Dohledový systém serverů

Server monitoring system

Bc. Monika Křupalová

Diplomová práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

*** nascannované zadání str. 1 ***

Motivace k monitorování webových serverů.

\item{} Rešerše metod monitorování serverů.

\item{} Návrh dohledového systému pro HTTP, MS SQL, FTP.

\item{} Realizace prototypu.

*** nascannované zadání str. 2 ***

ABSTRAKT

Hlavním cílem této diplomové práce je vytvoření systému pro vzdálený monitoring serverů a jejich webových služeb. Výsledná aplikace se bude automaticky dle nastavených intervalů kontrolovat dostupnost jednotlivých služeb a zaznamenávat jejich stavy do databáze. Dále bude možno zobrazovat různé statistiky o dlouhodobém stavu služeb. V první části práce jsou popsány softwarové technologie, které byly použity během vývoje aplikace. Dále je obsažena motivace pro monitoring serverů a popis možností, jak lze servery monitorovat. Praktická část je zaměřena zejména na výslednou aplikaci a její součásti. Práci uzavírá náhled do možného budoucího vývoje aplikace a popis možností implementace v praxi.

Klíčová slova: .NET, ASP.NET, LINQ, MS SQL Server, monitoring, serverová služba

ABSTRACT

Main idea of this thesis is the monitoring system which will check availability of remote servers and it's services. Whole application will be automatically checking all services in selected periods and will log their state to database. This application will be also providing web statistics which includes states of services. In first part of this thesis are described technologies used to develop this application and detailed informations about monitoring servers and their services. Second part of this thesis (practical part) describes final application, it's appearance and it's components. At the end of this thesis we will show possible options about implementation in future.

Keywords: .NET, ASP.NET, LINQ, MS SQL Server, monitoring, server service

Ráda bych poděkovala vedoucímu mé diplomové práce Ing. Petru Šilhavému za cenné rady připomínky a čas, který mi věnoval při konzultacích nad problémy řešení této diplomové práce.

Také bych chtěla poděkovat svým nejbližším a přátelům, kteří mi byli během psaní práce velkou podporou.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….

podpis diplomanta

OBSAH

ÚVOD

Peníze a dostupnost, to jsou slova, která v technických oborech v poslední době slýcháváme stále častěji. Pryč totiž doby, kdy výpadek serveru nebo zařízení nijak neovlivnil nebo ovlivnil pouze nepatrně chod firmy nebo nadnárodní korporace.

Každá minuta odstávky služby nebo kompletního výpadku daného serveru může stát firmu nemalou ztrátu, často v řádu tisíců až stovek tisíců. Pokud si představíme komplexní internetový obchod zaběhlé firmy, může se obrat v daném měsíci pohybovat až v řádu miliónu, kdy každá nepřijatá objednávka nebo ztracený zákazník může firmu poškodit a znamenat ztrátu. Taková ztráta se nemusí projevit ihned, ale až s postupem času.

I přesto, že v dnešní moderní době jsou kritické aplikace většinou řešeny formou clusteringu (například fail-over clustering) nebo load-balancingu, může výpadek jednoho ze strojů do jisté míry způsobit komplikace. V tuto chvíli přichází na řadu aplikace monitorující dostupnost dané služby s možností okamžitého upozornění.

Na řadu přicházejí aplikace, které provádí proaktivní monitoring, který nejenže dokáže chyby detekovat, až jakmile nastanou, ale dokáží také dle chování služby (posloupnosti výpadků a dostupnosti) vyhodnotit stav, kdy služba teprve nedostupná bude.

Cílem této diplomové práce je v základním měřítku takovou aplikaci vytvořit. Aplikace se bude skládat z webové části umožňující zobrazovat aktuální stav monitorovaných služeb, možnost přidávat nové služby, zobrazovat statistiky apod. Druhou částí bude samotná systémová služba, která bude v pravidelných intervalech kontrolovat dostupnost služeb a zaznamenávat ji do databáze.

I. TEORETICKÁ ČÁST

1 MOTIVACE K MONITOROVÁNÍ WEBOVÝCH SERVERŮ

Žádný web nefunguje nepřetržitě a bez výpadků. Potřebuje pravidelné kontroly a údržbu. Pokud má vlastník webu vlastní server, hosting, síť nebo jen www stránku, monitoring správného chodu služeb dnes není jen nadstandardem, ale nutností.

Automatizované monitorování systému a hardwaru je jedna z vlastností, která by neměla chybět na žádném profesionálně udržovaném serveru. Upozorní nás včas na případné problémy, které by mohly často vést i k porušení hardware celého serveru. Díky automatizovanému monitoringu poznáte například výpadek serveru včas a poté můžete brzy odhalit jeho příčinu. Jakmile monitorovací systém detekuje chybu, informuje ihned administrátora pomocí emailu nebo SMS.

Monitoring serverů vám zaručí, že vaše stránky, ale také e-mailové a další síťové služby budou dostupné a budou fungovat tak, jak mají.

Zaručí, že investice do rozvoje a zvyšování návštěvnosti vašich webových stránek budou skutečně efektivní a přinesou vám maximální zisk. Chyby identifikujete a opravíte ještě předtím, než je zaznamenají uživatelé (nebo vaši nadřízení) a návštěvníky služeb navíc ušetříte frustrujících okamžiků spojených s nedostupností.

Proč monitorovat server:

- Dozvíte se včas o výpadku serveru, jeho služby či webu a můžete sjednat okamžitou nápravu.
- Zjistíte procento dostupnosti.
- Vícebodový systém monitoringu odhalí možné problémy připojení k vašemu webu.

Konkurence na světovém i českém internetu se stále zostřuje, a řada firem proto vynakládá mnoho času i prostředků, aby na svůj web přilákaly návštěvníky. Pokud je ale internetový server mnohdy několik hodin mimo provoz, jsou investice do zvýšení návštěvnosti téměř zbytečné.

Pokud funguje připojení k vašim stránkám naprosto dokonale z vašeho domácího i pracovního počítače, je to dobrá zpráva, ale jak je to s návštěvníky z jiných koutů našeho státu, či dokonce z ciziny, to tímto testem zjistit nelze. Pomocí více měřicích bodů z různých umístění lze kontrolovat, zda jsou vaše služby dostupné z různých typů připojení a nejen z oblastí České republiky, ale také z ciziny.

Změřenou dostupnost služby by měla monitorovací služba zaznamenávat a zobrazovat stavy v denních, měsíčních a ročních grafech s možností upřesnění každého výpadku (kdy k němu došlo, o jaký výpadek se jednalo).

Zpravidla lze také nastavit interval měření. Platí úměra, že čím kratší čas, tím přesnější výsledky a rychlejší zjištění případného problému. Nabízené intervaly bývají od 10 až po 15 vteřin. Zpravidla je ale 1 minuta považována za plně dostačující interval.

1.1 Jaké testy pro servery a služby například získáte?

Pomocí monitorovacích systémů lze zajišťovat testy pro monitorování například Exchange Server, MS SQL, Oracle a ODBC databáze, využití CPU, FTP & HTTP servery, Active Directory, stav diskové jednotky, místo na disku, Event Log (s kontrolou obsahu), existence souboru (s kontrolou obsahu), TCP & ICMP Ping, SMTP & POP3 Mail servery, tiskárny, procesy, služby & démoni, skripty pro UNIX Shell, SNMP & terminálový server.

2 REŠERŠE METOD MONITOROVÁNÍ SERVERŮ

Dodnes mají někteří administrátoři vžitou představu, že monitorování znamená vzdálený přístup k serverům a sledování jejich event logů. Kdysi tomu tak bylo. U serverů se v event logu sledoval stav vybraných parametrů a zjišťovalo se, zda nejsou v event logu varovné hlášení apod. Pokud HW byl vypnutý, tak žádná vzdálená správa nebyla možná. Před cca 10 lety technologie serverů umožňovala vzdálený přístup, aniž by server byl zapnut. Na dálku šlo zapnout/vypnout a resetovat server.

Na trhu však existuje mnoho nástrojů, které možnost monitorování služeb zajistí za nás. Také je zde možnost vytvořit vlastní aplikaci pro automatickou kontrolu dostupnosti služeb pomocí libovolného programovacího jazyka.

2.1 Zdarma nebo za peníze

Většina na internetu dostupných aplikací není zdarma, a pokud ano, mají jen omezenou funkčnost nebo nemají komplexní využití pro více typů služeb. Nelze tím obecně říci, že volně dostupná řešení jsou špatná, ale pokud máme více serverů a služeb, které chceme sledovat, musíme si připlatit.

Tento stav je tedy i živnou půdou pro další komerční programy, které celý monitoring služeb zajistí. Ceny komerčních nástrojů zpravidla lákavě začínají na malých částkách. Tyto verze podporují buď jen tu nejzákladnější funkčnost, nebo mají mnohá omezení a jsou spíše koncipovány jako verze k vyzkoušení a jako první krůček k verzím placeným. Taktéž se ceny často odvíjejí podle počtu sledovaných služeb nebo od intervalů, ve kterých potřebujeme službu kontrolovat.

2.2 Vlastnosti dostupných řešení

Dostupná řešení mají většinou podobnou strukturu dohledového systému. Jedná se o více měřících bodů, kdy aplikace běží na více místech najednou. Tyto body se většinou nachází nejen na různých místech našeho státu, ale také mimo něj.

2.3 Monitoring ze serverů poskytovatele služby

Monitoring je většinou zajištěn přes servery firmy, u kterých jsme služby zakoupili, a administrátor serveru již jen nastavuje parametry služeb, které potřebuje kontrolovat.

Interval měření se pohybuje v řádech několika minut až několika sekund. Čím menší interval administrátor požaduje, tím více cena za tuto službu šplhá do závratných výšin.

Většina komerčních produktů poskytuje více měřených služeb včetně upozornění o chybách posílaném přes email či SMS.

Dále bývá dostupný log dosavadních kontrol u dané služby až s roční historií a z nich generované statistiky.

2.4 Monitoring z vlastních serverů

Dostupná jsou také řešení formou software, který nainstalujeme na vlastní server. Kontroly poté probíhají jen ze serverů, na kterých tento software nainstalujeme.

Zde bývají oproti předchozí variantě ceny nižší a toto řešení je tedy dostupnější. Musíme však zajistit běh programu na serveru, který by měl být umístěn mimo infrastrukturu sledovaných služeb, aby byla zajištěna kontrola zvenčí.

Struktura řešení je zde podobná, avšak pokud potřebujeme sledovat více služeb najednou, nemusíme za tuto možnost většinou připlácet.

3 MICROSOFT VISUAL STUDIO

Microsoft Visual Studio je vývojové prostředí (IDE) od Microsoftu. Může být použito pro vývoj konzolových aplikací a aplikací s grafickým rozhraním spolu s Windows Forms aplikacemi, webovými stránkami, webovými aplikacemi a webovými službami jak ve strojovém kódu, tak ve spravovaném kódu na platformách Microsoft Windows, Windows Mobile, Windows CE, .NET, .NET Compact Framework a Microsoft Silverlight.

Visual Studio obsahuje editor kódu podporující IntelliSense a refaktorování. Integrovaný debugger pracuje jak na úrovni kódu, tak na úrovni stroje. Další vestavěné nástroje zahrnují designer formulářů pro tvorbu GUI aplikací, designer webu, tříd a databázových schémat. Je možné přidávat rozšíření, což vylepšuje funkčnost na téměř každé úrovni – od přidání podpory pro verzovací systémy (jako Subversion a Visual SourceSafe) do přidání nových nástrojů jako editory a vizuální designery pro jazyky specifické pro obor nebo nástroje pro další aspekty návrhu programu (jako klient Team Foundation Serveru: Team Explorer).

Visual Studio podporuje jazyky prostřednictvím jazykových služeb, což umožňuje, aby editor kódu a debugger podporoval jakýkoliv programovací jazyk. Mezi vestavěné jazyky patří C/C++ (použitím Visual C++), VB.NET (použitím Visual Basic .NET) a C# (použitím Visual C#). Podpora dalších jazyků jako Chrome, F#, Python a Ruby spolu s ostatními může být přidána jazykovými službami, které musí být nainstalovány zvlášť. Také je podporováno XML/XSLT, HTML/XHTML, JavaScript a CSS. Existují i verze Visual Studia pro určitý jazyk, které uživateli poskytují omezenější jazykové služby. Tyto individuální balíčky jsou Microsoft Visual Basic, Visual J#, Visual C# a Visual C++.

3.1 Architektura

Visual Studio nepodporuje žádný programovací jazyk nebo nástroj samo o sobě. Místo toho je mu možno přidat různá rozšíření funkčnosti. Každá funkčnost je zabalena do balíčku VSPackage. Když je nainstalována, je dostupná jako služba.

IDE poskytuje tři služby:

- SVsSolution – umožňuje očíslovat projekty a sestavy
- SVsUIShell – poskytuje rozdělování na okna a UI funkce (jako panely, nástrojové lišty a okna nástrojů)

- SVsShell – stará se o registraci balíčků VSPackage

IDE je také odpovědné za koordinaci služeb a umožnění komunikace mezi nimi. Všechny editory, designery, typy projektů a další nástroje jsou implementovány jako balíčky VSPackage. Visual Studio používá COM pro přístup k balíčkům. SDK také obsahuje Managed Package Framework (MPF), což je sada spravovaných obalů okolo COM-rozhraní, které umožňují, aby mohly být balíčky psány v .NET jazycích. Nicméně, MPF neposkytuje veškerou funkčnost představenou COM rozhraními Visual Studia. Služby mohou být použity pro tvorbu dalších balíčků, což přidá funkčnost do IDE Visual Studia.

Podpora programovacích jazyků je přidána balíčkem zvaným Language Service (jazyková služba). Jazyková služba definuje různá rozhraní, která může implementace VSPackage implementovat pro přidání podpory různé funkčnosti. Funkčnost, která může být tímto způsobem přidána, zahrnuje zvýraznění syntaxe, doplňování příkazů, zvýrazňování párů závorek, tipy parametrů informací, seznamy členů a chybové značky pro kompilaci na pozadí. Pokud je rozhraní implementováno, funkčnost bude pro jazyk dostupná. Implementace jazykových služeb mohou použít kód z překladače nebo editoru jazyka. Jazykové služby mohou být implementovány jak ve strojovém kódu, tak ve spravovaném kódu. Pro strojový kód mohou být použity jak COM rozhraní, tak Babel Framework (součást SDK). Pro spravovaný kód obsahuje MPF obaly pro psaní spravovaných jazykových služeb.

Visual Studio neobsahuje žádnou vestavěnou podporu verzování, ale definuje MSSCCI (Microsoft Source Code Control Interface) implementováním toho, které verzovací systémy mohou být integrovány s IDE. MSSCCI definuje sadu funkcí, které jsou použity pro implementaci různé funkčnosti verzování. MSSCCI bylo poprvé použito pro integraci Visual SourceSafe s Visual Studiem 6.0, ale později bylo zpřístupněno přes Visual Studio SDK. Visual Studio .NET 2002 používalo MSSCCI 1.1 a Visual Studio .NET 2003 používalo MSSCCI 1.2. Visual Studio 2005 i 2008 používají MSSCCI verze 1.3, která přidává podporu přejmenovávání a mazání propagace, stejně jako asynchronní otevírání. Studio 2010 používá MSSCCI verze 3.3.

Visual Studio podporuje spuštění více instancí prostředí (každé s vlastní sadou balíčků). Instance používají jiné větve registrů k uložení stavu konfigurace a jsou rozlišeny AppId (Application ID). Instance jsou spouštěny specifickými .exe, které vyberou AppId, nastaví

kořenovou větev a spustí IDE. Balíčky registrované pro jedno AppId jsou integrovány s ostatními balíčky pro toto AppId. Různé edice Visual Studia používají jiná AppId. Produkty edice Visual Studio Express jsou nainstalovány se svými vlastními AppId, ale produkty Standard, Professional a Team Suite sdílí stejné AppId. Proto mohou být Express edice nainstalovány souběžně s ostatními edicemi, narozdíl od ostatních edic, které aktualizují stejnou instalaci. Profesionální edice obsahuje supersadu balíčků odvozené ze standardní edice a Team Suite obsahuje supersadu balíčků odvozené z obou nižších edic. Systém AppId je ovlivněn systémem Visual Studio Shell ve Visual Studiu 2008.

3.2 Prvky IDE

- Editor kódu – Visual Studio, jako každé jiné IDE, obsahuje editor kódu, který podporuje zvýraznění syntaxe a automatické dokončování za použití IntelliSense nejen pro proměnné, funkce a metody, ale také konstrukce jako cykly a dotazy. IntelliSense podporují zahrnuté jazyky, stejně jako XML, CSS a JavaScript při vývoji webových stránek a webových aplikací. Návrhy automatického dokončování se zobrazí v PopUp („vyskakovacím“) seznamu. Ve Visual Studiu 2008 může být dočasně poloprůhledné, abyste viděli kód za ním. Editor kódu je používán pro všechny podporované jazyky.

  Editor kódu Visual Studia také podporuje nastavování záložek v kódu pro rychlou navigaci. Další navigační pomůcky zahrnují sbalování bloků kódu a přírůstkové i normální hledání s podporou regulárních výrazů. Editor kódu také obsahuje vícepoložkovou schránku a seznam úkolů. Editor kódu podporuje snippety, což jsou uložené šablony opakujícího se kódu a mohou být do kódu vloženy a přizpůsobeny aktuálnímu projektu. Nástroj pro správu snippetů je také k dispozici. Tyto nástroje jsou plovoucí okna, která mohou být ukotvena nebo schována při nepoužívání. Editor kódu Visual Studia také podporuje refaktorování včetně změny pořadí parametrů, přejmenování proměnných a metod, extrakci rozhraní a zabalení členů třídy uvnitř vlastností mezi ostatní.

  Visual Studio umožňuje kompilaci na pozadí (někdy zvaná přírůstková kompilace). Jak je kód psán, Visual Studio jej na pozadí kompiluje, aby poskytlo informace o syntaktických a kompilačních chybách, které jsou po jejich detekci podtrženy červenou vlnovkou. Varování jsou podtržena zelenou vlnovkou. Kompilace na

pozadí negeneruje spustitelný kód, protože používá jiný kompilátor než ten, který generuje spustitelný kód. Kompilace na pozadí byla nejdříve dostupná pouze pro Visual Basic, ale nyní podporuje všechny zahrnuté jazyky.

