímu vývoji pouze omezenou sadu možností. Je možné implementovat vlastní komponenty kdy je nutné napsat implementaci pro každou platformu a pak je možné danou komponentu použít v Xamarin Forms.

Výhody:

- Jednotný kód pro všechny mobilní platformy.
- Rychlost aplikací (nepatrně pomalejší než nativní aplikace).

Nevýhody:

- Omezené možnosti uživatelského rozhraní kvůli omezenému počtu komponent a jejich možností. Velmi špatně použitelné pro aplikace, kde se může uživatelské rozhraní na každé platformě zásadně lišit.

1.5 Technická řešení

Vzhledem k nutnosti implementace serverové části, integračního frameworku a klientské aplikace by bylo vhodné k vývoji zvolit platformu, která umožňuje vývoj všech těchto částí v jednom programovacím jazyce. Proto byl zvolen Microsoft .NET Framework a programovací jazyk C#. Jednotlivé části systému následně je možné realizovat s využitím těchto technologií:

Serverová část K implementaci lze využít ASP .NET Web API nebo WCF.

Webový klient Pro vývoj webových aplikací .NET nabízí ASP .NET MVC nebo ASP .NET WebForms.

Mobilní klienti K vývoji mobilních aplikací lze použít Xamarin.

1.5.1 WCF

WCF (Windows Communication Foundation) je sadou knihoven prostředí .NET Framework, které umožňují implementaci aplikacích zaměňujících se na webové služby. WCF dovoluje zasílat asynchronií zprávy mezi službou a klienty různými podporovanými protokoly (TCP, HTTP, SOAP, REST, atd.) s různou úrovní zabezpečení (SSL, HTTPS, atd.)[24]. Ze strany vývojáře stačí definovat rozhraní služby a implementovat business logiku těchto služeb. WCF na základě definovaného rozhraní služby a konfigurace obsahující typ protokolu a zabezpečení je následně schopna automaticky vytvořit serverovou i klientskou část služby. V implementaci aplikace stačí pouze znát rozhraní služby, použitý protokol a endpoint pro připojení klienta (URI). Služby využívající WCF mohou být nasazeny ve dvou režimech[22]:

Self hosting Režim kdy aplikace obsahuje veškeré knihovny pro běh WCF služby. Může se tedy jednat o soběstačnou Windows aplikaci nebo Windows službu.

IIS hosted Režim kdy je služba spuštěna v serverovém prostředí Microsoft IIS.

1.5.2 ASP .NET

ASP .NET je technologie od společnosti Microsoft postavená nad .NET Framework určená k vývoji webových aplikací a služeb. ASP .NET lze rozdělit na tři hlavní části. Přičemž je důležité, že veškeré aplikace vyvinuté ASP .NET lze provozovat pouze v prostředí Microsoft IIS.

1.5.2.1 ASP .NET Web API

ASP .NET Web API je technologie určená k vývoji HTTP nebo REST webových služeb[19]. Primárně se počítá s použitím datového formátu JSON po výměnu dat mezi službou a klientem, ale není problém použít jiný datový formát jako např. XML.

1.5.2.2 ASP .NET WebForms

Vývoj webových aplikací v ASP .NET WebForms probíhá na podobném principu jako vývoj okenních Windows aplikací. Je dostupná celá řada komponent a prvků uživatelského rozhraní[20]. Při následném zobrazení stránek psaných ve WebForms dochází k automatickému převodu těchto stránek do HTML/CSS/Javascript. V dnešní době se již jedná o starší technologii a budoucí podpora ze strany společnosti Microsoft je nejistá.

1.5.2.3 ASP .NET MVC

ASP .NET MVCje nástupcem starších WebForms. Narozdíl od WebForm používá návrhový vzor MVC, který se snaží oddělit business logiku aplikace od uživatelského rozhraní[18]. Business logika je psaná v .NET Frameworku a programovacím jazyku C# nebo Visual Basic. Veškeré uživatelské rozhraní je naopak psano v HTML/CSS/Javascript.

1.5.3 Inversion of Control

Inversion of Control (IoC) je návrhový vzor pro tvorbu aplikací, který zjednodušuje závislost jednotlivých komponent aplikace mezi sebou. IoC se snaží porušit obecné pravidlo v objektovém programování, kdy komponenta, která závisí na jiné komponentě musí znát její implementaci (metody, konstruktory, apod.). Především je nutné při vytváření instance komponenty kdy musí být volán konstruktor. Princip IoC spočívá v tom, že veškeré komponenty implementují veřejné rozhraní obsahující definice metod[10]. Následně všechny komponenty pracují pouze s těmito rozhraními. Vzniká tu jeden problém. V těchto rozhraních není nijak definováno jak danou komponentu vytvořit nebo získat její instanci. K tomuto účelu v IoC slouží tzv. Container nebo jinak řečeno Locator. Locator je komponenta zcela nezávislá na ostatních komponentách systému, která se stará o vytváření komponent a správu jejich životního cyklu. Locator využívá techniku Dependency Injection (DI) pro řešení závislostí mezi komponentami. Znamená to, že IoC si za běhu programu tvoří grafu reprezentující závislosti mezi jednotlivými komponentami. Následně podle tohoto grafu injektuje potřebné komponenty.

Výhodou IoC je, že je možné přes Locator proces vytváření komponent konfigurovat. Lze tedy velice snadno vyměnit implementaci jedné komponenty za druhou. A to pouhou změnou konfigurace Locatoru je možné upravit chování celé aplikace. Tento přístup má své výhody například u automatizovaného testování kódu, kdy mohou být pouhou změnou konfigurace nahrazeny komponenty svými mockovanými alternativami.

Nevýhodou IoC je, že může při větším počtu komponent zpomalovat start aplikace. Při každém startu je nutné vytvořit graf závislosti komponent, což je časově náročný proces. U aplikací kde není start aplikace kritickým prvkem je použití IoC vhodné. Dále také mohou vznikat mezi komponentami tzv. cyklické závislosti. Naštěstí tyto závislosti je většina IoC frameworku schopna detekovat a upozornit na ně.

1.5.4 Databáze

K uložení naměřených dat metrik bude pravděpodobně vhodné použít nějakou relační databázi. Ve spojitosti s technologií .NET Framework se nabízí použití relační databáze Microsoft SQL Server, který je v .NET Frameworku a většině ORM frameworcích široce podporován. Zároveň bude nutné použít databázi i na klientské části aplikace, kde není možné využít žádné relační databáze, které vyžaduji vlastní běhové prostředí nebo instalaci. V tomto případě bude použita databáze SQLite, která je dostupná skrze programovou knihovnu a nevyžaduje žádné prostředí nebo instalaci[39]. Databáze v SQLite je tvořena jedním souborem.

Object-relational mapping (ORM) je technika, při které jsou databázové záznamy mapovány na objekty v programovacím jazyce[44]. Veškerá následná manipulace s databází v aplikaci probíhá zpravidla pouze přes tyto objekty. Pro .NET existují dva velké ORM frameworky NHibernate a Entity Framework.

1.5.4.1 NHibernate

NHibernate je ORM framework dostupný pod licencí Lesser GNU Public License (LGPL), version 2.1[11]. Jedná se o portovanou originální knihovnu Hibernate z programovacího jazyka Java[30].

Výhody:

- Dvou-úrovňová cache entit.
- Velice jednoduché použití transakcí v kombinaci s atributy metod.

Nevýhody:

- Nepodporuje automatické migrace databáze.

1.5.4.2 Entity Framework

Entity Framework je dostupný pod licencí Apache License, Version 2.0 a je vyvíjen organizací .NET Foundation[28].

Výhody:

- Automatické migrace databáze. Podpora automatické úpravy schématu databáze na základě úprav v objektovém modelu DB v implementaci aplikace.
- Integrace nástrojů ve vývojovém prostředí Visual Studio.

Nevýhody:

- Automaticky generované hodnoty primární klíče mohou být pouze číselného typu.
- Omezená podpora databáze SQLite (není dostupné automatické generování schématu DB a migrace).
- Velice omezená podpora dědičnosti databázových entit.

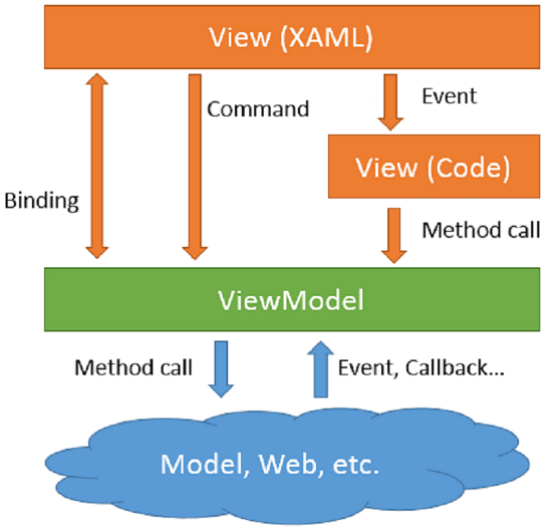
1.5.5 MVVM

Pro vývoj aplikací mobilních klientů se nabízí použití návrhového vzoru MVVM (Model-View-ViewModel), který vychází z návrhového vzoru MVC a odděluje uživatelské rozhraní od business logiky[42]. Je toho docíleno rozdělením aplikace do několika vrstev. Výhodou tohoto rozdělení je následná možnost úpravy uživatelské rozhraní aplikace bez zásahů do logiky aplikace. Zároveň je možné uživatelské rozhraní nahradit zcela jiným pro jinou platformu. V případě mobilních aplikací je toho možné využít vytvořením společné logiky pro všechny mobilní platformy a následně pouze rozdílné uživatelské rozhraní podle mobilních platforem. Další výhodou je usnadnění testovaní kdy lze snadno automatizovaně testovat business logiku, protože ta není nijak závislá na uživatelském rozhraní. MVVM rozděluje aplikaci na tři části:

Model Reprezentuje data aplikace. Může se například jednat databázi nebo objekty reprezentující komunikační zprávy.

View Reprezentuje uživatelské rozhraní.

ViewModel Reprezentuje mezivrstvou mezi Model a View. Jeho úkolem je poskytovat data z Modelu v takové podobě, aby je mohl View jednoduše zobrazit. Musí také reagovat na uživatelský vstup, který je zasílán z View. Zároveň udržuje veškerý stav aplikace.



Obrázek 1.5: Struktura aplikace při použití návrhového vzoru MVVM[15]

1.5.6 Logovací framework

Součástí funkčních požadavků je i funkcionalita logování. Logování je mechanizmus umožňující záznam textových událostí z instancí služeb a následnou možnost manipulace s těmito logy v klientské aplikaci. Zde je velice důležité napojení logované služby na klientský framework. V dnešních .NET aplikacích se velmi často v k záznamů logů využívají frameworky nebo knihovny třetích stran. Klientský framework by tedy měl podporovat napojení na nejrozšířenější z těchto frameworků. Proto, aby bylo toto napojení možné je nutné, aby logovací frameworky podporovali rozšířitelnost.

1.5.6.1 log4net

log4net je framework vyvíjený Apache Software Fountation a je dostupný pod stejnojmennou licencí Apache License 2.0. Jedná se o port původního log4j frameworku, který byl určen pro programovací jazyk Java. Framework nabízí velmi širokou škálu možností volby výstupu (do souborů, Windows Events, po síti přes HTTP nebo UDP, atd.)[6]. Zároveň nabízí i jednoduchou rozšířitelnost a možnost implementovat nové typy výstupů. Framework je plně konfigurovatelný jak v kódu aplikace, tak i přes separátní konfigurační xml soubor.

1.5.6.2 NLog

NLog je framework vyvíjený komunitou a dostupný pod licencí BSD[13]. Funkcionálně nabízí v podstatě totéž jak log4net (široká škála výstupů, rozšiřitelnost a konfigurovatelnost)[12]. NLog na rozdíl od log4net je v posledních letech ve velkém rozvoji a často jsou vydávány nové verze. Z toho důvodu je vývojáři oblíbený.

1.5.7 Message broker

Message brokem je programový modul, který mezi jednotlivými vrstvami softwaru zprostředkovává zasílání zpráv. K message brokeru přistupují dvě role. První je posluchač, který se registruje k odběru zpráv. Při registraci může specifikovat na základně konfigurace a filtrů jaký typ zpráv chce přijímat. Druhou rolí je odesílatel, který naopak produkuje zprávy pro příjemce. Úlohou message brokeru je právě propojení mezi odesílateli a příjemci. Mezi hlavní a volně dostupné řešení patří ActiveMQ[5] a RabbitMQ[35].

