VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Analýza a implementace požadavků firmy Gates Hydraulics s.r.o. pro podpůrné aplikace

Analysis and Implementation of Requirements from Gates Hydraulics s.r.o. Company for External Applications

2014 Miroslav Zajonc

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Miroslav Zajonc**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T059 Mobilní technologie

Téma: Analýza a implementace požadavků firmy Gates Hydraulics s.r.o. pro podpůrné aplikace
Analysis and Implementation of Requirements from Gates Hydraulics s.r.o. Company for External Applications

Zásady pro vypracování:

Student se bude spolu s dalšími kolegy podílet na tvorbě IS pro firmu Gates Hydraulics, jenž má sloužit pro správu a evidenci skladu a dalších externích procesů. Celý IS je poměrně rozsáhlý, proto se student zaměří hlavně na práci s datovou vrstvou. Úkolem bude zmapovat stávající datovou vrstvu současného firemního IS, provést nezbytná rozšíření této vrstvy a připravit rozhraní pro poskytování dat prezentační vrstvě aplikace.

Zadání práce lze shrnout do následujících bodů:
1. Seznámení se s problematikou fungování procesů ve firmě.
2. Získání požadavků firmy na nový informační systém.
3. Praktická realizace dle získaných požadavků, implementace datové vrstvy postavené nad Microsoft SQL Serverem, prozkoumání možností vystavení webových služeb na tomto serveru.

IS bude postaven na platformě .NET a očekává se jeho reálné uvedení do provozu.

Seznam doporučené odborné literatury:

M. Hotek: Microsoft SQL Server 2008 Step by Step, Microsoft, 2008

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Václav Svatoň**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2014

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Student: Bc. Miroslav Zajonc

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

Zveřejněna bude pouze veřejná část práce, která bude poskytnuta dle výše uvedených požadavků. Neveřejná část práce bude k dispozici pouze oponentovi práce a komisi pro potřeby obhajoby práce. Po obhajobě bude neveřejná část práce studentovi vrácena.

Student bude s neveřejnou částí práce nakládat v souladu s prohlášením v Rámcové dohodě o mlčenlivosti

V Karviné 5. května 2014

Ing. Anton Svrček, Gates Hydraulics s.r.o.

„Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal."

V Ostravě 5. května 2014

Tajovec

.....

Na tomto místě bych rád poděkoval našemu konzultantovi Ing. Antonu Svrčkovi, za jeho skvělý přístup při analýze interních procesů a jeho nekonečnou sílu tyto procesy vysvětlovat. Dále musím poděkovat vedoucímu práce panu Ing. Václavu Svatoňovi, za jeho ochotu a za zachování chladné hlavy při řešení problémů. Poslední poděkování patří kolegovi Bc. Lubomíru Fischerovi za spolupráci při vytváření projektu.

Abstrakt

Obsah této diplomové práce zachycuje rozšíření stávajícího řešení TrackMe a doplňuje jej o další možnosti pro podporu zpracování interních procesů. Zároveň přináší do problematiky nové pohledy za současného využití aktuálních IT technologií. Rovněž nastiňuje využití webových služeb v architektuře výsledné aplikace.

Klíčová slova: informační systém, databáze, SQL Server, ERP, webové služby, SOAP

Abstract

Contents of this diploma thesis shows the expansion of the existing application TrackMe and complements it with additional options to support the processing of internal processes. Thesis brings new insights into problems while using current technology in IT. It also outlines the use of Web services in the architecture of the resulting application

Keywords: information system, databases, SQL Server, ERP, web services, SOAP

Seznam použitých zkratek a symbolů

ASP	–	Active Server Page
BPCS	–	Bussiness planing and control system
CIL	–	Common Intermetiate Language
CLR	–	Common Language Runtime
ERP	–	Enterprise resource planning
HTML	–	Hyper Text Markup Language
LINQ	–	Language Integrated Query
MVC	–	Model-View-Controller
REST	–	Representational State Transfer
RPC	–	Remote procedure call
SaaS	–	Software as a service
SOAP	–	Simple object access protocol
SSIS	–	SQL Server Integration Services
URI	–	Uniform resource identifier
URL	–	Uniform resource locator
WCF	–	Windows Communication Foudation
WSDL	–	Web service definition language
XML	–	eXtended Markup Language

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

Úvod

Tato diplomová práce vznikla kvůli potřebě firmy Gates Hydraulics s.r.o. aktualizovat část jejich stávajícího informačního systému o nové možnosti a přiblížit jej tak současným trendům na poli informačních systémů. Vedení firmy, se po konzultaci se zástupci Vysoké školy Báňské rozhodlo, rozdělit tvorbu nového informačního systému na 4 části, z nichž každá byla přiřazena jednomu studentovi.

1. Analýza a sběr požadavků
   Nejprve bylo nutné zjistit a analyzovat požadavky, které chtělo vedení zapracovat. Výstupem analýzy byl seznam požadavků, případy užití, atd.

2. Navržení a implementace datové vrstvy
   Zpracování aplikace bylo rozděleno do dvou částí. První částí bylo navržení nové datové struktury na základě požadavků a vytvoření takových opatření, aby bylo možné s daty efektivně komunikovat.

3. Vytvoření nového uživatelského rozhraní
   Druhou částí aplikace bylo vytvoření takového uživatelského rozhraní, které umožní obsluze s aplikací pracovat. V prezentační části se kladl důraz na intuitivní návrh grafického rozhraní, přizpůsobení ovládání nově vznikajícím funkcím a rozmístění ovládacích prvků aplikace. Dále bylo důležité vyčlenit, které části stránky budou viditelné za jaké situace.

4. Tvorba reportovacího subsystému.
   Poslední částí byla tvorba subsystému, který dokáže hlídat změny ve skladových zásobách součástek, jejich historii a dopad na další díly, které firma produkuje

Tato diplomová práce se zaobírá analýzou a přizpůsobením datové vrstvy a návrhu business logiky aplikace. Zároveň popisuje vytvoření sady metod pro komunikaci s prezentační vrstvou. V průběhu tvorby informačního systému TrackMe se však ukázalo, že rozdělení na prezentační a datovou část se místy těsně prolínalo, takže byla nutná úzká spolupráce obou dvou zúčastněných studentů.

V kapitole 1 jsou uvedeny technologie, jež byly v projektu využity a proběhne seznámení s nimi. Následně v kapitole 2 se věnuje prostor prozkoumání možnosti nasazení webových služeb pro distribuci dat v aplikaci. Jelikož se ve skladu Gates Hydraulics používá velký informační systém pro sledování výroby, je krátce představen popis komplexních ERP systémů v kapitole 4.

Po představení společnosti Gates v kapitole 5.1 proběhne seznámení se s procesy ve skladu. Kapitola 6 se zabývá návrhem rozšiřující části pro systém TrackMe. Popis implementace jednotlivých funkcí systému je uvedeno v kapitole 7. Budoucí vývoj a nasazení aplikace je probrán v kapitole 8.

Obsah kapitol 5, 6, 7 a 8 se z důvodu ochrany citlivých údajů, které byly při psaní těchto kapitol zmíněny, nachází v neveřejné části práce.

1 Použité technologie a nástroje

V rámci vytváření této diplomové práce bylo zapotřebí zvolit určitou platformu.Volbu platformy ulehčila skutečnost, že ve společnosti Gates již mají existující software napsaný na platformě .NET.

1.1 Platforma .NET

Za vznikem této platformy stojí americká firma Microsoft [1]. Ačkoli Microsoft zamýšlel tuto platformu používat nezávisle na systému, tak je nakonec pevně spjata s operačními systémy Windows [16].