- Debugger – pracuje jak se spravovaným kódem, tak se strojovým kódem a může být použit pro debugování aplikací psaných v jakémkoliv jazyce podporovaném Visual Studiem. Navíc může být přiřazen běžícím procesům a debugovat je. Pokud je dostupný kód běžícího procesu, zobrazí se. Pokud není, zobrazí se kód v Assembleru. Debugger Visual Studia může také vytvořit obsahy paměti a později je nahrát pro debugging. Vícejádrové programy jsou také podporovány. Debugger může být nakonfigurován, aby byl spuštěn, když spadne aplikace běžící mimo Visual Studio.

  Debugger povoluje nastavování breakpointů (které umožňují zastavit běh programu na určité pozici) a watche (které sleduje hodnoty proměnných během procesu). Breakpointy mohou být podmíněné, tedy mohou být aktivovány také pouze v případě, že je splněna určitá podmínka. Kódem lze procházet, tedy spustit jeden řádek kódu najednou nebo můžete vstoupit do funkcí, aby je debugger debugoval uvnitř, nebo je přejít.

  Debugger také podporuje funkci Edit and Continue, takže je možné kód upravovat během debuggingu. Například pokud přejedete přes proměnnou, její aktuální hodnota je zobrazena v tipu, kde může být modifikována. Během kódování umožňuje debugger manuálně zavolat funkci pomocí okna Immediate, kde jsou také k dispozici parametry této funkce.

- WinForms Designer – je používán pro GUI aplikace za použití WinForms, kde je obsažena paleta ovládacích prvků (včetně tlačítek, progress barů, popisek a jiných prvků), které mohou být uchopeny a umístěny na povrch formuláře. Rozložení je možné ovládat ukládáním prvků do kontejnerů nebo uzamykáním na stranu formuláře. Prvky, které zobrazují data (textové pole, rozbalovací pole, tabulka, atd.) mohou být propojeny s datovými zdroji, jako jsou databáze nebo LINQ dotazy. UI je spojeno s kódem event-driven programovacím modelem. Designer vygeneruje buď C# nebo Visual Basic .NET kód pro aplikaci.

- WPF Designer (Cider) – byl představen ve Visual Studiu 2008. Stejně jako WinForms designer podporuje drag-and-drop. Za pomocí WPF Designeru lze vytvořit uživatelské rozhraní pro Windows Presentation Foundation. Podporuje všechny WPF funkce včetně propojení dat a automatickou správu rozložení. Generuje XAML kód pro UI. Vygenerovaný XAML kód je kompatibilní s Microsoft Expression Designem. XAML kód je s kódem spojen code-behind modelem.
- Web designer – Visual Studio také obsahuje editor webových stránek a designer, který je umožňuje vytvářet uchopováním a pokládáním prvků. Je používán pro vývoj ASP.NET aplikací a podporuje HTML, CSS a JavaScript. Používá code-behind model pro spojení s ASP.NET kódem. Zobrazovací engine je sdílen s Microsoft Expression Webem.
- Designer tříd – slouží pro vytváření a úpravu tříd (včetně jejich členů a přístupu) použitím modelu UML. Designer tříd může vygenerovat kódy C# a VB.NET pro třídy a metody. Také může vygenerovat diagramy z ručně psaných tříd.
- Designer dat – může být použit pro grafickou úpravu databázových schémat, včetně psaných tabulek, primárních a cizích klíčů a omezení. Také může být použit pro design dotazů v grafickém zobrazení.
- Designer mapování – od Visual Studia 2008 je designer mapování používán funkcí LINQ to SQL k zobrazení mapování mezi databázovými schématy a třídami, které zabalují data.
- Další nástroje – průzkumníci pro usnadnění práce a editaci vlastností
    - Editor vlastností (Properties Editor) – je použit pro editaci vlastností v GUI panelu uvnitř Visual Studia. Vypíše všechny dostupné vlastnosti (jak nastavitelné, tak jen pro čtení) ve všech objektech, včetně tříd, formulářů, webových stránek a dalších položek.
    - Průzkumník objektů (Object Browser) – průzkumník jmenných prostorů a knihoven tříd pro Microsoft .NET. Může být použit pro prozkoumávání jmenných prostorů (které jsou uspořádány hierarchicky) ve spravovaných

sestavách. Hierarchie může i nemusí odpovídat organizaci v souborovém systému.

- Průzkumník řešení – zde je řešení sada souborů kódu a dalších zdrojů, které jsou použity pro tvorbu aplikace. Soubory v řešení jsou uspořádány hierarchicky, což může i nemusí odpovídat organizaci v souborovém systému. Solution Explorer je použit pro správu a průzkum souborů v řešení.
- Průzkumník týmu (Team Explorer) – je použit pro integraci s možnostmi Team Foundation Serveru, Revision Control Systemu do IDE (a základ pro prostředí Microsoftu CodePlex pro open source projekty). Kromě kontroly kódu poskytuje schopnost vidět a spravovat jednotlivé pracovní položky (včetně chyb, úkolů a dalších dokumentů) a procházet TFS statistiky. Je zahrnut jako součást instalace TFS a je také dostupný ke stažení pro Visual Studio 2005 a vyšší. Team Explorer je také dostupný jako samostatné prostředí pro přístup k TFS službám.
- Průzkumník dat (Data Explorer) – je používán pro správu databází v instancích Microsoft SQL Serveru. Umožňuje vytvářet a upravovat databázové tabulky (použitím Transact-SQL příkazů nebo designeru dat). Také může být použit pro tvorbu dotazů a uložených procedur, buď v Transact-SQL nebo ve spravovaném kódu prostřednictvím SQL CLR. Podpora debuggingu a IntelliSense je také k dispozici.
- Průzkumník serveru – používá se pro správu připojení k databázi na přístupném počítači. Lze s ním také procházet běžící služby Windows, výpočty výkonu, záznamy událostí a zprávy a použít je jako zdroj dat.

3.3 Podporované programovací jazyky

Visual Studio je modulární a ve své podstatě neobsahuje žádný programovací jazyk nebo nástroj. S tímto vývojovým prostředím je ale dodáváno několik produktů, pomocí kterých může vývojář programovat ve zvoleném programovacím jazyce.

- Microsoft Visual C++ – implementace od Microsoftu překladače C a C++ a dalších jazykových služeb a specifických nástrojů pro integraci s Visual Studio IDE. Může

se kompilovat v C nebo C++ módu. Také podporuje C++/CLI specifikaci pro psaní spravovaného kódu, stejně jako kódu v kombinovaném módu (kombinace strojového a spravovaného kódu). Microsoft používá Visual C++ pro vývoj ve strojovém kódu nebo kódu, který obsahuje strojové i spravované komponenty. Dále podporuje COM stejně jako MFC knihovnu. Pro vývoj MFC poskytuje sadu průzkumníků pro vytváření a přizpůsobování standardního MFC kódu a vytváření GUI aplikací pomocí MFC. Visual C++ může také používat designer formulářů Visual Studia pro grafický design UI a může být také použito s Windows API. Také podporuje použití vestavěných funkcí, což jsou funkce rozpoznané přímo překladačem a nejsou implementovány jako knihovna. Vestavěné funkce jsou použity pro použití instrukční sady SSE v moderních procesorech. V neposlední řadě Visual C++ obsahuje také specifikaci OpenMP (verze 2.0).

- Microsoft Visual C# – implementace jazyka C# od Microsoftu, která je cílena na .NET Framework spolu s jazykovými službami, které Visual Studio IDE podporuje. Zatímco jsou jazykové služby součástí Visual Studia, překladač je k dispozici odděleně jako součást .NET Frameworku. Visual C# podporuje kromě ostatních designer tříd, designer formulářů nebo designer dat.

Microsoft Visual Basic – implementace jazyka VB.NET a příslušných nástrojů a jazykových služeb od Microsoftu. Byl představen ve Visual Studiu .NET (2002) a používá Visual Basic pro rychlý vývoj aplikací. Visual Basic může být použit pro vytváření konzolových i GUI aplikací. Stejně jako Visual C#, Visual Basic také podporuje mimo jiné designer tříd, designer formulářů nebo designer dat. Jako C#, kompilátor VB.NET je také k dispozici jako součást .NET Frameworku. Ale jazykové služby, které VB.NET projektům umožňují být vytvářeny Visual Studiem, jsou dostupné jako součást Visual Studia. [1]

3.4 Microsoft Visual Studio 2010

Větších či menších novinek je opravdové množství, nebudeme se proto zaměřovat na plné obsažení všeho nového.

3.4.1 Vylepšené prostředí

První a největší novinkou je zřejmě kompletní přepracování Visual Studia do WPF (Windows Presentation Foundation). WPF umožňuje editoru bohatě prezentovat informace o kódu v kontextu celého zdroje.

Tato schopnost také dovoluje třetím stranám vytvářet rozšíření, která ukazují vlastní zobrazení zdrojového kódu, jako vzít komentáře z XML dokumentu a převést je na formátovaný text s písmy, barvami a zvýrazňovači. Také Visual Studio umožňuje zobrazit různé vrstvy editoru, takže některé rozšíření může vzít vzorec z kódu a převést jej na jeho přirozenou matematickou reprezentaci.

Během reprezentace zdrojového kódu je důležité rozumět tomu, co kód vlastně dělá. K tomu slouží nové prvky Visual Studia 2010 jako Inline Call Hierarchy, umožňující vývojáři vidět, kde se používá vybraná vlastnost nebo metoda, a snáze tak pochopit interakci kódu, aniž by musel otevírat několik souborů. [5]

Další novinkou spjatou s WPF je přepracování úvodní obrazovky. Ta je nyní kompletně definovaná v XAMLu, čili XML formátu pro definování WPF. Praktickým důsledkem je to, že Home screen je teď mnohem flexibilnější a umožňuje kompletní přizpůsobitelnost.

Další urychlení naší programátorské práce také přinesou Code Snippets v ASP.NET stránkách.“Útržky kódu“, jak by asi zněl český překlad, již v předešlých verzích VS byly. Umožňují skrze klávesovou zkratku vložit kousky kódu buďto předpřipravené od vývojářů Microsoftu nebo vlastní. Doposud však tato funkcionalita chyběla pro editaci html(ASP) kódu.

Další ze zajímavých novinek je možnost exportovat a importovat definované breakpointy. Pokud tak pomáháte kolegovy s laděním kódu, můžete u sebe označit kritická místa breakpointy a poté mu výsledek předat k načtení do jeho Visual Studia.

Jedním ze způsobů, jak bylo Visual Studio vždy rozšiřováno, byli Extensions. Čili přídavky do Visual Studia třetích stran. Jedním z nejznámějších extensions je například Re-Sharper.

V nové verzi Visual Studia bude usnadněn vývoj těchto rozšíření skrze SDK. Běžný vývojář to zajisté nevyužije, díky snadnému vývoji však můžeme očekávat záplavu nových rozšíření, které nám – běžným programátorům – práci nakonec přeci jen usnadní. Do Visual Studia bude také integrován přístup k www.visualstudiogallery.com, což je online

repositář rozšíření. Bude tak možné stahovat a objevovat nová užitečná rozšíření přímo z Visual Studia.

3.4.2 UML

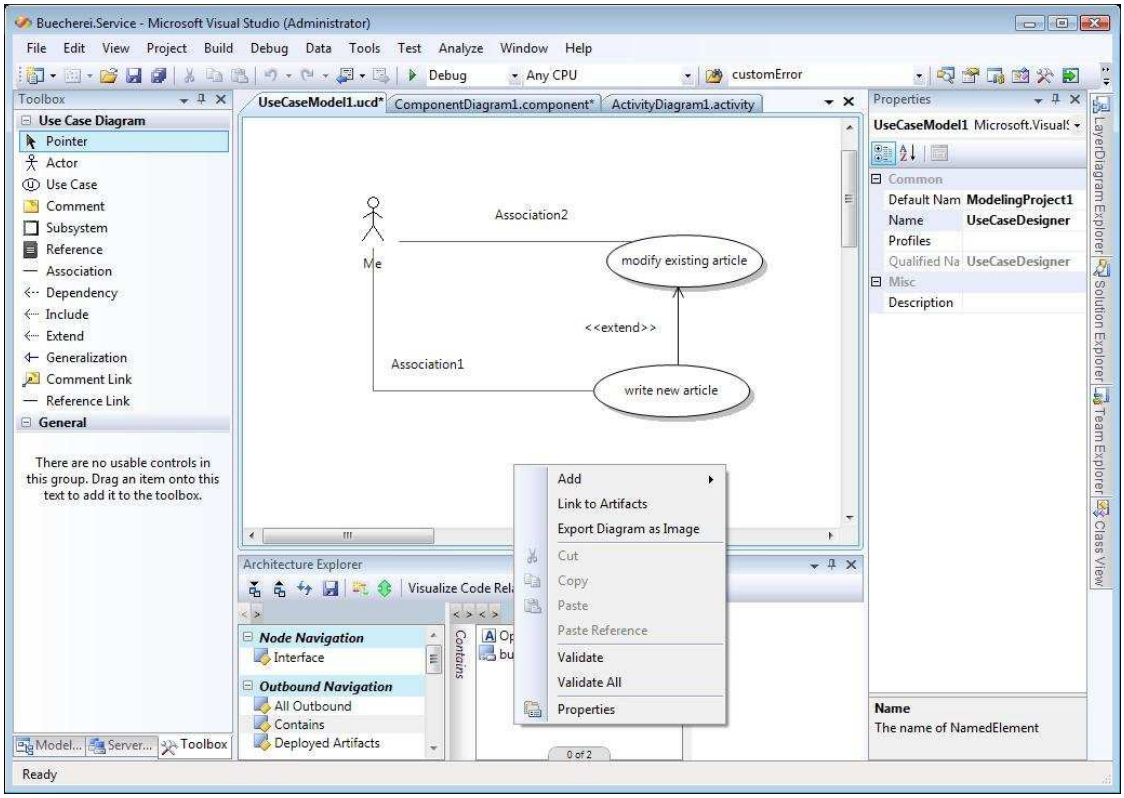
Velkou novinkou ve Visual Studiu 2010 jsou UML diagramy. Nyní je možné generovat jak diagramy z existujícího kódu (reverse engineering), tak kód z diagramů (Code Generation). Nejedná se však o takzvaný Round-trip engineering, který spočívá v tom, že kód a příslušné diagramy jsou konzistentní. Čili změny v jednom se neustále promítají do druhého. Toto nebylo záměrem Microsoft vývojářů a Reverse Engineering i Code Generation jsou spíše jednorázovou záležitostí.

K dispozici je pět typů diagramů. Nejedná se tedy o plnou podporu specifikace UML 2.0, která diagramů definuje 13.

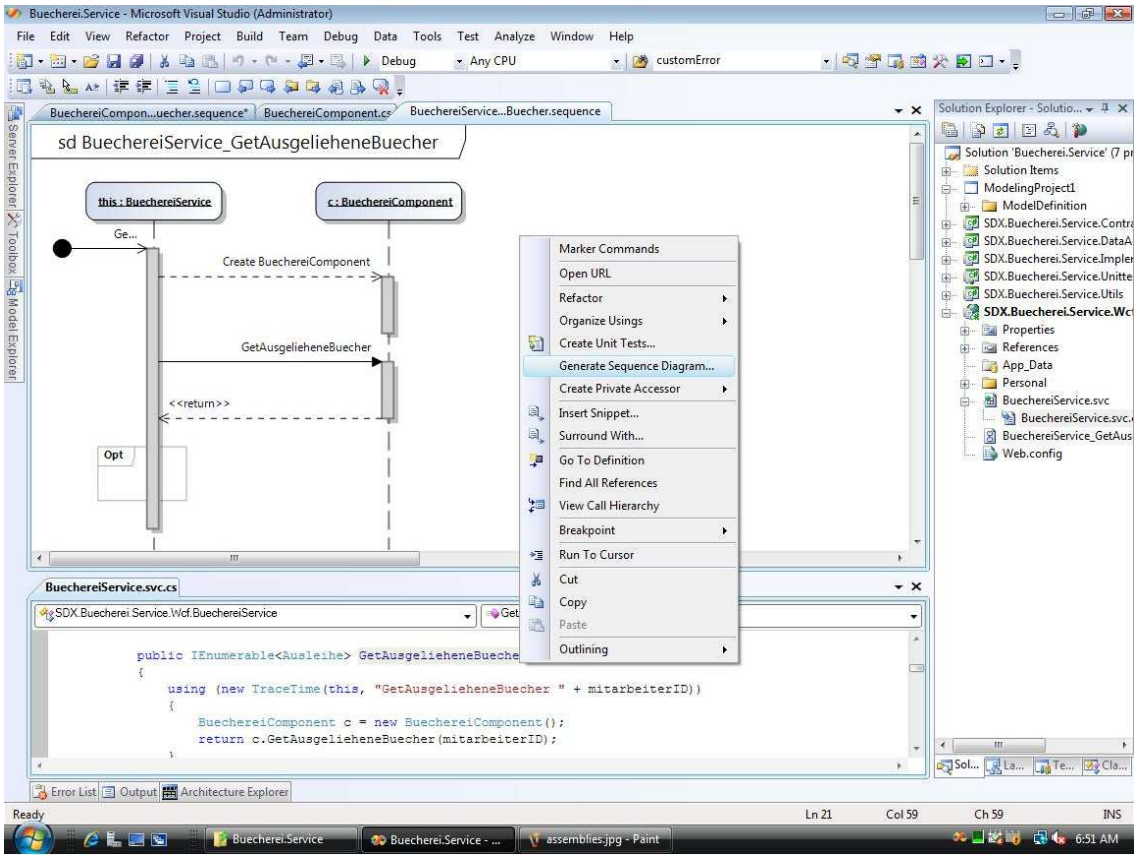
Podporované typy diagramů jsou:

- Activity diagram
- Component diagram
- (Logical) class diagram
- Sequence diagram
- Use case diagram

Zajímavou vlastnost skýtá Sequence diagram, který nám umožní ohlídat, zda veškerá volání probíhají opravdu pouze v rámci příslušné vrstvy (nebo komponenty, chcete-li). Architekt si tak může dobře ohlídat, zda jsou principy jím navržené architektury opravdu dodržovány.



Obr. 1 Ukázka UseCase diagramu



Obr. 2 Ukázka Sekvenčního diagramu

Velmi užitečným se zajisté prokáže také „Architecture Explorer“, který umožňuje vizualizovat vztahy mezi jednotlivými třídami/jmennými prostory/assemblies. Efektivně tak pomáhá proniknout do tajů cizího kódu a rychle se v něm zorientovat.

3.4.3 Novinky v kódu

Poměrně zásadní novinkou je zavedení odlišných konfiguračních souborů (web.config) pro jednotlivé kompilační profily. Web.config pro Release tak bude odlišný od web.configu pro Debug, Staging aj. Bude toho docíleno pomocí transformací. Pro každý kompilační profil si programátor nadefinuje, jak se web.config má změnit od své základní podoby. Bude tak možné jednotlivé elementy konfigurace přidávat, odebírat a měnit. A sice za pomocí nového transformačního jazyka ne nepodobného XSLT.