Kapitola 2

Návrh

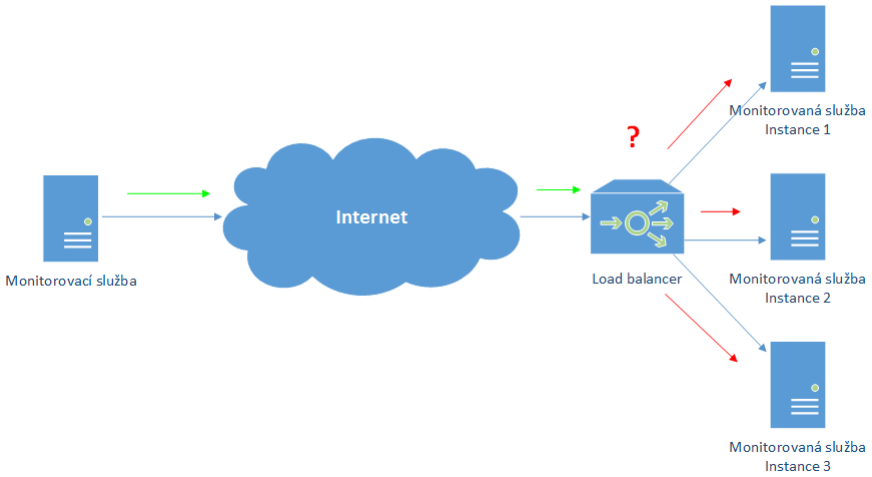
V kapitole návrh je popsána problematika komunikace s horizontálně škálovatelnými službami. Následně pokračuje podrobný popis navržené architektury systému včetně návrhu databází. V poslední části je popsána problematika distribuce událostí monitorovaným službám, vzdálená konfigurace a integrace vlastních metrik.

2.1 Komunikace s instancemi služby

Systém má být schopen monitorovat služby a jejich jednotlivé instance. S tím souvisí problematika vzájemné komunikace mezi monitorovanými instancemi a serverovou části systému (monitorovací službou). Je důležité zmínit, jak funguje komunikace v horizontálně škálovatelných systémem. Tyto systémy obsahují množinu nezávislých instancí služby, které jsou schopny obsluhovat požadavky od klientů. Zároveň zde musí existovat systém nebo zařízení, které provádí distribuci těchto požadavků od klientů mezi jednotlivé instance. Takový systému se nazývá load balancer[43]. Jde o zařízení nebo systém, který nejprve obdrží požadavek od klienta a pak ho podle svojí vnitřní logiky přepošle požadavek na některou z instancí. Hlavním úkolem load balanceru je především rovnoměrně rozesílat požadavky mezi instance, tak aby nedocházelo k přetěžování nebo naopak tzv. hladovění instancí.

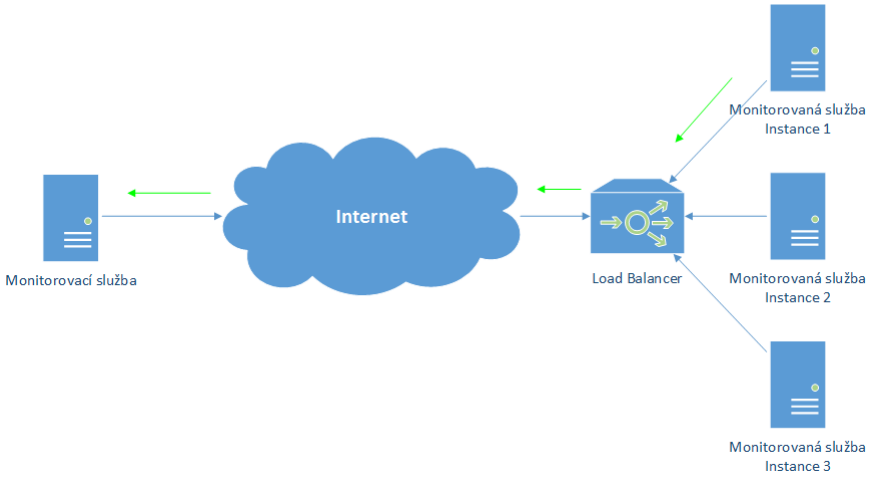
Při aplikaci load balancingu vzniká problém, kdy není možné přímo navázat komunikaci s konkretní instancí služby, což je nepostradatelné pro monitorování jednotlivých instancí. Vizualizaci takové situace lze vidět na obrázku 2.1. Některé z load balancerů umožní tzv. client affinity, což znamená, že load balancer se snaží požadavky od stejného klienta poslat stejné

instanci nebo je možné v dotazu klienta určit požadovanou instanci. Client affinity, ale není stoprocentním řešením problému.



Obrázek 2.1: Problém navázání spojení s konkrétní instancí monitorované služby

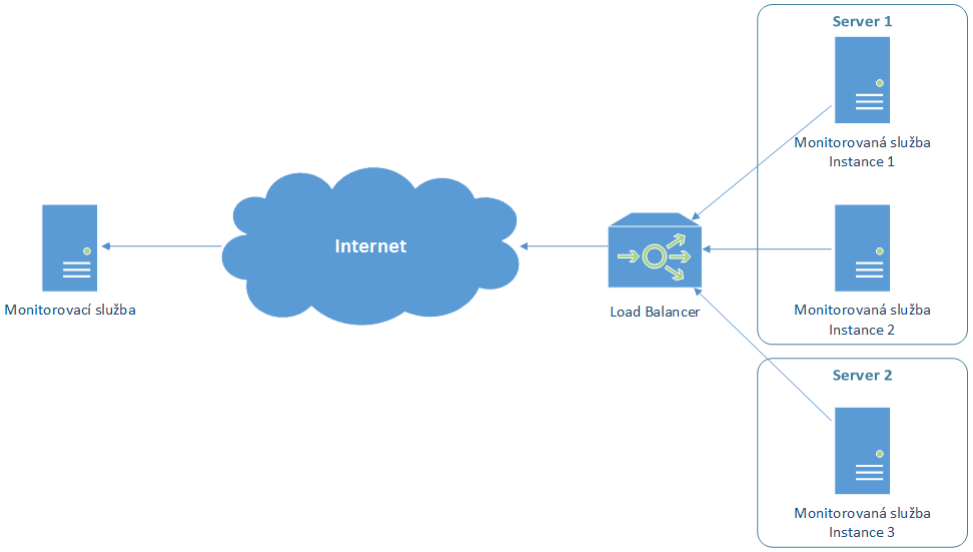
Tento problém lze vyřešit postupem zobrazeným na obrázku 2.2. Spojení nebude s instanci služby navazováno ze serverové části, ale naopak jednotlivé monitorované instance budou navazovat spojení se serverovou částí systému.



Obrázek 2.2: Navázání spojení z monitorované instance služby se serverovou částí

2.1.1 Monitoring instancí služby

Jedním z funkčních požadavků má být monitorování metrik jednotlivých instancí. Vzhledem k prostředí horizontálně škálovatelných služeb, kde přesně není definováno jakým způsobem a na jakém hardwaru jsou jednotlivé instance monitorované služby nasazeny, může nastat situace, která je zobrazena na obrázku 2.3. Může běžet více instancí různých nebo i stejných služeb na jednom fyzickém serveru. Takovéto rozložení může způsobovat problémy především při monitorování hardwarových metrik, jakými jsou například vytížení CPU, RAM, apod. Při zatížení jedné instance může dojít k ovlivnění metrik ostatních instancí.

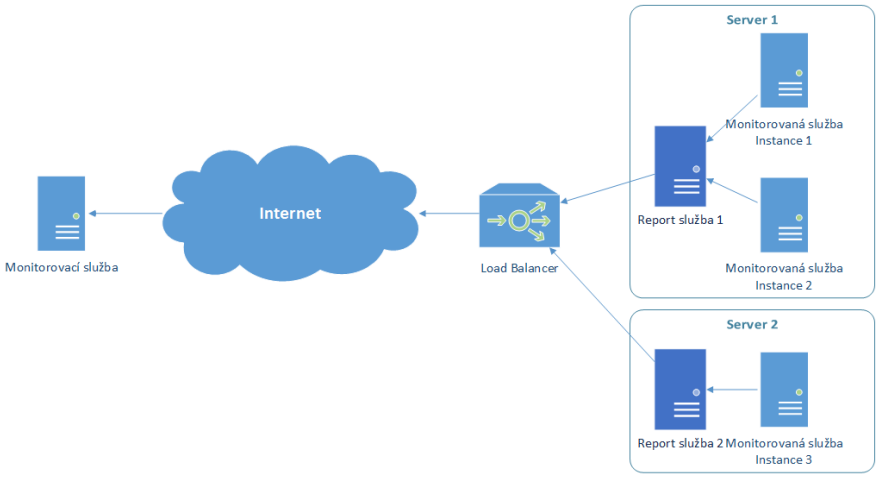


Obrázek 2.3: Rozložení monitorovaných instancí na fyzických serverech

Návrh řešení tohoto problému lze vidět na obrázku 2.4, kde je mezi monitorované instance na každém serveru vložena ještě pomocná služba tzv. Report služba, která se stará o kolekci hardwarových metrik sdílených pro všechny instance na jednom serveru.

Toto rozložení řeší zároveň další problém, kterým je zjištění stavu instance a jeho řízení. Řízením se myslí spuštění nebo pozastavení instance. Například v prostředí Microsoft IIS dochází po 20 minutách k uspání instance, pokud na ni za tu dobu nebyl proveden žádný požadavek. Vzhledem k řešení, u kterého veškerou komunikaci se serverovou částí provádí přímo sama monitorovaná instance, tak neexistuje způsob jak uspanou instanci probudit a změnit její stav. Report služba běží neustále a je schopna uspané

instance v případě potřeby probudit. Zároveň provádí kolekci hardwarových metrik i když jsou instance uspané.



Obrázek 2.4: Monitoring instancí služby s využitím Report služby

2.2 Zvolené měřené metriky

Na základě požadavků a provedené analýzy existujících řešení byly zvoleny tyto základní metriky, které bude systém měřit:

Vytížení CPU Měřeno bude vytížení celého procesu včetně všech jader. Na víceprocesorových systémem bude počítaná kombinovaná hodnota pro všechny procesory.

Využití paměti Měřena bude celková obsazenost operační paměti systému a zároveň i aktuálně dostupná volná paměť, aby bylo možné v každém okamžiku určit dostupnou maximální kapacitu paměti.

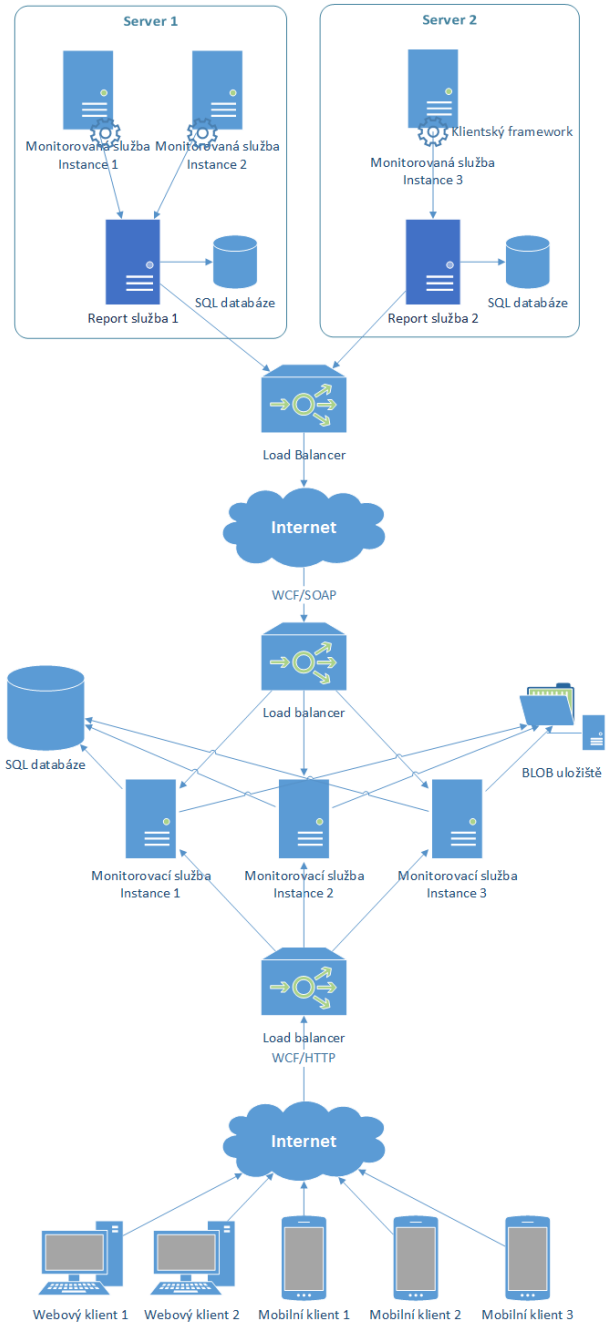
Počet běžících procesů Počet všech aktivních procesů na hardwarovém zařízení, kde běží instance služby.

Počet běžících vláken Celkový počet všech vláken na hardwarovém zařízení, kde běží instance služby.

Disk I/O Měřena bude vytíženost pevných disků v systému.

Doba odpovědi Časový interval od odeslání požadavku na službu až po obdržení odpovědi.

2.3 Architektura systému



Obrázek 2.5: Schéma architektury systému

Systém bude rozdělen do několika samostatných částí, které jsou zobrazeny na obrázku 2.5. Jednotlivé části spolu budou komunikovat přes definované komunikační rozhraní. Těmito částmi jsou:

- Monitorovací služba (serverová část)
- Report služba
- Klientský framework
- Mobilní klienti
- Webový klient

2.3.1 Monitorovací služba

Jedná se o hlavní serverovou část celého systému. Úlohou služby je obsluha připojení všech monitorovaných služeb a klientských aplikací. Z toho důvodu bude vystavovat dvě separátní webové služby:

Služba pro napojení Report služby SOAP služba poskytující veškeré metody nutné pro funkčnost Report služby

Služba pro napojení webových a mobilních klientů HTTP/REST služba poskytující rozhraní pro připojení klientský webových a mobilních aplikací.

