VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

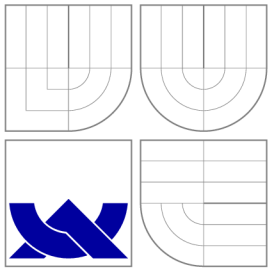
METODY VZDÁLENÉHO PŘÍSTUPU KE SLUŽBÁM INFORMAČNÍCH SYSTÉMŮ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

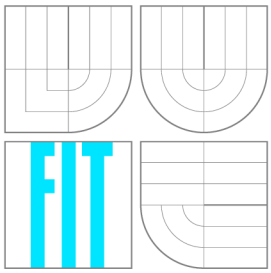
AUTOR PRÁCE
AUTHOR

MIROSLAV OUJESKÝ

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

METODY VZDÁLENÉHO PŘÍSTUPU KE SLUŽBÁM INFORMAČNÍCH SYSTÉMŮ

METHODS OF REMOTE ACCESS TO INFORMATION SYSTEMS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MIROSLAV OUJESKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

prof. Ing. JAN M. HONZÍK, CSc.

BRNO 2009

Abstrakt

Tato práce popisuje současné metody pro vzdálený přístup ke službám informačních sytémů. Dále popisuje návrh konkrétního rozhraní pro vzdálený přístup k informačnímu systému Webnode.

Abstract

This work describes currently available methods for remote access to information systems. This knowledge was used to design and implement remote access interface to Webnode information system.

Klíčová slova

webové služby, vzdálené volání procedur, Webnode, XML, XML-RPC, SOAP, REST, PHP

Keywords

web services, remote procedure call, Webnode, XML, XML-RPC, SOAP, REST, PHP

Citace

Miroslav Oujeský: Metody vzdáleného přístupu ke službám
informačních systémů, bakalářská práce, Brno, FIT VUT v Brně, 2009

Metody vzdáleného přístupu ke službám informačních systémů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana prof. Ing. Jana M. Honzíka, CSc. Další informace mi poskytl pan Ing. Lubomír Cvrk, Ph.D. (Westcom, s.r.o.). Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Miroslav Oujeský
19. května 2009

Poděkování

Tímto bych chtěl poděkovat vedoucímu své práce, panu prof. Ing. Janu M. Honzíkovi, CSc., a odbornému konzultantovi mé práce, panu Ing. Lubomíru Cvrkovi, Ph.D., za pomoc a veškerý čas, který mi věnovali.

Obsah

Úvod

Jedním z nejvýznamnějších trendů informačních technologií dnešní doby jsou webové informační systémy. Umožňují uživateli přístup téměř odkudkoliv, navíc mnoho systémů obsahuje rozhraní pro vzdálený přístup ke svým službám z jiných informačních systémů. Je tak například umožněno sdílení zajímavého obsahu v komunitních sítích nebo třeba získávání burzovních informací z různých zdrojů a mnoho dalších možností.

Zástupcem moderních informačních systémů dnešní doby je i systém *Webnode*. Pomocí něj dokáže i laik vytvořit a spravovat dobře vypadající webovou prezentaci. Projekty v systému *Webnode* jsou dostupné zdarma, placené jsou až nadstandardní balíčky (jako je větší prostor pro soubory atd.). Vzhledem k nárůstu uživatelské základny a celkovému renomé tohoto systému narostlo také množství zájemců o partnerský program, který partnerům umožňuje provozovat systém *Webnode* pod vlastní hlavičkou. Tím také vyvstal požadavek na snadné vytváření a správu projektů za použití vlastní infrastruktury.

Ideální řešení tohoto požadavku by mělo být snadno a levně nasaditelné, a to jak ze strany poskytovatele služeb, tak i ze strany klienta. Také by mělo být robustní a bezpečné, a bez velkých nároků na výpočetní sílu.

V první kapitole této práce naleznete krátký úvod do dnes používaných technologií pro výměnu informací mezi informačními systémy.

Druhá kapitola stručně popisuje systém *Webnode* a je zde do hloubky popsána zvolená metoda komunikace a její zabezpečení.

Ve třetí kapitole je popsán návrh rozhraní a konečně ve čtvrté kapitole je popsán způsob implementace rozhraní a příklady jeho použití.

Kapitola 1

Služby informačních systémů

Služby informačních systémů jsou v této práci chápány především jako webové služby (web services). Podle organizace W3C[1] lze webovou službu definovat jako softwarový systém, navržený tak, aby umožňoval strojovou výměnu informací přes počítačovou síť[5]. Jedná se tedy o zpřístupnění funkcionality informačního systému pro externí softwarové systémy. Může se jednat jak o získávání dat z informačního systému, tak i o řízení informačního systému.

1.1 Co jsou to služby informačních systémů

Webové služby je možné využít v informačních systémech různého zaměření. Pro obecnou představu je možné uvést například systémy ve finančním sektoru, které automaticky získávají burzovní informace, směnné kurzy měn, atd. nebo systémy zpravodajské, kdy je možné automaticky přejímat zprávy z různých zdrojů. Rozšířené jsou také systémy pro práci s geografickými daty (mapové systémy), které umožňují například vyhledat sídlo firmy z katalogu firem a zobrazit jej na mapě.

Dále je umožněn vznik desktop aplikací které spolupracují s webovými informačními systémy - například desktop blogovací nástroje, již zmiňované finanční systémy atd.

Samostatnou skupinou jsou stále oblíbenější sociální sítě (např. Facebook, MySpace, Last.fm, Youtube atd.). Tyto systémy umožňují svým uživatelům komunikaci a sdílení zajímavých informací navzájem mezi sebou, anonymně i mezi přáteli. Může se jednat jak o sdílení informací o daném uživateli – co zrovna dělá, co rád nakupuje, jaký film se mu líbil, jakou hudbu poslouchá, apod. – nebo třeba sdílení zajímavých odkazů na články nebo obecně jakékoliv www stránky. Velké sociální sítě zpravidla umožňují sdílení obsahu mezi sebou navzájem a tím je možné docílit například situace, kdy uživatel ve svém profilu v internetové seznamce zobrazuje data ze sítě pro sdílení hudebních preferencí, a tato data jsou automaticky aktualizována pomocí webových služeb. Sociálních sítí využívjí pravidelně miliony uživatelů na celém světě a jsou vyhledávaným médiem pro internetovou reklamu, vzhledem k relativně snadnému cílení reklamy.

Poskytovatelé služeb (provozovatelé informačního systému) dávají obvykle k dispozici podrobnou dokumentaci dané služby, která popisuje metodu vzdáleného přístupu ke službě a formát dat služby. Poskytovatelé větších služeb často zpřístupňují i hotové řešení pro přístup ke službě v některém z obvyklých jazyků používaných pro implementaci webových

[1] World Wide Web Consortium

informačních systémů (PHP[2], Ruby, ASP[3], apod.).

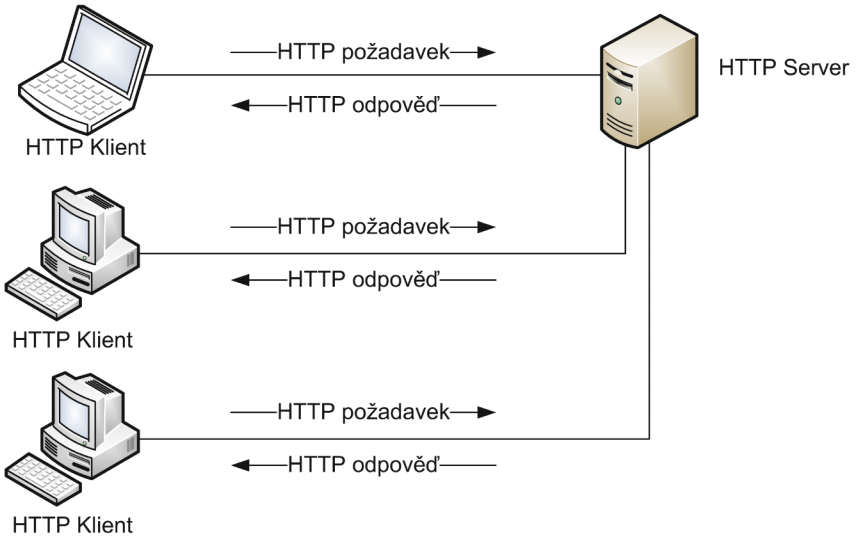
1.2 Principy vzdáleného přístupu ke službám IS

Služby informačních systémů jsou nenáročné na provoz, protože využívají HTTP protokol a s ním spojenou již existující infrastrukturu. Není tedy nutné žádné speciální hardwarové nebo softwarové vybavení. Pro předávání dat z webové služby je obvykle využíván jazyk XML, který je standardizovaný, otevřený a široce podporovaný a používaný. Tyto vlastnosti zaručují nezávislost na platformě a konkrétním programovacím jazyce. Klient služby nemusí být nutně programován ve stejném jazyce jako služba samotná.

