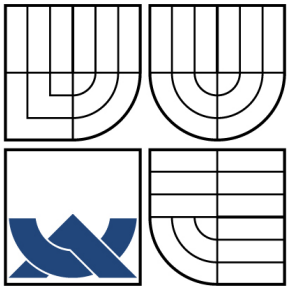
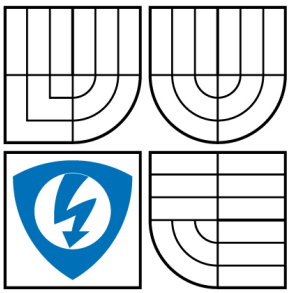


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

APLIKACE PRO PDA UMOŽŇUJÍCÍ PŘÍJEM A PRÁCI S MULTIMEDIÁLNÍM OBSAHEM

APPLICATION FOR PDA FOR MULTIMEDIA STREAM PRESENTATION AND PROCESSING

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

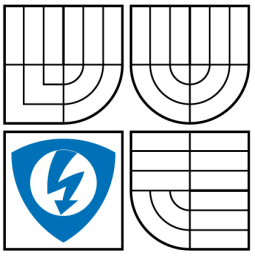
AUTOR PRÁCE — Bc. MARTIN JAVORČEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. JAKUB MÜLLER

SUPERVISOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Martin Javorček	***ID:***	83699
Ročník:	2	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Aplikace pro PDA umožňující příjem a práci s multimediálním obsahem

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s možnostmi platformy .NET Compact Framework a návrhem aplikací pro mobilní zařízení v jazyce C#. Navrhněte vhodnou realizaci aplikace běžící na PDA pod operačním systémem Windows Mobile 6.x, která by následně dokázala příjmout a přehrávat multimediální obsah.

DOPORUČENÁ LITERATURA:

[1] BURGET, R., KOMOSNY, D. Real-time control protocol and its improvements for Internet Protocol Television. International Transaction on Computer Science and Engineering, ISSN 1738-6438, 2006, roč. 2006, č. 31, s. 1 - 12.
[2] KOMOSNY D., NOVOTNY V. Tree Structure for Source-Specific Multicast with feedback Aggregation, in ICN07 - The Sixth International Conference on Networking . Martinique, 2007, ISBN 0-7695-2805-8.
[3] A. Wigley , D. Moth, P. Foot ,Microsoft® Mobile Development Handbook 2007, Pages 688, ISBN 9780735623583.

Termín zadání:	9.2.2009	***Termín odevzdání:***	26.5.2009

Vedoucí práce: Ing. Jakub Müller

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práve třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

ABSTRAKT

Předložená práce se zabývá vytvořením aplikace pro PDA umožňující příjem a práci s multimediálním obsahem. V úvodu práce je pozornost věnována hlavně teoretickému vysvětlení některých pojmů jako je PDA (Personál Digital Assistant) a Windows Mobile 6.0. Čtenář se obeznámí také s technologií .NET Framework a taktéž s její verzí pro mobilní zařízení .NET Compact Framework. Jsou tady rozebrány výhody a nevýhody této technologie, ale taktéž výhody a nevýhody programovacího jazyka C#, pod kterým je aplikace naprogramována. Následující kapitoly práce se věnují samotnému návrhu multimediální aplikace a to konkrétně návrhu z hlediska tříd. Pro lepší orientaci a pochopení je použitý UML diagram. UML (Unified Modeling Language) je grafický jazyk, který je určen pro vizualizaci a dokumentaci programových systémů. Kromě obecního popisu tříd se tady nachází podrobné vysvětlení jednotlivých atributů a metod aplikace. Je vysvětlená jejich funkce a účel. V další části práce se pozornost zaměřuje na síťovou komunikaci aplikace, či už z hlediska přenášení jednotlivých multimediálních souborů nebo z hlediska přenosu v reálném čase. Rozebírá se tady princip navazování a ukončování spojení mezi klientem a serverem. Klientská část aplikace je navrhnutá na mobilní zařízení s operačním systémem Microsoft Windows Mobile 6.0 a serverová část aplikace je navrhnutá pro počítač na kterém je nainstalován operační systém Microsoft Windows ve verzi XP. V práci můžeme také najít jednoduchý manuál k ovládaní obou aplikací. Taktéž jsou v této části popsány výjimky, které mohou nastat v případe výpadku spojení při komunikaci. Poslední kapitoly popisují synchronizaci mobilního zařízení, popřípadě emulátoru se stolovým počítačem a dávají čtenáři přehled o podporovaných formátech audio souborů multimediální aplikaci. Všechny dosažené výsledky jsou shrnuty v samotném závěru práce.

KLÍČOVÁ SLOVA

.NET Framework, C#, PDA, klient, server, přehrávač

ABSTRACT

This thesis deals with creating application for PDA for multimedia stream presentation and processing. Introduction of the thesis is devoted to the theoretical explanation of terms such as PDA (Personal Digital Assistant) and Windows Mobile 6.0. Readers will be acquainted with the technology of .NET Framework and also with its version for mobile equipment .NET Compact Framework. Advantages as well as disadvantages of .NET technology are dealt with in this section together with the same phenomena of C# programming language, which is the basic language used for running our application. Next chapters of the thesis deal with my own design of multimedia application, especially from the point of view of classes. For better orientation and understanding UML diagram is used. UML (Unified Modeling Language) is a graphic language intended for visualization and documentation of programming systems. Except of general description of the classes there is also a detailed explanation of individual attributes and application methods. Their function and purpose are explained here as well. In the next part of the thesis attention is paid to the network communication of the application, both in terms of individual multimedia files transmission and stream transmission. Principles of connection and disconnection between client and server are analyzed in this part. Client' s part of the application is designed for the mobile equipment with Microsoft Windows Mobile 6.0 operation system and server's part of application is designed for computers, where Microsoft Windows XP operation system is installed. A concise manual for operating both the applications is included. This part also describes some exceptions which may occur in case of problems with connection. Last chapters describe synchronization of mobile equipment or emulator with a desktop computer and provide the reader with the summary of supported audio formats in multimedia application. All achieved results are summarized in the concluding part of the thesis.

KEYWORDS

.NET Framework, C#, PDA, client, server, player

BIBLIOGRAFICKÉ CITACE MÉ PRÁCE

JAVORČEK, M. *Aplikace pro PDA umožňující příjem a práci s multimediálním obsahem.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 72 s., Vedoucí diplomové práce Ing. Jakub Müller.

PREHLÁSENIE

Prehlasujem, že svoju diplomovú prácu na tému „Aplikace pro PDA umožňující příjem a práci s multimediálním obsahem“ som vypracoval samostatne pod vedením vedúceho diplomovej práce a s použitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky uvedené v zozname literatúry na konci práce.

Ako autor uvedeného diplomovej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, predovšetkým som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a som si plne vedomí následkov porušenia ustanovenia § 11 a nasledujúcich autorských zákonov č. 121/2000 Zb., vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovení § 152 trestného zákona č. 140/1961 Zb.

V Brne dňa ....

podpis autora

POĎAKOVANIE

Ďakujem vedúcemu diplomovej práce Ing. Jakubovi Müllerovi, za veľmi užitočnú metodickú pomoc a cenné rady pri spracovaní diplomovej práce.

V Brne dňa ....

podpis autora

OBSAH

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ÚVOD

V poslednom čase sa čoraz viac dostávajú do popredia prenosné počítače triedy Handheld a Pocket PC. Je to podmienené hlavne tým, že sa stále rýchlejšie rozvíja mobilná komunikácia a s tým súvisiace služby. O osude a úspešnosti veľkých obchodných transakcií často rozhodujú doslova minúty. Preto každý správny obchodník alebo manažér, ktorý v práci veľa cestuje, musí byť vybavený práve takýmto zariadením, aby vedel včas reagovať na všetky požiadavky, ktoré sú na neho kladené [1].

Tým, že mobilné zariadenia ponúkajú čoraz väčšie možnosti, tým aj aplikácie, ktoré na nich bežia sú stále rozmanitejšie a ponúkajú širší okruh využitia. Jedným druhom aplikácií sú multimediálne aplikácie. Nasledujúca práca sa zaoberá práve vývojom takejto aplikácie, ktorá by dokázala prijímať a pracovať s multimediálnym obsahom. V dôsledku stále zrýchľujúcich sa sietí a prenosov je v posledných rokoch téma multimediálnych aplikácií stále viac a viac spomínaná. Ďalšia časť textu sa zaoberá práve touto tematikou, to znamená multimédiami a ich prehrávaním pod Windows Mobile 6.0, ale tiež popisom programov a prostredí, ktoré budeme k naprogramovaniu aplikácie potrebovať.

Práca sa zameria na možnosti príjmu multimédií pomocou sieťovej komunikácie, či už z hľadiska prenášania jednotlivých súborov a následného prehrávania, ale aj z hľadiska prehrávania v reálnom čase. V texte budú popísané výhody a nevýhody programovacieho jazyka C#, pod ktorým bude aplikácia naprogramovaná. Tiež budú spomenuté možnosti využitia aplikácie a následne jej výhody a nevýhody v praxi. V prvých kapitolách textu sa teoreticky priblíži a rozoberie dôležitá technológia ako .NET Compact Framework, ale reč bude aj o operačnom systéme Windows Mobile. Ďalšia časť sa venuje samotnému zadaniu práce od návrhu až po zrealizovanie a vysvetleniu jednotlivých úsekov zdrojového kódu. Posledná časť textu sa zameria na ovládanie aplikácie z užívateľského hľadiska a návodom na synchronizáciu aplikácie s programom ActiveSync, ktorý slúži ako synchronizačný program pre synchronizáciu zariadenia s počítačom.

1 TEORETICKÉ ZOZNÁMENIE S PRÁCOU

1.1 PDA (Personal Digital Assistant)

Výraz PDA (z anglickej skratky Personal Digital Assistant - alebo osobný digitálny pomocník) býva často prekladaný ako vreckový počítač. PDA slúži predovšetkým k uľahčeniu práce. Okrem vedenia rôznych schôdzok, kontaktných informáciách o osobách alebo plánovania úloh, ponúka PDA tiež mobilný prístup k internetu a e-mailu. Ďalšie využitie je napríklad v podobe používania kancelárskych aplikácií (Microsoft Word, Microsoft Excel a iné) , ktoré sú kompatibilné s bežným osobným počítačom či notebookom. Jednoducho pripojíte vreckový počítač k počítaču a pomocou programového vybavenia spolu môžu obidva komunikovať a prenášať dáta. Ponúka sa veľa zaujímavých programov, pričom PDA prináša tiež veľký zábavný potenciál, napríklad prehrávanie multimediálnych súborov (*.mp3, *.avi, *.jpg) alebo hranie hier.

PDA (vreckový počítač) máva zvyčajne dotykovú obrazovku, ktorá sa ovláda (a tým aj celé zariadenie) špeciálnym dotykovým perom, zvaným stylus. Displej je dotykový z dôvodu, aby bola možnosť vynechať klávesnicu, čím sa zníži hmotnosť a rozmery zariadenia, čím sa z neho stáva ľahký a malý mobilný pomocník.

Čo sa týka operačných systémov, ktorými sú DPA vybavené, v súčasnej dobe sú používané dva aktuálne operačné systémy, ktoré v podstate definujú dve hlavné platformy týchto vreckových počítačov. Prvou z nich je platforma Pocket PC s operačným systémom Microsoft Pocket PC (alebo Windows Mobile), druhá platforma je odvodená od operačného systému Palm. V práci sa ďalej budeme zaoberať už len platformou Pocket PC s operačným systémom Windows Mobile. Pre ilustráciu ako môže v praxi PDA zariadenie vypadať môžeme vidieť na obr. 1.1.

Obr. 1.1. Ukážka PDA zariadenia

1.2 Windows Mobile 6.0

Všetci používatelia stolových počítačov alebo notebookov používajú operačný systém k tomu, aby mohli na počítači pohodlne a rýchlo riešiť všetky úlohy. Nie je tomu inak ani pri používaní vreckových počítačov. Jedna z firiem, ktorá sa pokúša preniknúť do tejto sféry trhu je samozrejme aj Microsoft s jeho operačným systémom Windows Mobile. Windows Mobile je software, ktorý nájdete vo vyspelých mobilných zariadeniach, ktoré vám umožnia posielať e-maily, prechádzať internetom a pracovať s mobilnou verziou softwaru Office. Tento operačný systém sa všeobecne delí na dva druhy. A to Pocket PC Phone edition, ktorý je primárne určený pre mobilné telefóny s dotykovým displejom a smartphone edition, ktorý sa ovláda len klávesnicou. Windows Mobile 6.0 je vďaka tomu, že Microsoft doň implementoval to isté jadro, ako do Windows Mobile 5.0 kompatibilný skoro so všetkými aplikáciami, ktoré bežali pod staršou verziou. Vylepšená je stabilita systému, rýchlosť a novinkou je podpora VoIP.

Nezmenila sa ani základná obrazovka, akurát pribudlo pár nových položiek, napríklad plugin pre aktiváciu internetových volaní alebo Live plugin [3].

Ponuka štart sa tiež výrazne nezmenila, pribudol iba balíček Office Mobile, ktorý umožňuje aj editovať dokumenty Word, Excel, či PowerPoint. Tento balíček sa už napevno stane súčasťou Windows Mobile. Ukážku ako vyzerá Windows Mobile 6.0 môžeme vidieť na obr. 1.2.

Obr. 1.2. Ukážky z Windows Mobile 6.0

1.3 .NET Compact Framework

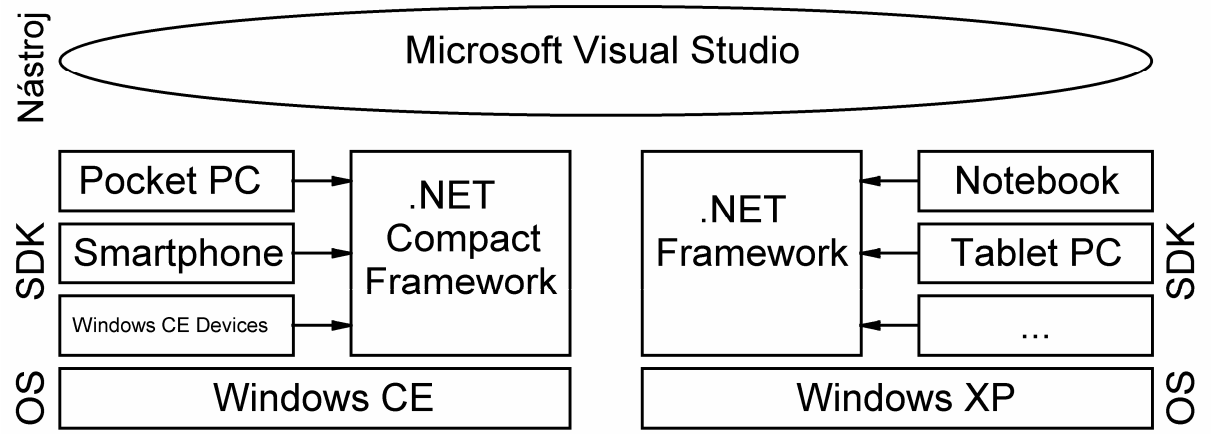
Microsoft .NET Framework je technológia od spoločnosti Microsoft pre vývoj aplikácií, ako pre desktop, tak aj pre web. Hlavným motorom .NET Framework je CLR (Common Language Runtime). Na tomto motore sú prevádzané jednotlivé prvky a ich komunikácia. Takzvane sa tu spravuje bežiaci kód. Celý kód je kompilovaný do MSIL (Microsoft Intermediate Language – obdoba strojového kódu). Platforma dokáže tiež ponúknuť širokú škálu programovacích jazykov. Užívateľ si môže vybrať medzi C# (C Sharp), C++.NET, VB.NET a J# JAVA.