Zároveň bude služba uchovávat veškeré naměřené metriky a logy ve své lokální databázi a BLOB uložišti. Monitorovací služba bude navržena, tak aby byla sama horizontálně škálovatelná.

2.3.1.1 Databáze

Vzhledem k nutnosti ukládat velké množství monitorovaných dat bude pro serverovou část jako primární úložiště zvolena relační databáze. Monitorovací služba bude umožňovat díky abstrakci použití různých typů relačních databází, přičemž záměna bude možná pouze úpravou konfigurace. Pro účely vývoje bude zvolena databáze Microsoft SQL Server. Na obrázku 2.6 je zobrazeno schéma databáze.

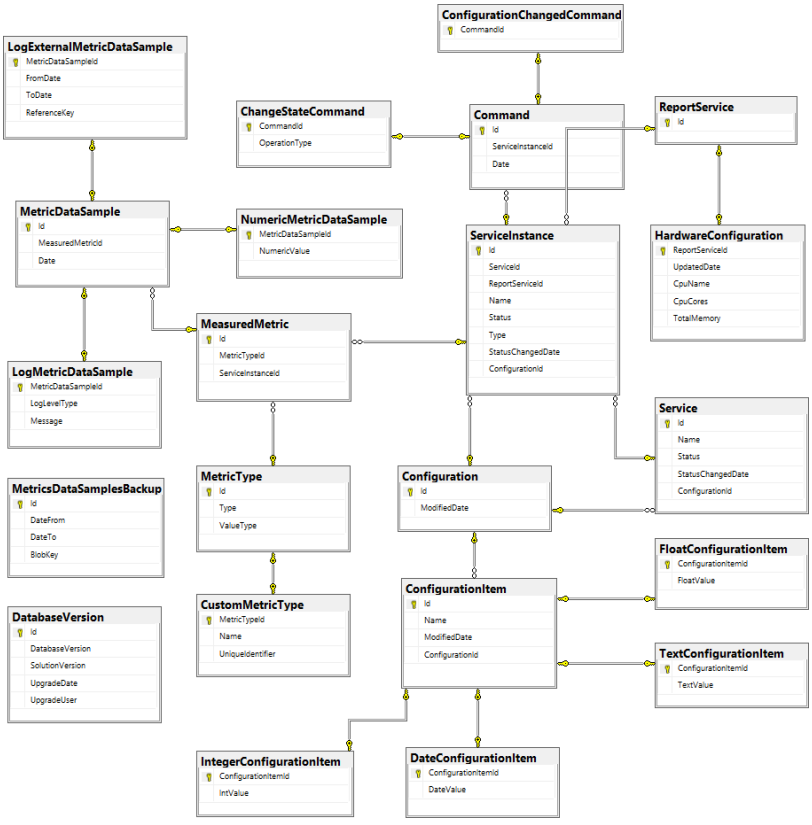
Informace o registrované službě a jejich instancích bude uložena v tabulkách Service a ServiceInstance. Údaje o přihlášených Report službách a hardwarových konfiguracích, na kterých běží, se budou nacházet v tabulkách ReportService a HardwareConfiguration. Tabulka MetricType bude obsahovat seznam všech typů měřitelných metrik (CPU, paměť, Disk I/O,

atd.), které bude možné v systému měřit. Zároveň tabulka CustomMetricType bude obsahut definice všech vlastních přídavných metrik. Vztah mezi měřenou metrikou a instancí služby bude definován v tabulce MeasuredMetric. Samotná naměřená data se pak budou nacházet v tabulkách MetricDataSample, NumericMeasuredMetric a LogMeasuredMetric, přičemž tabulky NumericMeasuredMetric a LogMeasuredMetric budou definovat datový typ měřené hodnoty metriky.

Tabulky Command, ChangeStateCommand a ConfigurationChangedCommand budou obsahovat frontu příkazů (viz. sekce 2.4), které budou čekat na vyzvednutí a zpracování od Report service.

V tabulkách Configuration, ConfigurationItem, IntegerConfigurationItem, FloatConfiguationItem, DateConfigurationItem, TextConfigurationItem se budou nacházet data klíčů a hodnot konfigurace služeb nebo jednotlivých instancí.

Poslední tabulkou bude MetricsDataSamplesBackup, která bude obsahovat odkazy na zálohované data metrik v BLOB uložišti.



Obrázek 2.6: Schéma databáze Monitorovací služby

2.3.1.2 BLOB uložiště

Monitorovací služba bude navíc obsahovat BLOB uložiště. Toto uložiště slouží k ukládání velkých binárních dat nebo souborů. Důvodem použití toho uložiště je především odkládaní naměřených dat původně pocházejících z relační databáze. V relační databázi může v průběhu času vznikat obrovské množství dat pocházejících z měřitelných metrik. Je velmi pravděpodobné, že historicky naměřená data (měsíc a starší) budou velmi málo uživatelsky využívána. Proto budou tyto data periodicky přesouvány z relační databáze do BLOB uložiště ve formě komprimovaných serializovaných dat. V případě, že by byla tato archivovaná data potřeba (např.: pro zobrazení historie, apod.), tak dojde k jejich zpětnému natažení do relační databáze. Veškerá takto archivovaná data, budou rozdělena po časových intervalech takovým způsobem, aby se vždy načetly pouze nutné časové intervaly. BLOB uložiště bude řešené jako modulární, takže bude možné použít různé formy jako např.: souborový systém nebo Azure BLOB uložiště.

2.3.2 Report služba

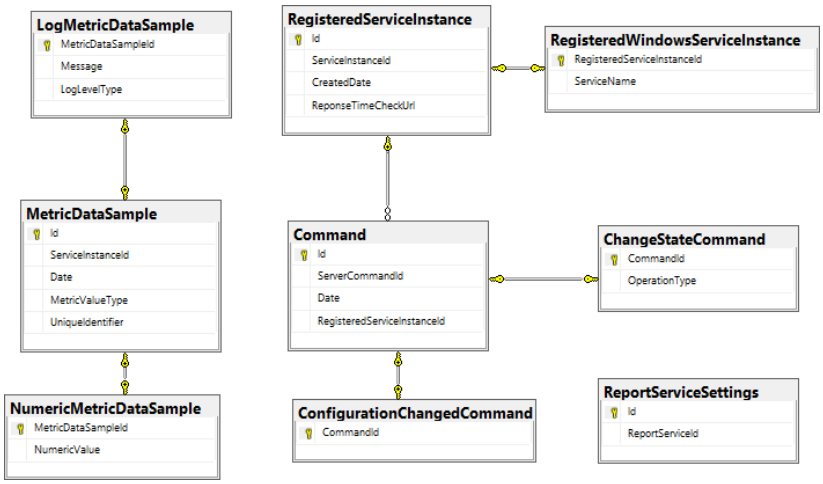
Jak bylo zmíněno v sekci 2.1.1, Report služba bude mezivrstvou mezi monitorovanou instancí služby a Monitorovací službou. Bude se jednat o Windows službu, která poběží na stejném serveru jako monitorovaná instance. Úkolem Report služby bude především:

- Monitorovat stav instancí. Zjišťovat jestli běží nebo jsou pozastaveny.
- Měnit stav instancí. Bude poskytovat možnosti pro spuštění a pozastavení instancí. Samotné instance samy sebe spustit nemohou.
- Monitorovat společné hardwarové metriky pro všechny instance napojení na report službu.
- Poskytovat cache naměřených metrik. Služba bude posílat balíky metrik ze všech napojených instancí po překročení určité hranice počtu záznamů naráz místo toho, aby každá instance zasílala své naměřené metriky zvlášť.
- Poskytovat SOAP službu pro napojení klientského frameworku.

2.3.2.1 Databáze

Z důvodu nutnosti cachování dat v Report službě bylo potřeba zvolit nějaké perzistentní úložiště pro uložení měřených dat předtím, než dojde k jejich

odeslání na Monitorovací službu. Byla zvolena databáze SQLite, kterou lze provozovat bez nutnosti instalace jakéhokoliv dalšího softwaru. Samotná databáze je následně tvořena jediným souborem[39]. Na obrázku 2.7 je zobrazeno schéma relační databáze Report služby. Informace o instancích služeb registrovaných u Report služby se budou nacházet v tabulkách RegisteredServiceInstance a RegisteredWindowsServiceInstance. Tabulky Command, ChangeStateCommand a ConfigurationChangedCommand budou obsahovat frontu příkazů (viz. sekce 2.4), které budou čekat na vyzvednutí a zpracování od klientského frameworku. Veškeré naměřené metriky se pak budou nacházet v tabulkách MetricDataSample, LogMetricDataSample a NumericMetricDataSample. Nakonec tabulka ReportServiceSettings, která bude tvořit uložiště konfiguračních dat Report služby.



Obrázek 2.7: Schéma databáze Report služby

2.3.3 Klientský framework

Bude se jednat o programovou knihovnu DLL (dynamic link library) platformy .NET Framework. Tato knihovna bude následně v kódu aplikace přímo linkována a využívána. Knihovna bude poskytovat všechny potřebné metody pro:

- Registraci a přihlášení instance do monitorování.
- Vytváření vlastní metrik.
- Publikaci naměřených dat metrik a událostí logů.

- Přístup ke konfiguračním datům instance.

Zároveň bude klientský framework obsahovat pluginy pro integrací do logovacích frameworků log4net a NLog.

2.3.4 Mobilní klienti

Jednou z typů aplikací pro uživatelský přístup budou mobilní aplikace pro platformy Android, iOS a Windows. Jak bylo zmíněno v analýze, tak vývoj nativních aplikací (tzv. v nativním prostředí pro každou platformu) by byl časově náročný. Proto zde bude využit jeden z frameworků pro vývoj multiplatformních mobilních aplikací. Konkrétně byl zvolen Xamarin Forms a to z důvodů:

- Vývoj probíhá na platformě .NET Framework a programovacím jazyku C#, který je použit i u ostatních částí systémů. Není tedy nutné používat k vývoji zcela jinou platformu a programovací jazyk.
- Umožňuje implementovat společnou business logiku a uživatelské rozhraní aplikace pro všechny platformy pouze jednou. Ve výsledku je možné napsat 90% společného kódu pro všechny platformy. Ostatních 10% je nutná integrace specifických částí platforem nebo vlastních uživatelských komponent.
- Při vývoji je možné použít návrhový vzor MVVM1.5.5.
- V případě nutnosti není problém dopsat vlastní komponenty tím, že se dopíši implementace pro všechny potřebné platformy a následně wrapper nad Xamarin Forms.
- Na rozdíl od HTML založených frameworků nabízí plný přístup k mobilnímu API všech platforem.

Po stránce funkčnosti bude mobilní klient nabízet pouze monitorování naměřených metrik služeb a jejich instancí. Mobilní klient nebude sloužit ke konfiguraci nebo správě jednotlivých služeb.

2.3.4.1 Vizualizace naměřených metrik

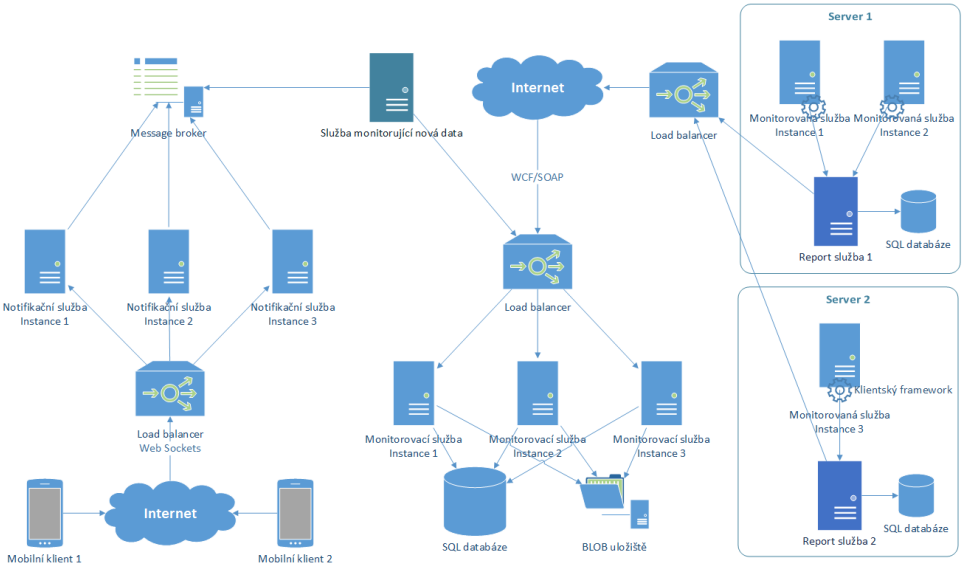
V aplikaci je nutná určitá forma vizualizace naměřených metrik, k čemuž se přesně hodí různé typy grafových komponent. Bohužel Xamarin Forms neobsahuje žádné integrované grafové komponenty. Proto bylo vybráno použití opensource knihovny OxyPlot pod licencí MIT[31], která je dostupná zdarma pro Xamarin Forms.

2.3.4.2 Možnosti aktualizace dat metrik

S vizualizací naměřených metrik do grafů souvisí i způsob aktualizace dat. Od aplikace se očekává, že veškerá grafy měřených metrik budou automaticky aktualizovány po přijetí nových dat. Existují dva přístupy, jak lze aktualizaci implementovat. Mobilní klient bude obsahovat oba přístupy a umožní jejich snadnou záměnu.