Velmi důležitou součástí této platformy je .NET Framework [2], který poskytuje obrovskou množinu funkcí a výrazně tím zlepšuje a zrychluje tvorbu aplikací. Nabízí spoustu možností, jak ovládat či programově manipulovat s prostředím, kde je daná aplikace spuštěna. Velmi populárním programovacím jazykem nad touto platformou je jazyk C#, který svou syntaxí připomíná jazyk Java [3]. Také použití jazyka C# při tvorbě aplikací bez spolupráce s .NET Frameworkem je téměř nemyslitelné. Obě technologie jsou úzce spjaty a vytváří tak jeden funkční celek.

Tvorba aplikací pro platformu .NET není omezena jen na použití programovacího jazyka C#. Díky architektuře platformy .NET lze vyvíjet aplikace i v jiných spřízněných jazycích. Poměrně známý programovací jazyk, jež je stále využívaný a dokonce o něj vzrůstá zájem [4], je Visual Basic. V aplikaci lze použít jednu knihovnu napsanou v jazyce Visual Basic a druhou v jazyce C#. Také lze navzájem z obou knihoven volat a spouštět metody. Tato provázanost není ovšem nekonečná. Nelze psát jeden kód syntaxí obou jazyků. Vždy se musí definovat hlavní jazyk.[16]

Vzájemná provázanost je dána díky skutečnosti, že programový kód je nejprve převeden do tvz. „zprostředkujícího jazyka“ CIL a až následně převáděn do nativního jazyka pro konkrétní platformu. Nativním jazykem je už nízkoúrovňový jazyk, který obsahuje instrukce pro procesor. To, jaký kód se převede do zprostředkujícího jazyka, je ve své podstatě nedůležité. Následný překlad do strojových instrukcí je již stejný. Za tento překlad je zodpovědný běhový modul CLR.

Poměrně vítanou vlastností této platformy je automatická správa paměti. Programátor již není nucen ve svém kódu odstraňovat nepoužívané objekty, jelikož toto konání mělo za důsledek jen těžké odladění aplikace v případě chyby. V aplikaci TrackMe byl využit .NET Framework ve verzi 4.0.

1.1.1 LINQ

Verze .NET Framework 3.5 rozšířila platformu .NET o dnes již nepostradatelnou součást LINQ. Při programování je programátor nucen často pracovat s daty, která jsou různě uložena v abstraktních strukturách (polích), XML dokumentech, či databázích. Aby se podařilo načíst správná data, bylo zapotřebí psát složité konstrukty pro jejich získání. Navíc se tyto konstrukty v programu nejednou opakovaly. Poté, co se data načetla do nějaké kolekce, bylo nutné touto kolekcí iterovat při hledání konkrétního záznamu. To

má za důsledek sníženou čitelnost kódu a zhoršenou orientaci v něm, případně nejasnost instrukcí ve specifických částech programu.

Zmíněné obtíže má za cíl odstranit právě technologie LINQ. Lze říci, že tato technologie je jakýsi další programovací jazyk v již existujícím programovacím jazyce. Dala by se najít jistá analogie k jazyku SQL, s tím rozdílem, že LINQ umí prohledávat jakékoli kolekce. Pro jeho použití je nutné se s ním blíže seznámit, protože vyžaduje jiný pohled na řešení běžných situací. Pro někoho, jež nemá LINQ zažitý, může být dokonce až nečitelný. Výhodou ovšem je, že není nijak zvlášť náročné pochopit, v čem se skrývá jeho krása a elegance. LINQ se snaží ulehčit dotazování dat na nejnižší možnou úroveň. S jeho použitím se programátor může více soustředit na řešení zadaného problému.

Použití LINQ lze vidět ve výpisu 1, kdy tato část kódu má za úkol v poli měst vyhledat všechna ta města, jež začínají na „O“. K provedení této, či mnohem složitější instrukce, často postačuje jeden řádek. Bez použití LINQ by bylo k vyřešení bylo spotřebováno mnohem více instrukcí. Příkazem *where* se na pozadí spustí průchod polem *cities* a pomocí lambda výrazů [7] se docílí omezení konkrétního výběru. Finální metoda *ToArray()* převede výsledek dotazu na pole string. Alternativní zápis bližší dotazovacímu jazyku je ve výpisu 1 také uveden. Hlubší analýza výhod této technologie není účelem této kapitoly.

```
string[] cities = new string[] { "Ostrava", "Praha", "Brno", "Olomouc", "Opava" };
string[] Ocities = cities.Where(s => s.StartsWith("O")).ToArray();
Ocities = from c in cities where c.StartsWith("O")
          select c;
```

Výpis 1: Krátká ukázka použití technologie LINQ

1.1.2 Entity Framework

Aplikace TrackMe vyžadovala pro svůj běh ukládání dat v databázi. Aby se mohlo s daty pracovat i programově, bylo nutné vytvořit tzv. objektově-relační mapování (ORM) [6]. Za tímto pojmem se neskrývá nic jiného, než potřeba reprezentovat objekty z databáze (tabulky, procedury, pohledy) jako třídy, či metody. Robustní ORM je alfou a omegou každé aplikace pracující s daty. Vytváření vlastního ORM je zdlouhavý proces, který výrazně prodlužuje čas nutný pro tvorbu aplikace. Navíc by se dalo říci, že každý programátor, provádí implementaci ORM obdobným způsobem.

Proto vznikl Entity Framework [5], který byl již součástí zmiňovaného .NET Frameworku ve verzi 3.5 a byl integrován do Visual Studia 2008 [32]. Jeho úkolem je převzít kontrolu nad tvorbou ORM a na základě zvolené databáze a vybraných entit vytvořit sadu tříd pro práci s databází. Entity Framework samozřejmě využívá i technologii LINQ, takže je psaní dotazů nad databází velmi rychlé a efektivní. Použít Entity Framework v projektu ovšem neznamená, že je tento přístup naprosto bezchybný a dokonalý. Nese s sebou vlastní množinu problémů, se kterými se musí programátor potýkat. Je poměrně obtížné a zdlouhavé provádět jakékoli úpravy týkající se modelu databáze.

Přidání tabulek nebo změna sloupců, či procedur je proto riskantní operací, u které je znám výsledek až při spuštění příkazu.

1.1.3 ASP.NET MVC

Na platformě .NET bylo vždy možné vytvářet i komplexnější webové stránky ASP [17]. Pro běh těchto stránek je nutné vykonávání příkazů na straně serveru. V souladu s touto myšlenkou existují dva způsoby, jakými lze tvořit tyto aplikace.

