Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Bakalářská práce

Návrh logovacího a monitorovacího modulu do webového portálu Areus .NET 3.0

Plzeň, 2014 Stanislav Boháč

Čestné prohlášení

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

V Plzni dne _____ _____

Stanislav Boháč

Poděkování

Rád bych touto cestou poděkoval Doc. Dr. Ing. Janě Klečkové za její konzultace, přípomínky a rady ohledně mé bakalářské práce. Také děkuji mé rodině za její trpělivost a společnosti Ambro Systems spol. s r.o. za jejich podněty a doporučení.

Abstrakt

Práce je vytvořena na žádost české IT společnosti Ambro Systems spol s r.o., která vyvíjí webový business to business portál Areus. V úvodu jsou vysvětleny pojmy nezbytné ke znalosti problematiky logování, monitorování a srovnání existujících řešení. Dále se práce zabývá optimálním návrhem obou modulů a řešením implementace, výstupního formátu a manipulací s logy. Závěr je věnován grafickému uživatelskému rozhraní, a administrátorskému pohledu na výsledná data.

Klíčová slova: log, monitorování.

Abstract

This work has been created on demand from czech IT company Ambro Systems Ltd., which develops web B2B portal Areus. The basics of logging, monitoring and comparison current solutions are introduced at the beginning of the thesis. Further the work deals with optimal desing of both modules, and solving implementation, the output format of log and handling with logs. The conclusion is dedicated to a graphical part and for a view to result data.

Keywords: log, monitoring.

Obsah

1 Úvod

V dnešní době, kdy je internet plný nejen webových stránek a prezentací, ale především rozsáhlých aplikací, je nezbytné mít pro správný chod každé aplikace přehled o činnostech, které aplikace a její uživatelé provádí, i o vznikajících chybách. Je tedy potřeba shromaždovat a analyzovat data, která nám tyto informace poskytnou. Hlavním nástrojem pro toto zaznamenávání je logování.

Cílem práce je vytvořit logovací modul, který zajistí přehled o běhu všech modulů aplikace Areus.NET 3.0 a který bude zaznamenávat nastalé události s možností spravovat a editovat záznamy včetně grafického uživatelského rozhraní. Také by měl mít možnost bezpečného napojení na stávající ticketovací systém firmy Ambro Systems spol. s r.o.

Cílem v další části práce je vytvoření monitorovacího modulu, který bude v reálném čase poskytovat přehled o stavu aplikace. Pro logovací modul je součástí práce programování databázové, prezentační i aplikační vrsty (tedy grafického rozhraní pro uživatele) a posléze implementace logování do celého systému a sepsání dokumentace.

2 Logování

Termín log je v počítačové technologii soubor či záznam (často s příponou .log), který nese informaci o činnosti a běhu aplikace. Součástí této informace je typicky čas, původce, popis, text a podobně. Tato informace pomáhá vývojářům, testerům, administrátorům a v některých připadech i uživatelům sledovat různé informace o standardních, ale i nestandardních úkonech aplikace. Stejně tak může mít i váhu důkazu v případě kriminálních aktivit nebo sporů mezi provozovatelem a uživatelem systému.

V dnešní době by měla mít každá správně a propracovaně naprogramovaná aplikace integrován i logovací systém.

Logy obsahující informace týkající se počítačové bezpečnosti nazýváme bezpečnostní logy. Jsou to logy z antivirových programů, firewallů, systémů IDS/IPS atd.

Speciální typ logu, který například eviduje změny oprávnění nebo vlastníky souborů v operačním systému, se nazývá audit; tohoto druhu logu využívá např. Microsoft Windows. S pomocí tohoto typu logu jsme v případě potřeby, např. při řešení bezpečnostního incidentu, schopni prokázat konkrétnímu uživateli konkrétní změnu v systému.

2.1.1 Druhy logování

Existuje více způsobů, jak ukládat či zapisovat logy. Mezi základní dělení patří, zda ukládat na lokální stroj nebo na vzdálené úložiště. Logování po síti je možné s využitím jednoho ze standardních protokolů: TCP (spojový) nebo UDP (nespojový).

Log je pak možné ukládat do textového souboru (`.txt`), strukturovaného souboru (`.xml, .html`) nebo do databáze.

Z důvodu zabezpečení, dostupnosti, popřípadě ladění systému je vhodné logovat na samostatný stroj, který je určený pouze pro tyto účely.

V každém případě bychom měli využít maximálních možností - jak zápis do databáze pro většinu logů, tak i logování do souboru v případech, kdy databáze není dostupná. [10]

2.2 Formáty logu

2.2.1 SysLog

Standard pro protokolování zpráv v počítačových technologiích se nazývá „Syslog“. Byl vyvinut v 80. letech pro potřeby poštovního serveru "`Sendmail`". Syslog dovoluje oddělit stroj, který log zpracovává a analyzuje, od stroje, který log vygeneroval. Jedná se tedy o klient-server aplikace. Syslog deamon přijímá a program, který log vygeneroval, odesílá. Syslog server tyto informace přijme, přidá časové razítko a případně další informace a záznam uloží. Výhodou tohoto způsobu je, že jeden stroj může centralizovat informace z více zdrojů.

Syslog se rozděluje do tří částí:

- PRI
- HEADER
- MSG

PRI

Priority (priorita) - Tato část obsahuje 9 úrovní priorit – významnost události (řazeno vzestupně od nejdůležitější):

- **Emergency** – systém není použitelný
- **Alert** – je nezbytné provést opravu okamžitě
- **Critical** – kritická zpráva
- **Error** – chybová zpráva
- **Warning** – varovná zpráva
- **Notice** – funkční, ale významný stav
- **Debug** – ladící zpráva
- **Info** – informační zpráva

Dále Facility (kategorie) - je používána pro identifikaci, ze které části programu zpráva pochází; těchto kategorií je 20, uvedu jich však pouze 12, protože posledních 8 kategorií, tj. Local0 – Local7, je připraveno pro libovolné použití.

- **auth** - autentizace, zprávy týkající se přihlašování/odhlašování uživatelů
- **authpriv** - bezpečnostní/autorizační zprávy
- **daemon** - blíže neurčené zprávy systémových aplikací
- **cron** - zprávy od softwaru, který spouští automatické úlohy
- **ftp** - zprávy z přenosu souborů TCP/UDP
- **lpr** - zprávy z tiskových periferií
- **kern** - jádro aplikace
- **mail** - mailový systém
- **mark** - vyhrazeno pro "Timestamp" - časové značky ukladající se do logu
- **news** - zprávy NNTP serveru
- **syslog** - zprávy syslogu
- **user** - zprávy na uživatelské úrovni
- **uucp** - zprávy aplikací UUCP (Unix to Unix protocol)

HEADER

Tato část obsahuje dva údaje:

- **timestamp** - datum a čas, kdy byla zpráva vygenerována
- **hostname** or IP of device - název hostitele nebo IP adresa zařízení

MSG

Poslední část zprávy obsahuje obvykle dodatečné informace:

- **tag** - jméno programu nebo procesu, který vygeneroval zprávu
- **content** - obsahuje detaily zprávy

[10]

2.2.2 Binární log

Nachází své uplatnění v situacích, kdy jsme omezeni přenosovou kapacitou sítě nebo úložnou kapacitou disků. Textová data totiž mohou být velmi efektivně zakódována do binární podoby a zmenšit tak několikanásobně svou velikost. [3]

2.3 SNMP

Simple Network Management Protocol (dále jen SNMP) je standardizovaný, široce používaný protokol pro monitorování a správu zařízení přes síť. [12] Funguje na principu klient/server. Na straně monitorovaného/spravovaného zařízení běží SNMP agent ve funkci serveru, který vyřizuje požadavky přijaté od SNMP správců (klientů). Standard definuje i speciální případ, kdy sám agent iniciuje odeslání zprávy klientům. Tento typ zprávy se nazývá SNMP trap a je odeslán v případě, kdy agent detekuje výjimečnou situaci, o které by měl být správce informován, například výpadek přívodního napájení u UPS. [9]

2.4 Textové formáty

2.4.1 Common Log Format

Je standardní textový formát, který je využívaný webovými, ftp a dalšími servery pro uchovávání přístupových záznamů (logů). Log ve formátu CLF je textový soubor, ve kterém každý řádek popisuje jeden obsloužený HTTP požadavek.

Struktura Common logfile formatu je následující:

- **remotehost** - Identifikátor stroje klienta. IP adresa - v případě, že není dostupný DNS záznam nebo je vypnutý DNS Lookup.
- **rfc931** - Identifikátor klienta.
- **authuser** - Uživatelské jméno použité při autentizaci.
- **date** - Datum a čas požadavku.
- **request** - Požadavek přesně tak, jak přišel od klienta.
- **status** - Status je numerický kod, který ukazuje, zda byl požadavek úspěšný nebo neúspěšný.
- **bytes** - Pole bajtů je numerické pole, které obsahuje počet bajtů přenesených dat jako součást HTTP požadavku, ale bez záhlaví HTTP. [14]

2.4.2 W3C Extended Log Format

- W3C Extended Log Format je výchozím formátem logu pro IIS. Jedná se opřizpůsobitelný textový formát založený na ASCII. **Version** – Verze použitého ELF standardu
- **Fields** – Seznam jednotlivých polí požadavku
- **Software** – Jméno programu, který vygeneroval log
- **Start-date** – Datum a čas počátku logu
- **End-date** – Datum a čas konce logu
- **Date** – Datum a čas přidání záznamu
- **Remark** – Poznámka

Názvy polí lze dále upřesnit pomocí prefixů. Například pole ip (IP adresa) lze upřesnit na s-ip (server IP) a c-ip (IP klienta). [10]

2.4.3 XML – eXtensible Markup Language

XML je zkratka z anglického eXtensible Markup Language - rozšiřitelný značkovací jazyk. XML je metajazyk, tedy rozšiřitelný značkovací jazyk, v jehož rámci je možné vytvářet (definovat) vlastní jazyky (DTD popis). Příkladem je XHTML, tedy kombinace XML a HTML.

XML narozdíl od známějšího jazyka HTML neobsahuje informace o způsobu zobrazení, v důsledku čehož dochází k naprostému oddělení formy od obsahu. Díky tomu je XML mnohem flexibilnější; umožňuje vlastní volbu zobrazení pro každou aplikaci, která s XML dokumentem pracuje. Pokud je přesto potřeba jednotný vzhled, lze ho definovat pomocí speciálních stylesheetů (kaskádové styly nebo XSL), které jsou připojeny v záhlaví XML dokumentu. XML se dnes používá především pro snadnou výměnu informací (např. výměnu faktur), v programování můžeme XML soubory najít použité jako konfigurační soubory. Hlavními výhodami XML oproti jiným formátům používaným pro přenos informací jsou jeho nezávislost, standardizace, podpora národních kódování a jednoduchý převod na jiné formáty. [5]

XML jako takový není formát logu, ale pro jeho výhody je často jako formát souboru pro logování používán.