Periodická aktualizace ze strany mobilního klienta
Implementačně se jedná o nejjednodušší způsob, kdy se mobilní klient bude periodicky dotazovat Monitorovací služby, zdali nejsou pro zobrazované metriky k dispozici nová data. Tento přístup má své negativum, při kterém dochází k neustálému generování dotazů na Monitorovací službu, i když nejsou k dispozici nová data u zobrazovaných metrik. Při větším počtu mobilní klientů to může být znatelná zátěž na Monitorovací službu.

Notifikace o nových datech ze strany Monitorovací služby
Tato implementace k problematice přistupuje zcela rozdílně než předchozí řešení. Princip spočívá v tom, že samotná Monitorovací služba bude upozorňovat mobilního klienta na nová data. Na obrázku 2.8 je zobrazeno schéma takového řešení, které je implementačně mnohem náročnější a obsahuje několik nových částí:



Obrázek 2.8: Schéma architektury systému systému s notifikační službou

Message broker Systém pro hromadnou distribuci zpráv ActiveMQ. Podrobné vysvětlení v sekci 1.5.7.

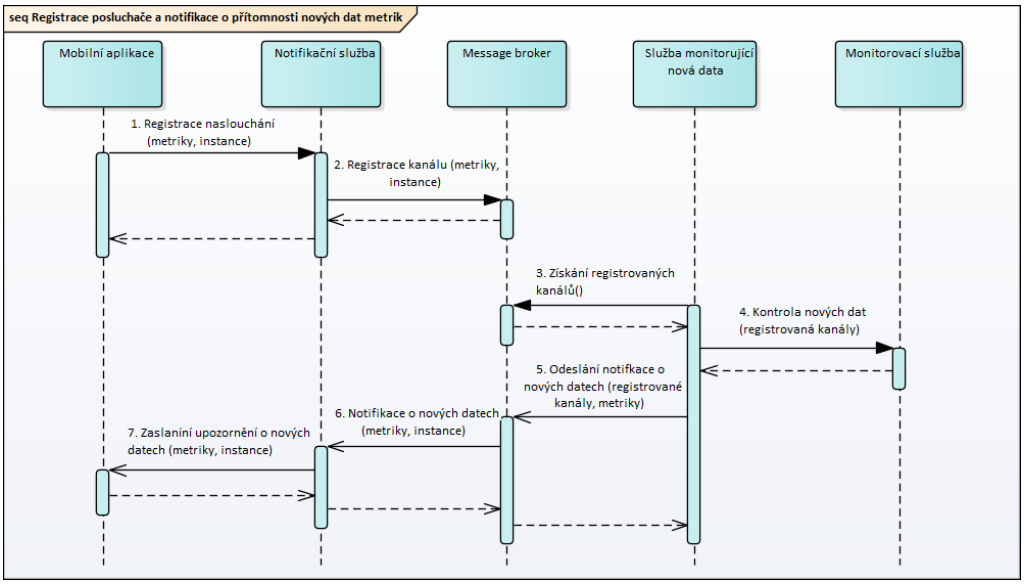
Služba monitorující nová data Soběstačná Windows služba monitorující přítomnost nových dat metrik na základě registrovaných kanálů v Message brokeru. Pokud narazí na nová data, pak upozorní všechny registrované kanály podle jejich nastavených filtrů. Bude fungovat v roli odesílatele na Message brokeru.

Notifikační služba Horizontálně škálovatelná služba, která bude udržovat s mobilními klienti otevřené spojení přes web sockety a zasílat upozornění o dostupnosti nových dat. Bude fungovat v roli posluchače na Message brokeru.

Na obrázku 2.9 je zobrazen postup registrace mobilního klienta k odběru upozornění a následné provedení takového upozornění. Celý postup lze rozdělit to těchto fází:

1. Připojení mobilní klienta k notifikační službě a provedení registrace na základě požadovaných metrik a instancí.
2. Vytvoření kanálu nebo přihlášení k existujícímu kanálu identifikovaného typem metrik a instancemi.
3. Získání všech registrovaných kanálů. Tedy všech registrovaných kombinací typů metrik a instancí.
4. Provedení kontroly zdali se v Monitorovací službě nachází nová data metrik od poslední kontroly.
5. Zaslání zprávy na všechny kanály Message brokeru, pro které existují nová data.
6. Zaslaní upozornění všech posluchačům (instancím Notifikační služby) na Message brokeru o nových datech.
7. Zaslání zprávy mobilní aplikaci přes otevřený web socket o dostupnosti nových dat.

Tento řešení je možné obdobně aplikovat i na webové klienta, ale v této práci je navrženo pouze pro mobilní aplikace.



Obrázek 2.9: Diagram znázorňující postup registrace klienta k odběru upozornění o nových datech metrik a provedení takového upozornění

2.3.5 Webový klient

Druhou klientskou aplikací bude webový klient, který bude nabízet mnohem širší možnosti než mobilní klienti:

- Monitorování metrik služeb a jejich instancí.
- Možnost vytvářet nové monitorované služby.
- Možnost měnit stav služeb nebo instancí.
- Možnost měnit konfiguraci služeb nebo instancí.

Webový klient bude implementován v technologii ASP .NET MVC. Technologie byla zvolena z toho důvodů, že se v současné době jedná o preferovanou technologii pro vývoj nových webových aplikací na platformě .NET Framework. Velký plusem ASP .NET MVC je možnost návrhu celého webového front-endu v klasických webových technologiích HTML/CSS/Javascript. Je tedy možné použít již existující rozšiřující front-endové frameworky. Při vývoji bude použit front-endový framework Bootstrap dostupný pod licencí MIT[8], který zjednodušuje vývoj grafického rozhraní webových aplikací. Bootstrap obsahuje vlastní systém pozicování uživatelských komponent na webové stránce, který počítá s různou velikosti stránky na různých

zařízeních. Zároveň obsahuje předefinované uživatelské komponenty, které je možné použít.

2.4 Distribuce událostí z Monitorovací služby do instance monitorované služby

V systému mohou vznikat požadavky ze strany Monitorovací služby ke konkrétním monitorovaným instancím (např.: změna stavu, konfigurace, atd.). Vzhledem ke zvolenému řešení komunikace, které je popsané v sekci 2.1, není možné navázat přímou komunikaci z Monitorovací služby ke konkrétní instanci. Na druhou stranu jednotlivé instance jsou schopny navázat spojení s Monitorovanou službu. Proto bylo zvoleno řešení, kdy jednotlivé instance se budou periodicky dotazovat Monitorovací služby, jestli pro ně má tzv. příkazy. Monitorovací služba si na své straně bude držet frontu těchto nezpracovaných příkazů a při dotazu je odešle a odstraní z fronty.

2.5 Konfigurace služeb

Jendou z hlavní funkcionalit systému má být možnost konfigurovat služby nebo jednotlivé instance. Konfigurace bude řešena jako seznam názvů klíčů a hodnot, přičemž nebudou povoleny duplicitní názvy klíčů. Hodnoty budou těchto typů:

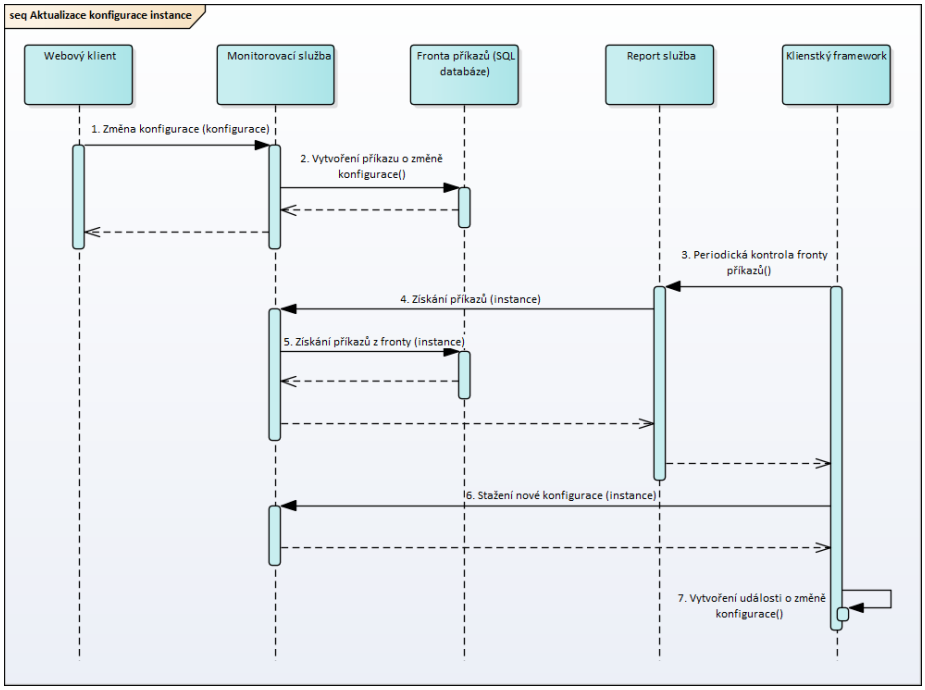
- Text
- Celé číslo
- Číslo s desetinou čárkou
- Datum

V systému bude existovat možnost rozšířit tento systém o další typy hodnot. Vzhledem k tomu, že konfiguraci mohou mít jednotlivé instance i celá služba, tak může dojít k situaci, kdy instance bude mít definované stejné klíče jako služba (mohou být klidně i rozdílných hodnot). V tom případě bude zvolena hodnota z instance služby jako primární.

Ke konfiguraci bude možné přistupovat přes metody definované v klientském frameworku. Klientský framework bude navíc obsahovat mechanizmus pro informování instance o tom, že došlo ke změně konfigurace. Tento mechanizmus ke svému fungování bude využívat princip front příkazů popsaný

v sekci 2.4. Celý postup aktualizace hodnot konfigurace, který je zobrazen na obrázku 2.10, lze rozdělit do těchto kroků:

1. Úprava konfigurace ve webovém prohlížeči.
2. Vytvoření příkazu o změně konfigurace pro instanci služby, kde došlo ke změně do její fronty příkazů.
3. Periodická kontrola příkazů z klientského frameworku.
4. Dotaz na získání čekajících příkazů.
5. Získání čekajících příkazů z fronty.
6. Pokud byl detekován příkaz oznamující změnu konfigurace, pak pokračuje dotaz na stažení celé konfigurace.
7. Aktualizace kopie konfigurace v klientském frameworku a zaslání interní programové události o změně konfigurace.



Obrázek 2.10: Diagram znázorňující postup úpravy konfigurace instance z webové klienta a následná propagace změn do klientského frameworku

2.6 Integrace služeb do monitorování

Důležité je nastínit, jakým způsobem v systému bude probíhat registrace služeb. Tento proces je rozdělen na registraci monitorovaných služeb a registraci jednotlivých instancí monitorované služby.

2.6.0.1 Registrace služby

Registraci nové monitorované služby bude možné provést pouze ve webovém klientovi. Po registraci uživatel obdrží tzv. Service Id, což je automaticky generovaný číselný identifikátor služby. Tento identifikátor bude pak nezbytný při dalším použití v klientského frameworku a Report službě.

2.6.0.2 Registrace instance služby

Instance se registruje přes metodu dostupné v klientském frameworku. Tato metoda bude vyžadovat právě výše zmíněné Service Id. Po úspěšné registrace vrátí automaticky generovaný identifikátor instance, tedy tzv. Service Instance Id. Tento identifikátor bude instance používat po celou dobu své existence.

2.6.1 Integrace logování

V rámci funkce monitorování bude možné provádět kolekci aplikačních záznamů událostí, neboli logů. Z pohledu návrhu systému bude logování pouze další metrikou, ale místo číselných hodnot bude ukládat textové řetězce reprezentující textový záznam logu. Kromě těchto textových řetězců bude u jednotlivých záznamů uložena navíc informace o důležitosti záznamu (informace, varování, chyba, atd.).

Pro integraci bude sloužit klientský framework, který bude poskytovat metody pro manipulaci s logy.

2.6.2 Integrace vlastních metrik

Kromě základních monitorovaných metrik bude systém umožňovat rozšíření o další nové číselné metriky. Číselné metriky jsou zvoleny především z důvodů vizualizace, u které je jasné, že takové metriky budou zobrazovány jako graf. Integraci nových metrik bude možné provést dvěma způsoby.

2.6.2.1 Report služba

Report služba bude rozšířitelná o další metriky formou tzv. pluginů. Uživatel dodá implementovanou logiku metrika jako DLL knihovnu, kterou následně přes konfigurační soubor IoC containeru integruje do Report služby.

2.6.2.2 Klientský framework

Klientský framework jako programová knihovna bude poskytovat metody pro vytváření nových metrik a zároveň i pro publikaci měřených dat.

Kapitola 3

Implementace

V této kapitola jsou popsány technické záležitosti týkající se implementace řešení. Nejprve je popsán veškerý software potřebný k vývoji a provozování systému. Následně kapitola obsahuje popis struktury projektu a jednotlivých komponent.

