

# Hudební přehrávač ovládaný prostřednictvím webové služby

Music player controlled via web service

Bc. Tomáš Seiner

---

Diplomová práce  
2013



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky  
akademický rok: 2012/2013

## ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Tomáš SEINER**  
Osobní číslo: **A11862**  
Studijní program: **N3902 Inženýrská informatika**  
Studijní obor: **Informační technologie**  
Forma studia: **kombinovaná**

Téma práce: **Hudební přehrávač ovládaný prostřednictvím  
webové služby**

Zásady pro vypracování:

1. Vytvořte rešerši na téma využití a tvorby webových služeb s důrazem na možnosti přehrávání multimediálních souborů v počítači.
2. Pomocí programovacího jazyka C/C++ a dalších vhodných technologií vytvořte hudební přehrávač s možností ovládání prostřednictvím webové služby.
3. Vytvořte aplikaci určenou pro ovládání aplikace optimalizovanou také pro mobilní platformy.
4. Vytvořte programovou a uživatelskou dokumentaci.
5. Systém publikujte pod licencí GPL2 a otestujte ho v reálném prostředí.
6. Zhodnoťte přínos a možný rozvoj do budoucna.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **ARLOW, Jim a Ila NEUSTADT.** UML a unifikovaný proces vývoje aplikací: průvodce analýzou a návrhem objektově orientovaného softwaru. Vyd. 1. Brno: Computer Press, 2003, xviii, 387 s. ISBN 80-722-6947-X.
2. **CORMEN, Thomas H.** Introduction to algorithms. 3rd ed. Cambridge: MIT Press, 2009, xix, 1292 s. ISBN 978-0-262-03384-8.
3. **KANISOVÁ, Hana a Miroslav MÜLLER.** UML srozumitelně. 2. aktualiz. vyd. Brno: Computer Press, 2006, 176 s. ISBN 80-251-1083-4.
4. **PRATA, Stephen.** Mistrovství v C. 3. aktualiz. vyd. Překlad Boris Sokol. Brno: Computer Press, 2007, 1119 s. ISBN 978-80-251-1749-1.
5. **SMART, Julian.** Cross-platform GUI programming with wxWidgets. Upper Saddle River: Prentice-Hall, 2006, xxxv, 700 s. ISBN 01-314-7381-6.
6. **VAN ENGELEN, Robert.** GSOAP: 2.8.14 User Guide. Fsu.edu [online]. GENIVIA INC, 2000, 3.2.2013. Dostupné z: <http://www.cs.fsu.edu/engelen/soap.html>
7. **UN4SEEN DEVELOPMENTS.** BASS documentation: 2.4 [online]. un4seen.com, 2012. Dostupné z: <http://www.un4seen.com/doc/bass/bass.html>

Vedoucí diplomové práce:

**Ing. Michal Bližňák, Ph.D.**

Ústav informatiky a umělé inteligence

Datum zadání diplomové práce:

**22. února 2013**

Termín odevzdání diplomové práce:

**22. května 2013**

Ve Zlíně dne 22. února 2013

prof. Ing. Vladimír Vašek, CSc.

*děkan*



doc. Mgr. Roman Jašek, Ph.D.

*ředitel ústavu*

## ABSTRAKT

Tato diplomová práce se zabývá návrhem a vytvořením hudebního přehrávače, ovládaného pomocí webové služby. I když je na internetu celá řada hudebních přehrávačů, žádný z nich není určen pro ovládání z více počítačů najednou. Sice k některým z nich existují rozšíření, která to umožňují, výsledek ovšem není moc přívětivý pro běžného uživatele.

Při vytváření serveru, se jako první musely navrhnout webové služby, pomocí kterých se bude server ovládat. Za pomoci gSOAPu se vygenerují potřebné hlavičky webového serveru. Při zpracovávání požadavků webových služeb je zapotřebí přehrávání hudby. K tomu se využívá zvukové knihovny BASS, která umožňuje přehrání základních hudebních formátů. Aby mohla aplikace běžet na pozadí počítače bez zásahu uživatele, byla obalena funkcemi pro vytvoření služby Windows. Tímto se zajistí běh aplikace již po startu počítače bez nutnosti přihlašování do Windows.

Pro navrhnutí grafického uživatelského rozhraní klienta bylo využito knihoven wxWidgets. Ovládáním klienta se volají příslušné webové služby, které ovládají přehrávání na počítači kde běží server. Mobilní verze klienta je napsána v JavaScriptu a na rozdíl od počítačové verze neumožňuje práci s playlistem.

Výsledkem je webový server, který běží jako služba Windows, dokáže přehrávat hudbu a ukládat aktuální playlist. Klienti z různých počítačů na stejné síti se k tomuto serveru připojí a mohou ho ovládat. Klient má vzhled běžného hudebního přehrávače.

Klíčová slova:

Hudební přehrávač, webové služby, služby windows, zvukové kodeky, BASS, wxWidgets, gSOAP, CodeLite, C++, JavaScript.

## ABSTRACT

This thesis describes the design and creation of the music player, controlled via Web Services. Although on the Internet is a lot of music players, none of them are designed for control from multiple computers. Some of them have extensions that allow it, but the result is not very friendly for common users.

When creating the server, first had to design a web service through which can server will be controlled. gSOAP is used to generate the necessary web server headers. When processing requests of web services, something for playback music is necessary. To that the BASS audio library is used and allows playback of basic audio files. To allow an application run in the background of the computer without user intervention, it was wrapped in functions to create a Windows service. This will ensure the application to run on startup before logging into Windows.

For designing graphical user interface of client was used wxWidgets libraries. By controlling the client it calls the appropriate Web services that control playback on the computer where the server is running. Mobile version of the client is written in JavaScript, and unlike computer version does not work with playlists.

The result is a web server that runs as a Windows service. It can play music and store current playlist. Clients from different computers on the same network can connect to the server and they can operate it. The client has the appearance of a common audio player.

Keywords:

Music player, web services, windows services, audio codecs, BASS, wxWidgets, gSOAP, CodeLite, C++, JavaScript.

Poděkování:

Děkuji vedoucímu práce panu Ing. Michalu Bližňákovi, Ph.D., za jeho cenné rady, připomínky, ochotu a vstřícnost při vypracovávání této diplomové práce.

Dále bych chtěl poděkovat kolegům v práci, za jejich rady, připomínky a hlavně trpělivost.

**Prohlašuji, že**

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

**Prohlašuji,**

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

Ve Zlíně

.....  
podpis diplomanta

**OBSAH**

<b>OBSAH .....</b>	<b>8</b>
<b>ÚVOD.....</b>	<b>10</b>
<b>TEORETICKÁ ČÁST .....</b>	<b>11</b>
<b>1 ZVUK .....</b>	<b>12</b>
1.1 ZVUKOVÉ KODEKY .....	12
1.1.1 Bezkompresní zvukové formáty .....	12
1.1.2 Bezztrátové kompresní zvukové formáty.....	13
1.1.3 Ztrátové kompresní zvukové formáty .....	13
<b>2 POUŽITÉ TECHNOLOGIE A PROGRAMY.....</b>	<b>15</b>
2.1 WEBOVÉ SLUŽBY.....	15
2.1.1 SOAP.....	15
2.1.2 WSDL.....	16
2.1.3 gSOAP.....	16
2.1.4 WS-Security .....	16
2.2 ZVUKOVÁ KNIHOVNA BASS .....	19
2.3 WXWIDGETS.....	20
2.4 VÝVOJOVÉ PROSTŘEDÍ CODELITE.....	21
<b>3 POŽADOVANÉ VLASTNOSTI APLIKACE.....</b>	<b>22</b>
<b>PRAKTICKÁ ČÁST .....</b>	<b>23</b>
<b>4 PROGRAMOVÁ DOKUMENTACE.....</b>	<b>24</b>
4.1 RPLAYSERVER .....	25
4.1.1 Vytváření webových služeb .....	26
4.1.2 Přehrávání hudby .....	31
4.1.3 Playlist.....	34
4.1.4 Konfigurace rPlayServeru .....	38
4.1.5 Implementace webových služeb .....	39
4.1.6 Služba windows .....	39
4.2 RPLAYER .....	42
4.2.1 myConfig.....	42
4.2.2 MainFrameBase .....	43
4.2.3 MainFrame .....	44
4.3 RPLAYER MOBILE .....	51
4.3.1 JavaScript .....	52
<b>5 UŽIVATELSKÁ DOKUMENTACE.....</b>	<b>55</b>
5.1 RPLAYSERVER .....	55

5.1.1	Instalace.....	55
5.1.2	Spuštění a zastavení služby rPlayServer.....	57
5.1.3	Nastavení.....	57
5.1.4	Odinstalování služby.....	58
5.1.5	Testovací provoz.....	58
5.2	RPLAYER.....	58
5.2.1	Instalace.....	59
5.2.2	Ovládání.....	59
5.2.3	Sdílení adresářů pro přehrávání.....	62
5.2.4	Odstranění aplikace.....	64
5.3	RPLAYER MOBILE.....	64
<b>ZÁVĚR.....</b>		<b>66</b>
<b>CONCLUSION.....</b>		<b>67</b>
<b>SEZNAM POUŽITÉ LITERATURY.....</b>		<b>68</b>
<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....</b>		<b>70</b>
<b>SEZNAM OBRÁZKŮ.....</b>		<b>71</b>
<b>SEZNAM TABULEK.....</b>		<b>72</b>
<b>SEZNAM PŘÍLOH.....</b>		<b>73</b>

## ÚVOD

Hudebních přehrávačů pro počítač je v dnešní době celá řada. Mezi nejznámější jistě patří Windows Media Player, Winamp, VLC, atd.. Některé z nich dokonce umožňují vzdálené ovládání, ovšem není to jejich primárním zaměřením a mají spoustu jiných, mnohdy zbytečných, funkcí navíc. Cílem této diplomové práce je navrhnout a vytvořit hudební přehrávač, který bude primárně určen pro ovládání prostřednictvím webových služeb.

V první kapitole bude definováno co je to zvuk, jak vzniká a jak se zaznamenává. Poté se přejde k popsání zvukových kodeků. Jak se rozdělují, jaký je mezi nimi rozdíl, jaké mají výhody, nevýhody a na co se používají.

Druhá kapitola popisuje všechny technologie potřebné k vytvoření aplikace. Čtenář bude seznámen s pojmem "webové služby", jaké protokoly využívají a jak spolu komunikují. Budou uvedeny i další technologie, kterých využívají, jako je WSDL či WS-Security. Dále bude popsána zvuková knihovna BASS a její možnosti využití. Nakonec budou v této kapitole zmíněny knihovny wxWidgets.

Třetí kapitola se věnuje požadavkům na vytvářenou aplikaci, jaké vlastnosti by měla mít, jak by se měla chovat.

Čtvrtou kapitolou začíná praktická část diplomové práce. Tato kapitola obsahuje programovou dokumentaci, kde je aplikace vysvětlena z pohledu programátora. Jsou zde popsány jednotlivé části programu, jejich rozdělení, k čemu slouží jednotlivé funkce a jejich vzájemné propojení.

V páté kapitole s názvem uživatelská příručka je popsán, a pomocí obrázků znázorněn, postup instalace jednotlivých částí aplikace.

## I. TEORETICKÁ ČÁST

## 1 ZVUK

Zvuk je mechanické vlnění, které vyvolává sluchový vjem. Frekvence tohoto vlnění, které člověk vnímá, jsou individuální a leží v intervalu přibližně 16 Hz až 20 000 Hz. Zdrojem zvuku může být každé těleso, které se chvěje. Zvuky se dají rozdělit na tóny a hluky. Tóny, někdy označovány jako zvuky hudební, jsou organizovaným systémem zvuků. Kvalitu a estetické působení hudby je určováno výběrem správných zvuků, jejich rytmickým členěním a uspořádáním. Záznam zvuku a jeho následné přehrávání má dlouhou historii. Od fonografu z roku 1877, přes gramofonové desky, magnetofonové pásky, kompaktní disky až po dnes nejrozšířenější ukládání zvukových souborů do počítače. Pro záznam a přehrávání zvuků v počítači jsou pak potřebné zvukové kodeky.

### 1.1 Zvukové kodeky

Kodek, jak je popsáno v knize *Mobile computing*[10], je zařízení nebo počítačový program, které dokáže transformovat(kódovat a dekódovat) datový proud (stream) nebo signál. V dřívější době se za kodek považoval hardware, který transformoval analogový signál, ale postupem času se tento výraz začal používat i pro software, který převádí jednotlivé digitální formáty mezi sebou. Kodeky ukládají data do zakódované formy (většinou za účelem přenosu, uchování nebo šifrování), ale častěji se používají naopak pro obnovení přesně nebo přibližně původní formy dat vhodné pro přehrávání, případně jinou manipulaci.

Hudební kodek je software nebo zařízení, které komprimuje nebo rozbaluje digitální hudební signál. Většinou jsou hudební kodeky v knihovnách, ke kterým pak mohou další programy, jako je Winamp nebo Windows Media Player, přistupovat. Hudební kodeky se dělí do tří hlavních skupin.

#### 1.1.1 Bezkompresní zvukové formáty

V této skupině je hlavním příkladem formát PCM, což je modulační metoda převodu analogového zvukového signálu na signál digitální, vytvořená roku 1937 Britem Alecem Reevsem. "Princip PCM spočívá v pravidelném odečítání hodnoty signálu pomocí A/D převodníku a jejím záznamu v binární podobě. Při reprodukci se k převodu z digitálního signálu na analogový používá D/A převodník. Určujícími parametry jsou vzorkovací frekvence a jemnost rozlišení jednotlivých hodnot. Vzorkovací frekvence se

pohybuje od 8 kHz u digitálních telefonních linek ISDN přes 44,1 kHz u zvukového CD po ještě vyšší hodnoty u profesionálních záznamových zařízení a ovlivňuje kvalitu (tedy „kotrbatost“) reprodukováného záznamu.“[11] Tento formát je obvykle ukládán v počítačích s příponou .wav nebo .aiff.

Dalším formátem bezkompresních formátů je BMF. Jde víceméně o rozšíření .wav souborů o metadata, jako je třeba časové razítko, které pomáhá při synchronizaci s videem a proto je tento formát zvuku rozšířen hlavně ve filmovém průmyslu.

### 1.1.2 Bezztrátové kompresní zvukové formáty

Jedná se o formáty, které dovolují zpětnou rekonstrukci původního signálu při menší potřebě místa než původní nekomprimovaný zdroj. Mezi nejznámější formáty této slupiny patří FLAC, WavPack, Monkey's Audio a ALAC.

WavPack je bezztrátový formát, který umožňuje hybridní kompresi. Ta vytváří dva soubory: jeden s kvalitní ztrátovou kompresí a druhý s daty, která byla ztracena.

FLAC používá lineární predikci pro konverzi zvukových vzorků do série malých čísel (tzv. reziduálů), která jsou efektivně uložena pomocí Golomb-Riceova kódování. FLAC dosahuje o něco horšího kompresního poměru než jiné zvukové kodeky jako WavPack, zato jeho dekódování je výpočetně méně náročné, umožňuje streamování a je více podporován.

Monkey's Audio je o něco lepší než předchozí dva formáty, to je ale zapláceno pomalejším přehráváním. Používá totiž symetrické algoritmy, což znamená, že potřebuje na dekódování stejný čas jako na zakódování a tak není moc vhodný do přenosných zařízení snažících se mít co největší výdrž. ALAC je obdoba formátu FLAC vytvořená Apple.

### 1.1.3 Ztrátové kompresní zvukové formáty

Ztrátová komprese je způsob ukládání zvukových dat pomocí speciálního algoritmu, který zmenšuje objem těchto dat na zlomek původní velikosti. Přitom se ale některé méně důležité informace ztrácejí a z vytvořených dat již nejdou zrekonstruovat. V této skupině jsou nejznámější formáty mp3 a aac.

Formát mp3 umožňuje zmenšit velikost hudebních souborů v CD kvalitě přibližně na desetinu a to při zachování poměrně vysoké kvality. "MP3 se snaží odstranit redundanci

zvukového signálu na základě psychoakustického modelu. Tedy ze vstupního signálu se odeberou informace, jež člověk neslyší, nebo si je neuvědomuje. Využívá se principů časového a frekvenčního maskování. Kompresi zvuku podle standardu MPEG-1 obsahuje 3 vrstvy, jež se liší kvalitou a obtížností implementace. Při kompresi mluveného slova jsou výsledky výrazně horší. Popsané maskování a potlačování tónů způsobuje, že u mluveného slova může být ve slově potlačena počáteční nebo koncová slabika či dokonce pauzy mezi jednotlivými slovy." [12]

Formát aac byl vyvinut jako logický následovník formátu MP3 na středních až vyšších bitratech v rámci standardu MPEG-4.

## 2 POUŽITÉ TECHNOLOGIE A PROGRAMY

### 2.1 WEBOVÉ SLUŽBY

"Webová služba je softwarový systém umožňující interakci dvou strojů na síti. Je popsána ve strojově zpracovatelném formátu, konkrétně WSDL. S webovou službou ostatní stroje komunikují způsobem, který je předepsaný v popisu služby, pomocí protokolu SOAP přepravené pomocí jiných, již zavedených protokolů.

Protokoly SOAP a WSDL jsou oba provedeny v syntaxi jazyka XML, který je navržen tak, aby byl snadno strojově zpracovatelný. Protokol SOAP i WSDL byly navrženy tak, aby byly co nejméně závislé na zvolené verzi standardu XML. Oba protokoly jsou zavázány dodržet příslušné standardy WSDL (SOAP) a zvolený standard XML.

- jednosměrná komunikace
- požadavek / odpověď
- Peer-to-Peer (konverzace)

a definuje dva druhy účastníků přenosu zprávy:

- odesílatel
- příjemce

Hlavním přínosem webových služeb je přijetí jejich protokolů jako standardů mezi-systémové komunikace a tak vytvářejí vhodný nástroj pro komunikaci mezi firemními systémy a tím i zefektivnění Business-to-Business operací.

Jako vedlejší přínos, ale stále velmi významný, se jeví přijetí protokolu SOAP jako standardu pro přenášení zpráv mezi programy nebo systémy. Mnoho programů tak nabízí rozhraní pro využívání jejich nástrojů prostřednictvím zpráv SOAP." [13]

#### 2.1.1 SOAP

"SOAP (Simple Object Access Protocol) je protokolem pro výměnu zpráv založených na XML přes síť, hlavně pomocí HTTP. Formát SOAP tvoří základní vrstvu komunikace mezi webovými službami a poskytuje prostředí pro tvorbu složitější komunikace.

Existuje několik různých druhů šablon pro komunikaci na protokolu SOAP. Nejznámější z nich je RPC šablona, kde jeden z účastníků komunikace je klient a na druhé straně je server. Server ihned odpovídá na požadavky klienta." [14]

### 2.1.2 WSDL

WSDL (Web Services Description Language) je jazyk, založen na XML, který popisuje funkce nabízející webovou službou. V tomto popisu webové služby, který je možné strojově přečíst, jsou informace jak má být služba volána, jaké parametry očekává a jaká data vrátí.