2.4.4 CSV – Comma Separated Values

Jde o jednoduchý textový formát, který je nejvíce využíván pro reprezentaci tabulkových dat, případně pro přenos dat mezi různými systémy. Na řádku jsou jednotlivá pole oddělena čárkou, odtud tedy získal svůj název. Využití čárky s sebou však přináší i komplikace, protože v některých jazycích (včetně češtiny) se jako oddělovač desetinných míst používá právě desetinná čárka. V těchto případech lze použít jako oddělovač polí znak tabelátor, středník, nebo i mezeru. Ani tyto další oddělovací znaky nám však nemusí zaručit, že nenastane žádný konflikt s obsahem polí. Je potřeba uvážit obsah polí a podle toho případně na jejich ohraničení použít uvozovky. Bývá zvykem, že první řádek souboru obsahuje pojmenování jednotlivých polí nebo sloupců. I když je tento formát standardizován v RFC, ne vždy je ho možné dodržet kvůli výše zmíněným problémům. Záleží na uživateli, aby vhodně uvážil a nastavil pravidla pro import a export dokumentu tak, aby byl korektně zpracován. [9]

2.5 Generátory logů

Generátory logu jsou programy k vytváření logů. Mohou to být například síťová zařízení jako jsou switche, routery, firewally, systémy IDS/IPS, hardwarová zařízení, servery, případně softwarové aplikace. Většina zdrojů je schopna logy odesílat na logovací server nativně, například pomocí protokolu *Syslog*. Takový režim provozu se nazývá agent-less, tedy bez agenta. Pokud však zdroj z nějakého důvodu není odesílání logů schopný (buď vývojáři s touto možností nepočítali, nebo vyžadujeme specifický způsob přenosu logů), přenos logů zajišťuje agent. Jedná se o aplikaci, která buď běží přímo na zdroji a aktivně přeposílá logy, nebo může běžet jinde a aktivním dotazováním (polling) získává ze zdroje požadované logy. Protože agenti běží přímo na hostitelských strojích, jsou platformově závislí a v případě heterogenního prostředí mohou nastat komplikace s dostupností agentů pro danou platformu. Další nevýhodou je to, že při svém běhu zatěžují hostitelský systém. [9]

2.6 Analyzátory logů

Jsou nejdůležitější a nejpřínosnější částí logovací infrastruktury. Můžeme se setkat s analyzátory v podobě jednoúčelových programů, specializovaných

samostatných zařízení (tzv. appliance), nebo i složitějších komplexních programů. Analyzátory zkoumají a prohledávají shromážděná data. Analýza může probíhat ve skutečném čase na aktuálně vygenerovaných a právě přijímaných a zpracovávaných datech; tato je označovaná jako real-time (RT). RT analýza se hodí zejména na odhalování kritických stavů, kdy je při odhalení takového stavu spuštěn poplach (alert). Lze ji také využívat pro procesní řízení.

Opakem je tzv. off-line analýza, která probíhá na starších uložených datech. Své uplatnění zde nachází především statistická analýza. Umožňuje nám například sledovat vytížení provozovaných zařízení a služeb, sledovat a odhadovat trendy, detekovat anomálie atp. Off-line analýzu lze také využít při zpětném řešení bezpečnostních incidentů (forenzní analýza), případně při hledání kořenové příčiny (root cause). Analyzátory mohou pracovat s logy v surovém (raw) formátu, tedy ve stejném, v jakém je obdrží. Variabilita různých formátů však analyzátorům znesnadňuje vyhledávání nad takto nestrukturovanými daty, a proto některé analyzátory pracují s logy v normalizované podobě.

Proces normalizace logů zpracovává nestrukturované příchozí logy, vyhledává v nich důležité části, extrahuje je a dále pracuje pouze s těmito extrahovanými částmi. Při normalizaci přicházíme o část informací z logů. Proto je vhodné, aby měl analyzátor možnost přístupu i k původním nezměněným logům pro případné dohledání detailů.

3 .Net Framework

dotNet(.NET) je název, který zahrnuje soubor softwarových technologii. [12]

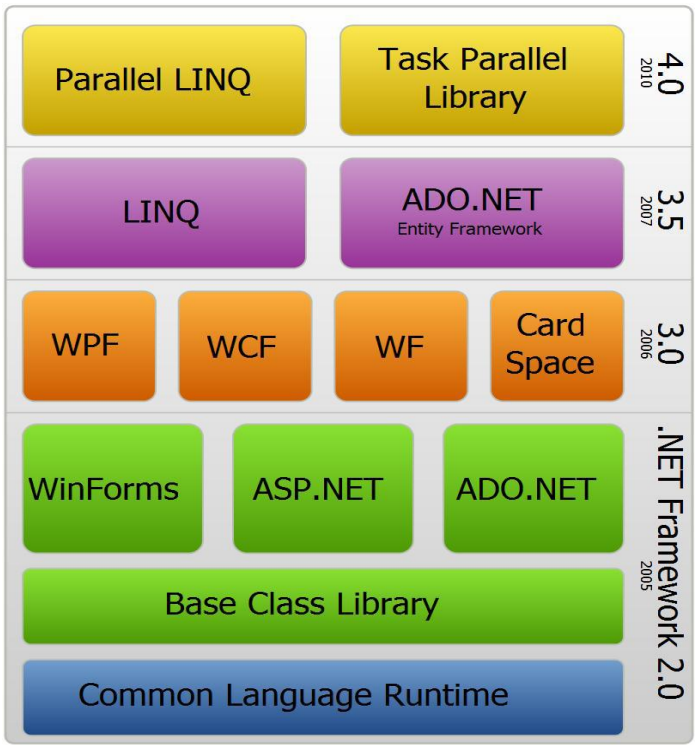
.NET Framework je platforma pro osobní počítače, nabízí prostředí pro běh aplikací, spouštěcí rohraní a potřebné knihovny.

Aplikace .NET mají stejný model návrhu, ať už jsou vyvíjeny pro přenosná zařízení "pocket PC", desktopová zařízení, serverové produkty nebo pro tvorbu webových stránek.

Jádro této platformy se nazývá .NET Framework. "Toto jádro se stará o tři základní úlohy:" [13]

- Run time prostředí pro běh aplikací. Aplikace pro něj napsané jsou s ním pevně spjaty, a nelze je spustit bez tohoto prostředí.
- Možnost tvorby ASP.NET stránek i webových služeb.
- Obsahuje knihovny objektově orientovaných tříd (bezpečnost, komunikace, práce s databázemi a datovými zdroji).

Struktura .NET framework je vidět na následujícím obrázku.



Obr. 1 Struktura .NET framework

Architektura frameworku stojí na Common Language Runtime (CLR), který tvoří nejdůležitější část frameworku.

Common Language Infrastructure (CLI)

CLI je obecná infrastruktura frameworku. Microsoft ji implementoval v CLR a CIL. CIL obsahuje definici jazyka, tzv. bytecodu, do kterého kompilátor některého vyššího jazyka (nejčastěji C#, Visual Basic .NET, Delphi nebo J#) zkompiluje kód. Tento kód je následně univerzální, a proto jej můžeme libovolně přenést i na jiné platformy. Tam je tento kód zpracován CLR, která podle dané platformy a hardwaru realtimově[4] překládá kód do nativních instrukcí procesoru pomocí metody JIT.

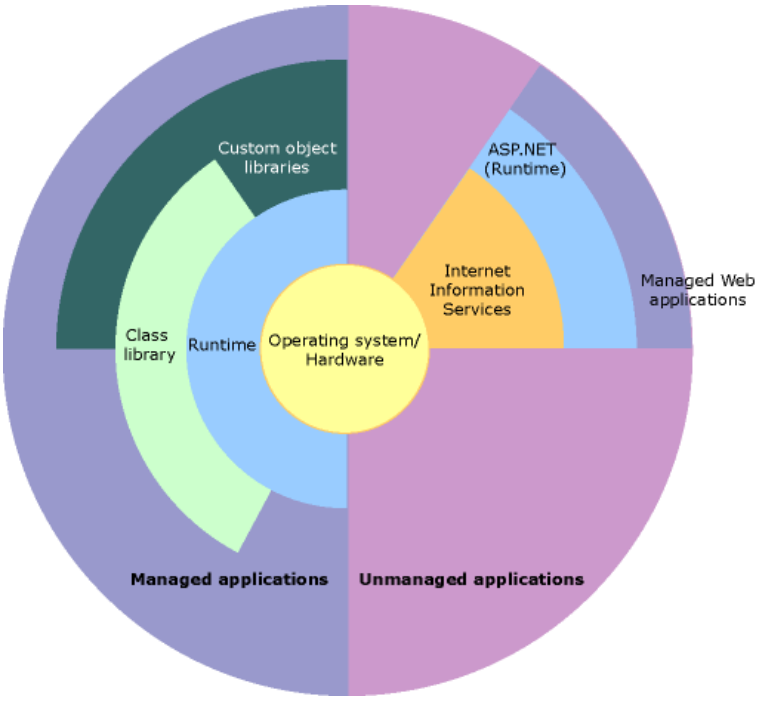
Knihovny

Framework obsahuje velké množství knihoven. Ty základní jsou souhrnně označovány jako BCL, které obsahují matematické funkce, vstupně-výstupní streamy, podporu pro kolekce, práci se sítěmi apod. Rovněž tvoří základ pro další knihovny, které jsou součástí frameworku od verze 3.0.

- **WinForms** pro tvorbu grafických API.
- **ASP.NET** pro vývoj webových aplikací.
- **ADO.NET** představující množinu tříd nabízejících služby pro přístup k datům a k tvorbě databázových aplikací.
- **Windows Communications Foundation (WCF)** je technologie pro vývoj webových služeb a komunikační infrastruktury aplikací.
- **Windows Workflow Foundation (WF)** je technologie pro definování heterogenních sekvenčních procesů.
- **Windows Presentation Foundation (WPF)** je technologie pro vytváření vizuálně působivého grafického uživatelského rozhraní pro aplikace.
- **Windows CardSpace**, implementace standardu Information Cards.
- **LINQ – Language Integrated Query** je objektový přístup k datům v databázi, XML a objektech, které implementují rozhraní Ienumerablez.

[15]

Pro vývoj .NET aplikací je k dispozici oficiání Microsoft produkt "Visual studio" v nejnovější verzi 2013, ve verzi "Express" je zdarma. Společně s tímto nástrojem tvoří .NET framework ideální prostředek pro vývoj serverových i webových aplikací.



Obr. 2 Napojení ASP.NET na knihovny .NET framework

3.1 Databáze

Řečeno velmi jednoduše, databáze je kolekce strukturovaných informací. Databáze jsou specificky navrženy pro správu velkého objemu informací, přičemž data uchovávají uspořádaným a strukturovaným způsobem, který uživatelům usnadňuje jejich správu a získávání.

Hlavní důvody, proč používat databázi:

- **Kompaktnost**: Databáze pomáhají udržovat velké množství dat, a proto zcela nahrazují objemné papírové kartotéky.
- **Rychlost**: Hledání určitého kousku dat či informací v databázi je mnohem rychlejší, než zdlouhavé probírání papírů.
- **Méně práce**: Je velice nešikovné udržovat papírové kartotéky ručně. Pokud využíváme databázi, je tato problematika zcela eliminována.

- **Životnost**: Databázové systémy lze ve většině případů velmi snadno aktualizovat, což znamená, že je můžeme využívat téměř neomezenou dobu. V databázi lze také snadno data zálohovat.

Desktopové databáze jsou navržené tak, aby sloužily omezenému počtu uživatelů a běžely na desktopových počítačích, přičemž nabízejí méně obsáhlá řešení. Mezi hlavní výhody patří nižší cena a uživatelská přívětivost. Do škály desktopových databázových řešení patří Microsoft Acces, Lotus, ale také Microsoft SQL Server ve verzi Express.

Serverové databáze jsou specificky navržené tak, aby sloužily většímu počtu uživatelů najednou, přičemž nabízejí možnosti, které umožňují velmi efektivní správu velkého množství dat současnou obsluhou požadavků od vícera uživatelů. Mezi hlavní představitele serverových databází patří: Oracle, Sybase a DB2 a databázový server Micorsoft SQL server (pozor, neplést s verzí Express).

[13]

Všechny zde uvedené databáze patří do rodiny relačních databází, existují také databáze nerelační, ale vzhledem k faktu, že Areus i logování fungují na databázi relační, nebude tento typ dále rozebírán.