Poté, co vyvineme novou webovou aplikaci, je potřeba ji umístit na server. K tomu již v minulosti ve Visual Studiu sloužil nástroj Publish, který umožňoval aplikace publikovat na vzdálené servery. Ve své nové verzi se dočkalo různých vylepšení, ale hlavě přibyl nový nástroj Publish to SQL, který umožňuje snadné publikování SQL databází.

Novinek se dočkal také ASP.NET output caching. Zjednodušeně řečeno je output caching metoda, která umožňuje ukládat vygenerované http odpovědi (stránky, které si uživatel vyžádá včetně finálního výstupu z code-behind funkcí) do paměti a poté je odsud znovu použít, pokud je vyžádána stejná stránka. Šetří se tak jak procesorové cykly, tak čas nutný k navrácení odpovědi ze serveru.

Jedním z problémů s touto, jinak velmi populární, optimalizační metodou je nemožnost cache sdílet mezi více servery. A také je potřeba cache znovu vybudovat, pokud dojde k restartu aplikačního poolu (serveru). Nově tedy je možné cache ukládat nejenom do paměti, ale i na disk. Zde je pak možné ji sdílet a také odolá pádu serveru.

3.4.4 Vývoj webu

Mnoho vývojářů ASP.NET si vyzkoušelo již ukázkové vydání ASP.NET MVC. Vývojové prostředí Visual Studia 2010 poskytuje veškerou potřebnou podporu pro tuto knihovnu. Ve Visual Studiu 2008 bylo věnováno hodně úsilí podpoře JavaScriptu ve vývojovém prostředí a jeho debuggeru. V nové verzi je Visual Studio 2010 díky spolupráci se

skupinou JQuery první verzí Visual Studia, která přináší JQuery jako standardní součást sady pro ASP.NET.

3.4.5 Vývoj pro Microsoft Windows 7

Ve Visual Studiu 2010 bylo hodně investováno do C++, aby byl vývoj aplikací pro Windows co nejjednodušší a nejproduktivnější. Byly přidány nástroje, které pomáhají vývojářům vyvíjet nové aplikace pro Windows 7 a existujícím aplikacím umožňují využívat výhod nových prvků Windows. Byla zahrnuta plná podpora knihoven a hlaviček Windows 7, důležitá vylepšení MFC pro podporu nových prvků uživatelského rozhraní Windows 7, jako je lišta, živé ikony, přístup k hledání a dokonce podpora pro dotyková zařízení.

3.4.6 Integrace s Microsoft Office

Visual Studio 2005 přineslo první vydání nástrojů Visual Studio Tools for Office. Od té doby se vývoj pro Microsoft Office stal integrovanou součástí Visual Studia a stejně jako Office se i Visual Studio mění tak, aby přinášelo nejlepší zážitek pro klienta i server.

Ve Visual Studiu 2010 jsou vývojáři schopni vytvářet klientské aplikace pro Office, které mohou spolupracovat s různými verzemi Office, jak 32 tak 64bitovými, a toto umožňuje jediný zaváděcí balíček. S vytvořením zaváděcích balíčků pomáhá přímo IDE, které mohou vývojáři použít pro grafické sestavení balíčku, který si koncový uživatel nainstaluje.

S novým nástrojem SharePoint Explorer jsou vývojáři schopni podívat se na existující SharePoint portál a otevřít různou infrastrukturu a entity, aby našli oblasti, se kterými chtějí pracovat. Vývojáři mohou tyto existující SharePoint stránky snadno přebudovat na svých klientských stanicích podporou generování SharePoint projektů z původních WSP balíčků, které byly použity pro vytvoření stránky.

3.4.7 Modelování, které pracuje s kódem

Ve většině podniků se dnes pouze 20 % kódu píše pro nové aplikace, většina práce spočívá v upravování existujícího kódu.

Při práci na existujícím kódu nemusí architekti a vývojáři nutně mít dostatečně dobré nástroje k tomu, aby porozuměli systému, věděli, co je třeba udělat pro potřebné změny, nebo byli schopni předvídat dopady provedených změn.

Modelovací nástroje jsou integrovány do vlastního kódu aplikace. To znamená, že vývojář nebo architekt může používat modely pro prozkoumávání existujících částí kódu. Nový nástroj Architecture Explorer ve Visual Studio Team Systemu dává vývojářům a architektům možnost vytvářet plně architektonický obraz existujícího kódu. To přispívá k lepším informacím o používání, znovupoužívání nebo odstraňování existujícího kódu. Architecture Explorer poskytuje architektům a vývojářům mechanizmus vizualizace kódu mnoha způsoby jako grafy, diagramy nebo nákresy závislostí.

Představení Architecture Layer Diagramu znamená, že vývojář nebo architekt může použít modely také pro omezení kódu. Architecture Layer Diagram se dá spojit s kódem vytvořením aktivního diagramu, který lze použít pro validaci. Například pokud architekt navrhuje systém, kde by prezentační vrstva neměla komunikovat s datovou vrstvou, můžete chtít toto umožnit při testování. Visual Studio Team System 2010 to dokáže udělat.

Nový nástroj Architecture Explorer umožní jednotlivcům vytvořit vizuální reprezentaci částí kódu. [5]

3.4.8 Ostatní novinky

Velkým zájmem mnoha vývojářů je v posledních letech agilní programování a Test Driven Development. Microsoft se snaží tento trend podpořit a na podporu TDD metodiky přidal do Visual Studia několik drobných vylepšení. Pokud si nyní napíšeme testy bez toho, aby testovaná funkcionalita již v kódu existovala, bude mnohem snazší vygenerovat potřebné třídy a rozhraní podle toho, jak je v našich Unit Testech používáme. Pokud se počet Unit Testů rozroste do rozměrů, kdy spouštění všech zabere dlouhou dobu, je k dispozici nová schopnost Visual Studia 2010 rozpoznat ty testy, jichž se změna kódu přímo dotýká a spustit pouze ty a ušetřit tím drahocenný programátorův čas.

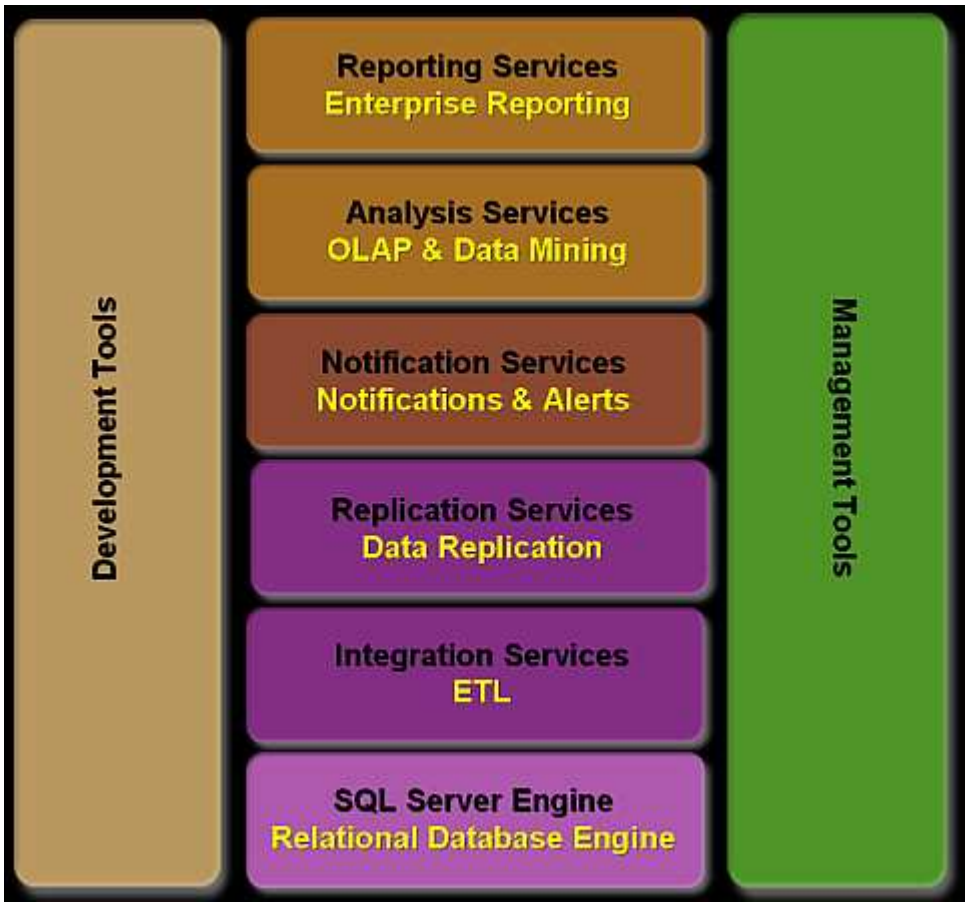
Visual Studio 2010 také obsahuje podporu pro vývoj Cloud aplikací (technologie Windows Azure), či stále více a více důležité paralelní programování. A to jak podpora v kódu – nové SDK, tak v IDE – nové nástroje pro ladění multi-vláknových aplikací. [8]

4 MICROSOFT SQL SERVER

SQL Server je všestranné, integrované a komplexní řešení pro data, které přispívá ke zvýšení výkonnosti uživatelů v celé organizaci poskytováním zabezpečenější, spolehlivější a produktivnější platformy pro podniková data a aplikace Business Inteligence. Prostřednictvím rozsáhlé sady funkcí, vzájemné funkční spolupráce se stávajícími systémy a automatizace rutinních úloh nabízí organizacím všech velikostí úplné datové řešení. Ve svém jádru je SQL Server relační databáze, schopná podporovat jakékoli projekty od malých domácích v řádu několika megabajtů až po podnikové multi-server systémy spravující terabajty dat.

4.1 Architektura

Služby SQL serveru jsou rozděleny do několika vrstev, zobrazených na následujícím obrázku.



Obr. 3 Architektura Microsoft SQL Serveru

- **Služba SQL Server Reporting Services (SSRS) -** představuje ucelenou na server orientovanou platformu pro vytváření, správu a doručování tradičních papírových, stejně jako interaktivních, webových sestav. Jako integrovaná součást produktů společnosti Microsoft pro práci s obchodními informacemi spojuje služba Reporting Services funkce pro správu dat serveru SQL Server a systému Microsoft Windows Server se známými výkonnými aplikacemi sady Microsoft Office a umožňuje práci s informacemi v reálném čase a podporu každodenních operací a rozhodování. Služba SQL Server Reporting Services podporuje celou řadu datových zdrojů, včetně OLE DB a Open Database Connectivity (ODBC), a množství výstupních formátů, například pro nejpoužívanější webové prohlížeče a aplikace sady Microsoft Office, PDF, CSV.
- **SQL Server Analysis Services (SSAS) –** přestože se jedná o službu, která je součástí Microsoft SQL Serveru, jedná se o samostatný OLAP(technologie uložení dat v databázi, která umožňuje srozumitelně uspořádat velké objemy dat) server, který ke komunikaci s klienty využívá rozhraní OLE DB for OLAP.
- **SQL Server Notification Services (SSNS)** - framework umožňující vytvářet aplikace, které generují a odesílají různá upozornění, a také je platformou pro hostování těchto aplikací. Rychle a jednoduše lze vytvořit aplikaci, která generuje hlášení na základě nějaké události, a tuto aplikaci poté zveřejnit na Notification Services server.
- **SQL Server Integration Services (SSIS) –** je novým nástrojem od verze SQL 2005, který umožní data z různorodých rozsáhlých zdrojů transformovat do potřebné podoby, zkontrolovat a nahrát do databáze (datového skladu). S takto různě získanými daty pak lze pracovat nezávisle na tom, odkud pochází, spojovat je a transformovat dle potřeby a ukládat do různých cílů. SSIS eliminují (nebo alespoň minimalizují) nutnost meziukládání dat (stagingu).

4.2 Databázový engine

Databázový engine je základní komponentou SQL Serveru, který je odpovědný za ukládání, správu a zabezpečení dat v SQL serveru. Data mohou být v Microsoft SQL Serveru ukládána ve formě relačních tabulek a také ve formě XML dokumentů. Ve verzi

SQL Server 2005 byla podpora pro práci s XML dokumenty výrazně rozšířena. Nyní poskytuje velké možnosti při práci s XML daty.

Mezi základní úkoly databázového enginu patří:

- **Spolehlivé úložiště pro data** – základem spolehlivosti je samozřejmě kvalitní hardware a data by měla být ukládána nejlépe na pole RAID. SQL Serveru jako takovému ale na použitém hardware nezáleží, server sám si organizuje data tak, aby byla uchována v konzistentní podobě, a sám spravuje své datové struktury. Jednotlivé řádky tabulky jsou ukládány v datové stránce, která má velikost 8KB, z toho nám také vyplývá jedno z design omezení při návrhu tabulek. Řádek nemůže být větší než datová stránka, tudíž řádek tabulky nemůže být delší než 8060 B.

- **Transakční log** – mechanismus, který nám v SQL serveru slouží k uchování informací o probíhajících transakcích. Transakční log využívá metodu write-ahead, to znamená, že při práci se serverem je nejprve zaznamenán záznam do logu, poté jsou teprve data měněna v cache. Systém sám v určitých intervalech ukládá změny přímo na disk u těch transakcí, které byly dokončeny. Transakční log velmi oceníme v případě havárie serveru, protože jej používáme při jeho obnově. Nelze se však spoléhat jen na transakční log, protože ten nemusí obsahovat veškeré záznamy o aktivitách na našem serveru. Záleží na nastavení recovery modelu u naší databáze a také na operácích, které probíhají.

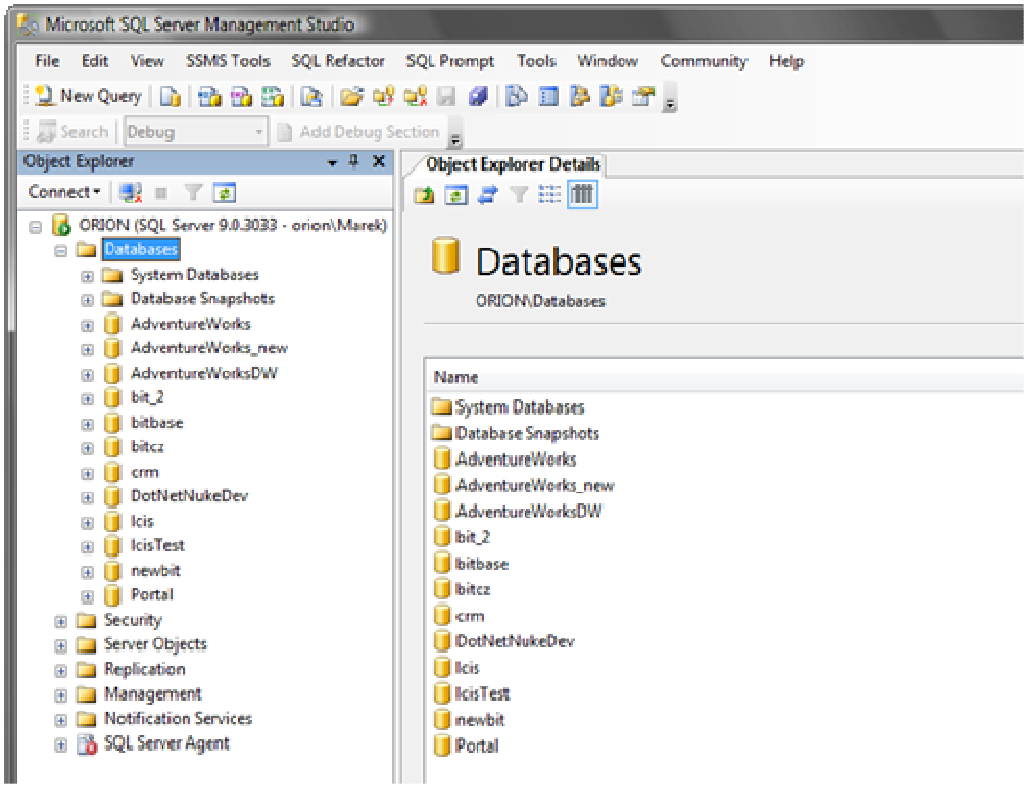
- **Kontrola přístupu** – SQL Server kontroluje přístup k datům na několika úrovních, a to na úrovni serveru, databáze a objektu. Novinkou do verze 2005 jsou schémata, která mohou být také použita ke kontrole přístupu. Uživatelské účty, které slouží pro přihlášení k serveru mohou být definovány v Active Directory (Integrated Authentication) nebo přímo na SQL serveru. Ve většině případů si vystačíme s Windows autentizací, účty na SQL serveru používáme hlavně v případě přístupu z Linux/Unix stanic nebo z přístupů mimo Active Directory doménu.

- **Datová integrita** – stav, při němž záznamy v celé databázi vyhovují soustavě určitých definovaných pravidel. Rozeznáváme tři úrovně datové integrity a to doménovou (hodnoty, které můžeme uložit do jednotlivých sloupečků), entitní (neexistují duplikátní záznamy) a referenční (odkazy mezi jednotlivými tabulkami,

případně mezi sloupci z jedné tabulky pouze na existující hodnoty). Pro zajištění integrity má SQL Server několik metod – constraints, triggery a schémata.

4.3 SQL Server Management Studio jako nástroj pro správu

SQL Server obsahuje několik nástrojů, které nám slouží pro správu, konfiguraci, ladění výkonu, monitoring. Centrálním nástrojem (opravdu základní konzolí) je SQL Server Management Studio (SSMS). Což je produkt, který v sobě spojuje Enterprise Manager a Query Analyzer známé z verze 2000. SSMS je nástroj, který nám slouží pro správu všech serverů v celém prostředí z jedné aplikace.



Obr. 4 Hlavní okno Management Studia programu Microsoft SQL Server

Mezi základní úlohy, ke kterým využíváme SSMS patří:

- Vytváření databází
- Správa databázových objektů – tabulky, schémata, procedury, funkce, triggery, indexy
- Správa uživatelských účtů, loginů a rolí
- Tvorba a správa automatizovaných úloh

- Zálohování a obnova
- Nastavení parametrů serveru – jazyk, využití CPU, paměti, výchozí umístění souborů
- Psaní a ladění SQL skriptů, včetně jejich organizace do projektů [9]

4.4 Microsoft SQL Server 2008

Microsoft uvedl nový databázový server přesně podle cyklu deklarovaného při uvedení předchozí verze (2005), tedy po necelých třech letech. Tento přesný odhad, který nebyl právě u tohoto produktu zvykem, zmátl několik jedinců natolik, že se začaly objevovat otázky, zda vůbec nová verze přináší tolik významných vylepšení proti předchozí.