Zpravidla tedy popisuje SOAP komunikaci. Podporované operace a zprávy jsou popsány abstraktně a potom se omezují na konkrétní síťový protokol a formát zprávy. WSDL se často používá v kombinaci se SOAP a XML, aby poskytovalo webové služby na internetu.

### 2.1.3 gSOAP

Nástroj gSOAP poskytuje automatické spojení SOAPu a XML pro jazyk C a C++. Tento nástroj zjednodušuje vývoj SOAP/XML webových služeb a používání XML v C a C++ použitím automaticky generovaného kódu a rozšířených mapovacích metod. Většina nástrojů pro webové služby nabízí API, které vyžaduje použití knihoven pro specifické XML datové struktury. To nutí uživatele přizpůsobit aplikační logiku těmto knihovnám a to často vede k řešením, která nezajišťuje konzistenci dat, jako bezpečnost a validnost s XML. Naopak gSOAP automaticky mapuje proměnné a uživatelem definované datové typy k ekvivalentním datovým typům v XML. To ve výsledku znamená plnou podporu SOAPu dosaženou pomocí jednoduchého API a umožňuje uživateli se plně soustředit na logiku aplikace bez nutné znalosti detailů WSDL, SOAP a XML.

### 2.1.4 WS-Security

WS-Security vylepšuje zprávy SOAP zajištěním kvality ochrany prostřednictvím integrity zpráv, důvěrnosti zpráv a jediné autentizační zprávy. Tyto mechanismy umožňují použití na širokou škálu bezpečnostních modelů a šifrovacích technologií.

WS-Security také poskytuje univerzální mechanismus pro přiřazování bezpečnostních tokenů se zprávami. Není potřeba žádných specifických typů těchto tokenů a je navržen tak, aby se mohl rozšiřovat o další bezpečnostní prvky (podpora více formátů

bezpečnostních tokenů). Například, klient může předložit doklad totožnosti a doklad konkrétní obchodní certifikace.

Navíc WS-Security popisuje, jak zakódovat binární bezpečnostní tokeny. Konkrétně, jak kódovat X.509 certifikáty a Kerberos tickety, nebo třeba jak se vkládají šifrované klíče. Dále zahrnuje také rozšiřitelné mechanismy, které mohou být použity k dalšímu popisu vlastností pověření, které jsou součástí zprávy.

Použitím rozšiřujícího modelu SOAP, specifikace na něm založené jsou navrženy tak, aby se vytvořilo prostředí pro časté zasílání zpráv. WS-Security samo o sobě, nezajišťuje bezpečnost, ani neposkytuje kompletní bezpečnostní řešení.

WS-Security je jako stavební blok, který se používá ve spojení s dalšími webovými službami a specifickými protokoly k přizpůsobení se velké škále modelů zabezpečení a šifrovacích technologií. Samozřejmě implementací WS-Security nezaručíme, že aplikace nemůže být napadena a nebo že bezpečnost nemůže být ohrožena.

#### **2.1.4.1 Namespace**

WS-Security je navržena pro práci s obecnou strukturou zpráv SOAP a obecným modelem zpracování zpráv a měla by být kompatibilní se všemi verzemi SOAPu.

XML namespace URI, které musí být použito podle specifikace pro wsse: <http://schemas.xmlsoap.org/ws/2002/04/secext>

#### **2.1.4.2 Kvalita ochrany**

Pro zajištění bezpečnosti zprávy SOAP, se uvažuje o dvou typech hrozby.

1. Zpráva může být podvodníkem přečtena či pozměněna.
2. Podvodník může poslat zprávu, které chybí potřebná bezpečnostní opatření.

#### **2.1.4.3 Zabezpečení zprávy**

Ochrana obsahu zprávy proti vniknutí (z hlediska důvěrnosti) nebo upravení (integrita zprávy) jsou primární bezpečnostní obavy. Integrita zprávy je zajištěna využitím XML podpisu ve spojení s bezpečnostními tokeny k zajištění, že zpráva bude přenesena bez jakékoliv úpravy. Mechanismy zajišťující integritu zprávy jsou navrženy tak, aby podporovaly více podpisů, případně i od více zdrojů, a tak, aby byla možnost rozšíření

i pro podporu dalších formátů podpisu. Zabezpečení zprávy využívá XML šifrování a bezpečnostní tokeny tak, aby určité části SOAP zprávy důvěrné. Šifrovací mechanismy jsou navrženy tak, aby podporovaly další šifrovací procesy a operace od více zdrojů.

Příjemce zprávy by měl odmítnout zprávu s neplatným podpisem, chybějícími nebo nesprávnými identifikačními údaji, jako neoprávněnou (nebo chybnou) zprávu.

#### **2.1.4.4 Bezpečnostní element**

Bezpečnostní hlavička <Security> umožňuje přidat bezpečnostní informace pro příjemce. To může být buď konečný příjemce zprávy nebo jen zprostředkovatel. Proto může být tento blok několikrát v SOAP zprávě. Zprostředkovatel může do zprávy přidat jeden nebo více nových subelementů do již existujícího <Security> bloku v případě, že jsou určeny pro stejný SOAP uzel, nebo může přidat jeden nebo více nových hlaviček potřebné pro další příjemce nebo zprostředkovatele.

Jak bylo uvedeno výše, zpráva může mít více <Security> bloků, pokud jsou určeny pro odlišné příjemce. Nicméně, jen jeden <Security> blok může vynechat S:Actor atribut a žádné dva <Security> bloky nemohou mít stejnou hodnotu pro S:Actor. Informace o zabezpečení pro různé příjemce musí být v různých blocích <Security>. Informace v <Security> bloku bez S:Actor jsou pro všechny, ale nesmí být odstraněny dokud zpráva nedorazí konečnému příjemci.

Když jsou jednotlivé elementy přidávány do <Security> bloku, měly by být přidány do již existujících elementů. Proto <Security> blok představuje blok pro podepisování a šifrování zprávy od odesílatele. Toto pravidlo zajišťuje, že příjemce může zpracovat jednotlivé elementy v pořadí, v jakém se zobrazují v bloku <Security>, protože není žádná dopředná závislost mezi jednotlivými elementy. Všimněme si, že specifikace nspecifikuje v jakém pořadí by se měli zpracovávat subelementy. Příjemce si může zprávu zpracovat podle své potřeby.

Když subelement odkazuje na klíč který je v jiném subelementu (například podpis, který se odkazuje na binární bezpečnostní klíč, který obsahuje certifikát), tak by bezpečnostní token ve kterém je tento klíč měl vložit následně klíč pomocí subelementu tak, že budou potřebné informace již přečteny, když se bude tento klíč číst.

## 2.2 Zvuková knihovna BASS

Tato knihovna umožňuje přehrávání nejrůznějších formátů hudebních souborů pod různými operačními systémy. Knihovna je zdarma pro nekomerční použití a umožňuje:

- přehrávání ze souborů MP3/MP2/MP1/OGG/WAV/AIFF
- vytváření ukázek z hudebních souborů
- přehrávání z internetových serverů (HTTP, FTP) s IDN a podporou proxy serveru
- podpora CoreAudio kodeků
- podpora více kanálového zvuku
- využívá stejný engine jako XMPlay pro přesné a vysoce kvalitní přehrávání
- podpora výstupu na více zvukových karet najednou
- pohodlné nahrávání s podporou více vstupů nebo jejich výběru
- dekódování zdroje bez jeho přehrání
- synchronizaci událostí v programu a synchronizaci přehrávání více kanálů najednou
- aplikaci equalizérů: chorus/compressor/distortion/echo/flanger/gargle/reverb
- přehrávání prostorové hudby z jakékoliv pozice
- přizpůsobivost - nastavení velikosti zásobníku pro stabilitu nebo rychlost

Dále se tato knihovna může rozšířit o balíčky, které umí přehrávat:

- wma soubory
- audio cd
- flac soubory
- midi soubory
- wavPacky
- opus soubory

## 2.3 wxWidgets

"wxWidgets je volně šiřitelný, open source multiplatformní widget nástroj. Je to knihovna základních elementů pro tvorbu grafického uživatelského rozhraní (GUI).

wxWidgets umožňuje zkompileovat a spustit program na několika počítačových platformách s minimálními nebo žádnými změnami kódu. To zahrnuje systémy jako Windows, Macintosh, Linux/Unix (X11, Motif, a GTK+), OpenVMS a OS/2. Verze pro embedded systémy je ve vývoji.

Knihovna je implementována v C++, ale její používání je možné v mnoha běžně používaných programovacích jazycích.

wxWidgets obsahuje několik důležitých tříd, jsou to především:

- wxString – implementace textových řetězců
- wxObject – hlavní třída ve wxWidgets, které používají běhové informace o typech
- wxEvtHandler – třída, která má na starosti předávání událostí
- wxWindow – objekty, které mají grafickou reprezentaci jsou potomky této třídy

Ovládací prvky se vytvářejí pomocí operátoru new. V konstruktoru se musí uvést ukazatel na rodičovské okno (oknem se zde rozumí objekt, jenž je potomkem wxWindow). Tím je dáno v jakém okně bude ovládací prvek vložený. Rodičovské okno se postará ve svém destrukturu o uvolnění všech objektů v něm vložených.

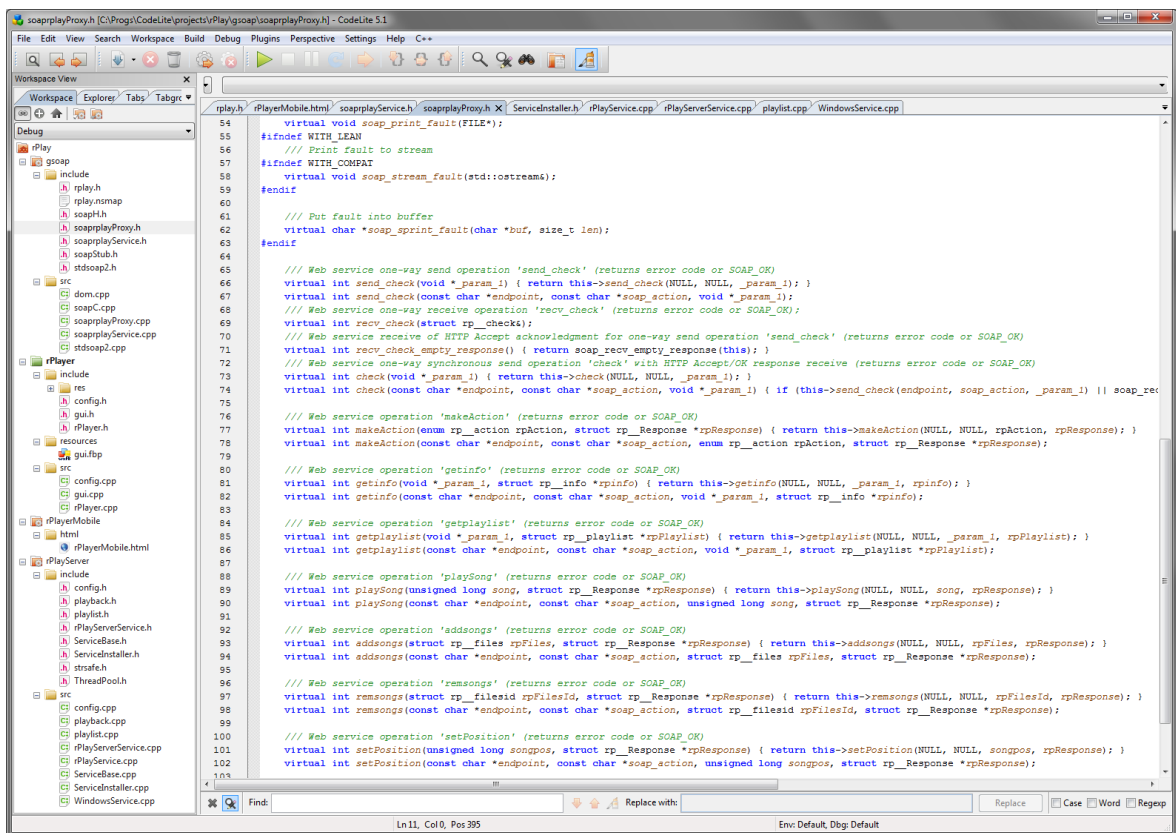
Obecně platí, že komponentám není dobré nastavovat absolutní velikosti a pozice, protože v rámci různých jazyků a nastavení zobrazovacího zařízení mohou mít jiné velikosti, než se kterými vývojář počítal. Ve wxWidgets tuto správu zajišťuje tzv. wxSizer, což je abstraktní třída, která obsahuje předpis [rozhraní] pro ostatní specializované sizery. Sizer dostává události o změně velikosti okna a upravuje velikosti a pozice prvků v něm vložených. Prvkem vloženým do sizeru může být i jiný sizer, ten má potom na starosti velikost a pozice svých prvků v přidělené buňce. Oknu, ve kterém má sizer působit, se musí předat na sizer ukazatel funkcí SetSizer, kterou má v sobě už wxWindow. Okno, ke kterému sizer náleží, si zajistí samo jeho uvolnění z paměti v případě zničení okna." [15]

## 2.4 vývojové prostředí CodeLite

Codelite je multiplatformní, bezplatné vývojové prostředí pro programování v C/C++ s podporou pro wxWidgets. Dál jsou podporovány jakýkoliv překladače nebo nástroje, které se dají ovládat z příkazové řádky.

Codelite nabízí řízení projektů, prohlížení a úpravu zdrojových souborů, doplňování kódu, zvýraznění syntaxe. Dále umožňuje integraci svn, cscope, Unittest++. Samozřejmě nechybí ani debugger založen na gdb.

Codelite je distribuován pod licencí GNU General Public Licence v2.



Obr. 1. Ukázka vývojového prostředí CodeLite

### 3 POŽADOVANÉ VLASTNOSTI APLIKACE

Hudební přehrávač by měl umožňovat přehrávání hudby v počítači. Požaduje se podpora nejen základních hudebních souborů jako jsou mp3, wav, ogg či aiff, ale i dalších jako jsou flac nebo wma. Samotný přehrávač nebude mít žádné uživatelské rozhraní a poběží v počítači na pozadí (ve Windows jako služba, v Linuxu jako démon). Ovládání tohoto přehrávače bude probíhat pouze přes webové služby. Pro definování těchto služeb použijeme jazyk wsdl.

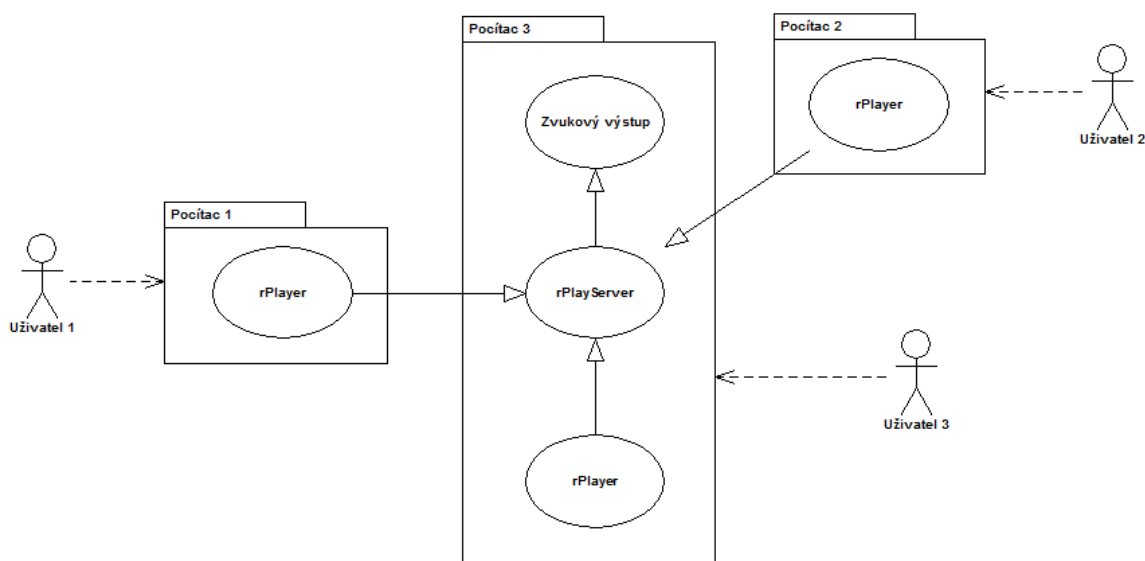
Grafické uživatelské rozhraní(GUI) klienta se bude skládat z několika prvků.

- Za prvé bude mít ovládací prvky pro přehrávání jako jsou spuštění přehrávání, pozastavení přehrávání, zastavení přehrávání, přesun na další nebo předchozí písničku v playlistu, posun na určitou pozici v písničce. Také by měl umět zobrazovat základní informace o písničce jako je její jméno, autor, aktuální čas, celkový čas.
- Druhým prvkem tohoto rozhraní bude playlist. Uživatelské rozhraní musí umožňovat vytvořit playlist. Dále bude umět: přidávat písničky do playlistu, mazat písničky z playlistu, zamíchat pořadí písniček v playlistu nebo zapnout opakované přehrávání písniček v playlistu.
- Třetí součástí tohoto rozhraní bude přihlašovací formulář, kde se bude nastavovat adresa a port pro připojení k vlastnímu přehrávači. Ten může být na stejném nebo jiném počítači připojeném v síti.

## **II. PRAKTICKÁ ČÁST**

## 4 PROGRAMOVÁ DOKUMENTACE

V této části diplomové práce je hudební přehrávač popsán z pohledu programátora. Je zde popsáno, jak jsou jednotlivé součásti programu vytvořené, a jak spolu vzájemně komunikují. Celý balík aplikací se nazývá rPlay, odvozeno od "remote play", a na *Obr. 2* je vidět jak celá tato aplikace může pracovat. rPlayServer je název serverové části a rPlayer je název pro klientskou část.



*Obr. 2. Zobrazení částí aplikace*

Každý z uživatelů má na svém počítači nainstalovaný rPlayer aby mohl ovládat přehrávání hudby. Na "Počítači 3" je pak nainstalovaný vlastní rPlayServer, kde se hudba přehrává. Pokud tedy "Uživatel 1" bude chtít přehrávat další písničku, stiskne tlačítko v rPlayeru, a ten pošle požadavek na rPlayServer přes webovou službu, že se má začít přehrávat další písnička. rPlayServer začne písničku přehrávat a pošle všem klientům informaci o tom, která se přehrává.