3.1.1 Microsoft SQL server

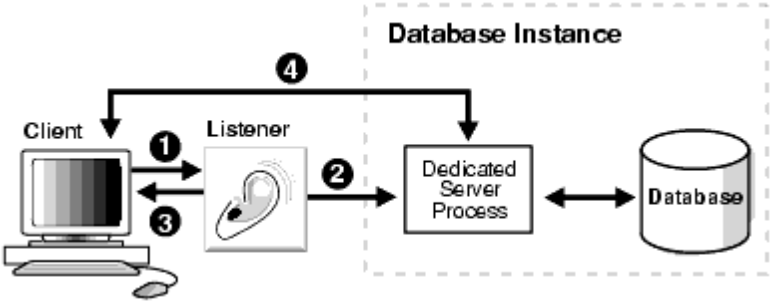
SŘBD MS SQL Server je navržen tak, aby mohl tvořit datbázovou vrstu informačních podnikových aplikací strategického významu, tedy aplikací, které vyžadují spolehlivý a robustní databázovou vrstvu. MS SQL server, lze také použít pro databázovou vrstvu u interaktivních internetových aplikací.

[16]

3.1.2 Oracle

Oracle se dá považovat za jeden z nejrozšířenejších produktů na databázovém trhu, podíl na trhu je zhruba 48%. V ČR získal Oracle na konferenci Databázový svět ocenění “Databázový produkt roku 2006”. Jednou z předností databáze Oracle je jeho multiplatformnost, tedy možnost využití v operačním systému Microsoft, ale také na systémech Unixových, např SUSElinux.

Oracle DBMS svojí architekturou pracuje na bázi komunikace klient-server. **Architekturu komunikace** znázorňuje následující obrázek.



Obr. 3 Architektura databáze Oracle

Platforma Oracle zahrnuje řadu nástrojů pro administraci a práci s daty. Standardními nástroji jsou jednak konzolové aplikace SQL*Plus, ale i grafický (GUI) správce Enterprise Manager.

[17]

3.1.3 DB2

Společnost IBM vyvíjí vlastní databázove řešení, systém nazvaný DB2. Stejně jako databáze Oracle, je multiplatformní. Obsahuje pokročilé možnosti kontroly dat, jejich využívání a ochrany. V DB2 najdeme kolekci grafických i negrafických nástrojů a komponent pro správu a vývoj aplikací. K dispozici je kompletní české prostředí během instalace i administrace.

Mezi standatní vybavení patří sada monitorovacích nástrojů, například analyzátor událostí, centrum narušení, monitor aktivity, správce neověřených stransakcí, vizualice paměti, a také asistent pro konfiguraci systému.

[17]

3.2 Vývojové prostředí

Pro vývoj aplikací v .NET frameworku lze použít několik různých nástrojů, podle funkcí můžeme najít neplacené i placené nástroje.

3.2.1 PSPad

Freewarový textový editor, který podporuje desítky typů souborů, dokáže např. zvýraznit syntaxi a obsahuje integrovaný FTP klient, záznam maker a práci s projekty.

3.2.2 SharpDevelop

SharpDevelop je open source IDE vývojové prostředí. Mezi hlavní výhody patří IntelliSence a expanze kódu, integrovaná utilita pro prohlížení objektů, integrace s dokumentací SDK.

3.2.3 Visual C# Express

Toto vývojové prostředí je velice podobné nástroji Visual Studio, nicméně je určené pro amatéry a studenty. Obsahuje designer formulářů Windows, dialogové okno Add Reference. Oproti SharpDevelop je dostupný například integrovaný grafický debugger.

3.2.4 Visual Studio

Profesionální prostředí vyvíjené firmou Microsoft, je nejkvalitnějším vývojovým prostředí pro C#, zahrnuje designer grafického prostředí, nástroje pro komunikaci a správu databází, utility pro prohlížení objektů a projektů.

Mezi hlavní přednosti patří:

- Vizuální editor a designer XML
- Podpora vývoje pro mobilní zařízení
- Podpora vývoje pro Microsoft Office
- Sledování změn a revizi daného zdrojového kódu
- Refaktoring kódu
- Knihovny pro expanzi kódu
- Visual designer

3.3 Možnosti logovaní .NET Framework

3.3.1 Apache log4net

Apache log4net je soubor knihoven a nástroj, který pomáhá programátorům zaznamenávat logy do různých výstupních cílů. Knihovny jsou tvořeny ve stejném duchu jako originální log4j, ale využívá specifických funkcí, které nabízí .NET prostředí. [7]

Hlavní výhody a funkce log4net:

- Výstup do různých logovacích uložišť.
- Nastavení a prizpůsobování XML souborů.
- Dynamická konfigurace.
- Logování kontextu.
- Modulární a rozšiřitelné provedení.

3.3.2 Syslog Sharp

Syslog Sharp je syslog server psaný v jazyce C# a využívá NET Framework. Je modulární a dovoluje tak vývojářům přidávat analyzátory a datové sklady. Součástí je i grafické uživatelské prostředí pro zobrazení zpráv v reálném čase.

Hlavní výhody a funkce Syslog Sharp:

- Webové filtry
- Zahrnuté uživatelské prostředí (GUI).
- Možnost přesměrování zpráv na jiné logovací servery.
- Rozšiřitelnost stávajících modulů.
- Možnost výběru skladovacích metod. [8]

4 Portál Areus

Portál Areus.NET 3.0 je webová aplikace postavená na .NET Frameworku 4.0, psaná v jazyce C#; společně s databazí Microsoft SQL Server tvoří Business to Business (B2B) portál.

Jako aplikační prostředí pro Areus slouží Internet Information Server verze 7 společnosti Microsoft.

Areus .NET je komunikační portál, který je primárně určen ke komunikaci mezi firmami a pro podporu obchodních zástupců. Snaží se maximálně využívat data, která si připojuje a která agreguje z ostatních systémů. Ve svém CRM může využívat i oboustranné interakce se zákazníkem, což běžné CRM systémy neumožňují. Mezi základní vlastnosti patří zejména:

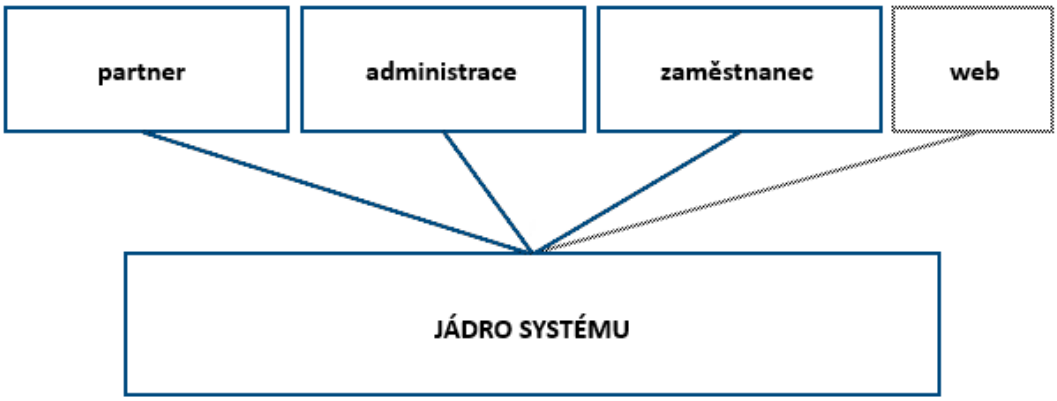
- Robustní profesionální aplikace
- Online přístup 24 hodin denně
- Postaveno na **ověřených profesionálních technologiích Microsoft**
- Široká **přizpůsobitelnost a individualizace**
- Snadná přenositelnost
- **Rychlé nasazení**, závisející především na nastavení a datových zdrojích
- Nadstandardní řízení uživatelských účtů a práv
- **Vysoká úroveň zabezpečení** systému

4.1 Architektura systému

Systém se skládá ze tří hlavních částí:

- Část partnerská podporuje komunikaci mezi firmou a obchodním partnerem.
- Část administrační slouží k celkové správě systému.
- Část zaměstnanecká slouží k podpoře obchodních zástupců.

Další volitelnou součástí je i webový modul, který umožňuje propojení webových stránek firmy s portálem AREUS. Grafické znázornění pro přehlednost v orientaci systému je následující:



Obr. 4 Architektura portálu Areus

4.2 Funkce portálu

Mezi funkce portálu patří zejména:

- **Jednoduchý objednávkový systém** s možnostmi importu objednávek ze systémů zákazníka v několika formátech – xlsx, csv, txt.
- **Přehled všech objednávek** (i telefonických, servisních atd.), dodacích listů, faktur, zásilek.
- Možnost vytvářet a kalkulovat nabídky.
- Možnost **trasování zásilek** přímo z přehledu.
- Možnost stažení účetních dokumentů (faktur, dodacích listů) ve formátu PDF.
- **Sofistikované vyhledávání a třídění dokumentů.**
- Možnost **konfigurace přístupových úrovní a uživatelských rolí.**
- Možnost **zveřejňovat události** (předváděcí akce, setkání, školení). Na tyto akce je možno se přihlašovat, systém hlídá celkovou kapacitu, je možno nastavovat přístupové úrovně, pro které je událost zveřejněna.
- Možnost **sdílet a rezervovat si nabízené prostředky** (předváděcí zařízení, servisní stroje, měřicí přístroje, atd.).
- Možnost **zveřejňovat ke stažení soubory** (letáky, návody, servisní dokumentace, katalogy).
- Možnost **zveřejňovat články** – texty.
- Možnost **vytvářet a odesílat newslettery a e-mailové obsílky.**

- Systém je **schopen pracovat až v 10 jazycích**. V současné době je k dispozici překlad do češtiny a angličtiny.

4.3 Datová konektivita

Datová konektivita s dalšími systémy (účetním, skladovým, docházkovým, CRM atd.) je řešena ve spolupráci s jejich integrátory. Na straně áuportálu Areus jsou poskytovaná data zpracovávána pomocí na míru vytvořených SQL procedur, které jsou do značné míry schopné data transformovat do požadovaného tvaru.

4.4 Systémové požadavky

- Microsoft Windows Server 2003 a vyšší.
- Microsoft .NET Framework 4.0.
- MS SQL 2008 a vyšší (v závislosti na velikosti databáze možno provozovat i na edici Express).
- 2 GB volné RAM pro webový server.
- 2 GB volné RAM pro databázový server.
- 100 MB volného prostoru na HDD + prostor pro data zákazníků. Standardně bývá zapotřebí 40-50 GB.
- Kvalifikovaný certifikát pro webový server pro použití zabezpečeného připojení (HTTPS).

5 Logovací a monitorovací modul

Požadavkem společnosti Ambro Systems spol. s r.o bylo vytvořit plnohodnotný logovací modul, který bude snadno implementovatelný, konfigurovatelný, s minimálním přístupem na souborový systém a s možností vytvoření ticketu v ticket-trackingovém systému *ZenTrack*.

5.1 Důvody pro vývoj vlastního řešení:

Hlavními důvody pro vytvoření vlastního řešení byly především snaha o maximální flexibilitu, nezávislost a množství specifických požadavků na přizpůsobení potřebám firmy a jejích zákazníků, již zaběhnutý vlastní framework a bezpečnost logovacího systému.

Výsledkem bylo rozhodnutí vytvořit vlastní systém: ekonomicky i časově bylo výhodnější napsat vlastní systém, než přizpůsobovat a integrovat již hotový systém a provádět jeho bezpečnostní audit.