Nejrozšířenější metody přístupu k webovým službám budou podrobněji popsány níže. Generování XML souborů je zpravidla implementováno ve stejném jazyce, ve kterém je implementován informační systém, který službu poskyzuje. Pokud je při návrhu informačního systému použita metodika MVC (model-view-controller) nebo jiný vhodný návrhový vzor, pak je možné využít již existující části kódu pro přístup k datům, které se využívají pro vygenerování výstupu koncovému uživateli.

1.2.1 HTTP

Protokol HTTP – Hypertext Transfer Protocol – je jedním ze stavebních kamenů dnešního internetu. Používá architekturu klient-server. HTTP klient se přes počítačovou síť spojí s HTTP serverem a zašle mu požadavek na dokument. Server požadavek zpracuje a vrátí klientovi odpověď[8]. Princip fungování protokolu HTTP je naznačen na obrázku (1.1).



Obrázek 1.1: HTTP protokol

HTTP je aplikačním protokolem, který pro vlastní přenos dat mezi počítači využívá protokol nižších vrstev, převážně protokol TCP[4]. Klient se obvykle k HTTP serveru připojuje na port 80.

[2]PHP Hypertext Preprocessor
[3]Active Server Pages
[4]Transmission Control Protocol

HTTP je bezstavový protokol – to znamená, že mezi jednotlivými požadavky klienta si server nemůže předat žádné informace. Pro obcházení tohoto nedostatku se využívá například mechanismu cookies nebo session. Cookies jsou soubory uložené u klienta, jejichž obsah je odesílán v hlavičce HTTP požadavku, jejich případná nová hodnota je předávána klientovi v hlavičce HTTP odpovědi. Session jsou data uložená na serveru, jednotlivým klientům se přiřadí na základě unikátního klíče, který serveru předají.

Dokumenty zpřístupněné HTTP serverem jsou adresovány pomocí URL – Uniform Resource Locator. Jedná se o řetězec znaků v definovaném formátu, jednoznačně identifikující umístění zdroje informací v rámci sítě Internet[14]. URL pro dokument přístupný pomocí protokolu HTTP může vypadat například následovně:

`http://uzivatel:heslo@domena:port/cesta/k/dokumentu.html`.

První část `http` se nazývá schéma URL a označuje použitý protokol (v tomto případě HTTP). Část `uzivatel` a `heslo` je nepovinná a používá se pokud je HTTP protokolem vyžadován přístup pod heslem (HTTP autentizace). Část `domena` určuje doménové jméno HTTP serveru, místo doménového jména může být uvedena i IP adresa. V nepovinné části `port` lze uvést TCP port, na kterém HTTP server naslouchá (pokud není uveden, uvažuje se jako výchozí port 80). Závěrečná část `/cesta/k/dokumentu.html` označuje cestu k dokumentu uloženého na HTTP serveru.

Dokumenty zpřístupněné HTTP serverem mohou být statické soubory (HTML, obrázky, atd.), je možné ale HTTP server nakonfigurovat, aby byly požadavky předány ke zpracování skriptu ve vhodném programovacím jazyce, který dokument vygeneruje dynamicky. Pro předání vstupů do takového skriptu slouží parametry URL, které vypadají například následovně:

`http://server/skript?param=hodnota&param2=hodnota`.

Protokol HTTP sám o sobě neposkytuje žádné možnosti zabezpečení proti odposlechu komunikace, existuje však jeho nadstavba HTTPS – Hypertext Transfer Protocol Secure[11], která vyžívá šifrovaný přenos pomocí SSL[5] nebo TLS[6].

HTTP požadavek

Obecný formát HTTP požadavku vypadá následovně:

```
<metoda> <URL dokumentu> <verze HTTP>
<HTTP hlavičky>
<prázdný řádek>
```

Nejpoužívanější metodou je metoda GET, která slouží k získání obsahu dokumentu podle zadaného URL. Metoda HEAD funguje podobně jako metoda GET, ale vrací jen hlavičky odpovědi, ne tělo požadovaného dokumentu. Metoda POST slouží k odesílání většího množství dat na server (například odeslání obsahu formuláře), tato data jsou vložena do požadavku za prázdný řádek. Dalšími metodami jsou PUT a DELETE, které slouží k vložení souboru na dané URL respektive smazání souboru na daném URL. V praxi se tyto metody nepoužívají.

V dnešní době je nejrozšířenější verzí protokolu HTTP verze 1.1. Z důvodu zachování kompatibility však většina klientů i serverů podporuje i starší verzi 1.0.

Hlavičky HTTP požadavku se zapisují ve formátu "`Hlavicka: hodnota`" a navzájem jsou oddělené znakem pro nový řádek. V hlavičkách je možné upřesnit různé parametry

[5] Secure Socket Layer
[6] Transport Layer Security

spojení – identifikace klienta (`User-Agent`), odkud se klient na tento dokument dostal (`Referer`), preferovaný jazyk požadovaného dokument (`Accept-Language`), hodnoty cookies (`Cookie`) a jiné.

HTTP odpověď

Obecný formát HTTP odpovědi má následující tvar:

```
<verze HTTP> <stavový kód> <textová reprezentace stavu>
<HTTP hlavičky>
<prázdný řádek>
<tělo odpovědi>
```

Stavový kód odpovědi dává klientovi informaci o výsledku požadavku. Tento kód je tříciferné číslo, kde první číslice určuje kategorii odpovědi. Kódy `1xx` jsou informační kódy, `2xx` označují úspěšné vyřízení požadavku, `3xx` označují přesměrování, `4xx` chyby na straně klienta a `5xx` chyby na straně serveru. Stavový kód je doprovázen svou textovou reprezentací[6].

Nejčastějšími odpovědmi jsou:

- `200 OK` – požadavek byl úspěšně zpracován.
- `301 Moved Permanently` – dokument byl přemístěn na jiné URL.
- `403 Forbidden` – požadavek je v pořádku, ale server nemá povoleno jej vykonat.
- `404 Not Found` – požadovaný dokument nebyl na serveru nalezen.
- `500 Internal Server Error` – na serveru se vyskytla chyba a požadavek nemohl být zpracován.

Hlavičky odpovědi mají stejný formát jako hlavičky požadavku. Mezi důležité hlavičky patří `Location`, která ve spojení se stavovým kódem `3xx` sděluje klientovi URL, na které byl požadovaný dokument přesunut.

V těle odpovědi je v závislosti na metodě požadavku obvykle obsah požadovaného dokumentu nebo může být i prázdné.

1.2.2 XML

Jazyk XML – Extensible Markup Language – je univerzálním jazykem určeným pro výměnu a ukládání strukturovaných dat. Vychází z jazyka SGML[7] a obsahuje i některé vlastnosti jazyka HTML[8]. Vytváří jednoduchý a zároveň i velmi účinný mechanismus pro ukládání, zpracování a šíření informací[1].

Zápis v jazyce XML sestává ze značek (tagů), které označují jednotlivé elementy XML dokumentu. Tyto značky jsou zapisovány jako `<element>` pro otevírací značku elementu a `</element>` pro uzavírací značku elementu. Vše co se nachází mezi otevírací a uzavírací značkou je hodnota elementu. Pro elementy, které mají prázdnou hodnotu je možné použít unární zápis `<element />`.

[7] Standard Generalized Markup Language
[8] Hypertext Markup Language

Hodnotou elementu může být text, jiný element či elementy nebo kombinace elementů a textu. Elementy dále mohou mít definovány atributy, které mohou nést doplňující informace o elementu v podobě textového řetězce. Atributy se zapisují do otevírací značky elementu jako `<element atribut="hodnota atributu">` a musí být uvozeny buď v uvozovkách nebo apostrofech.

Název elementu může obsahovat písmena, čísla i ostatní znaky, nesmí však začínat číslem nebo interpunkčním znaménkem nebo obsahovat mezery.

Existují dvě úrovně správnosti XML dokumentu. První úroveň je označována jako *well-formed* dokument. Aby byl dokument *well-formed*, musí splňovat všechna syntaktická pravidla XML, navíc smí obsahovat pouze jeden element nejvyšší úrovně (kořenový element). Před tímto elementem může být volitelně umístěna hlavička XML dokumentu, která uvádí verzi jazyka XML a případně i znakovou sadu použitou v dokumentu. Tato hlavička má tvar `<?xml version="1.0" encoding="UTF-8" ?>`.

Druhou úrovní je validní dokument. K validaci dokumentu je vyžadováno schéma, které definuje z jakých elementů může daný dokument sestávat. Pro zápis schématu se obvykle používá formát DTD[9] nebo novější XSD[10]. Validní dokument musí být *well-formed* a odpovídat pravidlům definovaných v přiřazeném schématu.