Jak už bylo zmíněno, aplikace byly vytvořené v prostředí Codelite, kde v hlavním adresáři nalezneme tyto podadresáře:

- bass - knihovny a hlavičkové soubory knihovny bass24
- buttons - adresář s obrázky potřebnými v uživatelském prostředí
- gsoap - knihovna s webovými službami
- rPlayer - klient přehrávače s grafickým uživatelským rozhraním

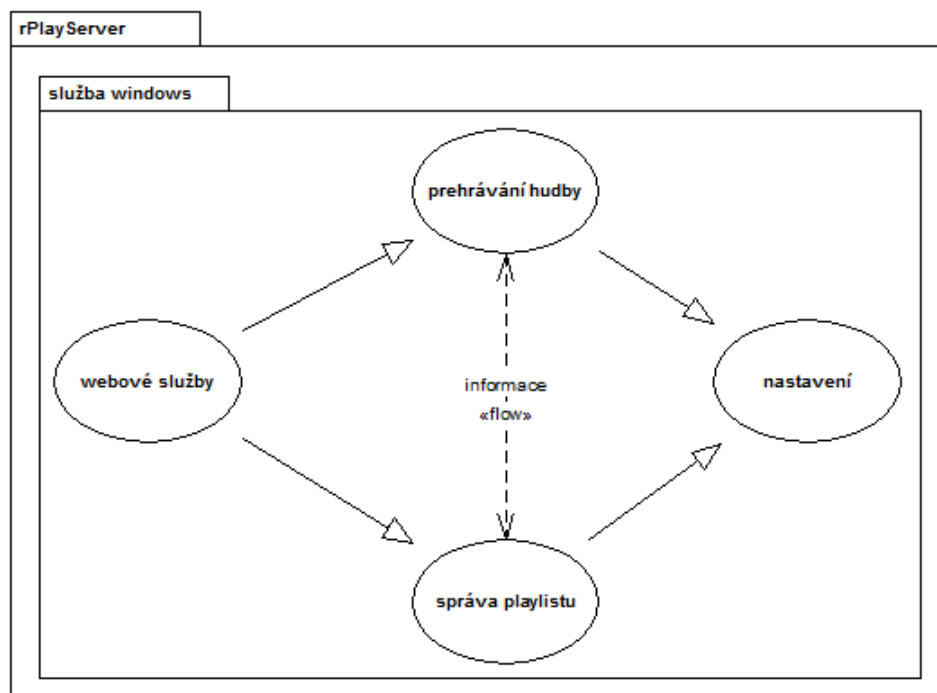
- rPlayer mobile – webové uživatelské rozhraní určené pro mobilní zařízení
- rPlayServer - serverová část přehrávače

#### 4.1 rPlayServer

rPlayServer je hlavní aplikace, která zprostředkovává přehrávání hudby. Skládá se ze tří základních součástí:

- knihovny pro přehrávání hudby: bass24
- knihovny pro vytvoření webových služeb: gSOAP
- obálky pro vytvoření služby na operační systém Windows

Aplikace funguje tak, že pomocí funkcí gSOAPu si vytvoříme server, který dokáže zpracovávat definované webové služby. Při přijetí požadavku, pak server zavolá přiřazenou funkci, která zajistí požadovanou akci. Například v případě přehrávání hudby, server použije některou z funkcí knihovny bass24, která se postará o přehrání aktuální písničky. Na *Obr. 3* je zobrazena základní struktura rPlayServeru a jednotlivé závislosti mezi hlavními třídami.



*Obr. 3. Struktura rPlayServeru*

#### 4.1.1 Vytváření webových služeb

Pro vytvoření webové služby se využívá nástroj *soapcpp2*, který dokáže ze správně definovaného hlavičkového souboru vytvořit základní kostru webové služby jak pro klientskou, tak pro serverovou část. Tento hlavičkový soubor se většinou vytváří pomocí nástroje *widl2h*, který ho vytvoří z dodaného wsdl. Jelikož se tyto služby vytváří od začátku, tak žádné wsdl k dispozici není a musí se tedy hlavičkový soubor vytvořit ručně. Tento hlavičkový soubor je pojmenován *rplay.h* a nachází se v adresáři *gsoap*, kde jsou veškeré soubory týkající se webových služeb společných pro klienta a server.

V tomto hlavičkovém souboru nejsou jen definované služby, ale i výchozí nastavení pro volání těchto služeb. V *Tab. 1. Ukázka nastavení ws v hlavičkovém souboru.* je ukázka toho jak to může vypadat.

*Tab. 1. Ukázka nastavení ws v hlavičkovém souboru.*

```
//gsoap rp service name:      rplay Server for playing music
//gsoap rp service style:     rpc
//gsoap rp service encoding:  encoded
//gsoap rp service namespace: http://rplay.cz/ws/rplay.wsdl
//gsoap rp service location:  http://localhost:53191
//gsoap rp schema namespace:  rp:rplay
```

Kde *rp* je namespace prefixu v xml, který se dále používá i v názvech proměnných a funkcí generovaných pro C/C++. Vysvětlení jednotlivých nastavení služby:

- service name - název služby a popis
- service style - způsob jakým služba pracuje (rpc|document)
- service encoding - jak jsou data posílána (literal|encoded)
- service namespace - jedinečná namespace daného webového serveru
- service location - výchozí nastavení volání webové služby
- schema namespace - nastavení namespace schématu

Definování samotných funkcí webové služby už je pak jednoduché. V *Tab. 2* je ukázka definování funkce pro nastavení hlasitosti.

Tab. 2. Definování webové služby v hlavičkovém souboru.

```
//gsoap rp service method-action: setVolume http://rplay.cz/ws/setVolume
struct rp__Response { char *strerror; };
int rp__setVolume(unsigned long volume, struct rp__Response *rpResponse);
```

Zde je vidět, že se prvně nastaví namespace pro danou funkci a pak se definují použité struktury. V tomto případě pouze struktura s chybou, která nemusí být vyplněná. Nakonec se definuje samotná funkce. Podobným způsobem se pak definují všechny funkce webové služby.

#### 4.1.1.1 *check*

Funkce, která slouží pouze na zjištění, zda je server aktivní nebo ne. V Tab. 3 je ukázka požadavku a odpovědi v xml.

Tab. 3. Ukázka požadavku a odpovědi ws check.

Požadavek	Odpověď
<pre>&lt;soap:Body&gt;   &lt;rp:check/&gt; &lt;/soap:Body&gt;</pre>	

Jak je vidět, tato funkce zpátky nic nevrací, ale tím že se provede a nevrátí chybu se dá zjistit, že server reaguje správně.

#### 4.1.1.2 *getinfo*

Tato funkce slouží k zjištění aktuálních informací, které klient potřebuje k zobrazení v uživatelském rozhraní.

Tab. 4. Ukázka požadavku a odpovědi ws getinfo.

Požadavek
<pre>&lt;soap:Body&gt;   &lt;rp:getinfo/&gt; &lt;/soap:Body&gt;</pre>
Odpověď
<pre>&lt;soap:Body&gt;   &lt;rp:info&gt;     &lt;tsplaylist&gt;2013-05-08 22:03:18&lt;/tsplaylist&gt;     &lt;plbutton&gt;6/44&lt;/plbutton&gt;     &lt;songname&gt;...silence...&lt;/songname&gt;     &lt;songtime&gt;0&lt;/songtime&gt;     &lt;progress&gt;0&lt;/progress&gt;     &lt;volume&gt;75&lt;/volume&gt;     &lt;shuffle&gt;&gt;true&lt;/shuffle&gt;     &lt;repeat&gt;true&lt;/repeat&gt;   &lt;/rp:info&gt; &lt;/soap:Body&gt;</pre>

#### 4.1.1.3 makeAction

Tato funkce je určena ke spuštění jednoduchých akcí. Jednotlivé enumerace, které se dají použít jsou:

- PLAY - začne přehrávat aktuální písničku
- PAUSE - pozastaví přehrávání
- STOP - zastaví přehrávání
- PREV - přesune se na předchozí písničku
- NEXT - přesune se na další písničku
- SHUFFLE - zamíchá nebo setřídí playlist
- REPEAT - zapne nebo vypne opakované přehrávání playlistu
- CLRPL - smaže všechny písničky z playlistu

Tab. 5. Ukázka požadavku a odpovědi ws makeAction.

Požadavek	Odpověď
<pre>&lt;soap:Body&gt;   &lt;rp:makeAction&gt;     &lt;rpAction&gt;PLAY&lt;/rpAction&gt;   &lt;/rp:makeAction&gt; &lt;/soap:Body&gt;□</pre>	<pre>&lt;soap:Body&gt;   &lt;rp:Response/&gt; &lt;/soap:Body&gt;</pre>

Server v tomto případě vrací prázdnou strukturu, pokud vše proběhne v pořádku.

#### 4.1.1.4 *playSong*

Funkce slouží k přehrání určité písničky. Příklad použití v *Tab. 6*.

*Tab. 6. Ukázka požadavku a odpovědi ws playSong.*

Požadavek	Odpověď
<pre>&lt;soap:Body&gt;   &lt;rp:playSong&gt;     &lt;song&gt;2&lt;/song&gt;   &lt;/rp:playSong&gt; &lt;/soap:Body&gt;□</pre>	<pre>&lt;soap:Body&gt;   &lt;rp:Response/&gt; &lt;/soap:Body&gt;</pre>

#### 4.1.1.5 *getplaylist*

Funkce slouží k získání seznamu písniček z playlistu. Spolu se seznamem písniček se také přidává aktuální časové razítko playlistu.

*Tab. 7. Ukázka požadavku a odpovědi ws getplaylist.*

Požadavek	Odpověď
<pre>&lt;soap:Body&gt;   &lt;rp:getplaylist/&gt; &lt;/soap:Body&gt;</pre>	<pre>&lt;soap:Body&gt;   &lt;rp:playlist&gt;     &lt;timestamp&gt;2013-05-01 00:00:00&lt;/timestamp&gt;     &lt;rpSongs xsi:type="rp:song"&gt;       &lt;file&gt;\\SEINER\Music\Lindsey.mp3&lt;/file&gt;       &lt;title&gt;River Flows In You&lt;/title&gt;       &lt;artist&gt;Lindsey Stirling&lt;/artist&gt;       &lt;album&gt;Stirling Strings&lt;/album&gt;       &lt;genre&gt;Other&lt;/genre&gt;       &lt;year&gt;2011&lt;/year&gt;       &lt;track&gt;9&lt;/track&gt;       &lt;albartist&gt;Lindsey Stirling&lt;/albartist&gt;       &lt;length&gt;3:04&lt;/length&gt;     &lt;/rpSongs&gt;   &lt;/rp:playlist&gt; &lt;/soap:Body&gt;</pre>

#### 4.1.1.6 *addsongs*

Funkce přidává písničky do playlistu. Písnička musí být dostupná z obou počítačů, takže by měla být ve sdílené složce, a cesta k ní by měla být v UNC formátu, který přesně identifikuje síťový zdroj.

Tab. 8. Ukázka požadavku a odpovědi ws addsongs.

Požadavek
<pre>&lt;soap:Body&gt;   &lt;rp:addsongs&gt;     &lt;rpFiles&gt;       &lt;file&gt;\\SEINER\Music\Lindsey1.mp3&lt;/file&gt;       &lt;file&gt;\\SEINER\Music\Lindsey2.mp3&lt;/file&gt;     &lt;/rpFiles&gt;   &lt;/rp:addsongs&gt; &lt;/soap:Body&gt;</pre>
Odpověď
<pre>&lt;soap:Body&gt;   &lt;rp:Response/&gt; &lt;/soap:Body&gt;</pre>

#### 4.1.1.7 *remsongs*

Funkce odebírá písničky z playlistu. Posílají se pouze jejich id.

Tab. 9. Ukázka požadavku a odpovědi ws remsongs.

Požadavek
<pre>&lt;soap:Body&gt;   &lt;rp:remsongs&gt;     &lt;rpFilesId&gt;       &lt;fileId&gt;3&lt;/fileId&gt;       &lt;fileId&gt;5&lt;/fileId&gt;     &lt;/rpFilesId&gt;   &lt;/rp:remsongs&gt; &lt;/soap:Body&gt;</pre>
Odpověď
<pre>&lt;soap:Body&gt;   &lt;rp:Response/&gt; &lt;/soap:Body&gt;</pre>

#### 4.1.1.8 *setVolume*

Funkce nastaví hlasitost přehrávání. Hlasitost se udává v procentech (0-100).

Tab. 10. Ukázka požadavku a odpovědi ws setVolume.

Požadavek	Odpověď
<pre>&lt;soap:Body&gt;   &lt;rp:setVolume&gt;     &lt;volume&gt;30&lt;/volume&gt;   &lt;/rp:setVolume&gt; &lt;/soap:Body&gt;</pre>	<pre>&lt;soap:Body&gt;   &lt;rp:Response/&gt; &lt;/soap:Body&gt;</pre>

#### 4.1.1.9 *setPosition*

Funkce nastaví přehrávání písničky do dané pozice. Ta se udává v procentech (0-100).

Tab. 11. Ukázka požadavku a odpovědi ws *setPosition*.

Požadavek	Odpověď
<pre>&lt;soap:Body&gt;   &lt;rp:setPosition&gt;     &lt;songpos&gt;70&lt;/songpos&gt;   &lt;/rp:setPosition&gt; &lt;/soap:Body&gt;</pre>	<pre>&lt;soap:Body&gt;   &lt;rp:Response/&gt; &lt;/soap:Body&gt;</pre>

#### 4.1.2 Přehrávání hudby

O samotné přehrávání hudby se stará třída *myPlayback*, která zastřešuje funkce knihovny *bass24*. Tato třída pracuje tak, že se prvně pomocí funkce *streamCreate()* vytvoří "kanál" pro přehrávání který v tomto případě obsahuje jednu písničku. Když je tento kanál aktivní, tak se k němu dá přistupovat a ovládat pomocí dalších funkcí kontrolujících přehrávání (*play*, *pause*, *stop*, *next*, *prev*) nebo dalších jako jsou nastavení pozice v písničce, hlasitosti přehrávání a dalších. Při přechodu na další písničku, se prvně musí uvolnit tento kanál pomocí funkce *streamFree()*.

##### 4.1.2.1 Vytvoření třídy

Jako první se volá funkce z knihovny *bass24* pro inicializaci zvuku:

```
BASS_Init(-1,44100,0,NULL,NULL)
```

jejíž parametry jsou:

- -1 - znamená, že bude použito výchozí zvukové zařízení
- 44100 - je frekvence výchozího zvuku
- 0 - znamená, že nebudou použita žádná z případných nastavení, jako např.: 3D hudba, mono, 8-bitové rozlišení atp.
- NULL - nastavuje se na rodiče pouze u okenních aplikací
- NULL - pro použití výchozího nastavení pro spuštění *DirectSound*

Pokud se podaří Inicializovat knihovnu *bass24* bez problému, tak se načtou pomocné knihovny pro přehrávání formátů *flac* a *wma* pomocí funkce *BASS\_PluginLoad()*. Nakonec

se z konfiguračního souboru načtou proměnné, zda je playlist zamíchán nebo setříděn a zda se má přehrávání playlistu opakovat.

#### 4.1.2.2 Vytvoření kanálu pro přehrávání

Funkce *streamCreate()* se stará o inicializační záležitosti při nahrání souboru do paměti. Jednotlivé volané funkce z knihovny *bass24* jsou:

- *BASS\_StreamCreateFile()* - Tato funkce vytvoří kanál ze souboru, který se pak ovládá pomocí dalších funkcí. Lze přidat i další nastavení jako 3D hudba, mono, atp..
- *BASS\_ChannelSetSync()* - Velice důležitá funkce, která umožňuje nastavení spuštění určité funkce při splnění určitých podmínek. V tomto případě se nastavuje spuštění funkce *chanEnded()* pokud přehrávaná písnička dojde až do konce.
  - Funkce *chanEnded()* má za úkol přejít na další písničku v playlistu a spustit její přehrávání. Pouze pokud přehrával poslední písničku a není zapnuté opakované přehrávání playlistu, tak přehrávání zastaví.
- *BASS\_ChannelGetInfo()* - zjistí informace o současném kanálu, které se mohou později použít v jiných funkcích
- *BASS\_ChannelGetLength()* - Zjistí délku právě přehrávané písničky
- *BASS\_ChannelBytes2Seconds()* - Převede délku písničky na vteřiny

#### 4.1.2.3 Uvolnění kanálu

Pro uvolnění přehrávané písničky z paměti slouží funkce *streamFree()*. Tato funkce vrací 0, pokud uvolnění proběhne v pořádku. Zároveň může vracet i stav v jakém se přehrávání nacházelo. To je důležité v případě, pokud chceme, aby se další písnička sama začala přehrávat nebo ne.

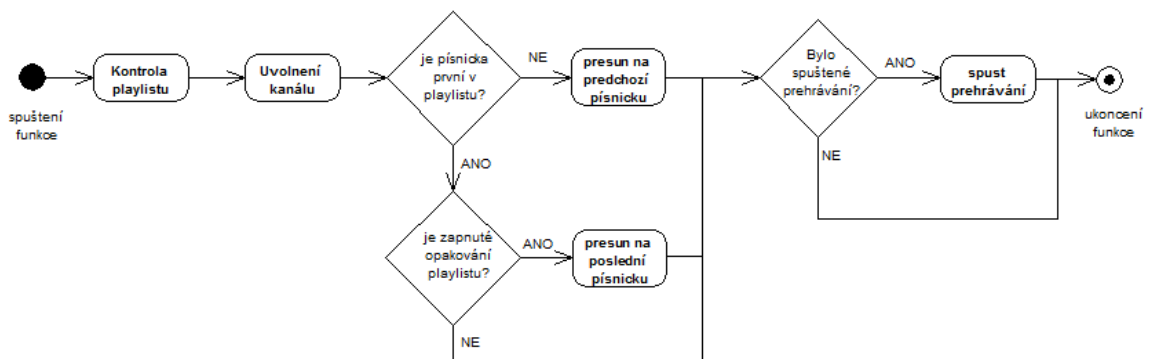
#### 4.1.2.4 Ovládání přehrávání

K ovládání přehrávání se počítají tyto funkce:

- *play()* - začne přehrávat aktuální nebo definovanou písničku. Po zkontrolování zda je co přehrávat vytvoří v případě potřeby kanál pro přehrávání a spustí ho pomocí

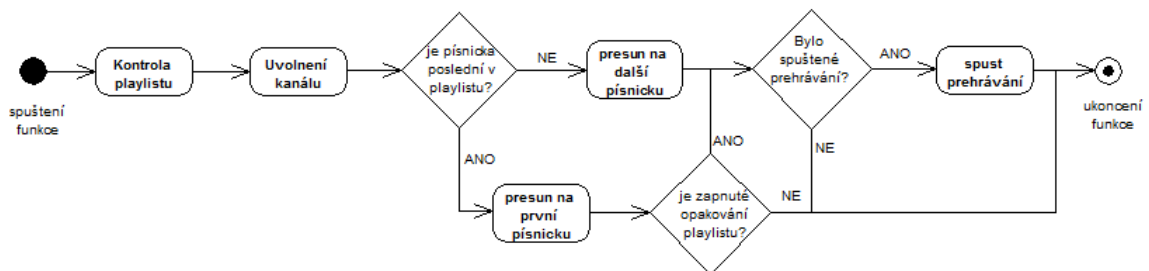
funkce *BASS\_ChannelPlay()*. Pokud vše proběhne v pořádku, tak nastaví text, který se zobrazuje u právě přehrávané písničky.