Požadavky na logovací komponentu jsou následující:

- zápis logů do databáze
- zápis logů do textového souboru (v případě, že není dostupná databáze)
- možnost víceúrovňového logování
- možnost administrace logů
- práce s databází: ukládání, náhrávání a editování logu
- možnost komentovat logy a přidávat k nim vlastní poznámky
- selekce úrovní a různých vlastnotí logu z důvodu analýzy
- zasílání e-mailových zpráv administrátorovi
- vytvoření ticketu v system ZenTrack
- automatická údržba logu na souborovém systému
- upozornění sms zprávou
- monitorování modulů aplikace
- monitorování databáze
- nefunkčnost logovací komponenty nesmí ovlivnit chod celé aplikace
- možnost vypnutí jakékoliv úrovně logování

5.2 Návrh řešení

Po prozkoumání několika již vyrobených a použitelných stávajících řešení a předvedení již funkčních řešení analytikům společnosti Ambro Systems spol. s r.o. jsme se rozhodli pro vytvoření vlastního logovacího modulu.

Základním kamenem celého modulu je objekt logu, který bude mít všechny specifické vlastnosti, které budou potřeba. Dále pak bude obsahovat metody pro ukládání, nahrávání a mazání logu.

Jelikož je nezbytné logovat separovaně různé úrovně (od nejkritičtější až po informační), bylo navrženo 5 úrovní, které by měly obsáhnout veškeré požadavky. Vzhledem k charakteru systému bylo nutné vytvořit metody, které budou schopny akceptovat různorodá data v různých situacích a především v různých částech systému. Ani zdaleka ne v každé části jsou k dispozici veškeré údaje o uživateli, chybě, volaných metodách atd. Z tohoto důvodu byly veškeré metody sloužící k zápisu logu dimenzovány tak, aby v rámci své úrovně mohly obsáhnout od minima poskytnutých dat až po maximum. Bylo také nezbytné připravit takové metody, které budou schopné separovat data z objektů, které aplikace Areus již využívá.

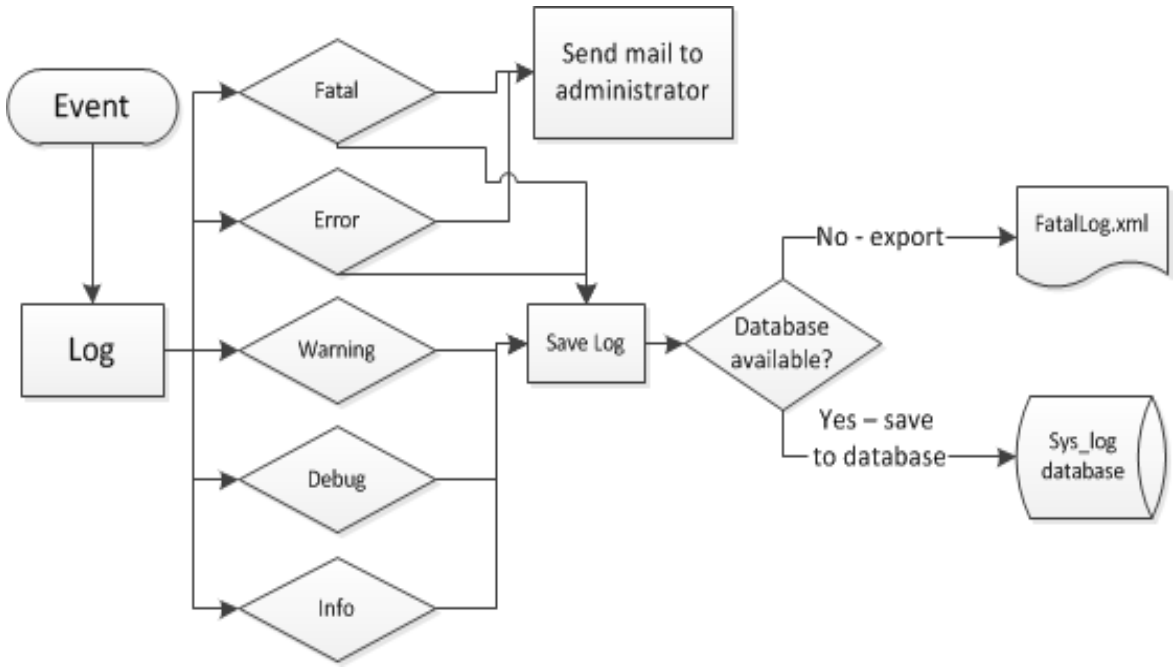
K zápisu logu na souborový systém byl zvolen formát XML. Nevýhodou tohoto formátu je především jeho velikost. Tagy, které využívá, by mohly při velkém nárůstu dat zabírat zbytečně mnoho místa na souborovém systému, nicméně vzhledem ke způsobu použití a k charakteru aplikace a také k faktu, že .NET má své nativní knihovny, je formát XML nejlepší volba.

Pro účely propojení s interním ticketovacím systémem ZenTrack byl vytvořen PHP script (integrovaný do tohoto systému), který je schopný akceptovat metodou post informace poslané z administrace Areus, automaticky vytvořit ticket a zpět vrátit jeho číslo. Jako datový formát byl zvolen Json RPC. Vzhledem k tomu, že propojení aplikace Areus a ticketovacího portálu ZenTrack je přes vnější internet, bylo nezbytné zabezpečit přístup ke komponentě pomocí API-Key tak, aby nebylo možné tickety jednoduše podvrhnout. Tento API-Key zároveň slouží k identifikaci instance Areus a k identifikaci zákazníka.

Bylo také potřeba navrhnout metodu, která dokáže automaticky provádět údržbu souborového systému tak, aby nedošlo ke zbytečnému nárustu dat a následnému zabrání cenného místa na disku.

Bohužel, striktní omezení zákaznických serverových administrátorů v některých případech nepřipouštěla jakoukoliv možnost vytvoření a spuštění jakékoliv služby v rámci operačního system serveru. Nezbývalo tedy než navrhnout řešení, které bude poskytovat aplikace jako taková. Po těchto úvahách byla navržena metoda, která při každém spuštění administrace logu nahraje veškerá data nacházející se na soubovém systému do databáze a po úspěšném ukončení celé operace logovací soubor smaže. Samozřejmě se při první inicializaci souborového logu soubor vytvoří znovu. Veškeré logy, které byly automaticky nahrány logovacím modulem, mají vlastní příznak DB_Log, který značí, že log pochází z této rodiny.

Následující obrázek zobrazuje návrh architektury logovacího modulu, tedy záznam události, výběr typu logu, dále zasílání upozornění administrátorovy a v jakých případech se používá pro záznam logu soubor či databáze.



Obr. 5 Návrh architektura logovacího modulu

Druhým požadavkem bylo navrhnout a vytvořit jednoduchý monitorovací modul. Vzhledem k tomu, že podstatnou část monitorování obstarává samotný logovací modul tím, že zasílá chybové zprávy při chybách, případně menších výpadcích systému tak by tento modul měl obstarat celý zbytek. Měla by tedy být možnost

zjištění, zda je systém on-line a zda jsou dostupné veškeré knihovny a databáze. MělMěla by mít možnost oznámit veškeré výpadky, ať již databázové nebo nekonzistentní třídy. V případně pádu aplikace musí upozornit administrátora sms zprávou.

Z důvodu firemní politky a charakterku aplikace není možné použít serverové služby, takže bylo navrženo řešení , které bude využívat externí služby serveru uptimerobot.com. Podrobnější popis této služby je v kapitole Použité technologie a nástroje - **UpTimeRobot.com**

5.3 Použité technologie a nástroje

Pro vývoj .NET bylo použito **Microsoft Visual studio 2013 ultimate.** Tato verze obsahuje veškeré funkce, které prostředí nabízí.

Hardwarové požadavky:

- 1.6 GHz nebo rychlejší procesor
- 1 GB operační paměti, 1,5GB v případě použití virtuálního stroje
- 10GB volného místa a formát souborového systému NTFS
- Rychlost disku alespoň 5400/otáček za minutu
- DirectX 9

Použitý operační systém:

- Vývoj server: Windows 7 32bit (použitý pro testování aplikace a provoz databáze)
- Vývoj stanice: Windows 7 32 bit

Databázový server:

- Microsoft SQL Server 2008 R2 SP2 – Express edice

Minimální požadavky:

- Framework
    1. .NET Framework 3.5
    2. SQL server native client
    3. SQL server setup support files

- Procesor
    1. Pentium III kompatibilní
    2. Rychlost minimum 1.0 GHz, doporučená 2.0 GHz
1. Operační pamět:Minimum 512 MB, Doporučená 2048 GB nebo více

V průběhu vývoje bylo přemigrováno na SQL Server 2012 – Express edice.

PHP

PHP je skriptovací programovací jazyk, s jehož pomocí lze vytvářet internetové stránky s údaji vloženými z databáze.

PHP stránka je html dokument s vloženými php skripty, uložený jako soubor s příponou .php. Při zavolání php stránky jsou nejprve webovým serverem vykonány příkazy php skriptů a výsledný, kompletní html kód je odeslán webovému prohlížeči.

UpTimeRobot.com

Je nástroj, který umožňuje monitorování webových aplikací.

Důležitým faktorem pro použití této služby bylo, že ve verzi zdarma můžete sledovat až 50 webů v jednu chvíli. Dotazování serveru probíhá pouze jednou za 5 minut, ale pro účely monitorování je tato četnost dostačující. Vzhledem k faktu, že portál Areus má několik instancí na různých serverech, je tato možnost ideální.

Služba monitoruje všechny servery zákazníků a interní server Ambro Systems. Načte a následně stáhne hlavičku stránky, vyhledá text v obsahu stránky a pokud narazí na chyby 4XX anebo 5XX, je web následně otestován každých 30 vteřin. Po druhém neúspěšném testu dojde k zaslání varování.

Upozornění o nefunkčním systémů může být formou e-mailové zprávy, případně sms zprávou, pro Českou republiku je dispozici pouze zasílání na mobilní telefony používající operátora O2.

Důležitým aspektem je i fakt, že tato služba administratory i vývojáře notifikuje také v případě že je celý Areus server offline.

5.4 Zachycení události a zápis logu

Událost je v rámci chápání tohoto úkolu podstatnou změnou ve stavu systémového prostředku, prostředku v síti nebo síťové aplikace. Událost může být generována pro problém, rozlišení problému nebo pro úspěšné dokončení operace. K tomu může dojít několika způsoby.

- Zachycení výjimky v klauzuli `try/catch`
- Neošetřená výjimka zachycená v `global.asax`
- Záměrně vyvolána výjimka vývojářem.
- Výjimka nebyla vyvolána, ale přesto je nutné zalogovat nastalou událost – například důležitá informace, ladící zpráva, úspěšné dokončení operace.

Tuto událost logovací modul zapisuje pomocí metod obsažených ve třídě `Log`. Metodě `Log` se předají příslušné parametry, ty jsou separovány a předány objektu log, který je poté uložen. Metody jsou odděleny podle úrovně logu:

- **Fatal** - nejkritičtější úroveň logu. V tomto stavu se předpokládá, že není dostupná databáze, a proto se zapisuje pouze na souborový systém.
- **Error** - kritická chyba. V tomto stavu je pravděpodobně dostupná databáze, ale aplikace nemůže bez vnějšího zásahu dále fungovat. Použití je jak na aplikační, tak na prezentační vrstvě. Proto byla metoda přetížena tak, aby bylo možné zapsat veškeré dostupné parametry.
- **Warning** - upozornění, například vstup uživatelských dat není v pořádku, soubor nebylo možné stáhnout a uložit, nepovedl se export dat. Pokud je zapsán log úrovně warning, aplikace sice funguje nadále dál, ale některá z funkcí není v pořádku; v nejbližší možné době se musí chyba opravit.
- **Info** - slouží pro informativní a statistické účely. K dispozici je většinou objekt uživatele. Například přihlášení/odhlášení uživatele, přehled používaných operačních systémů, webových prohlížečů používaných při návštěvě.
- **Debug** – tato úroveň slouží výhradně pro interní účely vývojářů a ladění aplikace. Většinou jsou umístěny pouze dočasně a po odladění daného problému jsou ze zdrojového kódu odstraněny.

