České vysoké učení technické v Praze
Fakulta elektrotechnická

# DIPLOMOVÁ PRÁCE

## Vzdálené volání optimalizačních nástrojů

**Praha, 2009** **Jan Ramba**

# Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne _____ _____

podpis

# Poděkování

Úvodem své práce bych rád velmi poděkoval vedoucímu své diplomové práce Ing. Janu Kelbelovi za to, že si vždy našel čas na konzultaci a že směroval mou práci k výsledku. Dále patří poděkování mé rodině, která mě podporovala při psaní této práce.

# Abstrakt

Tato práce se zabývá systémem pro zadáváním úloh ke vzdálenému zpracování. Základní funkcionalita implementovaného systému spočívá v možnosti uživatelů jednoduše formulovat úlohy a bezpečně a spolehlivě je zadávat na vzdálený server, kde se úlohy zařadí k řešení do fronty úloh v databázi. Tyto úlohy jsou postupně podle aktuální důležitosti úloh zadávány k vyřešení a po jejich vyřešení jsou výsledky předány uživateli stejnou cestou, jakou je zadal. Tento přístup umožňuje přesunout výpočetně náročné operace na vzdálený server, nezatěžovat tak klientský počítač během výpočtu a odbourává nutnost se přihlašovat na vzdálený server a zadávat ručně úlohy do cílové aplikace.

Díky použití Web Services jako vstupního interface do serverové aplikace je docíleno kompatibility s velkým počtem nástrojů, které tak lze použít jako klienty, a jednoduše generovat nové klienty.

# Abstract

This thesis deals with the remote job scheduler system. The implemented system's core functionality lies in enabling users to easily formulate a job and transport it to a remote server in a reliable and secure way. The server queues the problems to the database and submits the job to solvers based on their current importance. Jobs are returned the same way after they are finished. This solution makes it possible to move time and CPU demanding jobs to a remote server, leaves the client computer free for other work during a job execution and removes the need to login to a remote server and manually insert the job to a solver.

Using Web Services as the input interface makes the system compatible with large number of clients and makes it easy to generate new clients.

vlozit zadani!

# Obsah

**Seznam obrázků** **vi**

**1 Úvod** **1**
1.1 Motivace . . . 1
1.2 Rozbor . . . 2
1.2.1 Existující řešení . . . 2
1.2.2 Výhody navrženého řešení . . . 3

**2 Popis technologií** **6**
2.1 XML . . . 6
2.2 RPC . . . 6
2.3 SOAP . . . 8
2.4 Web Services . . . 8
2.4.1 WSDL . . . 8
2.4.2 WS-* . . . 9
2.5 .NET . . . 9
2.6 WCF . . . 10
2.6.1 Architektura WCF . . . 11
2.6.2 Kompatibilita . . . 13

**3 Popis systému** **14**
3.1 Stručný popis . . . 14
3.1.1 Architektura systému . . . 16
3.2 Podrobný popis implementace . . . 18
3.2.1 Databázový server . . . 18
3.2.2 Front-End Server . . . 23
3.2.3 Back-End server . . . 33
3.2.4 Klienti . . . 37

**4 Závěr** **40**

**Literatura** **41**

**A Obsah CD** **I**

# Seznam obrázků


2.1 Sekvence volání RPC. . . . . 7
2.2 Obecné schéma RPC. . . . . 7
2.3 Základní schéma webových služeb. . . . . 8
2.4 Struktura překladu a běhu programů v .NET frameworku. . . . . 10
2.5 Základní vrstvy architektury WCF. . . . . 11

3.1 Schéma architektury systému a základní komunikace mezi jednotlivými částmi systému. . . . . 16
3.2 Schéma databáze. . . . . 18
3.3 Detail ukončeného tasku v administračním rozhraní s neúspěšným pokusem o editaci jeho priority. . . . . 24
3.4 Detail folderu v administračním rozhraní s přehledem jeho funkcí. . . 26
3.5 Schéma mezi klientem a WS při zadávání tasku. . . . . 29
3.6 Přehled zadaných tasků ve webovém klientském rozhraní když je přihlášen administrátor. . . . . 32
3.7 Schéma komunikace mezi BE a solverem při současném zadávání dvou tasků. . . . . 36
3.8 Snímek obrazovky Picture Clientu. . . . . 38

# Kapitola 1

# Úvod

V této kapitole bude nejdříve vysvětlena motivace k vytvoření této práce a dále bude proveden rozbor řešení.

Ve druhé kapitole budou popsány použité technologie s důrazem na technologii Windows Communication Foundation, která je základem velké části systému.

Třetí kapitola bude věnována popisu systému. Nejprve bude uveden stručný popis systému a poté budou podrobněji vysvětleny její části.

Poslední kapitolou je závěr, kde jsou shrnuty výsledky práce.

## 1.1 Motivace

Pokročilá výpočetní technika umožňuje řešit stále složitější problémy, které jsou ale velmi výpočetně náročné a využívají celou výpočetní kapacitu stroje, kde jsou řešeny. Dalším problémem mohou být chybějící nástroje na řešení dané úlohy na lokálním počítači.

Řešením těchto problémů je distribuce výpočtů na vzdálený server. Uživatel na svém počítači formuluje úlohu, odešle data na server a uvolní tak uživatelský počítač pro jinou práci, případně se uživatel může odpojit od sítě nebo vypnout počítač úplně.

Cílem této práce je vytvořit systém, který bude umožňovat jednoduchou formulaci problémů a transparentní a bezpečný přenos úlohy na výpočetní server.

Systém pro řešení optimalizačních úloh může být zobecněn na řešení obecných úloh, pokud bude navržený systém dostatečně univerzální. Systém na lze dekomponovat do několika částí. Prvním dělením je přirozeně klient a server, ale server lze dále rozdělit. Funkcionalita serveru se skládá ze vstupního rozhraní (kterým budou Web Services), administračního rozhraní (webové rozhraní), podsystému pro uložení dat (transakční databáze, ke které bude přistupováno přes ODBC), rozvrhovače úloh (samostatná Web Services aplikace) a cílových programů pro řešení úloh. Tyto subsystému budou implementovány samostatně.

## 1.2 Rozbor

V této kapitole budou uvedeny existující řešení vzdáleného řešení optimalizačních, respektive obecných úloh s jejich výhodami a nevýhodami. Poté budou diskutovány výhody navrženého řešení.

### 1.2.1 Existující řešení

#### Programy pro dávkové zpracování optimalizačních úloh

Existuje jen relativně málo systémů specializovaných na vzdálené dávkové zpracování výpočetních úloh. Pravděpodobně nejznámějším a nejpoužívanějším je NEOS.

**NEOS server**

NEOS Server (Network Enabled Optimatimization System Server, [2]). Byl vyvíjen přibližně od roku 1997 skupinou programátorů z Mathematics and Computer Science division v Argonne National Laboratory. Binární soubory serveru jsou volně stažitelné, stejně jako klientské programy.

Server je volně přístupný. Server komunikuje (omezeno jedině licencí jednotlivých solverů) pomocí veřejně přístupného XML-RPC API (eXtensive Markup Language – Remote Procedure Call Aplication Programming Interface) - přímo programově, pomocí desktop klienta, přes E-mail nebo přes WWW formulář) - nebo pomocí AMPL (A Modeling Language for Mathematical Programming) a GAMS (General Algebraic Modeling systém) přes specializované klient-server aplikace (tzv. kestrely). NEOS Server umožňuje registraci vlastních solverů. K tomu je třeba stáhnout a nainstalovat vlastní instanci NEOS Serveru, nakonfigurovat NEOS pro komunikaci s uživatelským solverem a registrovat ho na centrálním NEOS Serveru (pokud je v plánu vytvořit veřejný solver). Registrace probíhá zase přes XML-RPC API. API nabízí funkce pro získání informací o NEOS serveru (nápověda, verze serveru, fronta úloh, seznam solverů), funkce pro zadávání úloh a získávání výsledků a funkce pro správu solverů.

Výhodou tohoto serveru je, že poskytuje výpočetní kapacitu a mnoho solverů zdarma bez nutnosti konfigurace, a možnost přidávání vlastních solverů. Jako nevýhody vidím uzavřenost zdrojových kódů, centralizaci přes cizí server a hlavně způsob komunikace. Komunikace se vede otevřenými kanály a není kontrolována konzistence dat a je třeba hledat definici metod na internetových stránkách projektu, které nemusejí být vždy aktuální.

#### Programy pro dávkové zpracovaní obecných úloh

Existuje mnoho systémů na vzdálné dávkové zpracování obecných úloh. Zde bude diskutováno jen několik nejznámějších.

**OpenPBS**

OpenPBS (Open Portable Batch System, [3]) byl původně vyvinut pro NASA na začátky 90. let společností Altair pro UNIXové clustery počítačů. OpenPBS je

sice open-source program, ale jeho vývoj byl ukončen; jeho oficiálním pokračovatelem je komerční systém PBS Professional. OpenPBS Server se skládá ze serveru pro komunikaci mezi jednotlivými částmi, rozvrhovače úloh a spouštěče úloh. Rozvrhovač úloh vytvoří frontu úloh na základě priorit a spouštěč úlohu zadá příslušnému solveru (cluster uzlu). Úlohy jsou zadávány pomocí jednoho ze systémových skriptovacích jazyků (se speciální syntaxe pro parametry pro PBS), skripty jsou spouštěny v příkazové řádce (v Linuxu např. `/bin/bash`) a zadání a výsledky jsou ukládány do souborů. Je také možné úlohy z příkazové řádky spravovat a získávat o nich informace. Programy řešící samotnou úlohu musí být již přítomny na cílovém počítači. Jako příklad implementace OpenPBS může sloužit CIV (Centrum Intenzivních Výpočtů, [4]), kde je po registraci u správců systému možno spouštět úlohy na super-výkonných počítačích.

Výhodou tohoto systému (a jeho klonů, případně pokračovatelů) je jejich vysoká variabilita. Nevýhodou naopak nutnost mít na PBS serveru systémový účet (pro přihlášení přes SSH), nutnost manuálního vytváření skriptů a přenosu vstupních a výstupních souborů na osobní počítač.

**TORQUE Resource Manager**

TORQUE Resource Manager ([5]), je příkladem open source iniciativy, původně založené na OpenPBS, určené pro větší clusterové operace.

**Tivoli Workload Scheduler**

Tivoli Workload Scheduler ([6]) je profesionální systém IBM, stvořený divizí Tivoli Software, pro řešení heterogenních dávkových úloh na firemní úrovni.

**Open Source Job Scheduler**

Open Source Job Scheduler ([7]) je open source řešení umožňující zadávání úloh pomocí web services, které se ale musí ručně konfigurovat a které nenabízí automaticky generovaný seznam funkcí pro snadnější vytváření klientů.

**Sun Grid Engine**

Sun Grid Engine ([8])je open source systém pro zadávání dávkových úloh podporovaný Sun Microsystems. Projekt se pokouší integrovat existující řešení pro vzdálené volání optimalizačních nástrojů a vytvořit standardy pro distribuované výpočty.

### 1.2.2 Výhody navrženého řešení

**Standardizace a interoperabilita**

Ve většině zmíněných systémů probíhá některá část komunikace pomocí nestandardních protokolů. To je třeba s ohledem na specifické požadavky jednotlivých aplikací, ale je složité se jednotlivým protokolům přizpůsobit nebo najít specifikaci k jejich používání.

Tyto problémy mohou být vyřešeny používáním Web Services (WS, [9]). WS je řešení definované konsorciem W3C ([10]) pro interoperabilní komunikaci mezi programy po síti a jsou relativně dobře specifikovány. WS jsou velmi rozšířeným způsobem komunikace, protože existuje mnoho implementačních nástrojů v různých

programovacích jazycích, které umožňují jejich snadnou konfiguraci a generování. Ve většině případů je také součástí implementace WS jejich strojově čitelný popis, pomocí kterého o sobě Web Service publikuje informace, proto je eliminováno studium dokumentace k jejich používání (na technologické, ne aplikační úrovni). Z těchto důvodů jsou v této práci použity Web Services pro komunikaci.

Dalšími používanými možnostmi pro meziaplikační komunikace jsou XML-RPC ([19]) a CORBA (Common Object Request Broker Architecture, [20]). Výhodou XML-RPC je hlavně jeho jednoduchost a nenáročnost, ale to je zároveň jeho nevýhodou pro složitější aplikace. Výhodou CORBY je mimo jiné binární komprese přenášených dat, a tedy menší objem přenášených dat, než v případě textově orientovaných protokolů (založených většinou na XML), jeho nevýhodou jsou ale problémy v implementaci v jednotlivých jazycích a chybějící implementaci pro některé další (hlavně chybějící oficiální implementace pro Microsoft.NET framework).

### Zabezpečená komunikace

V některých ze zmíněných systémů probíhá komunikace mezi klientem a serverem přes nezabezpečené kanály. Není automaticky kontrolována integrita přenesených dat (tj. zda při přenosu dat nedošlo k jejich ztrátě nebo jejich záměrnému poškození nebo změně), data nejsou přenášena přes šifrované kanály (tj. posluchač na přenosové cestě mezi serverem a klientem si může přečíst data), není kontrolován uživatel, který se pokouší data získat (data nejsou spjata s určitým uživatelem), a klient nekontroluje identitu serveru (posluchač na přenosové cestě může předstírat, že je serverem, zadržet data a odpovědět neplatnými daty).

Při použití WS se výše zmíněným rizikům dá předejít používáním rozšířených specifikací pro bezpečnost WS. Mezi tyto specifikace patří WS-Security ([11]), které zajišťuje zabezpečení dat z hlediska autentifikace, autorizace a utajení obsahu zprávy. V této práci je kladen důraz na používání rozšířených specifikací WS pro bezpečnost a spolehlivost dat.

### Ukládání dat

V případě některých systémů pro dávkové zpracování je jako úložiště dat jednotlivých úloh použita disková nebo operační paměť serveru. Toto řešení je sice výhodné z hlediska rychlosti zpracování dat, ale nevýhodné z hlediska přístupu k datům z jiných aplikací a kvůli možnosti ztráty dat při neplánovaném ukončení aplikace. Další nevýhodou jsou nestandardizované možnosti práce s daty.

Bude proto použita pro ukládání dat SQL (Structured Query Language, [13])) databáze. Ta zajišťuje standardizovaný přístup k datům a práci s nimi a uchování dat v případě neplánovaného ukončení aplikace.

### Systémové zdroje

Systém byl navrhován tak, aby šetřil systémové zdroje. Proto je ve většině případů používána událostmi řízená architektura namísto stálé nebo periodické kontroly požadovaných dat.

### Administrace serveru a přidávání nových funkcí

Většina administrace serveru se děje přes webové rozhraní. To umožňuje snadnou správu nastavení a vlastností úloh z počítačů majících přístup k serveru. Zbylá nastavení jsou definována v XML souborech umístěných na serveru, aby se předešlo zásahům do zdrojových kódů systému.