- Web Forms
- MVC

Použití webových formulářů mělo maximálně usnadnit tvorbu webových aplikací i bez hlubší znalosti fungování jazyka HTML. Při psaní webových formulářů je možno využít mnoho připravených komponent, které slouží pro práci s daty, např. dynamické tabulky, jež se automaticky vytvoří po načtení zdroje dat. Aplikace webových formulářů se rozděluje na klient-server architekturu. Na serveru je z kódu vykompilovan HTML výstup, který je poté zasílán na klienta. [18]

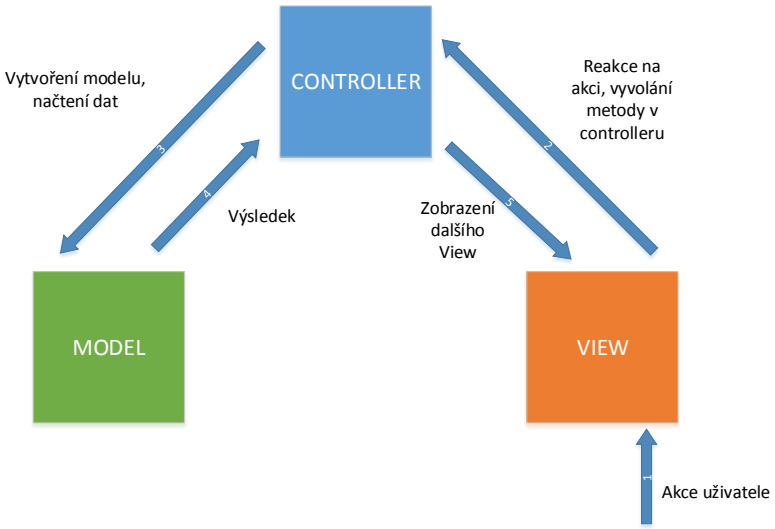
Navíc lze přistupovat k objektům poskytovaných komponent v kódu aplikace. Tímto jde bez větších obtíží kontrolovat obsah textových polí, atp. Jednou z nevýhod tohoto způsobu tvorby webových aplikací je, že vzniká velké množství automaticky vygenerovaného kódu a s růstem aplikace se v něm zhoršuje orientace. Nehledě na to, že získat úplnou kontrolu nad stránkou a tokem HTTP požadavků nebylo příliš jednoduché [17].

Zvýšit čitelnost vytvořeného kódu a zároveň nabídnout plnou kontrolu nad HTTP stránkami má ambice technologie MVC. S jejím použitím se rozděluje architektura webové aplikace na 3 základní vrstvy, z nichž každá má svůj specifický úkol. Zjednodušeně by se fungování MVC dalo popsat jak představuje obrázek 1. Controller reaguje na akci uživatelele provedenou na stránce, vytvoří odpovídající model a ten poté nechá zobrazit v konkrétní stránce. Model tedy obsahuje samotná data, ale neví, jak budou v budoucnu zobrazena. View zase nenese data, ale ví, jak je vykreslit v HTML podobě. Controller jen tento proces řídí.

1.2 Internetová Informační Služba (IIS)

Pokud by Microsoft přišel pouze s technologií ASP.NET pro tvorbu webových stránek a aplikací, pořád by webovému projektu něco scházelo. Taková aplikace musí být někde na počítači, resp. serveru spuštěna. O běh webových aplikací postavených na platformě .NET se stará velmi důležitá služba IIS.[8]

Výhodou je, že IIS není výhradně spjata pouze s hostováním webových aplikací ASP.NET, lze zde provádět rozsáhlou konfiguraci a nastavení i v rámci speciálně navržených Windows Server, které slouží jako operační systém pro serverové stroje. IIS dokáže zastat spoustu úkolů i v rámci domény. Od pokročilé správy přesměrování požadavků, přes zabezpečení, až např. po poštovní server.[14]



Obrázek 1: Schéma arhitektury MVC

1.3 SQL Server

Jestliže informační systém pracuje s daty, musí je nějakým způsobem ukládat. K uložení dat, která mají mezi sebou vztah, se používají relační databáze. Existuje několik možností, jaký typ databáze zvolit. Na začátku je vždy volba jestli zvolit open source nebo korporátní řešení.

Mezi velmi známé open source databáze patří databáze MySQL. K MySQL se může přiklonit každý, kdo hledá nějaké úložiště dat. Rozdíly mezi SQL Serverem nejsou v základu příliš dramatické[19]. Další možnou volbou je opustit vody open source databází a přiklonit se k databázi Oracle, jež poskytuje SQL nádstavbu PL/SQL [22]. Ruku v ruce s PL/SQL jde i T-SQL použitý v SQL Severu od firmy Microsoft. V tomto případě záleží, jaké technologie používá samotná firma. Každá z těchto dvou typů databází je již velmi pokročilá a záleží až na konkrétním použití, které plyne z konečného rozhodnutí zodpovědné osoby společnosti, jež chce do svého projektu začlenit databáze.

Volba, kterou databázi vybrat pro ukládání dat v aplikaci TrackMe, byla ulehčena skutečností, že ve společnosti Gates požívají MSSQL Server.

Microsoft nabízí ke svému SQL Serveru mnoho nástrojů, od základních, až po ty profesionální. Pro pohodlnou práci s daty je zde Management Studio[24] , které umožňuje psát dotazy a zobrazovat jejich výsledky v reálném čase. Dále díky němu je programátor schopen provádět i krokování jednotlivých příkazů a sledovat, kolik procent výkonu serveru vyžaduje provedení konkrétní procedury, což pomáhá v řadě případů k optimalizaci dotazů. Také SQL Server Profiler [20] je velmi užitečný nástroj, protože dokáže odposlouchávat veškerou činnost, kterou databázový stroj provádí. Po zapnutí analýzy může programátor sledovat, jaké dotazy jsou právě spouštěny.

2 Webové služby

Poměrně lákavé se zdálo zahrnout do vývoje i webové služby a učinit z nich jakéhosi prostředníka mezi daty uloženými v databázi a jejich reprezentací na straně uživatele. Ovšem po detailním zvážení začlenění webových služeb v projektu, bylo do této varianty odstoupeno, protože by systému nepřinesla žádnou výhodu navíc, ba naopak by zkomplikovala vývoj a vnesla do systému další množinu možných problémů.

Webové služby nenaleznou použití v každém případě, ale existují projekty, jež se bez jejich nasazení neobejdou. Využití naleznou v situaci, kdy je nutné distribuovat data uložená v databázi i jiným způsobem, než je pouze zobrazit na webové stránce, jak je tomu obvyklé. Aby se to mohlo uskutečnit, je zapotřebí vytvořit kolekci metod, které práci s daty dovolí. Tyto metody poté implementují tzv. klienti. Samotná implementace těchto metod v klientském prostředí už není starostí programátorů webové služby.

Hlavním cílem při návrhu webových služeb je osvobození se od závislosti na klientském prostředí a zamezení přímého přístupu klientského dotazu do databáze. To by mohlo mít katastrofální důsledky, jelikož by jeden (či více) klientů mohl být potenciální útočník, který by měl tu možnost nad daty převzít kontrolu a data modifikovat nebo v nejhorším případě je úplně odstranit. Tento útok by v podstatě nemusel být ani cílený, ale stačila by k tomu i krátkodobá nerozvážnost programátora, jež stál za vývojem příslušného klienta. Data by mohla být poškozena i špatně formulovaným dotazem. Jelikož je však tato problematika zajímavá, bude jí věnován prostor v tomto textu.

2.1 Jak fungují webové služby

Základní popis webových služeb už byl krátce zmíněný v předchozí kapitole. V době, kdy je nepřeberné množství programovacích jazyků a různých prostředí, bylo nutné stanovit určitý standard, podle kterého se dají mezi dvěma počítači, resp. mezi počítačem serveru a počítačem klienta, vyměňovat data. Dalo by se říct, že základním kamenem webových služeb je použití značkovacího jazyka XML [11], který slouží jako základní struktura pro komunikaci. Implementaci XML jazyka musí bezpodmínečně zvládnout každý vyspělejší programovací jazyk. Jazyk C# práci s XML také samozřejmě podporuje.

Každá správně naimplementována webová služba by měla splňovat dvě důležitá kritéria. Tato kritéria [15] by se dala shrnout následovně:

- *Dostupnost*. Nemělo by smysl vytvářet webovou službu, která by nebyla dostupná přes internet. Je velmi pravděpodobné, že funkce, které bude nabízet webová služba, nebudou dostupné komukoli. Proto je důležité navrhnout a správně omezit přístup nechtěným subjektům.