- *pause()* - zastaví přehrávání pomocí funkce *BASS\_ChannelPause()* a čeká v místě kde skončil.
- *stop()* - zastaví přehrávání pomocí funkce *BASS\_ChannelStop()* a pak pomocí funkce *BASS\_ChannelSetPosition()* nastaví přehrávání na začátek písničky.
- *prev()* - nalezne předchozí písničku v playlistu. Postup je zobrazen na Obr. 4.



Obr. 4. Diagram aktivit funkce *prev()*

- *next()* - nalezne následující písničku v playlistu. Postup je zobrazen na Obr. 5.



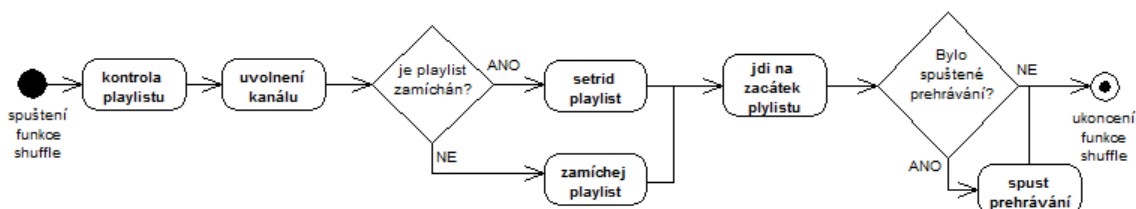
Obr. 5. Diagram aktivit funkce *next()*

#### 4.1.2.5 Nastavení přehrávání

- *setVolume()* - funkce pro nastavování hlasitosti přehrávání. Tato funkce využívá funkce z knihovny bass24 *BASS\_SetConfig()*, která slouží k nastavení kanálu. V tomto případě se používá flag *BASS\_CONFIG\_GVOL\_STREAM*, který nastavuje hlasitost tohoto kanálu a nastavení se pohybuje v rozmezí mezi 0(ticho) až 10000(hlasitost naplno). Mohla by se využít i funkce *BASS\_SetVolume()*, ta ale

nastavuje hlasitost celého prostředí a to není vhodné při používání více zdrojů zvuku, protože se pak nedají mezi sebou nastavit.

- *switchShuffle()* - funkce se přepíná mezi zamíchaným nebo seřazeným playlistem. Funkcionalita je znázorněná na Obr. 6.



Obr. 6. Diagram aktivit funkce *shuffle()*

- *switchRepeat()* - funkce přepíná mezi přehráváním ve smyčce a zastavením po přehrávání celého playlistu. To se celé děje za pomoci jedné proměnné, ve které se mění stav.

#### 4.1.2.6 Získání nastavení pro přehrávání

Jelikož jsou všechny proměnné v této třídě *protected*, tak pro získání jejich hodnoty je zapotřebí funkcí s prefixem *get*. Takže tu jsou funkce jako *getShuffle()*, *getRepeat()*, *getVolume()*, atp. Pro celkový seznam dostupných funkcí v této třídě je možno se podívat do hlavičkového souboru *playback.h*.

#### 4.1.3 Playlist

Playlist, ve třídě *myPlaylist*, je seznam písniček, které jsou určeny k přehrávání. Tyto písničky se mohou do playlistu libovolně přidávat nebo z něho mazat. Každá písnička má svoji vlastní třídu *mySong*, ve které jsou uložena veškerá data, která se dají získat z tagů. Kromě seznamu těchto písniček je v této třídě ještě časové razítko, *timestamp*, ve kterém je zaznamenaný čas poslední změny v playlistu.

Tato třída se vytváří na začátku při spuštění programu a zatím je podporován pouze jeden playlist. Ten je samozřejmě ukládán a při znovuspuštění aplikace je znova načten.

#### 4.1.3.1 Načtení písničky

Pro načtení písničky je potřeba znát pouze její cestu. Ta se dává jako parametr do konstruktoru třídy *mySong*. Tagy se ze souboru načítají pomocí knihovny *tags.dll* a pro její správnou funkci je potřeba nejprve vytvořit kanál pomocí funkce *BASS\_StreamCreateFile()*, ve které se použije parametr *BASS\_STREAM\_DECODE*, který značí, že nebudeme písničku přehrávat, ale pouze o ní zjišťovat informace. Pro získání jednotlivých tagů se používá funkce *TAGS\_Read()*, jejímž prvním parametrem je kanál, ve kterém je písnička načtená a druhém je formát toho co se má načíst. Jednotlivé identifikátory tohoto formátování jsou v *Tab. 12*.

*Tab. 12. Speciální identifikátory používající se ve funkci TAGS\_Read()*

Identifikátor	Název tagu
%TITL	Název písničky
%ARTI	Jméno interpreta
%ALBM	Název alba
%GNRE	Žánr
%YEAR	Datum vydání
%CMNT	Komentář
%TRCK	Číslo stopy
%COMP	Skladatel
%COPY	Copyright
%SUBT	Podtitul písničky
%AART	Autor alba (kresba)

Výrazy, které jsou podporovány knihovnou *tags.dll* jsou v *Tab. 13*.

Tab. 13. Speciální výrazy používající se ve funkci *TAGS\_Read()*

Výraz	Funkce výrazu
%IFV1(x,a)	Pokud není <i>x</i> prázdné, zpracuje <i>a</i> , jinak vrátí prázdný řetězec
%IFV2(x,a,b)	Pokud není <i>x</i> prázdné, zpracuje <i>a</i> , jinak zpracuje <i>b</i>
%IUPC(x)	Převede <i>x</i> na velká písmena
%ILWC(x)	Převede <i>x</i> na malá písmena
%ICAP(x)	Převede počáteční písmena v <i>x</i> na velká ostatní na malá
%ITRM(x)	Odstraní mezery na začátku a konci <i>x</i>
%UTF8(x)	Kóduje tagy v UTF-8(výchozí je v ANSI)

Po načtení tagů se zjistí velikost písničky pomocí funkce *BASS\_ChannelGetLength()* a převede na vteřiny pomocí *BASS\_ChannelBytes2Seconds()*. Ty se poté převedou na hodiny, minuty a vteřiny. Pokud má písnička méně jak hodinu, tak se hodiny nezobrazují.

Pro získání uložených hodnot slouží funkce s prefixem *get*, takže pokud je potřeba získat název písničky, zavolá se funkce *getTitle()*, a tak podobně u dalších proměnných. Další funkce jsou s prefixem *is*. Ty slouží k zjištění, zda je daná proměnná vyplněná nebo prázdná. Takže například funkce *isYear()* vrátí *true* pokud je u písničky vyplněný rok vydání, jinak vrátí *false*.

Pro změnu hodnot nejsou ve třídě *mySong* žádné funkce a jediná možnost je tedy písničku smazat a znovu načíst.

#### 4.1.3.2 Vytvoření playlistu

Vytvoření playlistu se děje v konstruktoru třídy. Jako parametry jsou zapotřebí cesta kam se může soubor uložit a jméno jak se bude soubor playlistu jmenovat. Nejprve se zjistí, zda soubor již existuje a pokud ne, tak se vytvoří nový. Cesty k písničkám jsou uloženy ve formátu *m3u*, což znamená, že každý soubor je na jednom řádku. Na rozdíl ale od klasického *m3u* playlistu, jsou v tomto playlistu cesty k písničkám uloženy v UNC formě.

Pokud tedy jsou v souboru nějaké písničky, tak se pro každou zavolá funkce *add()*, popsána níže, s parametrem cesty, která vloží informace o ní do paměti. Z konfiguračního souboru se načte, na které písničce se skončilo a vytvoří se časové razítko playlistu. Nakonec se zavolá *srand*, který vytvoří unikátní seznam náhodných čísel, která se později mohou použít při náhodném přehrávání písniček.

#### 4.1.3.3 *Spravování písniček v playlistu*

K přidávání a odstraňování písniček z playlistu slouží tyto funkce:

- *add()* - přidává písničku do playlistu. Lze zadat i pořadové číslo, kam se má písnička v playlistu zařadit. Pokud se nezadá, tak se automaticky zařazuje na konec. Ve funkci se nejprve zkontroluje, zda daný soubor existuje, poté se vytvoří nová třída *mySong*, kde jsou uloženy veškeré informace o písničce. Ukazatel na tuto třídu se pak zařadí do seznamu vektorů písniček. Pokud právě zařazená písnička je první, tak se automaticky spustí přehrávání.
- *clear()* - maže celý playlist
- *rem()* - odstraní písničku z dané pozice. Při odstraňování se musí kontrolovat zda se nemaže právě přehrávaná písnička a pokud ano, tak přeskočit na další. Dále pokud se maže písnička která předchází právě hranou písničku, tak se musí změnit id právě přehrávané písničky.

Na konci každé z těchto funkcí se navíc ještě aktualizují názvy tlačítek a textů, které se posílají klientovy do uživatelského rozhraní. Uloží se aktuální playlist do souboru pomocí funkce *savePlaylist()* a nastaví se časové razítko na aktuální čas.

#### 4.1.3.4 *Řazení písniček v playlistu*

Playlist umožňuje písničky zamíchat funkcí *shuffle()* a nebo seřadit pomocí funkce *sort()*. O vlastní zamíchání se postará funkce *std::random\_shuffle()* do které se jako parametry vkládají začátek a konec vektoru *mySong*. Aby tato funkce fungovala správně je potřeba zavolat *std::srand()*, který se volal v konstruktoru playlistu.

O seřazení se postará funkce *std::sort()* a její parametry jsou také začátek a konec vektoru *mySong*. Pro správné fungování této funkce je zapotřebí přetypovat operátor "<". Ten je přetypován tak, že porovnává stávající cestu souboru s cestou k druhému souboru pomocí metody *CmpNoCase()*. Řetězce se porovnávají písmenko po písmenku dokud nenarazí na rozdíl. Pokud je hodnota písmene z prvního řetězce větší než z druhého, tak funkce *CmpNoCase()* vrátí kladnou hodnotu a funkce přetypovaného operátoru vrátí *true*, jinak *false*.

#### 4.1.3.5 Ostatní funkce playlistu

- *getTS()* - funkce pro získání časového razítka. Jedná se o řetězec ve tvaru "rrrr-mm-dd hh:mm:ss".
- *getPlaylistButtonStr()* - funkce pro získání řetězce, který je na tlačítku pro playlist. Ten je ve tvaru "1/10", kde číslo před lomítkem značí pořadové číslo právě přehrávané písničky a číslo za lomítkem je celkový počet písniček v playlistu.
- *getActID()*, *setPrevID()*, *setNextID()*, *setID()* - funkce pro získání aktuálního, předchozího či následujícího id písničky a v případě funkcí s prefixem *set* i jeho uložení do proměnné *actSongID*.
- *actual\_file()* - funkce vrací cestu k aktuálně přehrávané písničce.
- *playingLast()* - funkce vrací *true* pokud právě přehrávaná písnička je poslední v playlistu.

#### 4.1.4 Konfigurace rPlayServeru

Veškerá potřebná nastavení se ukládají do souboru a protože rPlayServer nemá uživatelské rozhraní, tak jediná možnost jak je změnit, je pouze přímá úprava v souboru. Tento soubor se nachází v adresáři *%SystemDrive%\ProgramData* a jeho jméno je *rplayserver.cfg*. Třída *myConfig* má na starosti načítání a ukládání proměnných do tohoto souboru. Zatím jediná důležitá proměnná v tomto souboru je port, na kterém běží server, ostatní údaje, jsou jen zálohy proměnných jako je hlasitost, právě přehrávaná písnička atp., aby se v případě ukončení serveru mohlo načíst původní nastavení.

##### 4.1.4.1 Vytvoření třídy

V konstruktoru třídy *myConfig* se nejprve spojí cesta k souboru a jeho jméno, které dostane v parametrech vytváření třídy. Poté se zkontroluje zda soubor existuje a pokud ne, tak se vytvoří a vloží se do něj výchozí hodnota portu 53191. Pokud již soubor existuje, tak se už jen načte zmíněný port.

#### 4.1.4.2 Načítání a ukládání dat

O načítání dat ze souboru se stará funkce *cfgGetParam()*, jež prochází řádek po řádku souboru a podle jména v parametru funkce hledá schodu. Pokud najde, vrátí zbytek řádku, jinak vrací prázdný řetězec.

Ukládání dat je pomocí funkce *cfgSetParam()*, která prochází také řádek pořádku a hledá schodu jména podle prvního parametru. Pokud najde, tak smaže celý řádek a vloží nový s novou hodnotou z druhého parametru funkce, pokud nenajde, vloží nový řádek na konec souboru.

Pro každou proměnnou, která je ukládána do souboru jsou dvě funkce. Jedna s prefixem *get* pro načtení hodnoty a druhá s prefixem *save* pro uložení hodnoty. Tyto funkce jsou obálkou okolo předně popsaných funkcí.

#### 4.1.5 Implementace webových služeb

Po vygenerování souborů pomocí *soapcpp2.exe* v adresáři *gsoap* vznikne také soubor *soaprplayService.h* ve kterém jsou ke každé webové službě vytvořené virtuální funkce pod třídou *rplayService*, které se musí pro správnou funkci serveru dodělat. Implementace těchto funkcí je v souboru *rPlayService.cpp*.

Každá z těchto funkcí má dva parametry. V prvním je požadavek a v druhém je odpověď. Každý požadavek i odpověď mohou mít jinou strukturu. Ta je definovaná v souboru *rplay.h*. Tyto funkce pracují tak, že zpracují strukturu požadavku a zavolají vhodné funkce v playlistu nebo playbacku. Po jejich zavolání se vyplní struktura odpovědi a vrátí *SOAP\_OK*. Tímto způsobem se zpracovávají všechny požadavky a jde o jediný způsob jak ovládat *rPlayServer*.

#### 4.1.6 Služba windows

Pro vytvoření služby se musí celá aplikace obalit funkcemi služby. Pro tento účel, je vytvořena třída, která se o to postará. Funkce pro vytvoření této služby jsou rozděleny do několika souborů, podle jejich účelu:

#### 4.1.6.1 *WindowsService*

Soubor *WindowsService.cpp* obsahuje vstupní bod služby, tj. funkci *main()*, která podle argumentů při spuštění službu nainstaluje, odinstaluje, nebo ji spustí v testovacím režimu. Pokud je soubor spuštěn bez parametru, tak se předpokládá, že je spouštěn jako služba.

Také se tu nastavují základní parametry této služby:

- SERVICE\_NAME - jméno služby
- SERVICE\_DISPLAY\_NAME - název, který se bude zobrazovat v správci služeb
- SERVICE\_START\_TYPE - způsob, jakým se bude služba spouštět:
  - SERVICE\_DEMAND\_START - služba je spouštěna ručně nebo až na požádání od správce zařízení (PnP).
  - SERVICE\_AUTO\_START - služba je spuštěna během startu operačního systému.
  - SERVICE\_SYSTEM\_START - služba je spuštěna během inicializace operačního systému. Využívá se hlavně k nastartování ovladačů zařízení, které nejsou potřeba pro start operačního systému.
  - SERVICE\_BOOT\_START - služba je spuštěna spouštěčem operačního systému. Vyžadováno pro všechny ovladače, které se využívají při startu operačního systému.
- SERVICE\_DEPENDENCIES - seznam závislostí této služby
- SERVICE\_ACCOUNT - jméno účtu, pod kterým služba poběží
- SERVICE\_PASSWORD - heslo k tomuto účtu

#### 4.1.6.2 *ServiceInstaller*

Soubor *ServiceInstaller.cpp* implementuje funkce pro nainstalování a odinstalování služby. Při instalaci se nejprve získá úplná cesta k souboru, poté se otevře správce služeb pomocí funkce *OpenSCManager()* a nakonec se přidá samotná služba pomocí funkce *CreateService()*. Tímto je služba nainstalována. Operační systém zamkne všechny soubory potřebné k běhu služby a nepovolí s nimi žádnou manipulaci, dokud služba není odinstalována.

Při odinstalování služby se nejprve otevře správce služeb, jako při instalaci, a nastaví se oprávnění pro manipulaci se službou pomocí funkce *OpenService()*. Poté se zkusí zastavit služba pomocí funkce *ControlService()* a nakonec se služba smaže pomocí *DeleteService()*.

#### 4.1.6.3 *ServiceBase*

Soubor *ServiceBase.cpp* poskytuje třídu *CServiceBase* k ovládání služby. Jde o třídu, ze které se pak dědí ta, které má na starosti vlastní program. Funkce k ovládání služby jsou:

- *Start()* - spuštění služby.
- *Stop()* - zastavení služby.
- *Pause()* - pozastavení služby. V případě *rPlayServeru* je zakázáno.
- *Contiue()* - znovuspuštění služby. V případě *rPlayServeru* je zakázáno.
- *Shutdown()* - spouští se při vypínání počítače.

Všechny funkce v tomto souboru tvoří jakousi obálku, kolem virtuálních funkcí, které se doplní později v potomcích. Virtuální funkce mají prefix *On*.

#### 4.1.6.4 *rPlayServerService*

V třídě *rPlayServerService* v souboru *rPlayServerService.cpp* je implementace virtuálních funkcí z třídy *CServiceBase*. Nejdůležitější je funkce *OnStart()*, která zajišťuje vlastní spuštění aplikace. V této funkci se vytváří nové vlákno, které spouští funkci *MainThread()*, která by byla hlavní funkcí aplikace, pokud by neběžela jako služba. Další funkce *OnStop()* pouze nastavuje příznak *m\_fStopping*, který znamená, že má server při další příležitosti skončit.

V funkci *MainThread()* se nejprve zjistí do jakého adresáře lze ukládat data pomocí funkce *set\_data\_dir()*, poté vytvoří všechny dříve popsané třídy: *myConfig*, *myPlayback* a *myPlaylist*. Pokud se to podaří, pokračuje se nastavením proměnných webového serveru. Vytvoří se proměnná třídy *rplayService*, nastaví se timeout a další příznaky a zkusí přiřadit lokální adresu k socketu pomocí funkce *bind()*. Poté jde do nekonečné smyčky, kde čeká na připojení na daném portu pomocí metody *accept()*. Ta, pokud přijme

požadavek, vytvoří jeho kopii a tu zpracuje pomocí funkcí v *rPlayService.cpp*(4.1.5.), jinak podle definovaného timeoutu kontroluje zda se nezměnil příznak *m\_fStopping*, a pokud ano, tak zavolá všechny destruktory tříd a ukončí program.

Ve funkci *set\_data\_dir()* se pomocí funkce *wxGetOsVersion()* zjistí verze operačního systému a podle toho se najde adresář, ve kterém mohou aplikace běžící na tomto stroji zapisovat. V tomto adresáři se vytvoří podadresář *rPlay*, kam se budou ukládat všechny soubory spojené s touto aplikací.