4.4.1 Novinky v oblasti škálovatelnosti a optimalizace výkonu

SQL Server 2008 nabízí celou řadu vylepšení škálovatelnosti, včetně plné podpory 64bitových systémů s až 8 terabajty paměti, vysoce výkonných počítačů a také podpory přidávání paměti a procesorů do kompatibilních počítačů za chodu bez nutnosti vypnutí systému.

Další významnou novinkou je uvedení technologie Resource Governor (Správce prostředků), který poskytuje koncovým uživatelům konzistentní a předvídatelné doby odezvy. Organizace mohou přidělovat prostředky a definovat priority různých úloh tak, aby současně běžící úlohy neovlivňovaly výkon koncových uživatelů. Resource Governor v systému SQL Server nabízí oproti podobným řešením konkurence několik důležitých výhod. Stanovením minimálního využití CPU a paměti je možné určovat priority úloh s cílem zaručit, aby byla pro specifické úlohy v databázi zajištěna požadovaná úroveň služeb (SLA, service level agreement). Resource Governor také umožňuje omezit kapacitu paměti pro jednotlivé fondy prostředků a zabránit tak vzniku zacyklených dotazů.

Další novinkou v SQL Server 2008 je vícevláknový přístup k oddílům. Ten umožňuje systému SQL Server 2008 zvýšit výkon zpracování dotazů v dělených tabulkách pro paralelní plány. Vícevláknový přístup k oddílům dále také mění způsob reprezentace paralelních a sériových plánů a rozšiřuje informace o oddílech, které jsou poskytovány v plánech spouštění při kompilaci (compile-time) i za běhu (run-time).

4.4.2 Novinky v oblasti bezpečnosti

Nejvýznamnějším zabezpečením SQL Serveru je jeho samotná koncepce a odolnost produktu proti útokům. Od uvedení verze 2005 lze SQL Server nazývat „secure by default“, byl totiž vydán pouze jediný kritický Security bulletin týkající se tohoto produktu. Pokud tuto charakteristiku porovnáte s nejbližší konkurencí Oracle 10g, ale dokonce i 11g, dostáváte se k poměru 1:stovkám. Toto srovnání je průběžně sledováno a aktualizováno na nezávislých webových stránkách Secunia.com.

SQL Server 2008 ale nabízí i další vylepšené zabezpečení, umožňující efektivní správu konfigurace zabezpečení, silné ověřování a řízení přístupu, výkonné možnosti šifrování a komplexní auditování. Jednou z největších novinek je rozšíření možností šifrování dat. Šifrování dat bylo obsaženo již ve verzi 2005, ale v nové verzi je možné šifrovat data transparentně na úrovni databází prostřednictvím zabezpečeného šifrovacího klíče DEK, tedy bez jakékoliv nutnosti úpravy aplikace. Rovněž je možno využít rozšířené správy šifrovacích klíčů a ukládat je v samostatném hardwarovém zařízení mimo data.

Mezi další vylepšení v oblasti bezpečnosti, která stojí za zmínku, patří:

- Vynucení zásad zabezpečení pro datové služby v rámci podniku pomocí systému DMF (Declarative Management Framework)
- Řízení přístupů k datovým prostředkům pomocí vynucení zásad hesel, ochranou metadat pomocí katalogu
- či použitím rolí a proxy účtů
- Auditování všech akcí pomocí nového objektu Audit

4.4.3 Novinky v oblasti správy

SQL Server 2008 obsahuje sadu pokročilých nástrojů pro správu, jako jsou Management Studio, Performance Studio, správa založená na zásadách a PowerShell, tedy nové prostředí pro procházení a správu databází, tabulek a dalších databázových objektů systému SQL Server. S výjimkou prostředí SQL Server PowerShell jsou tyto nástroje přístupné prostřednictvím standardizovaných známých rozhraní.

Vůbec nejzásadnější je právě správa založená na zásadách, která přináší zcela novou koncepci pro správu jednoho a více databázových serverů. Správa založená na zásadách je

nový systém pro správu jedné či více instancí produktu SQL Server 2008 pomocí nástroje SQL Server Management Studio. Pomocí něj je možné vytvářet zásady pro správu entit, jako jsou instance systému SQL Server, databáze a další objekty systému SQL Server na databázovém serveru. K dispozici je podpora tvorby a nasazení rozsáhlých politik, využít lze i distribuované plánování úloh, kontrolovat jejich provedení a vynucovat aplikaci těchto politik na servery, které nesplňují daná kritéria. Správci databází tak získávají úplnou kontrolu nad svými databázovými servery.

4.4.4 Standard Edition vs Enterprise

Microsoft nabízi SQL Server v několika edicích. Jedná se o Express, Workgroup, Compact, Web, Developer, Standard a Enterprise. Často bývá diskutován rozdíl mezi jednotlivými edicemi, především pak mezi Standard Edition a Enterprise Edition. Rozdíl mezi těmito edicemi, které jsou obě určeny pro větší komerční řešení, je především ve škálovatelnosti, možnostech pro datawarehousing, podmínkách pro rozsáhlou datovou konsolidaci, apod. Zjednodušeně řečeno tvrzení, že rozdíl mezi Standard a Enterprise je pouze v clusteringu, je pouze přežitek z doby, kdy byl dostupný SQL Server 2000. Dnešní rozdíly tkví především ve vlastnostech, které mohou být velice důležité v dlouhodobém rozvoji aplikační infrastruktury. Při rozhodování o výběru vhodné edice je třeba si uvědomit, že Standard edice je určena do nekritických prostředí a naopak není vhodná pro použití v prostředí, kde se zpracovávají nebo plánují zpracovávat velké objemy dat. SQL Server Standard Edition neumí plně a efektivně využít všech hardwarových zdrojů (např. více procesorů v serveru, jak fyzických, tak logických) a nemá nástroje pro efektivní řešení vysoké dostupnosti. Jistá omezení jsou například i v oblasti Business Intelligence a bezpečnosti. Vedle toho u Enterprise edice nenarazíte na žádná omezení využití prostředků, takový databázový server je možné plně škálovat.

4.4.5 Stabilita

Jedna z největších referencí na nasazení SQL Serveru 2008, kde je zároveň velmi vysoký požadavek na stabilitu, bezvýpadkový provoz a rychlá latence a kde opravdu každá vteřina výpadku představuje citelnou ztrátu, není překvapivě žádná bankovní a investiční instituce, nýbrž společnost provozující největší online sázkový portál v Evropě – Bwin. Odhaduje se, že jedna minutu výpadku jejich Core systému přijde společnost na 6500 Eur. Technické

parametry řešení jsou naznačeny níže v tabulce.

Počet transakcí za vteřinu ve špičce: 6 000

Počet databázových operací za vteřinu ve špičce: 30 000

Počet databází: 850

Největší databáze: 4 TB

Největší tabulka: 2 miliardy řádků

Celková velikost databází: 100 TB [10]

5 LINQ

Microsoft .NET Framework verze 3.0 obsahuje techniku, která vývojářům dovoluje provádět operace nad daty v paměti mnohem jednodušším a efektivnějším způsobem a to pomocí dotazů podobných jazyku SQL. Jedná se o technologii LINQ (Language INtegrated Query). LINQ dovoluje dotazování, přidávání, filtrování, mazání a editaci dat v polích, kolekcích odvozených od generického rozhraní IEnumerable<T>, XML či SQL databázích.

S oficiálním příchodem .NET frameworku 3.5 vyšla na svět finální verze projektu, který se dá označit za revoluci v přístupu k datům na platformě .NET.

5.1 Co je LINQ?

Od svého počátku bylo možné v aplikacích postavených nad platformou .NET pracovat z relačními či hierarchickými daty a to hned několika způsoby. V případě dat, která jsou uložena v relační databázi, je možné přistupovat skrze známé rozhraní ADO .NET a to jak v připojeném, tak odpojeném režimu. Pokud jsou data reprezentována hierarchicky v populárním formátu XML, lze využívat typů z assembly System.Xml a pracovat s těmito daty buď sekvenčně nebo pomocí DOM (Document Object Model – API umožňující přístup či modifikaci obsahu, struktury, nebo stylu dokumentu, či jeho částí pomocí objektově orientovaná reprezentace XML nebo HTML dokumentu). Nyní ovšem přichází technologie, která umožňuje zcela nový způsob práce s daty na platformě .NET a nese název LINQ, což znamená Language INtegrated Query.

LINQ přináší podporu dotazování přímo do .NET programovacího jazyku a to s sebou nese hned několik výhod oproti výše zmíněným postupům. První a asi nejvíce viditelnou novinkou, které si ihned všimne kdokoli, kdo začne pracovat s daty v jazyku C# 3.0 (a vyšším) nebo Visual Basic 9, je nová paleta klíčových slov pro provádění LINQ dotazů nad daty. Další velmi příjemnou novinkou, která plyne z integrace dotazování přímo do .NET programovacího jazyku je odhalení chyb již v době kompilace, takže vaše dotazy budou kontrolovány hned a většina chyb nemá šanci projevit se až za běhu aplikace.

Pomocí tohoto nového přístupu je možné pracovat téměř s jakýmikoliv daty, protože architektura technologie LINQ je navržena tak, že je možné tvořit její implementace pro jednotlivé datové zdroje. Je to podobné jako v ADO .NET, kde je možné implementací rozhraní, vytvořit .NET provider pro specifický typ databáze, avšak v LINQ tato

rozšiřitelnost nezůstává jenom u relačních databází a je mnohem abstraktnější. Této volnosti je dosaženo hlavně díky novince v .NET 3.5, kterou jsou Expression Trees, jež umožňují pracovat s kódem jako s daty a tím pádem může být LINQ dotaz konkrétním providerem přeložen pro adekvátní datový zdroj a je jedno jestli se jedná o data relační, hierarchická či nějaká jiná. Tento koncept má mimo jiné za následek, že díky LINQ se sjednocuje způsob práce s daty různých druhů.

Spolu s .NET Frameworkem 3.5 je dodáno několik oficiálních implementací LINQ od Microsoftu, které by měly pokrýt běžné potřeby pro tvorbu aplikací pracujících s daty.

Jedná se o tyto implementace :

- LINQ to Objects – Implementace LINQ pro standardní kolekce nacházející se v paměti
- LINQ to SQL – Implementace LINQ pro Microsoft SQL Server 2000 a vyšší
- LINQ to XML – Implementace LINQ pro práci s XML daty
- LINQ to DataSet – Implementace LINQ pro práci s ADO .NET datasety

I přestože primárním přínosem LINQ je maximální efektivita při dotazování nad daty, u většiny implementací LINQ to u dotazování nekončí a kromě toho je možné data nejen efektivně dotazovat, ale také modifikovat. Takže například v případě LINQ to SQL máme k dispozici nástroj, který nejen, že umožní provádět dotazy nad databází SQL server jednoduše a typově ze C# 3.0, VB 9 nebo jiného jazyku, ale díky možnosti data i modifikovat se z této implementace LINQ stává skvěle použitelný objektově-relační mapper. [11]

5.2 Klíčová slova

Technologie LINQ umožňuje manipulaci s kolekcemi a poli pomocí nových syntaktických výrazů velmi jednoduchým způsobem podobným jazyku SQL. Díky tomu odpadává nutnost vytváření vlastních algoritmů pro vyhledávání a úpravy v kolekcích.

Přehled klíčových slov je uveden níže:

- Select – výběr hodnoty kterou chceme použít
- SelectMany – výběr více hodnot najednou (např. pole)

- Join – spojení více poskytovatelů dat
- GroupBy – rozdělení dat do více skupin podle určitého klíče
- Where – omezení výběru prvků podle specifikované podmínky
- OrderBy, OrderByDescending – specifikace třídění, umožňuje výběr elementu podle kterého se má třídit
- First, Last – výběr prvního nebo posledního prvku z kolekce
- ElementAt – výběr prvku podle udaného indexu
- Count – počet prvků v kolekci
- Union, Intersect, Except – definice množinových operací sjednocení, rozdíl a průnik
- Sum, Min, Max, Average – vrací součet, minimální, maximální či průměrnou hodnotu z dané kolekce
- Reverse – otočí pořadí prvků v kolekci
- Concat – spojí dvě kolekce dohromady
- OfType – výběr pouze těch prvků, které jsou specifikovaného typu

Obecná forma jednoduchého LINQ dotazu v C# tedy vypadá podobně jako v tomto příkladu:

from [typ] proměnná *in* datový_zdroj

[*where*] podmínka_restrikce

[*orderby*] klíč_řazení [*descending*]

select výraz_projekce

Jak je vidět, datový typ prvku nemusí být ve *from* klauzuli dotazu explicitně určen a je, stejně jako v případě klíčového slova *var*, odvozen v době kompilace. Ovšem ne vždy může být datový typ prvku v sekvenci takto odvozen. V případě generické kolekce to možné je, ovšem v případě negenerické kolekce jako je například ArrayList, musí být kolekce buď pomocí metody Cast převedena na kolekci generickou, nebo v případě C#

query keywords stačí v klauzuli „*from* typ prvku“ explicitně uvést. Ovšem v případě nesprávně definice typu prvku hrozí výjimka InvalidCastException. [12]

II. PRAKTICKÁ ČÁST

6 NÁVRH A ANALÝZA POTŘEBNÝCH VLASTNOSTÍ SYSTÉMU

Prvním krokem k tvorbě každé aplikace by měla být důkladná analýza požadavků a požadovaných funkcí, které má systém obsahovat. Proto i zde byla prvním bodem mé práce analýza těchto požadavků.

6.1 Požadavky na monitorovací systém

Po důkladném rozboru funkcí byly stanoveny tyto základní požadavky:

- Kontrola služeb bude prováděna pomocí aplikace, která bude spuštěna jako systémová služba.
- Zobrazení kontrolovaných služeb a umožnění jejich nastavení pomocí webové stránky.
- Možnost kontroly více typů webových služeb.
- Zajištění aktualizace zobrazovaných dat v malých časových intervalech, nikoliv však neustále.
- Do administrační části (webová aplikace) bude umožněn přístup pouze po zadání hesla. Do statistik a nastavení dohledového systému tak bude zamezen přístup nepovolaným osobám.

6.2 Systémová služba pro kontrolu služeb

Část systému bude naprogramována jako systémová služba. Dle nastavených parametrů bude kontrolovat jednotlivé služby a zaznamenávat výsledky do databáze.

Systémová služba by měla pracovat dle následujících požadavků:

- Kontrola bude probíhat pouze pro služby, které budou nastavené jako aktivní.
- Kontrola služeb pomocí Threadů (vlákna aplikace) z důvodu možných výpadků a tedy nepřijetí odpovědi služby v reálném čase.
- Naměřená data se budou ukládat do databáze pro pozdější vyhodnocení a vizualizaci.

- Ukládat se bude také poslední stav aplikace pro lepší detekci opakovaných chyb služby.
- Aplikace bude ohlašovat chyby dle nastaveného počtu opakování z důvodu zbytečného šíření poplašné zprávy.

6.3 Webové rozhraní dohledového sytému

Je zapotřebí vytvořit webovou aplikaci, která bude umožňovat jak přehled o sledovaných službách, tak jejich editaci a přidávání dalších služeb. Služby musí být zobrazeny v přehledné struktuře, nejlépe seřazené dle patřičných parametrů, například dle serverů, ke kterým služby patří. Musí být umožněna také změna serveru (skupiny) u dané služby.

Dále by měla aplikace umožnit zobrazení statistik dostupnosti služeb a výpis logů z dosud provedených kontrol.

Webové rozhraní musí umožňovat následující funkce:

- Vytvoření skupiny například dle serveru, do kterého služba spadá.
- Přidání či odebrání služby a nastavení parametrů pro kontrolu této služby.
- Vypínání a zapínání kontroly jak jednotlivých služeb, tak celé skupiny služeb.
- Vizualizace statistik a výpis logů dle naměřených dat.
- Možnost vytvoření kontaktů pro zasílání upozornění o chybách a přiřazení ke skupinám služeb.
- Kontakty musí umožňovat více typů upozornění o chybě, například pomocí emailu nebo zavoláním URL.
- Umožnění základního nastavení pro odesílání upozorňovacích emailů.
- Změna hesla pro přístup do aplikace včetně zaslání zapomenutého hesla na email.

6.4 Komunikace mezi součástmi

Komunikace mezi systémovou službou pro monitoring a webovou stránkou bude probíhat prostřednictvím databáze.

Webová stránka umožní editaci kontrolovaných služeb a tato data bude ukládat do databáze. Systémová služba monitoringu služeb bude v časových intervalech tato data číst a kontrolovat jednotlivé služby.

Výsledky z kontrol služeb se budou zapisovat do databáze a tato poté v pravidelných intervalech načte a zobrazí webová stránka.

7 ARCHITEKTURA DOHLEDOVÉHO SYSTÉMU

Po důkladné analýze jsem navrhla strukturu systému a jeho součásti. Dle požadavků byla zvolena struktura dvou aplikací. Výsledná architektura systému je uvedena níže.

7.1 Sběr dat

Sběr dat probíhá pomocí části aplikace, která bude spuštěna jako systémová služba. Podle nastavených parametrů probíhá v určitých časových intervalech kontrola služeb. Z důvodu možnosti, že služba nebude dostupná a ostatní kontroly by se tímto zpozdily, je sytém řešen pomocí Threadů (vláken procesu), které zajistí běh více kontrol najednou. Aplikace vždy podle aktuálního času vybere služby, které se mají právě kontrolovat a tyto kontroly pak spustí. Poté, co systém dostane od všech kontrolovaných služeb odpovědi nebo vyprší čas pro jejich odpověď, zapíše aplikace najednou všechny zjištěné stavy kontrolovaných služeb do databáze. Po uložení jsou dále porovnávány odpovědi s očekávanými dle nastavení. Pokud odpovědi nesouhlasí, vyhodnotí aplikace stav jako chybu a dle nastavení povoleného počtu chyb poté informuje uživatele aplikace o chybě. Počet povolených chyb zajistí, že aplikace nebude ohlašovat falešný poplach zbytečně již po jedné detekci chyby z důvodu například aktuálního vysokého vytížení sítě.

7.2 Zobrazení dat

Zobrazení dat zajišťuje část aplikace, která je naprogramována jako webová stránka. Tato aplikace zobrazuje stavy všech služeb, které si uživatel může seskupit dle serverů, na kterých tyto služby běží. Aplikace pak přehledně zobrazuje všechny servery a k nim přiřazené služby a jejich aktuální data. Mezi tato data patří zobrazení poslední stav služby, případně zobrazení chyby, dále datum poslední kontroly a průměrná dostupnost služby.