- *Jasné komunikační rozhraní*. Ke zveřejněné webové službě musí existovat takové rozhraní, které dovolí ostatním programátorům danou webovou službu naimplementovat. Do tohoto bodu se také zahrnuje i dobrá dokumentace, která jde ruku v ruce s webovou službou.

2.2 REST

Technologii REST definoval v roce 2002 v své dizertační práci Thomase Fiedlinga [27], čímž položil základ komunikace na internetu, jakou jí známe dodnes. Pokaždé, když uživatel internetu chce zobrazit nějakou webovou stránku, tak na pozadí probíhá vyhledávání zdrojů této stránky.

Základadní myšlenkou technologie REST je architektura s tzv. orientací na zdroje. Každý tento zdroj má na internetu daný jednoznačný název, jak jej získat. Samotné umístění zdroje nyní nehraje roli. Za odlišení identifikace prostředku je zodpovědná URL[9].

Uživatel internetu je již zvyklý, že po zadání URL adresy je mu zobrazena odpovídající stránka (např. http://www.seznam.cz). Samotnou stránku lze považovat jako zdroj. Zkrátka něco, co server vlastní a co je uživateli poskytnuto na požádání přes URL odkaz. Běžnou situací je, že za odkazem ještě následují doplňující parametry, které blíže specifikují rozsah uživatelova dotazu. Komplexnější dotaz lze spatřit ve výpise 2.

```
http : // www.ceskatelevize.cz/hledani/?q=HydePark\&cx=000499866030418304096%3
    Akbwsey4s4jw
```

Výpis 2: Struktura URL adresy

Po odeslání tohoto dotazu se v tomto případě na serveru České televize spustí funkce *hledání*, které byl předán parametr *q* s hodnotou *HydePark*. Parametr *cx* zajisté nese doplňující informace, které jsou pro tuto chvíli nepodstatné. Server zkusí podle zadaného parametru vyhledat pořad v seznamu pořadů a vrátí odpověď zpět uživateli. Tato odpověď má formát HTML [1] , tzn. že uživatel vnímá grafickou formu odpovědi. Odpověď ve formátu HTML ovšem není úplně elegantně zpracovatelná pro další operace v klientských programech. Čímž se pomalu odkrývá použití technologie REST i v kontextu webových služeb.

Pokud by server České televize odovídal v programově čitelné podobě, tedy v podobě odpovědi XML, už by se to svým způsobem dalo považovat za jakousi webovou službu, protože by program mohl na základě změny parametrů dotazu docílit jiných odpovědí. Typický výstup po zavolání akce můžeme vidět ve výpisu 3. Tento výstup už dokáže klientský program bez problému načíst a dále s ním pracovat. Za zmínku jistě stojí, že pro každou metodu musí existovat speciální url adresa, případně odlišení jménem parametru.

V komplexnějších případech se zapojují i autentizační a autorizační mechanismy, na základě kterých server dovolí příkaz spustit. Také může být v odpovědi zachycena i výjimka, která mohla při zpracování nastat např. kvůli špatným vstupním parametrům.

[1] Jako odpověď samozřejmě můžeme považovat i obrázek ve formě JPG, PNG, apod ...

```
<?xml version="1.0" encoding="UTF−8"?>
<porady>
   <porad>
      <Jmeno>Hydepark</Jmeno>
      <PocetZobrazeni>5005 </PocetZobrazeni>
      <Serie>5</Serie>
   </porad>
    ....
    ....
   <chyba stav="0">Text chyby ....  </chyba>
</porady>
```

Výpis 3: XML forma odpovědi serveru

2.3 XML-RPC

Tento způsob komunikace s webovým serverem je ve své podstatě stejný, jako je tomu v případě metody REST. Když programátor musel využít více funkcí, musel volat různě upravené URL odkazy ke spuštění těchto funkcí. U XML-RPC [28] však zůstává adresa pro všechny funkce stejná a na server se v HTTP POST požadavku posílá přímo zformátovaný XML soubor, ve kterém je, podle konkrétních specifikací dané služby, uvedena i metoda, která se má zpracováním této XML žádosti provést. Následuje opět odpověď ve formátu XML, kterou klientský program zpracovává. Možnou nevýhodou tohoto postupu je, že komunikace je synchronní [28], to znamená, že do doby, než dorazí odpověď na požadavek, klientský program stojí.

Určitou obtíží je zabezpečit při spouštění vzdálených procedur typovou bezpečnost. Mohlo by se lehce stát, že číslo napsané v elementech XML souboru by parser XML-RPC serveru považoval za řetězec, čímž by došlo k výjimce. Aby se zabránilo těmto nepříjemným komplikacím při tvorbě XML-RPC služeb, je podle specifikace [29] vyžadováno obalovat hodnoty argumentů funkcí do odpovídajícího datového typu. XML parser na straně serveru již bude vědět, jakou hodnotu a typ očekávat a tím zajistí správnou typovost.

Upravená URL z příkladu 2, by nenesla žádné doplňující parametry, ale sestávala by se teoreticky pouze z domény, resp. z takové URL, za kterou lze najít běžící XML-RPC server. Jak by vypadl požadavek na spuštění funkce hledání můžeme sledovat ve výpise 4. S dotazem i s odpovědí se ovšem přenáší velké množství dat navíc. To už je ovšem úděl jazyka XML. Bez něj by se jen velmi těžko daly v holém textu identifikovat správné části.

```
<?xml version="1.0" encoding="UTF−8"?>
<ceskaTelevize>
   <hledani>
      <parametry>
         <nazev>
            <string>HydePark</string>
         </nazev>
      </parametry>
   </hledani>
</ceskaTelevize>
```

Výpis 4: XML-RPC způsob volání funkce Hledat

2.4 SOAP

Asi nejzajímavějším a nejpokročilejším způsobem, jak vyvíjet webové služby, je použít k nim protokol SOAP [30], který částečně připomíná XML-RPC, avšak na rozdíl od něj dokáže přenášet komplexnější datové objekty, než pouze standardní hodnotové typy. A zatímco XML-RPC vyžaduje pro svůj přenos HTTP protokol, SOAP se snaží být nezávislý na transportním protokolu. Zpráva SOAP má přesně definovaný formát. Zpráva se v kontextu SOAP nazývá obálkou (envelope). Oblálka musí obsahovat hlavičku [25] a tělo zprávy. Hlavička nese doplňující informace ke zprávě. Obsahuje mimo jiné důležitý atribut mustUnderstand, který označuje, zdali informace přenášené v hlavičce zprávy jsou vyžadované, či nikoli. V hlavičce lze také nalézt případné autentizační údaje. Při implementaci SOAP webových služeb je nutno se seznámit ještě s dalšími dvěma pojmy Binding a Endpoint.

Binding

Binding se nedá úplně elegantně přeložit to ekvivalentního českého výrazu. Lze jej ovšem vysvětlit jakýmsi popisem. Binding je souhrn parametrů, které popisují samotný průběh komunikace mezi částí klienta a části serveru. Definuje typ přenášených zpráv, časová omezení pro komunikaci a to z toho důvodu, aby nedošlo k zamrznutí programu, když jedna ze stran do stanoveného času nevrátí odpověď. Stanovuje maximální limity pro délky zpráv, které se mají přenášet. Zároveň specifikuje úroveň použitého zabezpečení.

Endpoint

Naproti tomu Endpoint znamená konkrétní koncový bod komunikace. Tímto může být URL adresa, na které běží daná služba. Adresa, na kterou se budou zasílat zprávy SOAP.