Příklad *well-formed* XML dokumentu

```
<?xml version="1.0" encoding="UTF-8" ?>
<korenovyElement>
    <!-- komentar -->
    <element atribut="hodnota atributu">Hodnota elementu</element>
    <element atribut="jina hodnota">Dalsi hodnota</element>
    <element>
        <subElement>Hodnota subelementu</subElement>
    </element>
    <prazdnyElement />
</korenovyElement>
```

1.3 Současné metody vzdáleného přístupu

Většina v současnosti používaných protokolů pro vzdálený přístup ke službám informačních systémů využívá pro přenos strukturovaných dat mezi službou a jejím klientem jazyk XML[13].

V praxi se ale stále častěji můžeme setkat i s jinými formáty výměny dat jako je například JSON[11], který umožňuje snazší přístup k datům pomocí JavaScriptu, což má význam hlavně u AJAX[12] webových aplikací.

Samotný přenos probíhá pomocí HTTP protokolu za využití jeho standardních metod GET a POST. Ve většině případů tímto odpadají problémy s firewally nebo proxy servery. Je možné využívat i jiné transportní protokoly (SMTP, FTP, Jabber, atd.), ačkoliv to není příliš běžné.

[9] Document Type Definition
[10] XML Schema Definition
[11] JavaScript Object Notation
[12] Asynchronous JavaScript and XML

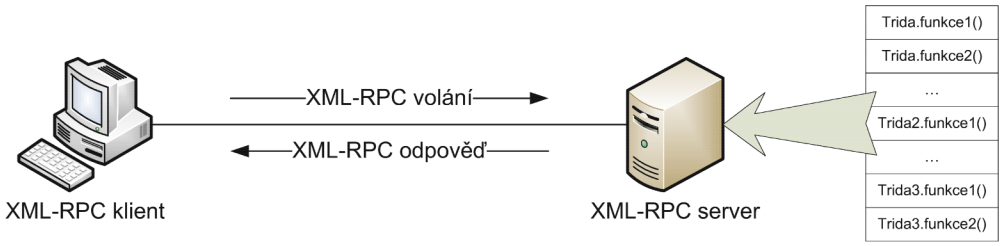
Pro následující nejpoužívanější protokoly pro vzdálený přístup ke službám informačních systémů existují knihovny pro většinu používaných relevantních programovacích jazyků, není tedy pro tvůrce služeb ani pro klienty služeb nutné studovat dopodrobna specifikace těchto protokolů.

1.3.1 XML-RPC

Protokol XML-RPC je jednoduchý RPC (remote procedure call - vzdálené volání procedur) protokol který využívá XML pro předávání parametrů volání vzdálené procedury a HTTP pro samotný přenos požadavku i výsledku volání procedury[15]. Byl vytvořen v roce 1998 panem D. Winerem. Později z něj byl vyvinut protokol SOAP (viz níže).

XML-RPC definuje pouze malé množství datových typů a příkazů, proto je snadné jej používat. Pro programátory může být tento způsob komunikace jednodušší pochopit, protože se jen mírně liší od klasického volání funkcí a procedur[2].

Na straně služby je zveřejněna sada funkcí nebo procedur, které jsou dostupné pomocí protokolu XML-RPC (viz obrázek 1.2). Po přijetí požadavku na volání procedury je tento požadavek zpracován a přijaté parametry jsou předány přímo navázané funkci nebo proceduře. Její návratová hodnota je poté vložena do XML obálky a pomocí HTTP protokolu vrácena klientovi.



Obrázek 1.2: XML-RPC protokol

XML-RPC využívá následující datové typy pro parametry i návratové hodnoty:

- **array** - neasociativní pole.
- **base64** - data v base64 kódování.
- **boolean** - logická hodnota (0 nebo 1).
- **dateTime** - datum a čas ve formátu ISO 8601.
- **double** - desetinné číslo.
- **int** - celé číslo.
- **string** - textový řetězec.
- **struct** - asociativní pole.

Příklad volání vzdálené procedury

```
<?xml version="1.0"?>
<methodCall>
  <methodName>examples.echo</methodName>
  <params>
    <param>
        <value><string>Hello world!</string></value>
    </param>
  </params>
</methodCall>
```

Příklad odpovědi vzdálené procedury

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
        <value><string>Hello world!</string></value>
    </param>
  </params>
</methodResponse>
```

1.3.2 SOAP

SOAP (Simple Object Access Protocol) je protokol určený pro výměnu strukturovaných dat služeb informačních systémů[9]. SOAP je pokračovatelem XML-RPC, ale je oproti němu složitější. SOAP protokol je W3C standardem a je podporován mnohými velkými společnostmi, jako je například Microsoft. SOAP tvoří jeden ze základních kamenů sady protokolů pro webové služby – poskytuje komunikační mechanismus k propojení webových služeb [10].

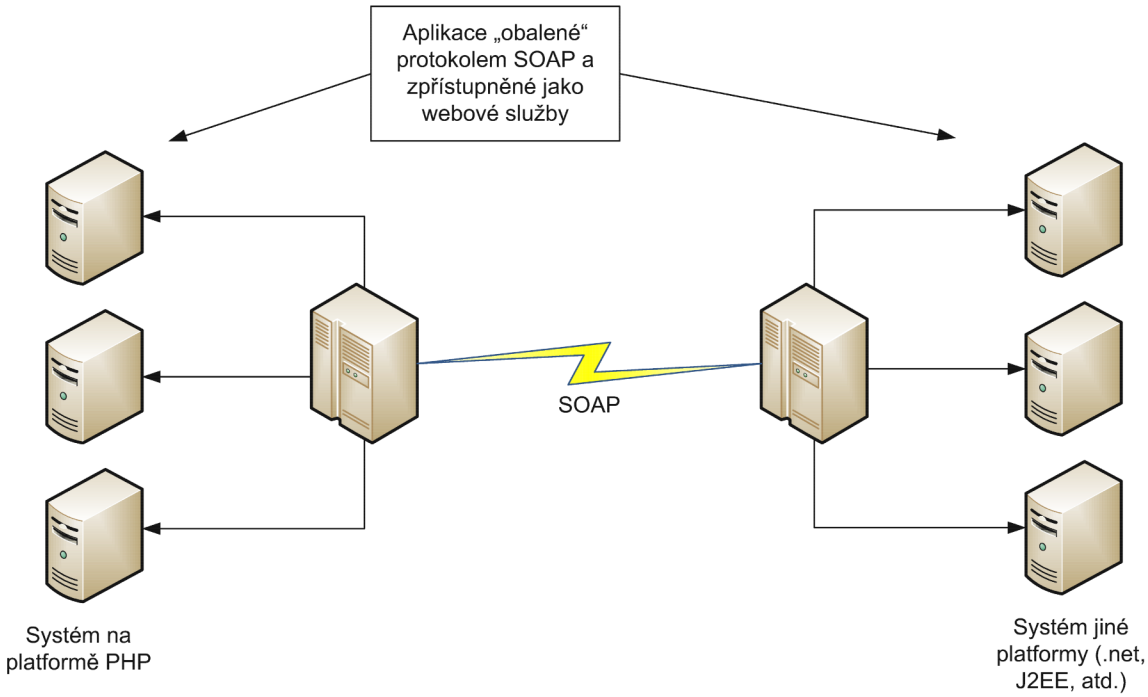
Specifikace SOAP definuje systém pro předávání dat v XML formátu přes internet. Je definován jako nezávislý na operačním systému nebo programovacím jazyce. Je zaměřen na poskytnutí základních možností pro přenos dat, na jejichž základě je možné postavit složitější protokoly. Pomocí protokolu SOAP je možné propojit již existující systémy na různých platformách (naznačeno na obrázku 1.3).

SOAP obaluje zprávy do obálek. Obálka (element `Envelope`) sestává z hlavičky (element `Header`) a těla zprávy (element `Body`). Hlavička není povinná a může obsahovat identifikaci zprávy nebo další řídíci informace, které se přímo netýkají dat, která jsou uložena v těle zprávy[7]. V případě chyby je v těle zprávy obsažen element `Fault`, který obsahuje informace o chybě.

SOAP je možné využít pro vzdálené volání procedur, není to ale podmínkou, protokol je univerzální. Oproti XML-RPC neobsahuje datové typy a nedefinuje pevný formát pro RPC. Pomocí SOAP je také možné přenášet celé XML dokumenty.

Příklad SOAP obálky

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    ...
  </env:Header>
  <env:Body>
    ...
  </env:Body>
</env:Envelope>
```



Obrázek 1.3: Propojení informačních systémů pomocí protokolu SOAP

1.3.3 REST

REST (Representational state transfer) není protokol sám o sobě, jedná se spíše o sadu principů návrhu softwaru[3], který se v poslední době stále častěji využívá při návrhu služeb informačních systémů.