Jednou z verzií .NET Frameworku je .NET Compact Framework, ktorý je určený pre beh na mobilných zariadeniach, ako sú PDA, mobilné telefóny a set-top boxy. Túto technológiu podporuje aj herná konzola Xbox 360. Prvá verzia .NET Compact Framework bola vydaná v druhej polovici roku 2002. Posledná verzia 3.5 bola vydaná 25.januára 2008. Microsoft .NET Compact Framework používa niektoré rovnaké knižnice ako klasický .NET Framework, ale tiež pár knižníc designovaných špeciálne pre mobilné zariadenia.

V prostedí .NET Compact Framework sme teda schopný tvoriť plnohodnotné aplikácie. Compact Framework sa skladá z dvoch častí. Prvou je vývojové prostredie a druhou prostredie pre beh aplikácií. Vývojové prostredie pre vytváranie aplikácií pre mobilné zariadenia je priamo integrované do Visual Studia. Vytvorená aplikácia je potom spúšťaná na Compact Framework Common Language Runtime (CF CLR). Mobilné zariadenia preto musia implementovať toto prostredie. Hlavné vlastnosti Compact Runtime CLR sú rovnaké ako u plnohodnotného .NETu. Medzi tieto vlastnosti patrí:

- typovo bezpečné bežiace prostredie
- garbage collection
- just-in-time kompilácia
- ošetrenie chýb pomocou výnimiek

Pre .NET Compact Framework je teda možné vytvárať aplikácie pomocou nástroja Microsoft Visual Studio, v programovacích jazykoch C# alebo Visual Basic.NET. Keďže projekt je zadaný v programovacom jazyku C#, ďalšie kapitoly sa budú venovať už len tomuto konkrétnemu jazyku. Pre lepšiu predstavu čo všetko budeme k programovaniu potrebovať, respektíve čo všetko Microsoft Visual Studio zastrešuje môžeme vidieť na obr. 1.3.



Obr. 1.3. Prehľad nástrojov a možností pre vývoj aplikácií pod Visual Studiom

1.4 Programovací jazyk C# (C Sharp)

C# (vyslovované anglicky ako C Sharp) je vysoko úrovňový objektovo orientovaný programovací jazyk vyvinutý firmou Microsoft zároveň s platformou .NET Framework. Môžeme povedať, že C# koncepčne vychádza z jazyka C++ a jeho tvorcovia hovoria, že všetky dobré a pokrokové črty jazyka C++ zostali zachované. Zanikli však niektoré nepopulárne črty jazyka, napríklad pointre. Namiesto pointrov sa v jazyku C# používajú odkazy. Programátorom uľahčia život aj ďalšie črty, hlavne Garbage Collection, čo znamená, že sa už nemusíte starať o upratovanie nepotrebných objektov z pamäti. Zdalo by sa teda, že pátranie po predkovi programovacieho jazyka C# má jednoznačné riešenie, ale nie je tomu úplne tak. Je to možno paradox, ale C# má bližšie skôr k programovaciemu jazyku Java, ako k C++. Jazyk C# teda prevzal výhodné črty z viacerých „klasických“ programovacích jazykov, najviac však z Javy a C++. Presne ako vyplýva z jeho názvu, mal by tento programovací jazyk predstavovať akúsi, hudobnou terminológiou povedané „o poltón zvýšenú“ formu než spomínané programovacie jazyky.

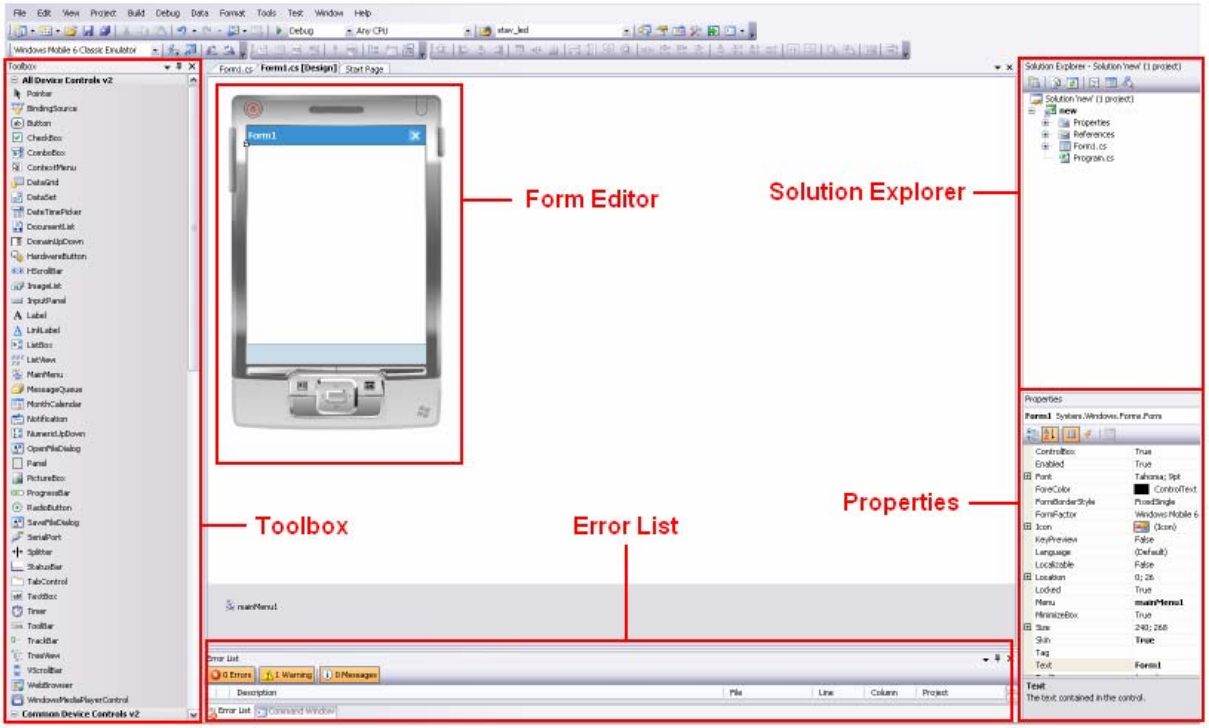
1.5 Popis Visual Studia 2008

Visual Studio 2008 od spoločnosti Microsoft predstavuje plne vybavený systém pre návrh a vývoj aplikácií či už pre desktopové počítače alebo pre mobilné zariadenia v prostredí .NET. Je tu možné tvoriť rýchlo a pohodlne aplikácie všetkých druhou. Ide o kompletné prostredie s editorom zdrojových textov, form editorom, debuggerom a kompilátorom. Microsoft Visual Studio 2008 obsahuje oproti starším verziám niekoľko vylepšení. Jednou z nich je výhoda výberu .NET platformy s ktorou chce programátor pracovať. Je tu možnosť

výberu medzi verziami .NET 2.0, 3.0 a najnovšou 3.5. To znamená, že prechod vývojárov na Visual Studio 2008 neznamená automatický vývoj aplikácií len pre .NET 3.5. Okrem toho, Visual Studio 2008 obsahuje aj iné vylepšenia. Aj keď ich v práci priamo nevyužijeme, stojí za zmienku aspoň novinka ako je napríklad LINQ (Language Integrated Query), čo je technológia umožňujúca vývojárom rýchly a jednoduchý prístup k dátam a prácu s nimi. Ostatné novinky nebudeme spomínať, pretože pri vytváraní multimediálnej aplikácie ich nebudeme využívať.

Ako bolo spomenuté Visual Studio 2008 je plne vybavené prostredie. Na obr. 1.4 môžeme vidieť ukážku prostredia po vytvorení projektu pre platformu Pocket PC. Nájdeme tu Form Editor, Toolbox, Error List, Solution Explorer a Properties. Form Editor môžeme brať tak isto ako pri Win32 aplikáciách. Ide o dizajnový editor, na ktorý môže vývojár ukladať komponenty z Toolboxu a vytvárať tak GUI (Graphical User Interface) danej aplikácie. Už v spomenutom Toolboxe sú všetky druhy komponentov, ktoré sa môžu ukladať do Form Editoru. Komponenty sa menia v závislosti od zvolenej platformy, pod ktorou bude aplikácia naprogramovaná a tiež od verzie .NET. Solution Explorer umožňuje vidieť presne štruktúru celého projektu. Dajú sa tu vytvárať nové triedy, formy a otvárať zdrojové kódy. Jednoducho Solution Explorer umožňuje rýchlejšiu a pohodlnejšiu prácu s projektom. V časti Properties je programátor schopný upravovať vlastnosti jednotlivých komponentov, ktoré vložil do Formu. Tiež okrem vlastností komponentov sa tu dajú nastavovať aj ich udalosti. Poslednou časťou, ktorú na obr. 1.4 môžeme vidieť je Error List. Ten slúži pre vypisovanie chýb a varovaní pri programovaní, ale aj pri dubuggovaní. Prostredie Visual Studia 2008 nemusí však ostať v popísanom tvare. Záleží len na rozhodnutí používateľa ako si sám prostredie upraví podľa svojich potrieb. To znamená, že ďalšie okná si môže pridávať alebo odoberať. Každé okno sa dá tiež zafixovať (Dockable) alebo len voľne položiť na ľubovoľnú časť v prostredí (Floating). Poprípade je tu možnosť automatického skrývania (Auto Hide).

Visual Studio 2008 je naozaj veľmi dobrý nástroj pre vývoj aplikácií, či už pre stolové (desktopové) alebo mobilné zariadenia. Uľahčuje programátorovi prácu a aj preto je vhodné pre vývoj našej multimediálnej aplikácie.



Obr. 1.4. Popis vývojového prostredia Visual Studia 2008

2 MULTIMEDIÁLNA APLIKÁCIA PRE WINDOWS MOBILE 6.X

2.1 Založenie projektu

Multimediálna aplikácia bude realizovaná pod Visual Studiom 2008. Ak spúšťame Visual Studio prvý krát po inštalovaní, objaví sa dialóg s výzvou implicitného nastavenia prostredia. Visual Studio 2008 sa vie automaticky prispôsobiť programovaciemu jazyku, v ktorom chce užívateľ programovať. Rôzne dialógy a nástroje, ktoré sú súčasťou integrovaného vývojového prostredia (IDE), upravia svoj vzhľad a zobrazovanú sadu nástrojov podľa vybraného jazyka. Keďže aplikácia má byť naprogramovaná programovacím jazykom C#, vyberieme z ponuky Visual C# Development Settings.

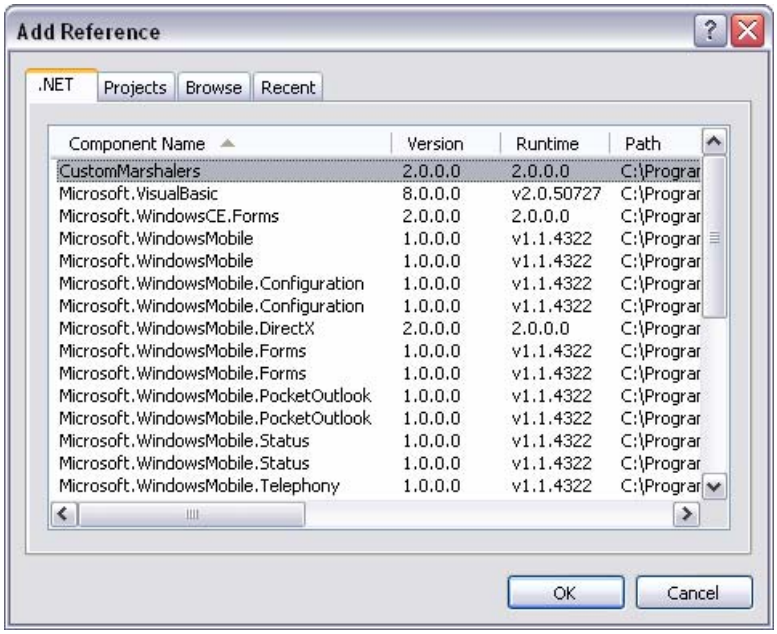
Po otvorení vývojového prostredia z ponuky File vyberieme New a v novo otvorenom menu vyberieme Project. Otvorí sa dialógové okno New Project , v ktorom môžeme vytvoriť nový projekt na základe rôznych šablón. Tiež je tu možné zadať názov projektu a umiestnenie na disku. V našom prípade vyberieme Smart Device Project a potvrdíme tlačidlom OK.

V nasledujúcom dialógovom okne si môže programátor vybrať zo širokej palety zariadení a platforiem, pod ktorou sa rozhodne aplikáciu vytvárať. Je možné vybrať riešenie založené na Windows Mobile 2003, Windows CE, Windows Mobile 5.0 alebo 6.0 pre Smartphone alebo Pocket PC. V našom prípade vyberieme ako cieľovú platformu (Target platform) Windows Mobile 6 a z verzií .NET Compact Frameworku vyberieme verziu 2.0. Na panely Templates už stačí vybrať len Device Application a potvrdiť tlačidlom OK. Tým je všetko pripravené k založeniu nového projektu.

2.2 Knižnice a Add Reference

Po vytvorení nového projektu si prostredie Visual Studia vytvorí triedu Form, ktorá už má na začiatku importované určité menné priestory. Menné priestory (namespace) uľahčujú prácu programátorom hlavne pri veľkých projektoch, kde pri veľkom počte ľudí, ktorí sa na projekte podieľajú, môže dôjsť napríklad k rovnakému pomenovaniu premenných a vznikli by tak zbytočné problémy. Preto sa v jazyku C# používa direktíva `using`, ktorá využíva toho, že práve .NET (CLR Common Language Runtime) používa dôsledne menné priestory. Teda pomocou nej si vieme do projektu pridať potrebné menné priestory.

Keďže sa jedná o multimediálnu aplikáciu, ktorá bude prehrávať audio súbory, musia sa predtým ako sa začne programovať doimportovať do projektu potrebné knižnice. Pre podporu multimédií pre Windows Mobile je to knižnica „WMPLib.dll“. Aby sa mohla použiť direktíva `using` a uskutočniť sa tak import dátových členov menného priestoru, musí sa najprv vložiť do projektu referencia na objektovú knižnicu „WMPLib.dll“. Vo Visual Studiu sa to robí tak, že z menu Project sa vyberie Add Reference. V novo otvorenom dialógovom okne, ktoré je vidieť na obr. 2.1 sa v záložke Browse vyhľadá príslušná knižnica. Potom už len ostáva pridať menný priestor v projekte pomocou direktívy `using`. Takýmto spôsobom si vieme pridať do projektu všetky knižnice a menné priestory, ktoré v projekte budeme využívať.