## 4.2 rPlayer

rPlayer je aplikace, typu klient, která komunikuje s hlavním serverem, *rPlayServer*, a zprostředkovává uživateli informace o přehrávání, o playlistu a umožňuje mu ho ovládat. Vzhled aplikace se dá rozdělit do čtyřech částí:

- přihlašovací panel – část aplikace, která se zobrazí v případě že se nepodaří se přihlásit k serveru.
- hlavní ovládání – panel kde je hlavní ovládání, základní informace o právě přehrávané písničce a tlačítka pro zobrazení rozšířeného ovládání a playlistu.
- rozšířené ovládání – panel kde se ovládá hlasitost, přidávají se písničky na playlist, nastavují se vlastnosti přehrávání, dále tlačítka pro odhlášení od serveru nebo ukončení aplikace.
- Playlist – zobrazení všech písniček v playlistu.

Z programového hlediska má aplikace tři třídy, které se starají o běh celé aplikace.

### 4.2.1 myConfig

Tato třída se stará o nastavení aplikace a načítání nebo ukládání tohoto nastavení do souboru. Fungování celé třídy je podobné jako u serveru (4.1.4.).

Prvním rozdílem je umístění konfiguračního souboru. Zatímco *rPlayServer* ukládá data do adresáře určeného pro nastavení aplikací na počítači, *rPlayer* ukládá data do adresáře určeného pro nastavení aplikací, ale pro každého uživatele do jeho vlastních dokumentů.

Druhým rozdílem jsou samozřejmě proměnné, které se ukládají:

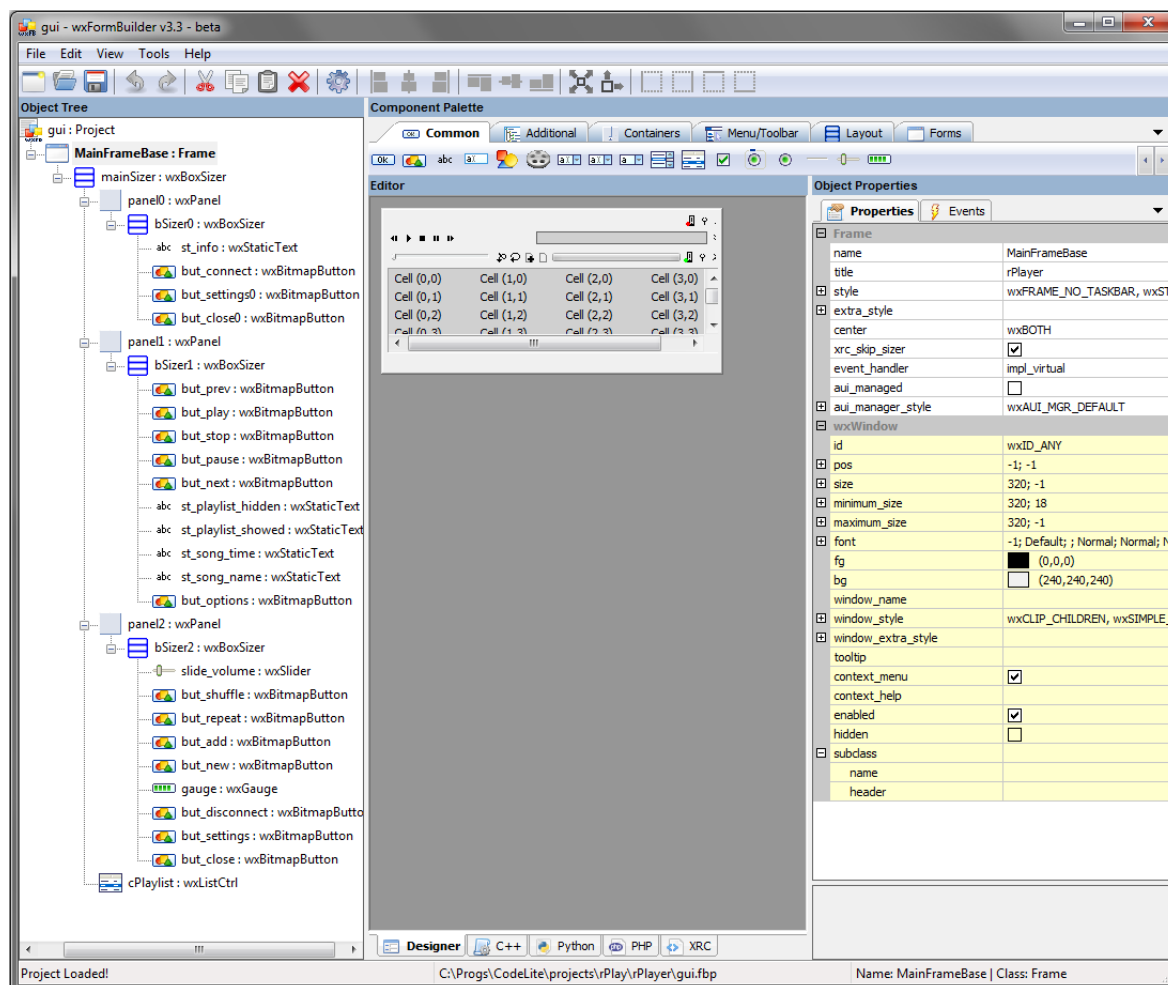
- addr - Adresa kam se rPlayer připojuje. Výchozí nastavení je "http://localhost". Pro připojení na jiný počítač je potřeba zadat jeho síťový název ("\\JINY-POCITAC") nebo jeho ip adresu("http://192.168.1.2").
- port - Port na kterém služba běží. Výchozí hodnota tohoto portu je 53191 a ve většině případů se nemusí měnit.
- posx - x-ová souřadnice pozice aplikace na obrazovce.
- posy - y-ová souřadnice pozice aplikace na obrazovce.
- autc – Automatické zkoušení připojování k serveru každých 10 vteřin.

#### 4.2.2 MainFrameBase

Třída MainFrameBase slouží k definování uživatelského rozhraní aplikace. To je navrhováno v programu wxFormBuilder, který poté vygeneruje soubory gui.cpp a gui.h, které se přidají k vlastní aplikaci. wxFormBuilder je designér grafického uživatelského rozhraní, který umožňuje vytvářet aplikace pod wxWidgets. Jak toto prostředí vypadá je zobrazeno na *Obr. 7*.

Kromě vytváření grafického rozhraní dokáže wxFormBuilder přiřadit k jednotlivým prvkům události, ze kterých pak vygeneruje virtuální funkce, které se dají doplnit v potomcích třídy *MainFrameBase*.

Další dobrou vlastností, která stojí za zmínku, je vložení obrázků z tlačítek přímo do programu. wxFormBuilder ze všech obrázků vytvoří hlavičkové soubory, kde je kromě samotného obrázku i funkce, která tento obrázek dokáže přiřadit k tlačítkům.



Obr. 7. Ukázka návrhu GUI v programu wxFormBuilder.

### 4.2.3 MainFrame

Tato třída slouží především k implementaci virtuálních funkcí z předchozí třídy.

#### 4.2.3.1 Vytvoření třídy

V konstruktoru této třídy se nejprve vytvoří třída *myConfig*, která načte z konfiguračního souboru všechna potřebná nastavení. Dále se vytvoří třída *rplayProxy*, která je základem pro komunikaci s *rPlayServerem*. Tato třída byla vygenerována v adresáři *gsoap* spolu s funkcemi pro *rPlayServer*, čímž se zajistí, že si klient se serverem budou vždy rozumět. V hlavičkovém souboru *rPlayProxy.h* jsou pak všechny funkce, které se mohou použít.

Po vytvoření třídy *rplayProxy* se nastaví pozice aplikace podle uložení v konfiguračním souboru a zkusí se připojit k serveru pomocí webové služby *check*. Pokud se mu to podaří, nastaví aplikaci jako připojenou a pokud ne, tak zobrazí panel pro připojení.

#### 4.2.3.2 Připojení a odpojení aplikace od serveru

Pokud se aplikace odpojí od serveru, to lze pomocí tlačítka "Disconnect" nebo pokud se nepovede připojit na server, tak se provede funkce *rPlayerDisconnected()*. Ta nastaví zobrazení připojovacího panelu (panel0) a skryje všechny ostatní. Nakonec, pokud je nastavené automatické zkoušení připojení, tak nastaví timer na 10 vteřin. Tato funkce je také přiřazena jako událost k tlačítku *but\_disconnect*.

Pokud se aplikace připojí, tak se provede funkce *rPlayerConnected()*. Ta zobrazí hlavní ovládací panel (panel1), všechny ostatní schová a nastaví timer na 1 vteřinu.

Pro kontrolu připojení se používá funkce *checkServer()*, která volá nejjednodušší webovou službu *check*. Ta neobsahuje žádný parametr a ani nemá žádnou odpověď, ale dá se zkontrolovat, zda na server došla nebo ne. Pokud ano, tak se spustí funkce *rPlayerConnected()*, jinak se spustí *rPlayerDisconnected()*. Tato funkce je také přiřazena jako událost k tlačítku *but\_connect*.

#### 4.2.3.3 Získávání aktuálních informací z *rPlayServeru*

Další funkcí, která nepřímo kontroluje připojení k serveru je *getInfo()*. Nepřímo proto, že jejím účelem je získat aktuální informace o právě přehrávané písničce, ale protože se volá každou vteřinu, tak je to většinou první funkce, která zjistí, že připojení není dostupné.

Pro volání webové služby *getinfo* není potřeba žádný parametr do požadavku, zato struktura odpovědi obsahuje tyto údaje:

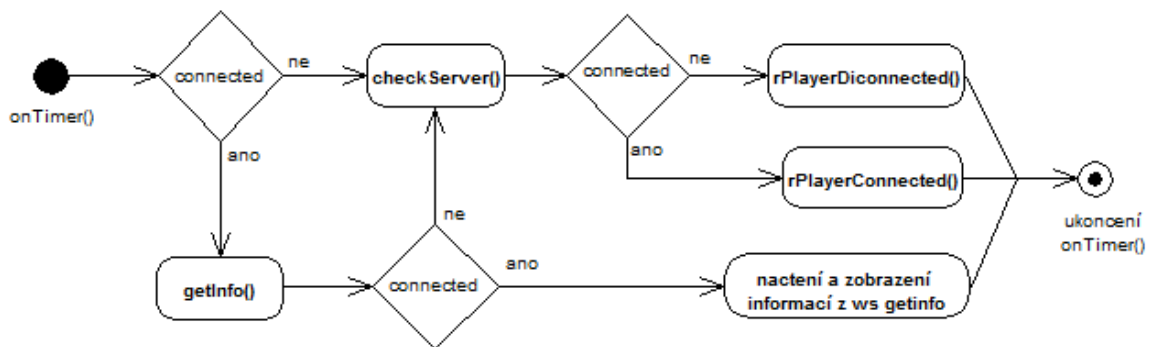
- *tsplaylist* - Řetězec, obsahující časové razítko poslední změny v playlistu. Pokud je novější, tak se zavolá funkce *loadPlaylist()*, která znovu načte playlist.
- *plbutton* - Řetězec, který se zobrazuje na tlačítku pro zobrazení playlistu. Ten obsahuje číslo právě přehrávané písničky a celkový počet písniček v playlistu. Pokud je prázdný, je nahrazen řetězcem *"-/-"*.
- *songname* - Řetězec obsahuje text, který se zobrazí v hlavním textovém poli. Tento řetězec je ve formátu: *"<interpret> - <název písničky> (<délka písničky>)"*.
- *songtime* - číslo udávající uběhlou délku písničky ve vteřinách. To se přepočítává na hodiny, minuty a vteřiny. Pokud písnička trvá déle alespoň hodinu, pak se zobrazí ve formátu *"hh:mm:ss"* jinak se zobrazí pouze ve formátu *"mm:ss"*.

- progress - číslo udávající kolik času uběhlo z písničky v procentech. Touto hodnotou se nastaví *progressbar*.
- volume - číslo udávající hlasitost přehrávání v procentech. Touto hodnotou se nastaví *slide\_volume*.
- shuffle - příznak udávající zda je zapnuto zamíchání playlistu. Podle jeho hodnoty(*true/false*) se nastaví obrázek tlačítka *but\_shuffle*.
- repeat - příznak udávající zda je zapnuto opakované přehrávání playlistu. Podle jeho hodnoty(*true/false*) se nastaví obrázek tlačítka *but\_repeat*.

Díky volání této funkce každou vteřinu, dochází k neustálé aktualizaci GUI a přehrávač tak působí dojmem klasického přehrávače hudby.

#### 4.2.3.4 Timer

Timer, nebo-li česky časový spínač, je funkce, která na pozadí počítá čas a v zadaném intervalu spouští definovanou funkci. V případě rPlayeru se spouští funkce *onTimer()*. Jak tato funkce pracuje je vidět na *Obr. 8*.



Obr. 8. Diagram aktivit funkce *onTimer()*.

#### 4.2.3.5 Ovládání přehrávání

Pro základní ovládání přehrávání slouží webová služba *makeAction*. Ta umožňuje 8 různých akcí, z čehož 5 se týká ovládání přehrávání a 3 se týkají jednoduchých akcí s playlistem. Ke všem těmto akcím je přiřazeno jedno tlačítko, které následně vyvolá jednu z těchto funkcí:

- *OnbPlayClick()* - Spuštění přehrávání.
- *OnbStopClick()*- Zastavení přehrávání.
- *OnbPauseClick()*- Pozastavení přehrávání.
- *OnbPrevClick()*- Skok na předchozí písničku.
- *OnbNextClick()*- Skok na další písničku.
- *OnbShuffleClick()*-Zamíchání/Seřazení playlistu.
- *OnbRepeatClick()*- Nastavení opakovaného přehrávání playlistu.
- *OnbNewClick()*- Vytvoření nového playlistu.

Všechny tyto funkce fungují stejně a to tak, že zavolají funkci *makeAction()* s příslušným parametrem (PLAY, STOP,...) a pokud se webová služba připojí, tak se vrátí odpověď ve struktuře *rp\_\_Response*. A protože tato struktura obsahuje pouze řetězec s chybou, tak se v případě bezchybného zpracování na straně serveru nastaví na *null*.

Pro posun v písničce se využívá události *OnProgressbarLeftUp()*, která reaguje na zdvih levého tlačítka myši nad ukazatelem průběhu písničky. Protože získaná hodnota je pozice v pixelech, je potřeba ji přepočítat na procenta. Tato hodnota se pak vloží jako parametr do volání webové služby *setPosition*. Odpověď je ve struktuře *rp\_\_Response*.

Poslední webová služba, která se počítá mezi základní, je *setVolume*, která slouží k ovládání hlasitosti přehrávání. Při změně hlasitosti se v aplikaci vyvolá událost, která zavolá funkci *OnSliderScroll()*. Ta, při volání webové služby *setVolume*, do ní vloží parametr nastavené hlasitosti získaný pomocí metody *slide\_volume->GetValue()*. Odpověď je zase ve struktuře *rp\_\_Response*.

#### 4.2.3.6 Načtení playlistu

Zobrazení a schování playlistu je přidáno k události na kliknutí na text playlistu, jehož chování je jako tlačítko ale z důvodu neustálého přepisování textu musí být definován jako text.

K naplnění playlistu daty se zavolá funkce *loadPlaylist()*, která zavolá webovou službu *getplaylist*. Struktura odpovědi se skládá ze dvou částí. První je řetězec, obsahující časové

razítko právě poslaného playlistu. Druhou částí je seznam písniček ve struktuře *rp\_\_song*. Z té lze načíst hodnoty uvedené v *Tab. 14*:

*Tab. 14. Seznam proměnných ve struktuře rp\_\_song.*

Proměnná	Popis
file	Cesta k souboru písničky
title	Název písničky
artist	Jméno interpreta
album	Název alba
genre	Žánr
year	Datum vydání
comment	Komentář
track	Číslo stopy
composer	Skladatel
copyright	Copyright
subtitle	Podtitul písničky
albartist	Autor alba (kresba)
length	Délka písničky

Po přijetí dat z webové služby se smaže aktuální playlist a nastaví se vkládání tří sloupců. Poté se prochází seznam všech přijatých písniček a po jedné se vkládají do playlistu ve formátu:

1. sloupec - pořadové číslo písničky v playlistu.
2. sloupec - <jméno interpreta> - <název písničky>.
3. sloupec - délka písničky ve formátu "mm:ss".

Po vložení všech písniček, se automaticky určí velikost pro první a třetí sloupec. Poté se dopočítá velikost druhého sloupce, aby vyplňoval celou šířku aplikace. Nakonec se uloží časové razítko a funkce skončí.

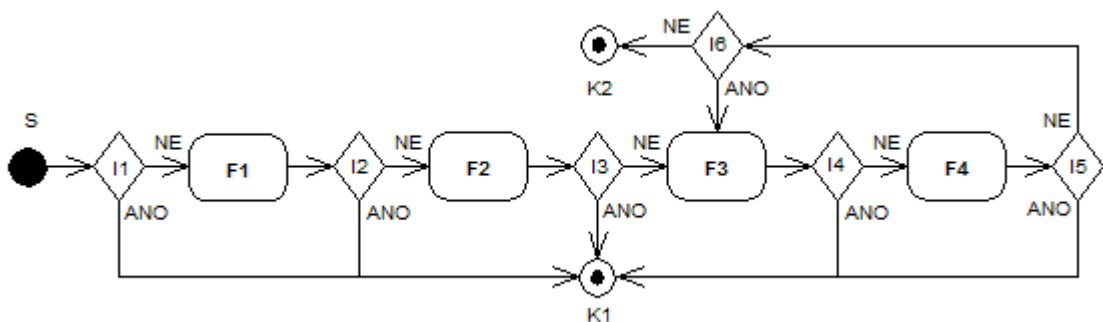
#### 4.2.3.7 Přidávání písniček do playlistu

K přidávání písniček jsou v aplikaci dvě tlačítka. Tlačítko *but\_add* písničky do playlistu přidává a tlačítko *but\_new* vytváří nový playlist. Ve výsledku se v obou případech volá

funkce *add2playlist()* s jedním parametrem, zda se má před přidáváním písniček playlist vymazat nebo ne.

Ve funkci *add2playlist()* se nejprve vytvoří třída *wxFileDialog*, která slouží k otevření dialogového okna. V konstruktoru se pomocí parametrů určí jaké soubory má filtrovat. Knihovna *bass* zvládne přehrávat tyto soubory: \*.mo3, \*.xm, \*.mod, \*.s3m, \*.it, \*.mtm, \*.umx, \*.mp3, \*.mp2, \*.mp1, \*.ogg, \*.wav, \*.aif, \*.flac, \*.wma. Dále se nastaví příznaky, že se soubory mají otevírat, že musí existovat a že jich může být vybráno více najednou. Poté se otevře dialogové okno pomocí metody *GetFileNames()* a uloží se výsledek do pole řetězců.

Pro získání cesty k souborům, kterou bude schopen server načíst slouží funkce *getUncPath()*. Jako parametr se do ní vloží cesta získaná metodou *GetDirectory()*. Popis, jak funkce *getUncPath()* funguje je na Obr. 9 a v Tab. 15.



Obr. 9. Zjednodušený diagram aktivit funkce *getUncPath()*.

Tab. 15. Legenda k diagramu aktivit funkce *getUncPath()* (Obr. 9).