To je také bráno v úvahu pro přidávání nových funkcí. Po definování umístění zdroje nových funkcí v administraci si systém sám zjistí, jaké funkce zdroj nabízí.

### Transparentní distribuovaná architektura

Každá z částí systému může být umístěna na jiném počítači a pouze serverová část systému ke svému chodu potřebuje specifické zdroje - tedy obecnou databázi, přístupnou pomocí ODBC (Open DataBase Connectivity, [14])) a Microsoft IIS (Internet Information server, [15]). To je přínosné ve chvíli, kdy jsou buď klient nebo zdroj funkcí k dispozici pouze z jiného prostředí (např. operační systém nebo jinak izolovaný počítač).

Systém je také navrhován jako plně transparentní z hlediska klienta, který předává WS pevně typovaná data (datům je přiřazen typ, není tedy nutné jakékoli parsování dat na straně klienta) a systém se chová, jako by klient volal lokální funkci.

# Kapitola 2

# Popis technologií

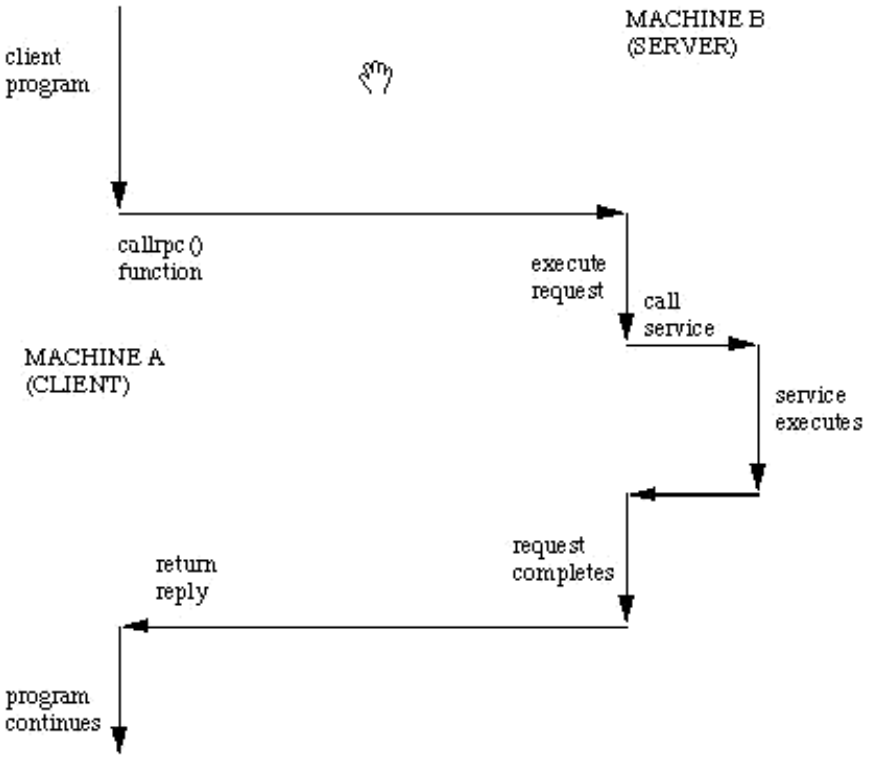
V této kapitole budou popsány technologie související s prací. Více prostoru bude věnováno popisu Windows Communication Foundation, protože představuje základní technologii použitou v této práci.

## 2.1 XML

XML ([16]) je obecný značkovací jazyk pro práci se strukturovanými daty, který byl vyvinut a standardizován konsorciem W3C. Tento jazyk je určen především pro výměnu dat mezi aplikacemi, čehož využívají některé protokoly použité v této práci. XML je ale také diky své hierarchické struktuře a relativní redundanci ve značkování dat snadno čitelný pro člověka, proto je zde použit pro některé části konfigurace.

## 2.2 RPC

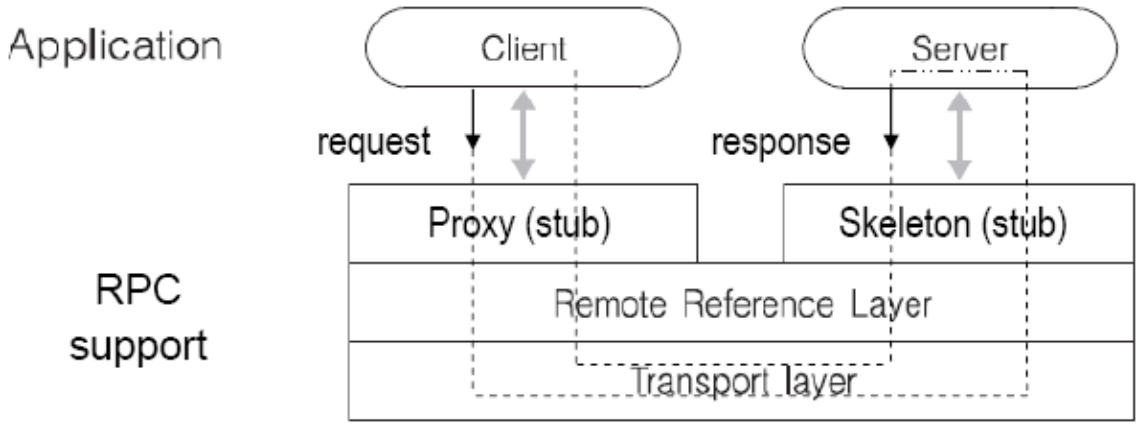
Obecné RPC (Remote Procedure Call, Vzdálené volání procedur [18]) je populární paradigma pro implementaci klient-server modelu distribuovaných aplikací. Volání je inicializováno odesláním klientské žádosti (zprávy) známému serveru, aby zahájil požadovanou akci se zadanými parametry. Odpověď je klientovi odeslána po skončení vykonávání akce na serveru a po dobu jejího vykonávání je klient zablokován. Existuje mnoho rozdílů v různých implementacích RPC, což způsobuje, že existuje velké množství různých vzájemně nekompatibilních RPC protokolů.




Obr. 2.1: Sekvence volání RPC.

Složky mechanismu RPC:

- Klientské rozhraní (proxy),
- Serverové rozhraní (skeleton),
- Vlastní kód RPC procedury na serveru.




Obr. 2.2: Obecné schéma RPC.

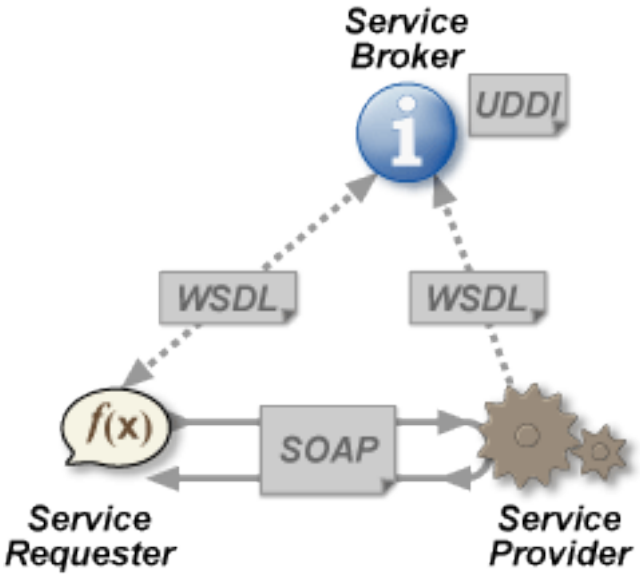
## 2.3 SOAP

SOAP (Simple Object Access Protocol, Service Oriented Architecture Protocol, [17]) je protokol pro výměnu strukturovaných dat prostřednictvím počítačové sítě v tzv. zprávách. Zprávy SOAP jsou založeny na formátu XML a jsou složeny z obálky a kódovacích pravidel. Inicializace přenosu a přenos zpráv většinou spoléhá na ostatní aplikační protokoly, především na HTTP (HyperText Markup Language) a RPC (Remote Procedure Call). Zprávy ve formátu SOAP se používají jako základní vrstva protokolu Web Services.

## 2.4 Web Services

Web Services (WS, [9])je řešení definované konsorciem W3C ([10]) pro interoperabilní komunikaci mezi programy distribuovanými po počítačové síti. WS používají XML zprávy ve formátu SOAP standardu a často také nabízejí strojově čitelnou definici nabízených služeb definovanou v jazyce WSDL. Služba naslouchá příchozím požadavkům na určitém místě, tzv. endpointu. Endpoint sdružuje všechny prostředky služby, jako jsou práve WSDL, samotná služba, případně další zdroje služby.

Aby se zvýšila interoperabilita, publikuje WS často také svůj profil. Ten, kromě specifikování základních částí (SOAP, WSDL) v určité verzi, publikuje také další požadavky pro omezení využití základních částí. To pak pomáhá při vytváření WS, které odpovídají určitému profilu.




Obr. 2.3: Základní schéma webových služeb.

### 2.4.1 WSDL

WSDL (WebService Description Language, [23]) popisuje komunikaci s web services ve formátu XML. Určuje jaké funkce web services nabízí, jaké mají parametry a jaké jsou návratové hodnoty. WSDL není podmínkou pro fungující webovou službu,

ale umožňuje automatizaci vytváření klientů pomocí nástrojů dodávaných s překladači pro různé implementace WS.

### 2.4.2 WS-*

Existuje mnoho specifikací asociovaných s WS. Souhrnně se označují jako WS-* (kde * značí jakoukoli specifikaci pro WS, protože všechny používají WS- prefix), ačkoli nejsou nijak centrálně spravovány. Liší se jak stádiem vývoje, tak v jejich důležitosti. Jejich funkce se mohou navzájem překrývat nebo dokonce potlačovat.

Následuje krátký popis rozšíření WS použitých v této práci.

#### WS-Security

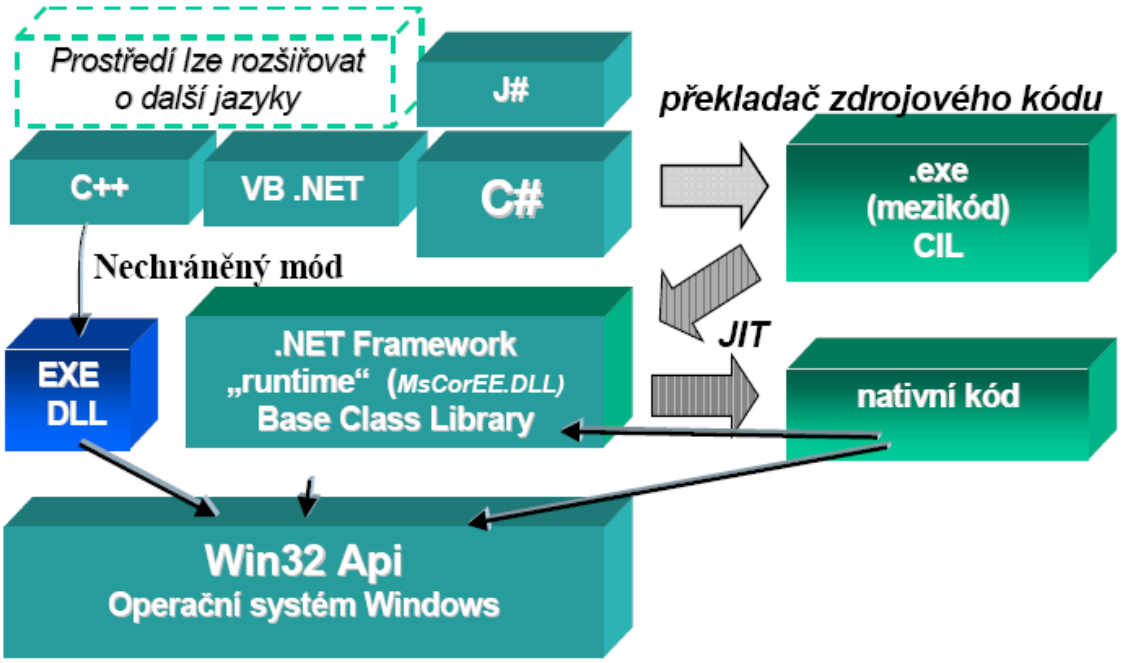
WS-Security ([11]) je komunikační protokol poskytující způsoby pro zabezpečení WS. Protokol byl standardizován společností Oasis-Open a obsahuje další přidružené rozšířené bezpečnostní specifikace (WS-SecureConversation, WS-Federation, WS-Authorization, WS-Policy, WS-Trust, WS-Privacy, WS-Test). Protokol obsahuje specifikace, jak vynutit integritu a utajení zpráv pomocí používání SAML, Kerberos nebo certifikátů. To zajišťuje zabezpečení na zprávy, ne přenosového kanálu.

#### WS-MetadataExchange

WS-MetadataExchange ([24]) je komunikační protokol, který přidává WS schopnost publikovat veškeré informace o sobě, včetně profilů a používaných omezení - tedy metadat. Toho je využíváno při tvorbě klientů plně odpovídajících konfiguraci serveru.

## 2.5 .NET

.NET Framework ([21],[22]) je relativně nové objektově orientované API pro běh aplikací. To umožňuje zavést multi-jazykové (C#.NET, VB.NET, J#.NET, C++, JScript, XAML) vývojové prostředí pro MS Windows. Všechny jazyky jsou v chráněném módu překládány do CIL (Common Intermedialte Language). Do nativního binárního kódu (CLR) je pak přeloženo CIL při načítání programu. Součástí .NET Frameworku jsou také API ASP.NET (Active Server Pages) pro tvorbu aktivních serverových stránek a ADO.NET (ActiveX Data Object) pro snadnější přístup k databázím. .NET Framework obsahuje velké množství předdefinovaných knihoven usnadňující často se opakující úkoly v programování. V této práci byl použit Microsoft.NET Framework ve verzi 3.5 SP1, vydaný v roce 2008.




Obr. 2.4: Struktura překladu a běhu programů v .NET frameworku.

### C#

C# (CSharp) je jednoduchý, moderní, objektově orientovaný programovací jazyk odvozený z jazyka C++. Překladač pro C# je dodáván jako část Microsoft Visual Studia .NET. C# byl použit jako hlavní programovací jazyk pro tuto práci.