2.4.1 WSDL

Velmi silná stránka protokolu SOAP je skryta za tímto akronymem. Jedná se o jazyk službu popisující. WSDL [30] je doplňující XML soubor, který jde ruku v ruce s webovou službou.V souboru WSDL se vyskytuje prakticky celý popis dané služby. Definují

se zde jednotlivé typy objektů, jejich metody a vstupní parametry metod. Vše je vždy doplněno o konkrétní typ, takže je zaručena silná typová kontrola. Moderní vývojová prostředí dokáží na základě tohoto WSDL souboru vytvořit kompletní třídní strukturu pro danou webovou službu. Což má jistě za výhodu spoustu ušetřeného času, který programátor může věnovat řešení konkrétního problému a zároveň odstíní programátory od tvorby této struktury, což minimalizuje až eliminuje riziko dalších chyb. Vyskytují se zde taktéž informace o typu bindingu a endpointu.

3 Využití webových služeb v TrackMe

V systému TrackMe, jehož popis začíná kapitolou **??**, nebyly při vývoji webové služby zahrnuty. Hlavním z důvodů je architektura stávajícího systému a jeho použití v prostorách skladu. Aplikace TrackMe běží na interním počítači, na kterém je spuštěna IIS služba, která umí zpracovávat stránky v ASP.NET. Tento počítač skrze knihovny aplikace TrackMe dále komunikuje s databázovým serverem. Když operátor spustí některou z částí aplikace, vytvoří se požadavek, který se zašle na počítač se spuštěným IIS a ten jej, na základě charakteru požadavku dále zpracuje. Jedná-li se o dotaz, který ke svému vyhodnocení potřebuje spolupráci s databází, je vytvořeno spojení s databázovým serverem a data jsou načtena a vrácena zpět přes IIS na počítač operátora. Obě dvě části, jak prezentace dat, tak i uložení dat je pod správou firmy a není vyžadováno, aby jakákoliv z těchto částí byla dostupná i zvenčí.

Tato architektura je pro takovou komunikaci zcela dostatečná. Přidáním mezičlánku v podobě webové služby by mohlo docházet k větším latencím v odpovědích, protože by se nejprve musely požadavky zabalit do SOAP zpráv na jedné straně a na straně druhé opět rozbalit.

Kdyby ovšem ke stejnému databázovému serveru museli pracovníci přistupovat jinak, než-li z intranetu, už by tento model nebyl dostačující. Další slabina této architekury by vyplynula na povrch v případě, že by byla spravovaná data o součástkách jakýmsi způsobem zajímavá i pro okolí, které by chtělo s daty dále pracovat. V tomto případě by se musely do hry zapojit právě webové služby, aby tento požadavek splnily.

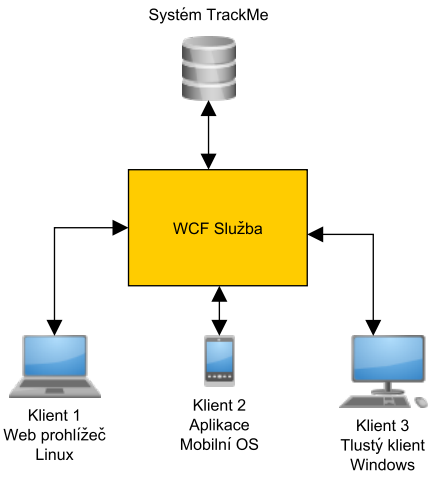
3.1 WCF

WCF [23], jehož vývoj započal v roce 2003 a poprvé se objevil v .NET Frameworku 3.5, je způsob, jakým vytvářet distribuované aplikace, jež budou používány jako webové služby. Jak už předchozí text naznačuje, v rámci distribuovaných aplikací se rozděluje architektura aplikace na klient-server. V případě WCF je zde ovšem drobná změna v tom smyslu, že místo označení server se používá označení služba. Klient může komunikovat se službou různými způsoby [10] a to protokolem SOAP (viz kap.2.4) nebo přes vlastní binární WCF protokol, či jinými způsoby. V tomto případě je důležitý prokol SOAP. Celá architektura WCF se snaží být platformně nezávislá.

Pravděpodobně nejjednodušším způsobem, jak pochopit fungování a práci s WCF, je vytvoření modelového příkladu použití. Po založení nového projektu typu *WCF Service Application* v programu Visual Studio [12] se vytvoří předpřipravená kostra způsobu vývoje aplikace, která má napovědět a pomoci s počáteční fází aplikace. Po rozkliknutí souboru s příponou „cs“ lze vidět, že rozhraní a třída, jež jsou zde uvedeny, jsou dekorovány dodatečnými atributy v hranatých závorkách. Tyto atributy [31] je potřeba blíže představit.

- **ServiceContract** - tímto atributem se opatřuje třída, či rozhraní, jež je hlavní komunikační rozhraní mezi klientem a službou.

- **OperationContract** - Metody uvnitř rozhraní opatřeného atributem ServiceContract se označují atributem OperationContract. Tímto se definuje seznam operací, které může klient spouštět.
- **DataContract** - Pokud se budou službou přenášet komplexní datové struktury (třídy, výčty, struktury), musí být před jejich deklarací uveden atribut DataContract. Atribut zajistí správnou serializovatelnost těchto atributů pro přenos mezi klientem a službou.
- **DataMember** - Aby bylo jasně zřetelné, které z třídních proměnných jsou přístupné klientovi, musí se ty z nich, jež mají být viditelné, opatřit tímto atributem. DataMember se použije až po atributu DataContract.
- **MessageContract** - V některých případech je potřeba převzít plnou kontrolu nad automatickým vytvářením SOAP zpráv, které se dějí přes atributy DataContract and OperationContract. Použití nachází především tehdy, jeli nutné rozlišit, které z elementů se budou vyskytovat v hlavičce, a které v těle SOAP zprávy.
- **FaultContract** - Vznikne-li chyba při provádění operace na straně serveru, je zamezeno, aby klient obdržel výjimku, jež chybu způsobila. Jakákoli výjimka se mění na typ FaultException. Tímto atributem se specifikuje typ výjimky, jež obdrží klient.



Obrázek 2: Architektura s použitím webových služeb

Nyní přichází na řadu ukázka použití v případě aplikace TrackMe. Pokud by ostatní společnosti potřebovaly mít přístup k informacím o součástkách, se kterými by dále ve svých klientských prostředích pracovaly, bylo by vhodné navrhnout k vyřešení této situace použití WCF služby, která by měla přístup do systémové databáze SQL Serveru. Architektura systému by se dala pozorovat v obrázku 2. Jsou zde uvedeny různé druhy

klientských aplikací, jež komunikují s jednou službou, která poté interně komunikuje s databází systému TrackMe.

3.1.1 Ukázka implementace

Jelikož WCF nebyly v projektu využity, proto zde uvedená ukázka bude pouze ilustrační a bude popisovat, jak by vypadalo začlenění webových služeb do systému TrackMe. Bude vytvořena služba, jež bude umět vypsat, kolik je v systému beden a vrátí detaily o zadaném materiálu.