S	Začátek funkce <i>getUncPath()</i>
I1	Zjištění zda cesta už není ve formátu UNC. Začíná řetězec "\\\"?
F1	Spuštění funkce <i>WNetGetUniversalName()</i> , která získá UNC adresu z jednotek připojených ze sítě.
I2	Prošla funkce <i>WNetGetUniversalName()</i> bez chyby?
F2	1. Do proměnné <i>hostName</i> se uloží jméno počítače pomocí funkce <i>wxGetHostName()</i> 2. Od cílové cesty smaže dvojtečku co následuje po označení disku a uloží do proměnné <i>path</i> .
I3	Je sdílený celý disk? Existuje adresář "< <i>hostName</i> >\< <i>path</i> >"?
F3	1. Z cesty se smaže první adresář (vyhledává se první zpětné lomítko) 2. Od cesty se oddělí druhý adresář a uloží do proměnné <i>shareDir</i> . 3. Zbývá cesta se uloží do proměnné <i>path</i> .
I4	Je <i>shareDir</i> sdíleným adresářem? Existuje adresář: "< <i>hostName</i> >\< <i>shareDir</i> >\< <i>path</i> >"?
F4	Pro kontrolu, zda sdílený adresář není schovaný, přidá se za jméno '\$', takže jméno sdíleného adresáře bude vypadat " <i>shareDir</i> \$".
I5	Existuje adresář: "< <i>hostName</i> >\< <i>shareDir</i> \$>\< <i>path</i> >" ?
I6	Je možnost oddělit další podadresář od cesty?
K1	Funkce vrátí cestu ve formátu UNC.
K2	Funkce vrátí prázdný řetězec.

Funkce *getUncPath()* vrátí cestu k souboru přes sdílený adresář, pokud ne, vrátí se chyba v chybové hlášce. Vytvoří se struktura požadavku pro webovou službu *addsongs* a naplní se postupně všemi soubory s cestou ve formátu UNC.

Pokud bylo definováno, že se má vytvořit nový playlist, zavolá se webová služba *makeAction* s parametrem *CLRPL*, která předchozí playlist vymaže. Pote se nakonec zavolá již výše zmíněná služba *addsongs*.

#### 4.2.3.8 Mazání písniček z playlistu

Mazání písniček se provádí ve funkci *OnPlaylistKeyDown()*, což je funkce která se spouští na událost stisknutí tlačítka v playlistu. Nejdříve se tedy zjistí, zda byla stisknuta klávesa *Delete*. Pokud ano, projdou se všechny položky v playlistu a zjistí se, které jsou označené.

Jejich id se pak ukládá do požadavku webové služby *remsongs*, která po zavolání smaže vybrané písničky z playlistu.

#### 4.2.3.9 Posouvání aplikace

Pro posun aplikace je zapotřebí dvou událostí. První událostí je zjištění zda bylo stisknuto levé tlačítko, ta spustí funkci *movingOnLeftDown()*, která uloží aktuální pozici myši. Druhou událostí je zachycení pohybu myši nad objektem. Ta spouští funkci *movingOnMotion()*, která zjistí aktuální pozici a odečte od ní tu, která byla zjištěna ve funkci *movingOnLeftDown()*. Tento rozdíl je dán jako parametr do funkce *Move()*, která zařídí posun aplikace na danou pozici.

### 4.3 rPlayer mobile

rPlayer mobile je odlehčená verze rPlayeru, která umožňuje základní ovládání a zobrazování právě přehrávané písničky. Tato aplikace je určena pro mobilní zařízení, je napsaná v JavaScriptu a obalená jednoduchou webovou stránkou zobrazenou v *Tab. 16*.

Tab. 16. Zdrojový kód html aplikace rPlayer mobile.

```
<html>
<head>
  <title>rPlayer mobile</title>
  <style>... </style>
  <script type="text/javascript">... </script>
</head>
<body onload="Initialize()" onresize="reloadMe()" >
  <div id="songNameDiv"><span id="songName"></span></div>
  <div id="songInfoDiv"><span id="songInfo"></span></div>
  <form id="playbackMenu" action="" method="post">
</body>
</html>
```

Ze zdrojového kódu je zřetelné, že se uživatelské rozhraní skládá ze tří částí, které se mění podle toho zda je aplikace připojena k serveru nebo ne. Pokud je aplikace odpojena, vyplní se daná id následovně:

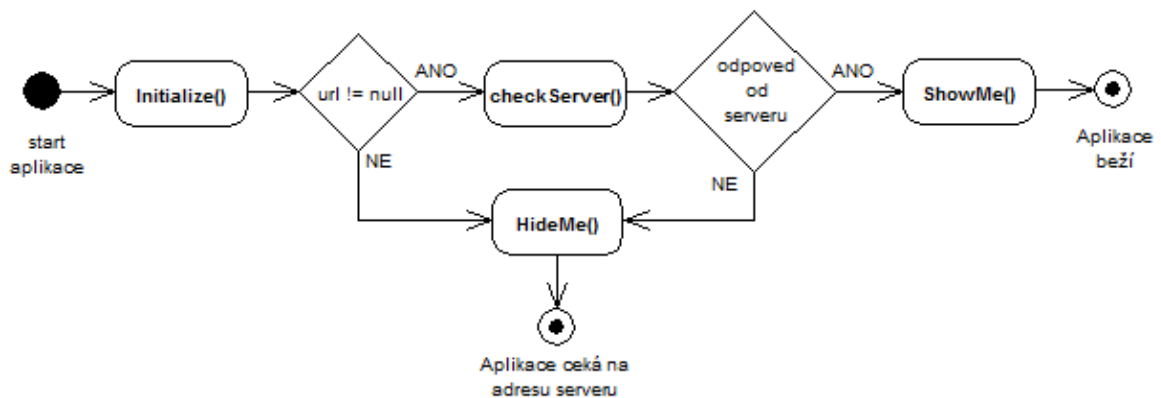
- *songNameDiv* - název aplikace "rPlayer mobile".
- *songInfoDiv* - Výzva k zadání adresy a portu kde se nachází rPlayServer.
- *playbackMenu* - V prvním řádku je input typu text, kam se zadává adresa a na druhém řádku tlačítko pro připojení.

Pokud je aplikace připojena, vyplní se daná id následovně:

- *songNameDiv* - Název písničky, jeho interpret a délka písničky.
- *songInfoDiv* - Hlasitost a čas přehrávané písničky, její pořadí a celkový počet písniček v playlistu, indikace zamíchání playlistu a opakované přehrávání.
- *playbackMenu* - Deset tlačítek pro ovládání ve dvou řádcích.

### 4.3.1 JavaScript

Pro přepsání html kódu stránky se využívá funkcí JavaScriptu, jež je schopný to měnit, aniž by byla potřeba obnovovat stránku. A i když ve zdrojovém html kódu prakticky nic není, tak díky JavaScriptu se to hned po načtení stránky změní. Postup volání jednotlivých funkcí při načtení stránky je zobrazen na *Obr. 10*.



*Obr. 10. Diagram aktivit startu aplikace rPlayer.*

#### 4.3.1.1 Spuštění programu

Nejprve se nastaví proměnné *soapBeg* a *soapEnd*, které se používají při vytváření volání webové služby. Pak se aplikace pokusí načíst url z Cookies pomocí funkce *getCookieUrl()*. Pokud tam je, zkusí se připojit pomocí funkce *CheckServer()*, pokud ne spustí funkci *HideMe()*, která nastaví aplikaci do odpojeného stavu.

Funkce *CheckServer()*, zkontroluje zda se dá připojit k serveru tím, že vytvoří nejjednodušší webovou službu *check*, která nemá žádný parametr požadavku ani žádnou odpověď a tu pošle na danou adresu. V případě úspěšného připojení, se vrátí status 202, který znamená, že vše proběhlo v pořádku. Adresa serveru se uloží do cookies a pomocí

funkce *ShowMe()* se zobrazí a spustí aplikace. Pokud se připojit nepodaří, proměnná s adresou se vymaže a zobrazí se přihlašovací stránka pomocí funkce *HideMe()*.

#### 4.3.1.2 Vytvoření požadavku webové služby

Pro vytváření požadavků se využívá JavaScriptové funkce *XMLHttpRequest()*. Tu nejprve nastavíme pomocí metody *Open('POST', url, true)*, kde se nastaví že se na server budou posílat data ke zpracování, na jakou adresu se to bude posílat a že to bude asynchronní připojení. Poté se vytvoří xml požadavek, popsáný v kapitole 4.1.1. Jelikož jsou proměnné *soapBeg* a *soapEnd* přednastavené od začátku aplikace, stačí pouze doplnit tělo tohoto požadavku. Poté se nastaví metoda *onreadystatechange*, která definuje jaká funkce se má volat po přijetí odpovědi. Tato funkce se vyplní na místě a ve většině případu v ní je pouze vyskočení dialogového okna při chybě. Nakonec se doplní hlavička požadavku pomocí metody *setRequestHeader()*, nastaví se timeout a požadavek se odešle.

#### 4.3.1.3 Zobrazení přihlašovací stránky nebo přehrávače

K zobrazení přihlašovací stránky se využívá funkce *HideMe()*, která nejprve zastaví Timer, nastaví textová pole k výzvě zadání adresy serveru, zjistí si šířku stránky a do formuláře vloží na první řádek textové pole pro zadání adresy a na druhý řádek tlačítko, které při stisknutí spustí funkci *ConnectMe()*. Tato funkce nastaví zadanou adresu do proměnné *url* a spustí funkce *CheckServer()*, která se postará o připojení.

K zobrazení přehrávače se využívá funkce *ShowMe()*. Tato funkce nejprve nastaví Timer pomocí funkce *setInterval()* a to tak, aby se každou vteřinu spouštěla funkce *getInfo()*. Dále se zjistí šířka a výška stránky, aby se dala spočítat výška a šířka jednotlivých tlačítek. Nakonec se vygeneruje html kód pro zobrazení deseti tlačítek ve dvou řádcích a v každém tlačítku je přiřazena jedna funkce při stisknutí.

#### 4.3.1.4 Získání informací o přehrávání

K získání aktuálních informací o přehrávání slouží funkce *getInfo()*. Jak už bylo zmíněno, tato funkce se volá každou vteřinu, aby se správně aktualizoval čas přehrávání a jejím účelem je vypisovat aktuální informace na první dva řádky aplikace. Vytvoření požadavku webové služby *getinfo* je stejné jako u všech ostatních webových služeb. Jediný rozdíl je v tom, že tato služba jako jediná zpracovává odpověď od serveru. Pokud se vrátí chyba,

pak se chová stejně jako funkce *CheckServer()* a zavolá funkci *HideMe()* pro zobrazení přihlašovací stránky. Jinak zpracovává xml odpověď pomocí metod v *responseXML* následovně.

Na první řádek, s id "songName", se vloží název písničky z proměnné *songname*. Tento řetězec je už je od serveru zpracován pro zobrazení v hlavním textovém poli a je ve formátu: "<interpret> - <název písničky> (<délka písničky>)".

Na druhý řádek, s id "songInfo", se ze zapíše zbývající informace: hlasitost přehrávání, pořadí písničky a celkový počet písniček v playlistu a aktuální čas přehrávání. Informace o zamíchání a opakovaném přehrávání playlistu se vypisují vždy, ale mění se jejich barva podle toho, zda jsou zapnuté či ne. Jednotlivé informace jsou od sebe odděleny třemi až čtyřmi pevnými mezerami.

#### 4.3.1.5 Ovládání přehrávání

K ovládání přehrávání slouží 9 tlačítek a k nim přiřazených 9 funkcí. Tyto funkce vypadají všechny úplně stejně, kromě rozdílného vyplnění těla požadavku. Přehledný seznam je v *Tab. 17*. Poslední, desáté tlačítko s názvem "Exit", slouží k odhlášení od serveru a volá funkci *hideMe()*.

*Tab. 17. Přehled tlačítek a k nim přiřazených funkcí v aplikaci rPlayer mobile.*

Název tlačítka	Jméno funkce	Webová služba	Parametr ws	Popis funkce tlačítka
<b>Play</b>	play()	makeAction	PLAY	Spuštění přehrávání
<b>Stop</b>	stop()	makeAction	STOP	Zastavení přehrávání
<b>Pause</b>	pause()	makeAction	PAUSE	Pozastavení přehrávání
<b>Prev</b>	prev()	makeAction	PREV	Skok na předchozí písničku
<b>Next</b>	next()	makeAction	NEXT	Skok na další písničku
<b>Vol-</b>	volD()	setVolume	vol - 10	Ubrání hlasitosti o 10%
<b>Vol+</b>	volU()	setVolume	vol + 10	Přidání hlasitosti o 10%
<b>Shuffle</b>	shuffle()	makeAction	SHUFFLE	Zamíchání/Seřazení playlistu
<b>Repeat</b>	repeat()	makeAction	REPEAT	Nastavení opakovaného přehrávání

## 5 UŽIVATELSKÁ DOKUMENTACE

Balíček aplikací rPlay slouží k přehrávání hudby na jiném počítači dostupného ze sítě. Základem je mít nainstalovanou aplikaci rPlayServer na počítači, kde se bude hudba přehrávat. K ovládání přehrávání na tomto počítači slouží aplikace rPlayer nebo rPlayer mobile, které se pouští na zařízení ke kterému má uživatel přístup a zároveň je schopné se přes síť spojit s počítačem kde je nainstalovaný rPlayServer.

### 5.1 rPlayServer

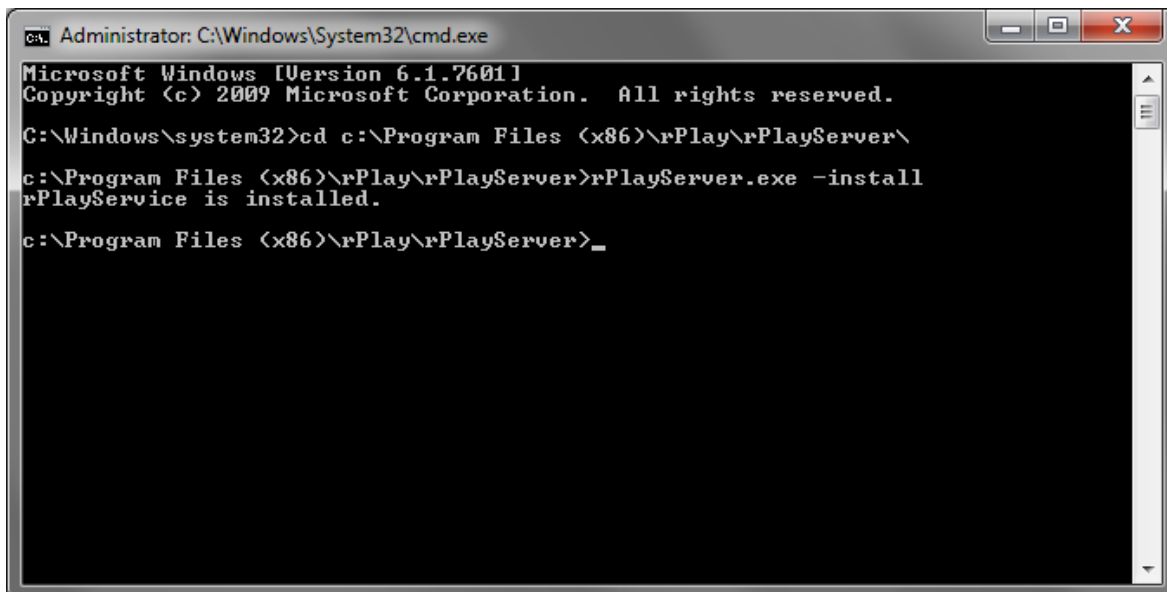
rPlayServer je serverovou částí aplikace, která má na starosti přehrávání hudby na pozadí počítače a spouští se automaticky při startu operačního systému. rPlayServer je určen pro operační systém Windows XP a vyšší.

#### 5.1.1 Instalace

Před samotným nainstalováním, je dobré umístit aplikaci do Počítače na místo, ze kterého se bude moct v budoucnu pouštět. Doporučené umístění je v adresáři "Program Files" vytvořit složku "rPlay", která označuje skupinu programů tohoto balíčku a do toho adresáře pak nakopírovat adresář rPlayServer kde jsou následující soubory:

- rPlayServer.exe - hlavní část programu která se spouští
- README.TXT - pokyny k instalaci a odinstalování
- wxbase294u\_gcc\_cl.dll - knihovna pro wxWidgets
- bass.dll - knihovna pro přehrávání hudby
- tags.dll - knihovna pro načítání tagů z hudebních souborů
- bassflac.dll - podpora přehrávání flac souborů
- basswma.dll - podpora přehrávání wma souborů

Pro nainstalování je potřeba spustit příkazový řádek s právy správce počítače a přejít do adresáře, kde je rPlayServer. Pro nainstalování služby se program spustí s argumentem "-install", jak je vidět na *Obr. 11*. Pokud se vypíše hláška "rPlayService is installed", je služba nainstalovaná. Pro její spuštění je potřeba restartovat počítač nebo službu pustit ručně ze Správce služeb.

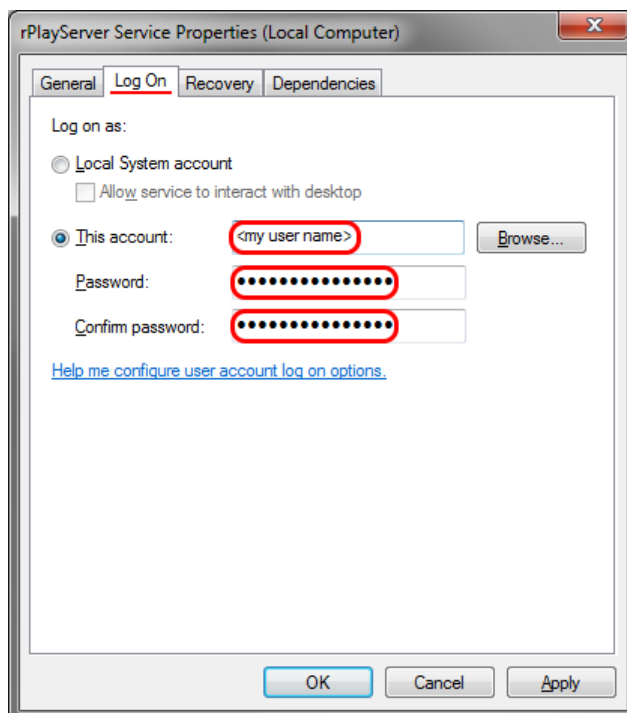


```
Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd c:\Program Files (x86)\rPlay\rPlayServer\
c:\Program Files (x86)\rPlay\rPlayServer>rPlayServer.exe -install
rPlayService is installed.
c:\Program Files (x86)\rPlay\rPlayServer>_
```

Obr. 11. Ukázka instalace služby *rPlayServer* z příkazového řádku.

Aby měla služba přístup k souborům jako uživatel, musí mít i stejná práva. Proto je potřeba u ní nastavit uživatelský účet a heslo, pod kterým se bude spouštět. Nastavení je ve Správci služeb(Obr. 13), ale místo spuštění služby se na ni klikne pravým tlačítkem a dají *vlastnosti*. Zobrazí se nastavení, jak je vidět na Obr. 12., kde se nastaví uživatelské jméno a heslo.