### ASP.NET

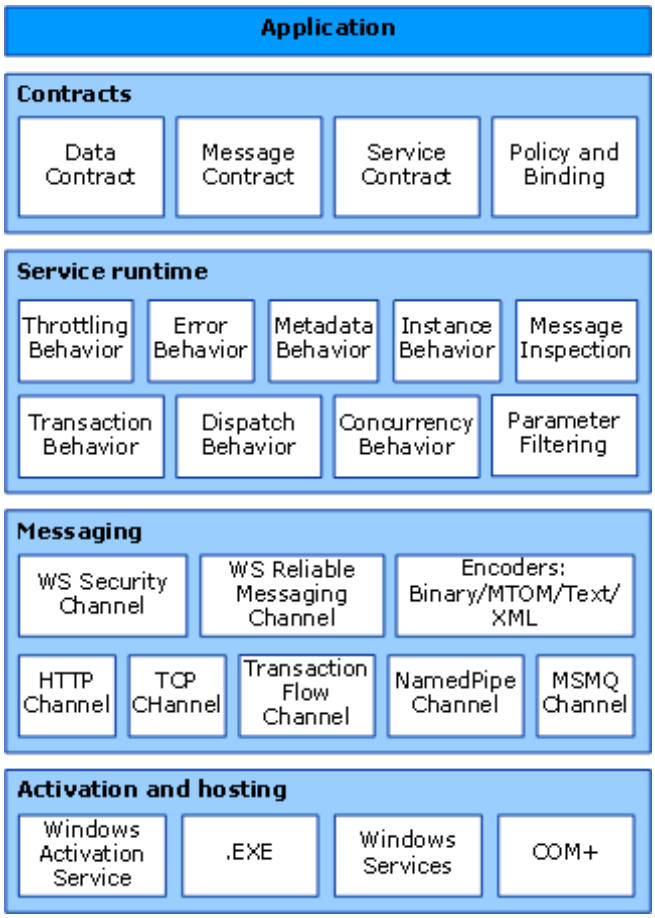
ASP.NET (Active Server Pages) je programovací prostředí pro tvorbu dynamických webových aplikací postavené na Microsoft.NET Frameworku. Obsahuje statické (X)HTML tagy a dynamickou část, která může být definována ve všech jazycích Microsoft.NET Frameworku. ASP.NET bylo v práci použito pro tvorbu webového administračního rozhraní a webového klienta.

## 2.6 WCF

Windows Communication Foundation (WCF) je prostředí a sada API funkcí pro vytváření systémů, které posílají zprávy mezi WS a klientem. WCF je navržena tak, aby nabídla lehce zvládnutelný přístup k distribuovaným programům, široké interoperabilitě a přímé podpoře pro architekturu orientovanou na služby (Service Orientated Architecture, SOA). WCF podporuje mnoho způsobů vývoje distribuovaných aplikací díky vícevrstvé architektuře. Na základní asynchronní vrstvě pro předávání netypovaných dat jsou postaveny pevně typované (u přenášených dat je definován jejich typ) vyšší vrstvy, které zajišťují bezpečnost, spolehlivost, transakce atd.

Typový programovací model (nazývaný také Service model, typ je u přenášených dat pevně dán) je navržen tak, aby usnadnil vývoj distribuovaných aplikací pomocí flexibilního a rozsáhlého mapování Web Services do .NET Framework CLR v jazycích jako C# a VB. To umožňuje interoperabilitu s existujícími distribuovanými systémy v .NET Frameworku, jako např. Message Queuing (MSMQ), COM+, ASP.NET Web Services, Web Services Enhancements (WSE) aj. V této práci bude používána WCF, hostované v IIS serveru, jako implementace pro WS.

### 2.6.1 Architektura WCF




Obr. 2.5: Základní vrstvy architektury WCF.

#### Kontrakty a Popisy

Kotrakty (Contracts) definují různé aspekty zprávy. Datové kontrakty (data contracts) popisují všechny parametry všech zpráv, které daná služba může vytvořit nebo přijmout. Parametry zprávy jsou definovány pomocí XML Schema Definition languge (XSD) dokumentů, což umožňuje, aby každý systém, který rozumí XML, uměl zpracovat tyto zprávy. Kontrakty zpráv (message contracts) definují přesněji části zprávy pomocí SOAP protokolu; to je třeba, pokud požadavky na interoperabilitu nutí k takovému upřesnění. Kontrakty služeb (service contracts) specifikují

popisy služeb samotných. V programovacích jazycích, které podporují interfacy (VB a C#), jsou pomocí interfaců tyto kotrakty služeb ditribovány.

Politiky (policies) a navázání (bindings) stanovují podmínky požadované pro komunikaci se službou. Např. navázání musí (přinejmenším) definovat transportní vrstvu, která bude použita, a encoding. Politiky obsahují bezpečnostní požadavky a ostatní podmínky, které jsou požadovány pro komunikaci se službou.

### Provoz služby

Vrstva provozu služby (service runtime) obsahuje chování, které nastává pouze v průběhu skutečného provozu této služby. Škrcení (throttling) řídí, kolik zpráv může být současně zpracováváno, což může být změněno, pokud poptávka po službě roste k přednastavenému limitu. Chybové chování (error behavior) specifikuje, co nastane, když dojde k vnitřní chybě služby. Např. kontroluje, jaké informace jsou o chybě zasílány klientovi (příliš mnoho informací může pomoci uživateli se zlými úmysly v pokusu o útok). Chování metadat (metadata behavior) určuje, jak a zda budou k dispozici metadata okolnímu světu. Chování instance (instance behavior) určuje, kolik instancí služby lze současně spustit (např. singleton určuje, že pouze jedna instance bude zpracovávat všechny zprávy). Transakční chování (transaction behavior) umožňuje rollback operací v transakci, pokud nastane chyba. Odbavovací chování (dispatch behavior) definuje, jak je zpráva zpracována WCF infrastrukturou.

Rozšiřitelnost (extensibility) umožňuje přizpůsobení průběhu zpracování. Např. inspekce zprávy (message inspection) je mechanizmus pro prohlížení částí zpráv a filtrování parametrů (parameter filtering) umožňuje přednastavit akce založené na filtrech, reagujících na obsah hlavičky zprávy.

### Posílání zpráv

Vrstva zpráv (messaging) se skládá z kanálů. Kanál (channel) je komponenta, která určitým způsobem zpracovává zprávy, např. autentifikací zprávy. Soubor kanálů je také znám jako zásobník kanálů (channel stack). Kanály operují se zprávou a záhlavím zprávy. To je odlišné od vrstvy provozu služby, která je především určena pro zpracovávání obsahu zprávy. Existují dva typy kanálů: transportní kanály (transport channels) a kanály protokolů (protocol channels).

Transportní kanály čtou a zapisují zprávy z/do počítačové sítě (nebo jiného endpointu). Některé dopravní kanály používají enkodér pro konverzi zpráv (které jsou reprezentovány jako XML Infosety) z a do bitové reprezentaci používané sítí. Příklady z transportů, jsou HTTP, named pipes, TCP a MSMQ. Příklady kódování jsou XML a optimalizované binární.

Kanály protokolů implementují zpracování zprávy často čtením nebo zápisem další hlavičky zprávy. Příkladem takových protokolů jsou WS-Security a WS-Reliability.

Vrstva zpráv znázorňuje možné formáty a vzory výměny dat. WS-Security je implementace specifikace zabezpečení přenosu umožňující zabezpečení na úrovni zprávy. Kanál WS-Reliable Messaging umožňuje zaručit doručení zprávy. Enkodéry přestavují mnoho nejrůznějších enkódování, které mohou být použity tak, aby vyhovovaly potřebám zprávy. Kanál HTTP deklaruje, že bude použit HyperText Transport Protocol k doručení zprávy. TCP kanál podobně specifikuje protokolu na TCP.

Kanál toku transakcí řídí zprávy v transakci podle určitého vzoru. Kanál Named Pipes umožňuje meziprocesovou komunikaci. Kanál MSMQ umožňuje interoperabilitu s MSMQ aplikacemi.

#### Hosting a spouštění

Ve své konečné podobě je služba program. Podobně jako ostatní programy musí služba běžet ve spustitelném programu. To je známé jako self-hosted služby.

Služby mohou také být hostovány či spouštěny ve spustitelném programu spravovaném externím agentem jako je IIS nebo Windows Activation Services (WAS). WAS umožňuje WCF aplikacím, aby byly automaticky spouštěny při jejich umístění na počítači, kde běží WAS. Služby lze také provozovat jako ručně spustitelné soubory (.exe soubory). Služba může být také spouštěna automaticky jako služba Windows. COM+ komponenty mohou být také hostovány jako WCF služby.

### 2.6.2 Kompatibilita

Základním komunikačním mechanizmem WCF jsou WS založené na SOAPu. Protože je SOAP standardizovaný protokol, aplikace založené na WCF mohou teoreticky komunikovat s ostatním softwarem, který rozumí SOAP protokolu. Vzhledem k velkému počtu rozšířených specifikací WS (WS-*) se ale liší podpora těchto rozšíření jednotlivými implementacemi WS.

Sun Microsystems založili open-source projekt Tango ([26]), který se snaží vyřešit nekompatibilitu mezi implementacemi WS - WCF v .NET Frameworku a WS v Javě, hlavně pak JAX-WS ([27]). Na projektu spolupracují jak Sun Microsystems, tak Microsoft. Pomocí utilit JAX-WS (nástroje wsimport, [28]) lze díky tomu vygenerovat klientský software v jazyce Java, který je kompatibilní s WCF. Protože Java není omezená pouze na prostředí Microsoft Windows, rozšiřují se tak možnosti nasazení WCF na UNIXové systémy. Java interface je také součástí programovacích prostředí jako je MATLAB, kde tak mohou být WS založené na WCF také nasazeny.

# Kapitola 3

# Popis systému

V této kapitole bude nejdříve uveden stručný popis systému a poté bude následovat podrobnější popis jeho částí.

## 3.1 Stručný popis

Celý systém slouží jako transparentní rozhraní pro snadný přenos zadání úloh (dat) na vzdálené počítače, případně i lokální počítač. Tomu je přizpůsobena architektura systému, která je ve stylu klient-server. Klientem je v tomto případě strana zadávající úlohu a server je strana přenášející, respektive řešící úlohu. Přenos dat je řešen pomocí Web Services (WS).

Na straně klienta je k použití WS třeba mít informace o nabízené službě a způsob jejích volání. To je možné buďto užitím zdrojových kódů nebo knihoven vygenerovaných pomocí nástrojů poskytovaných k WCF nebo užitím dynamického zjišťování nabízeních služeb pomocí knihoven .NET Frameworku. Poté, co má klient informace a způsob volání služeb, toho může využít v aplikační logice a začít využívat tyto služby.

Na straně serveru lze funkce nabízené službou rozdělit do dvou skupin. První skupinou jsou funkce poskytované koncovým řešitelským softwarem (tzv. solvery), tedy ty funkce, které slouží ke zpracování úloh (dat). Druhou skupinou jsou funkce, které jsou obecné a nezáleží na funkcích poskytovaných solvery. Obecné funkce slouží k práci s jednotlivými zadáními (tasky), zjišťování jejich stavu atp.

Aby se docílilo co největší variabilnosti, je mezi solvery a klienty vložena mezivrstva takzvaných složek (folderů). Foldery slouží k zapouzdření objektů souvisejících se službou. Tyto objekty jsou poskytované funkce, nastavení služby a kontrola přístupu ke službě. Každý folder poskytuje jednu službu s přiřazenými funkcemi. Jakémukoli folderu lze přidělit funkce z jakéhokoli solveru a zajistit tak, že například jedna služba neposkytuje příliš funkcí, což by znepřehledňovalo jejich výběr nebo že uživatel nemá přístup ke službám, které nejsou určeny pro něj. Nastavení folderu také kontroluje, zda může být služba folderu automaticky upravována v závislosti na změnách, které solvery nabízejí.

Foldery jsou generovány systémem na základě uživatelských požadavků. Pro konfigurovatelnost tohoto procesu byly zavedeny šablony pro foldery (templaty). Templaty obsahují soubory s implementaci obecných funkcí služby a soubor s nastavením

služby. Po vytvoření obsahuje folder pouze obecné funkce, další je třeba přiřadit z nabídky funkcí. Nastavení služby v template probíhá pomocí standardního konfiguračního souboru WCF. V konfiguračním souboru služby je definováno bezpečnostní nastavení služby a způsob komunikace se službou. Zvolené nastavení pak bude použito pro všechny foldery vytvořené podle této template.

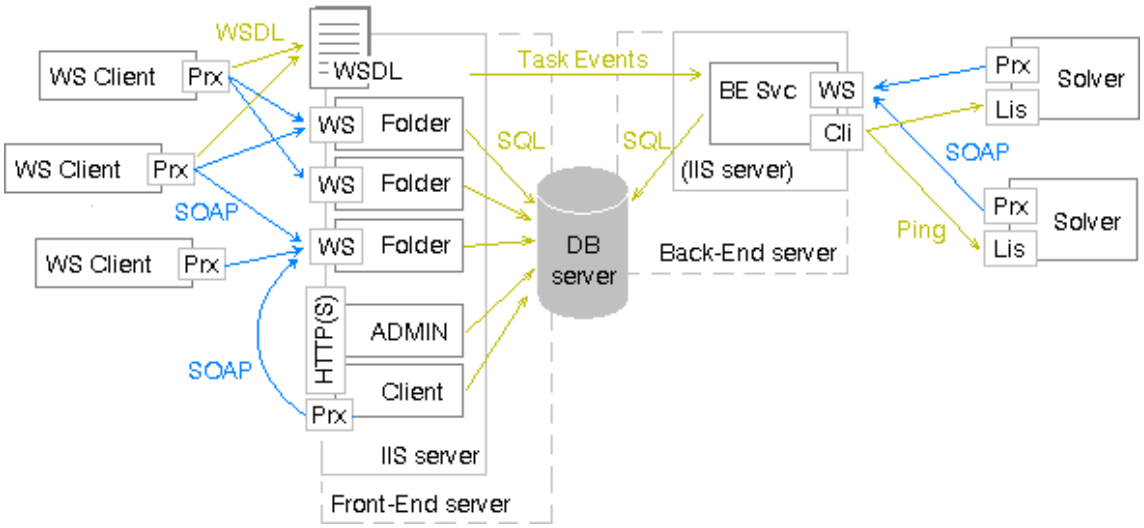
V tomto systému existují dva typy uživatelů - běžní uživatelé a administrátoři. Administrátoři mají plný přístup ke všem funkcím, běžní uživatelé mají jen omezený přístup. Uživatelé jsou pak organizováni do skupin, což umožňuje jejich snadnější správu. Např. přístup k službě folderu může být povolen jak samotnému uživateli, tak skupině uživatelů. Administrátoři ale mají automaticky přístup ke všem folderům.

Nastavení jednotlivých částí systému probíhá hlavně pomocí administračního rozhraní. Přístup k adminsitračnímu rozhraní mají pouze adminstrátoři. Administrační rozhraní je používáno pro vytváření nových folderů a jejich správu, definování umístění solverů, umístění templatů a správu uživatelů. Další nastavení, jako je absolutní umístění souborů jednotlivým částí, a specifické nastavení počítače, jako je hostname, je definováno v XML souborech umístěných v adresářích jednotlivých částí systému. Některé další části nastavení, týkající se prostření MS Windows nebo IIS serveru, jsou prováděna konfiguračními nástroji poskytovanými s prostředím.