Webovou službu využívající principy REST (anglicky RESTful service) je možné pojmout jako sadu zdrojů[12]. Definice takovéto služby sestává ze tří částí:

- URI služby (např. `http://domena.cz/zdroje/auta`).
- MIME typ dat získaných ze služby (například XML).

- Sada operací, které služba podporuje za využití standardních HTTP metod (GET, POST)[13].

Jednotlivé zdroje jsou adresovány za pomoci unikátního identifikátoru zdroje (např. `http://domena.cz/zdroje/auta/skoda-120`).

Operace nad celou sadou mohou být definovány následovně:

- metoda GET - seznam prvků sady.
- metoda POST - vytvoření nového prvku.

Operace nad jednotlivými zdroji pak mohou být definovány následovně:

- metoda GET - získání konkrétního prvku sady.
- metoda POST - modifikace konkrétního prvku sady.

[13] Pro RESTful služby jsou využity i metody PUT a DELETE, ty ale nebývají obvykle na běžných HTTP serverech dostupné

Kapitola 2

Vzdálený přístup ke službám systému *Webnode*

2.1 Systém *Webnode*

Systém *Webnode*[1] je online nástroj pro tvorbu webových prezentací, který umožňuje i laikovi vytvořit si profesionálně vypadající webovou prezentaci během krátké doby. Uživatel si může pomocí přehledného rozhraní (viz obrázek 2.1) metodou drag&drop[2] editovat obsah jednotlivých stránek své prezentace, přidávat do nich články, ankety, katalogy, diskuze atd. K dispozici je přes 40 vzhledů stránek, pokročilejší uživatel si může vytvořit vzhled vlastní. Systém také automaticky zaznamenává statistiky přístupů k prezentaci (viz obrázek 2.2).

Obrázek 2.1: Webnode Builder – editor webové prezentace

Základní verze systému je k dispozici zdarma. Pokud by uživateli nestačil diskový prostor

[1] `http://www.webnode.com`

[2] Přetahování částí obsahu pomocí myši – doslova "táhni a pusť"

nebo měsíční přenosy dat, případně pokud by měl zájem o některé rozšířené možnosti jako je vícejazyčná prezentace, emailové schránky nebo více uživatelů, pak si může pořídit placený rozšiřující balíček.

Na jaře roku 2009 používalo systém *Webnode* přes 200 tisíc uživatelů. Získal již také několik prestižních ocenění, například na mezinárodní konferenci LeWeb'08, která proběhla v prosinci 2008 v Paříži. Tyto úspěchy zvýšily zájem o partnerství s *Webnode*.

Obrázek 2.2: Webnode Statistics – statistiky návštěvnosti prezentace

Partneři mohou přeprodávat systém *Webnode* pod svou vlastní hlavičkou. Je ovšem nutné partnerům umožnit snadnou správu svých projektů v systému *Webnode* z vlastní infrastruktury. Stejně tak je nutné umožnit partnerům jednoduchým způsobem přístup k statistickým údajům o svých projektech (návštěvnost, přenesená data, atd.).

Docílit tohoto je možné navržením a implementací aplikačního rozhraní (API) systému Webnode pro správu projektů (vytvoření nového projektu, zablokování nebo smazání projektu) a pro získávání statistických údajů o projektech. Toto API bude možné snadno rozšířit o další funkcionalitu.

2.2 Zvolená metoda pro vzdálený přístup

Pro systém *Webnode* byla zvolena kombinace REST a RPC přístupu s odpověďmi ve formátu XML.

Požadavek na volání funkce nebude sám o sobě obsahovat žádné XML, bude ale využito parametrů HTTP požadavku. Tím odpadá režie nutná pro zpracování XML požadavků a při vyšším počtu požadavků takto výrazně klesne zatížení serverů na kterých je systém *Webnode* provozován.

Pro programátora klientské aplikace je tento způsob komunikace intuitivnější (teoreticky) a také jednodušší na implementaci. Není nutné vytvářet XML pro požadavek, stačí vytvořit URL s parametry. Je také možné tento způsob komunikace snadno ladit pomocí

volně dostupných nástrojů.

Na straně systému *Webnode* bude rozhraní vytvořeno modulárně, tedy tak, aby bylo v případě potřeby možné, bez nutnosti zásahů do jádra rozhraní, rozšířit toto rozhraní o další metody vzdáleného přístupu (XML-RPC, SOAP). Rozhraní bude implementováno v jazyce PHP, stejně jako většina systému *Webnode*.

Na základě požadavku na volání jedné z funkcí je klientovi vrácena odpověd v XML. Pokud došlo při zpracování požadavku k chybě (ať už ze strany klienta nebo systému *Webnode*), pak je v odpovědi vrácen kód a popis chyby. Pokud vše proběhlo v pořádku, pak je v odpovědi vráceno potvrzení, že operace proběhla v pořádku a případně data která si klient vyžádal.

2.3 Zabezpečení přístupu

Některé ze zpřístupněných funkcí vyžadují určitou úroveň ověření uživatele, který k nim přistupuje. K ověření slouží dvojice klíčů, které jsou pro každého uživatele unikátní. Tyto klíče vydává na požádání provozovatel systému *Webnode*.

Prvním klíčem je *API klíč* sloužící k autorizaci požadavku – identifikaci uživatele, který vytvořil požadavek. Toto je nutné jak pro řízení přístupu, tak i pro vytváření statistik o použití rozhraní.

Druhým klíčem je *podepisovací klíč*. Tento klíč je tajný a uživatel by jej neměl nikde zveřejňovat. Slouží k autentizaci požadavku – zda požadavek přichází skutečně od uživatele identifikovaného daným API klíčem. Za pomoci podepisovacího klíče se vytvoří unikátní signatura požadavku, která se stejným klíčem na straně systému *Webnode* ověří a tím se požadavek autentizuje.

Pro zvýšení bezpečnosti je možné pro přístup k rozhraní použít protokol HTTPS.

Kapitola 3

Návrh rozhraní

Základní myšlenkou rozhraní je přijetí požadavku, jeho zpracování a následné odeslání odpovědi. Rozhraní bude v budoucnu možné rozšířit o další formáty požadavků a odpovědí, proto je vhodné využít objektově orientovaného návrhu a dědičnosti.

Po příchodu požadavku je za pomoci třídy `WebnodeApiRequestFactory`, která implementuje návrhový vzor *Factory*[4], vytvořen objekt třídy, která daný formát požadavku dokáže zpracovat. Tato třída je potomkem abstraktní třídy `WebnodeApiRequest` a musí implementovat metodu `WebnodeApiRequest::loadParameters()`. Tato metoda je volána při vytváření objektu a zajistí správné načtení parametrů požadavku. Tyto parametry jsou název funkce, formát odpovědi, ověřovací klíče a ostatní parametry předávané volané funkci. Pro použitý formát požadavku je k dispozici konkrétní potomek `RestRequest`.

Po načtení parametrů požadavku je na základě zjištěného názvu funkce, která se má zavolat, připraven objekt třídy, která zpřístupňuje tuto funkci. Tato třída musí být potomkem abstraktní třídy `WebnodeApiClass`. Potomek při vytváření instance zaregistruje zpřístupněné funkce, zda je k jejich volání nutná autorizace nebo autentizace a také je uveden výčet povinných parametrů dané funkce. Potomek také musí implementovat metodu `WebnodeApiClass::runMethod()`, která zajistí zavolání požadované funkce a navrácení jejího výsledku obaleného v objektu třídy `WebnodeApiResult`.

Objekt potomka třídy `WebnodeApiClass` je získán třídou `WebnodeApiClassFactory` (návrhový vzor *Factory*).