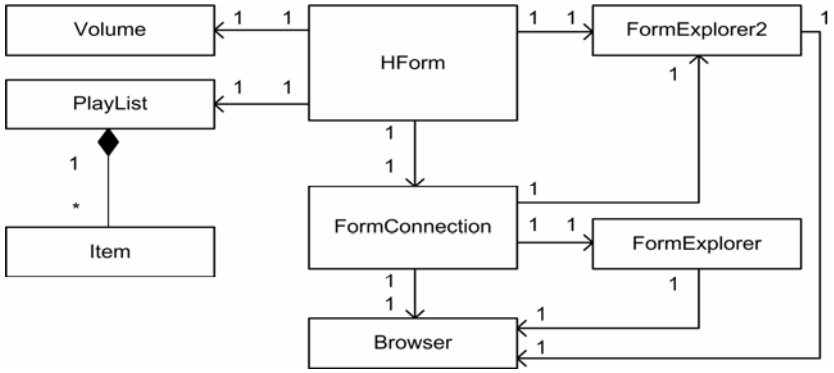


Obr. 2.1. Okno pre pridávanie referencií vo Visual Studiu 2008

2.3 Štruktúra programu z hľadiska tried – UML diagram

V tejto kapitole si popíšeme naprogramovanú aplikáciu z hľadiska tried. Pre lepšiu orientáciu a pochopenie je použitý UML diagram. UML (Unified Modeling Language) je grafický jazyk, ktorý slúži pre vizualizáciu, špecifikáciu a dokumentáciu programových systémov. Základným stavebným kameňom v UML diagrame tried je práve trieda. Trieda je popisom množina objektov, ktoré zdieľajú rovnaké atribúty, metódy a vzťahy. V rámci UML diagramu je určená svojim názvom. Popisovaná multimediálna aplikácia sa skladá z ôsmich tried a to z triedy HForm, FormConnection, FormExplorer, FormExplorer2, Volume,

Browser, Playlist a Item. Vývojový diagram je na obr. 2.2. Na tomto obrázku sú naznačené hlavne väzby medzi jednotlivými triedami. V podrobnejšom popise UML diagramov tried, ktoré nájdeme v prílohe A, môžeme pod každým názvom triedy nájsť atribúty triedy a nakoniec metódy triedy. Atribúty predstavujú vlastnosti triedy a metódy predstavujú operácie danej triedy. Pri každom atribúte a metóde je na začiatku naznačená ich viditeľnosť. Private atribúty a metódy sú značené „-“ a public zase „+“.



Obr. 2.2. UML diagram tried celej aplikácie

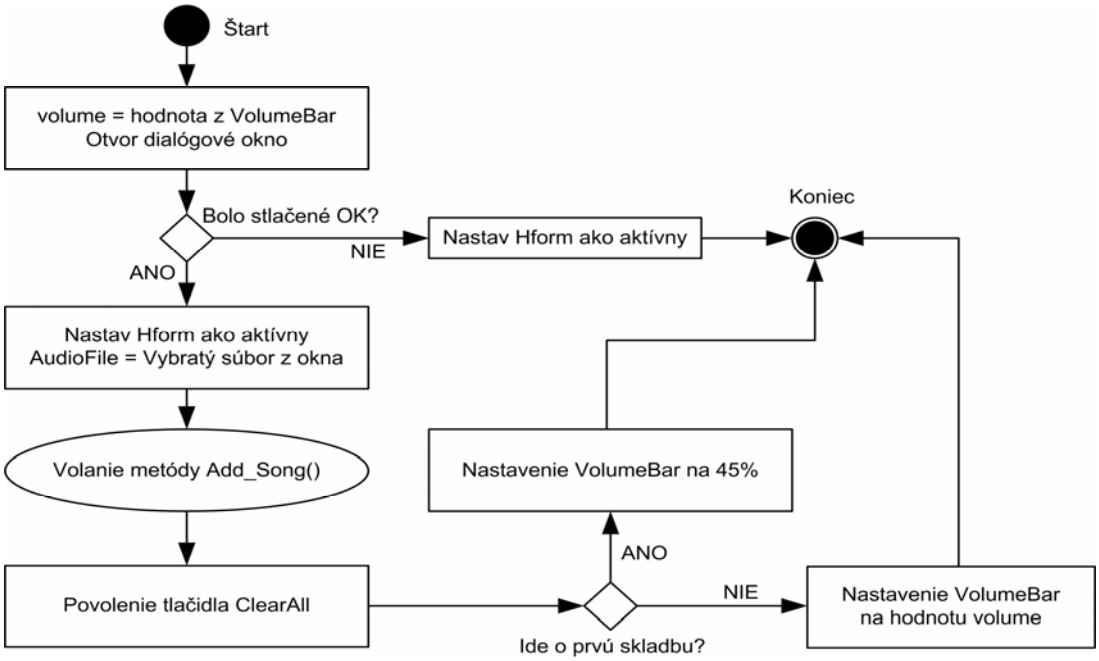
2.3.2 Trieda HForm

Trieda HForm je hlavná trieda programu. Ide o triedu formuláru, ktorá sa stará o zachytávanie všetkých „Event Handlerov“ na danom formulári a vytvárajú sa tu tiež objekty druhých tried ako je Playlist, Volume, FormExplorer2 a FormConnection. Obsahuje atribúty a metódy, ktoré budú vysvetlené v nasledujúcom texte. Podrobnejší UML diagram triedy HForm môžeme nájsť v prílohe A.

Prvou metódou je konštruktor `HForm`, ktorý sa volá hneď po spustení aplikácie. Jeho úlohou je inicializácia komponentov, ktoré sa nachádzajú na formulári a tiež inicializácia všetkých objektov a atribútov. Ďalej sú tu nastavené niektoré hodnoty jednotlivých komponentov a tiež sa tu zakazujú určité tlačidlá, ktoré by nemali byť užívateľovi dostupné pri spustení programu.

Druhou metódou, ktorá môže byť po spustení aplikácie volaná je metóda `bOpen_Click`. Ide o metódu, ktorá je volaná po stlačení tlačidla Open. Okrem toho, že úlohou metódy je vybrať a pridať súbor do playlistu, metóda nastavuje aj hlasitosť prehrávaného súboru. Metóda pracuje nasledovne. Po stlačení tlačidla sa otvorí dialógové okno, kde si užívateľ môže nájsť a vybrať skladbu, ktorú bude chcieť prehrávať. V okne je nastavený filter, ktorý zabezpečuje, aby sa zobrazoval len podporovaný formát mp3 audia.

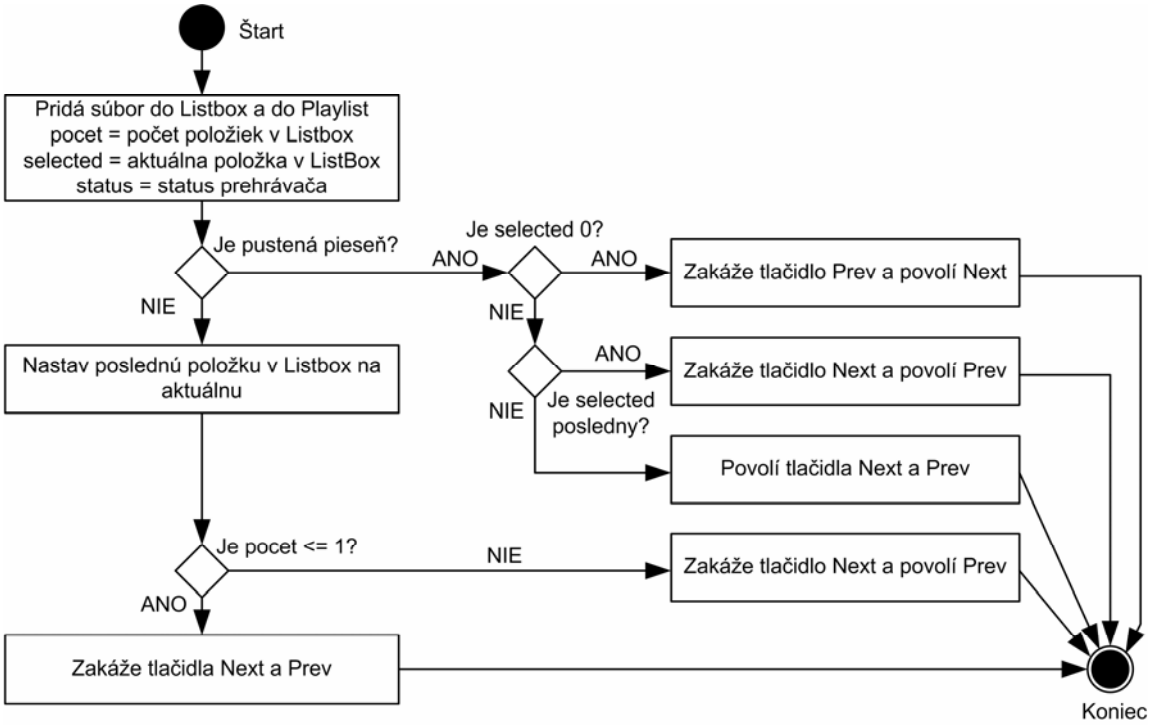
Dialógové okno je objekt triedy FormExplorer2. K zobrazeniu dialógu dôjde volaním metódy `ShowDialog`, ktorá vracia položku typu `System.Windows.Forms.DialogResult` predstavujúcu tlačidlo v dialógovom okne, ktorým užívateľ toto okno uzavrel. Pokiaľ bol stlačený Cancel, metóda `bOpen_Click` nastaví ako aktívny formulár HForm a ukončí sa. Ak bolo však stlačené tlačidlo OK, znamená to, že si užívateľ vybral súbor, ktorý má byť otvorený. Zároveň sa do atribútu *volume* uloží aktuálna hodnota z komponenty VolumeBar. Tá bude slúžiť na to, aby sme vždy mali uchovanú aktuálnu hodnotu hlasitosti. Ak teda bola nejaká skladba vybratá, uloží sa do atribútu *AudioFile*, zavolá sa metóda `Add_Song` (bude popísaná ďalej) a povolí sa tlačidlo ClearAll. V poslednom kroku sa už len testuje, či ide o prvú pridávanú skladbu do playlistu. Ak ide, nastaví sa hodnota hlasitosti na 45%, v opačnom prípade sa nastaví podľa toho, ako je práve nastavená na komponente VolumeBar. Metóda sa potom ukončí. Vývojový diagram je vidieť na obr. 2.3.



Obr. 2.3. Vývojový diagram metódy bOpen_Click

Z metódy `bOpen_Click` je volaná už spomínaná metóda `Add_Song`. Ako už z názvu vyplýva, jej úlohou je pridať vybratú skladbu do playlistu a to vždy na jeho koniec. Teda hneď po štarte metódy sa pridá názov skladby do playlistu (komponenta listBox) a tiež sa skladba pridá do obojsmerného lineárneho zoznamu (popísané v triede Playlist). V ďalšej časti sa podmienkou testuje, či sa práve prehráva alebo neprehráva pieseň. Ak sa neprehráva, nastaví sa ako aktuálny prvok v playliste nový pridaný prvok na konci zoznamu. Ak sa však určitá pieseň prehráva, ponechá sa ako aktuálna práve prehrávaná pieseň. V posledných

fázach metódy sa testujú ďalšie podmienky, ktoré súvisia so zakazovaním a povoľovaním tlačidiel Prev a Next. Záleží len na tom, kde sa v playliste práve nachádzame. Pre lepšie porozumenie metódy je nakreslený vývojový diagram na obr. 2.4.



Obr. 2.4. Vývojový diagram metódy Add_Song

Metódy `bPlay_Click`, `Prev` a `Next` sú z hľadiska funkčnosti veľmi podobné. Ich úlohou je spustiť skladbu. Metóda `bPlay_Click` je volaná po stlačení tlačidla Play. Najprv sa vždy vytvorí vlákno, takzvaný „Worker thread“, ktoré sa bude používať na zobrazenie času piesne, zobrazenie aktuálneho času v piesni a tiež na posúvanie ukazovateľa pozície v skladbe, pomocou ktorého sa vie užívateľ v práve prehrávanej skladbe pohybovať. V ďalšom kroku nastavíme hodnotu atribútu *Terminate* na `false`. Atribút slúži na zrušenie (riadenie) práve spusteného vlákna (Worker thread). Nakoniec sa testuje či bola nejaká skladba vybratá na prehrávanie. Ak nie, objaví sa užívateľovi oznam o tom, aby najprv vybral skladbu, ktorú chce prehrávať. Ak však skladba už bola vybratá, spustí sa samotné prehrávanie skladby. Metódy `Prev` a `Next` sa volajú po stlačení tlačidiel Prev a Next. Odlišujú sa hlavne tým, že najprv sa snažia ukončiť vytvorené vlákno, ktoré vzniklo pri volaní metódy `bPlay_Click`. To je realizované tak, že atribút *Terminate* sa nastaví na hodnotu `true` a zavolá sa metóda `Join`, ktorá počká pokiaľ sa vlákno neukončí. V ďalšom pokračovaní sa zisťuje, či sa skladba práve prehráva alebo nie. Podľa toho sa po stlačení

príslušného tlačidla budeme v playliste len pohybovať bez prehrávania, alebo s tým, že sa budú jednotlivé skladby aj prehrávať.

Metóda `bStop_Click` sa volá po stlačení tlačidla Stop. Jej úloha je veľmi jednoduchá. Zastaví práve prehrávanú skladbu a tiež nastaví atribút *Terminate* na hodnotu `true`. Podobne ako to bolo v predchádzajúcich metódach `Next` a `Prev` je tu metóda `Join`, ktorá čaká na ukončenie vlákna vzniknutého pri začatí prehrávania. Ak ukončenie vlákna prebehlo v poriadku, povolí sa tlačidlo Play a pomocou metódy `ClearTimes` sa nastavia ukazovatele časov a posúvanie v skladbe (komponent progressBar) na pôvodnú hodnotu.