SQL databáze využívána pro veškerou konfiguraci používanou administračním rozhraním. Databáze ale slouží hlavně jako vyrovnávací paměť mezi klienty a solvery. Klienti ukládají data do databáze a v databázi jsou zařazeny do virtuální fronty. Pořadí ve frontě závisí na prioritě tasku a jeho stáří. Solverům pak jsou tato data předávána, pokud to umožňuje stav solveru (tedy např. obsazenost). V případě, že instance funkce existuje ve více solverech, závisí to, zda je solveru předáno zadání, také na prioritě solveru.

### 3.1.1 Architektura systému

V této kapitole se budu krátce zabývat jednotlivými částmi architektury systému.




Obr. 3.1: Schéma architektury systému a základní komunikace mezi jednotlivými částmi systému.

#### Klienti

Klienti zajišťují vstup dat do systému, jejich čtení výsledků a administraci pomocí komunikace s WS nabízené serverovou částí. Klientem tedy může být jakýkoli software, který je schopen komunikovat s WCF. Ve schématu architektury (3.1) jsou označeni jako "WS Client".

Každému novému zadání úlohy (tzv. tasku) je přidělen jednoznačný identifikátor, který slouží k pozdějšímu odkazování na task.

#### Front-End Server

Front-End server (FE server) je jediná část, se kterou klient komunikuje. Skládá se ze tří částí: jednotlivých folderů (WS), administračního rozhraní a obecného webového klienta (který je také schopen zadávat tasky pomocí WS). Všechny tyto části jsou hostovány na Microsoft IIS serveru 6.0 a využívají jeho serverových technologií jako jsou ASP.NET a schopnost hostovat WCF. Tato část komunikuje přímo s databází. Ve schématu architektury (3.1) je tato část označena jako "Front-End server".

#### Databázový server

Jako databázový server je v tomto systému použit Microsoft SQL Server 2005, ale díky tomu, že bylo pro spojení s SQL Serverem použito ODBC rozhraní, je možné použít jakýkoli jiný databázový server, kompatibilní s ODBC. Ve schématu architektury (3.1) je tato část označena jako "DB server".

### Back-End server

Back-End server (BE server) je WS určená pro komunikaci se solvery. Zajišťuje vybírání dat tasků z DB a jejich odeslání solverům, kontaktování solverů v případě, že je třeba jim předat nějaký příkaz a ukládání výsledků tasků ze solverů do DB. Tato část je také provozována na IIS serveru, ale může být provozována i samostatně, například jako self-hosted service. Ve schématu architektury (3.1) je tato část označena jako "Back-End server".

### Solvery

Solvery jsou koneční adresáti zadání tasků, zajišťují funkcionalitu nabízených funkcí. Solver se skládá z posluchače (listeneru), který je kontaktován ve chvíli, kdy BE server požaduje nějakou akci po solveru - například vyzvednutí tasku pro určitou funkci solveru. Listener je tedy také WS klientem, který komunikuje s BE WS. Posluchač je implementován jako TCP/IP listener z důvodu šetření systémových zdrojů. Druhou částí solveru je spustitelný soubor, nebo knihovna s funkcionalitou solveru. Ve schématu architektury (3.1) jsou solvery označeny jako "Solver".

## 3.2 Podrobný popis implementace

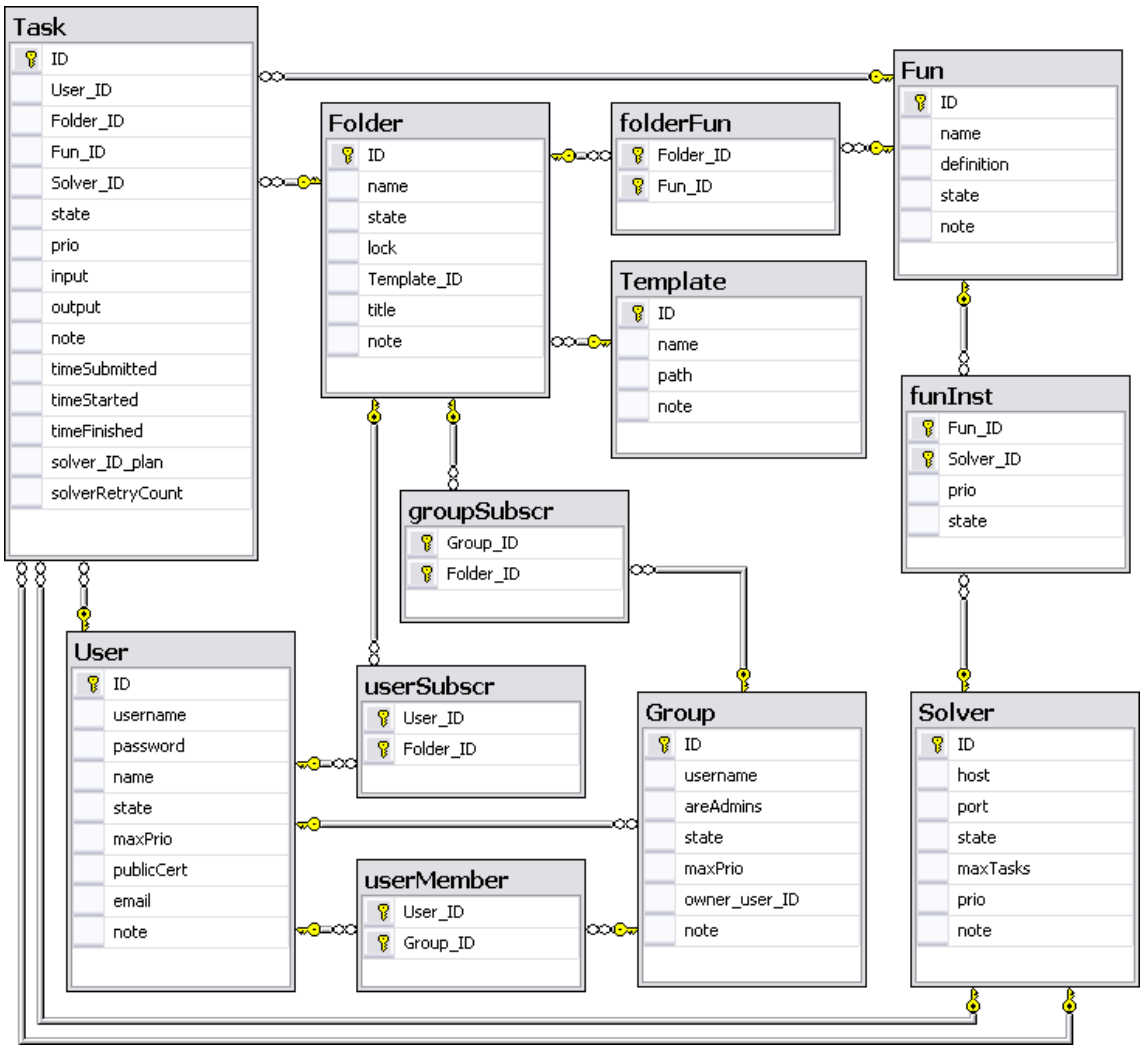
V této kapitole bude detailněji popsána implementace jednotlivých částí.

### 3.2.1 Databázový server

Jako databázový server je v tomto systému použit Microsoft SQL Server 2005, ale díky tomu, že bylo pro spojení s SQL Serverem použito ODBC rozhraní, je možné použít jakýkoli jiný databázový server kompatibilní s ODBC.

Použitá databáze je na stejném hostovacím serveru jako Front-End a Back-End, ale mohla by být stejně tak umístěna na vzdáleném serveru (detaily o nasazení, použitých instancích a přihlašovacích údajích jsou umístěny na přiloženém CD).

#### Schéma databáze




Obr. 3.2: Schéma databáze.

V následujících podkapitolách bude popsán význam sloupců a dat v systémové databázi.

**Tabulka Folder**

- ID, int - primární klíč tabulky,
- name,nvarchar(50) - jméno adresáře vytvořeného v souborovém systému, jméno musí být jedinečné.
- state, int - číslo, označuje stav folderu:
  - 0 - zakázaný,
  - 1 - povolený,
  - 2 - označený k updatování funkcí adresáře,
- lock, int - určuje, zda je možné automaticky přegenerovat kód folderu,
  - 0 - zamčený (nejde),
  - 1 - odemčený (jde updatovat),
- Template_ID, int - cizí klíč tabulcky Template - určuje, jaká template byla při vytváření folderu použita,
- title, nvarchar(50) - nadpis adresáře,
- note, nvarchar(255) - textová poznámka k folderu.

**Tabulka folderFun**

Určuje, jaké funkce folder obsahuje.

- Folder_ID, int - cizí klíč tabulky Folder,
- Fun_ID, int - cizí klíč tabulky Fun.

**Tabulka Fun**

Obsahuje všechny funkce publikované jednotlivými solvery.

- ID, int - primární klíč tabulky,
- name,varchar(50) - jmeno funkce,
- definition, varchar(8000) - serializovaná definice funkce - obsahuje definici parametrů, typu a jména funkce,
- state,int - stav funkce,
  - 0 - zakázaná,
  - 1 - povolená,
- note, nvarchar(255) - textová poznámka.

**Tabulka funInst**

Určuje, jaké funkce jsou k dispozici na jakých solverech.

- Fun_ID, int - cizí klíč tabulky Fun,
- Solver_ID, int - cizí klíč tabulky Solver,
- prio, int - priorita instance,
- state, int - stav instance
  - 0 - zakázaná,
  - 1 - povolená.

**Tabulka Group**

- ID, int - primární klíč tabulky,
- username, varchar(50) - unikátní jméno skupiny,
- areAdmins, int - určuje, že jsou všichni členové skupiny administrátory,
- state, int - stav skupiny,
  - 0 - zakázaná,
  - 1 - povolená,
- maxPrio, int - maximální priorita nastavitelná členy skupiny,
- owner_user_ID, int - vlastník skupiny,
- note, nvarchar(255) - poznámka.

**Tabulka groupSubscr**

Určuje, které foldery mohou členové dané skupiny používat,

- Group_ID, int - cizí klíč tabultky Group,
- Folder_ID, int - cizí klíč tabulky Folder.

**Tabulka Solver**

- ID, int - primární klíč tabulky,
- host, varchar(50) - hostname pro identifikaci solveru BE serverem,
- port, int - port, na kterém poslouchá listener solveru,
- state, int - stav solveru,

  - -4 - error solveru,
  - -3 - příkaz k zakázání,
  - -2 - zakázáno,
  - -1 - příkaz k zastavení tasků,
  - 0 - tasky zastaveny,
  - 1 - příkaz k publikování funkcí,
  - 2 - solver k dispozici,
  - 3 - solver plný - limit tasků byl dosáhnut,
- maxTasks, int - maximální počet paralelně běžících tasků na solveru,
- prio, int - priorita solveru,
- note, nvarchar(255) - poznámka.

**Tabulka Task**

- ID, int - primární klíč tabulky,
- User_ID, int - cizí klíč tabulky User, identifikace zadavatele,
- Folder_ID, int - cizí klíč tabulky Folder, identifikace folderu, kde byl task zadán,
- Fun_ID, int - cizí klíč tabulky Fun, identifikace typu tasku,
- Solver_ID, int - cizí klíč tabulky User, identifikace solveru, na kterém je task řešen,
- state, int -
  - 0 - error tasku,
  - 1 - zařazený do fronty,
  - 2 - zadaný na solver,
  - 3 - vyřešený,
  - 4 - vyřešený a navrácený uživateli,
- prio, int - priorita tasku,
- input, nvarchar](max) - vstupní argumenty funkce,
- output, nvarchar](max) - návratová hodnota funkce,
- note, nvarchar](255) poznámka,
- timeSubmitted, datetime - čas, kdy byl task zařazen do fronty,
- timeStarted, datetime - čas, kdy byl task zadán solveru,

- timeFinished, datetime - čas, kdy byl task vyřešen,
- solver_ID_plan, int - plán BE serveice, na jaký solver task umístit
- solverRetryCount, int - počet průběhů vlákna BE service bez toho, aby task získal naplánovaný solver.

**Tabulka Template**

Šablona pro foldery,

- ID, int - primární klíč tabulky,
- name, nvarchar jméno šablony,
- path, nvarchar(255) absolutní cesta k šabloně,
- note, nvarchar(1023) poznámka.

**Tabulka User**

- ID, int - primární klíč tabulky,
- username, varchar - užvatelské jmého,
- password, varchar(50) - hash hesla,
- name, nvarchar(50) - vlastní jméno uživatele,
- state, int,
  - 0 - zakázaná,
  - 1 - povolená,
- maxPrio, int - max. priorita povolená zadávat uživatelem,
- publicCert, nvarchar(1023) = veřejný certifikát uživatele,
- email, varchar(50) - email uživatele,
- note, nvarchar(1023) poznámka.

**userMember**

Uživatel je členem těchto skupin,

- User_ID, int - cizí klíč tabulky User,
- Group_ID, int - cizí klíč tabulky Group.

**userSubscr**

Uživatel má právo k přístupu do folderu,

- User_ID, int - cizí klíč tabulky User,
- Folder_ID, int - cizí klíč tabulky Folder.

### 3.2.2 Front-End Server

Front-End je hostován na IIS serveru. Web Site byl přidělen self-signed certifikát, tj. certifikát podepsaný nedůvěryhodnou autoritou, aby bylo možno komunikovat přes HTTPS. Jako technologie pro tvorbu dynamického obsahu webových stránek bylo použito ASP.NET s C# jako "code-behind" jazykem (detaily o nasazení, použitých instancích a přihlašovacích údajích jsou umístěny na přiloženém CD).

Celá FE aplikace je umístěna v jednom adresáři. V tom lze také nalézt hlavní konfigurační soubor aplikace - web.config. V tom je definováno, že se k aplikaci může přistupovat pouze přes HTTPS protokol a omezení přístupu k jednotlivým adresářům - do administračního rozhraní mají přístup pouze administrátoři a ke webovému klientovy mají přístup pouze autentifikovaní uživatelé (administrátoři jsou současně také uživatelé). Kontrola autentifikace a role uživatele probíhá pomocí implementace MembershipProvidera (pro autentifikaci uživatele) a RoleProvidera (pro určení role uživatele). Tito provideři jsou pak snadno použitelní se standardními komponenty ASP.NET. Provideři jsou nastaveni, aby kontrolovali přihlašovací údaje oproti databázi systému. Implementace providerů je poskytnuta v knihovně `bin/Lib.dll`.