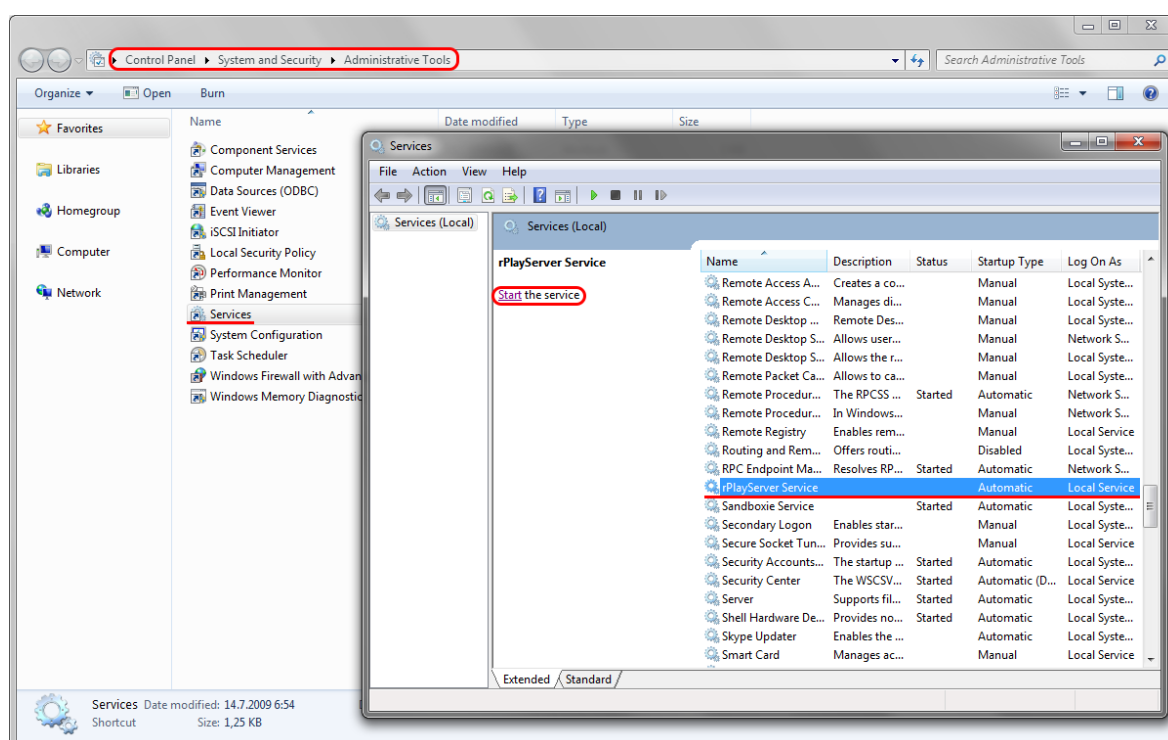


Obr. 12. Nastavení účtu, pod kterým služba *rPlayServer Service* poběží.

Dále je potřeba se ujistit, že ji nebude blokovat firewall a povolit komunikaci po síti pro tuto aplikaci. Pro povolení firewallu se dá spustit aplikace v testovacím režimu a ve většině případů vyskočí okno s dotazem zda se má spojení povolit.

### 5.1.2 Spuštění a zastavení služby rPlayServer

Služba se ovládá přes Správce služeb, který je umístěn v "Kontrolní panel->Systém a zabezpečení->Nástroje pro administraci". Po vybrání položky "rPlayServer Service" lze službu spustit, zastavit nebo restartovat jak je vidět na Obr. 13.



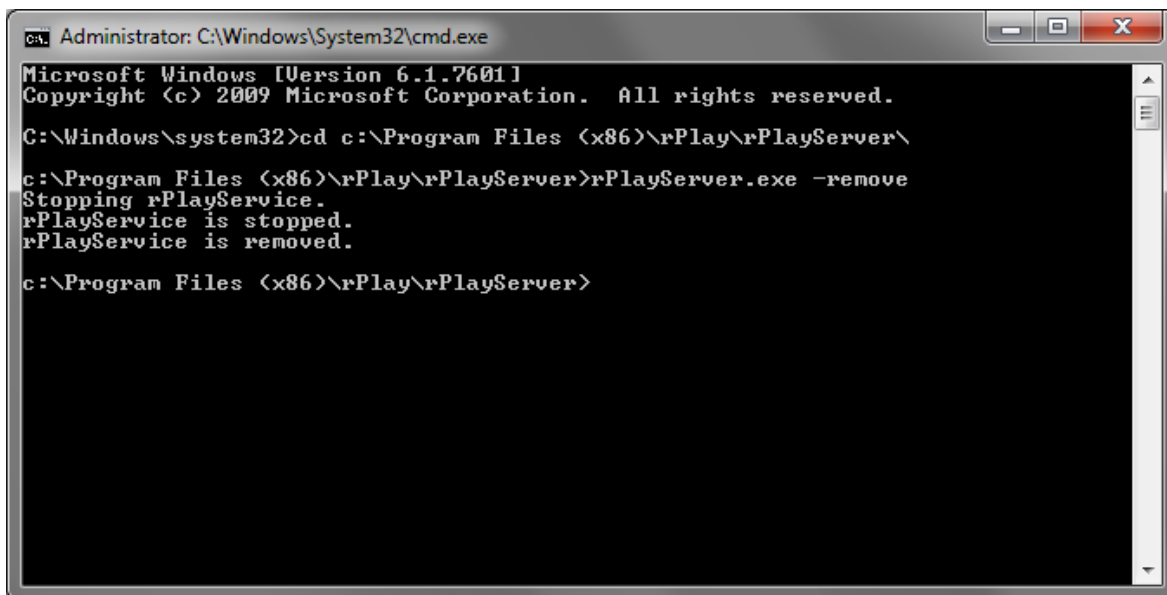
Obr. 13. rPlayServer Service ve správci služeb.

### 5.1.3 Nastavení

Pro nastavení rPlayServeru se používá soubor *rplayserver.cfg*, který je uložen v adresáři "%PROGRAMDATA%\rPlay". Pokud je potřeba něco změnit, musí se to změnit přímo v tomto souboru. rPlayServer z tohoto souboru načítá port na kterém běží, hlasitost a číslo poslední přehrávané písničky.

#### 5.1.4 Odinstalování služby

Pro odinstalování je potřeba spustit příkazový řádek s právy správce počítače a přejít do adresáře, kde je rPlayServer. Pro odebrání služby se program spustí s argumentem "-remove", jak je vidět na *Obr. 14*. Pokud služba běží, program se ji pokusí nejprve zastavit a až potom odebrat. Zároveň se může odebrat adresář s nastavením, který je umístěn "%PROGRAMDATA%\rPlay".



```
Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd c:\Program Files (x86)\rPlay\rPlayServer\
c:\Program Files (x86)\rPlay\rPlayServer>rPlayServer.exe -remove
Stopping rPlayService.
rPlayService is stopped.
rPlayService is removed.

c:\Program Files (x86)\rPlay\rPlayServer>
```

*Obr. 14. Ukázka odebrání služby rPlayServer z příkazového řádku.*

#### 5.1.5 Testovací provoz

Za účelem vyzkoušení, nebo testování lze aplikaci spustit bez potřeby instalace služby a bez správcovských práv. Pokud je potřeba spustit aplikaci v tomto režimu, spouští se s argumentem "-test".

## 5.2 rPlayer

rPlayer je klientskou částí aplikace a má na starosti ovládání přehrávání, vytváření playlistu a dalších nastavení, které rPlayServer nabízí přes webové služby. rPlayer je určen pro operační systém Windows XP a vyšší.

### 5.2.1 Instalace

Tento program nepotřebuje žádnou instalaci a může se rovnou spouštět. Ale jak bylo napsáno výše, tak je doporučeno nakopírovat program do složky "Program Files\rPlay\rPlayer". Tato složka by měla obsahovat následující soubory:

- rPlayer.exe - hlavní část programu která se spouští
- README.TXT - doporučené poznámky před spuštěním
- wxbase294u\_gcc\_cl.dll - knihovna pro wxWidgets
- wxmsw294u\_core\_gcc\_cl.dll - knihovna s GUI pro wxWidgets

### 5.2.2 Ovládání




Při spuštění aplikace se nejprve pokusí připojit k serveru, pokud se to nepodaří, zobrazí se přihlašovací panel (*Obr. 14*), kde se dá změnit nastavení.



*Obr. 15. Ukázka přihlašovacího panelu rPlayeru.*

rPlayer se dá posouvat při tažení myši na textem o připojení k serveru. Popis jednotlivých elementů přihlašovacího panelu je v *Tab. 18*.

*Tab. 18. Popis přihlašovacího panelu z Obr. 14.*

http:/...	Adresa ke které se aplikace připojuje
	Připojení aplikace k serveru
	Nastavení aplikace
	Ukončení aplikace

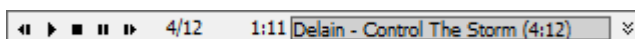
po stisku tlačítka nastavení aplikace se otevře konfigurační soubor v notepadu. V tomto souboru se dají měnit tyto proměnné:

- *addr* - Adresa kam se rPlayer připojuje. Výchozí nastavení je "http://localhost". Pro připojení na jiný počítač je potřeba zadat jeho síťový název ("\\JINY-POCITAC") nebo jeho ip adresu ("http://192.168.1.2").
- *port* - Port na kterém služba běží. Výchozí hodnota tohoto portu je 53191 a ve většině případů se nemusí měnit.

- *posx* - x-ová souřadnice pozice aplikace na obrazovce.
- *posy* - y-ová souřadnice pozice aplikace na obrazovce.
- *autc* - Automatické zkoušení připojování k serveru každých 10 vteřin.

Aby se nastavení načetlo, je nutné notepad uložit a zavřít.

Při připojení aplikace k serveru se zobrazí hlavní ovládací panel (Obr. 15) a rPlayer začne načítat informace.



Obr. 16. Ukázka hlavního panelu rPlayeru.

Posun aplikace se dá docílit uchopením v místech kde je text a tažením myši. Popis jednotlivých prvků hlavního panelu je v Tab. 19.

Tab. 19. Popis hlavního panelu z Obr. 15.

◀	Skok na předchozí písničku
▶	Spuštění přehrávání
■	Zastavení přehrávání
⏸	Pozastavení přehrávání
⏭	Skok na další písničku
4/12	Zobrazení playlistu
1:11	Čas přehrávání
Delain...	Název interpreta a písničky
⌵	Zobrazení rozšířeného nastavení

Pro zobrazení celé aplikace je zapotřebí stisknout tlačítka rozšířeného nastavení a playlistu. Poté celá aplikace vypadá jak na Obr. 16.



Obr. 17. Ukázka úplně rozevřeného rPlayeru.

Playlist umožňuje spustit danou písničku při dvojkliku nebo vybrání a stisknutí klávesy *Enter*. Pro smazání písniček se vyberou dané písničky v seznamu a stiskne klávesa *Delete*.

Hlavní panel bude zobrazen vždy, když bude rPlayer připojen k serveru, ale panel s rozšířeným nastavením a playlist se mohou libovolně zapínat a vypínat. Popis prvků panelu rozšířeného nastavení je v *Tab. 20*.

Tab. 20. Popis vedlejšího panelu z Obr. 16.

	Nastavení hlasitosti
	Zamíchání/Seřazení playlistu
	Nastavení opakovaného přehrávání
	Přidání písniček do playlistu
	Vytvoření nového playlistu
	Posun na pozici v písničce
	Odhlášení od serveru
	Nastavení aplikace
	Ukončení aplikace

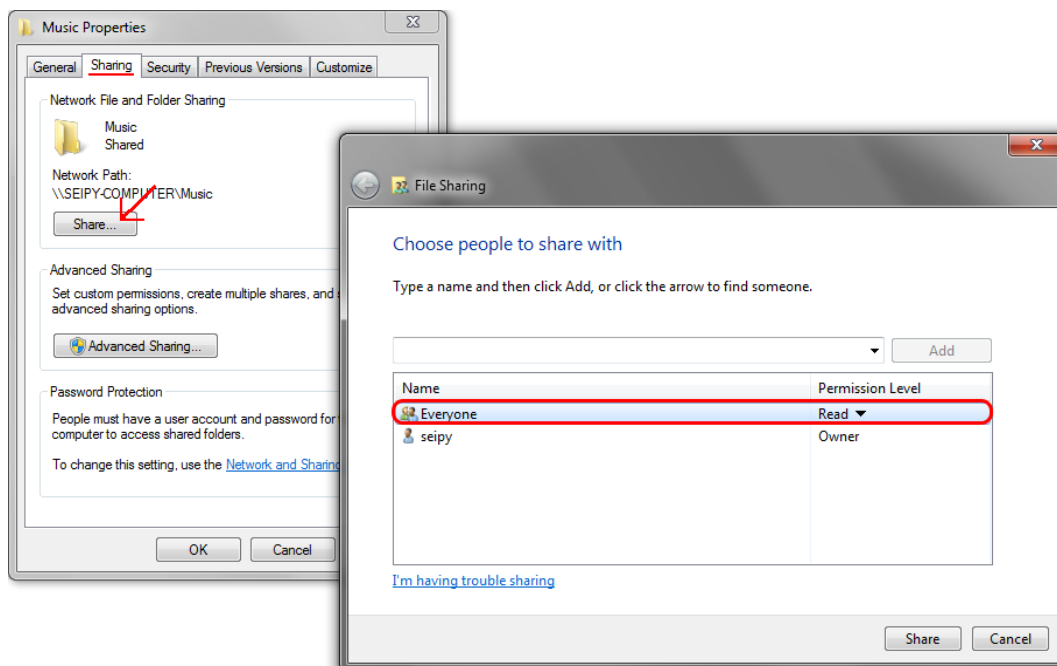
Při stisknutí tlačítka na *zamíchání playlistu*, se zamíchají písničky v playlistu a začne přehrávat od první písničky. To že je playlist zamíchán se pozná podle toho, že tlačítko zůstane stisknuté. Pro seřazení písniček se tlačítko stiskne znovu. Řazení písniček je podle cesty k souboru a názvu souboru.

Stejně jako u tlačítka *zamíchání playlistu*, i při zapnutém opakovaném přehrávání playlistu, zůstane tlačítko *opakované přehrávání playlistu* zatlačené.

### 5.2.3 Sdílení adresářů pro přehrávání

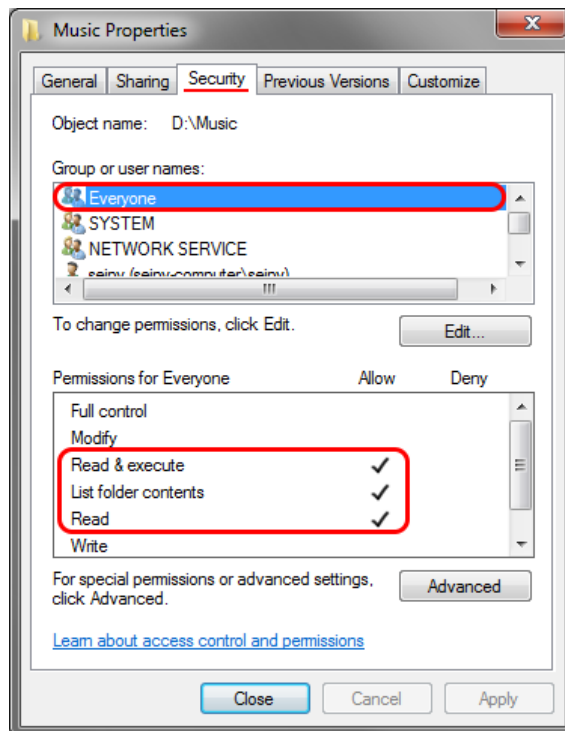
Při vkládání písniček do playlistu je zapotřebí myslet na to, aby se k nim dalo dostat ze sítě. To kontroluje samotná aplikace a nedovolí spustit písničku, která není sdílená v síti. Postup pro sdílení adresáře v síti je následující:

1. Nejdříve se musí adresář nasdílet ve vlastnostech adresáře a nastavit sdílení pro všechny, podle *Obr. 17*.



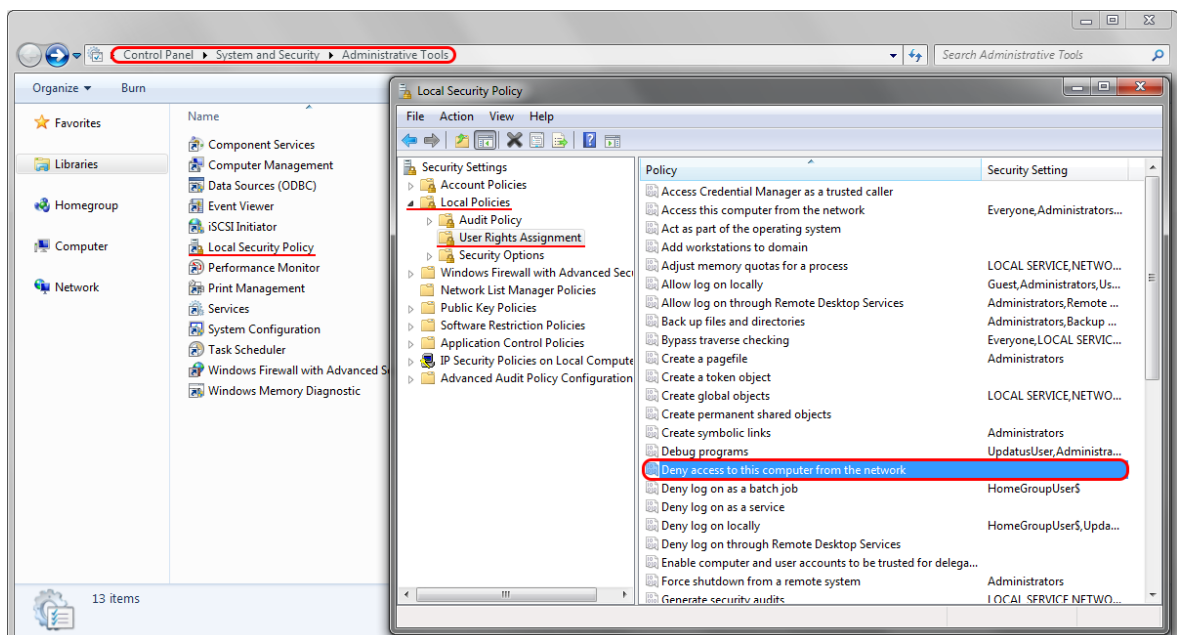
*Obr. 18. Nastavení sdílení adresáře.*

2. Ve vlastnostech zabezpečení musí mít každý právo na čtení a spouštění jak je zobrazeno na *Obr. 18*.



*Obr. 19. Nastavení zabezpečení adresáře*

3. V místních zásadách zabezpečení v položce "Odepřít přístup k tomuto počítači ze sítě" nesmí být žádný uživatel, jak je zobrazeno na *Obr. 19*.