Dále lze pro každou službu zobrazit statistiky dostupnosti dle naměřených dat a je také umožněn výpis logů z databáze.

7.3 Nastavení parametrů

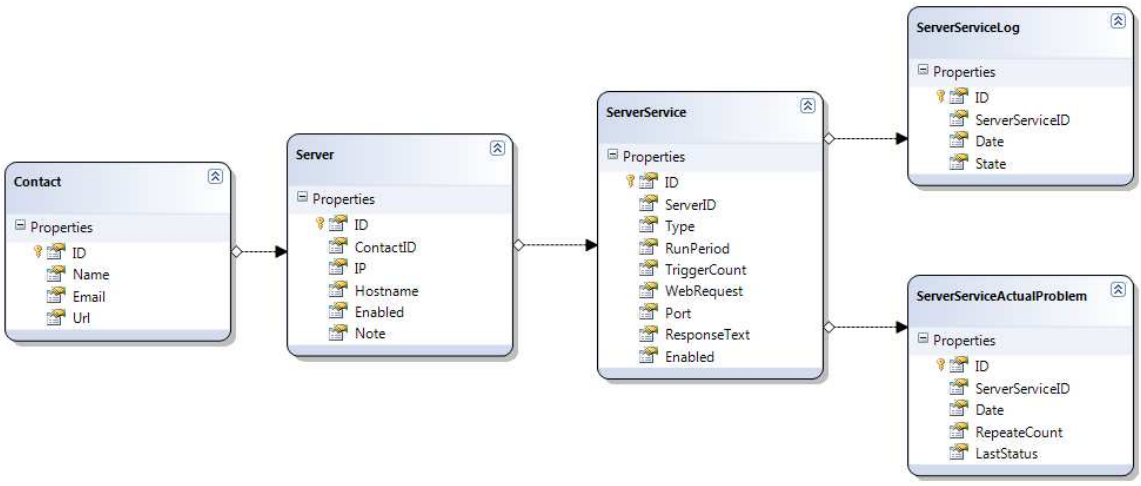
Nastavení aplikace je zajištěno přes stejnou webovou stránku jako zobrazení dat. Lze zde nadefinovat servery a jejich služby a editovat jejich parametry. Patří sem nastavení jako typ

služby, port, očekávaný text odpovědi služby, případně posílaný požadavek. U každé služby lze také kontrolu dostupnosti vypnout a znovu zapnout.

8 DATABÁZE SYSTÉMU

Při návrhu databáze byl kladen důraz na co nejmenší redundanci a maximální rozšiřitelnost položek jednotlivých tabulek. Navržená databáze se skládá z pěti hlavních tabulek. Schéma výsledné databáze je vyobrazeno na obrázku níže včetně relačních vztahů mezi tabulkami.

Kromě databázových tabulek, ve kterých jsou uloženy hodnoty se v databázi nacházejí také ASP.NET Membership tabulky (aspnet_), které umožňují jednoduchou správu uživatelských účtů od vytvoření až po odhlášení a smazání. To vše integrovaným mechanismem ASP.NET.



Obr. 5 Schéma navržené databáze

8.1 Tabulka Contact

Tabulka *Contact* slouží pro uchování informací pro možnost posílání chybových hlášení různých typů. Každému kontrolovanému serveru lze přiřadit jednotlivý kontakt, na který se budou chybová hlášení posílat. Pro možnost posílání hlášení o chybách pomocí emailu slouží položka *Email*. V neposlední řadě je zde možnost zvolit upozornění za pomocí uživatelsky nadefinované stránky, která samotné upozornění zajistí například pomocí dalších komunikačních protokolů. Pro tato upozornění slouží položka *Url*.

Contact

	Column Name	Data Type	Allow Nulls
🔑	ID	int	☐
	Name	varchar(MAX)	☐
	Email	varchar(MAX)	☑
	Url	varchar(MAX)	☑
			☐

Obr. 6 Tabulka Contact

8.2 Tabulka Server

Pro seskupení služeb do jednotlivých skupin dle serverů, na kterém serveru je služba spuštěna, byla vytvořena tabulka *Server*. Pomocí položky *ContactID* lze ke každému serveru přiřadit Kontakt pro posílání upozornění o chybách. Dále byla vytvořena položka pro pojmenování serveru *Hostname* a položka pro zadání adresy či IP serveru *IP*. Každou skupinu služeb přiřazenou k serveru lze najednou vypnout pomocí položky *Enabled*. Toto umožní, že celou tuto skupinu služeb nebude systém dočasně kontrolovat. Pro lepší identifikaci skupiny služeb byla také vytvořena položka *Note*, kam lze vkládat souhrnné informace.

Server

	Column Name	Data Type	Allow Nulls
🔑	ID	int	☐
	ContactID	int	☐
	IP	varchar(15)	☑
	Hostname	varchar(MAX)	☐
	Enabled	bit	☐
	Note	varchar(MAX)	☑
			☐

Obr. 7 Tabulka Server

8.3 Tabulka ServerServices

V této tabulce se nacházejí informace již pro samotné služby, které chceme kontrolovat. Jedná se o položku *ServerID*, která odkazuje do tabulky *Server* a udává, ke kterému serveru či skupině položka patří. Položka *Type* byla vytvořena z důvodu volby typu služby. Dle této položky také aplikace dále určuje, zda se pro kontrolu dostupnosti jednotlivé služby

použijí také další položky této tabulky *WebRequest* a *Port*. Pro určení intervalu pro kontrolu dostupnosti každé služby byla vytvořena položka *RunPeriod.* Z důvodu minimalizace posílaných chybových hlášení byla vytvořena také položka TriggerCount. Aplikace poté informuje o chybách pouze, pokud se opakují v zadaném počtu. Dále byla vytvořena položka *WebRequest*, která je použita pouze v případě, kdy je typ služby *Type* nastaven na „*Web*“, kde je nutno posílat službě požadavek, na který následně volaná služba odpoví. Použití další položky *Port* je také vázáno na již popsanou položku *Type* a je uplatněna pouze tehdy, pokud chceme kontrolovat služby typu „*TCP*“. Předposlední položka *ResponseText* slouží pro identifikaci správnosti odpovědi každé služby. Tuto hodnotu aplikace vždy porovnává s aktuálně přijatou odpovědí, a pokud tyto odpovědi nesouhlasí, vyhodnotí kontrolu služby jako chybovou. Kontrolu dostupnosti služby lze dočasně vypnout či zapnout pomocí položky *Enabled*.

ServerService

Column Name	Data Type	Allow Nulls
ID	int	☐
ServerID	int	☐
Type	varchar(10)	☐
RunPeriod	int	☐
TriggerCount	int	☐
WebRequest	varchar(MAX)	☑
Port	int	☑
ResponseText	varchar(MAX)	☑
Enabled	bit	☐
		☐

Obr. 8 Tabulka Server

8.4 Tabulka ServerServiceLog

Zde se ukládají výsledky ze všech prováděných kontrol. Každá položka tohoto seznamu logů uchovává údaj o tom, ke které službě daný log patří pomocí položky *ServerServiceID*. Dále je zde záznam o datu provedené kontroly *Date*. Stav služby u každé kontroly dostupnosti jednotlivé služby uchovává položka *State*.

ServerServiceLog

Column Name	Data Type	Allow Nulls
ID	int	
ServerServiceID	int	
Date	datetime	
State	varchar(MAX)	

Obr. 9 Tabulka Server

8.5 Tabulka ServerServiceActualProblem

Tato tabulka slouží pro lepší identifikaci chyb a detekci opakovaných chyb. Zde je pro každou službu uchován jeden záznam. Mezi položky každého takového záznamu patří položka *ServerServiceID* odkazující do tabulky *ServerService*. Udává, ke které službě daný záznam přísluší. Položka *Date* ukládá datum poslední kontroly služby a údaj *LastStatus* poslední naměřený stav služby. Údaj o počtu opakování, sloužící pro chybové hlášení, uchovává položka *RepeateCount.*

ServerServiceActualProblem

Column Name	Data Type	Allow Nulls
ID	int	
ServerServiceID	int	
Date	datetime	
RepeateCount	int	
LastStatus	varchar(MAX)	

Obr. 10 Tabulka Server

9 PODROBNÝ POPIS SYTÉMU

Jako vývojové prostředí bylo použito Microsoft Visual Studio 2010, které je nyní nejnovější verzí tohoto nástroje. Tento vývojový nástroj sem si během mého studia nejvíce oblíbila a použila jsem jej už při vývoji své Bakalářské práce. Jako programovací jazyk byl použit jazyk C#.

Výsledný dohledový systém je rozdělen na dvě samostatné aplikace:

- monitorovací aplikace,
- administrační aplikace.

9.1 Monitorovací aplikace

Pro monitorování služeb slouží aplikace, která je naprogramována jako systémová služba.

Jelikož v takové aplikaci není jednoduché kontrolovat běh aplikace a ošetřovat výjimky, byla velká část aplikace naprogramována nejprve jako konzolová aplikace. Teprve po vyzkoušení naprogramovaných funkcí byla aplikace převedena do systémové služby.

Výsledná aplikace se složena z několika metod, které zajišťují načtení dat z databáze, kontrolu služeb a zápis výsledků kontrol do databáze pomocí logu.

Monitorovací aplikace provádí monitorování několika typů služeb. Další služby mohou být zcela jednoduše přidání implementováním rozhraní *IRequest*. Jedná se o tyto služby:

- WEB – webové služby kontrolované metodou GET. V případě potřeby monitorování i pomocí metody HEAD nebo POST může být část kódu jednoduše rozšířena
- TCP – kontrola generické služby komunikující na daném portu pomocí TCP protokolu. Takto můžeme monitorovat typy služeb jako SMTP, POP3, IMAP i další generické služby
- PING – kontrola dané IP adresy pomocí ICMP protokolu ping.

9.1.1 Výběr služeb pro monitoring a jejich kontrola

Prvním krokem aplikace je výběr služeb, které se mají v danou chvíli běhu aplikace kontrolovat.

Aplikace se nejprve připojí do databáze a načte všechny služby, u kterých je zapnut monitoring. Dle nastavených parametrů zadaných pomocí webové aplikace se poté spustí najednou všechny kontroly služeb pomocí *Threadů.* Toto zajistí kontrolu všech služeb najednou bez zbytečného čekání pouze na odpověď například právě nedostupné služby z důvodu chyby.

Poté aplikace čeká na obdržení odpovědí od všech kontrolovaných služeb.

9.1.2 Vyhodnocení kontroly služeb a uložení do databáze

Po přijetí odpovědí od všech právě monitorovaných služeb musí aplikace tyto odpovědi vyhodnotit. Kontrola je provedena na základě porovnání definovaných očekávaných odpovědí v databázi s právě přijatými daty.

Pokud se daná odpověď vyhodnocena jako chyba, je tento stav také zaznamenán do databáze pomocí tabulky *ServerServiceActualProblem*. Pokud se tato chyba opakuje vícekrát po sobě, je tato skutečnost indikována iterací položky *RepeateCount* této tabulky.

Na závěr se všechny výsledky kontroly uloží do databáze *ServerServiceLog*.

9.1.3 Upozornění o chybě

Při detekci chyby aplikace posílá dle přiřazených kontaktů ke skupinám služeb chybová hlášení. Dle vyplněných údajů kontaktu lze posílat upozornění pomocí emailu či pomocí URL adresy, která již upozornění sama vykoná. Ukázka poslání emailu je uvedena níže.

Konfigurace nastavení pro posílání zpráv včetně zvolení SMTP serveru se provádí v konfiguračním souboru aplikace *app.config.*

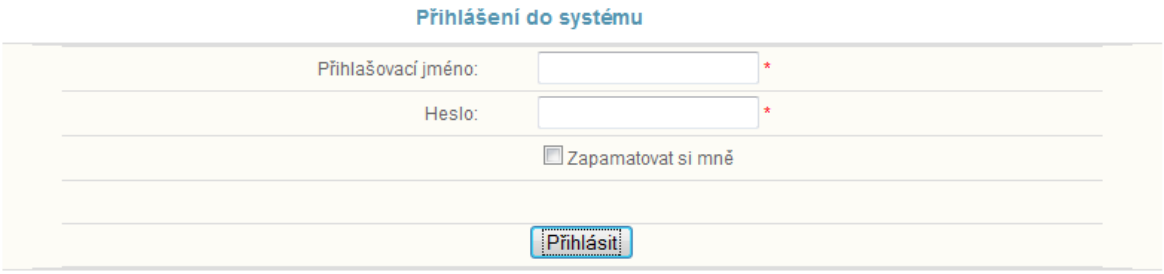
```
private void SendMail(string to, string text)
{
  try
  {
    string SmtpServerHost = string.IsNullOrEmpty
                (ConfigurationSettings.AppSettings["SmtpServerHost"]) ?
                "localhost" : ConfigurationSettings.AppSettings
                ["SmtpServerHost"];
    string SmtpServerLogin =
string.IsNullOrEmpty(ConfigurationSettings.AppSettings["SmtpServerLogin"]
) ? "" : ConfigurationSettings.AppSettings["SmtpServerLogin"];
```

```
    string SmtpServerPass =
string.IsNullOrEmpty(ConfigurationSettings.AppSettings["SmtpServerPass"])
? "" : ConfigurationSettings.AppSettings["SmtpServerPass"];

    int SmtpServerPort =
string.IsNullOrEmpty(ConfigurationSettings.AppSettings["SmtpServerPort"])
? 25 :
Convert.ToInt32(ConfigurationSettings.AppSettings["SmtpServerPort"]);

    string SmtpFrom =
string.IsNullOrEmpty(ConfigurationSettings.AppSettings["SmtpFrom"]) ?
"monitoring@company.local" :
ConfigurationSettings.AppSettings["SmtpFrom"];


    // To

    MailMessage mailMsg = new MailMessage();

    mailMsg.To.Add(to);

    // From

    MailAddress mailAddress = new MailAddress(SmtpFrom);

    mailMsg.From = mailAddress;


    // Subject and Body

    mailMsg.Subject = "Monitoring ERROR";

    mailMsg.Body = text;


    // Init SmtpClient and send

    SmtpClient smtpClient = new SmtpClient(SmtpServerHost,
SmtpServerPort);

    if (!string.IsNullOrEmpty(SmtpServerLogin) &&
!string.IsNullOrEmpty(SmtpServerPass))

    {

      smtpClient.Credentials = new NetworkCredential(SmtpServerLogin,
SmtpServerPass);

    }

    smtpClient.Send(mailMsg);

  }

  catch (Exception ex)

  {

    //Console.WriteLine(ex.ToString());

    WriteToLog(ex.ToString());

  }

}
```

Obr. 11 Ukázka kódu pro posílání emailového upozornění

9.2 Administrační aplikace

Vstup do webové aplikace je chráněn heslem. Je tak zamezen přístup nepovolaným osobám do nastavení a zobrazení informací o službách. Ukázka přihlašovacího okna je uvedena na následujícím obrázku.

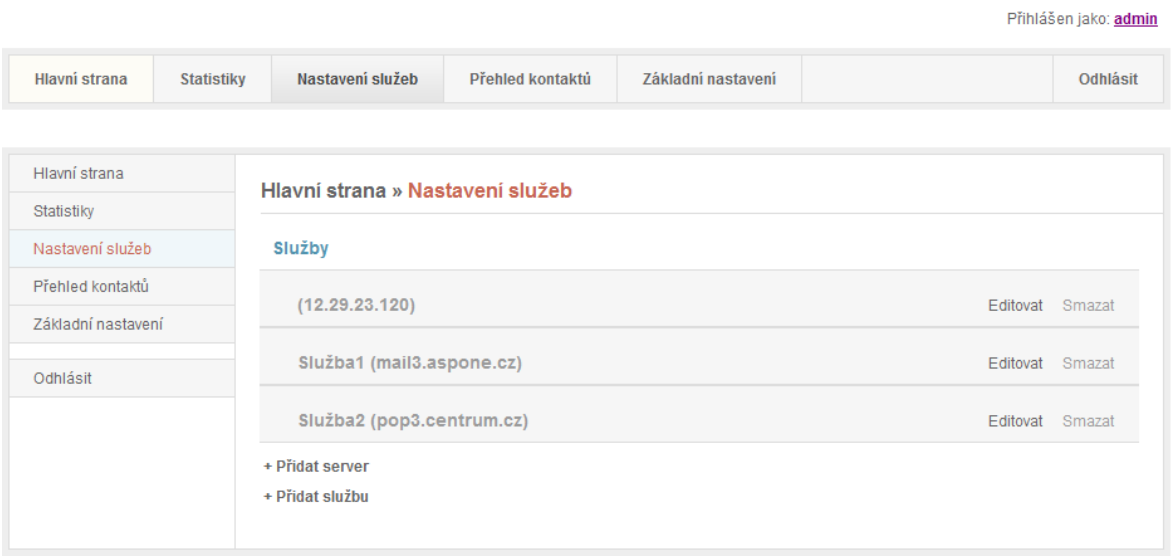


Obr. 12 Přihlašovací formulář

Výsledná aplikace je vytvořena za pomocí *MasterPage*, která je použita ve všech stránkách webové aplikace. Skládá se z menu aplikace a části pro zobrazení aktuální stránky.

Jednotný vzhled stránek byl nadefinován také pomocí společné *StyleSheet.*

Kontrola přihlášení je po celou dobu práce s aplikací zabezpečována pomocí session proměnných serverové ovládací komponenty *Login*, která je součástí ASP.NET. Díky této komponentě není třeba při každé editaci položek kontrolovat, zda je uživatel přihlášen a lze již přímo definovat editovací metody.



Obr. 13 Výsledná webová aplikace

9.2.1 Menu

Pohyb v nabídkách je umožněn pomocí menu v horní části aplikace. Obsaženy jsou tyto položky:

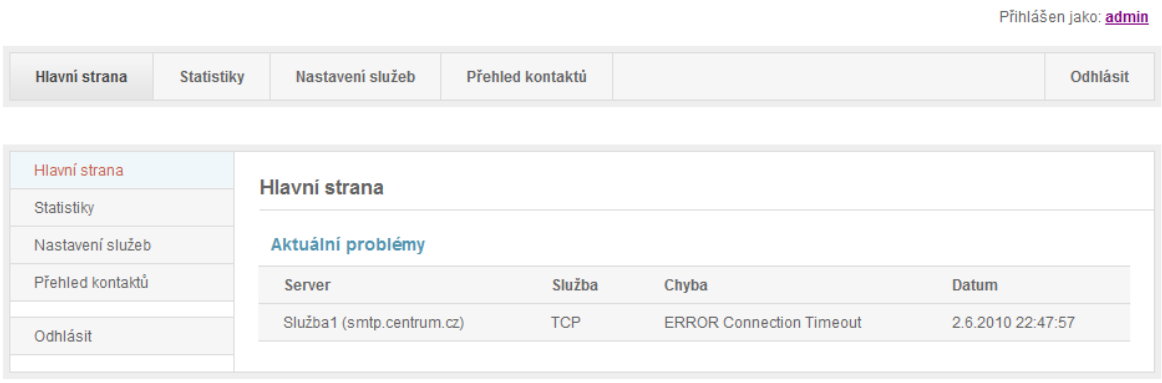
- Hlavní strana
- Statistiky
- Nastavení služeb
- Přehled kontaktů
- Základní nastavení
- Odhlásit

Jednotlivé položky menu jsou více popsány níže.