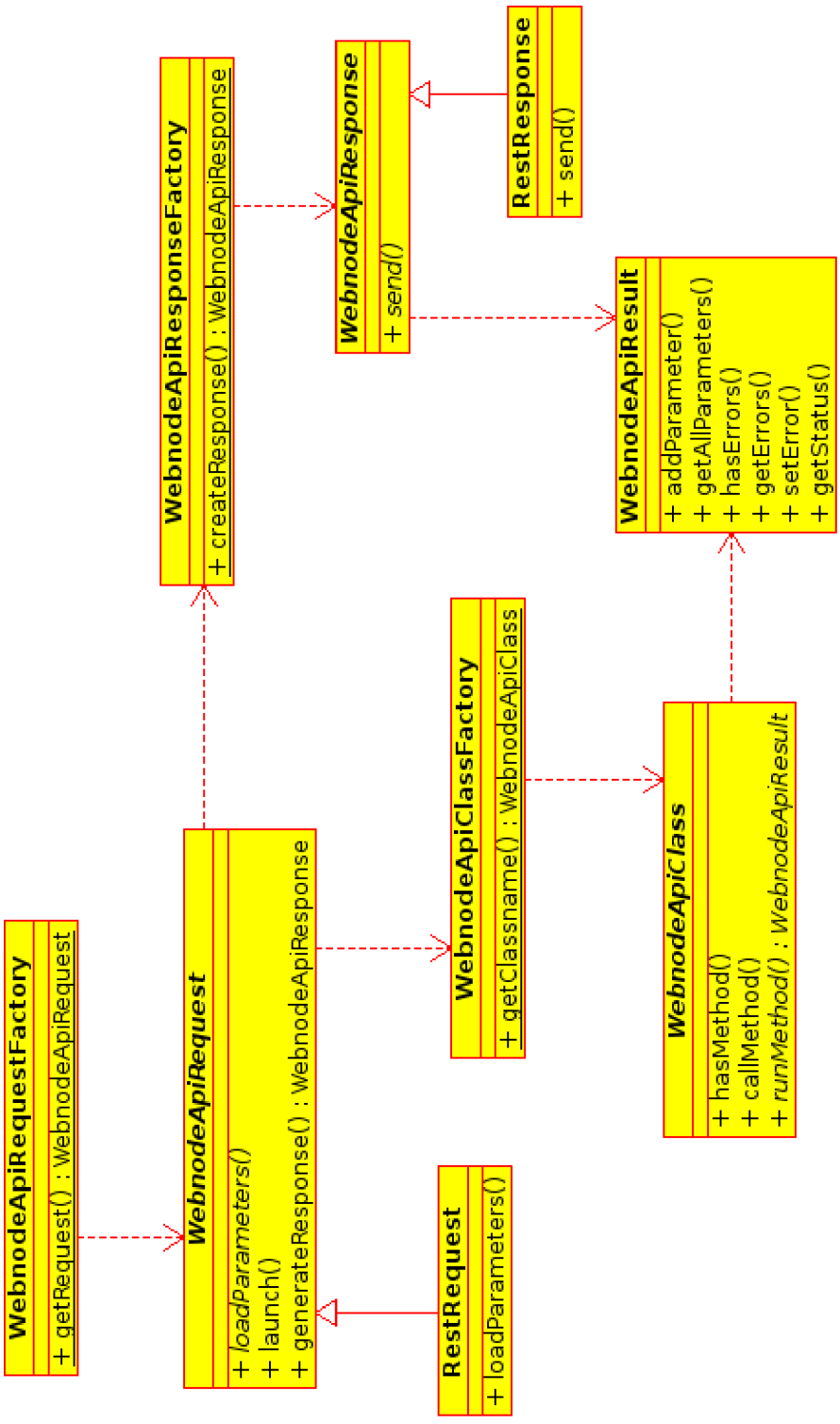
Po získání třídy obsahující volanou funkci je ověřena případná autorizace a autentizace požadavku, na základě definice dané funkce. Pokud vše sedí, pak je požadavek připraven k zpracování. To proběhne po zavolání metody `WebnodeApiRequest::launch()`, kdy dojde k volání samotné požadované funkce pomocí metody `WebnodeApiClass::callMethod()`. Výsledek volání je uložen do atributu `WebnodeApiRequest::result` a připraven k předání do odpovědi.

Po zavolání metody `WebnodeApiRequest::generateResponse()` je na základě formátu odpovědi pomocí třídy `WebnodeApiResponseFactory` (návrhový vzor *Factory*) vytvořen objekt potomka abstraktní třídy `WebnodeApiResponse` a je mu předán získaný výsledek požadavku v objektu třídy `WebnodeApiResult`. Potomek třídy `WebnodeApiResponse` musí implementovat metodu `WebnodeApiResponse::send()`, ta se postará o vrácení výsledku v daném formátu zpět volajícímu.

Pokud v jakémkoliv kroku dojde k chybě, kterou lze zachytit, pak je vrácena chybová odpověď. Pro zachytitelné chyby je použit systém výjimek, pro všechny chyby související s rozhraním je použita třída `WebnodeApiException`.

Na obrázku (3.1) je diagram tříd v jazyce UML, který popisuje navržené rozhraní.

3.1 Model rozhraní



Obrázek 3.1: Diagram tříd rozhraní

3.2 Formát požadavku

Požadavek ve formátu REST je standardní HTTP požadavek s GET parametry na URL `http://api.webnode.com/services/rest/`. Poslední část URL (`rest/`) označuje formát požadavku. Pokud by v budoucnu byly přidány další formáty, pak by se k nim přistupovalo pomocí URL `http://api.webnode.com/services/nazev_metody/`.

Jediným povinným parametrem je `method`, který určuje, která funkce zpřístupněná rozhraním se má volat. Funkce jsou pojmenovány ve formátu `webnode.trida.funkce`. Nejjednodušší požadavek tedy může vypadat například jako:

`http://api.webnode.com/services/rest/?method=webnode.test.echo`.

Existuje několik rezervovaných parametrů, které mají speciální význam, další parametry pro volání metod mohou mít jakýkoliv jiný název. Rezervované parametry jsou:

- `method` – název volané funkce
- `type` – formát požadavku
- `key` – klíč pro autorizaci
- `signature` – kontrolní signatura pro autentizaci
- `response` – požadovaný formát odpovědi

3.3 Formát odpovědi

Odpovědí je XML dokument s kořenovým elementem `response`, který má jeden atribut `status` obsahující stav provedení požadavku. Tento může nabývat následujících hodnot:

- `ok` – požadavek proběhl v pořádku
- `fail` – při zpracování požadavku došlo k chybě
- `yes` – kladná odpověď na volání funkce, která vrací boolean hodnotu. Tyto funkce začínají prefixem *is-* (například `webnode.projects.isProjectAvailable`).
- `no` – záporná odpověď na volání funkce, která vrací boolean hodnotu.

Pokud se požadavek na volání funkce provede korektně, v těle elementu `response` se nacházejí elementy obsahující jednotlivé části odpovědi. Tvar výstupu je vždy uveden v dokumentaci k jednotlivým zpřístupňovaným funkcím. Například pro volání funkce `webnode.test.echo` může vypadat odpověď následovně:

```
<?xml version="1.0" encoding="utf-8" ?>
<response status="ok">
    <output>Hello world</output>
</response>
```

Pokud dojde při zpracování požadavku k chybě, je vrácna chybová odpověď. Ta obsahuje v těle elementu `response` jeden další element `error`, který obsahuje název chyby a má jeden atribut `number` s číselným kódem chyby. Chybové stavy s kódem 1xx jsou společné pro všechny funkce. Jejich výčet je následující:

- **100** – neznámá chyba
- **101** – chyba autorizace, neplatný API klíč
- **102** – chyba autentizace, signatura parametrů není správná, nebo podepisovací klíč nesedí k API klíči.
- **103** – neznámý formát požadavku
- **104** – neznámý formát odpovědi
- **105** – nesprávné použití požadavku
- **106** – volaná funkce nebyla nalezena
- **107** – chybí jeden nebo více povinných parametrů pro volanou funkci

Výčet dalších možných chybových stavů je vždy uveden v dokumentaci k jednotlivým funkcím. Chybová odpověď může vypadat následovně:

```
<?xml version="1.0" encoding="utf-8" ?>
<response status="fail">
    <error number="101">exceptions.unknownApiKey</error>
</response>
```

Kapitola 4

Implementace rozhraní

Rozhraní je implementováno v programovacím jazyce PHP 5 a plně využívá jeho objektově-orientované možnosti.

4.1 Poskytované služby

V této části jsou popsány jednotlivé třídy a jejich funkce zpřístupněné pomocí rozhraní. Ke každé funkci je uveden její krátký popis, úroveň zabezpečení, výčet jejich parametrů, formát odpovědi a případné chybové kódy. Tato část může sloužit jako rychlá referenční příručka pro uživatele rozhraní.

4.1.1 webnode.test

Tato třída obsahuje testovací funkce sloužící pro ověření správného fungování rozhraní.

webnode.test.echo

Funkce vrací v odpovědi hodnoty všech předaných nerezervovaných parametrů oddělených znakem pro nový řádek. Slouží k nejzákladnějšímu otestování funkčnosti rozhraní.

Zabezpečení přístupu

Tato funkce nevyžaduje žádné zabezpečení.

Parametry

Tato funkce nemá žádné povinné parametry. Akceptuje libovolné množství dalších parametrů.

Odpověď

Pokud je volání úspěšné, vrácený stav odpovědi je `ok`.

Všechny parametry předané voláním této funkce jsou vráceny v parametru odpovědi `output` oddělené znakem pro nový řádek.

webnode.test.authorizedEcho

Jedná se o totožnou funkci, jako `webnode.test.echo`, s tím rozdílem, že je vyžadována autorizace požadavku.