*Obr. 20. Nastavení přístupu k počítači ze sítě.*

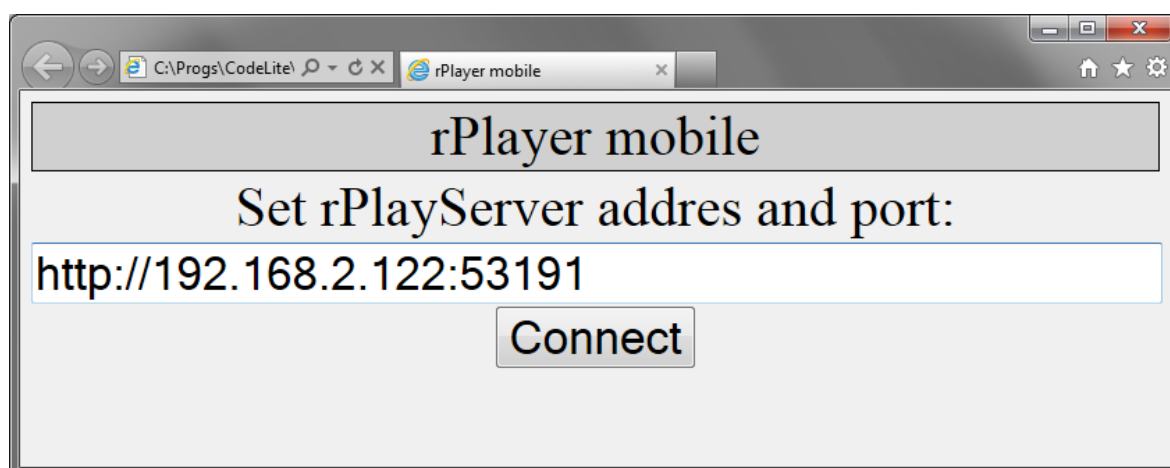
### 5.2.4 Odstranění aplikace

Kromě smazání adresáře se spouštěnou aplikací rPlayer je zapotřebí také smazat adresář "%APPDATA%\rPlay", kam se ukládalo nastavení aplikace.

## 5.3 rPlayer mobile

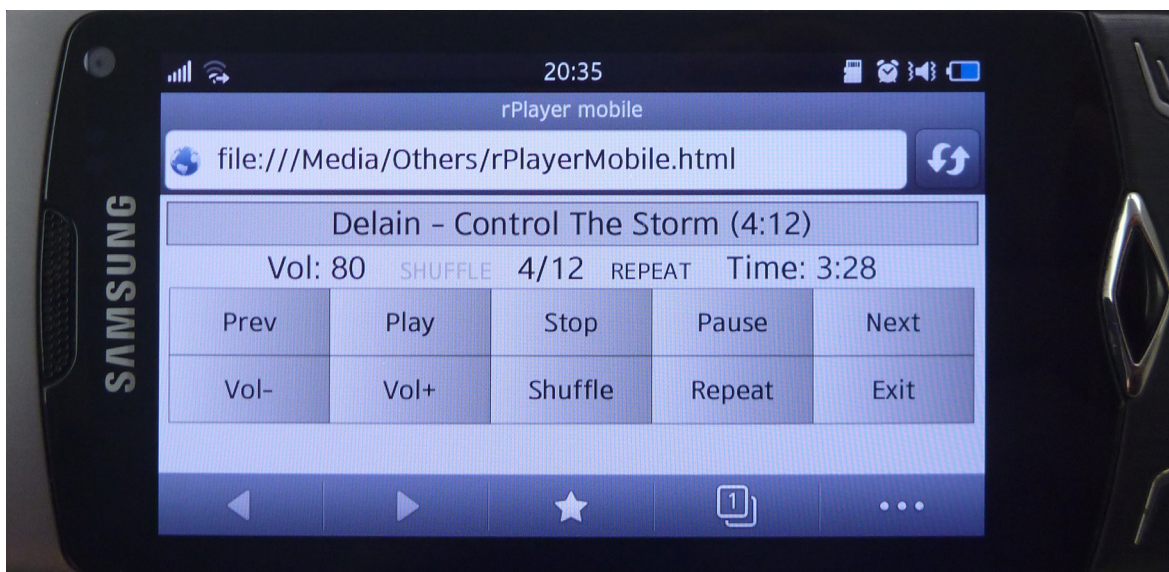
rPlayer mobile je odlehčenou verzí rPlayeru spouštěnou v prohlížeči, která nabízí ovládání přehrávání a zobrazení některých informací o písničce. Tato aplikace je určena především pro mobilní zařízení. rPlayer mobile funguje pouze v prohlížečích, které nepodporují tzv. "Preflighted requests" s hlavičkou OPTIONS[18].

Při otevření webové stránky se zobrazí přihlašovací stránka, ve které se zadá adresa serveru ve formátu "<adresa serveru>:<port>". Ukázka přihlašovací stránky rPlayeru mobile v internet Exploreru 9 je na *Obr. 17*.



*Obr. 21. Ukázka přihlašovací stránky rPlayer mobile v Internet Exploreru 9.*

Po přihlášení k serveru se zobrazí stránka, kde na prvním řádku je zobrazeno jméno interpreta, název písničky a délka písničky. Na druhém řádku jsou další informace o přehrávání, jako hlasitost, aktuální čas písničky, pořadí přehrávané písničky, celkový počet písniček v playlistu a zobrazení zda je playlist zamíchán či seřazen a zda se přehrávání playlistu bude opakovat. Na dalších dvou řádcích je 10 tlačítek, pomocí kterých se dá rPlayServer ovládat. Na *Obr. 18* je zobrazen rPlayer mobile připojený na server v mobilním prohlížeči Samsung Dolfin Browser v3.0.



*Obr. 22. Ukázka připojeného rPlayer mobile v mobilním telefonu.*

Pokud se povede připojit k serveru a v prohlížeči jsou povolené cookies, tak se adresa uloží do cookies a při dalším přihlášení není potřeba adresu znova zadávat a stránka se pokusí automaticky připojit.

## ZÁVĚR

Cílem této diplomové práce bylo naprogramování hudebního přehrávače ovládaného přes webové služby. Celý hudební přehrávač, pojmenovaný rPlay, se skládá ze tří částí.

První částí je samotný hudební přehrávač, pojmenovaný rPlayServer, protože plní úlohu serveru který přehrává hudbu. Tato aplikace se instaluje do Windows jako služba a při spuštění aktivuje server, na kterém běží webové služby a přes které se ovládá přehrávání hudby.

Druhou částí je rPlayer, klientská část aplikace, s grafickým uživatelským rozhraním pro přehrávání hudby, která se po zadání adresy serveru k němu připojí a může ho ovládat. rPlayer umožňuje jak přehrávání a ovládání hudby, tak vytváření playlistů.

Poslední částí je webová stránka rPlayer mobile určená především pro mobilní zařízení. Tato verze rPlayeru umožňuje pouze přehrávání hudby.

Aplikace byly testovány na dvou různých místech, doma a ve firmě, a pod operačním systémem Windows 7 a Windows XP. V provozu aplikace fungovala bez vážnějších problémů. rPlayer se občas zasekne na pár vteřin z důvodu čekání na odpověď webové služby, během které není schopen reagovat, avšak po vypršení timeoutu se zase zprovozní. U mobilní verze stačí dát v tomto případě refresh stránky.

V budoucnu by bylo dobré vylepšit způsob přehráváním písniček z jiného počítače, protože nastavování sdílené složky Windows tak, aby k ní mohl přistupovat rPlayServer i z jiného počítače není z bezpečnostního hlediska nejlepší řešení a z uživatelského hlediska je to velice náročné. Pak by bylo také vhodné vylepšit rPlayer mobile tak, aby fungoval ve všech prohlížečích.

Dále by se samozřejmě mohl vylepšit rPlayServer a jeho webové služby tak, aby poskytovaly další možnosti, jako např. podpora více playlistů, equalizér, přehrávání internetového rádia, atp..

## CONCLUSION

The objective of this thesis was to create a music player controlled via Web Services. The entire music player named rPlay consists of three parts.

The first part is the music player named rPlayServer because it acts as a server that plays music. This application installs itself as a service to the Windows and when it is activated, starts running the Web service and that is used to control music playback.

The second part is rPlayer. The client's part of the application with a graphical user interface for playing music. After entering the server address it connects and controls the server. rPlayer plays music, controls music and creates playlists.

The last part of this application is rPlayer mobile website, which is designed especially for mobile devices. This version can only play music.

Applications were tested in two different places, at home and in the company, and under the operating system Windows 7 and Windows XP. During the operation, the application worked without major problems. rPlayer sometimes gets stuck for a few seconds due to a waiting for a reply from web service, during which it is unable to respond, but after a timeout expires it will start working again. In the mobile version just push refresh button in this case.

In the future, it would be good to improve a method of playing songs from another computer, because Windows's shared folder is hard to set that the rPlayServer could access it from another computer. It is not the best solution from a security point of view and it is very difficult from a user perspective. Then it would also be appropriate to improve rPlayer mobile to work in all browsers.

Furthermore rPlayServer and Web services could be upgraded to provide additional options, such as support for multiple playlists, equalizer, playback, Internet radio, etc..

**SEZNAM POUŽITÉ LITERATURY**

- [1] ARLOW, Jim a Ila NEUSTADT. UML a unifikovaný proces vývoje aplikací: průvodce analýzou a návrhem objektově orientovaného softwaru. Vyd. 1. Brno: Computer Press, 2003, xviii, 387 s. ISBN 80-722-6947-X.
- [2] CORMEN, Thomas H. Introduction to algorithms. 3rd ed. Cambridge: MIT Press, c2009, xix, 1292 s. ISBN 978-0-262-03384-8.
- [3] KANISOVÁ, Hana a Miroslav MÜLLER. UML srozumitelně. 2. aktualiz. vyd. Brno: Computer Press, 2006, 176 s. ISBN 80-251-1083-4.
- [4] PRATA, Stephen. Mistrovství v C. 3. aktualiz. vyd. Překlad Boris Sokol. Brno: Computer Press, 2007, 1119 s. ISBN 978-80-251-1749-1.
- [5] SMART, Julian. Cross-platform GUI programming with wxWidgets. Upper Saddle River: Prentice-Hall, c2006, xxxv, 700 s. ISBN 01-314-7381-6.
- [6] VAN ENGELEN, Robert. GSOAP: 2.8.14 User Guide. Fsu.edu [online]. GENIVIA INC, 2000, 3.2.2013. Dostupné z: <http://www.cs.fsu.edu/~engelen/soap.html>
- [7] UN4SEEN DEVELOPMENTS. BASS documentation: 2.4 [online]. un4seen.com, 2012. Dostupné z: <http://www.un4seen.com/doc/#bass/bass.html>
- [8] VEER, By Emily A. Vander. *JavaScript™ For Dummies®*. 4th ed. Hoboken: John Wiley, 2004. ISBN 07-645-8407-3.
- [9] FRASER, Stephen. *Pro Visual C /CLI and the .NET 3.5 Platform*. New York: Distributed to book the trade by Springer-Verlag, c2009, xxx, 1048 p. Expert's voice in .NET. ISBN 14-302-1053-2.
- [10] ASOKE K. TALUKDER, Asoke K.Hasan Ahmed. *Mobile computing: technology, applications, and service creation*. 2nd ed. New Delhi: Tata McGraw Hill, 2010. ISBN 00-701-4457-5.
- [11] Pulzně kódová modulace. *Wikipedia* [online]. 2005, 8.3.2013 [cit. 20.5.2013]. Dostupné z: [http://cs.wikipedia.org/wiki/Pulzně\\_kódová\\_modulace](http://cs.wikipedia.org/wiki/Pulzně_kódová_modulace)
- [12] MP3. *Wikipedia* [online]. 2005, 12.3.2013 [cit. 20.5.2013]. Dostupné z: <http://cs.wikipedia.org/wiki/MP3>

- [13] Webová služba. *Wikipedia* [online]. 2006, 10.3.2013 [cit. 20.5.2013]. Dostupné z: [http://cs.wikipedia.org/wiki/Webová\\_služba](http://cs.wikipedia.org/wiki/Webová_služba)
- [14] SOAP. *Wikipedia* [online]. 2006, 13.5.2013 [cit. 20.5.2013]. Dostupné z: <http://cs.wikipedia.org/wiki/SOAP>
- [15] WxWidgets. *Wikipedia* [online]. 2007, 9.5.2013 [cit. 20.5.2013]. Dostupné z: <http://cs.wikipedia.org/wiki/WxWidgets>
- [16] A basic Windows service in C++ (CppWindowsService). *MSDN* [online]. 2008, 2.3.2012 [cit. 20.5.2013]. Dostupné z: <http://code.msdn.microsoft.com/windowsdesktop/CppWindowsService-cacf4948>
- [17] Where Should I Store my Data and Configuration Files if I Target Multiple OS Versions?. In: *MSDN Blogs* [online]. 2010 [cit. 20.5.2013]. Dostupné z: <http://blogs.msdn.com/b/patricka/archive/2010/03/18/where-should-i-store-my-data-and-configuration-files-if-i-target-multiple-os-versions.aspx>
- [18] HTTP access control (CORS). *Mozilla developer* [online]. 2008, 6.5.2013 [cit. 20.5.2013]. Dostupné z: [https://developer.mozilla.org/en-US/docs/HTTP/Access\\_control\\_CORS?redirectlocale=en-US&redirectslug=HTTP\\_access\\_control#Preflighted\\_requests](https://developer.mozilla.org/en-US/docs/HTTP/Access_control_CORS?redirectlocale=en-US&redirectslug=HTTP_access_control#Preflighted_requests)

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

PCM	Pulzně kódová modulace (Pulse-code modulation).
A/D	Analogově digitální.
D/A	Digitálně analogový.
ISDN	Digitální síť integrovaných služeb (Integrated Services Digital Network).
FLAC	Bezstrátový hudební kodek zdarma (Free Lossless Audio Codec)
ALAC	Bezstrátový hudební kodek Apple (Apple Lossless Audio Codec)
CD	Kompaktní disk (Compact Disk)
SOAP	Protokol založený na jednoduchých zprávách (Simple Object Access Protocol)
WSDL	Jazyk popisující webové služby (Web Services Description Language)
XML	Rozšiřitelný značkovací jazyk (Extensible Markup Language)
HTTP	Protokol pro výměnu hypertextových dokumentů (Hypertext Transfer Protocol)
RPC	Vzdálené volání procedur (Remote procedure call)
API	Rozhraní pro programování aplikací (Application Programming Interface)
C	programovací jazyk
C++	Objektově orientovaný programovací jazyk
WS	Webová služba (Web Service)
URI	Jednotný identifikátor zdroje (Uniform Resource Identifier)
FTP	Protokol pro přenos souborů (File Transfer Protocol)
GUI	Grafické uživatelské rozhraní (Graphical User Interface)

**SEZNAM OBRÁZKŮ**

<i>Obr. 1. Ukázka vývojového prostředí Codelite</i> .....	21
<i>Obr. 2. Zobrazení částí aplikace</i> .....	24
<i>Obr. 3. Struktura rPlayServeru</i> .....	25
<i>Obr. 4. Diagram aktivit funkce prev()</i> .....	33
<i>Obr. 5. Diagram aktivit funkce next()</i> .....	33
<i>Obr. 6. Diagram aktivit funkce shuffle()</i> .....	34
<i>Obr. 7. Ukázka návrhu GUI v programu wxFormBuilder.</i> .....	44
<i>Obr. 8. Diagram aktivit funkce onTimer()</i> .....	46
<i>Obr. 9. Zjednodušený diagram aktivit funkce getUncPath()</i> .....	49
<i>Obr. 10. Diagram aktivit startu aplikace rPlayer.</i> .....	52
<i>Obr. 11. Ukázka instalace služby rPlayServer z příkazového řádku.</i> .....	56
<i>Obr. 12. Nastavení účtu, pod kterým služba rPlayServer Service poběží.</i> .....	56
<i>Obr. 13. rPlayServer Service ve správci služeb.</i> .....	57
<i>Obr. 14. Ukázka odebrání služby rPlayServer z příkazového řádku.</i> .....	58
<i>Obr. 15. Ukázka přihlašovacího panelu rPlayeru.</i> .....	59
<i>Obr. 16. Ukázka hlavního panelu rPlayeru.</i> .....	60
<i>Obr. 17. Ukázka úplně rozevřeného rPlayeru.</i> .....	61
<i>Obr. 18. Nastavení sdílení adresáře.</i> .....	62
<i>Obr. 19. Nastavení zabezpečení adresáře</i> .....	63
<i>Obr. 20. Nastavení přístupu k počítači ze sítě.</i> .....	63
<i>Obr. 21. Ukázka přihlašovací stránky rPlayer mobile v Internet Exploreru 9.</i> .....	64
<i>Obr. 22. Ukázka připojeného rPlayer mobile v mobilním telefonu.</i> .....	65

**SEZNAM TABULEK**

<i>Tab. 1. Ukázka nastavení ws v hlavičkovém souboru.</i>	26
<i>Tab. 2. Definování webové služby v hlavičkovém souboru.</i>	27
<i>Tab. 3. Ukázka požadavku a odpovědi ws check.</i>	27
<i>Tab. 4. Ukázka požadavku a odpovědi ws getinfo.</i>	28
<i>Tab. 5. Ukázka požadavku a odpovědi ws makeAction.</i>	28
<i>Tab. 6. Ukázka požadavku a odpovědi ws playSong.</i>	29
<i>Tab. 7. Ukázka požadavku a odpovědi ws getplaylist.</i>	29
<i>Tab. 8. Ukázka požadavku a odpovědi ws addsongs.</i>	30
<i>Tab. 9. Ukázka požadavku a odpovědi ws remsongs.</i>	30
<i>Tab. 10. Ukázka požadavku a odpovědi ws setVolume.</i>	30
<i>Tab. 11. Ukázka požadavku a odpovědi ws setPosition.</i>	31
<i>Tab. 12. Speciální identifikátory používající se ve funkci TAGS_Read()</i>	35
<i>Tab. 13. Speciální výrazy používající se ve funkci TAGS_Read()</i>	36
<i>Tab. 14. Seznam proměnných ve struktuře rp__song.</i>	48
<i>Tab. 15. Legenda k diagramu aktivit funkce getUncPath() (Obr. 9).</i>	50
<i>Tab. 16. Zdrojový kód html aplikace rPlayer mobile.</i>	51
<i>Tab. 17. Přehled tlačítek a k nim přiřazených funkcí v aplikaci rPlayer mobile.</i>	54
<i>Tab. 18. Popis přihlašovacího panelu z Obr. 14.</i>	59
<i>Tab. 19. Popis hlavního panelu z Obr. 15.</i>	60
<i>Tab. 20. Popis vedlejšího panelu z Obr. 16.</i>	61

## SEZNAM PŘÍLOH

Příloha 1. CD se zdrojovými kódy a spustitelnou verzí aplikace.