Třída *TrackMeService*, jež bude implementovat *ITrackMeService* bude obsahovat dvě metody *GetBinCount* a *GetPartinfo*. První z nich vrátí odpověď v číselném formátu, zatímco ta druhá vrací komplexní odpověď typu *Part*. Třída part je atributem označená jako DataContract a nese tři proměnné *Weight*, *Length* a *Name*. K těmto proměnným se přistupuje přes vlastnosti (property), které jsou opatřeny atributy DataMember. Po spuštění této služby se automaticky spustí WCF Test Client viz obrázek 3. Tento klient umožňuje volat metody dané služby. Rovněž umí zachytit i podobu SOAP zpráv. Lze vidět, že se zavolala funkce *GetPartInfo* s parametrem part, na kterou přišla od služby odpověď typu *GetPartInfoResult*, ve které už jsou uvedeny vlastnosti hledaného dílu. V tomto případě nebyly načteny informace z databáze, ale to není žádnou překážkou. Důležité pro tento okamžik je demonstrovat základní funkčnost WCF.

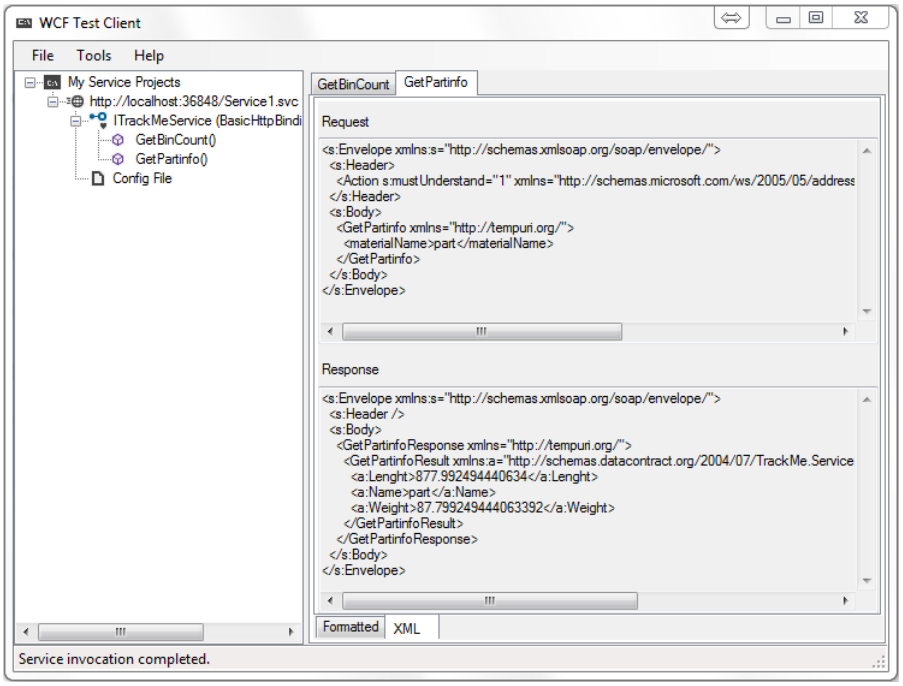
Pro implementaci v aplikaci se vytváří *TrackMeServiceClient*, přes který se volají funkce služby. Implementace na různých platformách se mohou lišit, ale v podstatě každý jazyk musí na konci odeslat a přijmout SOAP zprávu.

3.2 SSIS

V předchozí kapitole 3.1 bylo pojednáváno o možném způsobu, jak přistupovat k databázi SQL Serveru. Tento přístup je natolik obecný, že typ databáze není výhradně omezen pouze na SQL Server. WCF v tomto případě vystupoval jako jakýsi prostředník mezi různými klienty a databází. Takový návrh architektury spatřuje výhody, pokud se v aplikaci vyžaduje použití napříč různými systémy a databáze je zde jako pasivní článek celé komunikace.

Existuje ovšem další způsob, který nabízí využití integrovaných služeb SQL Serveru (SSIS)[21]. SSIS fungují odlišně, než klasické webové služby. Jejich hlavním účelem je poskytnout integrovaný nástroj, který dovolí extrakci, transformaci a načtení dat z různých zdrojů do databáze. Tento přístup ale není pro použití v TrackMe příliš vhodný, protože se jedná pouze o prostředek jakým do databáze bez zásahu dalších aplikací vložit data. Zdroje dat mohou být různé, od textových souborů s určeným formátem dat, souborů Excelu, či dalších databází a v neposlední řadě je zde možnost načítat data i z webových služeb.

Pro spouštění úloh slouží tzv. „balíčky“ (angl. package)[34], jež představují komplexnější postup, který se skládá z vícero kroků (angl. tasks). Obecně se nejprve musí inicializovat zdroj dat, poté se data požadovaným způsobem zpracují a nakonec uloží do databáze. Jednotlivé kroky jsou navzájem propojeny. Spojení znázorňuje tok dat z jednoho



Obrázek 3: Architektura s použitím webových služeb

úkolu do druhého. Výhodou těchto balíčků je, že dokáží být automaticky spouštěny nástrojem SQL Server Agent [35], jež poskytuje SQL Server. SQL Server Agent dokáže plánovat spouštění různých skriptů a dalších důležitých úloh v rámci práce s databází.

Relevantní vůči této kapitole je balíček s příznačným názvem „Web Service Task“, což znamená část podúlohy, jež je spojena se zakomponováním webové služby jako zdroje dat. Při konfiguraci této podúlohy se zadává do určených polí URL adresa služby, a také cesta k WSDL souboru. Po otestování spojení si vývojové prostředí nastaví všechno automaticky. Na výběr poté zůstane, které metody, resp. proměnné se použijí pro další postup. Všechny tyto úlohy se píší v nástroji Business Intelligence Development Studio [36].

Využití úloh spojených s webovou službou nalezne využití ve všech případech, kdy je zapotřebí přesně rozvrhnout plánování kontaktování webové služby, která nabízí metody, jež v ideálním případě nevyžadují žádné vstupní parametry. Jako příklad by posloužil systém, jež shromažďuje historické záznamy o změnách kurzu. Po nalezení patřičné webové služby, která umožňuje po zavolání metody vrátit aktuální kurz k danému času spuštění, a poté by se zjištěný kurz uložil do databáze. Díky plánování úloh v SQL Server Agentu je zajištěno pravidelné získávání aktuálních dat. Po uplynuté době se na stažených datech dají provádět různé analýzy jak se kurzy měn měnily. Existuje nepřeberné množství podobných služeb s takovým charakterem.

Spouštět balíčky lze i dalšími způsoby [37]. Lze toho docílit např. přes příkaz „dtexecui“, či přímo z uložené procedury. Použít k nim SQL Server Agenta je vzhledem k charakteru služby nejoptimálnější.

4 Podnikové informační systémy

Informační systémy obklopují život každého z nás. Vyskytují se ve všech sférách lidských činností. Bylo by velmi vzácné nalézt člověka, jež není veden v žádném informačním systému.Vstoupí-li pacient do čekárny, už si o něm lékař, či sestra nechává vypisovat dlouholetou anamnézu. U best-selleru v knihovně může knihovnice zjistit, kdo ze čtenářů tuto knihu má vypůjčenou. Zavolá-li klient na infolinku svého operátora, může si operátorka zobrazit jeho aktuální výši útraty, tarif a poslední zaplacené faktury. Také obchodní řetězce, které nabízejí svým zákazníkům zvýhodňující zákaznické karty, sledují, jaké zboží si zákazník kupuje, v jaké výši a v jakém období. Následně dokáží s těmito informacemi lépe cílit svou reklamu.

Takovými příklady by se dalo zaplnit několik stran. Důležitým poselstvím je však existence různých informačních systémů, které existují na pozadí každodenního života. Informační systémy jsou hojně využívány ve velkých podnicích pro zachycení a automatizaci podnikových procesů.