5.5 Uložení informací

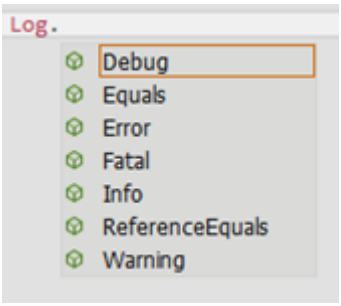
Pro snadné volání a implementaci logování byly vytvořeny 2 vlastnosti v objektu log typu `enum`. Vlastnost `SystemPart,` která obsahuje veškeré části systému, a `LogLevel` obsahující úrovně logu. Tyto enumerátory byly vytvořeny také proto, aby se předešlo chybám vyvojářů a aby bylo možno snadno logovat, ze které části systému log pochází a které úrovně log je. Zároveň bylo nezbytné přizpůsobit volání tak, aby nebylo nutné definovat celý objekt.

Proto se třídy, které slouží pouze pro zápis logu, nachází v namespace *Areus* a ostatní třídy v namespace *Areus.Kernel.Log*. Je tedy nezbytné nadefinovat počáteční using:

namespace *Areus.Kernel.Log*. Je nezbytné tedy nadefinovat počáteční using:

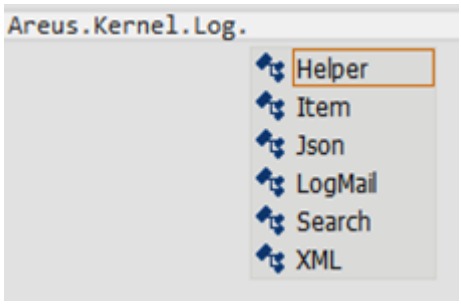
```
using Areus;
using Areus.Kernel.Log;
```

Zápis logu



Obr. 6 Struktura třídy Log

Správa logu



Obr. 7 Struktura třídy Kernel.Log

Byla implementována metoda, která v případě, že v logu je k dipozici chybová zpráva, tedy výpis ze stacku, nebo celá chyba - exception, umožnuje vyseparovat název souboru, kde chyba vznikla, název metody a také cestu řádek po řádku až ke konečnému řádku, kde byla chyba vyvolána. Toto umožnuje snadné a přehledné prohledávání logu, případně ladění chyb. Stack obsahuje mnoho zanoření, která jsou obsahem samotného frameworku; tyto hodnoty jsou přeskočeny a nejsou zapisovány do samotného výpisu. Tento výpis provádí následující kód:

```
StackTrace st = new StackTrace(true);
            StackFrame sf = st.GetFrame(1);
            int s;
            string pageName = sf.GetFileName();
            for (int i = 0; i <= st.FrameCount - 1; i++)
            {
                sf = st.GetFrame(i);
                s = sf.GetFileLineNumber();
                if (s != 0)
                {
                    pageName += "." + sf.GetMethod().Name + "
- Line number: " + s.ToString() + Environment.NewLine;
                }
            }
```

Ukázkový výsledek:

p:\AplikaceAreus\hlavniObrazovka\login.aspx.cs
Info – Line number: 504
ButtonChange_Click: Line number: 94

Veškerý zápis logu probíhá na principu předávání dostupných parametrů objektu `Item`. Tento objekt obsahuje veškeré vlastnosti, které jsou potřeba zapsat. Vzhledem k velkému počtu různých míst, kde lze logování použít, bylo nezbytné vytvořit metody dostatečně flexibilní, aby bylo možné je použít ve všech potřebných lokacích a situacích.
Metody `AddLog` slouží pro předávání parametrů samotné třídě Item obsahující objekt logu a metody pro ukládání do databáze. Na začátku volání se vytvoří prázdný objekt logu – `Item, který` obsahuje pouze defaultní hodnoty, a do toho se předají parametry. Tento objekt je následně použit pro uložení.

Primárně se dělí na logy, které obsahují či neobsahují HttpContext.Ve většině případá log vyvolaný na prezentační vrstvě HttpContext obsahuje, log z aplikační vrstvy nikoliv. Dále je možnost logovat ID uživatele a ID firmy ve které se uživatel nachází.

Volání metody `AddLog` s HttpContextem

```
AddLog(<úroveň logu>,<popis>,<název stránky/metody>,<ID
uživatele>,<ID společnosti>,<HttpContext>)
```

Volání bez HttpContextu

```
AddLog(<úroveňlogu>,<obsah logu>, <popis>, <název
stránky/metody>, <ID uživatele>, <ID firmy>)
```

5.5.1 Databáze

Jako primární uložiště záznamu se používá databáze. Logovací modul využívá samostatnou databázi, a to především z důvodu snadné správy a vytvoření nezávislosti na aplikační databázi nebo tabulkách.

Tabulky databáze

- Sys_Log - do této tabulky se zapisují logy všech úrovní, vyjma úrovně Fatal - ta se zapisuje pouze na souborový systém.
- Sys_LogLevel - tabulka Sys_LogLevel obsahuje veškeré úrovně logu, například Error, Debug.

  Sys_Notes - tabulka připravena pro případné rožšíření modulu o procházení historie a poznámek k logu.
- Sys_SystemPart - tabulka Sys_SystemPart obsahuje části systému.

Modul využívá pro některé funkce databázi Areus, je tedy nezbytné pro bezchybný běh modulu nastavit parametry v konfigurační tabulce. Proto byl vytvořen script app_config.sql, hodnoty jsou po spuštění scriptu následující:

Název	Hodnota(datový typ)	Popis
WarningStatus	True(bit)	Povolen zápis
DebugStatus	True(bit)	Povolen zápis
InfoStatus	True(bit)	Povolen zápis
FatalStatus	True(bit)	Povolen zápis
ErrorStatus	True(bit)	Povolen zápis
PathToXmlFile	<cesta k souboru>(nvarchar)	Určuje fyzickou cestu k souboru kam se má zapisovat log
companyID_HelpDesk	1(int)	ID společnosti kterým se identifikuje ticket v portále
apiKey_HelpDesk	XXXXX(nvarchar)	Heslo pro přístup k ticket.portálu
HelpDesk_Url	http://www.adresa.cz/Složka/server.php (nvarchar)	Adresa serveru

Tabulka 1 Konfigurační parametry

Tyto parametry slouží pro snadnou konfiguraci logování, pokud bude potřeba vypnout/povolit některou z úrovní logu stačí jen pomocí administrační obrazovky změnit hodnotu parametru, následující podmínka zamezí zápisu logu:

```
Private static string FatalStatus =
Areus.AppConfig.Search.GetParameterValue("FatalStatus");
if (InfoStatus == "true")
AddLog(myLevel, myPart, description, pageName, userId, companyId,
myContext);
```

Napojení na databázi obstarává samostatná třída LogDatabase.

5.5.2 XML

Sekundárním uložištěm je soubor log.xml uložený na souborovém systému. XML log se pro zápis používá pouze v případě, pokud z nějakého důvodu není dostupná databáze a je tedy nezbytné uložit log do souboru na pevném disku. Samotné volání je obsaženo pouze v klauzulích `try/catch`, aby bylo ošetřeno, že se bude používat výhradně ve stavu nefunkční databáze, byl implementován přímo do databázové třídy. Tedy v případě, že nelze uložit do databáze (databáze je nedostupná, chceme vložit hodnotu do sloupce který neexistuje, nebo vkládáme špatný typ, databázové třídy vyvolají výjimku, tu zachytíme a předáme metodě `WriteToXml` pro uložení.

```
try
{
<Kód obsluhující databázi>
}
catch (Exception e)
{
Log.XML.WriteToXml(LogItem.SystemPart.Logic_Main, e.ToString(),
"Nelze zapsat do DB");
}
```

Pro práci s tímto souborem byla vytvořena samostatná třída XML obsahující metody pro zápis a čtení, cesta k souboru je uložená v souboru *web.config.*

Metoda void WriteToXml()

Metoda zapisuje obsah objektu Log na souborový systém. Při první inicializaci soubor vytvoří, v případě, že soubor již existuje, přidá informace do stávajícího souboru.

5.6 Notifikace správce a vývojáře

Pokud je nutné v aplikaci zaznamenat log úrovně Fatal, Error nebo Warning, je obsah tohoto logu paralerně zaslán administrátorovi na e-mail. Parametry typu: z

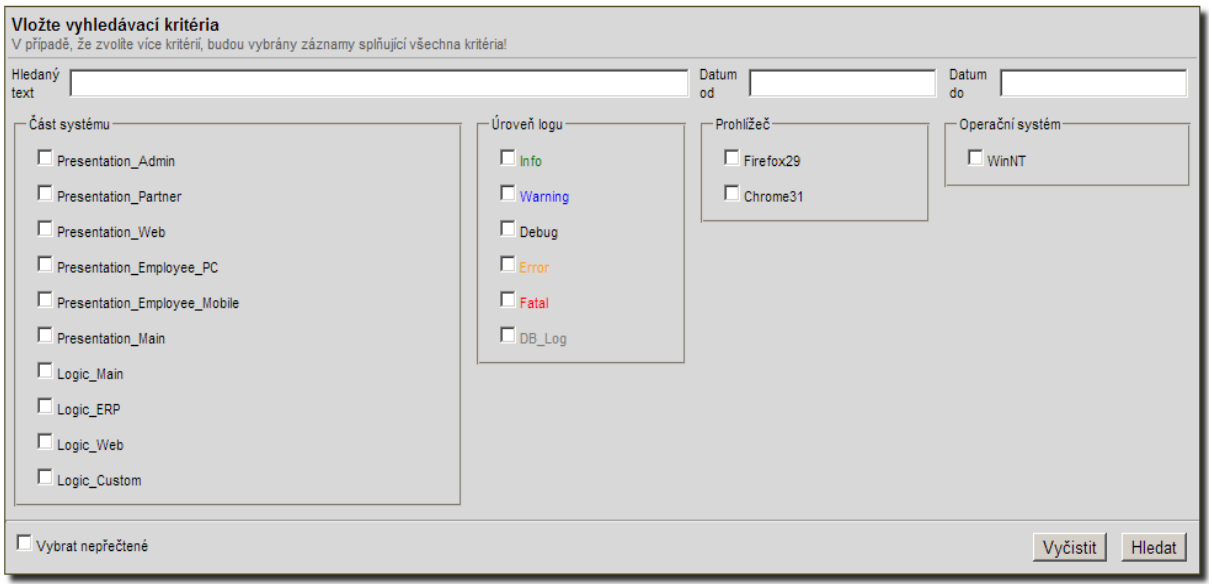
jaké e-mailové adresy, na jakou e-mailovou adresu a konfigurační údaje musí být nastaveny v konfiguračním souboru web.config.

5.7 Správa logu

Aby bylo možné s logy efektivně pracovat, byla vytvořena samostatná komponenta, která umožňuje vyhledávat, editovat, ukládat a mazat logy. Tato komponenta se skládá z několika administračních obrazovek.