Zabezpečení přístupu

Tato funkce vyžaduje platný API klíč.

Parametry

- `key` (povinný) – platný API klíč

Funkce dále akceptuje libovolné množství dalších parametrů.

Odpověď

Pokud je volání úspěšné, vrácený stav odpovědi je `ok`.

Všechny parametry předané voláním této funkce jsou vráceny v parametru odpovědi `output` oddělené znakem pro nový řádek.

webnode.test.authenticatedEcho

Jedná se o totožnou funkci, jako `webnode.test.echo`, s tím rozdílem, že je vyžadována autentizace požadavku.

Zabezpečení přístupu

Tato funkce vyžaduje platný API klíč a podpis požadavku.

Parametry

- `key` (povinný) – platný API klíč
- `signature` (povinný) – platná signatura parametrů požadavku

Funkce dále akceptuje libovolné množství dalších parametrů.

Odpověď

Pokud je volání úspěšné, vrácený stav odpovědi je `ok`.

Všechny parametry předané voláním této funkce jsou vráceny v parametru odpovědi `output` oddělené znakem pro nový řádek.

4.1.2 webnode.user

Tato třída obsahuje funkce pro práci s uživatelskými účty v systému *Webnode*.

webnode.user.isExistingUser

Funkce kontroluje zda v systému *Webnode* již existuje uživatel se zadanou e-mailovou adresou.

Zabezpečení přístupu

Tato funkce vyžaduje platný API klíč a podpis požadavku.

Parametry

- `key` (povinný) – platný API klíč
- `signature` (povinný) – platná signatura parametrů požadavku
- `e-mail` (povinný) – e-mailová adresa uživatele

Odpověď

Pokud uživatel v systému *Webnode* již existuje, je vrácen stav odpovědi `yes`, pokud neexistuje, je vrácen stav odpovědi `no`.

webnode.user.getUserId

Funkce vrací identifikační číslo (ID) uživatele se zadanou e-mailovou adresou.

Zabezpečení přístupu

Tato funkce vyžaduje platný API klíč a podpis požadavku.

Parametry

- `key` (povinný) – platný API klíč
- `signature` (povinný) – platná signatura parametrů požadavku
- `e-mail` (povinný) – e-mailová adresa uživatele

Odpověď

Pokud požadavek proběhl korektně, je vrácena odpověď se stavem `ok`. V parametru odpovědi `id` je uloženo ID hledaného uživatele.

Pokud uživatel se zadanou e-mailovou adresou neexistuje, pak je vrácena odpověď se stavem `fail` a chybovým kódem 301.

Chybové kódy

- **301** – Uživatel se zadanou e-mailovou adresou neexistuje

4.1.3 webnode.project

Tato třída obsahuje funkce pro zakládání nových projektů nebo manipulaci a zjišťování informací o projektech již existujících v systému *Webnode*.

webnode.project.getProjectStatus

Funkce vrací stav projektu v systému *Webnode* podle zadané adresy.

Zabezpečení přístupu

Tato funkce vyžaduje platný API klíč.

Parametry

- `key` (povinný) – platný API klíč
- `projectUrl` (povinný) – celé doménové jméno projektu v systému *Webnode* (např. test.webnode.cz).

Odpověď

Pokud požadavek proběhl korektně, je vrácena odpověď se stavem `ok`. V parametru odpovědi `status` je vrácen stav projektu. Možné stavy projektu jsou následující:

- **available** – adresa projektu je volná k registraci
- **registered** – projekt je zaregistrován, adresa není k dispozici
- **invalid** – adresa projektu obsahuje neplatné znaky
- **banned** – adresa projektu je nedostupná, protože obsahuje zakázaná nebo rezervovaná slova
- **installing** – projekt je zaregistrován a právě probíhá jeho instalace
- **installed** – projekt je zaregistrován, je nainstalován a čeká na aktivaci
- **ready** – projekt je zaregistrován, je aktivován a čeká na dokončení úvodního průvodce
- **running** – projekt je zaregistrován a běží
- **disabled** – projekt je zaregistrován, ale byl deaktivován
- **deleted** – projekt je zaregistrován, ale byl smazán uživatelem
- **unknown** – adresa není dostupná k registraci, ale důvod není známý

Pokud došlo při zpracování požadavku k chybě, je vrácena odpověď se stavem `fail`.

webnode.project.addProject

Funkce vytvoří nový projekt v systému *Webnode* s daným jménem a přiřadí je danému uživateli. Pokud uživatel v systému *Webnode* ještě neexistuje, pak je vytvořen. Je možné zakládat projekty i pod vlastní doménou (v případě partnerů).

Zabezpečení přístupu

Tato funkce vyžaduje platný API klíč a podpis požadavku.

Parametry

- `key` (povinný) – platný API klíč
- `signature` (povinný) – platná signatura požadavku
- `projectUrl` (povinný) – celé doménové jméno nového projektu (např. test.webnode.cz).
- `email` (povinný) – e-mailová adresa uživatele, kterému má být projekt přiřazen

- `password` (povinný) – heslo uživatele (minimálně šest znaků dlouhé). Pokud uživatel s danou e-mailovou adresou již v systému *Webnode* existuje, toto heslo musí odpovídat heslu tohoto uživatele.
- `sendLoginData` (volitelný) – určuje, zda se má na e-mailovou adresu uživatele odeslat e-mail s informací o vytvoření projektu. Povolené hodnoty jsou `1` (informace se odešlou) a `0` (informace se neodešlou). Výchozí hodnota je `0`.
- `activate` (volitelný) – určuje, zda se má projekt automaticky po vytvoření aktivovat. Povolené hodnoty jsou `1` (projekt bude automaticky aktivován) a `0` (projekt aktivován nebude). Výchozí hodnota je `0`.
- `language` (volitelný) – kód jazyka administračního rozhraní vytvořeného projektu. Výchozí hodnota je `en`.

Odpověď

Pokud požadavek proběhl korektně, je vrácena odpověď se stavem `ok`. V parametru odpovědi `project` je vrácena adresa nově vytvořeného projektu.

Pokud se vytváří projekt pod partnerskou doménou a k této doméně si partner sám spravuje DNS záznamy, je vrácena sada parametrů odpovědi `dnsrecord`, které obsahují DNS záznamy nutné pro správné fungování projektu.

Pokud došlo při zpracování požadavku k chybě, je vrácena odpověď se stavem `fail`.

Chybové kódy

- **200** – Neplatná e-mailová adresa uživatele
- **201** – Neplatné heslo uživatele. Pokud již uživatel v systému *Webnode* existuje, pak tato chyba značí, že zadané heslo neodpovídá jeho heslu. Pokud uživatel zatím neexistuje, pak je heslo příliš krátké nebo obsahuje nepovolené znaky.
- **501** – Projekt nelze zaregistrovat z jiných důvodů.
- **502** – Adresa projektu není platná, protože obsahuje nepovolené znaky.
- **503** – Projekt se zadanou adresou již existuje.
- **504** – Adresa projektu obsahuje zakázaná nebo rezervovaná slova.
- **505** – Neplatná partnerská doména.

webnode.project.sendProjectActivation

Funkce znovu odešle aktivační e-mail k projektu na e-mailovou adresu jeho vlastníka. Vhodné například, pokud uživateli první e-mail z nějakého důvodu nebyl doručen.

Zabezpečení přístupu

Tato funkce vyžaduje platný API klíč a podpis požadavku.

Parametry

- `key` (povinný) – platný API klíč
- `signature` (povinný) – platná signatura požadavku

- `projectUrl` (povinný) – celé doménové jméno projektu v systému *Webnode* (např. test.webnode.cz).
- `password` (povinný) – heslo uživatele, který je vlastníkem projektu v systému *Webnode*.

Odpověď

Pokud je požadavek vykonán korektně, pak je vrácena odpověď se stavem `ok`. Současně jsou navráceny dva parametry odpovědi a to `project`, ve kterém je uložena plná adresa projektu a `address`, kde je uložena e-mailová adresa uživatele, kterému byl aktivační e-mail odeslán.

Pokud došlo při zpracování požadavku k chybě, je vrácena odpověď se stavem `fail`.

Chybové kódy

- **201** – Neplatné heslo uživatele.
- **500** – Projekt se zadanou adresou neexistuje.
- **506** – Projekt již byl aktivován.
- **507** – Projekt ještě není k aktivaci připraven.

webnode.project.deleteProject

Funkce deaktivuje nebo smaže projekt v systému *Webnode*. Deaktivace není trvalá a projekt lze následně znovu aktivovat.

Zabezpečení přístupu

Tato funkce vyžaduje platný API klíč a podpis požadavku.