4.1 ERP

Historie informačních systémů sahá až do 60.let 20.století, kdy začaly vznikat první systémy pro řízení zdrojů v podnicích. ERP jako takové se objevily [26] až v 90.letech. ERP systém si lze představit jako továrnu. Tato továrna se skládá z několika oddělení a všechna oddělení jsou svým způsobem provázána. Pravděpodobně v té největší části továrny bude oddělení zodpovědné za výrobu. Aby byli schopni dělníci vyrábět, musí existovat další oddělení, ve kterém probíhá plánování. Některé z předmětů se musí dovést od jiných podniků, což musí sledovat oddělení zásobitelů. Vyrobené předměty se musí dostat k zákazníkovi, což je úkol pro oddělení distribuce. Určitě každý zaměstnanec této továrny bude chtít na konci měsíce obdržet výplatní pásku se svou mzdou. Zajistit správné účtování by nebylo možné bez účetního oddělení. Jednotlivé podniky mohou obsahovat další a další různě zaměřená oddělení. Pro ilustraci toto základní rozdělení postačuje.

Bylo by nesmyslné, aby každé oddělení pracovalo zcela izolovaně, tzn. s jinými daty. Takový podnik by asi dlouho nepřežil. Výše zmíněné části podniku se snaží reflektovat a vzájemně propojit v jedné centrální databázi ERP [39] systém. ERP systém tudíž není jeden malý program, ale jedná se o obrovskou množinu funkcí, která se snaží zachytit podnikové procesy.

Mezi aktuální trendy [40] v ERP systémech se řadí velmi populární využití „cloud computingu“, kdy se celá aplikace pronajímá za smluvně vázanou částku. Ačkoli je tento trend v IT čerstvý, je stále potenciálně rizikový, jelikož ke správné činnosti takto zřízených systémů je klíčové bezchybné fungování internetu a jeho rychlá odezva. I drobný výpadek internetu v řádech hodin, by mohl znamenat velké komplikace v takových systémech, ve kterých jsou procesy velmi těsně spjaty s časem. Takovým mohou být např. systémy pro logistiku. Daleko zajímavější je trend použití tzv. „in-memory computingu“, což znamená, že veškeré operace se provádí v reálném čase, protože data jsou umístěna v paměti systému. K těmto datům je díky vlastnostem pamětí velmi rychlý přístup ve srov-

nání s ostatními druhy úložišť. Tím se docílí zkrácení času systémové odezvy a umožní za kratší časový úsek provést více operací.

Největšími dodavateli [38] na poli ERP systémů jsou německá firma SAP s přibližně 25% podílem, následovaná americkým Oraclem s 13% podílem. Existují samozřejmě i stovky dalších poskytovatelů, ale tito dva jmenovaní jsou nejvýznamnějšími.

4.2 CRM

Dalším významným typem podnikových informačních systémů jsou systém CRM [41]. Hlavním myšlenkou CRM systémů je zastat kompletní správu zákazníků dané firmy, protože zákazníci jsou vždy klíčovým artiklem každé společnosti, jež je ke své činnosti potřebuje. CRM systémy spojují data zákazníků do jednoho funkčního celku. Dobrý CRM systém umožní společnosti větší vhled do způsobu chování zákazníků a potřeb. Poté lze provádět analýzy jejich potřeb a lépe zaměřit poskytování služeb s přidanou hodnotou. Celý CRM systém integruje[42] procesy, lidi a marketingové schopnosti společnosti.

Daly by se vyčlenit tři [43] možné pohledy na využití CRM systémů. V prvním z nich je CRM zaměřen na cílený marketing, kdy z analýzy požadavků zákazníků je nabízená služba, či výrobek. Druhé zaměření je v případě, kdy je nutné vést o zákaznících kompletní agendu. Dobrým příkladem pro toto zaměření je typ systémů, jež je používán např. v call centrech. Po spojení zákazníka s operátorem umožňuje systém rychlou orientaci v účtu klienta, dostupnost posledních faktur, aktuální tarif a umožní aktivaci, resp. deaktivaci další služby. Posledním zaměřením je spojení CRM systémů řídících investice s existujícími databázemi spotřebitelů.

Stejně jako v oblasti ERP, tak i v oblasti CRM lze vypozorovat obdobné trendy. Nejznámějším poskytovatelem CRM systémů je společnost SalesForce.com [44] a již zmiňovaný SAP. Zároveň lze pozorovat, že významné procento prodejů CRM systémů souvisí s modelem SaaS, kdy je software za úplatu doručován zákazníkům ve formě služby.

5 Popis současného stavu

5.1 Představení společnosti

Firma Gates Hydraulics s.r.o [46], je dceřinou společností nadnárodní korporace Gates Corporation [45], jejíž založení se datuje od roku 1911. Gates Corporation sídlí v Americkém Denveru ve státě Colorado. Mimo pobočku v České republice má spoustu dalších poboček rozesetých na každém obyvatelném kontinentu. Gates Corporation působí na celosvětovém trhu jako výrobce a dodavatel důležitých součástí pro strojní průmysl, který zásobuje např. o hnací řemeny, sestavováním hydraulických systémů, a také úzkou spoluprací s automobilovým průmyslem. Gates Hydraulics sídlí v průmyslové zóně v Novém Poli v Karviné a působí na českém trhu již od roku 2005. Zaměření moravskoslezské pobočky je především na vytváření komponent zejména pro strojní a zemědělskou techniku a jejich distribuce v rámci celé Evropy. Komponenty nacházejí své uplatnění v hydraulických a vysokotlakých systémech.

Kompletní znění této kapitoly se nachází z důvody ochrany důverných údajů v neveřejné části práce.

6 Návrh nové části systému TrackMe

Kompletní znění této kapitoly se nachází z důvody ochrany důverných údajů v neveřejné části práce.

7 Implementace případů užití

Kompletní znění této kapitoly se nachází z důvody ochrany důverných údajů v neveřejné části práce.

8 Budoucí vývoj a nasazení

Kompletní znění této kapitoly se nachází z důvody ochrany důverných údajů v neveřejné části práce.

9 Závěr

Cílem této práce bylo pomoci firmě Gates Hydraulics v implementaci rozšiřujícího modulu pro již používaný systém TrackMe. Tento modul měl doplnit systém TrackMe o funkčnost, která se zabývá podporou odesílání materiálu na externí lokace dodavatelů. V rámci návrhu aplikace se kladl velký důraz na propojení nové aplikace s existujícími databázemi, jež se ve firmě využívají k ukládání informací o materiálech.

Během procesu analýzy nám byla dána možnost vyzkoušet si práci v různorodém týmu a dbát na dobrou úroveň komunikace mezi námi a společnotí Gates. Poté na základě zjištěných požadavků provést návrh řešení, které po průběžné kontrole přestoupilo do implementační fáze. I ve fázi implementace se objevovaly další pohledy na tutéž skutečnost, takže bylo zapotřebí vždy rozhodnout, který způsob řešení je střetem zájmů obou zúčastněných stran.

Díky skutečnosti, že vývoj aplikace byl rozdělen mezi dva studenty, bylo zapotřebí sladit rychlost vývoje obou částí programu a zároveň objevit nové způsoby při vývoji aplikace v týmu dvou programátorů, z nichž každý byl zaměřen na přidělenou část.

Obsahem této práce bylo i prozkoumání začlenění webových služeb do nově vznikající aplikace. Po zvážení dopadů na architekturu systému bylo od této varianty odstoupeno. Architektura systému, jak je využíván doposud, byla dodržena i v rámci návrhu rozšiřujícího modulu.