3.1 Vývojový software

Pro vývoj byla zvolena platforma .NET Framework. Konkrétně ve verzi 4.5. Z této volby následně plyne i volba vývojového softwaru. Veškerý vývoj kromě vývoje mobilního iOS klienta je možné provést na jednom stroji s operačním systémem Windows. K vývoji pro produkty společnosti Apple je nutné vlastnit zařízení s operačním systémem macOS (dříve OS X). Vzhledem k tomu, že licence macOS je svázána přímo s fyzickým zařízením, je nutné vlastnit počítač od společnosti Apple (Mac, MacBook). K vývoji použitý software:

Microsoft Windows 10 Professional x64 Operační systém od společnosti Microsoft potřebný k běhu vývojové prostředí Visual Studio 2015 Enterprise a také jediný systém, u kterého je možné zkompilovat mobilní Windows aplikaci.

Microsoft Windows Server 2016 Standard Serverový systém od společnosti Microsoft. Byl využit při finálním testování kdy bylo nutné nasadit aplikaci do IIS.

Microsoft SQL Server 2016 Developer Edition Databázový systém od společnosti Microsoft.

Microsoft Visual Studio 2015 Enterprise IDE pro vývoj aplikací postavených na .NET Frameworku.

Android SDK 25.2 Sada nástrojů pro kompilaci a vývoj Android aplikací.

Xamarin.Android 7.0 Sada knihoven a nástrojů pro vývoj Xamarin aplikací na platformě Android.

macOS 10.12 Operační systém od firmy Apple nutný ke spuštění vývojového prostředí Xcode.

Xcode 8.2 IDE pro vývoj aplikací běžících na OS X nebo iOS. Nutná podmínka ke kompilaci iOS aplikací.

Xamarin.iOS 10.3 Sada knihoven a nástrojů pro vývoj Xamarin aplikací na platformě iOS.

Veškerý výše zmíněný software od společnosti Microsoft je dostupný zdarma studentům v rámci programu Microsoft Imagine. Veškerý software od společnosti Xamarin a Android vývojové nástroje jsou dostupné také zdarma. Operační systém macOS je dostupný na zařízeních společnosti Apple. Vývojové nástroje xCode jsou dostupné zdarma, ale ve volné verzi není dostupné nasazení aplikací na fyzická zařízení.

3.2 Rozdělení kódu do částí

Projekt je rozdělen do velkého množství komponent, ale všechny tyto komponenty lze zařadit do části implementace Monitorovací služby, Report služby, webového nebo do mobilních klientů, klientského frameworku a nebo do implementace notifikační služby.

3.2.1 Monitorovací služba

Část obsahující implementaci Monitorovací služby, která je rozdělena na tyto komponenty:

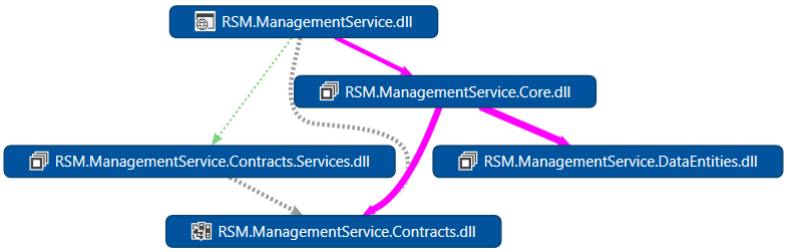
RSM.ManagementService Implementace webových služeb a jejich napojení na logiku. Obsahuje také i všechny konfigurační soubory.

RSM.ManagementService.Core Implementace veškeré business logiky.

RSM.RSM.ManagementService.DataEntities Obsahuje přístup k databázi.

RSM.RSM.ManagementService.DataContracts Obsahuje objektovou reprezentaci datových typů používaných ve webových službách.

RSM.RSM.ManagementService.DataContracts.Services Obsahuje rozhraní definující obě webové služby, které vystavuje serverová část.



Obrázek 3.1: Diagram závislostí komponent Monitorovací služby

3.2.2 Report služba

Implementaci Report služby je rozdělena do těchto komponent:

RSM.ReportService Implementace Windows služby a její napojení na logikou. Součástí je i implementace webové služby, přes kterou se napojuje klientský framework.

RSM.ReportService.Core Implementace veškeré business logiky.

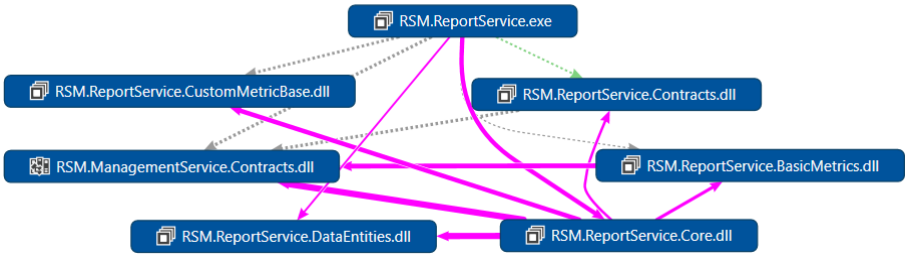
RSM.ReportService.DataEntities Obsahuje přístup k lokální databázi.

RSM.ReportService.DataContracts Obsahuje objektovou reprezentaci datových typů a služby pro komunikaci mezi Report službou a klientským frameworkem.

RSM.ReportService.Installer Obsahuje implementaci instalátoru Report služby s použitím knihovny WiX.

RSM.ReportService.BasicMetrics Implementuje sadu základním měřitelných (využití CPU, RAM, Disk I/O).

RSM.ReportService.CustomMetricBase Obsahuje rozhraní umožňující implementaci dalších metrik integrovatelných do Report služby.



Obrázek 3.2: Diagram závislostí komponent Report služby

3.3 Webový a mobilní klienti

Rozdělení webového a mobilních klientů do jednotlivých komponent:

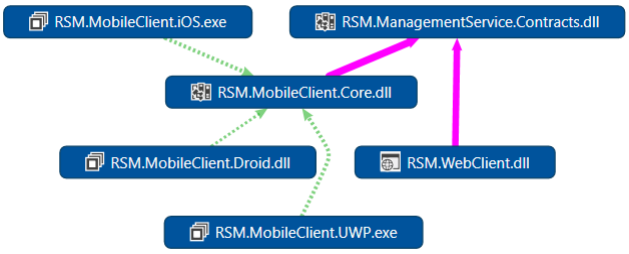
RSM.WebClient Implementace webového klienta v ASP .NET MVC.

RSM.MobileClient.Core Implementace business logiky mobilní aplikace včetně grafického rozhraní pro všechny platformy.

RSM.MobileClient.Droid Implementace spustitelné Android aplikace. Obsahuje napojení na Core část.

RSM.MobileClient.iOS Implementace spustitelné iOS aplikace. Obsahuje napojení na Core část.

RSM.MobileClient.UWP Implementace spustitelné Windows aplikace. Obsahuje napojení na Core část.



Obrázek 3.3: Diagram závislostí komponent webového a mobilních klientů

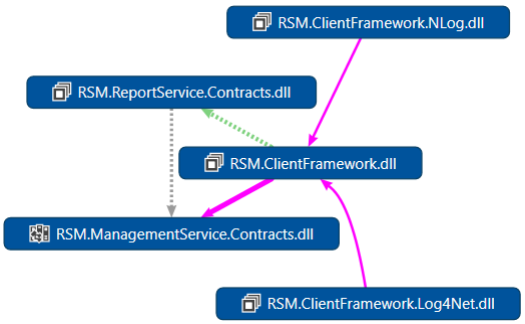
3.3.1 Klientský framework

Klientský framework je jediná DLL knihovna obsahující potřebná rozhraní a implementaci pro snadnou integraci do aplikací. Skládá se z komponent:

RSM.ClientFramework Implementace klientského frameworku.

RSM.ClientFramework.Log4Net Obsahuje rozšíření klientského frameworku o podporu logování skrze knihovnu Log4Net.

RSM.ClientFramework.NLog Obsahuje rozšíření klientského frameworku o podporu logování skrze knihovnu NLog.



Obrázek 3.4: Diagram závislostí komponent klientského frameworku

3.4 Notifikační služba

Pod touto částí se skrývá implementace jak samotné Notifikační služby, tak Windows služby monitorující změny a message brokeru. Celé řešení se skládá z:

RSM.DataUpdaterWorkerService Implementace Windows služby periodicky monitorující data metrik na základě registrovaných kanálů v message brokeru. Pokud aplikace detekuje změnu dat ohledně některého registrovaných kanálů, tak zašle na daný kanál notifikaci o změně.

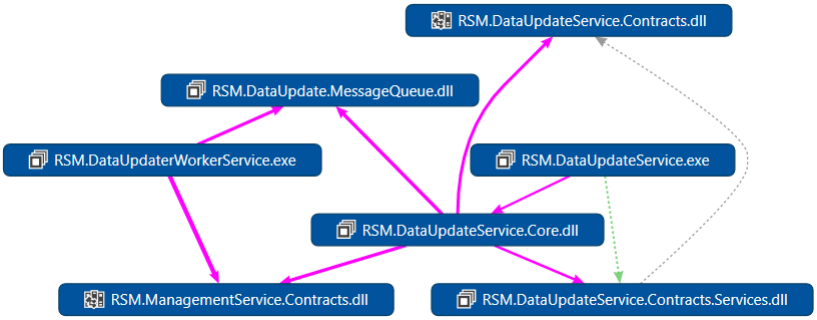
RSM.DataUpdateService Implementaci webové služby poskytující rozhraní pro notifikaci o monitorovaných metrikách klientským mobilním aplikacím.

RSM.DataUpdateService.Core Obsahuje business logiku notifikační služby.

RSM.DataUpdateService.Contracts Obsahuje objektovou reprezentaci datových typů používaných ve webové službě.

RSM.DataUpdateService.Contracts.Services Obsahuje rozhraní definující webovou službu pro notifikaci o změně měřených metrik.

RSM.DataUpdate.MessageQueue Obsahuje implementaci napojení na ActiveMQ.



Obrázek 3.5: Diagram závislostí komponent notifikační služby

3.5 Nasazení Report služby

Během návrhu bylo definováno, že se Report služba bude nasazovat automaticky při použití klientského frameworku. Přímo samotný klient framework by provedl kompletní nasazení a spuštění Report služby jako Windows služby. Zde se objevil problém, kdy pro nasazení Windows služeb jsou nutné vysoká uživatelská práva (často práva administrátora), přičemž aplikace, které využívají klientský framework těmito právy většinou nedisponuje. Například webové služby nasazené pod Microsoft IIS běží pod separátními uživatelskými účty s minimálními pravomocemi v oblasti správy Windows služeb. Proto bylo zvoleno jiné řešení nasazení, kdy byl vytvořen automatický instalátor pro Report službu. Implementace instalátoru byla realizována s použitím knihovny WiX Toolset[29].

Kapitola 4

Testování

Testování je nezbytnou, ačkoliv často velmi opomíjenou součástí vývoje softwaru. V této kapitole je popsáno prováděné testovaní na implementovaném systému. Vzhledem k povaze celého řešení bylo rozhodnuto provést otestování spolehlivosti a zátěže. Především při monitorování v cloudovém prostředí s velkou množinou monitorovaných instancí může vznikat obrovské množství naměřených záznamů metrik. Je nezbytné, aby systém běžel bez chyb nebo výpadů po dlouho dobu. Celé testování probíhalo nasazením celého systému na testovací prostředí, kde v každé komponentě bylo integrováno logovaní chyb a zároveň bylo monitorováno zatížení jednotlivých strojů v testovacím prostředí.

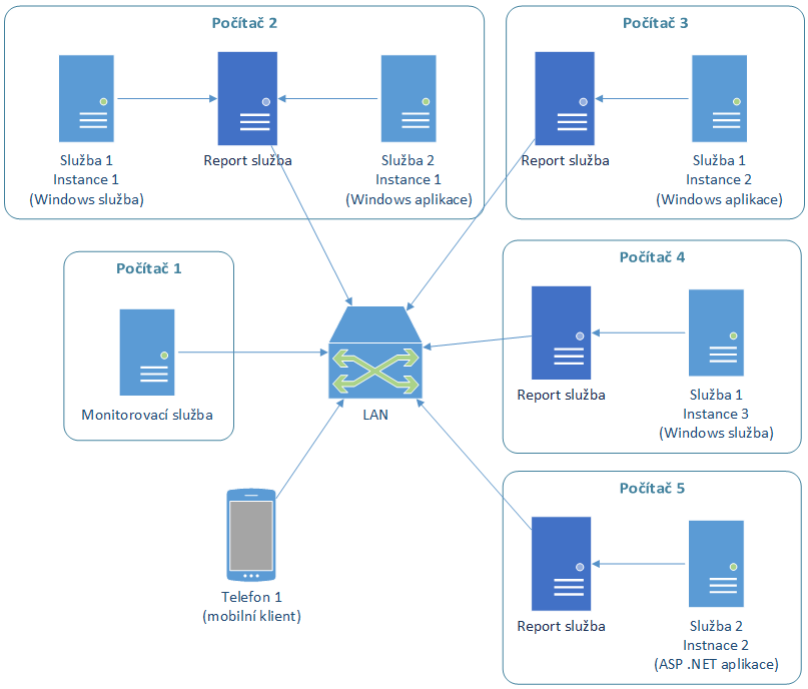
4.1 Testovací prostředí

Pro účely testovaní bylo potřeba vytvořit patřičné testovací prostředí. S ohledem na zaměření celého systému bylo nutné zvolit větší množství zařízení k testu. Jednotlivá zařízení jsou popsána v tabulce 4.1.