Na základní obrazovce je možnost podle zadaných kriterií vyhledat veškeré logy. Jednotlivé úrovně logu jsou pro lepší orientaci barevně odlišeny. Aby bylo dosaženo maximální flexibility, byly zvoleny tyto parametry pro vyhledávání:

- Fulltextové vyhledávání - prohledá veškerý text, a pokud hledaný text log obsahuje, zobrazí se.
- Části system - vybere logy, které byly zapsány ve zvolené části systému. Může být zvoleno více částí.
- Úroveň logu - vybere logy podle zvolené úrovně. Může být vybráno více úrovní.
- Datum od – datum, od kterého chceme logy vybrat.
- Datum do - nejpozději do kterého data byly logy zaznamenány.
- Prohlížeč - jaký typ prohlížeče byl použit při záznamu.
- Operační systém – určuje, jaký typ operačního systému byl použit při záznamu. K dispozici jsou pouze prohlížeče a jejich verze, které se v logu vyskytují.
- Přečtené - každý log, který je prohlížen do detailu, dostane příznak "*přečtený*". Ve výchozím nastavení se z databáze nahrávájí logy, které jsou nepřečtené. Touto volbou vybereme všechny, to znamená "*přečtené*" i "*nepřečtené*".



Obr. 8 Obrazovka pro zadávání vyhledávacích kriterií

Výsledná data jsou zobrazena pomocí .NET komponenty DataGridView. Na této obrazovce je vidět pouze nejdůležitější výběr z vlastnotí logu. K lepšímu prohlížení byl použit atribut “ToolTip”, který je navázán na pole “Popis”; ve výsledku je pole omezeno délkou, ale je k dispozici kompletní náhled. Z tohoto přehledu je možné jednotlivě i hromadně označit logy a následně je smazat, přidat příznak “přečtený”, nebo vytvořit ticket.

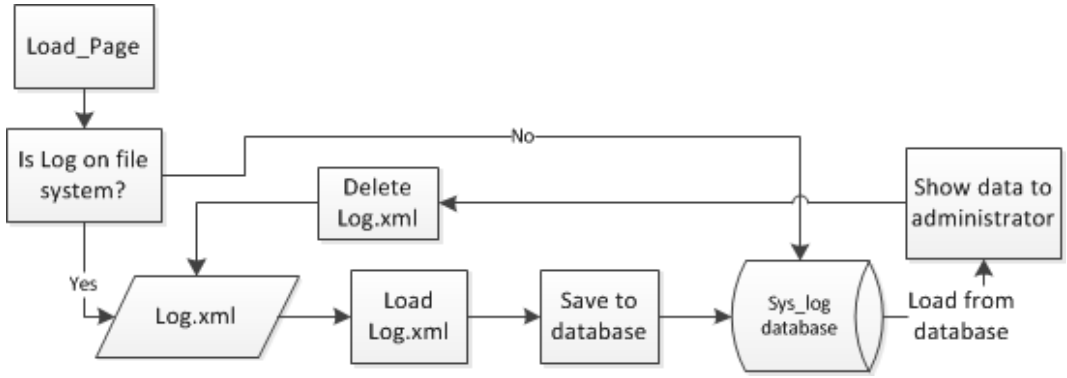
	Datum vytvoření	Prohlížeč	Operační systém	Popis	Metoda	Úroveň	Část systému	ID tick.		
✓	16.6.2014 0:45:10	Firefox29	WinNT	test	p:_Develo...	Info	Presentation_Admin	0	☐	
✓	20.6.2014 0:04:26	Firefox29	WinNT	Testovací log	p:_Develo...	Info	Presentation_Admin	0	☐	
✓	20.6.2014 0:04:34	Firefox29	WinNT	Testovací log	p:_Develo...	Info	Presentation_Admin	0	☐	
✓	20.6.2014 0:04:41	Firefox29	WinNT	Testovací log	p:_Develo...	Info	Presentation_Admin	0	☐	
✓	20.6.2014 0:04:51	Firefox29	WinNT	Testovací log	p:_Develo...	Info	Presentation_Admin	1779	☐	
✓	20.6.2014 0:04:58	Firefox29	WinNT	Testovací log	p:_Develo...	Info	Presentation_Admin	0	☐	
✓	20.6.2014 0:05:03	Firefox29	WinNT	Testovací log	p:_Develo...	Info	Presentation_Admin	0	☐	
✓	20.6.2014 0:05:32			System.Data.SqlClient.SqlException (0x80131904): L...	p:_Develo...	DB_Log	Logic_Main	0	☐	
	19.6.2014 21:21:18			System.Data.SqlClient.SqlException (0x80131904): L...	p:_Develo...	DB_Log	Logic_Main	0	☐	
	20.6.2014 0:06:33	Chrome31	WinNT		p:_Develo...	Info	Presentation_Admin	0	☐	
	20.6.2014 0:06:37	Chrome31	WinNT		p:_Develo...	Info	Presentation_Admin	0	☐	

Vybrat vše | Označit jako přečtený | Vytvořit ticket | Smazat záznam

Obr. 9 Ukázková výsledná data

Pro čtení ze souboru se využívá void metoda ReadFromXML, a to jedině v případě, kdy je stav aplikace i databáze již plně konzistentní, a administrátor otevře sekci „Správa logů“.

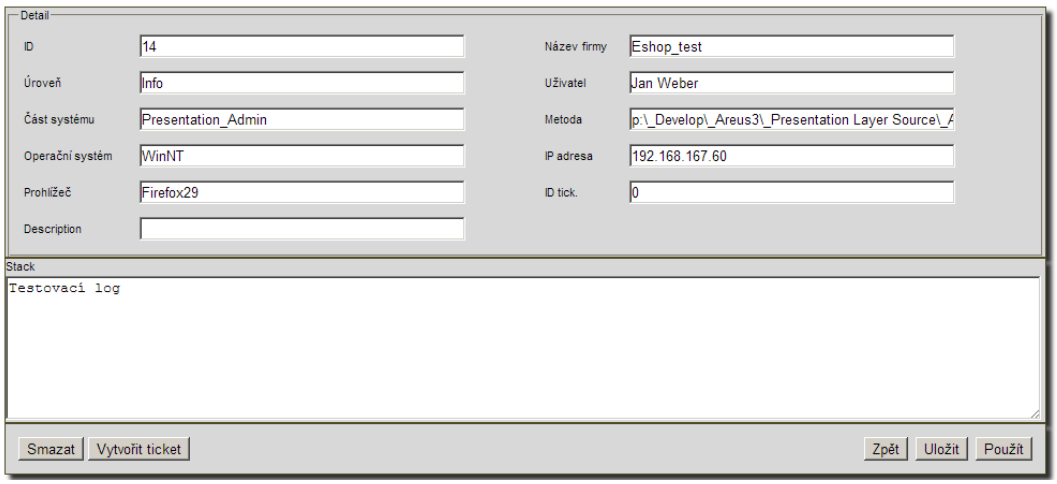
Metoda nepotřebuje žádný vstupní parametr, přečte soubor na souborovém systému, nahraje data ze souboru a v cyklu je naplní do vlastnosti objektu log a poté uloží objekt, případně objekty do databáze. V rámci údržby pevného disku a po úspěšném vykonání všech operací je následně soubor smazán. Při první inicializaci, tedy pokusu a zapsání logu, si metoda soubor vytvoří. Tímto způsobem je v podstatě automatizován systém pro přesun logu a je zajištěno, že logovací soubor nebude zabírat zbytečné místo.



Obr. 10 Proces přesunu logu z file systému do databáze

5.7.1 Editace

Logovací modul umožnuje detailní zobrazení jednotlivých logů, kde jsou vidět veškeré parametry, které log obsahuje. V této editační obrazovce je možné log upravit a následně přeuložit do databáze, případně změnit obsah “Stacku” a vytvořit ticket v ticketovacím portálu. Každý log má vlastní pole poznámek - toto bylo nezbytné k ladění určitého problému. Tyto poznámky se zaznamenávají do samostatné tabulky, sql dotazem jsou vybírany a připojeny k určitému logu. Poznámky lze měnit a uložit do databáze. Celý objekt je poté uložen do databáze.



Obr. 11 Editační obrazovka

5.7.2 Vytvoření ticketu

Jak již bylo zmíněno, logovací modul obsahuje možnost vytvoření logu v interním ticketovacím systému společnosti. Pokud je žádoucí ladit aplikaci, tato možnost usnadní managementu firmy Ambro Systems spol. s r. o., vytvářet úkoly pro vývojáře.

Vzhledem k faktu, že ticketovací portál je provozován na jiném serveru než aplikace a že aplikační prostředí je Apache Tomcat, bylo nutné nejen vytvořit rozhraní v prostředí .NET, ale i vytvořit odpovídající rozhraní v jazyce PHP a zaintegrovat jej do systému ZenTrack.

Komponenta pro vytváření ticketu je umístěna na ftp serveru. Primárně voláme index.php, kde je umístěn JSON RPC server. Tento server zpracuje parametry přijaté metodou POST a vytvoří z nich JSON objekt. Tento objekt obsahuje ID požadavku a název a parametry metody; tyto parametry obsahují asociované pole s názvy parametru a jeho hodnoty.

Pro autorizaci požadavku byla vytvořena metoda *addTicket*; tato metoda jako parametry vyžaduje následující údaje: *ApiKey* a *Company Id* pro autorizaci požadavku a pole záznamů pro vytvoření ticketů.

Systém autorizace je následující: v databázi ZenTrack je uložena tabulka obsahující klíče a identifikátor společností, které jsou unikatní pro každou

společnost, a každá instance systému Areus obsahuje tyto údaje jako konfigurační parametry.

Funkce ověřující autorizaci:

```
private function hasPermission($appKey, $companyId){
        $isAllowed = $this->database->query("SELECT
count(company_id) FROM `ZENTRACK_COMPANY` WHERE `place` =
'$appKey' AND `company_id` = '$companyId'");
        if($isAllowed->fetchColumn() == 0){
            return false;
        }
        return true;
    }
    if(!$this->hasPermission($appKey, $companyId)){
            throw new Exception('Wrong api key', 401);
```

Pokud server JSON RPC neobdrží správné údaje, vygeneruje chybu “401”, tedy neautorizovaný přístup a metoda je ukončena. Logovací modul v systému Areus tuto chybu zachytí a uloží ji v objektu log do databáze.

Pokud je autorizace úspěšná, vložené parametry jsou uloženy do databáze.

Logovací modul přečtě odpověd *httpResponse,* ta obsahuje asociované pole vložených záznamů: ID chyby (200 – není chyba, 400 – nelze zapsat ticket do databáze, 401 – neautorizovaný požadavek) jako klíč, ID záznamu v databázi jako parametr. Tím systém dostane informaci o úspěšném uložení ticketu.