9.2.2 Hlavní strana

Tato stránka slouží pro obecný přehled o aktuálních chybách jednotlivých monitorovaných služeb. Je zde uveden vždy *Název* serveru, *Typ* chybové služby, *Datum* naměřené chyby. Aktuální *Chyba* služby je zde uvedena také.

Pro tabulkový výpis jednotlivých služeb zde byla použita komponenta *Repeater*.



Obr. 14 Ukázka výpisu chybových služeb

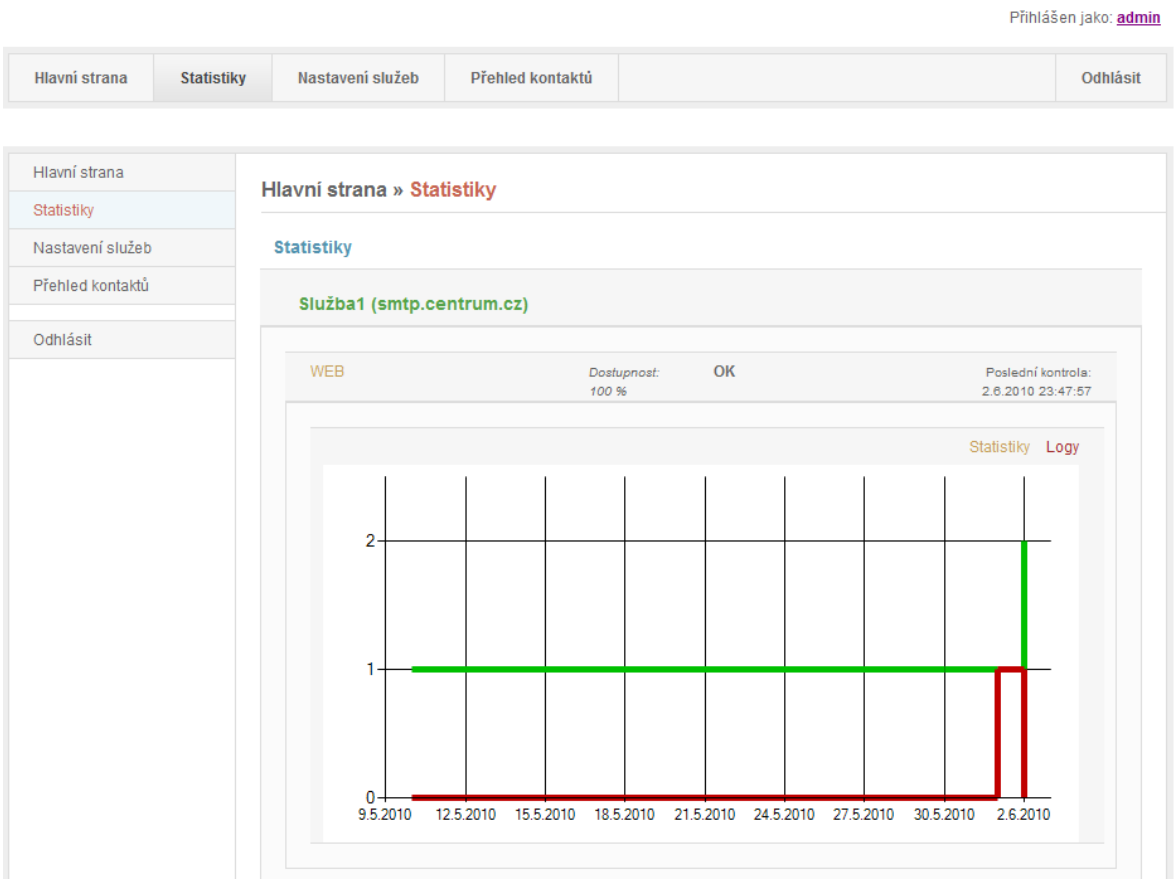
9.2.3 Stránka Statistiky

Statistiky a logy monitorovaných služeb lze zobrazit pomocí stránky Statistiky zvolené pomocí menu webové aplikace. Podobně jako u stránky *Nastavení služeb* byla také zde, pro zobrazení strukturovaného seznamu serverů a k němu přiřazených monitorovaných služeb,

použita komponenta *Repeater*. Tato komponenta nejlépe umožňuje vlastní návrh zobrazení položek a neklade žádné meze pro použití dalších ovládacích prvků.

Kliknutím na jednotlivé položky seznamu serverů lze plynule rozbalit seznam přiřazených služeb, který se zobrazí jako podnabídka daného serveru. Toto plynulé rozbalení umožnila komponenta AJAXu *CollapsiblePanelExtender*, která je vložena pomocí *Repeateru* ke každé položce seznamu serverů.

Pro správné zobrazení přiřazených služeb byl vnořen další *Repeater*, který přebírá některá data z předchozího. Také v podnabídce služeb lze kliknout na jednotlivé položky a plynulým rozbalením již zobrazit statistiky dané služby či přímo výpis logů z databáze. Ukázka je vyobrazena níže.



Obr. 15 Ukázka vygenerované statistiky dostupnosti

Pro výpis statistik a také výpis seznamu služeb, které jsou zobrazeny jako podnabídky, byl vytvořen zvláštní LINQ dotaz vložený do kódu stránky. Ukázka je uvedena níže.

```
MonitoringDataContext db = new MonitoringDataContext();
var data = from f in db.Servers
    orderby f.Hostname
    select new
```

```
    {
        ID = f.ID,
        ContactI = f.ContactID,
        IP = f.IP,
        Hostname = f.Hostname,
        Fullname = f.Hostname + " (" + f.IP + ")",
        Enabled = f.Enabled,
        Note = f.Note,
        ServerServices = from d in f.ServerServices
            select new
            {
                ID = d.ID,
                ServerID = d.ServerID,
                Type = d.Type,
                RunPeriod = d.RunPeriod,
                TriggerCount = d.TriggerCount,
                WebRequest = d.WebRequest,
                ResponseText = d.ResponseText,
                Port = d.Port,
                Enabled = d.Enabled,
                ServerServiceActualProblems =
                                d.ServerServiceActualProblems.First(),
                ServerServiceLogs = d.ServerServiceLogs,
                Chart = from c in d.ServerServiceLogs
                        group c by c.Date.Date into y
                        select new
                        {
                            Day = y.Key,
                            CountOK = y.Where(w =>
                                w.State.StartsWith("OK")).Count(),
                            CountERROR = y.Where(w =>
                                w.State.StartsWith("ERROR")).Count()
                        },
                Availability = d.ServerServiceLogs.Count()==0?0:
                 (d.ServerServiceLogs.Select(s=>s.State.StartsWith("OK"))
                               .Count()/ d.ServerServiceLogs.Count())*100
            }
    };
```

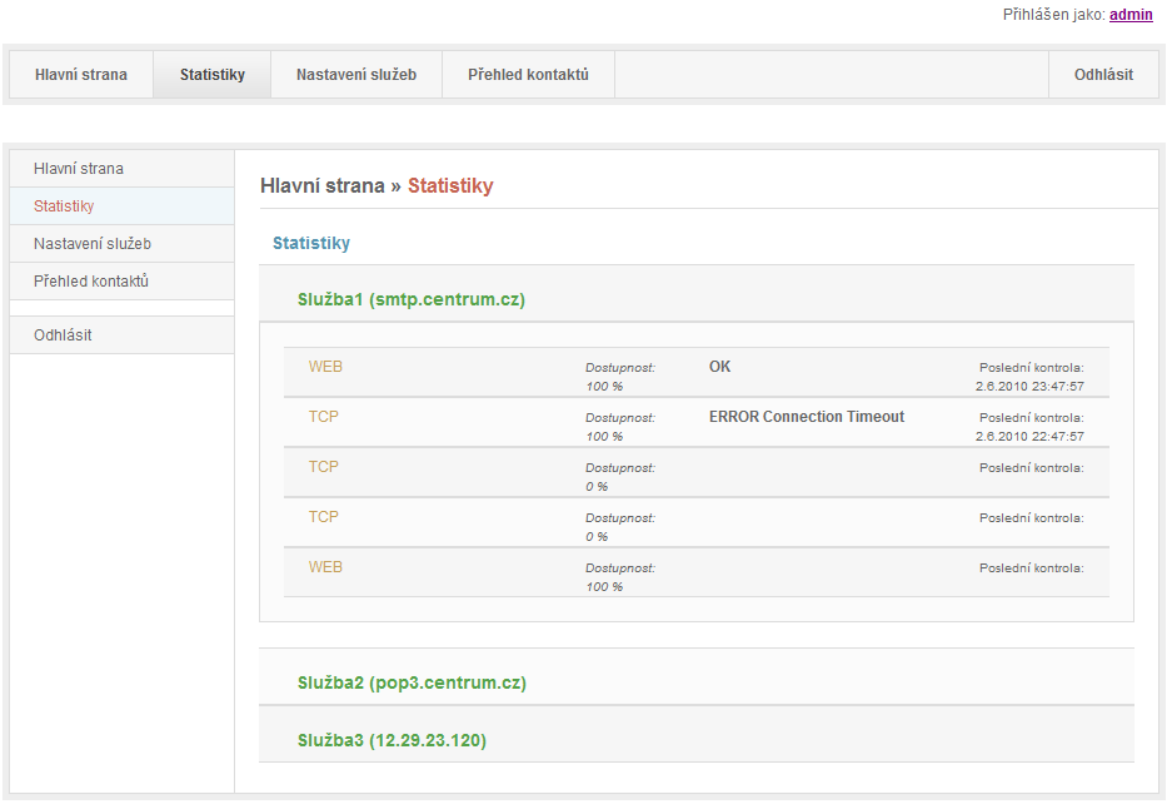
Obr. 16 Ukázka dotazu LINQ

9.2.4 Zobrazení informací o stavu služeb

Pomocí seznamu serverů, zobrazeného na stránce Statistiky, lze rozbalit seznam služeb, které jsou k danému serveru přiřazeny.

Přímo zde lze již vidět aktuální stavy jednotlivých služeb jako je uvedeno na obrázku.

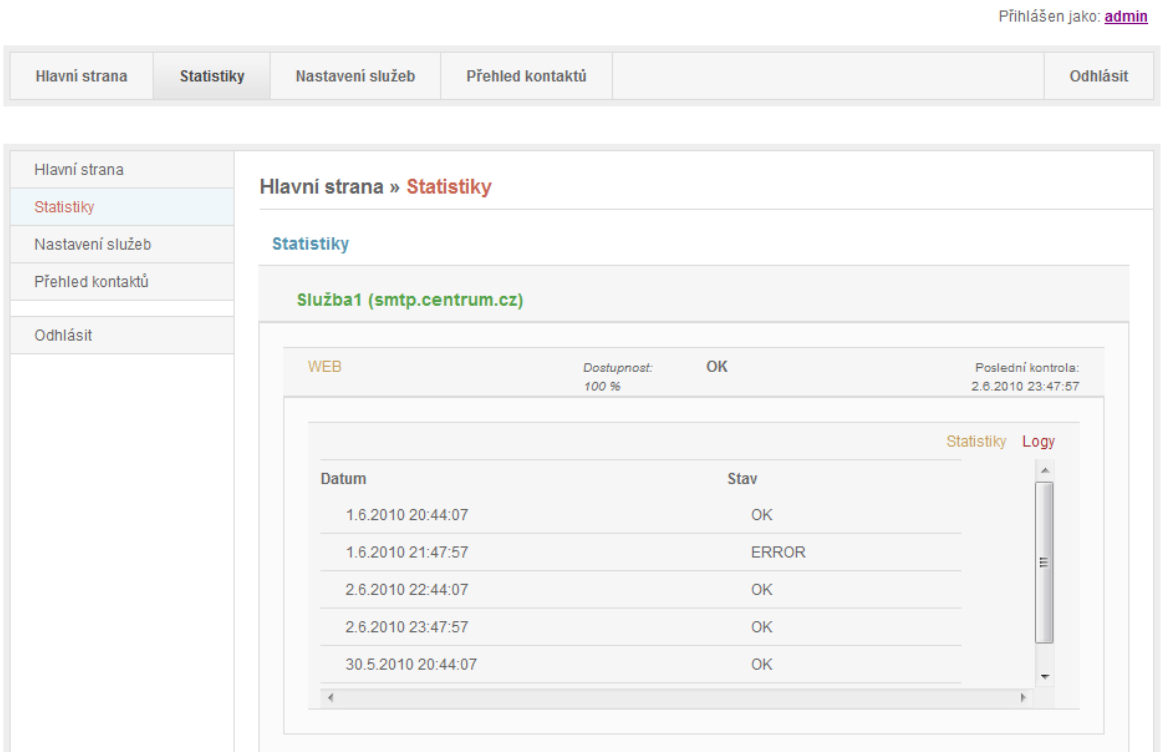
Ukázka rozbalení služeb je uvedena níže.



Obr. 17 Zobrazení služeb serveru

Zobrazení aktuálního stavu zahrnuje výpis *Názvu*, *Typu* a průměrné dostupnosti služby dle dat uložených v databázi. Dále je zde uveden *Stav* poslední kontroly dle přijaté odpovědi serveru a *Datum* této kontroly.

Pro zobrazení již samotných statistik slouží odkazy vytvořené z položek seznamu služeb. Pomocí nich lze plynule zobrazit statistiky vygenerované z logu *Monitorovací aplikace* nebo přímo zobrazit logy měření služby.



Obr. 18 Zobrazení logů z provedených kontrol

9.2.5 Stránka Nastavení služeb

Prostřednictvím této stránky je umožněno nastavení mnoha parametrů aplikace.

Jako první je na stránce zobrazen výpis skupin serverů, ke kterým jsou definované služby přiřazeny. Hned pod tímto seznamem jsou umístěna tlačítka pro přidání dalších serverů a příslušných služeb.

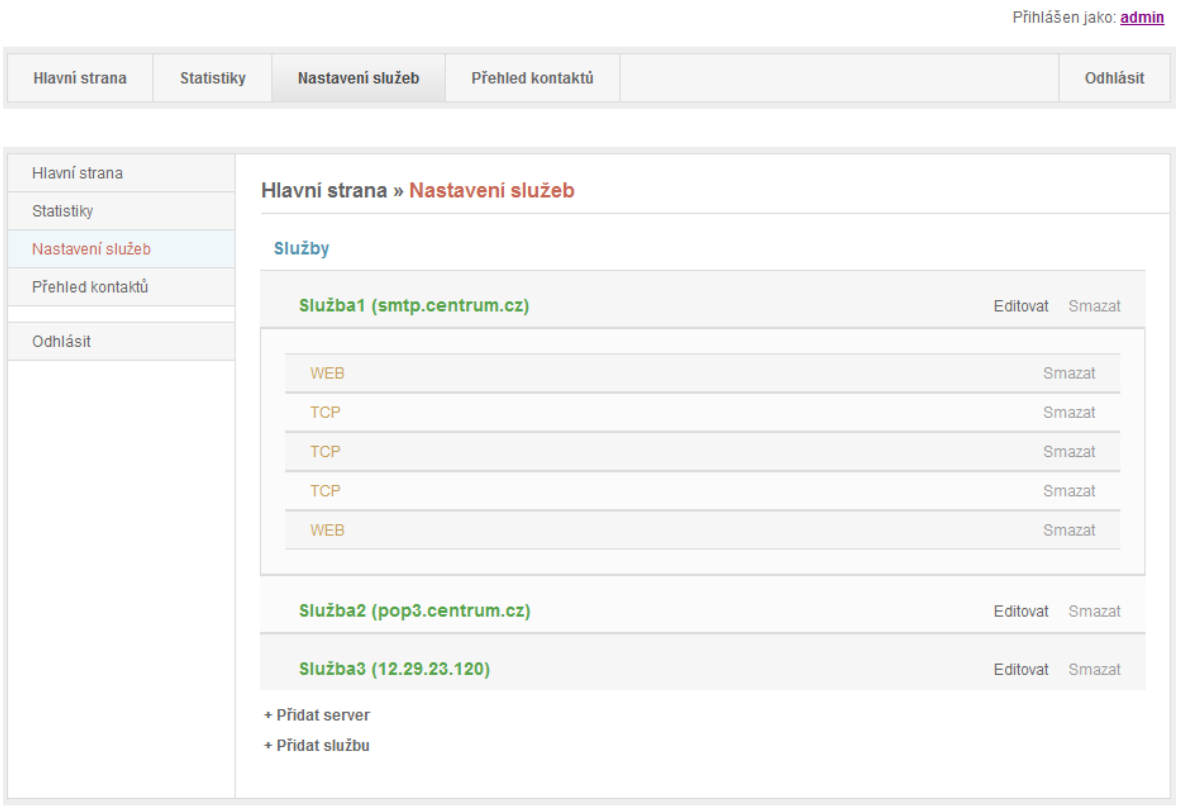
Výpis seznamu serverů byl implementován, podobně jako u stránky Statistiky, pomocí komponenty *Repeater*, do kterého byl pro umožnění zobrazení a editace přiřazených služeb vnořen *Repeater* další.

Obr. 19 Stránka Nastavení služeb

V seznamu serverů lze jednotlivé položky editovat pomocí tlačítek *Editovat* a také mazat tlačítky *Smazat*, která jsou umístěna v pravé části tabulky.

Po kliknutí na jednotlivé servery v seznamu se rozbalí seznam přiřazených služeb. Toto bylo zajištěno pomocí AJAX komponenty *CollapsiblePanelExtender*, která je použita také pro dynamické rozbalení editace serverů a služeb.

Zde lze pak tyto služby dále editovat a také mazat či dočasně vypnout. Aplikace s rozbalenými službami je vyobrazena na následujícím obrázku.