Parametry

- `key` (povinný) – platný API klíč
- `signature` (povinný) – platná signatura požadavku
- `projectUrl` (povinný) – celé doménové jméno projektu (např. test.webnode.cz).
- `delete` (volitelný) – určuje, zda se má projekt fyzicky smazat. Povolené hodnoty jsou `1` (projekt bude nenávratně smazán) a `0` (projekt bude pouze deaktivován). Výchozí hodnota je `0`.

Odpověď

Pokud požadavek proběhl korektně, je vrácena odpověď se stavem `ok`.

Pokud došlo při zpracování požadavku k chybě, je vrácena odpověď se stavem `fail`.

Chybové kódy

- **500** – Projekt se zadanou adresou neexistuje.

webnode.project.getAuthServer

Funkce získá adresu přihlašovacího serveru systému *Webnode* platného pro přihlášení k zadanému projektu.

Zabezpečení přístupu

Tato funkce vyžaduje platný API klíč.

Parametry

- `key` (povinný) – platný API klíč
- `projectUrl` (povinný) – celé doménové jméno projektu v systému *Webnode* (např. test.webnode.cz).

Odpověď

Pokud je požadavek vykonán korektně, pak je vrácena odpověď se stavem `ok`. Současně je navrácen parametr odpovědi `authserver`, ve kterém je uložena adresa přihlašovacího serveru systému *Webnode*, který k danému projektu přísluší.

Pokud došlo při zpracování požadavku k chybě, je vrácena odpověď se stavem `fail`.

Chybové kódy

- **500** – Projekt se zadanou adresou neexistuje.

webnode.project.getProjectStatistics

Funkce vrací statistická data o zadaném projektu v systému *Webnode*. Je možné získat počty unikátních návštěv, počet zobrazených stránek a objem přenesených dat za konkrétní měsíc.

Zabezpečení přístupu

Tato funkce vyžaduje platný API klíč a podpis požadavku.

Parametry

- `key` (povinný) – platný API klíč
- `signature` (povinný) – platná signatura požadavku
- `projectUrl` (povinný) – celé doménové jméno projektu v systému *Webnode* (např. test.webnode.cz).
- `year` (volitelný) – rok pro který se mají vracet statistická data. Výchozí hodnotou je aktuální rok.
- `month` (volitelný) – měsíc pro který se mají vracet statistická data. Výchozí hodnotou je aktuální měsíc.

Odpověď

Pokud je požadavek vykonán korektně, pak je vrácena odpověď se stavem `ok`. Odpověď obsahuje následující parametry:

- `visits` – počet unikátních návštěv.
- `pages` – počet zobrazených stránek.

- `traffic` – objem přenesených dat.

Pokud došlo při zpracování požadavku k chybě, je vrácena odpověď se stavem `fail`.

Chybové kódy

- **500** – Projekt se zadanou adresou neexistuje.

4.2 Autentizace požadavku

Pro volání některých funkcí zpřístupněných rozhraním je vyžadována přítomnost platné signatury požadavku. Ta je vytvořena za pomoci neveřejného podepisovacího klíče a je připojena jako parametr `signature` požadavku.

Signatura se vytváří ze všech parametrů požadavku, které nejsou mezi rezervovanými parametry. Postup pro vytvoření signatury je následující:

1. Předávané parametry požadavku se seřadí podle abecedy na základě názvu parametru. Např. *projectUrl=test.webnode.cz, email=mira@test.cz, password=webnode* se seřadí jako *email=mira@test.cz, password=webnode, projectUrl=test.webnode.cz*

2. Názvy parametrů i jejich hodnoty se postupně spojí do jednoho řetězce. Např. *emailmira@test.czpasswordwebnodeprojectUrltest.webnode.cz*

3. Tomuto řetězci se předřadí podepisovací klíč. Např. *PODPISKLICemailmira@test.czpasswordwebnodeprojectUrltest.webnode.cz*

4. Vypočítá se `sha1` hash[1] z výsledného řetězce. Tento tvoří signaturu požadavku a použije se jako hodnota parametru `signature`.

Následuje příklad vytvoření signatury požadavku v jazyce PHP 5. Definovaná funkce `create_signature()` vyžaduje jako první paramter podepisovací klíč a jako druhý parametr asociativní pole s parametry, kde klíčem v tomto poli je název parametru a hodnotou hodnota parametru.

```
<?php
    function create_signature($signatureKey, array $parameters)
    {
        $reserved = array('type', 'key', 'signature', 'method', 'response');
        $string = $signatureKey;
        $sortedParameters = $parameters;
        ksort($sortedParameters);

        foreach ($sortedParameters as $name => $value)
        {
            if (false === in_array($name, $reserved))
            {
                $string .= $name . $value;
            }
```

[1] Secure Hash Algorithm – viz `http://en.wikipedia.org/wiki/Sha1`

```
        }

        return sha1($string);
    }
?>
```

4.3 Příklad použití

Následující příklad ukazuje vytvoření nového projektu v systému *Webnode* v jazyce PHP 5.

V poli `$parameters` se nachází parametry požadavku, v řetězci `$signatureKey` pak podepisovací klíč. Příklad využívá funkci `create_signature()` definované výše.

```
<?php
    $parameters = array(
        'projectUrl' => 'test.webnode.cz',
        'email' => 'mira@test.cz',
        'password' => 'webnode',
        'key' => 'ea48a55051d485ee8246203855c20808',
        'method' => 'webnode.projects.addProject'
    );
    $signatureKey = '394e8a405e22f42aeb1da47f3a294d19';
    $signature = create_signature($signatureKey, $parameters);
    $parameters['signature'] = $signature;

    $url = 'http://api.webnode.com/services/rest/';
    $concatParameters = array();
    foreach ($parameters as $name => $value)
    {
        $concatParameters[] = $name . '=' . $value;
    }
    $url .= '?' . implode('&', $concatParameters);

    $result = file_get_contents($url);
    $xml = simplexml_load_string($result);
    if ($xml['status'] == 'fail')
    {
        // nastala chyba
    }
    else if ($xml['status'] == 'ok')
    {
        // vse probehlo v~poradku
    }
?>
```

4.4 Výkonnostní testy

Pro odhad snesitelného zatížení vytvořeného rozhraní byly provedeny výkonnostní testy za pomoci programu *Webserver Stress Tool*[2].

Byl testován požadavek nevyžadující autentizaci (volání funkce `webnode.test.echo`) a požadavek vyžadující autentizaci (volání funkce `webnode.test.authenticatedEcho`). V průběhu 10 minut byly prováděny požadavky od 400 klientů (klienti byli do testu přidáváni postupně) v maximálním objemu 50 požadavků za vteřinu. Tato testovací situace je podstatně nadhodnocena oproti předpokládanému provozu, který je odhadován na maximálně desítky požadavků za minutu.

Z výsledku testu vyplývá, že čas vyřízení požadavku lineárně stoupá v závislosti na současném počtu přistupujících klientů. Vyřízení autentizovaného požadavku trvá přibližně o 40 % déle než vyřízení neautentizovaného požadavku, což odpovídá dodatečné režii související s ověřením signatury požadavku.

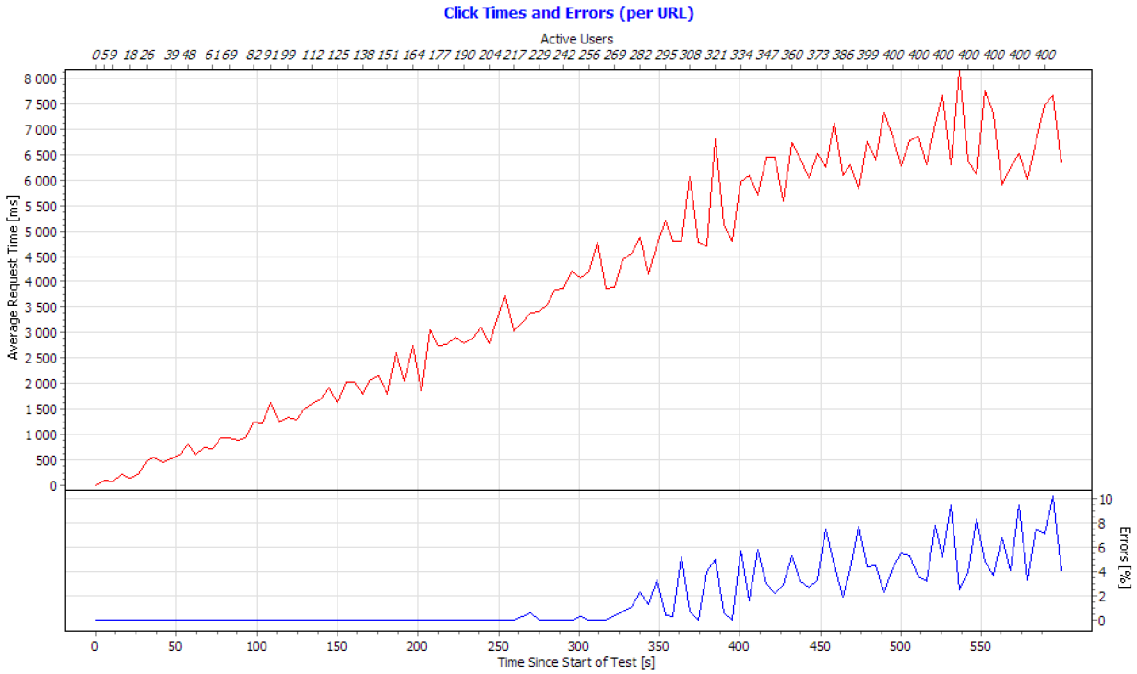
Pro neautentizovaný požadavek se při současném přístupu přibližně 250 klientů začaly vyskytovat chyby při spojení, které při počtu 400 klientů dosáhly až 10 % celkového počtu požadavků za vteřinu. U autentizovaného požadavku se tyto chyby začaly vyskytovat při současném přístupu přibližně 200 klientů a při počtu 400 klientů dosahovaly až 14 % celkového objemu požadavků.