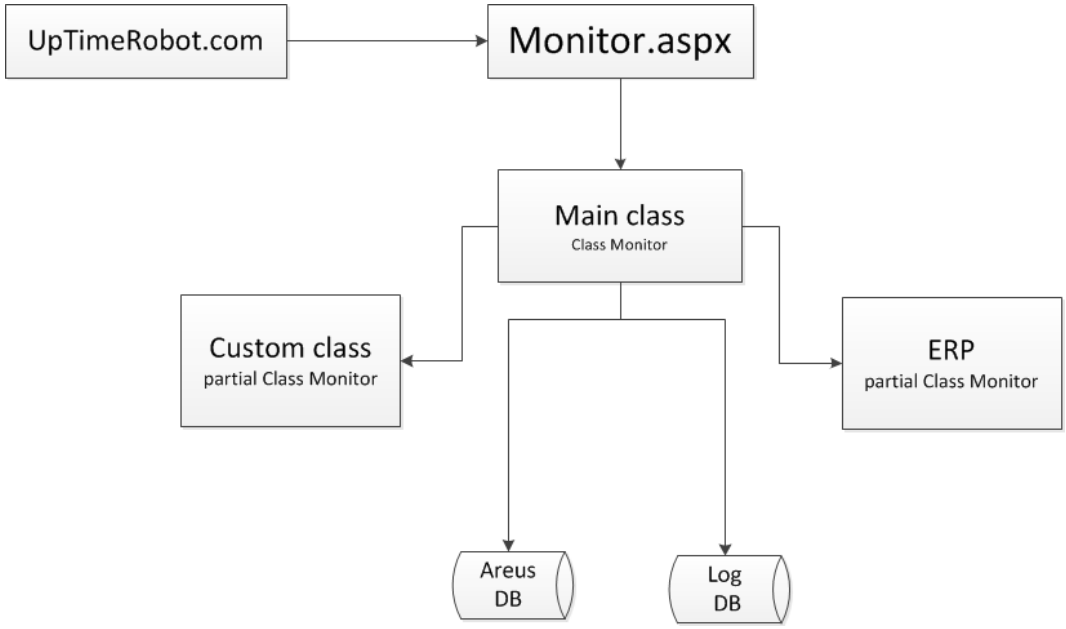
```
JSON response : ID požadavku(stejné ID jako při přijmání), Status
– číslo, bud chyba nebo data(Asociované pole).
```

5.8 Monitorovací modul

Monitorovací modul využívá již zmíněných služeb uptimerobot.com. Využívá toho, že u funkční webové stránky umožňuje její prohledání a hledání klíčových výrazů (např. ERROR, WARNING apod.) a umožňuje rovněž podle nastavených pravidel adekvátně reagovat a posílat notifikace.

Požadavkem bylo vytvořit nejen monitorovací modul, ale především prostředí pro monitorování a způsob, kterým se toto bude používat. Bylo navrženo takové, aby mohli sami vyvojáři jednoduše přizpůsobit aktuální stav instance k monitorování.

Základ monitorovacího modulu obsluhuje třída Monitor.cs, která je typu partial, a to právě z důvodu aby bylo možné vytvořit základní metody pro monitorování, ale zároveň také monitorovat specifické požadavky jednotlivých částí a samostatných knihoven aplikace Areus v rámci jednoho jmenného prostoru a jedné třídy. Jedním z těchto specifických požadavků je například kontrola databáze, zda-li některý řádek v tabulce neobsahuje nepovolené, nebo nekonzistentní údaje. Následující obrázek popisuje schéma monitorování.



Obr. 12 Schéma monitorovacího modulu

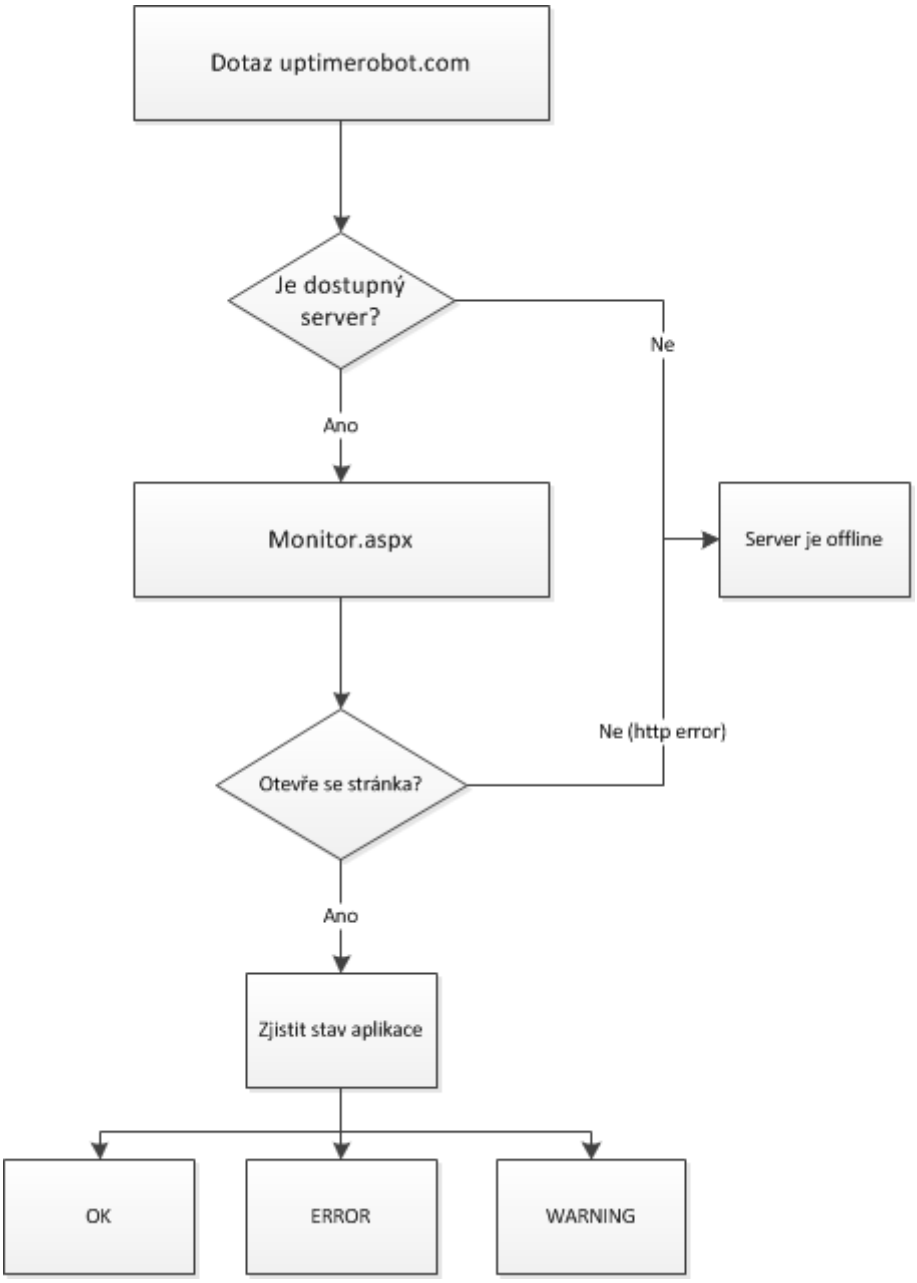
Základní předpoklad, aby aplikace byla funkční - musí běžet IIS a databáze.

Pokud je aplikace v chodu, jsou tedy splněné výše uvedné podmínky, systém uptimerobot.com vyhledává na stránce monitor.aspx zvolený řetězec. Tento řetězec je generován metodami obsaženými ve třídě monitor.cs.

Těmito řetězci jsou:

- OK - v případě, že všechny moduly jsou funkční, běžící a konzistentní.

- ERROR - znamená, že v aplikaci je závažná chyba. Systém zašle administrátorovi sms a e-mailovou zprávu o chybě.
- WARNING – v aplikaci se vyskytla chyba, ale aplikace je aktuálně funkční. Tato chyba nemusí být globální pro všechny instance Areus, ale například některým zákazníkům může aplikace padat, protože databáze obsahuje nekonzistentní data. Systém zašle administrátorovi upozornění e-mailem.



Obr. 13 Schéma monitorovacího modulu

Záleží pouze na vývojářích společnosti Ambro-Systems spol. s r.o., které metody budou implementovány, a jaké výsledky budou tyto metody vracet.

5.9 Technický popis řešení

5.9.1 Nastavení Web.config

Pro správný chod logovacího a monitorovacího modulu je nezbytné mít nastavené parametry v souboru web.config. Tento soubor slouží jako konfigurační soubor pro celou aplikaci a je standardně součástí každé .NET aplikace.

```
<configuration xmlns:xdt="http://schemas.microsoft.com/XML-
Document-Transform">
  <connectionStrings>
    <add name="MyDB"
      connectionString="ReleaseSQLServer"
      xdt:Transform="SetAttributes"
xdt:Locator="Match(name)"/>
  </connectionStrings>
</configuration>
```

Pro správny chod logovacího modulu je nutné nastavit následující parametry.

- Nastavení e-mailových informací

  <add key="SmtpServer" value="smt.server"/>

  <add key="SmtpLogin" value="smtp.server"/>

  <add key="SmtpPassword" value="smtp.server"/>

  <add key="EmailTemplateFolder" value="šablona emailu"/>

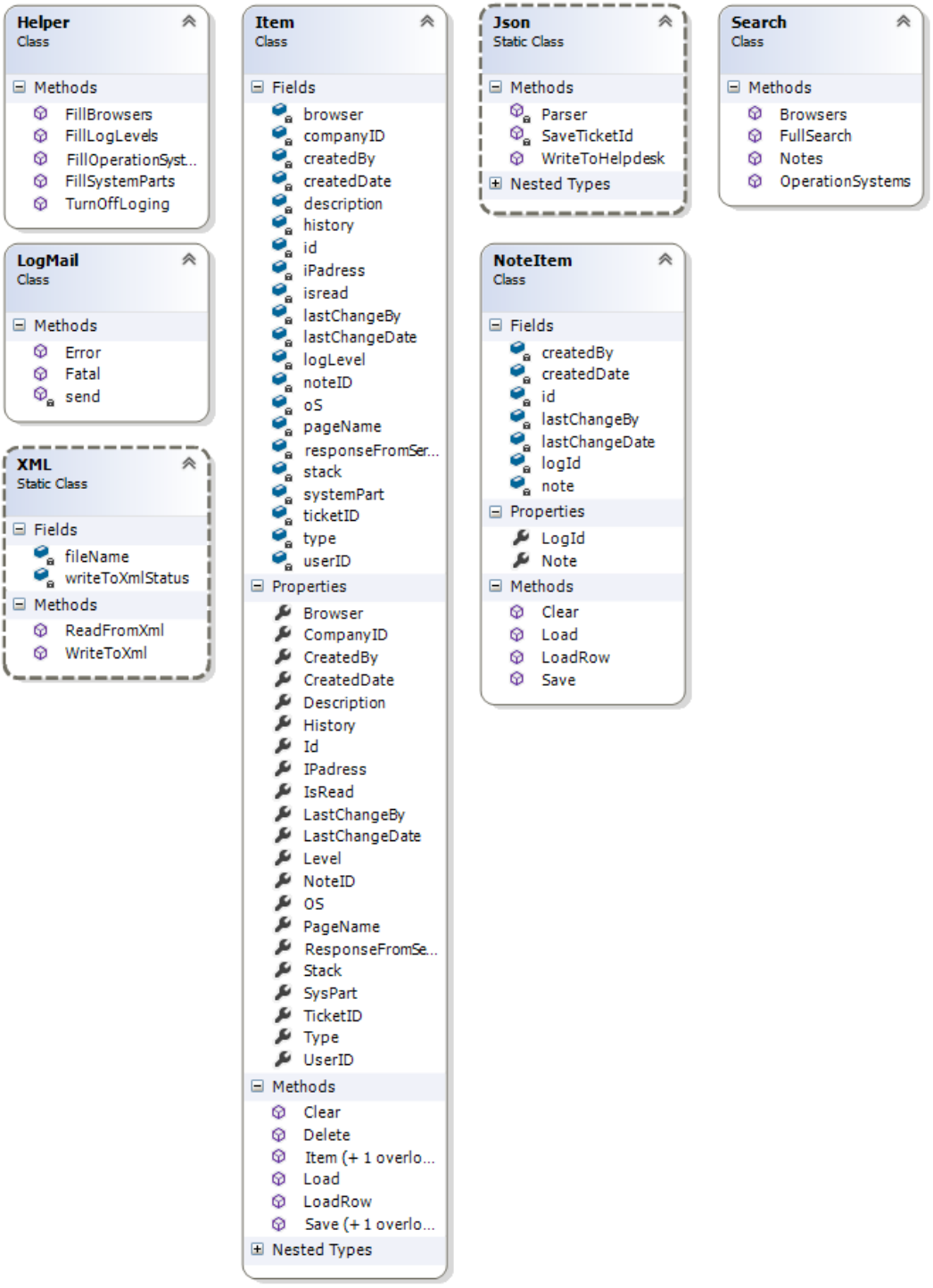
  <add key="EmailFromAdress" value="výchozí adresa"/>
- Nastavení logování – udává, zda se má zapisovat log do souboru či nikoliv.