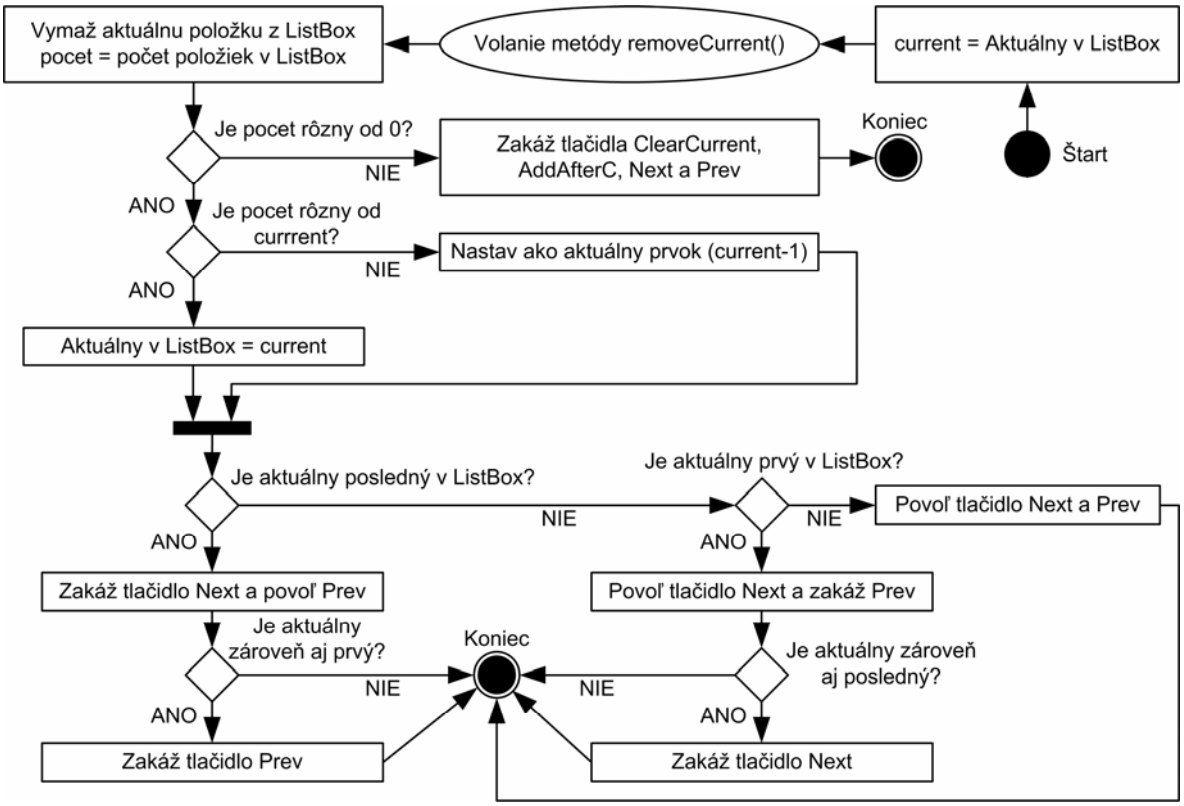
Metóda `VolumeBar_ValueChanged` je volaná vždy pri zmene hlasitosti v programe. Nastaví sa tu najprv minimálna a maximálna hodnota rozsahu, ktorá je v percentách od 0 do 100%. Nová hodnota sa zapíše do ukazovateľa (komponenta label) na ľavo od VolumeBar, aby mal užívateľ prehľad o hlasitosti práve prehrávanej skladby. Ďalej je volaná metóda `VolumeSet` z triedy Volume, ktorá sa stará o nastavenie zvuku na zariadení. Jej vstupným parametrom je práve hodnota z komponenty VolumeBar. Popísaná metóda teda nastaví novú úroveň hlasitosti a ukončí sa.

Metóda `listBox1_SelectedIndexChanged_1` je volaná pri zmene aktuálneho indexu v playliste (komponent listBox). Metóda vždy pri zmene aktuálneho indexu užívateľom v playliste volá metódu `listActNumber` z triedy Playlist (vysvetlená v kapitole 2.3.4), ktorá na základe zmeneného indexu uloží novo vybratú skladbu do atribútu *AudioFile.* Toto sa deje vždy pri zmene aktuálneho indexu v playliste. V ďalšej časti metódy sa už len podľa toho, kde sa nachádzame v playliste zakazujú alebo povoľujú jednotlivé tlačidlá.

V poradí ďalšou metódou je `bClear_Click`. Ide o metódu volanú po stlačení tlačidla ClearA. Jej úlohou je po stlačení tlačidla vymazať všetky prvky z playlistu. To sa deje tak, že sa zavolá metóda `removeAll` z triedy Playlist (vysvetlená v kapitole 2.3.4), ktorá vymaže všetky prvky v obojsmernom lineárnom zozname. Ďalej sa vymažú všetky názvy z komponenty listBox1, ktorá tvorí v prehrávači playlist. V poslednej časti sa v metóde nastaví do atribútu *AudioFile* hodnota `null` a opäť sa zakážu príslušné tlačidlá.

Metóda `bClearC_Click` je volaná po stlačení tlačidla ClearCurrent a jej úlohou je vymazať aktuálny prvok z playlistu. V prvom kroku je volaná metóda `removeCurrent` z triedy Playlist (vysvetlená v kapitole 2.3.4). Tá odstráni aktuálny prvok z obojsmerného lineárneho zoznamu. Tiež sa odstráni názov aktuálnej skladby z playlistu (komponenta listBox). Hneď potom ako sa skladba vymaže, nasleduje niekoľko podmienok, ktorými sa

zisťuje o ktorý prvok z playlistu išlo, respektíve kde sa v playliste nachádzal. Podľa toho sa nastaví aktuálny prvok a povolia sa alebo zakážu jednotlivé tlačidlá. Napríklad, ak išlo o skladbu niekde zo stredu playlistu, nastaví sa ako aktuálna nasledujúca v poradí. Ak však išlo o skladbu na konci playlistu, nastaví sa ako aktuálna predchádzajúca v poradí. Pre lepšie pochopenie metódy je nakreslený jej vývojový diagram na obr. 2.5.



Obr. 2.5. Vývojový diagram metódy bClearC_Click

Ďalšie dve metódy, ktoré sa nachádzajú v triede HForm sú `bAddC_Click` a `Add_Song_After_Current`. Ide o veľmi podobné metódy ako boli `bOpen_Click` a `Add_Song`. Metóda `bAddC_Click` je volaná po stlačení tlačidla AddAfterC. Po stlačení tlačidla sa otvorí opäť dialógové okno, kde si užívateľ môže pridať skladbu do playlistu. Po vybratí skladby je volaná metóda `Add_Song_After_Current`, ktorá vždy pridá novú skladbu za práve aktuálnu v playliste. Je to výhodné v tom, ak si chce užívateľ pridať skladbu za ktorúkoľvek v playliste.

Metóda `Time` sa volá vždy po stlačení tlačidiel Play, Next alebo Prev. Ide o metódu, ktorá slúži ako štartovací delegát pre nové vlákno. Pri volaní štartovacieho delegáta sa vlastne volá konštruktor z triedy Thread. Prvú vec, čo je však potrebné urobiť, je vytvoriť v hlavnom

vlákne vlákno druhé alebo takzvaný „Worker thread“. Ako takéto vytvorenie nového vlákna môže vyzerať, je ukázané v nasledujúcom zdrojovom kóde.

```
//vytvorenie nového vlákna „t“ a štart. delegáta „Time“
Thread t = new Thread(new ThreadStart(Time));
//spustenie nového vlákna
t.Start();
```

Ďalšou dôležitou vecou, ktorú je potrebné vyriešiť v novovzniknutom vlákne je aktualizácia užívateľského rozhrania. Keďže sa jedná o iné vlákno ako je to hlavné, dá sa urobiť aktualizácia pomocou metódy `Control.Invoke`. Metóda pracuje tak, že vytvorí v novom vlákne delegáta, ktorý bude aktualizovať užívateľské rozhranie v hlavnom vlákne. Tento delegát musí byť typu EventHandler. Nevýhodou je, že v .NET Compact Framework sa oproti klasickému .NET Frameworku môže aktualizovať len jeden parameter v `Control.Invoke`. Preto, ak budeme aktualizovať viacej parametrov, musí sa volať viacej metód `Control.Invoke`. Zápis kódu v programe môže vypadať následovne:

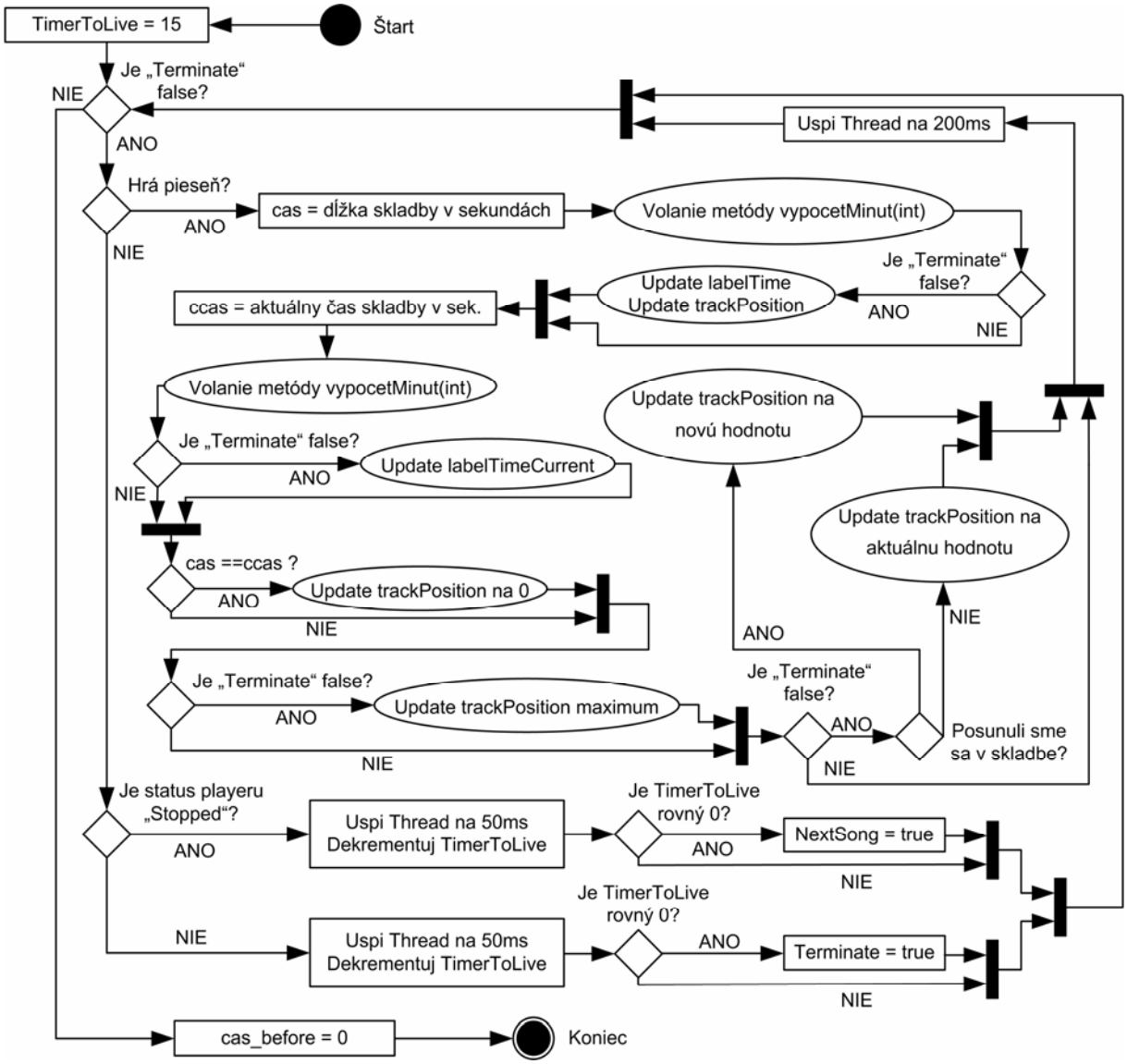
```
// Vytvorenie delegáta WorkerUpdate7, pomocou ktorého updatujeme
trackBar
this.trackPosition.Invoke(new EventHandler(WorkerUpdate7));

//Metóda WorkerUpdate7, ktorá nastaví trackBar na hodnotu 0 a
updatuje užívateľské prostredie
public void WorkerUpdate7(object sender, EventArgs e)
{
     this.trackPosition.Value = 0;
     this.trackPosition.Update();
}
```

Poslednú vec, ktorú je potrebné urobiť v zdrojovom kóde hlavného vlákna, je zavolať metódu `DoEvents`. Metódu je dobré zavolať vtedy, keď sa aktualizuje užívateľské rozhranie pomocou vedľajšieho novovytvoreného vlákna. Je to hlavne z toho dôvodu, aby sa docielilo vykonanie všetkých udalostí, ktoré sa nahromadili pri vykonávaní programu [7].

Funkcia metódy `Time` (vedľajšieho vlákna) je aktualizovať v určitých časový intervaloch komponenty labelTime, labelTimeCurrent a trackPosition. LabelTime slúži na výpis celkovej dĺžky skladby a labelTimeCurrent slúži na výpis aktuálneho času skladby. Komponent trackPosition ukazuje aktuálnu pozíciu skladby a slúži tiež na posúvanie sa v skladbe. Vždy po vykonaní aktualizácie sa vlákno uspí na 0,2 sekundy. Po tejto dobe sa opäť aktualizujú spomínané komponenty. Ak by bolo vlákno vytvorené a status prehrávača by nebol „Playing“, vlákno sa automaticky za určitý časový interval samé zničí (terminuje). O to sa stará vytvorený časovač. Časovač je nastavený pomocou atribútu *TimerToLive*, ktorého hodnota pri spustení vlákna je 15. Táto hodnota udáva počet cyklov, ktoré sa musia vykonať

pokiaľ sa vlákno terminuje. Medzi každým cyklom je pauza 50ms. Časovač má ale aj iné využitie. Predstavme si situáciu, že máme v playliste viacej skladieb, ktoré chceme počúvať a chceme si tieto skladby vypočuť všetky bez toho, aby sme museli vždy stlačiť tlačidlo Next pri prechode z jednej skladby na ďalšiu. Vždy, ak počúvaná skladba dohrá, zmení sa status prehrávača z „Playing“ na „Stopped“. Tento stav automaticky metóda `Time` rozpozná a atribút *TimerToLive* sa každých 50ms zmenší o hodnotu jedna až bude nulový, teda časovač vyprší. Vtedy sa zmení hodnota atribútu *NextSong* na `true`. Hodnota atribútu *NextSong* je snímaná časovačom timer1 každú sekundu. Ak timer1 zistí, že hodnota *NextSong* je `true`, je volaná metóda `Next`, ktorá spustí ďalšiu skladbu v playliste. Takýmto spôsobom sa prehrajú všetky skladby v playliste. Vývojový diagram metódy `Time` je vidieť na obr. 2.6.



Obr. 2.6. Vývojový diagram metódy Time

Ďalšou metódou v triede HFrom je metóda `vypocetMinut`. Je volaná z metódy `Time` a jej úlohou je prepočítať sekundy na minúty a sekundy. Vstupným parametrom je počet sekúnd. Metóda vypočíta z tohto vstupného čísla, o koľko ide minút a sekúnd. Tieto dve čísla si zapíše do matice. Matica slúži ako návratová hodnota. Teda metóda po výpočte vždy vráti aktuálnu maticu a ukončí sa.