Obr. 20 Ukázka výpisu služeb jednotlivých serverů

9.2.5.1 Editace serveru

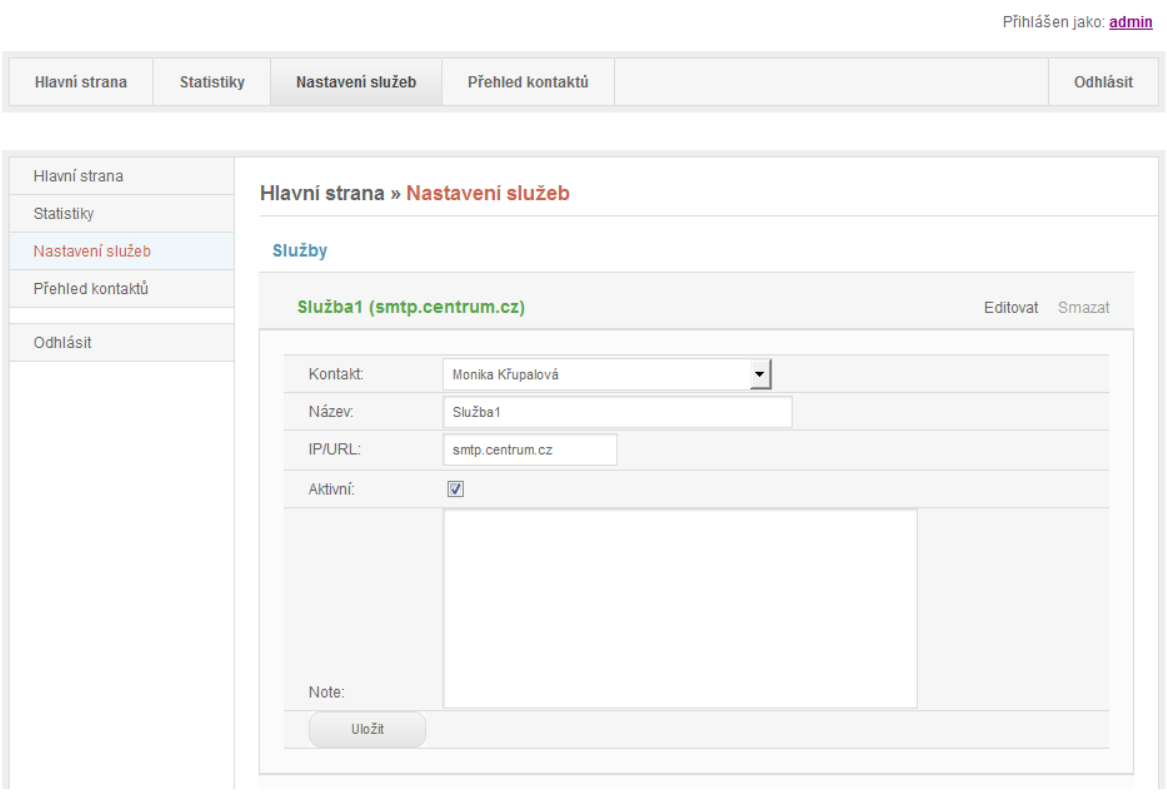
Po kliknutí na tlačítko editovat se rozbalí formulář pro editaci daného serveru.

Formulář obsahuje nastavení pro definování *Kontaktu*, na který se budou posílat upozornění o chybách, dále *Název*, *IP* případně *URL* adresu serveru. Zde lze také celou skupinu monitorovaných služeb, které jsou k tomuto serveru přiřazeny, dočasně vypnout a poté opět zapnout pomocí položky *Aktivní*.

Tento formulář je před uložením validován pomocí komponenty *RequiredFieldValidator* u položky *IP/URL*.

Kliknutím na tlačítko *Uložit* se zadaná data zkontrolují a poté se tato data aktualizují v databázi. Na závěr editace je zobrazovaný seznam serverů aktualizován.

Formulář editace lze také znovu zavřít opětovným klikem na tlačítko *Editovat*.



Obr. 21 Editace serveru

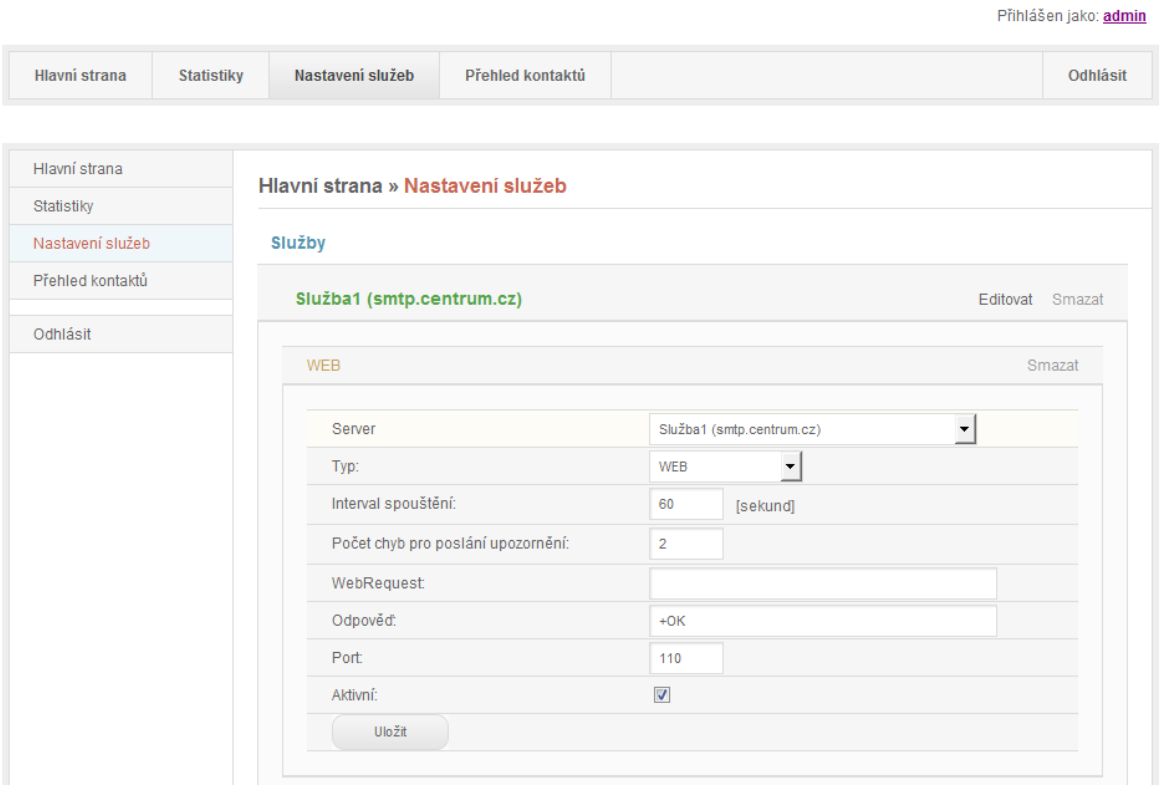
9.2.5.2 Editace služeb

Po kliknutí na server, ve kterém jsou uloženy již samotné služby, lze tyto služby dále editovat. Zde je možno měnit *Server*, ke kterému má být služba přiřazena a také *Typ* služby, která se má kontrolovat. Jak už bylo popsáno výše, dle této položky jsou následně použity také další položky formuláře. Mezi tyto položky patří *WebRequest*, dle které se volá požadavek u služby typu „*Web*“ a položka *Port*, která se použije pouze v případě služby typu „TCP“.

Dalšími položkami, sloužícími pro editaci služeb jsou *Interval spouštění* ukládán v sekundách a *Počet chyb pro posílání upozornění*. Pomocí této položky lze zajistit, že dohledový systém nebude posílat chybová hlášení při detekování každé chyby, ale až po více opakovaných chybách. Toto slouží také pro minimalizaci počtu poslaných chybových hlášení.

Důležitou položkou je zde *Odpověď*. S touto odpovědí se při monitorování služby porovnávají přijaté odpovědi a při neshodě se detekuje chyba. Monitorování služby lze také dočasně vypnout pomocí položky *Aktivní*.

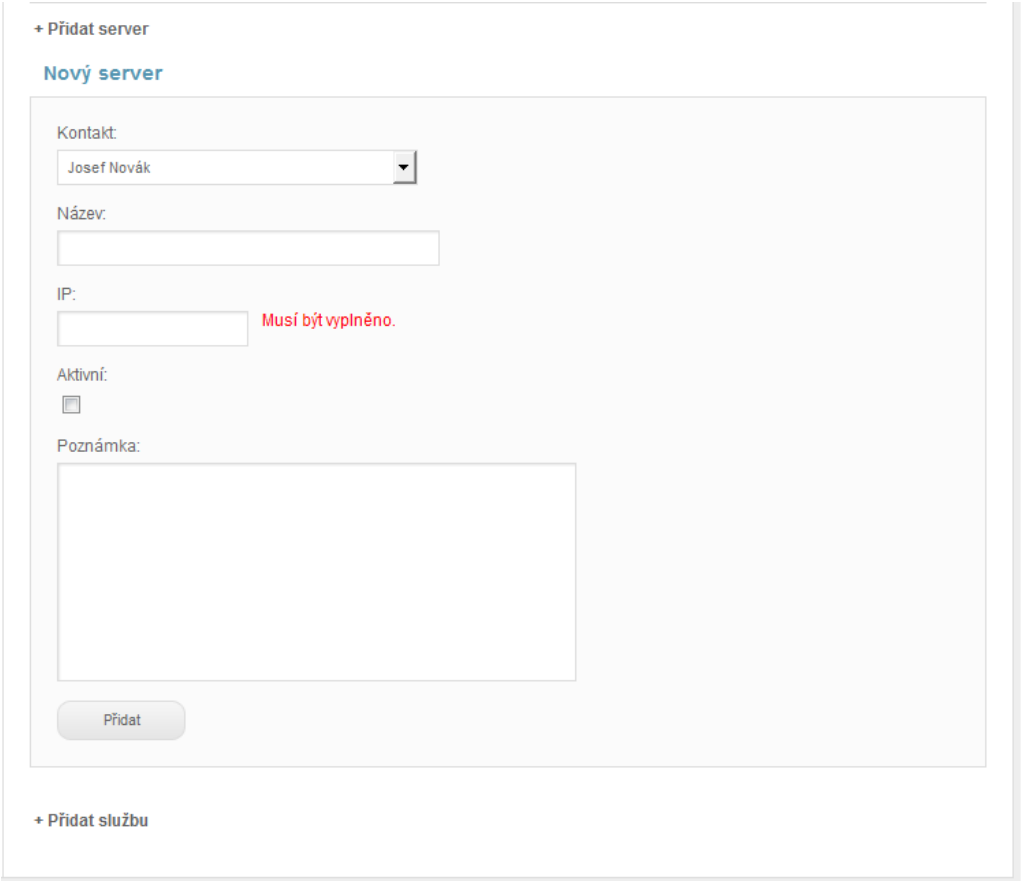
Pro uložení editovaných hodnot slouží tlačítko *Uložit* nebo je možno editaci zavřít opětovným klikem na název služby.



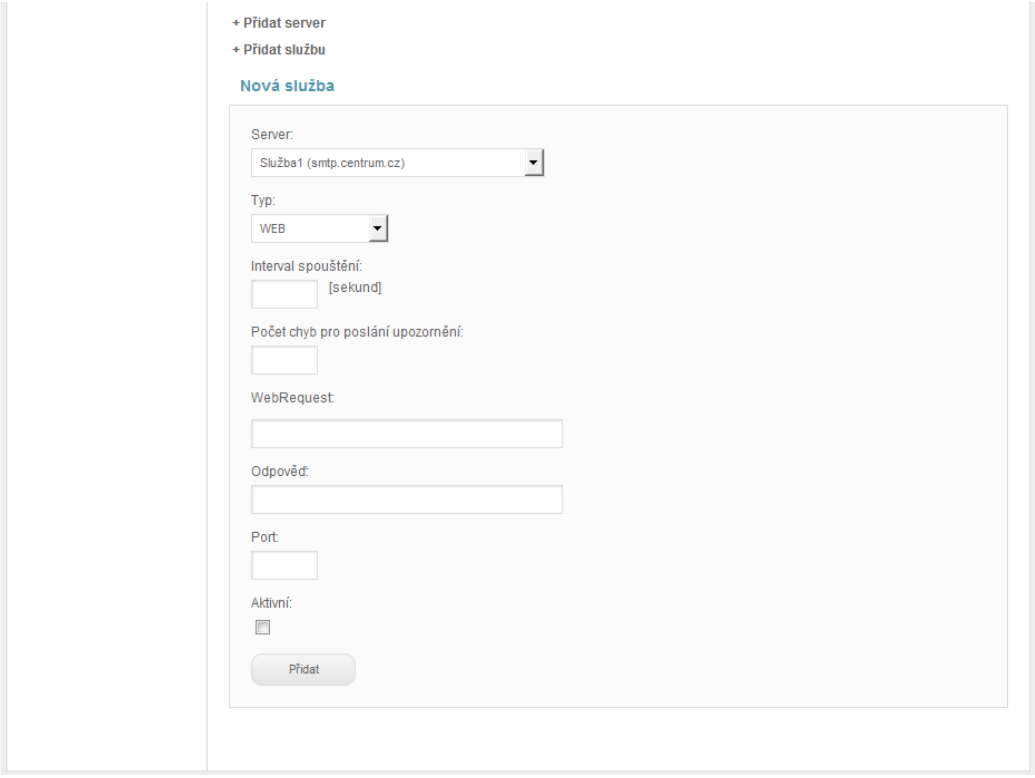
Obr. 22 Editace služby

9.2.5.3 Přidání nových položek

Ve spodní části stránky *Nastavení služeb* se nacházejí tlačítka pro rozbalení formulářů k vytvoření dalších serverů a přiřadit k serverům další služby. Jednotlivé položky mají stejný účel, jak již bylo popsáno výše u editací těchto položek. Proto zde přikládám pouze ukázky přidání položek.



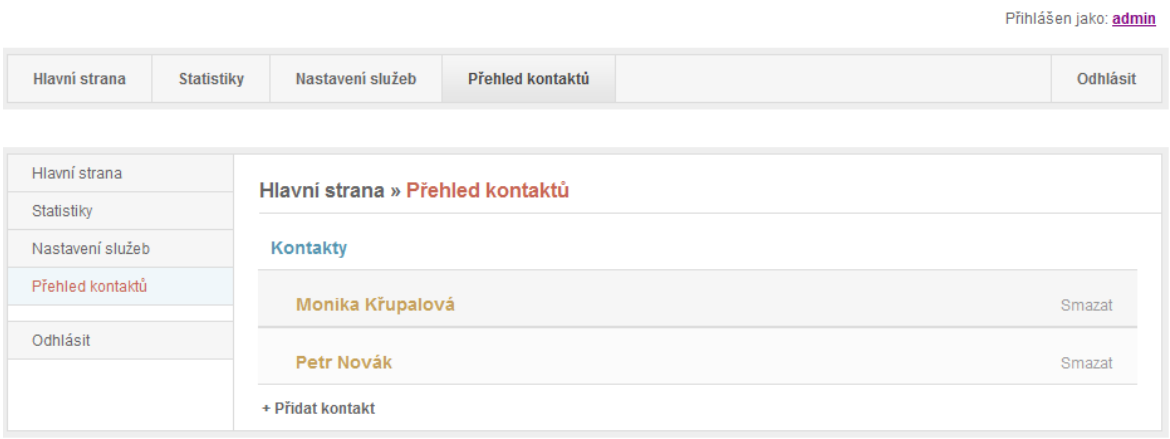
Obr. 23 Přidání nového serveru



Obr. 24 Přidání služby k serveru

9.2.6 Stránka Přehled kontaktů

Pomocí této stránky je umožněna správa kontaktů, na které jsou posílány chybová hlášení.

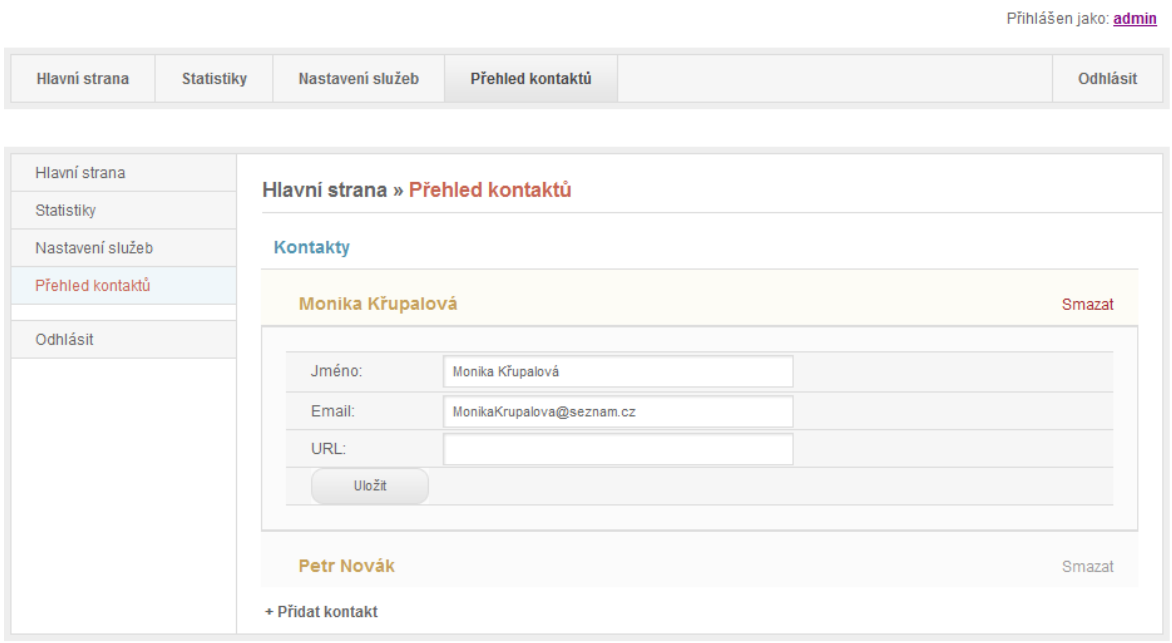


Obr. 25 Zobrazení kontaktů

Pomocí komponenty *Repeater* je zde vypsán seznam všech kontaktů. Přístup k editaci je zajištěn pomocí AJAX komponenty *CollapsiblePanelExtender*. Tato komponenta po kliku na jednotlivý kontakt umožní plynulé rozbalení formuláře pro editaci.

9.2.6.1 Editace kontaktů

Po kliknutí na položku v seznamu kontaktů lze tento kontakt editovat. Zde bylo použito komponenty *RequiredFieldValidator* pro zajištění kontroly vyplnění položky *Jméno*. Dále byla také definována položka této komponenty *ValidationGroup* identifikována také pomocí *ID* položky. Toto bylo nutné pro správnou validaci *Repeateru* pomocí kterého je seznam kontaktů zobrazován.
Editační část aplikace je ukazuje následující obrázek.