  <add key="writeToXmlStatus" value="true/false"/>
- Nastavení cesty k logovacímu souboru.

  <add key="PathToXmlFile" value="cesta k souboru"/>
- Nastavení připojení k databázi.

```
<connectionStrings>
<add name="LogDatabaseConnStr" connectionString="Data
Source=IpAdress;Initial
Catalog=catalog;User Id=user;Password=pass;"
providerName="System.Data.SqlClient"/>
</connectionStrings>
```

5.9.2 Struktura jednotlivých tříd

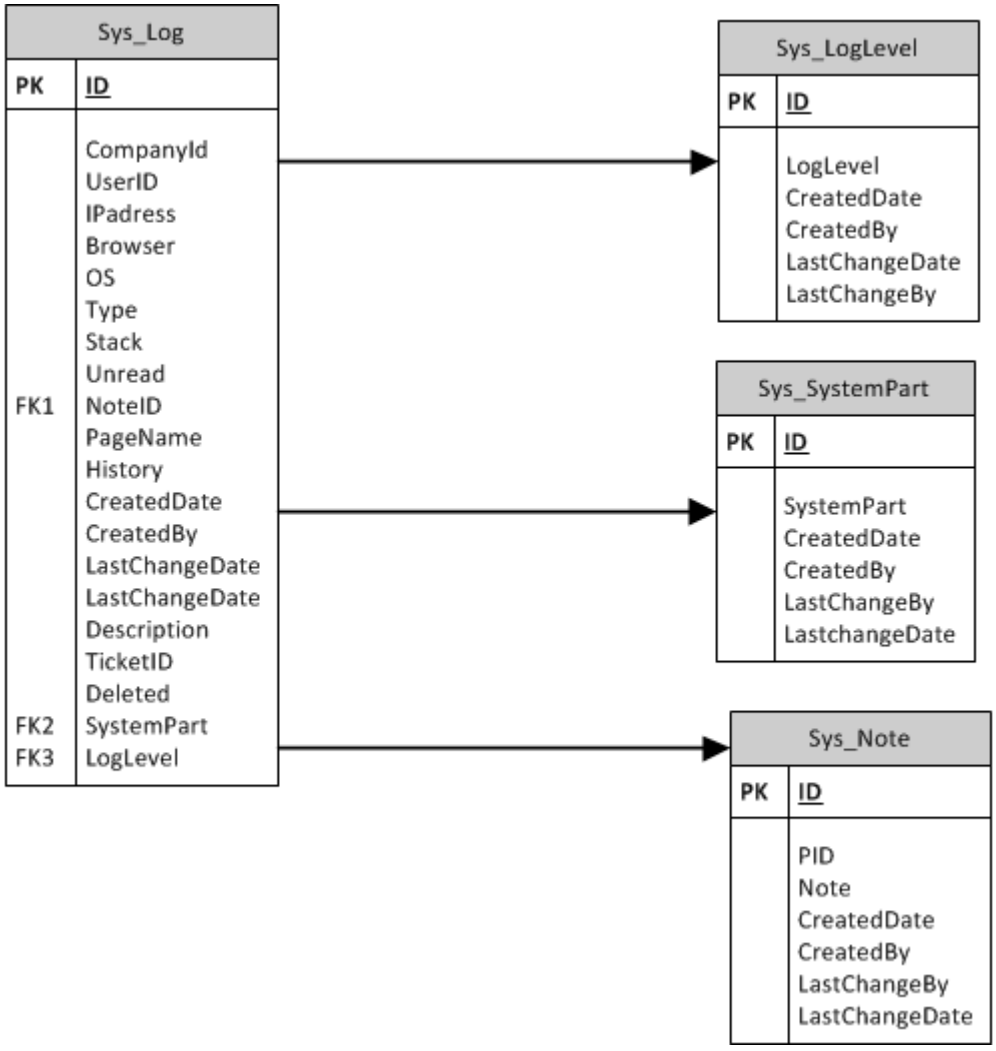


Obr. 14 Diagram tříd

5.9.3 Struktura datového souboru XML

```
<!--Log xml-->
<Log>
  <Item>
    <SystemPart>Část systému</SystemPart>
    <LogLevel>Úroveň logu</LogLevel>
    <Stack>Obsah, většinou exception</Stack>
    <PageName>Název souboru, třídy a metody kde se událost
nastala</PageName>
    <Description>Popis</Description>
    <CreatedDate>Datum vytvoření</CreatedDate>
    <CreatedBy>Kdo log vytvořil</CreatedBy>
    <LastChangeDate>Datum poslední změny</LastChangeDate>
    <LastChangeBy>Kdo naposledy log modifikoval</LastChangeBy>
  </Item>
</log>
```

5.9.4 Struktura databáze



Obr. 15 Struktura databázových tabulek a relací

6 Závěr

V bakalářské práci byly vysvětleny pojmy nezbytné k pochopení celého problému, základy pro orientaci nejenom v problematice logování, ale i v tématech přímo příbuzných. V teoretické části je stručný úvod do logování, byly uvedeny nejrozšířenější formáty logu, způsoby, kterými logy získávat a jakým způsobem je analyzovat. Práce dále přibližuje .NET framework a jeho součásti, možnosti vývojových prostředí pro .NET framework a nejrozšířenější databáze. Závěr teoretické části je věnován možnostem použití již existujících řešení logovaní ve zmiňovaném frameworku.

V praktické části práce je představen cílený portál Areus, návrh a výsledná realizace spolu s konečným řešením obou požadovaných modulů. Tato realizace zahrnuje implementaci do samotného portálu na úrovni všech vrstev. Jsou popsány jednotlivé postupy a formáty dat, které byly použity, a ukázka výsledné práce.

Předpokládá se další vývoj modulů a pokračovaní ve společné spolupráci.

Literatura

[1] VOLODARSKY, Mike. IIS 7.0: Explore The Web Server For Windows Vista And Beyond. *MSDN Magazine* [online]. 2014 [cit. 2013-12-27]. Dostupné z: http://msdn.microsoft.com/en-us/magazine/cc163453.aspx

[2] *Microsoft Internet Information Services - IIS7* [online]. 2014 [cit. 2013-12-27]. Dostupné z: http://www.iis7.cz/

[3] *MySQL* [online]. 2014 [cit. 2014-06-24]. Dostupné z: http://dev.mysql.com/

[4] DOČEKAL, Daniel. Co je to vlastně IIS?. In: *Krátké povídání o tom co je to Microsoft IIS* [online]. 2002 [cit. 2013-12-27]. Dostupné z: http://www.pooh.cz/1001/a.asp?a=2003782&db=1001

[5] XML. In: *Www.adaptic.cz: Internetový slovníček* [online]. 2014 [cit. 2014-06-24]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/xml/

[6] Log File Formats. In: *Http://www-947.ibm.com/support/entry/portal/support* [online]. 2014 [cit. 2014-01-10]. Dostupné z: http://publib.boulder.ibm.com/tividd/td/ITWSA/ITWSA_info45/en_US/HTML/guide/c-logs.html

[7] Apache log4net™ Features. *Http://logging.apache.org* [online]. 2014 [cit. 2014-06-24]. Dostupné z: http://logging.apache.org/log4net/release/features.html

[8] *Syslog Sharp* [online]. 2010 [cit. 2014-01-12]. Dostupné z: http://syslogsharp.sourceforge.net/

[9] VÍTEČEK, Antonín. *Správa a analýza bezpečnostních logů*. Brno, 2012. Diplomová práce. MASARYKOVA UNIVERZITA. Vedoucí práce RNDr. Václav Lorenc.

[10] HVĚZDA, Pavel. *Analýza a vytvoření automatického systému sledování logů*. Plzeň, 2013. Bakalářská práce. Západočeská univerzita v Plzni.

[11]*Microsoft.NET* [online]. 2014 [cit. 2014-04-21]. Dostupné z: http://www.microsoft.com/net

[12] Www.w3.org. In: *Logging Control In W3C httpd* [online]. 1995 [cit. 2014-06-20]. Dostupné z: http://www.w3.org/Daemon/User/Config/Logging.html

[13] VRAT AGARWAL, Vidya a James HUDDLESTON. *Databáze v C# 2008: Průvodce programátora*. První vydání. Brno: Computer Press, 2009. ISBN 978-80-251-2309-6.

[14] PROSISE, Jeff. *Programování v Microsoft .NET: Webové aplikace v .Net Framework, C# a ASP.NET*. Brno: Computer Press, 2013. ISBN 80-7226-879-1.
[15] ESPOSITO, Dino. *ASP.NET a ADO.NET: tvorba webových stránek*. Praha: Grada Publishing a.s., 2003. ISBN 80-247-0474-9.

[15] POKORNÝ, Jan. *Grafické nadstavby použitelné pro .NET Framework.* Plzeň, 2013. Bakalářská práce. ZÁPADOČESKÁ UNIVERZITA V PLZNI. Vedoucí práce Ing. Petr Weissar Ph.D.

[16] JAKEL, Milan. Sblížení s Microsoft SQL Server. *Www.interval.cz* [online]. 2002 [cit. 2014-06-25]. Dostupné z: http://interval.cz/clanky/sblizeni-s-microsoft-sql-server/

[17]VÍCH, Marek. *SROVNÁNÍ ROBUSTNÍCH A OPEN SOURCE DATABÁZOVÝCH SYSTÉMŮ*. Praha, 2007. Diplomová práce. Vysoká škola ekonomická. Vedoucí práce Ing. Dušan Chlapek.

Přehled použitých zkratek

IIS (Internet information server) – webový server

B2B (Business to business) – obchodní vztahy mezi společnostmi

dotNet/.NET – soubor softwarových technologií

Visual Studio – nástroj a prostřední pro práci s knihovny C#

IDS (Intrusion Detection System) - systém pro detekci průniku

IPS (Prevention System) – systém pro prevenci průniku

TCP (Transmission Control Protocol) – internetový protokol

UDP (User Datagram Protol) – internetový protokol

TXT – přípona textového souboru

XML (Extensible Markup Language) – rozšířitelný značkovací jazyk

XHTML (Extensible Markup Language) – rozšířitelný hypertextový značkovací jazyk

CSV (Comma-separated values) – souborový formát určený pro výměnu tabulkových dat

RFC (Request for comments) – označení standardů popisujících internetové protokoly

HTML (HyperText Markup Language) – značkovací jazyk

RT (Real-Time) – v této souvislosti využívaný jako pojem: reálný čas

Root - administrátor

HttpContext – třída v programovacím jazyce C#, zapouzdřuje všechny HTTP specifické informace u jednotlivých požadavků HTTP

CLI (Common Language Infrastructure) - specifikace vlastností kódu

JIT (Just in Time) – metoda překladu

BLC – soubor knihoven .NET framework

SŘBD – systém řízení báze dat

DBMS (Database management system) – systém řízení báze dat

GUI (Graphical user interface) – grafické uživatelské prostředí

FTP (File Transfer protocol) – protocol pro přenos souborů

IDE (Integrated Development Environment) – vývojové prostředí

HTTPS (Hypertext Transfer Protocol Secure) – nadstavba síťového protokolu HTTP

Seznam obrázků

Seznam tabulek