Název	Zařízení	Procesor	RAM	Operační systém
Počítač 1	Notebook	Intel Core i7-6700HQ	16GB	Windows 10 Pro x64
Počítač 2	Notebook	Intel Core i5-M560	4GB	Windows 10 Pro x64
Počítač 3	Notebook	Intel Core 2 Duo P8400	4GB	Windows 7 Pro x64
Počítač 4	Desktop	Intel Core i5-750	4GB	Windows 10 Home x64
Počítač 5	Server	Intel Atom D525	4GB	Windows Server 2016 Standard
Telefon 1	Smartphone	Qualcomm MSM8992	3GB	Windows 10 Mobile

Tabulka 4.1: Seznam zařízení pro testování

Na obrázku 4.1 je zobrazeno zapojení zařízení a nasazení jednotlivých částí systému. Všechny zařízení byly připojeny na stejné lokální síti přes ethernet nebo Wi-Fi.



Obrázek 4.1: Schéma testovacího prostředí

Pro monitorování stavu a činnosti všech částí systémů byly při testu využity tyto aplikace:

NLog Logovací knihovna integrovaná do všech testovaných částí pro výpis důležitých událostí.

Microsoft SQL Server Management Studio 16.5.1 Nástroj pro management Microsoft SQL Server databází. Použit pro monitorování stavu databáze Monitorovací služby.

SQLiteStudio 3.1.1 Aplikace pro správu SQLite databází. Byla použita pro monitorování stavu lokálních databází Report služeb.

Windows Performance Monitor Integrovaný nástroj v operačním systému Windows pro monitorování zdrojů (vytížení CPU, RAM, atd.)

LinX 0.6.5 Software pro simulaci zátěže CPU.

4.1.1 Testovaní v průběhu vývoje

Pro účely testů byly vytvořeny tři testovací aplikace v roli monitorovaných služeb. Byly zvoleny tři typy aplikací, u kterých se na platformě Windows předpokládá největší využití a to tedy:

- Windows aplikace (formou konzolové aplikace)
- Windows služba
- ASP .NET aplikace

Všechny tyto tři aplikace v sobě integrují klientský framework pro účely testu. Na obrázku 4.1 lze vidět nasazení těchto typů aplikací v testovacím prostředí.

V průběhu implementace byla neustále pomocí těchto aplikací ověřována funkčnost systému. Testování probíhalo formou black-box, kdy tyto testovací aplikace přistupovali k systému pouze skrze metody dostupné klientským frameworkem a neměli tedy jinou návaznost na zbytek systému.

4.2 Testování stability

Účelem tohoto testu bylo ověřit stability systému v zátěži po delší časové období. Celý systém byl po dobu 12 hodin nasazen v testovacím prostředí a po celou dobu bylo monitorováno chování systému. Během testů také náhodně docházelo k úmyslnému zapínáni a vypínání testovacích aplikací a Report služby simulující podobné chování v cloudovém prostředí.

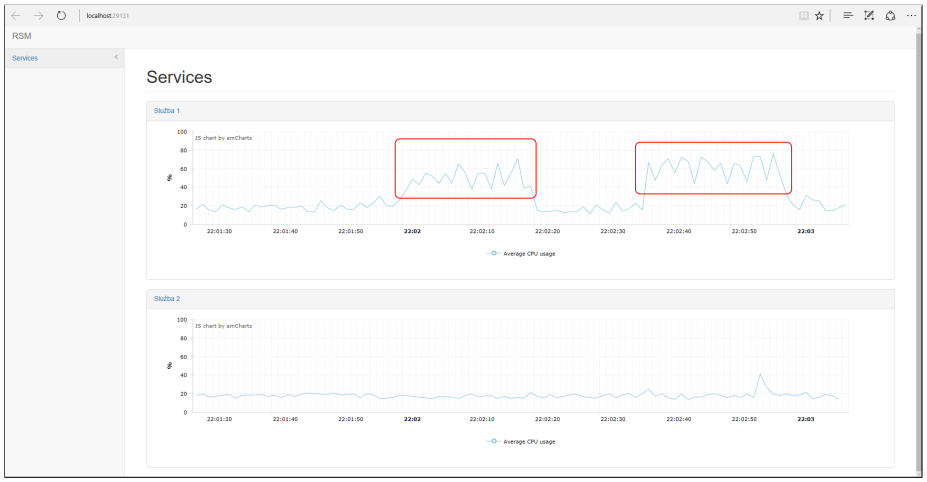
4.3 Testování měření metrik

Účelem testu bylo ověřit, zda při změně měřených metrik dochází k jejich správnému měření a indikaci v systému. Důležitý je především test agregovaných metrik. Tedy jestli při skokové změně některé metriky (například vytížení CPU) u jedné instance, je tato změně viditelná i v agregované metrice celé služby. Test byl prováděn se všemi základními metrikami, které systém měří.

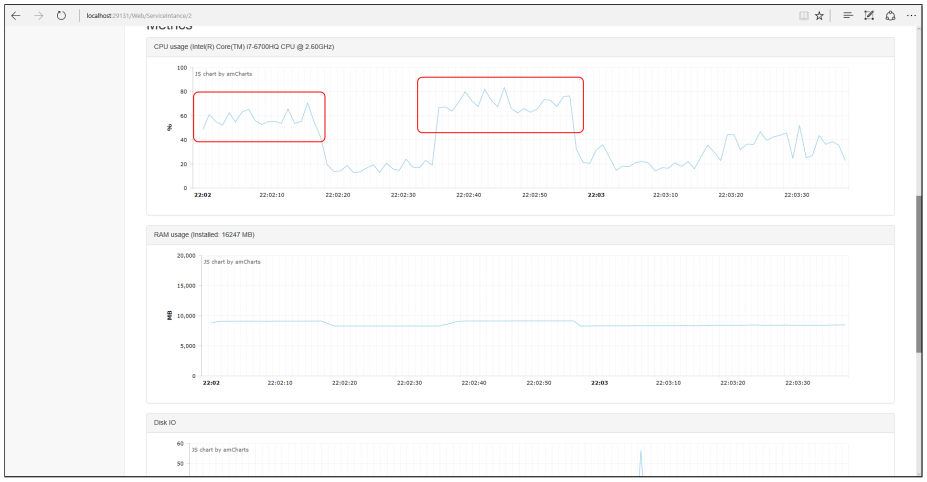
4.4 Výsledek testování

V průběhu vývoje docházelo k testování popsaným postupem výše. Díky němu byla odhalena řada problému a chyb, které byly následně vyřešeny.

Nevýhodou tohoto testování je limitující počet použitých testovacích zařízení. V reálném prostředí může systém obsluhovat mnohonásobně více služeb a instancí. Potenciálně se mohou ve finálním řešení objevit chyby tykající se právě obsluhy velmi velkého počtu klientů.



Obrázek 4.2: Zvýrazněn potenciální problém služby (zvýrazněné zatížení CPU)



Obrázek 4.3: Zvýrazněn potenciální problém konkrétní instance služby (zvýrazněné zatížení CPU)

Závěr

V rámci této práce se podařilo navrhnout a implementovat škálovatelný systém, umožňující monitorování základních metrik jako vytížení CPU, RAM, disku apod. a aplikačních logů událostí. Zároveň byl systém navržen jako modulární, takže je snadno rozšířitelný o další metriky. Systém kromě monitoring umožňuje i vzdálenou konfiguraci služeb. Veškeré monitorování a konfigurace je navíc prováděno přímo na jednotlivých instancích služeb. Systém byl rozdělen na horizontálně škálovatelnou serverovou část provádějící obsluhu monitorovaných služeb a klientských aplikací. Součástí řešení je klientský framework, který implementuje veškeré API pro napojení do systému a je možné ho snadno integrovat do existujících služeb. K systému byla implementována webová aplikace umožňující provádět uživatelskou správu monitorování a konfigurace služeb. Byly také vytvořeny doplňující mobilní aplikace pro platformy iOS, Android a Windows umožňující provádět monitorování služeb. Výsledný systém splňuje všechny definované cíle a požadavky.

Návrhy dalších možných rozšíření systému

I když systém splnil všechny stanové požadavky, tak stále existuje prostor pro další rozšíření:

- Implementaci Report služby a klientského frameworku pro další platformy a operační systémy.
- Přidání automatizované analýzy dat metrik, u kterých by byl hledán výskyt nestandardního chování monitorované služby a následně i vytvořena upozornění pro uživatele systému.

- Přidání autorizace uživatelů a s tím související i uživatelské účty a práva.
- Zabezpečení komunikace s použitím například HTTPS.

Literatura

[1] Adobe Systems Incorporated: PhoneGap. 2016, [cit. 2016-12-27]. Dostupné z: http://phonegap.com

[2] Amazon.com, Inc.: Amazon CloudWatch Cloud Network Monitoring Services. 2016, [cit. 2016-12-27]. Dostupné z: https://aws.amazon.com/cloudwatch/

[3] Amazon.com, Inc.: Amazon CloudWatch Metrics and Dimensions Reference. 2016, [cit. 2016-12-27]. Dostupné z: http://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CW_Support_For_AWS.html

[4] Andy De George, John Taubensee, C.J. Gronlund, Ralph Squillace, Cynthia Nottingham: Should I choose cloud services or something else? 2016, [cit. 2016-12-27]. Dostupné z: https://docs.microsoft.com/en-us/azure/cloud-services/cloud-services-choose-me

[5] Apache Software Foundation: Apache ActiveMQ Features. 2016, [cit. 2016-12-27]. Dostupné z: http://activemq.apache.org/features.html

[6] Apache Software Foundation: Apache log4net Features. 2016, [cit. 2016-12-27]. Dostupné z: https://logging.apache.org/log4net/release/features.html

[7] Apache Software Foundation: Architectural overview of Cordova platform. 2016, [cit. 2016-12-27]. Dostupné z: https://cordova.apache.org/docs/en/latest/guide/overview/index.html

[8] Bootstrap Core Team: Bootstrap The world's most popular mobile-first and responsive front-end framework. 2016, [cit. 2016-12-27]. Dostupné z: http://getbootstrap.com

[9] Facebook Inc.: React Native Docs. 2016, [cit. 2016-12-27]. Dostupné z: https://facebook.github.io/react-native/releases/next/docs/getting-started.html

[10] Fowler, Martin: Inversion of Control Containers and the Dependency Injection pattern. 2004, [cit. 2016-12-27]. Dostupné z: http://www.martinfowler.com/articles/injection.html

[11] Free Software Foundation: GNU Lesser General Public License v2.1. 1999, [cit. 2016-12-27]. Dostupné z: http://www.gnu.org/licenses/lgpl-2.1-standalone.html

[12] Jaroslaw Kowalski, Kim Christensen, Julian Verdurmen: NLog. 2016, [cit. 2016-12-27]. Dostupné z: http://nlog-project.org

[13] Jaroslaw Kowalski, Kim Christensen, Julian Verdurmen: NLog license. 2016, [cit. 2016-12-27]. Dostupné z: https://github.com/NLog/NLog/blob/master/LICENSE.txt

[14] John Kemnetz, Dominic Betts, Nicole Berdy, Rob Boucher Jr.: Supported metrics with Azure Monitor. 2016, [cit. 2016-12-27]. Dostupné z: https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/monitoring-supported-metrics

[15] Laurent Bugnion: MVVM Messenger and View Services in MVVM. 2013, [cit. 2016-12-27]. Dostupné z: https://msdn.microsoft.com/en-us/magazine/jj694937.aspx

[16] Massimo Della Rovere: Amazon CloudWatch per monitorare le risorse su AWS. 2014, [cit. 2016-12-27]. Dostupné z: https://emarket.pe/it/blog/amazon-cloudwatch-per-monitorare-risorse-e-applicazioni-su-aws/

[17] Microsoft Corporation: Application Insights. 2016, [cit. 2016-12-27]. Dostupné z: https://azure.microsoft.com/en-us/services/application-insights/

[18] Microsoft Corporation: ASP.NET MVC Overview. 2016, [cit. 2016-12-27]. Dostupné z: https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx

[19] Microsoft Corporation: ASP.NET Web API. 2016, [cit. 2016-12-27]. Dostupné z: https://msdn.microsoft.com/en-us/library/hh833994(v=vs.108).aspx

[20] Microsoft Corporation: ASP.NET Web Forms Pages Overview. 2016, [cit. 2016-12-27]. Dostupné z: https://msdn.microsoft.com/en-us/library/428509ah.aspx

[21] Microsoft Corporation: Develop apps for the Universal Windows Platform (UWP). 2016, [cit. 2016-12-27]. Dostupné z: https://msdn.microsoft.com/en-us/library/dn975273.aspx

[22] Microsoft Corporation: Hosting Services. 2016, [cit. 2016-12-27]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms730158(v=vs.110).aspx

[23] Microsoft Corporation: Microsoft software license terms. 2016, [cit. 2016-12-27]. Dostupné z: https://www.visualstudio.com/license-terms/mt171547/

[24] Microsoft Corporation: What Is Windows Communication Foundation. 2016, [cit. 2016-12-27]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms731082(v=vs.110).aspx

[25] Nagios Enterprises: Download Nagios Nagios Core is free. 2016, [cit. 2016-12-27]. Dostupné z: https://www.nagios.org/downloads/