Obr. 26 Editace kontaktů

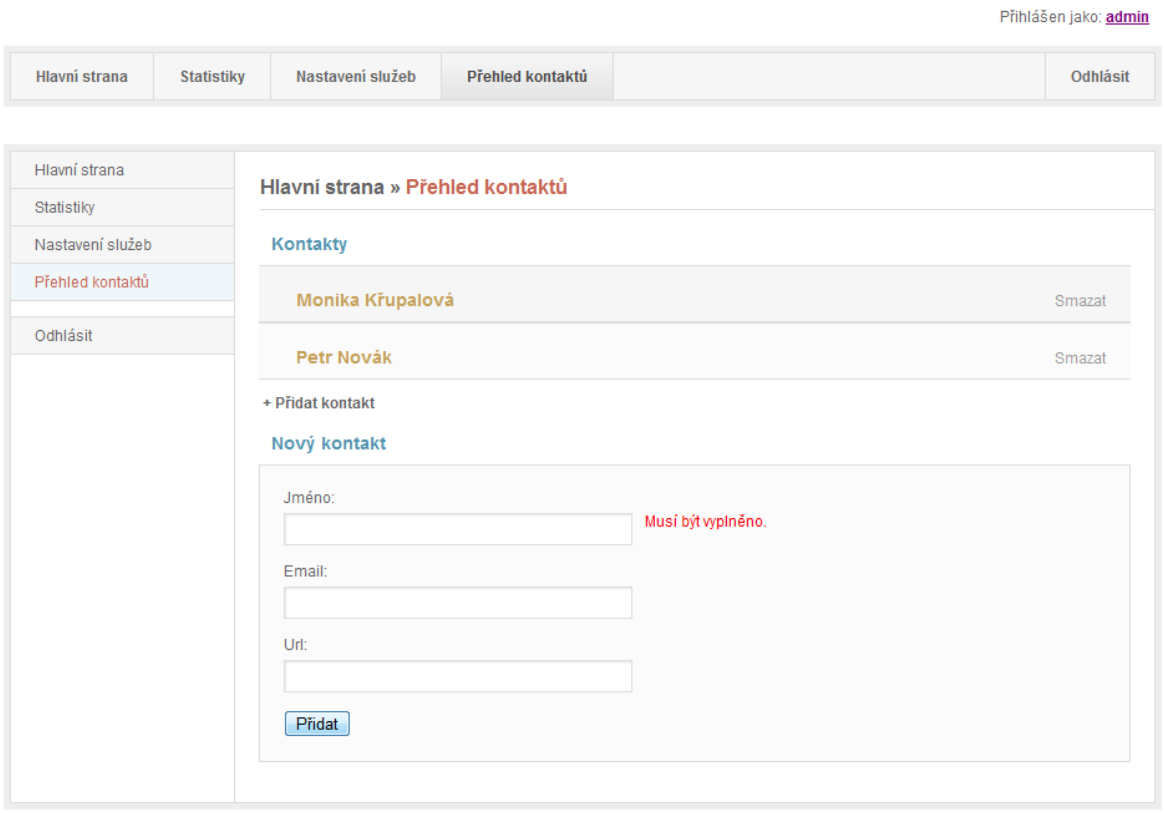
V editačním formuláři lze měnit položku *Jméno*, sloužící pro identifikaci kontaktu, dále lze editovat *Email* a *SMS*. Dle těchto údajů aplikace posílá upozornění o chybách.

Poslední položka *URL* slouží pro zadání webové adresy, která se bude volat při detekované chybě služby. Pomocí tohoto nastavení lze definovat také spuštění stránky a ta může již sama poslat upozornění o chybách dalšími způsoby.

9.2.6.2 Přidání kontaktů

Pomocí stránky Přehled kontaktů, kde je ve spodní části tlačítko *Přidat kontakt*, lze definovat nové kontakty.

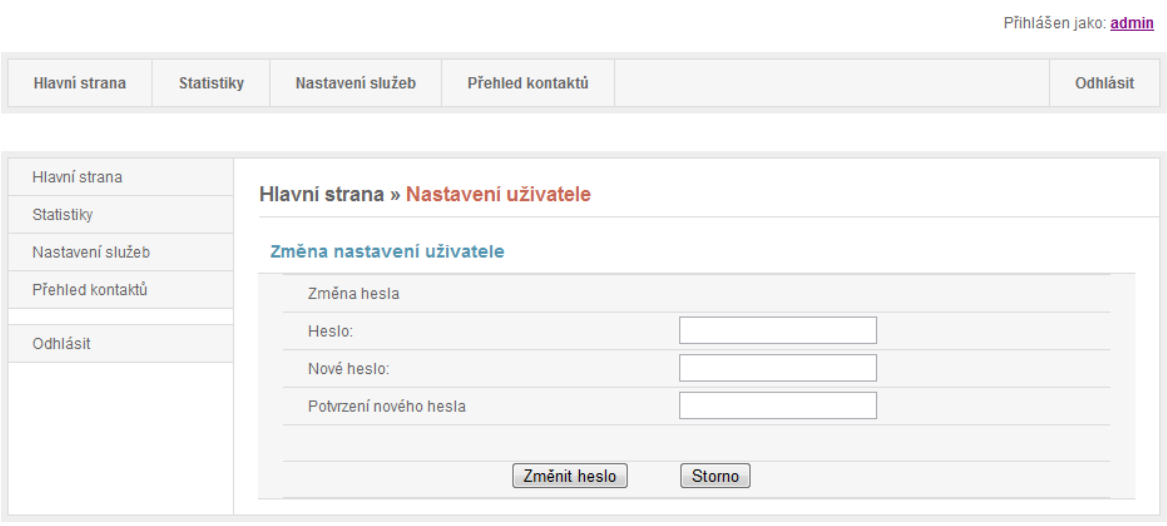
Zde byla pro rozbalení formuláře taktéž použita komponenta *CollapsiblePanelExtender* a identifikace *ValidationGroup* pro správnou validaci všech součástí stránky.



Obr. 27 Vložení nového kontaktu

9.2.7 Změna hesla pro přístup do aplikace

Heslo pro přihlášení do aplikace lze změnit po kliknutí na odkaz se jménem přihlášeného uživatele, který je umístěn v pravém horním rohu aplikace. Ke změně hesla je také nutno zadat heslo stávající pro zamezení změny hesla nepovolanou osobou.



Obr. 28 Stránka pro změnu hesla

ZÁVĚR

V této diplomové práci byla popsána tématika vývoje aplikace dohledového systému serverů za pomoci dvou aplikací.

Jednotlivé použité technologie jsou popsány v teoretické části. Díky tomuto přehledu si může čtenář vytvořit hrubý přehled o možnostech použitého vývojového prostředí Microsoft Visual Studio, nástroje pro tvorbu databází Microsoft SQL Server, technologii LINQ a dalších. Teoretická část dále popisuje možnosti dohledu nad serverovými službami a také motivaci pro monitorování služeb.

V rámci praktické části diplomové práce byl vytvořen dohledový systém serverů pro monitorování serverových služeb. Mezi hlavní cíle patřilo vytvoření systémové služby pro monitoring serverových služeb a webové aplikace pro administraci. Cílem bylo vytvořit dohledový systém, který bude moci monitorovat více typů služeb a v případě detekce chyby poslat chybové hlášení.

Důležitým prvkem webové aplikace je zobrazení statistik a výpis logů jednotlivých služeb. Nastavení monitorovaných služeb lze měnit také v této aplikaci včetně přidání a odebrání služeb.

Obsah kapitol praktické části postupně nastínil definici a analýzu potřebných funkcí a vlastností dohledového systému. Dále je také popsán podrobný popis architektury, navržená struktura databáze a popis vlastností jednotlivých tabulek databáze.

Praktickou část zakončuje kapitola, která obsahuje podrobný popis obou naprogramovaných aplikací dohledového systému. Čtenář zde dostává přehled nejen o samotném systému, ale také o činnosti jednotlivých součástí aplikací.

Výsledkem této práce je funkční systém plnící funkčnost dohledového systému, který lze dále rozšiřovat bez větších zásahů do jádra.

V rámci rozvoje do budoucna se počítá s přidáním dalších typů upozornění, například pomocí SMS a také umožnění monitoringu pro další typy serverových služeb.

Z hlediska použitelnosti je systém připraven na ostrý testovací provoz. Jednotlivé aplikace poté čeká dostatečné odladění díky nasazení do ostrého provozu. Po odladění všech případných chyb bude systém připraven pro konečné použití.

ZÁVĚR V ANGLIČTINĚ

In this thesis was described technology for development of server monitoring system using two applications.

Used technologies are described in the theoretic part. This involves possibilities of development environment Microsoft Visual Studio, relational model database server Microsoft SQL Server, LINQ technology and others. The theoretic part also describes possibilities of server monitoring and monitoring motivation.

Within the frame of the practical part was developed monitoring server system for monitoring server services. One of the main points of this project was to develop system service for the monitoring and web application for administration. The main point of this project was monitoring of more service types and in case of error detection to send error message.

A part of the web application is displaying information of monitored services and listing of logs. Also the settings of monitored services can be changed in this web application including editing, adding and deleting of services.

The practical part of thesis describes an analysis of needs and functions of monitoring system. Further description of system architecture, database structure and description of database tables follow.

The practical part ends with detailed description of both developed applications of monitoring system. This part describes developed system with detailed description of applications parts.

The result of this work is a functional application for monitoring of server services which can be expanded with other service types for monitoring without too much interference in the application itself.

There are many possibilities to improve the whole system in the future by sending error messages via SMS and by others service types of server services for monitoring.

The information system is now fully ready for testing. After debugging of all application mistakes will be system prepared for use.

SEZNAM POUŽITÉ LITERATURY

[1] Microsoft Visual Studio [online]. 2002 [cit. 2010-06-07]. Wikipedia. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Microsoft_Visual_Studio>.

[2] SHARP, John. Microsoft Visual C-Sharp 2008 Krok za Krokem. Brno : Computer Press, 2008. 592 s.

[3] AGARWEL,Vidya Vrat. HUDDLESTON, James. Databáze v C-Sharp 2008 Průvodce programátora. Brno: Computer Press 2009. 424s. ISBN 978-80-251-2309-6.

[4] NEGEL, Christian. EVJEN, Bill. GLYNN, Jay. WATSON, Karli. SKINNER, Morgan. C-Sharp 2008 Programujeme Profesionálně. Brno: Computer Press 2009. ISBN 978-80-251-2401-7.

[5] Výběr několika novinek z Visual Studio 2010 [online]. 2009 [cit. 2010-06-08]. Daquas. Dostupné z WWW: <http://www.daquas.cz/Articles/278-vyber-nekolika-novinek-z-visual-studio-2010.aspx>.

[6] MACDONALD, Matthew, SZPUSZTA, Mario. ASP.NET 3.5 a C-Sharp 2008 : Tvorba dynamických stránek profesionálně. Jan Pokorný, Jan Gregor. 1. vyd. [s.l.] : Zoner Press, 2008. 1584 s. ISBN 978-80-7413-008-3.

[7] BILL, Evjen, et al. ASP.NET 2.0 : Programujeme profesionálně. Karel Voráček. Brno : Computer Press, a.s., 2006. 1224 s. ISBN 978-80-251-1473-5.

[8] Visual Studio 2010 [online]. 5/10/2009 [cit. 2010-06-08]. Ondrej Stastny. Dostupné z WWW: <http://www.ondrejstastny.cz/Blog/Post/Visual-Studio-2010>.

[9] Moderní výuka [online]. 2009 [cit. 2010-06-08]. Serverové technologie. Dostupné z WWW:

<http://www.modernivyuka.cz/Serverov%C3%A9OSatechnologie/SQLServer/tabid/434/ctl/Details/mid/1339/ItemID/208/language/cs-CZ/Default.aspx>.

[10] ZIVE.cz [online]. 12. 1. 2009 [cit. 2010-06-08]. DB admine, raduj se. SQL Server 2008 přichází. Dostupné z WWW: <http://www.zive.cz/clanky/db-admine-raduj-se-sql-server-2008-prichazi/sc-3-a-145198/default.aspx>.

[11] PUŠ, Petr. Vyvojar.cz [online]. 24. ledna 2008 [cit. 2010-06-08]. Úvod do LINQ. Dostupné z WWW: <http://www.vyvojar.cz/Articles/563-uvod-do-linq.aspx>.

[12] MAREŠ, Amádeo. ZIVE.cz [online]. 13. 8. 2009 [cit. 2010-06-08]. Staňte se programátorem: Kouzelný LINQ. Dostupné z WWW: <http://www.zive.cz/clanky/stante-se-programatorem-kouzelny-linq/sc-3-a-148308/default.aspx>.

[13] BRUST, Andrew J. FORTE ,Stephen. Mistrovství v programování SQL Serveru 2005. Brno: Computer Press, 2007. 848s. ISBN 978-80-251-1607-4.

[14] PROSISE, Jeff. Programování v Microsoft .NET. Brno: Computer Press, 2003. 736s. ISBN 80-7226-879-1

[15] PETZOLD, Charles. Mistrovství ve Windows Presentation Foundation. Brno: Computer Press, 2008. 928s. ISBN 978-80-251-2141-2

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ADO	ActiveX Data Objects
ADO.NET	ActiveX Data Object.NET
AJAX	Asynchronous JavaScript and XML – technologie pro vývoj webových aplikací na bázi JavaScript kódu a XML.
ASP	Active Server Pages – serverově aktivní stránky
ASP	Active Server Pages – klasické skriptování. Technologie před nástupem ASPNET.
ASP.NET	Active Server Pages.Network
CLR	Common Language Runtime – prostředí pro běh kódu v rámci .NET Frameworku.
CSS	Cascading Stye Sheets – kaskádové styly
DOM	Document Object Model
FTP	File Transfer Protocol – protokol pro přenos binárních a textových souborů
GUI	Graphical User Interface – uživatelské grafické rozhraní
HTML	HyperText Markup Language – hypertextový značkovací jazyk
HTTP	HyperText Transfer Protocol – protokol pro přenos hypertextu
ICMP	Internet Control Message Protocol
ID	Identification
IP	Internet Protocol
LINQ	LINQ to SQL – nová technologie pro práci s databází a tabulkami.
MPF	Managed Package Framework
MSDN	Microsoft Developer Network – rozsáhlá vývojářská dokumentace a nápověda k dispozici volně na Internetu nebo na DVD.
MSSCCI	Microsoft Source Code Control

MSSQL	Microsoft SQL Server – databázový server společnosti Microsoft pro operační systém Microsoft Windows.
MVC	Model View Controler – softwarová architektura, která rozděluje datový model aplikace.
ODBC	Open Database Connectivity – standardizované softwarové API pro přístup k databázovým systémům
POP3	Post Office Protocol version 3 – je internetový protokol pro příjem (stahování) emailových zpráv ze vzdálených serverů.
SDK	Software development kit – rozšiřující balíček pro vývojáře
SMTP	Simple Mail Transfer Protocol – je internetový protokol pro emailovou komunikaci mezi počítači. Standardně používá TCP port 25.
SMTP	Simple Mail Transfer Protocol – protokol určený k posílání emailových zpráv
SNMP	Simple Network Management Protocol – protokol používaný pro monitoring síťových zařízení
SOAP	Simple Object Access Protocol – standard pro přenos XML dat pomocí internetu.
SSMS	SQL Server Management Studio
TCP	Transmission Control Protocol – základní protokol pro komunikaci na Internetu. Protokol zajišťuje spolehlivé doručení paketů ve správném pořadí.
TFS	Team Foundation Server – softwarové prostředí pro vývoj aplikací na platformě .NET
UML	Unified Modeling Language – jazyk pro popis a návrh aplikací a softwarového inženýrství.
WPF	Windows Presentation Foundation
WWW	World Wide Web – celosvětová pavučina – služba, která je založena na protokolu HTTP.
XHTML	eXtensible HyperText Markup Language

XML Extensible Markup Language – jeden z nejpoužívanějších značkovacích jazyků využívaný pro různé přenosy dat a popisy struktur.

XSLT Extensible Stylesheet Language Transformations – Transformace slouží k převodům zdrojových dat ve formátu XML do libovolného jiného formátu

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: UKÁZKA KÓDU TŘÍDY MONITORINGREQUEST

```
using System;
using System.Configuration;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;

// URLs
using System.Net;
using System.IO;

// Mails
using System.Net.Mail;

namespace Server
{
    public class MonitoringRequest
    {
        /// <summary>
        /// Internal thread pool event
        /// </summary>
        private ManualResetEvent doneEvent;
        public int sleepTime = 0;
        private int id = 0;
        private string response = string.Empty;
        private string type = string.Empty;
        private string webrequest = string.Empty;
        private string host = string.Empty;
        private int port = 0;
        private string responsetext = string.Empty;
        private DateTime date = DateTime.Now;

        /// <summary>
        /// Main constructor for MonitoringRequest class
        /// </summary>
        /// <param name="doneEvent"></param>
        public MonitoringRequest(int id, string type, string responsetext,
ManualResetEvent doneEvent)
        {
            this.doneEvent = doneEvent;
            this.id = id;
            this.type = type;
            this.responsetext = responsetext;
        }
        public MonitoringRequest(int id, string type, string webrequest, string host,
int port, string responsetext, ManualResetEvent doneEvent)
        {
            this.doneEvent = doneEvent;
            this.id = id;
            this.type = type;
            this.webrequest = webrequest;
            this.host = host;
            this.port = port;
            this.responsetext = responsetext;
        }


        // METHODS
        public void ThreadPoolCallback(Object threadContext)
```

```
        {
            int threadIndex = (int)threadContext;
            Console.WriteLine("thread {0} started...", threadIndex);
            DoCheck();
            Console.WriteLine("thread {0} ended...", threadIndex);
            this.doneEvent.Set();
        }
        private void DoCheck()
        {
            switch (this.type)
            {
                case "PING":
                    this.response = new PingRequest(this.host).Check();
                    break;

                case "TCP":
                    this.response = new TcpRequest(this.host, this.port,
this.responsetext).Check();
                    break;

                case "WEB":
                    this.response = new WebRequest(this.host, this.port,
this.webrequest, this.responsetext).Check();
                    break;

                default:
                    break;
            }
        }


        // PROPERTIES
        public string Response
        {
            get { return this.response; }
            set { this.response = value; }
        }
        public int ID
        {
            get { return this.id; }
            set { this.id = value; }
        }
        public DateTime Date
        {
            get { return this.date; }
            set { this.date = value; }
        }
    }
}
```