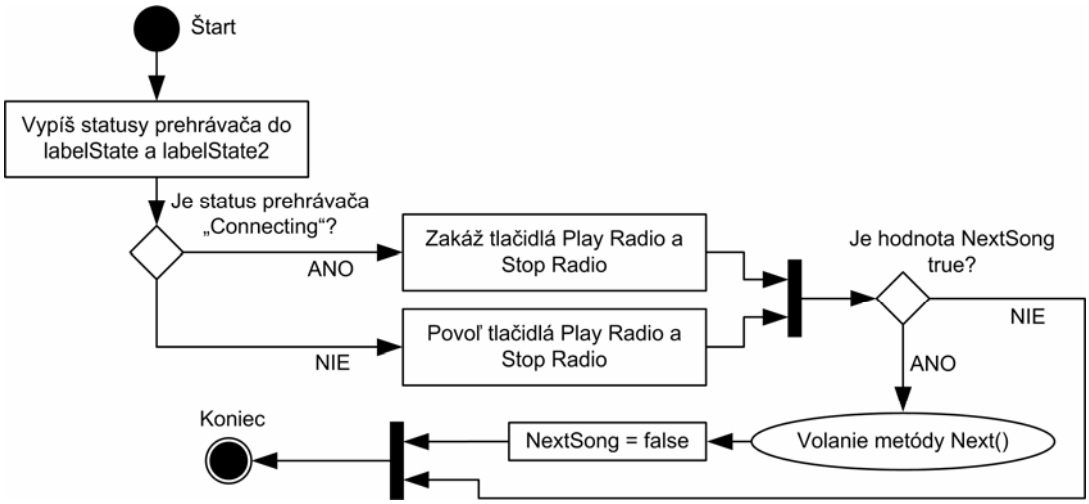
Metódy `buttonSwitch_Click` a `buttonStream_Click` majú veľmi podobné funkcie. Metóda `buttonSwitch_Click` je volaná po stlačení tlačidla File Transfer. V prvom kroku sa testuje, či je status prehrávača „Playing“. Ak áno, zobrazí sa pre užívateľa výzva k tomu, aby najprv zastavil prehrávanú skladbu. Ak však prehrávač momentálne nič neprehráva, vytvorí sa nový objekt triedy FormConnection a zobrazí sa formulár (popis triedy FormConnection nájdeme v kapitole 2.3.5). Ak sa formulár FormConnection zatvorí, metóda nastaví ako aktívny formulár HForm a ukončí sa. Metóda `buttonStream_Click` je volaná po stlačení tlačidla Stream. Jej funkcia je veľmi podobná `buttonSwitch_Click`. Rozdiel je len v tom, že sa nevytvára nový objekt triedy, ale zviditeľní sa komponent panelRadio na ktorom sú všetky potrebné komponenty k prehrávaniu streamovaných rádií. Metóda tiež zakazuje a povoľuje príslušné tlačidlá.

Presne opačnú funkciu má metóda `buttonCloseRadio_Click`. Tá práve naopak komponentu panelRadio schová a súčasne s ňou aj všetky ostatné komponenty, ktoré sú na panelRadio umiestnené. Tým sa opäť zviditeľní prehrávač audio súborov. Metóda má tiež za úlohu zastaviť prehrávanie streamu, ak sa momentálne nejaký prehráva.

Metóda `buttonPlayRadio_Click` je volaná po stlačení tlačidla Play Radio. Jej úlohou je prehrávať streamované rádia. Rádia si môže užívateľ vybrať z už preddefinovanej ponuky alebo adresu zadať ručne. Záleží len na tom, čo si vyberie pomocou prepínača, ktorý sa skladá z dvoch radioButtonov. Na zastavenie prehrávania streamovaného rádia slúži metóda `buttonStopRadio_Click`.

Ďalšou, veľmi dôležitou metódou v triede HForm je `timer1_Tick`. Ide vlastne o časovač, ktorý je volaný v pravidelných časových intervaloch. V našej aplikácií to je každú sekundu. Metóda má za úlohu sledovať stavy (statusy) prehrávača a vypisovať ich do príslušných miest na displeji (obr. 2.23). V nasledujúcej časti kódu sa podmienkou testuje, či je status prehrávača „Connecting“. Tento status môže mať prehrávač, ak sa pripája na nejakú adresu na ktorej sa streamuje rádio. Ak je teda podmienka splnená, zakážu sa tlačidlá Play Radio a Stop Radio. Je to z toho dôvodu, aby užívateľ nemohol znovu stlačiť tieto tlačidlá

práve vtedy, keď sa pripája prehrávač na určitú adresu. V poslednom časti kódu sa testuje hodnota atribútu *NextSong*. Atribút slúži na rozpoznanie stavu, kedy sa má a kedy nemá volať metóda `Next`, ktorá slúži na prehrávanie ďalšej skladby. Vývojový diagram tejto metódy je vidieť na obr. 2.7.



Obr. 2.7. Vývojový diagram metódy timer1_Click

Poslednou metódou je `HForm_Closing`. Metóda je volaná pri zatvorení aplikácie. Jej úlohou je zastaviť prehrávanie, ak sa práve prehráva nejaká skladba a ukončiť pustené vlákno, ak práve nejaké beží. Je to veľmi dôležité, pretože ak by sa vlákno neukončilo, aplikácia by sa zavrela, no vlákno by stále ostalo bežať niekde v pozadí, čo by malo za následok prebytočné vyťaženie procesoru a tým plytvanie batérií. Keďže sa jedná o aplikáciu, ktorá bude bežať na mobilných zariadeniach, je výdrž batérií veľmi dôležitým kritériom pri vývoji aplikácie.

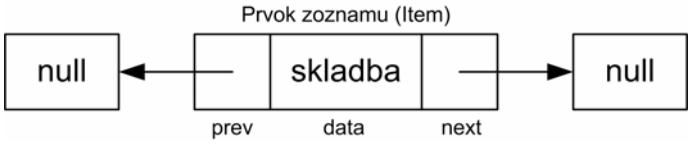
2.3.3 Trieda Volume

Trieda Volume obsahuje dve metódy. Ako jej samotný názov napovedá, jej hlavná úloha je ovládanie zvuku prehrávača. Prvá metóda je `VolumeSet`. V nej sa nastavuje úroveň hlasitosti a to tým spôsobom, že je volaná metóda `waveOutSetVolume`, ktorá je do triedy importovaná pomocou atribútu `DllImport` z knižnice „coredll.dll“. Druhá metóda `PercentaNaVolume` má viditeľnosť private, to znamená, že je dostupná len v triede Volume. Metóda je volaná z `VolumeSet` a jej úlohou je konvertovať vstupné číslo, ktoré je v percentách od 0 do 100 na požadovanú hodnotu, ktorú pozná zariadenie. Tým sa dá dosiahnuť využitia celej úrovne zvukového rozsahu zariadenia.

2.3.4 Trieda Playlist a Item

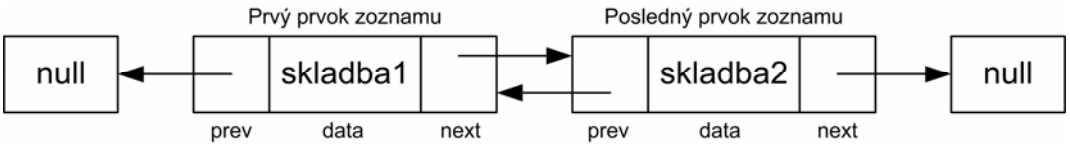
Každý správny prehrávač by mal mať playlist, kde si užívateľ môže pridávať skladby, ktoré by chcel prehrávať. V programe je playlist realizovaný ako obojsmerný lineárny zoznam. Ide o abstraktný dátový typ, ktorý umožňuje, aby počet jeho prvkov rástol alebo klesal do ľubovoľnej dĺžky. Aj keď je lineárny zoznam náročnejší na implementáciu jeho použitie je oveľa výhodnejšie oproti statickému poľu, pretože v poli je počet prvkov pevne daný. Zoznamov existuje viacej druhov, no v aplikácií je použitý práve obojsmerný lineárny zoznam. Je to z toho dôvodu, pretože každý prvok v zozname obsahuje dva ukazovatele. Jeden ukazovateľ na predchádzajúci prvok a druhý ukazovateľ na prvok nasledujúci v zozname. V aplikácii sa tieto ukazovatele naplno využívajú pri prepínaní v playliste medzi jednotlivými skladbami [4], [5].

Obojsmerný lineárny zoznam je realizovaný v programe triedami Playlist a Item. Trieda Item špecifikuje prvok zoznamu. Obsahuje tri atribúty a to *data* (obsahuje priamo skladbu), *prev* (ukazovateľ na predchádzajúci prvok v zozname) a *next* (ukazovateľ na nasledujúci prvok v zozname). Pre lepšiu ilustráciu je naznačený prvok zoznamu na obr. 2.8. Trieda Item ďalej obsahuje šesť metód. Jedna z nich je konštruktor, ktorý vždy nastaví počiatočné hodnoty pri vytvorení objektu tejto triedy, teda nového prvku zoznamu. Ďalšou metódou je `getData`, ktorá vráti dátovú zložku daného prvku zoznamu, to znamená vybratú skladbu. Posledné štyri metódy sú `getPrev`, `getNext`, `setPrev` a `setNext`. Metódy get majú návratovú hodnotu, v ktorej vracajú buď predchádzajúci alebo nasledujúci prvok zoznamu. Metódy set nastavujú ukazovatele daného prvku.



Obr. 2.8. Ukážka prvku zoznamu

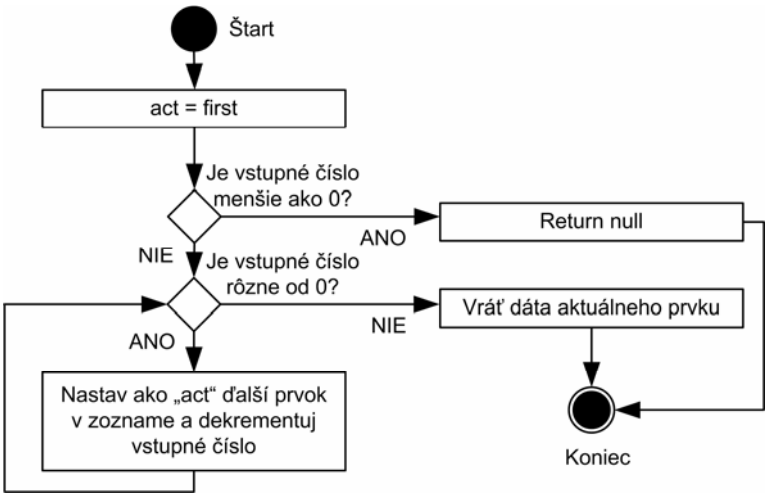
Trieda Playlist špecifikuje operácie, ktoré môžeme v zozname využívať. Obsahuje podobne ako trieda Item tri atribúty a to *first* (ukazovateľ na prvú položku v zozname), *last* (ukazovateľ na poslednú položku v zozname) a *act* (ukazovateľ na aktuálnu položku v zozname). Trieda obsahuje deväť metód. Pomocou nich môžeme prvky do zoznamu pridávať alebo odoberať. Vždy však musia byť správne nastavené ukazovatele jednotlivých prvkov, inak by sa medzi prvkami stratila väzba a zoznam by sa rozpadol. Ak sú však ukazovatele nastavené správne, obojsmerný lineárny zoznam vyzerá ako na obr. 2.9.



Obr. 2.9. Ukážka obojsmerného lineárneho zoznamu

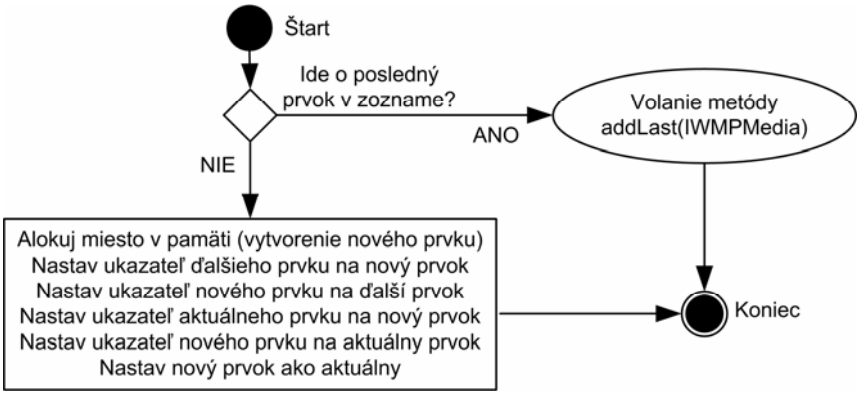
Prvou metódou v triede Playlist je konštruktor, ktorý po vytvorení objektu triedy Playlist nastavuje počiatočné hodnoty. Druhou metódou je `isEmpty`, ktorá vracia hodnotu pravda (`true`), ak lineárny zoznam neobsahuje žiadny prvok alebo vráti hodnotu nepravda (`false`), ak sa nachádza v zozname aspoň jeden prvok. Ďalšou metódou je `addLast`. Jej úlohou je pridať prvok na koniec zoznamu. Metóda si najprv overí pomocou `isEmpty`, či ide alebo nejde o prázdny zoznam. Ak je zoznam prázdny, vytvorí prvý prvok zoznamu. Ak prázdny nie je, pridá nový prvok na koniec zoznamu a nastaví príslušné ukazovatele. Ďalšie dve metódy `removeFirst` a `removeLast` sú svojou funkciou veľmi podobné. Obidve vymažú prvok zo zoznamu, no každá z iného konca. Metóda `removeFirst` vymaže prvok zo začiatku zoznamu a to tak, že najprv nastaví ako *first* ďalší prvok v poradí a zruší väzbu na predchádzajúci prvok. Tým sa tento prvok zo zoznamu vymaže. Metóda `removeLast` podobne ako `removeFirst` vymaže prvok, no z konca zoznamu.

Ďalšou metódou je `listActNumber`. Ide o špeciálnu metódu v zozname, ktorá na základe čísla skladby v playliste, vráti požadovanú skladbu na prehrávanie (obr. 2.10). Metóda v prvom kroku priradí do atribútu *act*, hodnotu prvej skladby v playliste. Potom v prvej podmienke testuje, či nebolo zadané záporné číslo. Ide vlastne o ošetrenie, pretože záporné číslo skladby by sa v playliste nikdy nenašlo. Ak je teda zadané vstupné číslo skladby záporné, vráti sa hodnota `null` a metóda sa ukončí. Ak je vstupné číslo nula, metóda vráti prvý prvok zoznamu. Ak je jedna, vráti druhý prvok zoznamu. Takto vždy vráti ten prvok zoznamu, ktorého číslo je zadané. Metóda sa v aplikácii využíva tak, že na základe označenej skladby užívateľom v playliste sa vráti priamo skladba, ktorá sa má prehrávať.