[26] Nagios Enterprises: Nagios XI Easy Network, Server Monitoring and Alerting. 2016, [cit. 2016-12-27]. Dostupné z: https://www.nagios.com/products/nagios-xi/

[27] Nagios Enterprises: System Metrics Nagios Exchange. 2016, [cit. 2016-12-27]. Dostupné z: https://exchange.nagios.org/directory/Plugins/System-Metrics

[28] .NET Foundation: .NET Foundation Entity Framework. 2016, [cit. 2016-12-27]. Dostupné z: https://dotnetfoundation.org/entity-framework

[29] .NET Foundation: WiX Toolset Documentation. 2016, [cit. 2016-12-27]. Dostupné z: http://wixtoolset.org/documentation/

[30] NHibernate Community: NHibernate Relational Persistence for Idiomatic .NET. 2016, [cit. 2016-12-27]. Dostupné z: http://nhibernate.info/doc/nh/en/index.html

[31] Oystein Bjorke: OxyPlot. 2016, [cit. 2016-12-27]. Dostupné z: http://www.oxyplot.org

[32] Paessler AG: PRTG Manual List of Available Sensor Types. 2016, [cit. 2016-12-27]. Dostupné z: https://www.paessler.com/manuals/prtg/available_sensor_types

[33] Paessler AG: PRTG Network Monitor download. 2016, [cit. 2016-12-27]. Dostupné z: http://www.paessler.com/prtg/download

[34] Paessler AG: PRTG Network Monitor Powerful Network Monitoring Software. 2016, [cit. 2016-12-27]. Dostupné z: https://www.paessler.com/prtg

[35] Pivotal Software, Inc.: RabbitMQ Documentation. 2016, [cit. 2016-12-27]. Dostupné z: https://www.rabbitmq.com/documentation.html

[36] René Bos: Monitoring done differently. A closer look at PRTG Network Monitor. 2015, [cit. 2016-12-27]. Dostupné z: https://snowvm.com/2015/01/08/monitoring-done-differently-a-closer-look-at-prtg/

[37] Rob Boucher Jr., Andy De George, Saurabh Bhatia: How to Monitor Cloud Services. 2015, [cit. 2016-12-27]. Dostupné z: https://docs.microsoft.com/en-us/azure/cloud-services/cloud-services-how-to-monitor

[38] Simon Ng: What You Need to Begin iOS Programming. 2012, [cit. 2016-12-27]. Dostupné z: http://www.appcoda.com/what-you-need-to-begin-ios-programming/

[39] SQLite Consortium: About SQLite. 2016, [cit. 2016-12-27]. Dostupné z: https://sqlite.org/about.html

[40] Trigger.io: Why Trigger.io doesn't use PhoneGap 5x faster native bridge. 2012, [cit. 2016-12-27]. Dostupné z: http://trigger.io/cross-platform-application-development-blog/2012/02/24/why-trigger-io-doesnt-use-phonegap-5x-faster-native-bridge/

[41] Trigger.io: How it works - Trigger.io. 2016, [cit. 2016-12-27]. Dostupné z: https://trigger.io/how-it-works/

[42] Troelsen, Andrew, Philip Japikse: *C# 6.0 and the .NET 4.6 Framework*. Apress, 2015, ISBN 978-1-4842-1333-9.

[43] Wikimedia Foundation, Inc.: Load balancing (computing). 2016, [cit. 2016-12-27]. Dostupné z: https://en.wikipedia.org/wiki/Load_balancing_(computing)

[44] Wikimedia Foundation, Inc.: Object-relational mapping. 2016, [cit. 2016-12-27]. Dostupné z: https://en.wikipedia.org/wiki/Object-relational_mapping

[45] Xamarin Inc.: Build a Native Android UI and iOS UI with Xamarin.Forms. 2016, [cit. 2016-12-27]. Dostupné z: https://www.xamarin.com/forms

[46] Xamarin Inc.: Mobile Application Development to Build Apps in Xamarin. 2016, [cit. 2016-12-27]. Dostupné z: https://www.xamarin.com/platform

[47] Xamarin Inc.: Store Xamarin. 2016, [cit. 2016-12-27]. Dostupné z: https://store.xamarin.com

Příloha **A**

Seznam použitých zkratek

API Application Programming Interface

BLOB Binary Large OBject

DLL Dynamic Linked Library

HTTP Hypertext Transfer Protocol

HTTPS HTTP Secure

IDE Integrated development environment

IIS Internet Information Services

MVC Model View Controller

MVVM Model View ViewModel

ORM Object-relational mapping

REST Representational state transfer

SDK Software Development Kit

SNMP Simple Network Management Protocol

SOAP Simple Object Access Protocol

SSL Secure Sockets Layer

TCP Transmission Control Protocol

URL Uniform Resource Locator

URI Uniform Resource Identifier

WCF Windows Communication Foundation

WMI Windows Management Instrumentation

Příloha B

Návod na nasazení systému

Tento návod popisuje nasazení serverové části, uživatelských klientských aplikací a integraci do monitorovaných služeb.

B.1 Serverová část

Minimální prerekvizity:

- Windows Server 2012
- Microsoft IIS 8.0
- Microsoft SQL Server 2012
- .NET Framework 4.5 s povoleným WCF HTTP Activation

Vytvoření databáze:

1. Vytvořte databázi s názvem RSM
2. Spusťte postupně všechny databázové skripty ze složky bin\Server\Database

B.1.1 Monitorovací služba

1. Vytvořte v IIS webovou službu pod názvem RSM.ManagementService a běžící v .NET Frameworku 4.5
2. Nakopírujte do umístění vytvořené služby soubory ze složky bin\Server\ManagementService
3. V souboru RSM.ManagementService.Container.config upravte:

a) Connection string pro přístup k databázi

b) V případě potřeby změňte umístění lokálního BLOB uložiště

4. Spusťte webovou službu v IIS.

B.1.2 Notifikační služba (volitelná část)

Notifikační služba je volitelná část, která není nutná pro správnou funkčnosti systému. Přidává pouze alternativní možnost aktualizace dat metrik pro mobilní aplikace.

Minimální prerekvizity:

- Apache ActiveMQ 5.14.3

Postup nasazení notifikační služby:

1. Zkopírujte soubory ze složky bin\Server\DataUpdateService.
2. V souboru RSM.DataUpdateService.Container.config upravte adresu ActiveMQ serveru.
3. Nainstalujte a spusťte RSM.DataUpdateService.exe jako Windows službu.

Postup nasazení služby monitorující nová data:

1. Zkopírujte soubory ze složky bin\Server\DataUpdaterWorkerService.
2. V souboru App.config upravte:
   a) Adresu ActiveMQ serveru.
   b) Adresu monitorovací služby.
3. Nainstalujte a spusťte RSM.DataUpdaterWorkerService.exe jako Windows službu.

B.2 Monitorovaná služba

V této části je popsáno jak nasadit monitorovanou službu a její závislosti.

B.2.1 Report služba

Report službu je nutné nasadit na stejný stroj, kde bude běžet monitorovaná služba. Postup nasazení:

1. Nainstalujte Report službu pomocí instalátoru (nutné spustit s administrátorskými právy) z bin\ReportService\Installer.msi.
2. V umístění instalace Report služby upravte soubor RSM.ReportService.exe.config kde v sekci system.serviceModel/client změňte adresu Monitorovací služby na skutečnou.
3. Ve Windows správci služeb spusťte službu s názvem Report service.

B.2.2 Monitorovaná služba nebo aplikace

Spusťte monitorovanou službu nebo aplikaci využívající klient framework. Integrace klientského frameworku je popsáno v příloze C.

B.3 Webový klient

Postup nasazení:

1. Vytvořte v IIS webovou aplikaci pod názvem RSM.WebClient a běžící v .NET Framworku 4.5.
2. Nakopírujte do umístění vytvořené webové aplikace soubory ze složky. bin\Client\WebClient.
3. V souboru Web.config v sekci system.serviceModel/client upravte adresu Monitorovací služby a skutečnou.
4. Spusťte webovou aplikaci v IIS.

B.4 Mobilních klienti

B.4.1 Android

Postup nasazení:

1. Povolte v Android zařízení instalaci z neznámých zdrojů.
2. Zkopírujte soubor z bin\Client\Android\AndroidClient.apk do zařízení a nainstalujte.

3. Po spuštění aplikace otevřete její nastavení a upravte adresy serverových služeb podle skutečnosti.

B.4.2 iOS

Přímá instalace aplikace aniž by musela být publikovaná na App Storu není jednoduše možná. Nasazení je tedy nutné provést přes vývojářské nástroje. Je nutné mít nainstalovaný tento software:

- Visual Studio 2015
- OS X 10.11 (nutné vlastnit zařízení Apple iMac nebo MackBook)
- Nainstalovaný Xamarin na Windows a Apple zařízení

Pro nasazení aplikace na fyzické zařízení (netyká se simulátoru) jsou navíc nutné ještě další prerekvizity:

- Vlastnit vývojářský Apple Developer účet
- Mít vytvořený v účtu pro aplikaci AppId a provising profile namapovaný na testovací fyzické zařízení

Postup nasazení:

1. Otevřete zdrojové kódy projektu ve Visual Studiu přes soubor src\Project\Solution\RSM.sln.
2. Označte projekt RSM.MobileClient.iOS jako startup project.
3. Vyberte zařízení pro nasazení.
4. Proveďte rebuild a deploy projektu.
5. Po spuštění aplikace otevřete její nastavení a upravte adresy serverových služeb podle skutečnosti.

B.4.3 Windows

Windows aplikaci je možné spustit pouze na Windows 10 nebo Windows 10 Mobile. Proto je zde zmíněn postup pouze pro tyto dvě verze operačního systému.

B.4.3.1 Windows 10 Mobile

Přímá instalace aplikace aniž by musela být publikována na Microsoft Store není na Windows 10 Mobile jednoduše možná. Nasazení je nutné provést přes vývojářský software Visual Studio 2015. Postup je následující:

1. Otevřete zdrojové kódy projektu ve Visual Studiu přes soubor src\Project\Solution\RSM.sln.
2. Označte projekt RSM.MobileClient.UWP jako startup project.
3. Vyberte zařízení pro nasazení.
4. Proveďte rebuild a deploy projektu.
5. Po spuštění aplikace otevřete její nastavení a upravte adresy serverových služeb podle skutečnosti.

B.4.3.2 Windows 10

Lze použít stejný postup jako u Windows 10 Mobile v předešlé části. Navíc u stolní verze Windows 10 existuje ještě alternativní jednoduší postup přímé instalace:

1. Na zařízení povolte vývojářský režim.
2. Rozbalte soubor bin\Client\Windows\WindowsClient.zip.
3. Spusťte s administrátorskými právy skript Add-AppDevPackage.ps1.
4. Ve skriptu potvrďte instalaci certifikátu a aplikace.
5. Po spuštění aplikace otevřete její nastavení a upravte adresy serverových služeb podle skutečnosti.

Příloha C

Návod na integraci klientského frameworku

Pro použití klientského frameworku stačí do vlastní aplikace nalinkovat knihovnu ze složky bin\ClientFramework\RSM.ClientFramework.dll. Veškerý přístup probíhá přes následně statickou třídu RsmClient, která obsahuje tyto metody a vlastnosti:

Název	**Popis**
ConfigurationManager	Přístup ke konfiguraci instance.
LogManager	Přístup k logování.
MetricsManager	Přístup k metrikám.
ReportServiceUri	URL adresa Monitorovací služby. Je nutné nastavit před jakýmkoliv voláním.
IsInitialized	Indikace zdali byl framework úspěšně inicializován.
RegisterInstance	Registrace nové instance do monitorování. Vrátí přidělené ServiceInstanceId.
Intialize	Inicializace frameworku. Je nutné použít přidělené ServiceInstanceId.

C.1 Inicializace frameworku

Postup inicializace:

1. Nastavte správnou adresu pro Monitorovací službu přes vlastnost ReportServiceUri. Tato vlastnost není nikde ukládána. Je tedy nutné ji

nastavit při každém spuštění instance znovu.

2. Při prvním spuštění musíte zavolat metodu RegisterInstance, která provede registraci a vrátí přidělené ServiceInstanceId. Toto Id se bude používat po celou dobu existence instance.

3. Zavolejte metodu Initialize s přiděleným ServiceInstanceId.

Po provedení postupu je klient framework úspěšně inicializován a je možné využívat všechny jeho metody.

C.2 Metriky

Přístup k metrikám je možný přes třídu MetricsManager. Instanci třídy lze získat přes třídu RsmClient. MetricsManager poskytuje tyto metody:

Název	**Popis**
EnableResponseTimeMonitoring	Zapnutí monitorování doby odezvy na zadanou URL adresu.
DisableResponseTimeMonitoring	Vypnutí monitorování doby odezvy.
IsResponseTimeMonitoringEnabled	Ověření stavu monitorování doby odezvy.
RegisterCustomMetric	Registrace vlastní metriky. Je nutné specifikovat unikátní identifikátor a zobrazovaný název metriky.
IsCustomerMetricRegistered	Ověření zdali metrika se zadaným unikátním identifikátorem již existuje.
SendCustomMetricValue	Odeslání naměřené hodnoty vlastní metriky.