Student: Bc. Miroslav Zajonc

Aplikace pro správu externího zpracování je jednou z nejstarších aplikací MES v naší firmě. Postupně byla rozšiřována podle měnících se požadavků na proces. Rozhodli jsme se, že ji integrujeme do námi vyvíjeného programu TrackMe, který v naší firmě vyvíjíme a kterým chceme pokrýt všechny oblasti výroby a skladování.

Jednotlivé části vývoje (sběr požadavků a analýza, aplikační design, grafický design, implementace) byly rozděleny mezi 4 studenty, kteří na projektu pracovali společně v rámci svých diplomových prací.

Studentům se podařilo kompletně zmapovat a zdokumentovat proces, pro který vytvořili use case diagramy a otestovali novou funkci nástroje Enterprise Architectu (strukturovaný zápis scénáře use case a následné generování aktivity diagramu). Podařilo se také navrhnout program pro automatizaci čištění rozdílových stavů (ve firemním slangu přehazování výhybek).

V rámci architektury TrackMe bylo navrženo rozšíření databázové struktury tak, aby byla dodržena základní vize (dohledatelnost původu materiálu) a aby bylo umožněno spravovat tyto parametry skrz webové rozhraní. Při návrhu databáze došlo ke zjednodušení struktury a odstranění duplicitních atributů. Byl modernizován grafický design s důrazem na ergonomii práce lidí ve skladu, styly sjednoceny do jedné knihovny. Pro vlastní implementaci studenti použili architekturu MVC pro ASP.NET.

Funkční verze programu, vyhovující naší představě, nám byla předvedena. Po kompletním otestování jej nasadíme do produkčního prostředí.

V Karviné 5. května 2014

Ing. Anton Svrček, Gates Hydraulics s.r.o.

10 Reference

[1] Dr. Herong Yang Introduction of .NET Framework http://www.herongyang.com/Computer-History/Dot-NET-Framework-Introduction.html

[2] Tomáš Herceg Úvod do .NET Frameworku http://www.dotnetportal.cz/clanek/125/Uvod-do-NET-Frameworku

[3] Základní charakteristika jazyka C# http://www.cs.vsb.cz/behalek/vyuka/pcsharp/text/ch02.html

[4] David Ramel Visual Basic .NET Sees Popularity Hike http://visualstudiomagazine.com/blogs/data-driver/2014/02/visual-basic-.net-popularity.aspx

[5] What is Entity Framwork? http://www.entityframeworktutorial.net/what-is-entityframework.aspx

[6] Martin Fowler Patterns of Enterprise Application Architecture 2002

[7] Augustýn Michal LINQ a lambda expressions http://www.zdrojak.cz/clanky/linq-a-lambda-expressions/

[8] Mike Volodarsky ASP.NET Integration with IIS 7 http://www.iis.net/learn/application-frameworks/building-and-running-aspnet-applications/aspnet-integration-with-iis

[9] URL http://www.computerhope.com/jargon/u/url.htm

[10] David Chappell Introducing Windows Communication Foundation, 2010

[11] Introduction to Web Services http://www.w3schools.com/Webservices/ws_intro.asp

[12] Visual Studio http://www.visualstudio.com/

[13] Obrázek dílu. http://bbhydraulics.com/oscommerce/images/picture%20072.jpg.

[14] Ken Schaefer a kol. Professional IIS 7

[15] Ethan Cerami. *Web Services Essentials Distributed Applications with XML-RPC, SOAP, UDDI and WSDL*. O'Reilly, 2002.

[16] Christian Nagel, Bill Evjen, Jay Glynn, Karli Watson, Morgan Skinner *C# 2008 Programujeme profesionálně* 2009

[17] Freeman Adam Pro ASP.NET MVC4 4th Edition

[18] Introduction to ASP.NET Web Forms http://www.asp.net/web-forms/what-is-web-forms

[19] Lorini Matteo Microsoft SQL Server vs. MySQL http://www.mssqltips.com/sqlservertip/1920/microsoft-sql-server-vs-mysql/

[20] SQL Server Profiler http://technet.microsoft.com/en-us/library/ms181091.aspx

[21] Patric LeBlanc Microsoft SQL Server 2012 Step By Step

[22] PL/SQL http://www.oracle.com/technetwork/database/features/plsql/index.html

[23] Nishith Pathak Pro WCF 4, Practical Microsoft SOA Implementation SECOND EDITION

[24] SQL Server Management Studio Express http://technet.microsoft.com/cs-cz/library/ms365247%28v=sql.100%29.aspx

[25] SOAP Header Element. http://www.w3schools.com/webservices/ws_soap_header.asp

[26] F.C. Weston Jr. F. Robert Jacobs. *Enterprise resource planning (ERP)—A brief history*. 2006.

[27] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm, 2000

[28] Edd Dumbill Simon St. Laurent, Joe Johnston. *Programming Web Services with XML-RPC*. O'Reilly, 2001.

[29] Online specifikace XML-RPC. http://xmlrpc.scripting.com/spec.html.

[30] Brian Suda. *SOAP Web Services*. http://suda.co.uk/publications/MSc/brian.suda.thesis.pdf, 2003

[31] Praveen Kumar Katiyar Understanding Contracts in WCF http://www.codeproject.com/Articles/664238/Understanding-Contracts-in-WCF

[32] Historie verzí Entity Framework http://msdn.microsoft.com/en-us/data/jj574253

[33] Mauro Marinilli *The Theory Behind User Interface Design, Part One http://www.developer.com/design/article.php/10925_1545991_2/The-Theory-Behind-User-Interface-Design-Part-One.htm*

[34] Integration Services (SSIS) Packages http://technet.microsoft.com/en-us/library/ms141134.aspx

[35] SQL Server Agent http://technet.microsoft.com/en-us/library/ms189237.aspx

[36] Integration Services in Business Intelligence Development Studio http://msdn.microsoft.com/en-us/library/ms174181%28v=sql.105%29.aspx

[37] Execution of Projects and Packages http://technet.microsoft.com/en-us/library/ms141708.aspx

[38] Louis Columbus 2013 ERP Market Share Update: SAP Solidifies Market Leadership http://www.forbes.com/sites/louiscolumbus/2013/05/12/2013-erp-market-share-update-sap-solidifies-market-leadership/

[39] Liaquat Hossain, Jon David Patrick and M.A. Rashid Enterprise Resource Planning: Global Opportunities & Challenges

[40] Petr Sodomka, Hana Klčová Aktuální trendy českého ERP trhu, Systems Online 1-2/2014

[41] Alok Mishra, Deepti Mishra Customer Relationship Management: Implementation Process Perspective , 2009

[42] Robert Fabac, Ivan Mance Customer relationship management system in occupation safety & health companies : research on practice and preliminary design solution

[43] Ben Light A review of the issues associated with customer relationship management systems, 2001

[44] Louis Columbus 2013 CRM Market Share Update: 40% Of CRM Systems Sold Are SaaS-Based http://www.forbes.com/sites/louiscolumbus/2013/04/26/2013-crm-market-share-update-40-of-crm-systems-sold-are-saas-based/

[45] Oficiální WWW Gates http://www.gates.com/

[46] Oficiální WWW Gates Hydraulics CZ http://www.gateshydraulics.cz/

[47] Ivo Vondrák Úvod do softwarového inženýrství

A Obsah přiloženého CD

CD je součástí neveřejné verze práce. Na přiloženém CD se nachází:

- Dokument analýzy - Analýza_TrackMe.pdf
- Ve složce **aplikace/program** je uveden projekt programu TrackMe
- Ve složce aplikace/databaze jsou uvedeny dva **bak** soubory, které je nutné obnovit jako databáze v SQL Serveru
- Stručný postup a vzorová data jsou uvedeny v souboru **Postup.pdf**