Obr. 2.10. Vývojový diagram metódy listActNumber

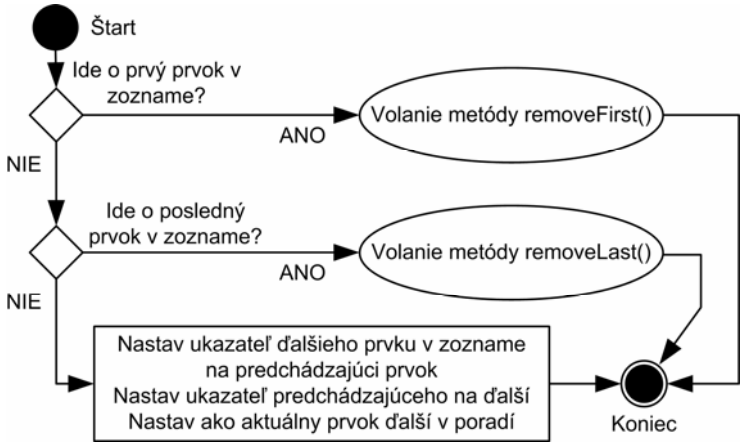
Po metóde `listActNumber` nasleduje metóda `addAfterCurrent`, ktorej úlohou je pridať nový prvok do zoznamu práve za aktuálny prvok. Metóda má jednu vstupnú hodnotu a to skladbu, ktorú chceme pridať. Funkcia je nasledovná. V prvom kroku si podmienkou otestuje, či ide alebo nejde o posledný prvok zoznamu. Ak ide, všetko čo metóda urobí je, že zavolá metódu `addLast`, ktorá pridá prvok na koniec zoznamu. Metóda `addAfterCurrent` sa potom ukončí. Ak však nejde o posledný prvok zoznamu, vytvorí sa nový prvok zoznamu (objekt triedy Item), ktorý bude obsahovať vstupné dáta. V ďalších krokoch metóda musí zabezpečiť správne pospájanie ukazovateľov medzi prvkami zoznamu. Nakoniec sa nastaví pridaný prvok ako aktuálny prvok a metóda sa ukončí. Pre lepšie pochopenie je funkcia metódy naznačená na vývojovom diagrame na obr. 2.11.



Obr. 2.11. Vývojový diagram metódy addAfterCurrent

Ďalšou metódou v poradí je `removeCurrent`. Jej úlohou je odstrániť aktuálny prvok zo zoznamu. Metóda je veľmi jednoduchá. Najprv testujeme či ide o prvý prvok zoznamu. Ak áno, volá sa metóda `removeFirst`, ktorá odstráni tento prvok zo zoznamu. Metóda sa potom

ukončí. Ak však nejde o prvý prvok zoznamu, testuje sa či ide o posledný prvok zoznamu. Ak ide, volá sa metóda `removeLast`, ktorá odstráni tento prvok zo zoznamu. Metóda sa potom opäť ukončí. Posledný prípad, ktorý môže nastať je, že nejde ani o prvý ani o posledný prvok zo zoznamu. Ide teda o prvok niekde zo stredu zoznamu. Vtedy sa musia nastaviť správne ukazovatele medzi prvkami a to tak, že najprv sa nastaví ukazovateľ nasledujúceho prvku v zozname na prvok, ktorý je pred aktuálnym a naopak ukazovateľ predchádzajúceho sa nastaví na prvok ďalší v poradí za aktuálnym. V poslednom kroku sa nastaví nasledujúci prvok v poradí v zozname ako aktuálny. Tým je všetko hotové a metóda sa ukončí. Vývojový diagram metódy je naznačený na obr. 2.12.



Obr. 2.12. Vývojový diagram metódy removeCurrent

Poslednou metódou v triede Playlist je metóda `removeAll`. Jej funkciou je vymazať celý zoznam. Aby sa zoznam celý vymazal, stačí nastaviť všetky tri atributy (*act*, *first* a *last*) na hodnotu `null`. Tým sa stratia väzby medzi prvkami a zoznam sa rozpadne.

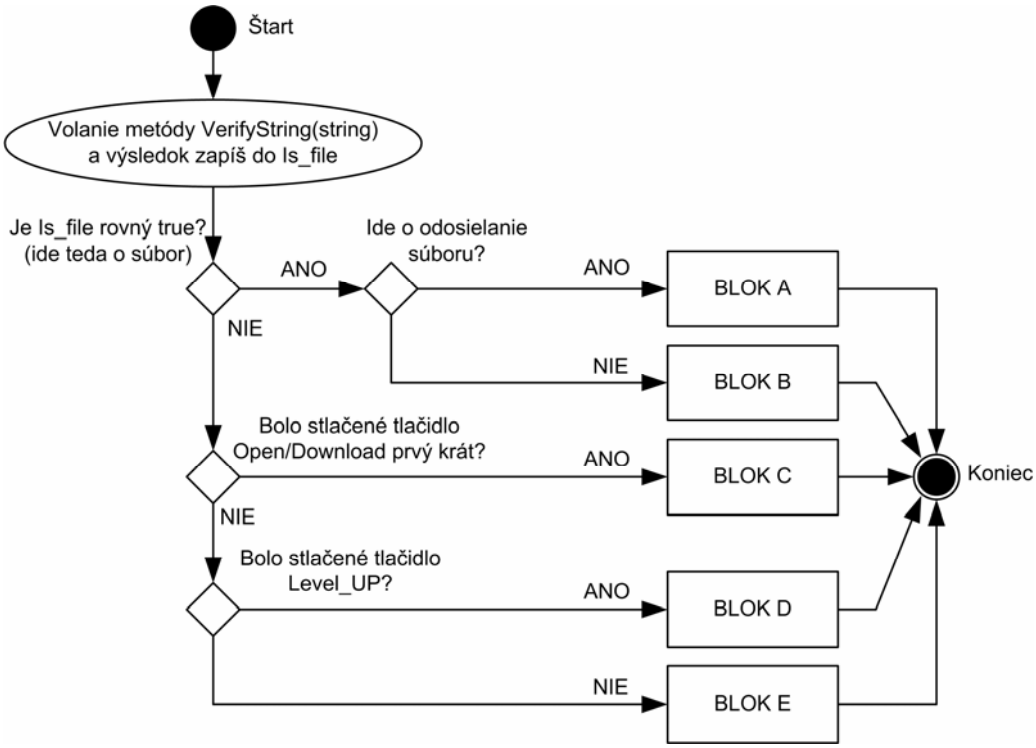
2.3.5 Trieda FormConnection

Trieda FormConnection sa stará ako aj jej názov napovedá o sieťovú komunikáciu. Jej úlohou je vytvorenie užívateľského prostredia pre spojenie mobilnej aplikácie so serverom, na ktorom beží ďalšia serverová aplikácia prostredníctvom ktorej si môže užívateľ sťahovať požadované skladby do svojho mobilného zariadenia. V tejto kapitole si opíšeme triedu FormConnection z hľadiska funkcií jednotlivých metód a atribútov. Podrobnejšie sa budeme venovať sieťovej komunikácií v kapitole 2.4.

Prvou metódou je metóda `FormConnection`, ktorá slúži ako konštruktor tejto triedy. Jej úlohou je inicializácia všetkých komponentov, ktoré sa nachádzajú na formulári a tiež inicializácia všetkých objektov a atribútov na požadované počiatočné hodnoty.

Ďalšou veľmi dôležitou metódou, ktorá je volaná po stlačení tlačidla Connect je metóda `connctionButton_Click`. Metóda sa stará o pripojenie klienta na server. Metóda na začiatku prečíta zadanú IP adresu, ktorú užívateľ zadá do príslušných políčok, vytvorí si socket na ktorom bude komunikácia prebiehať a pokúsi sa pripojiť na server pomocou metódy `connect`. Ak by sa pripojenie nepodarilo, užívateľ je o tomto stave informovaný textom „Unable to connect server“ a vytvorený socket sa uzavrie pomocou metódy `close`. Ak sa však podarí klientovi na server pripojiť, čaká sa na príjem správy „Server is connected“, ktorá sa vypíše na displej, aby bol užívateľ informovaný o tom, že komunikácia so serverom prebehla úspešne. Metóda tiež zakazuje a povoľuje príslušné komponenty podľa toho, či sa klientovi podarilo alebo nepodarilo na server pripojiť. Viac o pripojovaní klienta na server bude písané v kapitole 2.4.

Ak je klient na server úspešne pripojený a užívateľ chce prijímať, poprípade odosielať súbory (dáta), bude s istotou volaná metóda `SendFile`. Ide o metódu, ktorá sa stará o samotný prenos dát sieťou. Jej funkcia je dosť zložitá, preto je aj vývojový diagram, ktorý môžeme vidieť na obr. 2.13 rozdelený na bloky. Tieto bloky predstavujú ďalšie samostatné vývojové diagramy, ktoré si popíšeme v nasledujúcom texte.

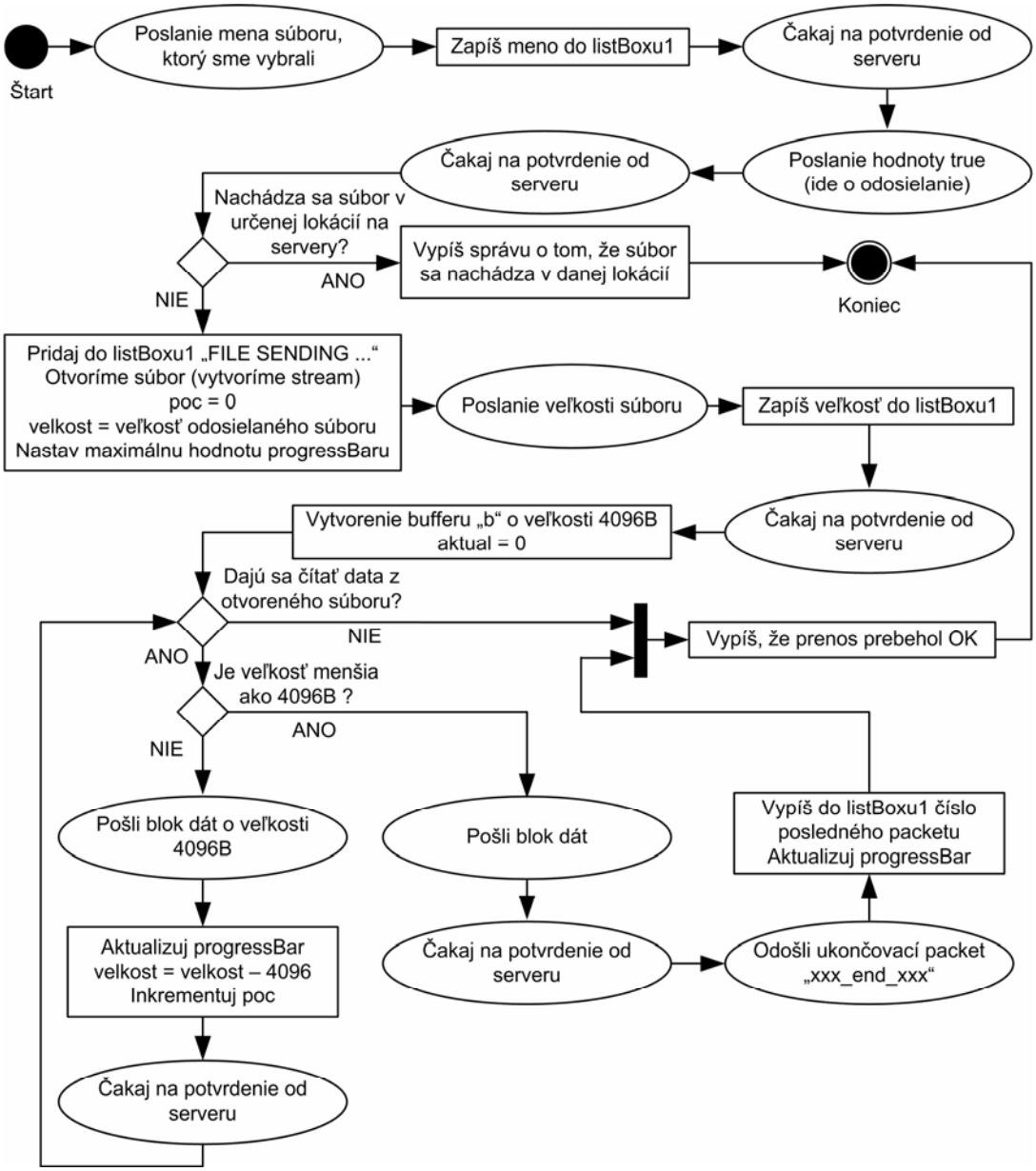


Obr. 2.13. Vývojový diagram metódy SendFile

V prvom kroku je volaná metóda `VerifyString`, ktorá má za úlohu zistiť, či sa jedná alebo nejedná o súbor. Ak metóda zistí, že ide o súbor, vráti hodnotu `true`, v opačnom prípade vráti hodnotu `false`. Hodnota sa zapíše do atribútu *Is_file*. V nasledujúcom kroku metóda `SendFile` testuje, aká hodnota sa zapísala do *Is_file*. Ak ide o súbor, ďalšou podmienkou sa rozlíši, či ide o odosielanie alebo prijímanie súboru. Ak ide o odosielanie súboru pokračuje sa vo vykonávaní programu blokom A. V opačnom prípade sa pokračuje blokom B. Ak sa však o súbor nejedná, testuje sa, či bolo stlačené tlačidlo Open/Download prvý krát. V programe tento stav nastane poslaním textu „xxx_start_xxx“ serveru a vykoná sa kód v bloku C. V opačnom prípade sa testuje stlačenie tlačidla Level_UP, ktoré sa zistí poslaním textu „xxx_level_xxx“. Podľa toho, či sa prijatý text zhoduje s „xxx_level_xxx“ alebo nie, pokračuje sa vykonávaním blokov D alebo E.