V kořenovém adresáři jsou přístupné dvě jednoduché webové stránky - Default.aspx, která obsahuje rozcestník k aplikacím a Login.aspx, která obsahuje přihlašovací formulář. Přihlašovací údaje jsou sice zasílány stránce v plain-textu, ale díky vynucení použití HTTPS to není bezpečnostní riziko. Pokud je uživatel administrátor, obsahuje rozcestník odkazy jak na administrační rozhraní, tak do webového klienta, pokud je běžným uživatelem, obsahuje pouze odkaz do klienta, a pokud není přihlášen, neobsahuje odkazy žádné.

Každý folder je v metabázi IIS nastavený jako samostatná aplikace, takže se na ně přihlašování do webového rozhraní nevztahuje.

Jak už bylo uvedeno výše, skládá se FE ze tří hlavních částí, které budou popsány v následujících kapitolách.

#### Administrační rozhraní

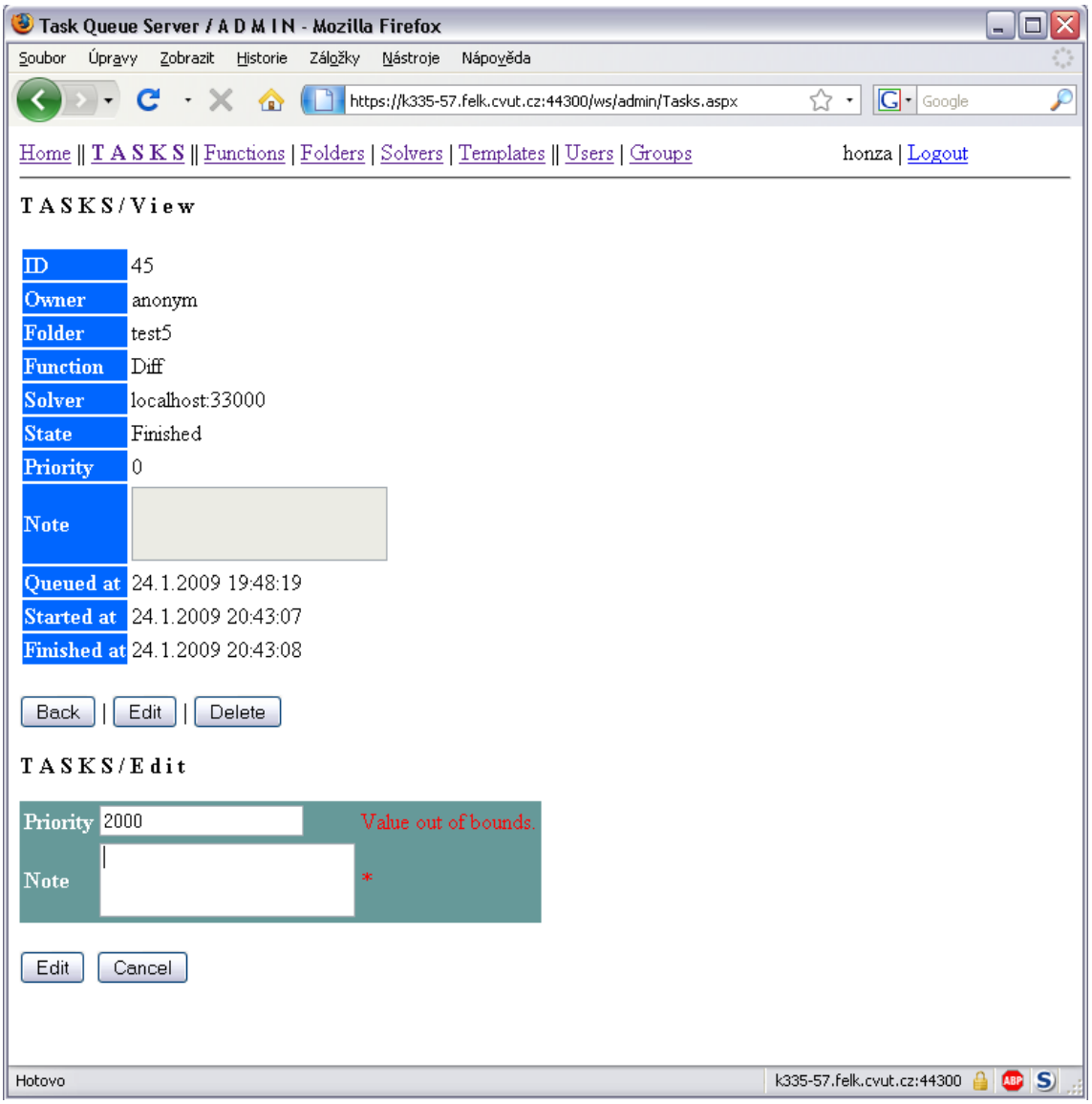
Administrační rozhraní je přístupné z kořenového adresáře pod adresářem "admin". Administrační rozhraní z velké části kopíruje strukturu databáze. To umožňuje nasazení takzvané data-driven architecture, která usnadňuje práci s daty v databázích pomocí předdefinovaných kompoment .NET. Je rozděleno do osmi sekcí, které odpovídají tabulkám, které nejsou typu m:n a tedy definují vlastní data. Ke každé z nich je pak přiřazena administrace členů tabulek m:n pro vybraný řádek hlavní tabulky, případně další specializované akce.

**Home.aspx**

Obsahuje rozcestník pro administrační rozhraní a odkazy nazpět do kořenového adresáře a na web klienta.

**Tasks.aspx**

Stránka obsahuje přehled tasků, rozdělených podle jejich stavu. Pokud je task vybrán, jsou zobrazeny jeho detaily a možnosti k vymazání a editaci tasku. Pokud je task zrušen (Cancel) nebo vymazán (Delete) a je právě řešen na solveru, je task nastaven do stavu "ke zrušení", respektive k "k vymazání" a BE serveru je signalizována nová událost. Podobně pokud je task znovu zařazen do fronty (do stavu Queued), tak je BE signalizována nová událost také. Není dovoleno definovat nové tasky, protože to se děje jen pomocí WS.




Obr. 3.3: Detail ukončeného tasku v administračním rozhraní s neúspěšným pokusem o editaci jeho priority.

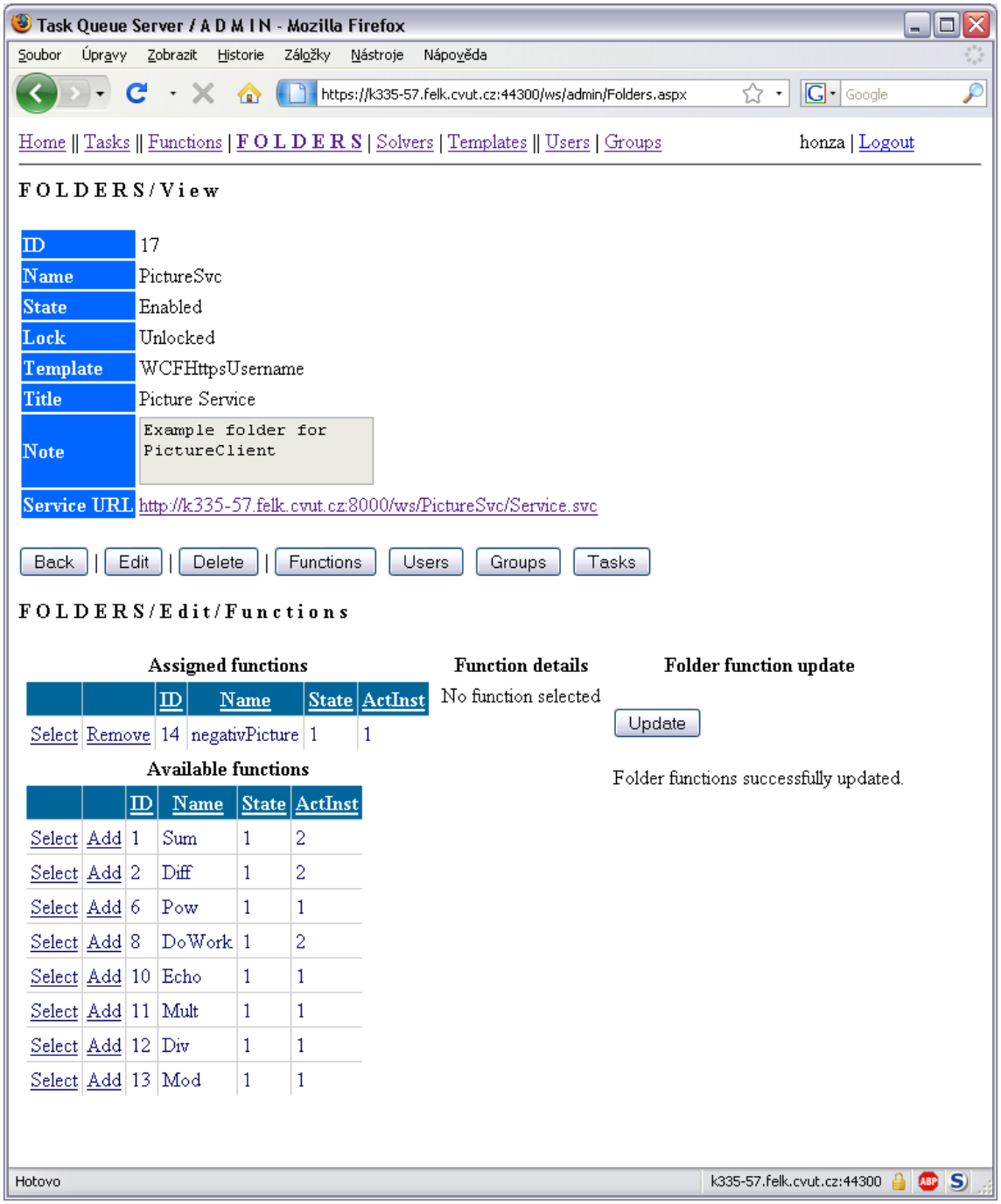
**Functions.aspx**

Stránka obsahuje pouze přehled funkcí. Detail funkce pak nabízí možnost přiřadit funkci folderu, což změní stav folderu na "k updatu", pokud není folder zakázaný nebo zamknutý. Dále možné povolit nebo zakázat instanci funkce na určitém solveru.

**Folders.aspx**

Základní pohled nabízí klasický přehled všech definovaných objektů tabulky, ale také možnost updatovat všechny foldery ve stavu "k updatu" najednou, pokud nejsou uzamčeny, a odkaz na vytvoření nového folderu. Při zadávání definice nového folderu jsou všechny zadávané údaje validovány a pokud je vše v pořádku, rozhraní se pokusí vytvořit folder. Více informací o vytváření folderů obsahuje další kapitola. Po vytvoření folderu přesměruje rozhraní uživatele k detailu vytvořeného folderu a k nabídce funkcí. Poté, co uživatel přiřadí folderu nějaké funkce je třeba folder updatovat (3.4, jen pokud není folder zamčený nebo zakázaný), jinak bude při volání jeho funkcí nastávat výjimka.

Detail funkce obsahuje také WS URL, na které je přístupná informační stránka generovaná samotnou WS. Dále je možno přiřadit folderu uživatele a skupiny, které mají přístup k jeho službě. Poslední možností je prohlídka seznamu tasku zařazených službou tohoto folderu.




Obr. 3.4: Detail folderu v administračním rozhraní s přehledem jeho funkcí.

### Solvers.aspx

Stránka Solvers.aspx obsahuje přehled všech solverů, které jsou známy systému. Definováním nového solveru se myslí zařazení definice solveru do databáze ve stavu "Discover", tedy prohledej. Po uložení soveru do databáze je BE serveru oznámena nová událost a BE se postará a zjištění jeho funkcí. Po vybrání konkrétního solveru se zobrazí standardní možnosti editace a vymazání. Navíc je zde možno povolovat a zakazovat jednotlivé funkce na solveru. Pokud neexisují žádné další instance dané funkce po zakázání instance funkce, jsou k updatu označeny ty foldery, které funkci

obsahují.

**Templates.aspx**

Webová stránka Templates.aspx obsahuje definované templaty a umožňuje definovat nové pomocí cesty k jejím souborům. Detail template obsahuje možnosti k jejímu editování, vymazání a přehled folderů, ve kterých je použita.

**Users.aspx**

User.aspx obsahuje, jako předchozí stránky, možnosti pro přehled objektů, vytváření nových, editaci údajů a vymazání. Navíc ale ještě obsahuje dialog na definování nového hesla, přiřazení do adresáře, do skupiny a přehled zadaných tasků.

**Groups.aspx**

Stránka Groups.aspx je velmi podobná předchozí stránce, ale na rozdíl od ní obsahuje definici adminstrátorských skupin, které určují, kdo má neomezený přístup k systému.

**Foldery**

Folder je složka systému, která umožňuje organizování funkcí do služeb. Folder je vlastně služba, ale vzhledem k jejímu tvoření byla pojmenována jako folder, neboli složka. Folder je složen z tzv. folderových obecných funkcí, které jsou pro všechny foldery stejné, a specifických solverových funkcí, které folderu přiřadí administrátor. Foldery generuje administrační rozhraní pomocí tzv. templates. Zabezpečení WS folderu závisí na použité template (obsahuje mimo jiného konfiguraci WS).

Foldery jsou tvořeny jako IIS "Web Site", to znamená, že obsahují adresář `App_Code` v kořenovém adresáři aplikace. Adresář `App_Code` má tu vlastnost, že pokud se změní jeho obsah, projekt se velmi rychle automaticky překompiluje ve chvíli, kdy je žádána jeho služba. Toho se využívá pro vytváření nových folderů a změně už existujících folderů; stačí jen změnit text zdrojového kódu. Web Site projekty musí mít vytvořený adresář ve struktuře IIS a registrovat je jako aplikaci, pokud je požadavek používat automatické kompilování. Registrace do struktury IIS se děje při vytváření adresáře.

**Folderové funkce**

Folderové funkce jsou součástí template. Template je množina souborů a adresářů, pomocí kterých je při vytváření nového folderu tvořena služba. Využívá se vlastnosti .NETu, takzvaných "partial classes". Ta část, která je stálá, je obsažena v jednom souboru, variabilní část je obsažena ve druhém souboru, ale kompilátor je při překladu složí dohromady.

Služby folderů se vždy jmenují `<URL folderu>/Service.svc`. Některé informace o WCF jsou pomocí metadat distribovány jako interface pro danou funkci, proto jsou typickou strukturou pro WCF WS:

- `App_Code/Service.cs` - obsahující implementaci funkcí
- `App_Code/IService.cs` - iterface funkcí

- `Service.svc` - WCF service
- `web.config` - nastavení WCF Serice

V tomto projektu jsou soubory `Service.cs` a `IService.cs` využity pro definování folderových funkcí.