C.3 Logování

Přístup k logování je dostupný přes třídu LogManager, jejíž instanci lze získat v třídě RmClient. LogManager poskytuje jedinou metodu a to je metoda Log pro vytvoření nového záznamu události v logu. Součástí frameworku je také integrace do logovacích frameworků log4net a NLog.

C.3.1 log4net

Pro použití integrace je nutné do vlastního projektu nalinkovat navíc knihovnu bin\Framework\RSM.ClientFramework.Log4Net.dll. Knihovna přidává nový appender RsmLog4NetAppender.

C.3.2 NLog

Pro použití integrace je nutné do vlastního projektu nalinkovat navíc knihovnu bin\Framework\RSM.ClientFramework.NLog.dll. Knihovna přidává nový target RsmNLogTarget.

C.4 Konfigurace

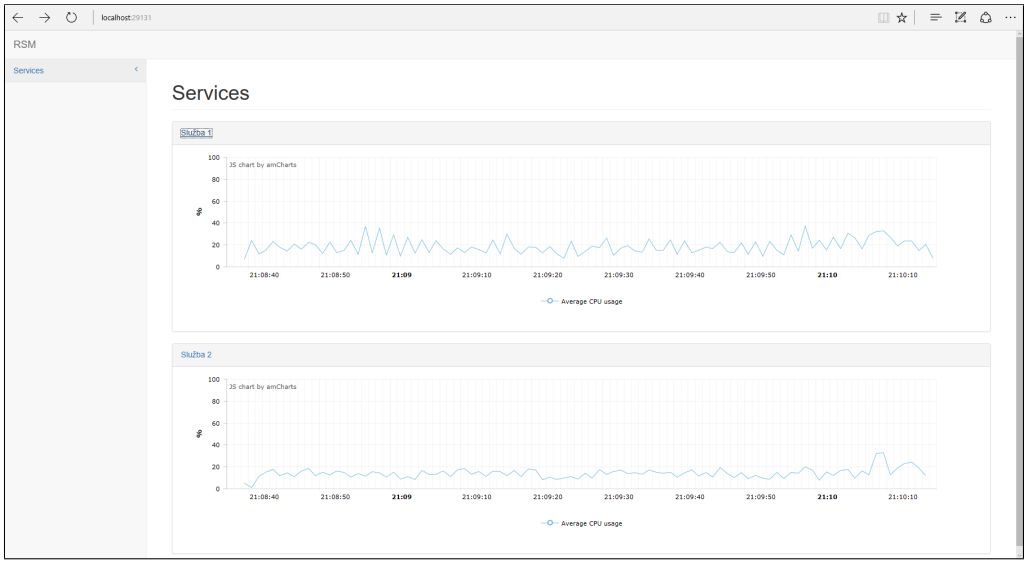
Přístup ke konfiguraci je možný skrze třídu ConfigurationManager. Instanci této třídy lze získat přes třídu RsmClient. ConfigurationManager poskytuje tyto metody:

Název	**Popis**
ContainsParameter	Ověření zdali konfigurace obsahuje hodnotu pro zadaný klíč.
GetParameter	Získání hodnoty z konfigurace podle zadaného klíče.
ConfigurationChanged	Event handler, který je zavolán v případě změny konfigurace.

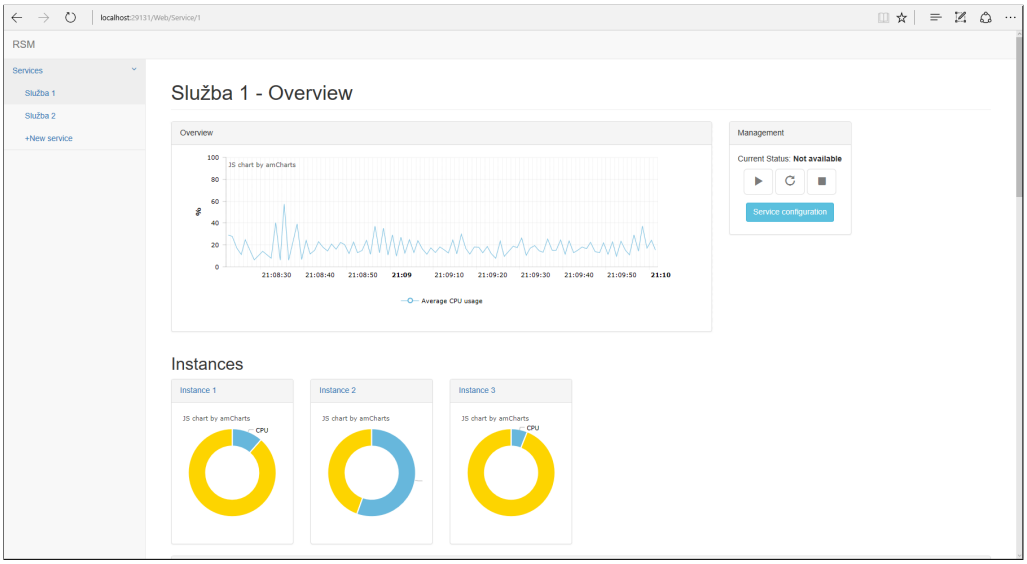
Příloha D

Snímky obrazovky klientských aplikací

D.1 Webový klient



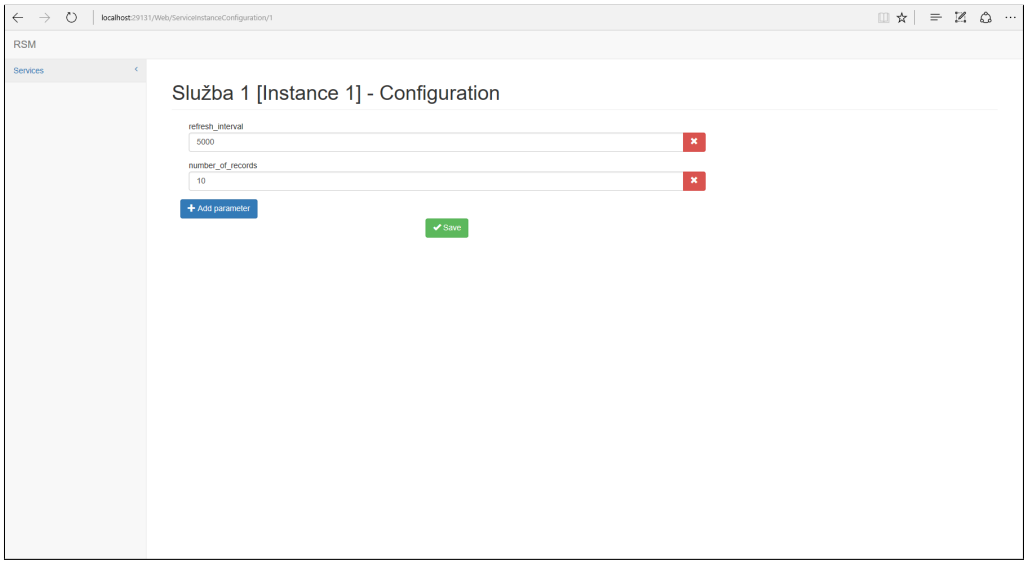
Obrázek D.1: Obrazovka přehledu monitorovaných služeb



Obrázek D.2: Obrazovka detailu monitorované služby

Obrázek D.3: Obrazovka detailu monitorované instance služby

Obrázek D.4: Obrazovka detailu monitorované instance služby

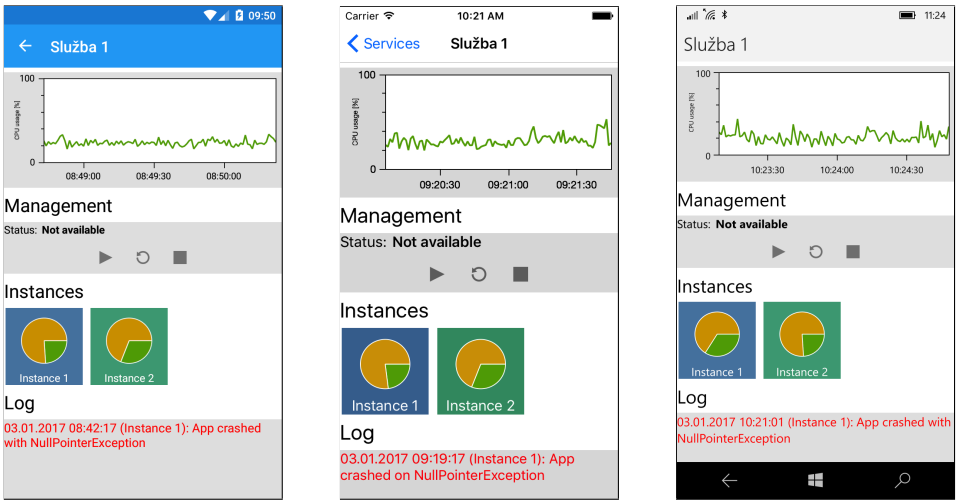


Obrázek D.5: Obrazovka konfigurace instance služby

D.2 Mobilní klient

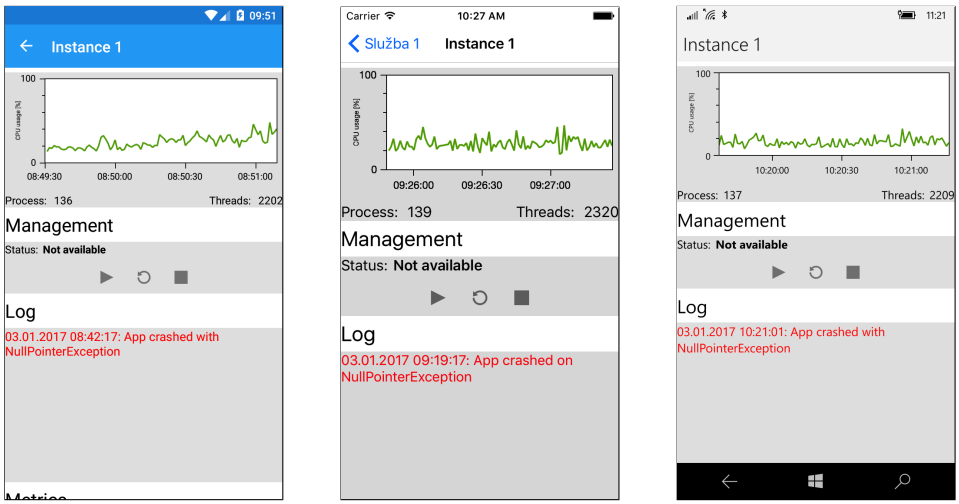
(a) Android (b) iOS (c) Windows

Obrázek D.6: Obrazovka přehledu monitorovaných služeb



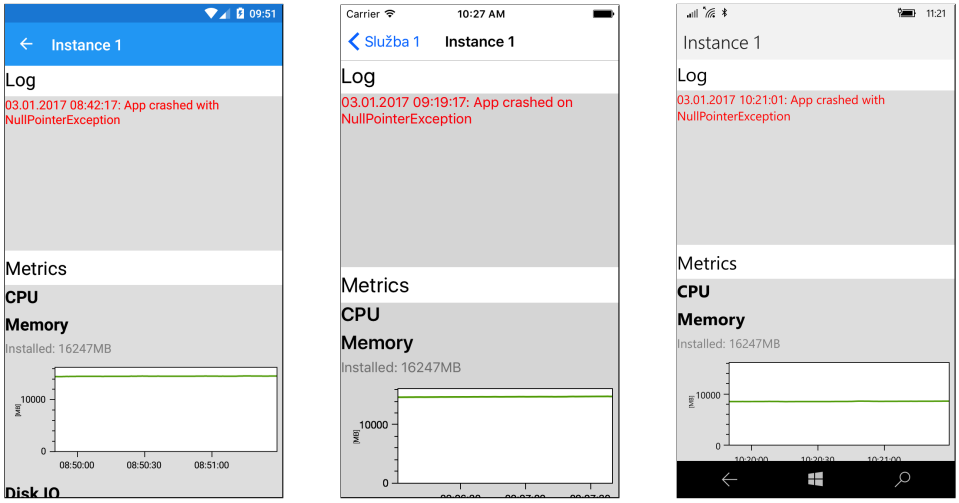
(a) Android (b) iOS (c) Windows

Obrázek D.7: Obrazovka detailu monitorované služby



(a) Android (b) iOS (c) Windows

Obrázek D.8: Obrazovka detailu monitorované instance služby



(a) Android (b) iOS (c) Windows

Obrázek D.9: Obrazovka detailu monitorované instance služby

Příloha E

Obsah přiloženého CD

- `Readme.txt` ..... stručný popis obsahu CD
- `bin` ..... binární a spustitelné soubory
  - `Server` ..... soubory serverové části soubory
    - `ManagementService` ..... soubory Monitorovací služby
    - `DataUpdateService` ..... soubory notifikační služby
    - `DataUpdaterWorkerService` .. soubory služby monitorující nová data
  - `ReportService` ..... instalátor Report služby
  - `ClientFramework` ..... klientský framework
  - `Client` ..... soubory klientských aplikací
    - `WebClient` ..... soubory webového klienta
    - `Android` ..... spustilená klientská Android aplikace
    - `Windows` ..... spustilená klientská Windows aplikace
- `src` ..... zdrojové kódy
  - `Project` ..... zdrojové kódy implementace
  - `Thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `DIP_Lysak_Adam_2017.pdf` ..... text práce ve formátu PDF