Z predchádzajúcej časti textu je zrejmé, že pomocou sledu podmienok sa pri volaní metódy `SendFile` zavolá vždy len jeden blok. Každý z týchto blokov má podobnú funkciu, ale jedinečné použitie. Prvý blok, ktorý si opíšeme je blok A. Ten je volaný vtedy, ak sa jedná o odosielanie súboru od klienta na server. Na začiatku sa odošle názov súboru, ktorý sa bude odosielať. Zároveň sa tento názov zapíše do komponenty listBox. Komponent listBox slúži pre lepšiu orientáciu užívateľa, aby vedel, čo sa v danej chvíli na prenose deje. Po odoslaní názvu súboru sa čaká na potvrdenie od serveru, aby sme sa uistili, že poslaný názov server prijal. Potvrdzovanie spočíva v tom, že sa pošle paket s textom „OK“. Paket neslúži na kontrolu integrity dát. O to sa stará TCP protokol, na ktorom je celá komunikácia založená (kapitola 2.4) . Slúži hlavne k synchronizácii, aby bol klient vždy informovaný o tom, že poslaný paket server prijal. Klient po prijatí potvrdenia posiela serveru hodnotu `true` a opäť čaká na potvrdenie od serveru. Hodnotou `true` dáme serveru vedieť, že ide o odosielanie súboru od klienta. V tejto fázy prenosu však nemusí potvrdenie od serveru prísť vo forme „OK“, ale vo forme špeciálneho paketu, ktorého obsahom bude text „xxx_present_xxx“. To znamená, že server našiel súbor s rovnakým názvom na mieste, kde sa má uložiť práve posielaný súbor. Klient po prijatí takejto správy vypíše na displej upozornenie pre užívateľa, aby vybral iný súbor a ukončí prenos. Ak sa však súbor s rovnakým názvom na servery nenachádza, posiela sa veľkosť prenášaného súboru a čaká sa znovu na potvrdenie. V poslednej fázy prenosu sa vytvorí buffer o veľkosti 4096 B (bajtov) a začnú sa posielať samotné dáta skladby. Dáta sa posielajú práve po blokoch 4096 B a vždy po poslaní bloku sa čaká na potvrdenie serverom. Ak sú všetky dáta poslané na server, posiela sa ukončovací paket s textom „xxx_end_xxx“, ktorý dá serveru vedieť, že všetky dáta sú odoslané a prenos

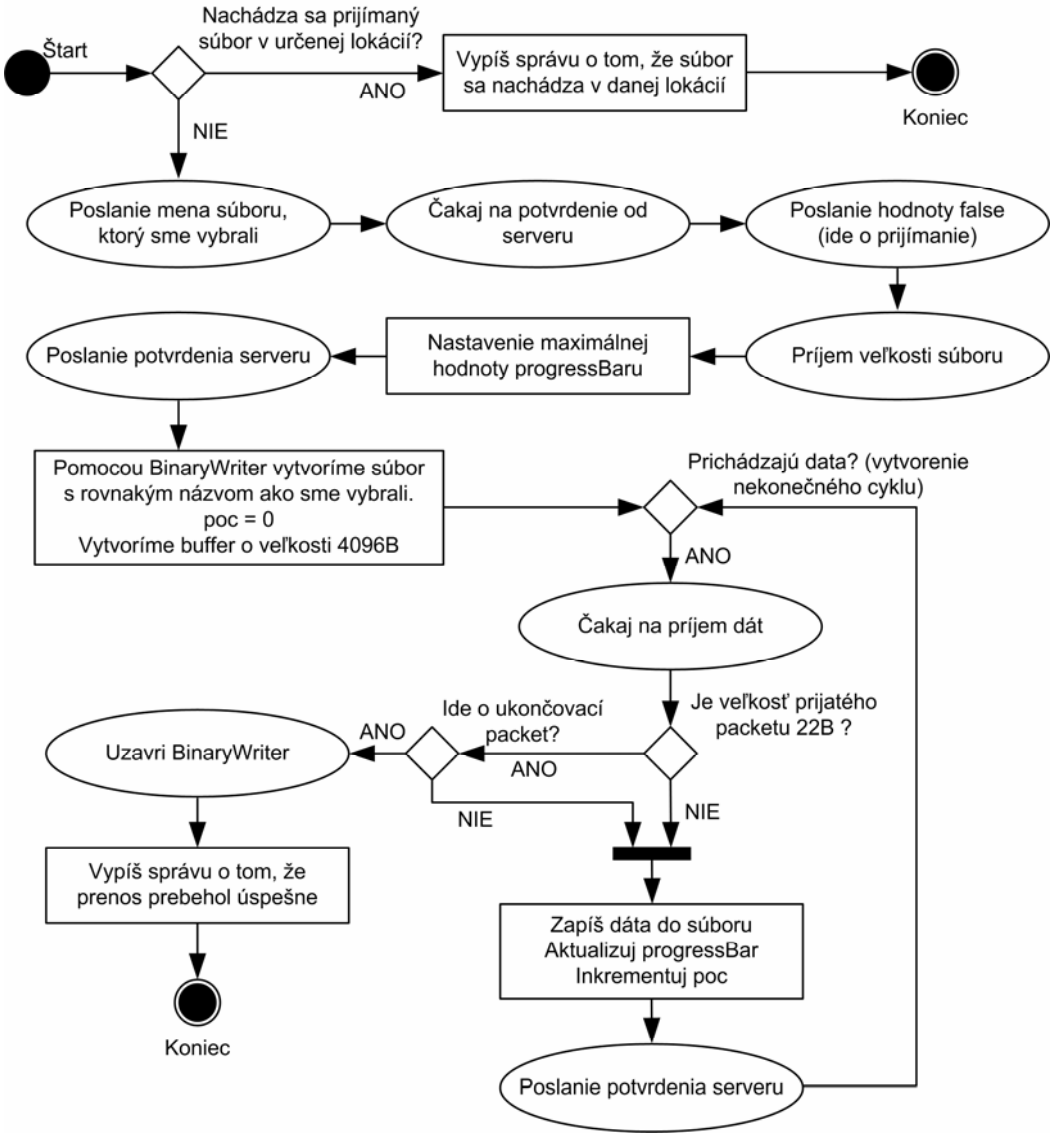
sa môže ukončiť. Užívateľ môže sledovať priebeh prenosu pomocou komponenty progressBar. Po úspešnom prenesení všetkých dát sa do komponenty listBox vypíše číslo posledného preneseného paketu a na displeji sa zobrazí oznam o tom, že prenos prebehol úspešne. Vývojový diagram bloku A metódy `SendFile` je naznačený na obr. 2.14.



Obr. 2.14. Vývojový diagram metódy SendFile – BLOK A

Blok B metódy `SendFile` sa vykoná v prípade, ak sa jedná o prijímanie súboru zo serveru smerom ku klientovi. Postup akým sa zdrojový kód prevádza je podobný bloku A, až na malé rozdiely, ktoré si popíšeme. Skladba, ktorú chce užívateľ stiahnuť zo serveru sa vyberá v zozname, ktorý je presným obrazom práve aktuálneho adresáru na počítači, kde beží serverová aplikácia. Ako sa tento zoznam získa, bude popísané nižšie, pri popisovaní bloku C.

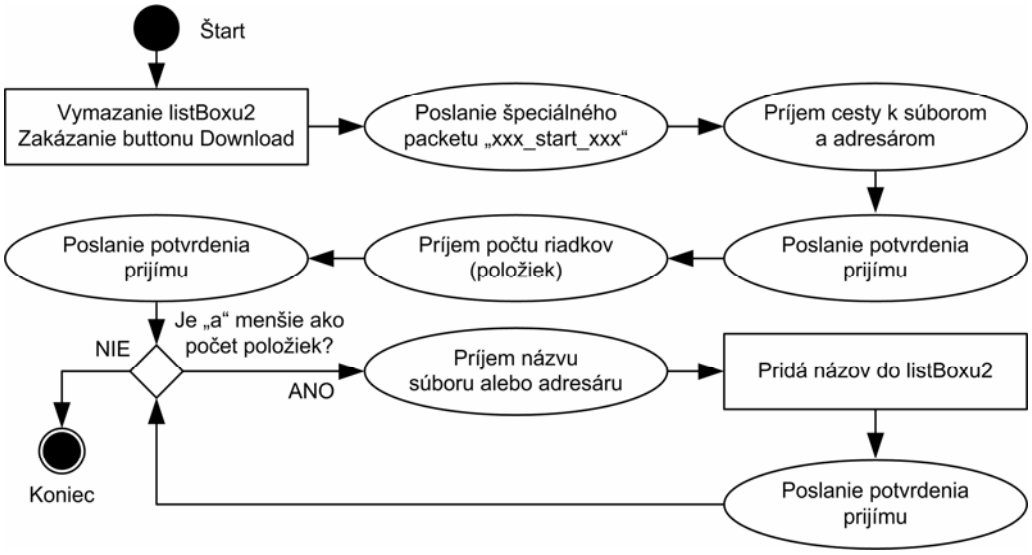
Po vybratí skladby sa v prvom kroku porovnáva, či sa daná skladba už nachádza v adresári na mobilnom zariadení, ktorý sme určili pre príjem súborov. Ak sa v adresári skladba nachádza, vypíše sa správa o upozornení tejto situácie a metóda sa ukončí. V druhom prípade sa pošle meno súboru serveru a čaká sa na potvrdenie. Ostatné kroky sú identické s krokmi, ktoré sú opísané pri bloku A. Ukončenie prijímania súboru nastane v prípade prijatia ukončovacieho paketu s textom „xxx_end_xxx“. V poslednej časti sa uzavrie BinaryWriter, ktorý slúži na zapisovanie prijatých dát do súboru a opäť sa vypíše správa o úspešnom dokončení sťahovania. Vývojový diagram bloku B metódy `SendFile` môžeme vidieť na obr. 2.15.



Obr. 2.15. Vývojový diagram metódy SendFile – BLOK B

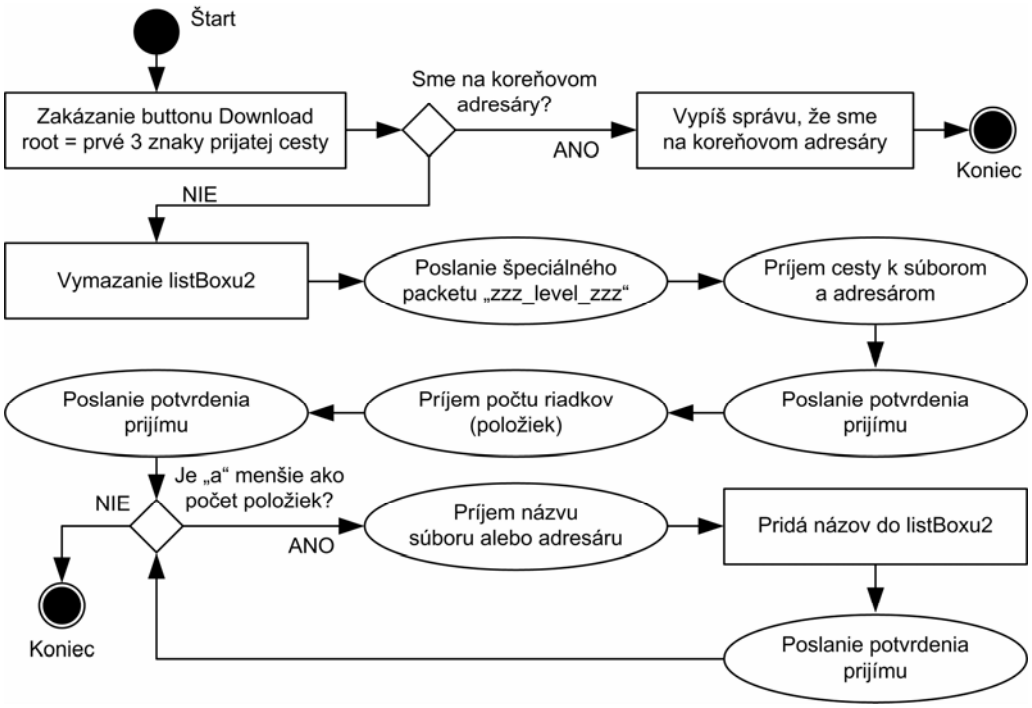
Blok C metódy `SendFile` je volaný vždy len raz a to pri prvom stlačení tlačidla Open/Download. Slúži na získanie zoznamu adresárov a súborov zo serveru. Užívateľ tak získa presnú kópiu svojho disku na serverovej strane a prostredníctvom tlačidiel Level_UP

a Open/Download sa vie v tejto adresárovej štruktúre pohybovať až k požadovanej skladbe, ktorú si chce stiahnuť. Vývojový diagram bloku C metódy `SendFile` je vidieť na obr. 2.16. Činnosť kódu je jednoduchá. Najprv sa vymaže komponent listBox, aby sme nemali prebytočné prvky v zozname a zakáže sa tlačidlo Open/Download. To sa opäť povolí, až keď užívateľ označí jeden z prijatých prvkov v zozname. V ďalšom kroku sa pošle špeciálny paket s textom „xxx_start_xxx“, ktorý dá serveru vedieť, že tlačidlo Open/Download bolo stlačené prvý krát. Od serveru sa potom očakáva príjem cesty k súborom a adresárom, aby aplikácia vedela, kde sa nachádza v adresárovej štruktúre. Po úspešnom doručení cesty sa posiela potvrdenie a čaká sa na príjem počtu riadkov a adresárov. To znamená, ak vybratý adresár na serveri obsahuje napríklad sedem adresárov a tri súbory (počítajú sa len súbory mp3), prijaté číslo bude desať. Po prijatí čísla sa pošle potvrdenie serveru a môže sa začať so samotným prenosom názvov jednotlivých adresárov a súborov. Samozrejme každý prvok zoznamu sa znovu potvrdí. Takýmto spôsobom užívateľ získa presný obsah vybratej zložky na servery. Po prijatí posledného názvu sa metóda `SendFile` ukončí.



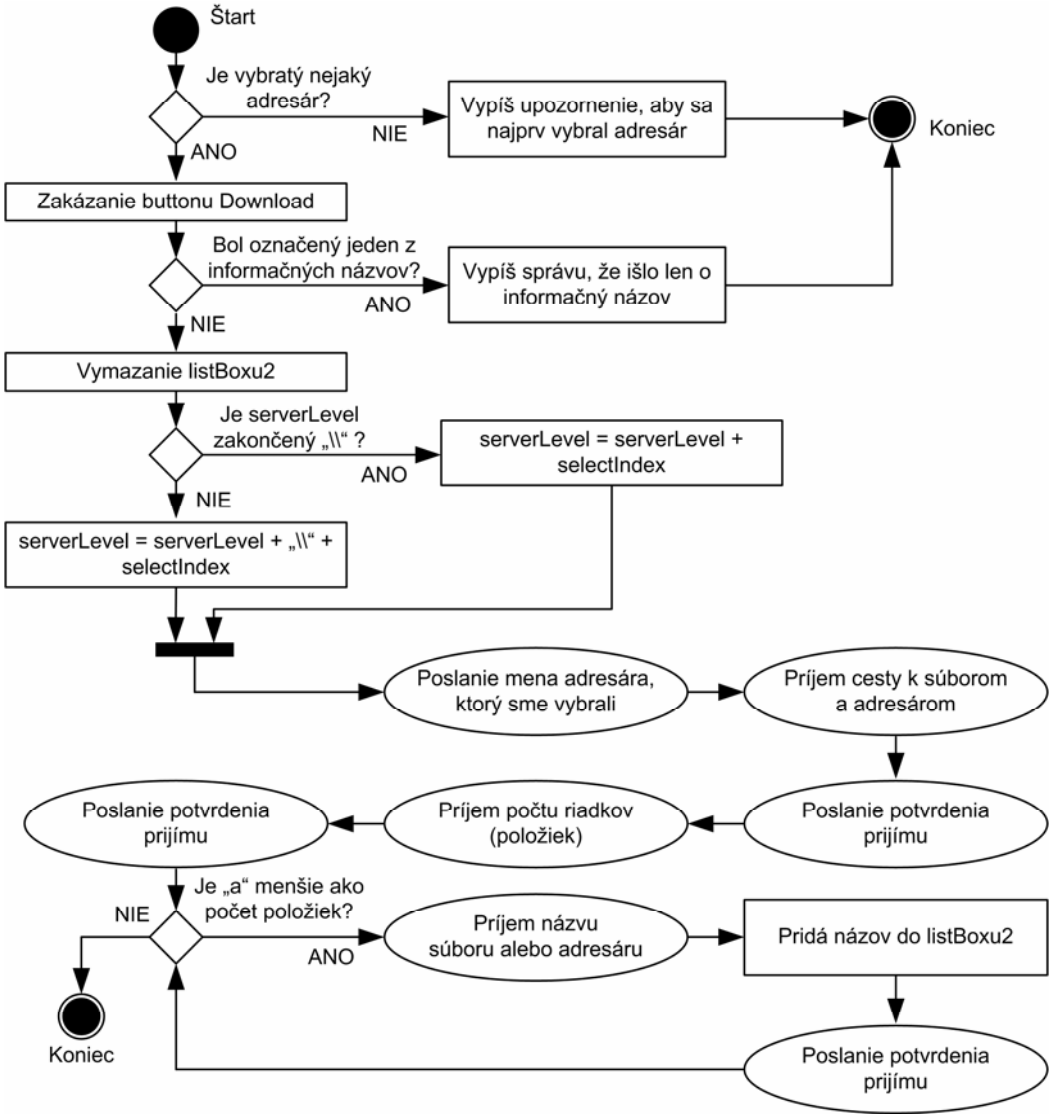
Obr. 2.16. Vývojový diagram metódy SendFile – BLOK C

Blok D metódy SendFile je volaný po stlačení tlačidla Level_UP. Má presne tú istú funkciu ako blok C. Rozdiel je len v tom, že podmienkou testujeme, či sme na koreňovom adresári na disku. Ak áno, vypíše sa o tom užívateľovi správa a metóda sa ukončí. Je to z toho dôvodu, aby sme sa nedostali mimo adresárovú štruktúru disku. Vývojový diagram bloku D je na obr. 2.17.