Veřejné folderové funkce služby jsou následující:

- `string getTaskStatus(int Task_ID)` - neblokovací funkce, jejíž návratová hodnota popisuje stav tasku, ID bylo zadáno jako parametr.
- `string[] listTasks()` - vrátí pole textových řetězců, které obsahují informace o taskách uživatele v daném folderu. Pokud je uživatel adminsitrátor, vrátí informace o všech taskách ve folderu.
- `string[] listTasks(int[] Task_IDs)` - vrátí pole textových řetězců, které obsahují informace o taskách uživatele v daném folderu pokud jsou jejich task ID obsaženy v seznamu task ID v argumentu funkce. Pokud je uživatel adminsitrátor, není brán zřetel na vlastnictví tasků.
- `int setTaskProperty(int Task_ID, string Property, string Value)` - nastaví zadanou vlastnost tasku (Property) na zadanou hodnotu (Value). Přípustné hodnoty property a možné asociované hodnoty value:
  - "state" (stav) - "cancel" (zrušit), "delete" (vymazat), "resume" (revokovat),
  - "priority" (priorita) - "0" - "100",
  - "note" (poznámka) - řetězec kratší než 255 znaků.

  Pomocí `setTaskProperty` je tedy hlavně možné smazat task ze serveru a zvýšit prioritu tasku. Priorita tasku je omezena právy uživatele definovanými pomocí administračního rozhraní.

Neveřejné folderové funkce jsou tyto:

- `int addTask(string fun, string input)` - uloží data tasku do databáze a vrátí task ID nově vytvořeného tasku.
- `getTaskResult(string fun, int Task_ID)` blokovací funkce, která ve zvětšujících se intervalech kontroluje, zda je task už hotový. V případě, že ano, vrátí návratovou hodnotu z databáze. Pokud v průběhu čekání nastane, že task už neexistuje (byl smazán) nebo je ve stavu "Queued" (zařazen do fronty) a není k dispozici žádná aktivní instance volané funkce, je čekání přerušeno a vrácena programová výjimka.

**Solverové funkce**

Solverové funkce jsou druhá, variabilní část, která je generována na základě definice v databázi. Definice v databázi je uložena jako serializovaný datový objekt (textová reprezentace objektu), který obsahuje všechny potřebné informace o funkci - jméno, návratovou hodnotu a pole argumentů, které obsahují kromě jména a typu

také hodnotu argumentu. Definice této třídy `Fun` je obsažena v celé serverové a solverové části systému, aby bylo možno tyto serializované informace podle potřeby deserializovat do objektové podoby a serializovat do textové podoby vhodné pro přenos a ukládání do databáze. Třída `Fun` (pro serializování celé funkce) a třída `FunArg` (pro serializování argumentů, která je součástí třídy `Fun`) je uložena v souboru `Fun.cs`

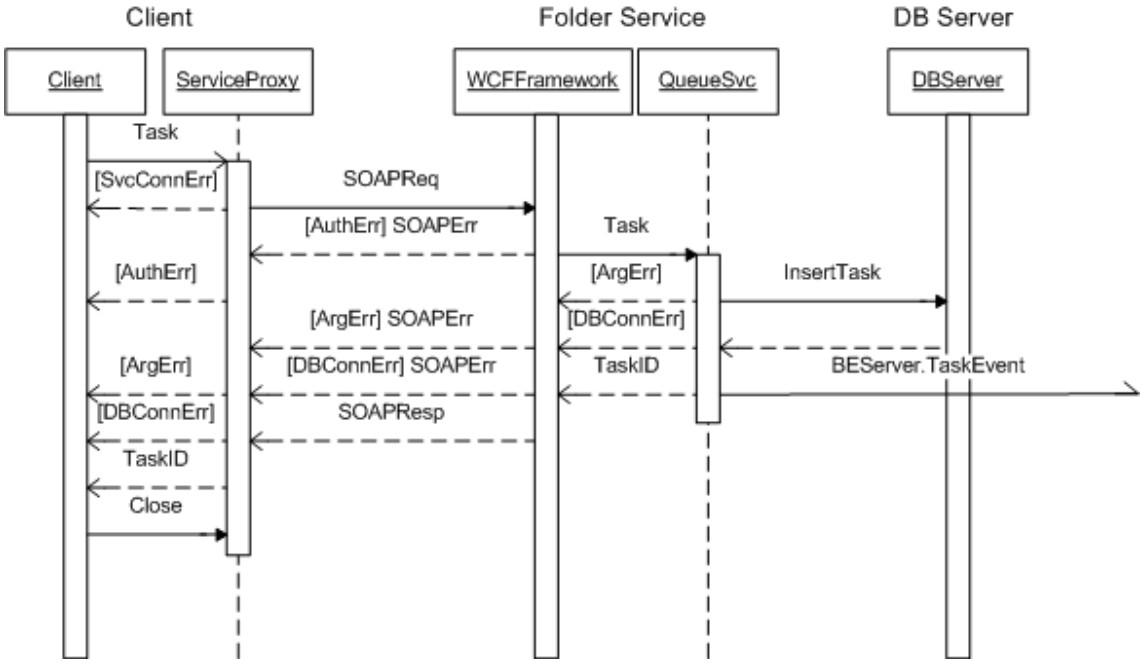
Do folderu jsou přidány následující soubory s partial classes, do kterých jsou generovány solverové funkce na základě definice v databázi:

- `Service.part.cs` - obsahující implementaci variabilní části
- `IService.cs` - iterface k implementaci variabilní části

Solverová funkce je definována dvojicí funkcí v `Service.part.cs`:

- `int <název funkce>(<parametry funkce>)` - zadávací část, která serializuje vstupní data a předá je folderové funkci `addTask`, která je uloží do databáze a vrátí task ID, které je vráceno jako návratová hodnota této funkce.
- `<návratová hodnota> <název funkce>Result(int Task_ID)` - zavolá folderovou blokovací funkci `getTaskResult`, která vrátí výsledek ve chvíli, kdy je task dokončen. Vrácená serializovaná data jsou deserializována a vrácena jako originální typ solverové funkce.

Následující sekvenční diagram (3.5)ukazuje způsob zadávání tasku na server ke zpracování. Uživatel zavolá solvervou funkci "Task", která předá serializovaná data do databáze, funkce kontaktuje BE service, aby oznámila novou taskovou událost, a vrátí task ID pro pozdější referenci tasku. Pokud nastane během zpracování výjimka, je propagována přes SOAP protokol až do klienta.




Obr. 3.5: Schéma mezi klientem a WS při zadávání tasku.

**Templaty**

Templaty jsou adresáře, které používá administrační rozhraní k vytváření nových folderů. Obsahují zdrojové soubory pro folderové funkce a konfigurační soubory pro folderovou službu. Zdrojové soubory se solverovými funkcemi jsou generovány ve chvíli, kdy je soubor vytvářen nebo při jeho updatu. Templaty byly vytvořeny proto, aby bylo možné vytvořit nové konfigurace pro vytvářené foldery bez nutnosti zásahu do kódu administrační aplikace.

Po vytvoření nové template je třeba jí zaregistrovat v administračním rozhraní. administrační rozhraní je připraveno generovat WCF web services (což je preferovaný přístup) a WSE web services (Web Service Enhancements - starší implementace WS od Microsoftu ([29]); přidáno pro případné klienty nekompatibilní s WCF).

V rámci práce byly vytvořeny 3 templaty:

- **WCFHttpsUsername** - toto je preferovaná template. Jako techonologii WS používá a WCF a navíc implementuje zabezpečení "TransportWithMessageCredential". Ten kombinuje výhody jak zabezpečení na úrovni zprávy, tak zabezpečení na úrovni přenosu. V tomto módu je zabezpečení na úrovni zprávy používáno pro autentikaci klienta a zabezpečení na úrovni přenosu pro autentifikaci serveru a zajištění utajení a integrity zprávy. Autentifikace probíhá pomocí MembershipProvider komponenty používané také v administračním rozhraní, která ověřuje totožnost uživatele pomocí uživatelského jména a hesla uloženého v databázi. Implementace providera je poskytována v knihovně `bin/Lib.dll`, jejíž vytvoření bylo také součástí práce. Zabezpečení na úrovni přenosu je zajištěno pomocí použití HTTPS jako nosného protokolu pro komunikaci. Všechny výše uvedené vlastnosti jsou nakonfigurovány pomocí souboru web.config, který je hlavním konfiguračním souborem webové služby. Dalším nastavením ve web.config jsou různé limity přenesených dat a timeouty pro spojení se službou. Limit pro přenášená data jsou 2GB.

  Při použití tohoto módu zabezpečení ale nastávají problémy v případě, že je použit self-signed certifikát (jako v této práci). Pak je třeba na straně klienta obejít kontrolu platnosti certifikátu buď přidáním certifikační autority, která certifikát vydala do skladu důvěryhodných certifikačních autorit na klientském počítači, nebo kontrolu certifikátu obejít, jak je to předvedeno v ukázkovém Picture Clientu, popsaném níže.

- **WCFHttpAnonym** - tato šablona by se měla používat jen v případě, že klientský software neumožňuje používání služby vytvořené pomocí předchozí šablony. Neobsahuje totiž žádnou autentifikaci ani zabezpečení na transportní úrovni (je používán protokol HTTP jako nosný protokol). Nastavení limitů je schodné s předchozí template.

  Zabezpečení takovéto služby je možné provést nepřímo pomocí omezení seznamu IP adres, které mají přístup ke službě, v administračním rozhraní serveru IIS.

- **WSEHttpAnonym** - stejně, jako předchozí šablona, i tato by se měla používat jen v případě, že klientský software neumožňuje používání služby vytvořené pomocí WCFHttpsUsername šablony, ale ani WCFHttpAnonyum . Ani tato

šablona totiž neobsahuje žádnou autentifikaci ani zabezpečení na transportní úrovni (je používán protokol HTTP jako nosný protokol).

Zabezpečení takovéto služby je možné provést nepřímo pomocí omezení seznamu IP adres, které mají přístup ke službě, v administračním rozhraní serveru IIS.

Při definování template v administračním rozhraní je třeba dodržet pojmenovávací konvenci pro šablony, protože se toho využívá v různých částech systému. Pokud je šablona určena pro WCF service musí začínat písmeny "WCF" a pokud neobsahuje autentifikaci kleintů, musí obsahovat slovo "Anonym".

**Zabezpečení folderů**

V případě, že je folder vytvořený pomocí šablony, která používá autentifikaci, je možné operovat s uživatelovou identitou. Pak je možné určit, zda má uživatel k adresáři, případně tasku, přístup a jakou maximální prioritu může taskům nastavit.

Běžný uživatel má přístup pouze k folderům, do kterých je zapsán (subscribed) nebo do kterých je zapsána aslespoň jedna ze skupin, kterých je členem. Běžný uživatel má právo pracovat jen s tasky, které byly podány v rámci folderu, který kontaktuje a které sám zadával. Maximální priorita, přidělitelná tasku, je určena jako maximum z maximální povolené priority uživatele a maximálních povolených priorit skupin, jíž je členem.

Uživatel s administrátorskými právy (ten, který je členem skupiny, která je označena jako administrátorská) má přístup ke všem adresářům, všem taskům folderu a může přidělovat maximální možnou prioritu - tedy 100.

V případě, že folder umožňuje anonymní přístup mají všichni připojení klienti přístup k všem taskům, které byly v rámci tohoto folderu zadány. Maximální možná priorita, kterou může nastavit anonymní uživatel, je 0, tedy minimální priorita tasku.

**Webový klient**

Webový kleint je součást FE aplikace, která umožňuje přihlásit se i běžným uživatelům. Stejně jako administrační rozhraní musí být prohlíženo přes HTTPS. Má sloužit uživatelům systému k přehledu stavu zadaných tasků, ke změně hesla a okrajově k zadávání tasků a prohlížení jejich výsledků.

Následuje popis webových stránek klienta.

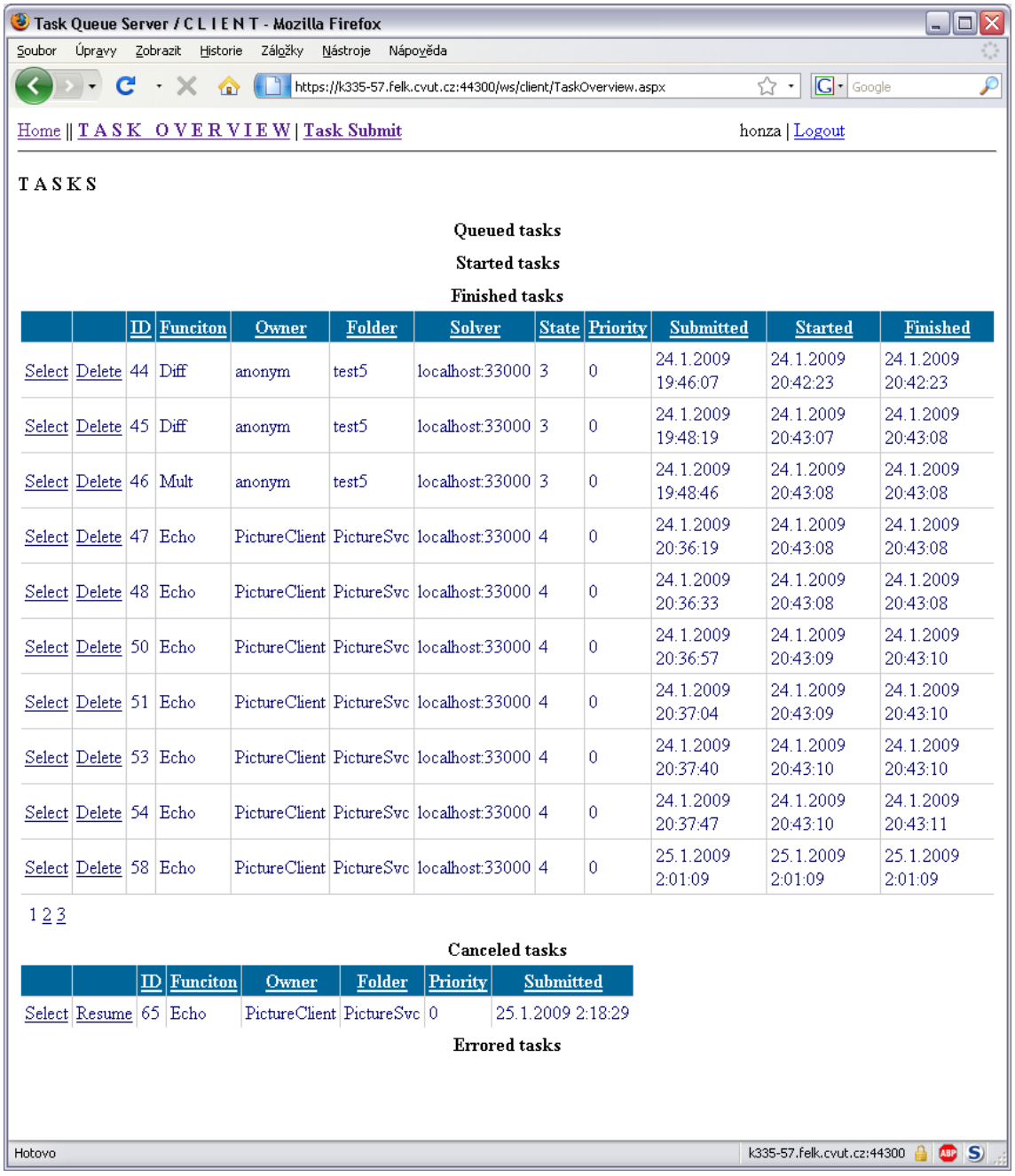
**Home.aspx**

Obsahuje rozcestník pro klientské rozhraní a odkazy nazpět do kořenového adresáře a administračního rozhraní (pokud na to má klient právo).