Pro předpokládané zatížení (desítky požadavků za minutu) je výkon dostačující.

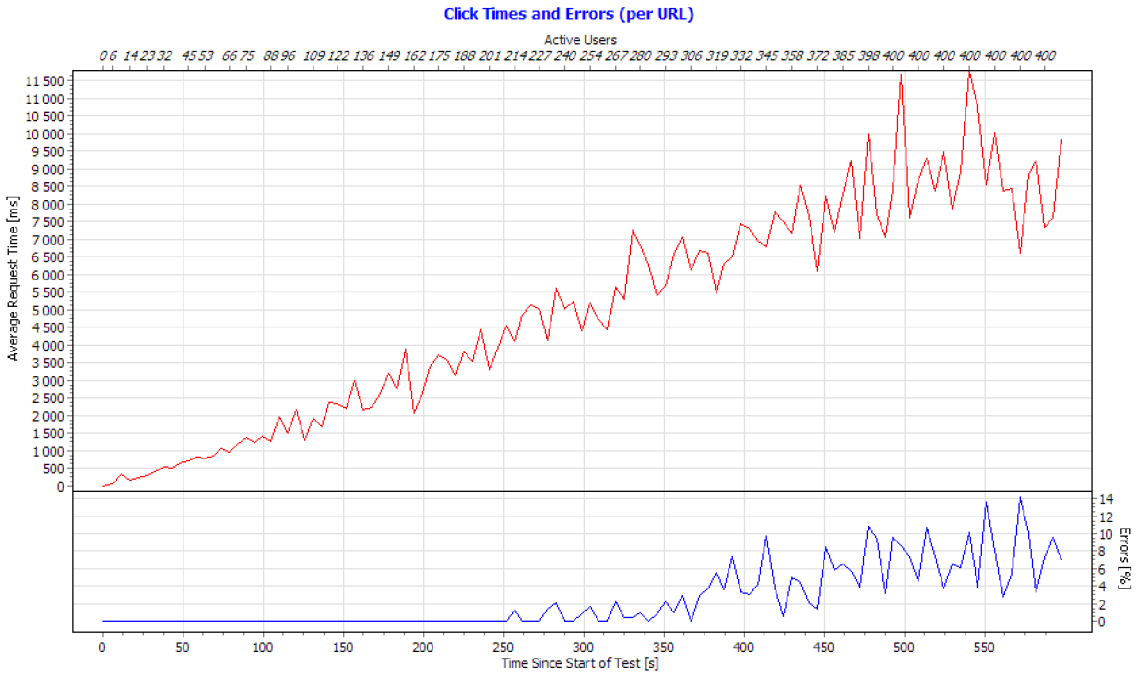
V grafu na obrázku 4.1 je zobrazen průběh testu pro neautentizovaný požadavek, na obrázku 4.2 pak průběh testu pro autentizovaný požadavek.

Oba grafy mají na vodorovné ose znázorněn čas v sekundách. V horní části grafu se nachází průměrný čas odpovědi na požadavek (červenou barvou), na svislé ose je vyznačen čas v milisekundách. V dolní části grafu se nachází počet neúspěšných požadavků, na svislé ose je vyznačeno procentuální zastoupení chyb v celkovém objemu požadavků.

[2] `http://www.paessler.com/webstress`



Obrázek 4.1: Požadavek bez autentizace – průměrný čas požadavku a objem chyb



Obrázek 4.2: Autentizovaný požadavek – průměrný čas požadavku a objem chyb

Závěr

Cílem této práce bylo seznámit se se současnými technologiemi a prostředky pro vzdálený přístup ke službám informačních systémů a na základě těchto poznatků navrhnout a implementovat aplikační programové rozhraní (API) pro vzdálený přístup k informačnímu systému *Webnode*.

Byla provedena rešerše dostupných relevantních technologií a na základě této rešerše pak byla provedena analýza a návrh konkrétního řešení API a jeho zabezpečení.

Navržené rozhraní využívá kombinaci REST (representational state transfer) architektury spolu s RPC (remote procedure call), za využití jazyka XML jako obálky pro přenos odpovědí. Samotné komunikační rozhraní je pomocí objektově orientovaného návrhu odstíněno od vlastní funkcionality, kterou zpřístupňuje, a proto je v budoucnu možné systém rozšířit o další metody vzdáleného přístupu, pokud by tato potřeba vyvstala. K zabezpečení rozhraní je využito dvojice klíčů pro podepisování požadavků a protokolu HTTPS.

Rozhraní zpřístupňuje funkcionalitu pro správu webových prezentací vytvořených v systému *Webnode* – vytváření, mazání, deaktivace – a získávání statistických dat o těchto prezentacích – počet přístupů, přenesená data, atd.

Podle vytvořeného návrhu proběhla implementace rozhraní v jazyce PHP. Rozhraní bylo důkladně otestováno a předáno zadavateli.

Možné náměty pro další vývoj této práce jsou například již zmíněné další metody pro vzdálený přístup nebo rozšíření zpřístupněné funkcionality o další prvky, jako je například vzdálená publikace obsahu nebo propojení s jinými informačními systémy poskytujícími zajímavý obsah.

Literatura

[1] Bradley, N.: *XML – Kompletní průvodce*. Grada, 2000, ISBN 80-7169-949-7.

[2] Cerami, E.: *Web Services Essentials*. O'Reilly, 2002, ISBN 0-596-00224-6.

[3] Fielding, R. T.: *Architectural Styles and the Design of Network-based Software Architectures*. Dizertační práce, University of California, 2001.

[4] Gamma, E.; Helm, R.; Johnson, R.; aj.: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994, ISBN 0-201-63442-2.

[5] Haas, H.; Brown, A.: Web Services Glossary [online]. `http://www.w3.org/TR/ws-gloss/`, 2004-02-11 [cit. 2008-12-05].

[6] Kolektiv autorů: RFC 2616 – Hypertext Transfer Protocol – HTTP/1.1 [online]. `http://www.ietf.org/rfc/rfc2616.txt`, 1999 [cit. 2009-03-14].

[7] Kolektiv autorů: SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) [online]. `http://www.w3.org/TR/2007/REC-soap12-part0-20070427/`, 2007 [cit. 2008-12-05].

[8] Kosek, J.: *PHP – Tvorba interaktivních internetových aplikací*. Grada, 1998, ISBN 80-7169-373-1.

[9] Mitra, N.; Lafon, Y.: SOAP Version 1.2 Part 0: Primer (Second Edition) [online]. `http://www.w3.org/TR/2007/REC-soap12-part0-20070427/`, 2007 [cit. 2008-12-05].

[10] Newcomer, E.: *Understanding Web Services – XML, WSDL, SOAP and UDDI*. Addison-Wesley, 2002, ISBN 0-201-75081-3.

[11] Rescorla, E.: RFC 2818 – HTTP Over TLS [online]. `http://www.ietf.org/rfc/rfc2818.txt`, 2000 [cit. 2009-03-14].

[12] Wikipedia: Representational State Transfer [online]. `http://en.wikipedia.org/wiki/REST`, 2009 [cit. 2008-12-05].

[13] Wikipedia: Web services protocol stack [online]. `http://en.wikipedia.org/wiki/Web_services_protocol_stack`, 2009 [cit. 2008-12-05].

[14] Wikipedia: Uniform Resource Locator [online]. `http://en.wikipedia.org/wiki/URL`, 2009 [cit. 2009-02-17].

[15] Winer, D.: XML-RPC Specification [online]. `http://www.xmlrpc.com/spec/`, 2003-03-30 [cit. 2008-12-05].