**TaskOverview.aspx**

Stránka obsashuje přehled tasků, rozdělených podle jejich stavu. Pokud je task vybrán, jsou zobrazeny jeho detaily a možnosti k vymazání a editaci tasku. Pokud je task zrušen (Cancel) nebo vymazán (Delete) a je právě řešen na solveru, je task nastaven do stavu "ke zrušení", respektive k "k vymazání" a BE serveru je signalizována nová událost. Podobně pokud je task znovu zařazen do fronty (do stavu

Queued), tak je BE signalizována nová událost také. Je zde také možnost prohlédnout si výsledky tasků, ale pouze v serializované podobě.




Obr. 3.6: Přehled zadaných tasků ve webovém klientském rozhraní když je přihlášen administrátor.

### TaskSubmit.aspx

Základní pohled nabízí uživatelovi přehled přístupných folderů. Pokud je folder vybrán, zobrazí se nabídka funkcí folderu, pokud folder umožňuje anonymní přístup. Po výběru funkce se zobrazí pole na zadání argumentů a tlačítko pro zadání tasku.

**UpdateFolders.aspx**
Aby měli běžní uživatelé možnost reagovat na výjimku, která signalizuje, že je folder označen pro update, je zde možnost, jak updatovat foldery z webového klienta.

**ChangePassword.aspx**
Umožňuje změnit uživatelské heslo.

### 3.2.3 Back-End server

Back-End (BE) server je WS určená pro komunikaci se solvery, která zajišťuje vybírání dat tasků z DB a jejich odeslání solverům, kontaktování solverů v případě, že je třeba jim předat nějaký příkaz, a ukládání výsledků tasků do DB. Detaily umístění BE serveru na hostovacím serveru jsou k dispozici na přiloženém CD.

BE service nabízí následující služby:

- `newTaskEvent` - činnost BE se spouští, když nějaký klient (buď folder nebo administrační rozhraní nebo solver) kontaktuje funkci `newTaskEvent`, která spustí v novém vláknu (aby neblokovala klienta) funkci `signalQueueTaskEvent`. Je povoleno pouze jedno běžící vlákno s touto funkcí, aby nedocházelo k chybám souběhu při práci s databází.

- `signalQueueTaskEvent` - funkce je řídícím funkcí aplikace - na základě dat v DB rozhoduje, které solvery je třeba kontaktovat, aby si vyzvedly task, aby vymazaly task, aby přerušily svou činnost nebo aby publikovaly své funkce. Kontaktování se děje pomocí vytvoření TCP/IP spojení s vybraným solverem (hostname a port jsou součástí definice solveru v DB).

  Funkce také plánuje, jaký task bude přidělen na jaký solver. Pořadí tasků se stanovuje podle priority tasku a stáří tasku (konstanta určující kolik jednotek priority je přidáno za minutu je součástí konfigurace BE aplikace). Na jaký solver task je umístit je rozhodnuto podle priority solveru a počtu tasků už běžících na solveru.

- `publishTasks` - funkce, kterou volají solvery, pokud jsou po spuštěny nebo pokud k tomu dostanou příkaz. Argumentem této funkce je serializované pole funkcí, které solver nabízí.

- `getCommand` - základní funkce, kterou má solver zavolat, pokud je kontaktován na svém TCP/IP portu. Návratová hodnota funkce je příkaz, který má solver vykonat.

- `getTask` - poté,co solver dostane příkaz, aby si přečetl task ze serveru a začal ho řešit, použije tuto funkci.

- `setResults` - pomocí této funkce solver zapíše výsledky tasku do databáze.

Počítá se, že BE service bude součástí sítě privátních adres a nebude vystavena internetu. Proto je zabezpečena pouze omezením přístupu k service na základě IP adres.

**Solvery**

Solver zajišťuje funkcionalitu požadovanou tasky. Solver se skládá ze dvou částí: posluchače (listeneru), v tomto případě Windows system service, a spustitelného souboru nebo knihovny s funkcionalitou solveru. Detaily o umístění a konfiguraci jednotlivých souborů jsou k dispozici na přiloženém CD.

Listener je implementován jako Windows service (solver service) pro snadnější administraci a především zbavení se nutnosti, aby byl po celou dobu běhu solveru přihlášen uživatel, který solver spustil. Windows service mohou běžet na pozadí pomocí systémového účtu. Součástí solveru je klientská proxy třída (rozhraní) pro komunikaci s BE service. Při startu solver service začne listener poslouchat na definovaném portu a publikuje své funkce pomocí BE serveru, BE server pak rozhodne, zda má pro solver nějaké příkazy. Při zastavení solver service jsou zastaveny všechny řešené tasky a zastaveno naslouchání na definovaném portu.

Pokud je Listener kontaktován na svém portu, spustí nové "conf" vlákno, které zavolá BE funkci, getCommand, která vrátí příkaz pro solver. Možné příkazy jsou následující:

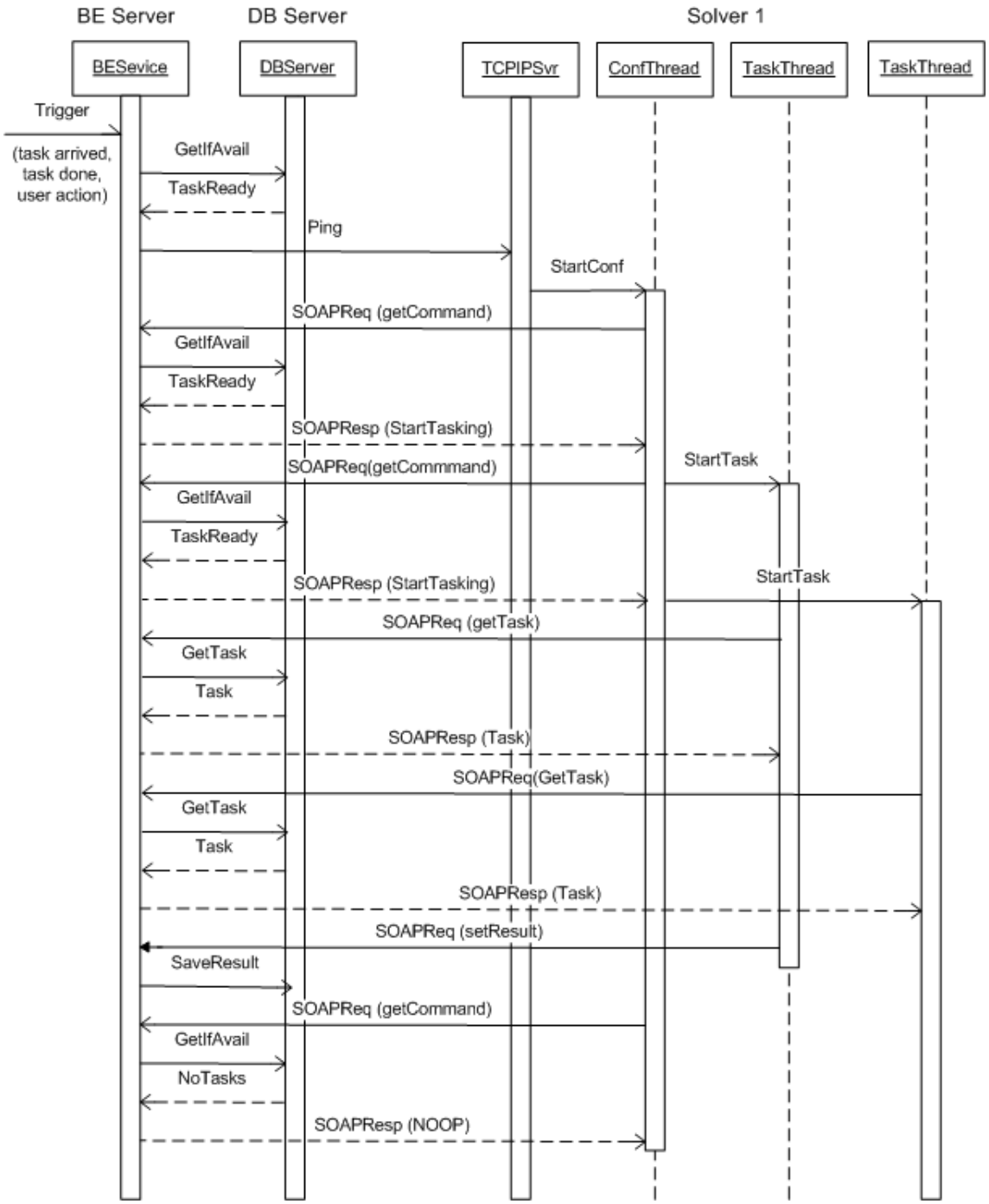
- `KILL` - zasataví service.
- `STOP` - zastaví vlákna s tasky.
- `NOOP` - žádné příkazy pro solver, zastaví "conf" vlákno.
- `TASK <ID>` - vytvoří nové vlákno s parametrem ID, které si přečte z databáze požadované zadání a na základě deserializování zadání zavolá příslušný spustitelný soubor nebo vnitřní funkci, která zadání vyřeší. Po skončení výpočtu zapíše vlákno výsledek na server v serializované podobě.
- `DELETE <ID>` - zastaví vlákno, které řeší task s daným ID

"Conf" vlákno volá funkci getCommand v cyklu tak dlouho, než obdrží jeden z příkazů KILL, STOP nebo NOOP, a poté čeká na nový kontakt Listeneru.

Druhou částí jsou spustiltelné soubory, knihovny nebo funkce přímo v listeneru, které řeší task. V rámci práce byly implementovány dva spustitelné soubory, které jako své parametry na příkazové řádce přijímají název funkce, která se má spustit a cestu k adresáři s uloženými vstupními argumenty. Implementované solvery mají sloužit hlavně jako demonstrace, jak by měly být solvery používány. Definice umístění a typu funkcí ve spustitelných souborech se děje v kořenovém adresáři solver service, v souboru "config.xml", kde je třeba každou funkci solveru definovat.

Tento způsob umožňuje se například připojit pomocí FTP k severu, kde jsou foldery, nahrát nový spustitelný soubor, upravit definici v "config.xml" a použít administrační rozhraní k zjištění nových funkcí. Případně se dá použít obyčejný telnet klient a připojit se na daný listener, což aktivuje "conf" vlákno, které prohledá novou konfiguraci a publikuje funkce stejně, jako by byl kontaktován z administračního rozhraní. Uživatel se tak nemusí přihlašovat na solverový server, aby překompiloval listener. Solver service loguje své veškeré aktivity do souboru log/¡datum¿.log, který je možné použít pro debugovací účely.

Následující diagram (3.7) znázorňuje postup BE a solveru při současném zadání dvou tasků pro jeden solver. BE server je kontaktován vnějším klientem, který signalizuje,že je třeba obnovit frontu tasků (Trigger). BE server na základě toho kontaktuje DB server s SQL dotazem, který zjistí, zda je potřeba kontaktovat některý solver kvůli změně stavu tasku, případně solveru, případně zadat tasky, čekající na vyřešení. Zde je zjištěno, že je třeba zadat tasky (GetIfAvail), proto je vybrán nejvhodnější solver a ten je kontaktován (Ping). TCP/IP listener vytvoří nové "conf" vlákno. Ten se dotáže BE serveru na příkaz (getCommand) a dostane odpověď, že vyřešit task (startTasking). Vytvoří nové vlákno (startTask) a vlákno se inicializaci dotáže BE serveru na zadaný task (getTask). Mezitím se "conf" vlákno dotáže znovu na příkaz a znovu dostane odpověď, aby vyřešil další task. Ten zadá obdobně jako předtím a teprve nyní dostane na getCommand odpověď NOOP, tedy, že už nejsou žádné příkazy pro daný solver. Až vlákna tasků dostanou výsledek, je výsledek zapsán na server (setResult).




Obr. 3.7: Schéma komunikace mezi BE a solverem při současném zadávání dvou tasků.

**Implementované solvery**

Solvery, implementované jako součást této práce, slouží hlavně pro příklad pro implementaci budoucích solverů, které budou poskytovat více užitečné funkce. Přesto jsou tyto dva solverové spustitelné soubory CalcSolver.exe (osahuje základní matematické operace, jako sčítání, odčítání, modulo, mocnění atd.) a WorkerSolver.exe (obsahuje funkce DoNothing, která nic nedělá, DoWork která uspí vlákno na zadaný počet milisekund, Echo, která vrátí beze změny textový řetězec, a negativePicture,

která vrátí negativ zadaného obrázku enkódovaného v Base64 textovém řetězci) plně funkční. Konfigurace umístění a atributů funkcí je uložena v souboru config.xml v kořenovém adresáři solver service.

**Vytváření nových solverů**

Vytváření nových solverů je možné buď na základě implementované solver service, popřípadě implementovaných spustitelných souborů nebo vytvořit zcela nový solver, který je schopen komunikovat s BE service.

- **Úprava spustitelných souborů** je nejjednodušší varianta. K vytvoření nebo modifikaci solverové funkce stačí úprava projektu WorkerSolver nebo CalcSolver, které jsou k dispozici na přiloženém CD, úprava config.xml, které je v kořenovém adresáři, a kontaktování solver service listeneru.
- **Úprava nebo zkopírování solver service** je o trochu složitější - je totiž třeba upravit cesty v solver service projektu a instalovat solver service jako windows service. To se možné po zkompilování projektu pomocí nástroje "instalutil.exe" ([31]), který je k dispozici jako součást .NET Framerork 2.0, konkrétně v
  `C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\installutil.exe`
- **Vytvoření zcela nového solveru** je asi nejnáročnější možnost. Je nutné dodržovat protokol komunikace s BE service naznačený v předchozích kapitolách.

### 3.2.4 Klienti

V rámci práce byli vytvořeni 2 klientské programy. Vzhledem k univerzálnosti systému mají sloužit spíše pro ukázku možností systému než k používání v budoucnu dodaných funkcí solverů. Hlavní důraz je kladen na automatickou generaci klientského kódu pomocí nástrojů vývojových prostředí.

**Generování nových klientů**

V této práci bylo jako hlavní vývojový nástroj použito Micorosoft Visual Studio 2008. To umožňuje přidávání referencí webových služeb pomocí položky "Add Service Reference..." kontextového menu projektu, do kterého se plánuje webové služby na dané adrese zakomponovat. Visual Studio po definování URL příslušné služby v dialogu stáhne všechny informace o službě do adresáře v kořenovém adresáři projektu a referencuje službu (endpointy, chování a navázání) v standardním nastavení projektu. V případě webových aplikací jsou metadata služeb ukládána do adresáře `App_WebReferences` a soubor s nastavením je `web.config` ([30]), v případě desktopových aplikací jsou metadata služby ukládána do adresáře "Service References" a soubor s nastavením aplikace je "app.config". Důležitý je právě soubor s nastavením aplikace, kde je definována reference služby. Některé parametry služeb, jako jsou například timeouty pro spojení se službou a omezení velikosti přenesených dat, nejsou při tomto způsobu generování služby převzaty z nastavení WS, proto je třeba je v případě potřeby změnit ručně v tomto konfiguračním souboru. Volání funkcí referencované služby probíhá pomocí Vytvoření instance takzvané service proxy, která

kromě metod pro práci s objektem proxy samotným obsahuje funkce, nabízené službou.

Další možností, jak zakomponovat do projektu zdrojové kódy vygenerované pomocí nástroje ServiceModel Metadata Utility Tool ("Svcutil.exe", [32]). Ten je dostupný jako součást Windows SDK, přesněji v adresáři `C:\Program Files\Microsoft SDKs\Windows\v6.0\Bin\`. Je možné specifikovat, jaký jazyk bude použit jako výstup (C# nebo VB). Pokud budou vygenerované třídy použity jako součást už existující aplikace, je nutné buď ručně zkopírovat nastavení služby vytvořené Svcutil nastavení aplikace nebo použít volby mergeConfig. Více informací je možno nalézt na citované webové stránce dokumentace nástroje.

### Picture Client

Picture Client je ukázkový klient pro zaslání obrázku na server, zjišťování stavu této úlohy, jejímu případnému vymazání a přečtení vráceného obrázku ze serveru. Klient pro práci s obrázky byl vybrán proto, aby se demonstrovala schopnost systému pracovat také s relativně větším množstvím přenášených dat.

Dále je zde ukázán přístup pro přihlášení k webové službě pomocí uživatelského jména a hesla a způsob, jak obejít kontrolu certifikátu pro neplatné nebo self-signed certifikáty.


Obr. 3.8: Snímek obrazovky Picture Clientu.

Picture client je aplikace Windows Forms, tvořená jedním oknem. Hlavní část okna zabírá prostor, kam se nahrávají obrázky, v pravé části jsou příkazová tlačítka. Seznam identifikátorů zadaných tasku. Chybové a stavové zprávy jsou vypisovány do stavového pruhu v dolní části okna.

Po nahrání obrázku z lokálního souborového systému (pomocí tlačítka "Load Image") je možné obrázek zadat na server ke zpracování. To se děje pomocí tlačítka "Submit img", které obrázek převede z binární reprezentace na textovou reprezentaci pomocí Base64Encodingu, a poté předá textový řetězec solverové funkci negativePicture, která požaduje jako parametr právě textový řetězec. Funkce negativePicture po spuštění na příslušném solveru zkonvertuje textový řetězec zpět do binární reprezentace, vytvoří obrázek, z něho vytvoří negativ, endkóduje obrázek do Base64Ecoded textového řetězce a vrátí ho jako návratovou hodnotu.

Každému nově zadané tasku je přidělen jednoznačný celočíselný identifikátor - task ID. Task ID je vráceno jako návratová hodnota při každém volání solverové funkce. To umožňuje se na jednotlivé tasky později odkazovat. Po zadání obrázku se přidělené task ID přidá do seznamu tasků v pravé části okna. Seznam je možno uložit do textového souboru pro pozdější použiti, aby se neztratila informace o task ID zadaných tasku - užitím "Save list", popřípadě takový textový soubor do seznamu zpět nahrát užitím "Load list". Při nahrávání se provede sloučení se záznamy, už existujícími v seznamu.

Kontrola stavu tasku se provádí aktivací tlačítka "Check ID". "Check ID" používá obecnou folderovou funkci getTaskStatus, které se jako parametr předává taks ID, proto je třeba mít označeno jedno z task ID v seznamu tasků. Podobně také "Delete ID" potřebuje mít vybranou položku ze seznamu. "Delete ID" využívá obecnou folderovou funkci setTaskProperty, pomocí které nastaví status tasku na "k vymazání". Akce "Get ID" se pokusí přečíst ze serveru výsledek tasku. Akce používá solverovou funkci, komplementární k zadávací funkci negativePicture - negativePictureResult, které se jako parametr předává task ID a návratová hodnota je schodná s návratovou hodnotou originální funkce na solveru. V tomto případě funkce negativePictureResult vrací textový řetězec. Všechny solverové návratové funkce jsou blokovací - čekají na to, až bude task dokončen. To může být problém v případě, že je solver dočasně zaneprázdněn. V případě, že se během čekání vyskytne error je čekání přerušeno, popřípadě může být čekání přerušeno také vypršením časového limitu žádosti (timeoutem) nastaveným pro konkrétní webovou službu.

Klient je vytvořen v jazyce C# pomocí Microsoft Visual Studia 2008.

### Webový klient

Webový klient je součástí Front-End serveru a nepoužívá ke svému chodu jen WS, ale i databáze systému, takže byl zmíněn v předchozí části, která se FE serverem zabývá.

# Kapitola 4

# Závěr

V této kapitole budou shrnuty výsledky práce.

V rámci práce byla navržena klient-server aplikace pro vzdálené zadávání úloh a implementována serverová a klientské aplikace.

Základní funkcionalita implementovaného systému spočívá v možnosti uživatelů jednoduše formulovat úlohy a bezpečně a spolehlivě je zadávat na vzdálený server, kde se úlohy zařadí do fronty úloh k řešení databázi. Tyto úlohy jsou postupně podle aktuální důležitosti úloh zadávány k vyřešení a po jejich vyřešení jsou výsledky předány uživateli stejnou cestou, jakou je zadal. Tento přístup umožňuje přesunout výpočetně náročné operace na vzdálený server, nezatěžovat tak klientský počítač během výpočtu a odbourává nutnost se přihlašovat na vzdálený server a zadávat ručně úlohy do cílové aplikace.

Zadávání optimalizačních úloh bylo zobecněno na zadávání všech typů úloh na vzdálený server. To je umožněno transparentní architekturou nezávislou na přenášených datech. Díky použití Web Services jako vstupního interface do serverové aplikace je docíleno kompatibility s velkým počtem nástrojů, které tak lze použít jako klienty, a jednoduše generovat nové klienty.

Serverová aplikace je tvořena ze čtyř na sobě nezávislých částí. První z nich je Front-End server, který umožňuje uživatelům volat funkce definované v serveru pomocí Web Services (Windows Communication Foundation implementace Web Services, v jazyce C#) a spravovat systém pomocí administračního rozhraní (napsaného v ASP.NET). V administračním rozhraní lze generovat nové Web Services podle uživatelských požadavků a přiřazovat různé funkce cílových aplikací různým Web Services. To vše je umístěno na webovém serveru (Internet Information Services server), který je jako jediná serverová část přístupná uživatelům.

Další částí je databáze, která slouží hlavně jako úložiště fronty úloh, kterou je možné dynamicky vyhodnocovat. Databáze také obsahuje nastavení aplikace. Databáze je instalována SQL Serveru 2005.

Třetí částí je Back-End server, který vyhodnocuje důležitost úloh ve frontě a na jejím základě předává úlohy k vyřešení cílovým aplikacím.

Poslední serverovou částí jsou cílové aplikace (solvery). Ty vyřeší danou úlohu a vrátí její výsledek Back-End serveru. Solvery mohou být umístěny na stejném počítači jako Back-End nebo na vzdálených počítačích.

V rámci práce byli implementováni dva klienti, webový klient (napsaný pomocí

ASP.NET) a ukázkový klient pro přenos obrázků (napsaný v jazyce C#). Hlavní důraz je ale kladen na možnost automatického generování klientů podle uživatelových potřeb, proto jsou popsány způsoby, jak toho relativně jednoduše docílit.

Přínos této práce je především ve snadno konfigurovatelném rozhraní pro tvoření a užívání Web Services založených na WCF pro formulaci úloh.

# Literatura


[1] Dalibor Kačmář, *Programujeme .NET aplikace ve Visual Studiu .NET*. Computer Press, 2001. ISBN 80–7226–569–5.

[2] Argonne National Laboratory, *Network Enabled Optimalization Server*. [online, 19.1.2009] `http://neos.mcs.anl.gov/neos/`

[3] Altair Engineering, *Open Portable Batch System*. [online, 19.1.2009] `http://www.pbsgridworks.com/PBSTemp1.3_2.aspx?top_nav_name=Products&item_name=OpenPBS&top_nav_str=1`

[4] Výpočetní a informační centrum ČVUT, *Centrum intenzivních výpočtů*. [online, 19.1.2009] `http://www.civ.cvut.cz/info/info.php?did=484`

[5] Cluster Resources Inc., *TORQUE Resource Manager*. [online, 19.1.2009] `http://www.clusterresources.com/pages/products/torque-resource-manager.php`

[6] Tivoli Software, *Tivoli Workload Scheduler*. [online, 19.1.2009] `http://www-01.ibm.com/software/tivoli/products/scheduler/`

[7] Software- und Organisations-Service GmbH, *Open Source Job Scheduler*. 2008. [online, 19.1.2009] `http://jobscheduler.sourceforge.net`

[8] Sun Microsystems, *Sun Grid Engine*. [online, 19.1.2009] `http://gridengine.sunsource.net/`

[9] David Booth a spol., *Web Services Architecture W3C Recommandation*. 2004. [online, 19.1.2009] `http://www.w3.org/TR/ws-arch/`

[10] W3C, *World Wide Web Consortium*. [online, 19.1.2009] `http://www.w3.org/`

[11] Kelvin Lawrence a spol., *OASIS WS-Security Technical Committee*. 2006. [online, 19.1.2009] `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss`

[12] Kazunori Iwasa a spol., *OASIS WS-Reliable Messaging Technical Committee*. 2004. [online, 19.1.2009] `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrm`

[13] ISO, *ISO 9075, Structured Query Language*. 1992. [online, 19.1.2009] `http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt`

[14] Microsoft Corporation, *Open DataBase Connectivity overview.* [online, 19.1.2009] `http://support.microsoft.com/kb/110093`

[15] Microsoft Corporation, *Internet Information Services.* [online, 19.1.2009] `http://www.iis.net/`

[16] Tim Bray a spol., *Extensible Markup Language (XML) 1.1 W3C Recommandation.* 2006. [online, 19.1.2009] `http://www.w3.org/XML/`

[17] Yves Lafon a spol., *SOAP Version 1.2 W3C Recommandation.* 2007. [online, 19.1.2009] `http://www.w3.org/TR/2007/REC-soap12-part0-20070427/`

[18] Dave Marshall, *Tutoriál na RPC.* 1999. [online, 19.1.2009] `http://www.cs.cf.ac.uk/Dave/C/node33.html`

[19] UserLand Software, Inc., *Archivovaná domácí stránka XML-RPC projektu.* 2003. [online, 19.1.2009] `http://web.archive.org/web/20070609094314/www.xmlrpc.com/`

[20] Object Management Group, Inc., *CORBA Component Model, v4.0.* [online, 19.1.2009] `http://www.omg.org/technology/documents/formal/components.htm`

[21] Microsoft Corporation, *.NET Framework Developer Center.* [online, 19.1.2009] `http://msdn.microsoft.com/cs-cz/netframework/default(en-us).aspx`

[22] Ing. Richarda Šusty, PhD., *Přednášky z předmětu X35PJR.* 2008. [online, 19.1.2009] `http://dce.felk.cvut.cz/pjr/prednasky/plan.htm`

[23] Erik Christensen, *Web Services Description Language (WSDL) 1.1.* 2006. [online, 19.1.2009] `http://www.w3.org/TR/wsdl`

[24] Keith Ballinger a spol., *Web Services Metadata Exchange (WS-MetadataExchange) for Service Endpoints.* 2006. [online, 19.1.2009] `http://xml.coverpages.org/ni2004-03-05-a.html`

[25] Microsoft Corporation, *What Is Windows Communication Foudation?.* [online, 19.1.2009] `http://msdn.microsoft.com/en-us/library/ms731082.aspx`

[26] David Totzke, *Project Tango - WCF And Java Interop.* 2006. [online, 19.1.2009] `http://www.infoq.com/news/Project-Tango-WCF-Java-Interop`

[27] The Apache Software Foundation, *JAX-WS Guide.* [online, 19.1.2009] `http://ws.apache.org/axis2/1_4_1/jaxws-guide.html`

[28] The Apache Software Foundation, *JAX-WS Guide, Developing a JAX-WS Web service from a WSDL document.* [online, 19.1.2009] `http://ws.apache.org/axis2/1_4_1/jaxws-guide.html#TopDownService`

[29] Microsoft Corporation, *What's New in Web Services Enhancements (WSE) 3.0.* 2005. [online, 19.1.2009] `http://msdn.microsoft.com/en-us/library/ms977317.aspx`

[30] Microsoft Corporation, *Configuring Services Using Configuration Files*. [online, 19.1.2009] `http://msdn.microsoft.com/en-us/library/ms733932.aspx`

[31] Microsoft Corporation, *Installer Tool (Installutil.exe)*. [online, 19.1.2009] `http://msdn.microsoft.com/en-us/library/50614e95(VS.80).aspx`

[32] Microsoft Corporation, *ServiceModel Metadata Utility Tool (Svcutil.exe)*. [online, 19.1.2009] `http://msdn.microsoft.com/en-us/library/aa347733.aspx`

# Příloha A

# Obsah CD

Přiložené CD obsahuje následující soubory a adresáře:

- apps/ - adresář se všemi aplikacemi implementovanými v rámci práce,
  - Lib/ - knihovna, která osahuje implementaci providerů,
  - PictureClient/ - ukázkový WCF klient,
  - Solver/ - adresář se aplikacemi souvisejícími se solverem,
    - CalcSolver/ - ukázkový solver s matematickými funkcemi,
    - SolverService/ - windows service pro příjem tasků,
    - WorkerSolver/ - ukázkový solver se textovými a bitmapovými funkcemi,
  - www/ - adresář obsahuje část systému přístupnou přes webové rozhraní,
    - BEService/ - Back-End aplikace pro předávání tasků solverům,
    - ws/ - Front-End aplikace s foldery a webovým rozhraním,
- scripts/ - adresář s užitečnými skripty,
- DP-Jan_Ramba-2009.pdf - text diplomové práce,
- info.txt - soubor s informacemi o přihlašovacích údajích a umístění jednotlivých aplikací.