Obr. 2.17. Vývojový diagram metódy SendFile – BLOK D

Posledným blokom v metóde `Sendfile` je blok E. Zdrojový kód tohto bloku sa vykonáva po ďalšom stlačení tlačidla Open/Download, no za predpokladu, že vybratou položkou zo zoznamu v komponente listBox je adresár. Na začiatku vykonávania bloku sa jednou podmienkou testuje, či bol nejaký adresár vybratý pre prenos. Ak bol, druhou podmienkou sa otestuje, či nejde o informačný text. Informačné texty sú v prijatom zozname vždy dva. Sú to „********** FOLDERS **********“ a „********** FILES **********“. Ich úlohou je len sprehľadniť prijatý zoznam, aby bolo jasné, ktoré názvy sú adresáre a ktoré súbory. Ak obidve podmienky prebehli úspešne, vymažeme sa predchádzajúci zoznam z komponenty listBox a do atribútu *serverLevel* sa uloží cesta spolu s vybratým adresárom. Takto pripravenú ju pošleme serveru. Server nám vráti cestu k adresárom a súborom a klient pošle späť správu o potvrdení. Nasledujúca komunikácia medzi klientom a serverom je rovnaká ako to bolo v bloku C. Znovu po prenesení posledného názvu sa metóda `SendFile` ukončí. Výsledkom po ukončení činnosti bloku E (a teda metódy `SendFile`) je zoznam všetkých adresárov a súborov, ktoré sa nachádzali vo vybratom adresári z predchádzajúceho kroku. Vývojový diagram bloku E metódy `SendFile` je na obr. 2.18.

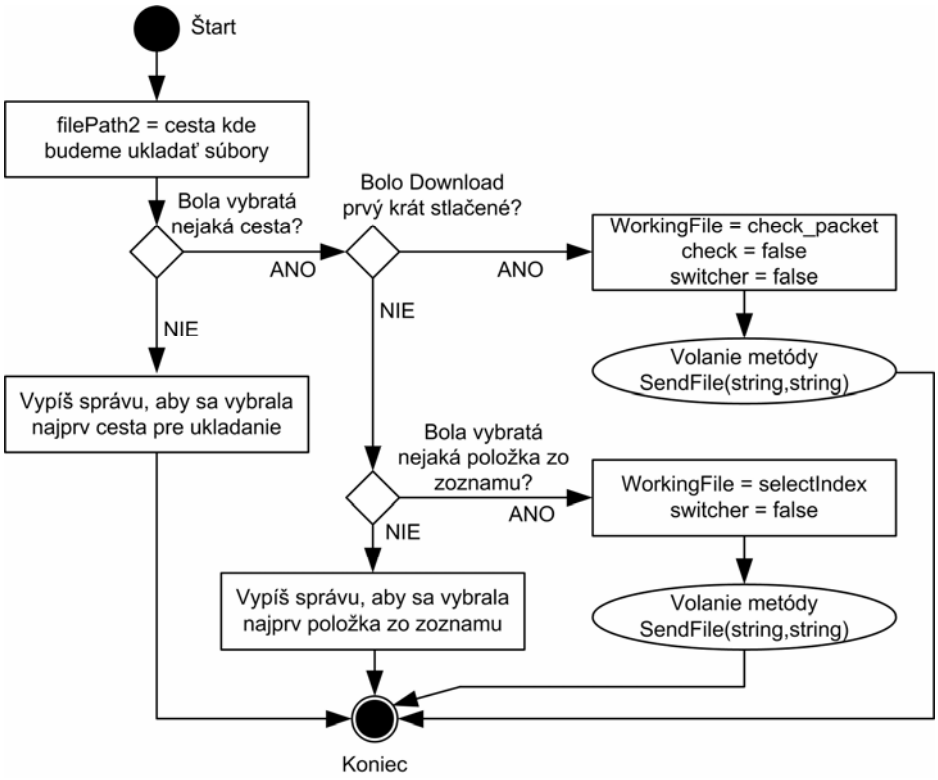


Obr. 2.18. Vývojový diagram metódy SendFile – BLOK E

Ďalšou metódou v triede FormConnection je metóda `buttonOpF_Click`, ktorá má za úlohu po stlačení tlačidla Open file otvoriť nové dialógové okno, ktoré je objektom triedy FormExplorer2 (kapitola 2.3.7). V novovytvorenom dialógovom okne si užívateľ môže vybrať skladbu, ktorú bude chcieť odosielať sieťou na server. Po stlačení tlačidla OK sa nastaví formulár FormConnection ako aktívny a vybratá cesta sa zapíše do atribútu *cestaOdoslat* a tiež do komponenty listBox. V poslednom kroku sa celá cesta k súboru rozdelí len na cestu k súboru a na samotný názov piesne. Z hľadiska ďalšieho použitia je to výhodnejšie. Po dokončení tejto metódy je názov vybratej skladby k odoslaniu zapísaný v atribúte *cestaOdoslat* a cesta k tejto skladbe v atribúte *filePath*.

Metóda `Download_Click` je volaná vždy po stlačení tlačidla Open/Download. Práve tu dôjde k rozlíšeniu, koľký krát bolo tlačidlo stlačené a podľa toho sa volá metóda

`SendFile` s príslušnými parametrami. Ako prvá sa zapíše do atribútu *filePath2* cesta, kde sa budú prípadne stiahnuté súbory ukladať. Ak by sa cesta nevybrala, automaticky po stlačení tlačidla sa na displeji objaví oznam o tom, aby sa najprv nejaká cesta vybrala. Ak sa však cesta vybrala, dochádza práve tu k rozhodovaniu o tom, koľký krát bolo stlačené tlačidlo Open/Download. K tomu slúži atribút *check*. To znamená, že po prvom stlačení tlačidla je volaná metóda `SendFile`, v ktorej sa bude vykonávať blok C (popísaný vyššie v texte) a po všetkých nasledujúcich stlačeniach blok B alebo blok E podľa toho, či ide o adresár alebo súbor. Hlavnou funkciou tejto metódy je teda volať metódu `SendFile` s príslušnými parametrami pre prenos. Vývojový diagram je vidieť na obr. 2.19.



Obr. 2.19. Vývojový diagram metódy Download_Click

V doterajšom popísanom texte o triede FormConnection bolo spomenutých veľa paketov so špeciálnym textom, ktoré sa v komunikácií medzi klientom a serverom využívajú. Pre sprehľadnenie situácie slúži tab. 2.1, kde môžeme vidieť všetky pakety, ktoré sú v komunikácií použité. Tiež je pri každom názve popísaná aj funkcia tohto paketu.

Tab. 2.1. Tabuľka paketov so špeciálnou funkciou

Typ paketu	Funkcia
xxx_start_xxx	Je poslaný ak je stlačené tlačidlo Open/Download prvý raz
xxx_level_xxx	Je poslaný ak je stlačené tlačidlo Level_UP
xxx_present_xxx	Je poslaný, ak sa zhoduje prijímaný názov súboru s názvom súboru v adresári určenom pre prijatie tohto súboru
xxx_end_xxx	Jw poslaný, ak sa má ukončiť prenos súboru

2.3.6 Trieda FormExplorer

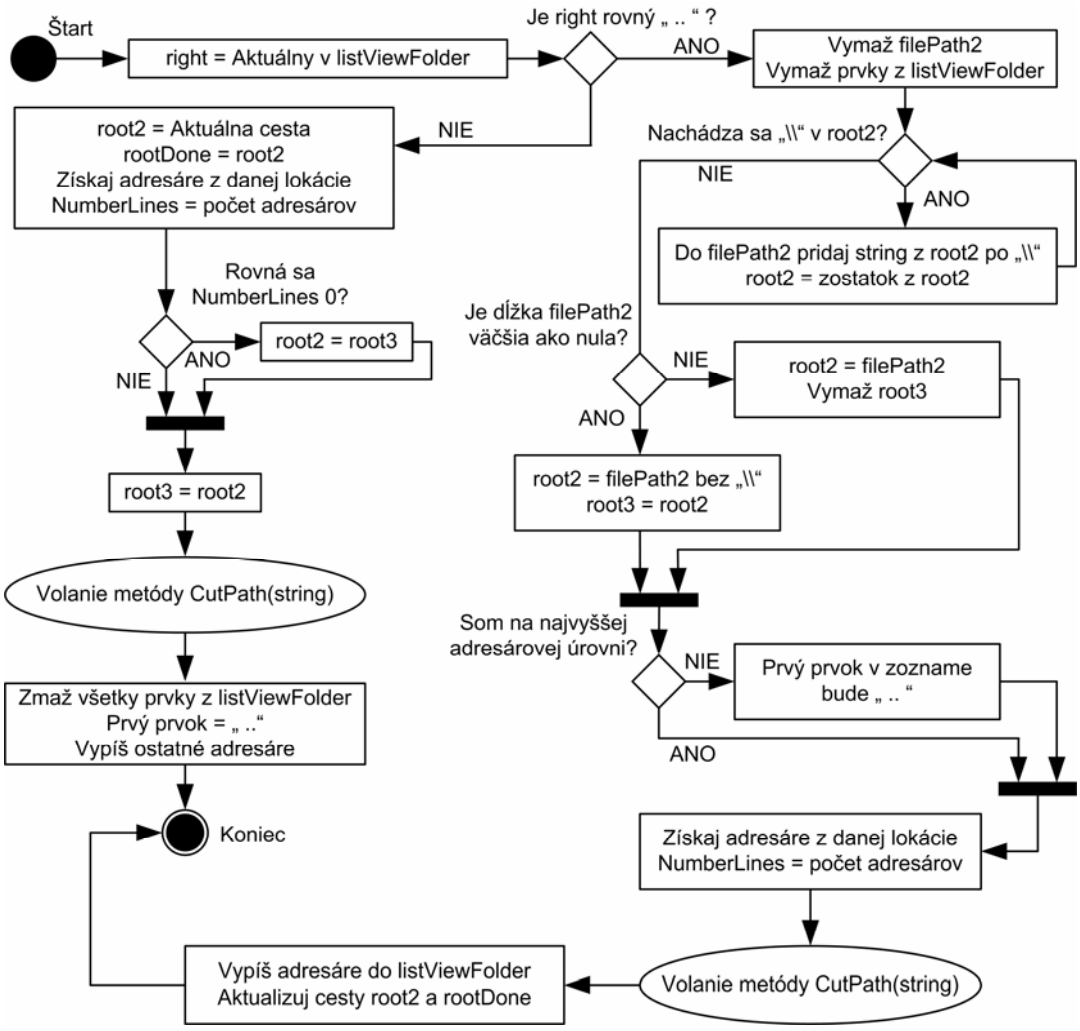
Trieda FormExplorer slúži v programe ako dialógové okno, kde si užívateľ môže vybrať adresár do ktorého sa budú ukladať prípadné stiahnuté súbory. Vo Visual Studiu 2008 sa nachádza komponent OpenFileDialog, ktorý slúži ako dialógové okno, no jeho nevýhodou je, že užívateľ si môže zadať cestu len do pamäti telefónu. Výhodou nášho vlastného dialógového okna je, že je možné vybrať cestu aj na pamäťovej karte, čo bude ocenené hlavne pri prijímaní väčšieho počtu skladieb.

Trieda FormExplorer obsahuje tri metódy, ktoré si v nasledujúcom texte popíšeme. Prvou z nich je metóda `FormExplorer`, ktorá slúži ako konštruktor danej triedy. Po vytvorení objektu triedy je volaná práve táto metóda. Jej úlohou je inicializovať komponenty na danom formulári a vypísať koreňovú adresárovú štruktúru do komponenty listView. To sa deje tak, že najprv sa získajú všetky názvy ciest adresárov pomocou metódy `GetDirectories`, ktoré sa nachádzajú v koreňovom adresári „\\“. Tieto cesty sú uložené do atribútu *path*. Hneď potom je volaná metóda `CutPath` z triedy Browser (kapitola 2.3.8), ktorá všetky cesty skráti len na samotné názvy skladieb. V nasledujúcom kroku sa všetky názvy vypíšu do komponenty listView a priradia sa k nim príslušné obrázky. Výsledkom je teda to, že po otvorení dialógového okna, užívateľ môže hneď začať vyberať adresár, kde bude chcieť ukladať sťahované skladby.

Ďalšou metódou je `listViewFolder_SelectedIndexChanged`. Metóda je volaná vždy pri zmene indexu v komponente listView. Predpokladá sa, že po každom kliknutí užívateľa na daný adresár si užívateľ chce obsah vybratého adresáru prezrieť a preto je volaná práve táto metóda. Na začiatku sa zapíše aktuálne vybratý prvok zo zoznamu do atribútu *right*. Hneď potom sa podmienkou overuje, či sa *right* rovná „ .. “, čo by znamenalo, že ide o presun na vyššiu úroveň v adresárovej štruktúre. Ak sa však *right* nerovná tomuto reťazcu, znovu sa získajú všetky adresáre z vybratého adresáru pomocou metód, ktoré boli popísané

vyššie v texte pri opisovaní činnosti metódy `FormExplorer`. Keby sa zistilo, že vybratý adresár v sebe neobsahuje žiadne ďalšie adresáre, vybratý prvok v zozname ostane len označený a nič iné sa nezobrazí. Ak by sa *right* rovnal reťazcu „ .. “, vypísali by sa adresáre, ktoré sa nachádzajú o úroveň vyššie. Podmienkou sa však ešte pred samotným vypisovaním prvkov zoznamu overí, či sme na koreňovej časti adresárovej štruktúry. V prípade, že áno, tak prvý prvok zoznamu sa nevypíše „ .. “, ale len prvky zoznamu ako to bolo v prípade volania metódy `FormExplorer`. Pre informáciu užívateľa sa zobrazuje vždy aktuálna cesta, aby vedel, kde sa momentálne v adresárovej štruktúre nachádza. Vývojový diagram metódy `listViewFolder_SelectedIndexChanged` je vidieť na obr. 2.20.

Po vybratí cesty sa potvrdí vybratie stlačením tlačidla OK a zavolá sa metóda `buttonOK_Click`. Jej úlohu je len zavrieť dialógové okno a nastaviť DialogResult na `OK`, aby mohlo pokračovať vykonávanie programu v triede FormConnection.



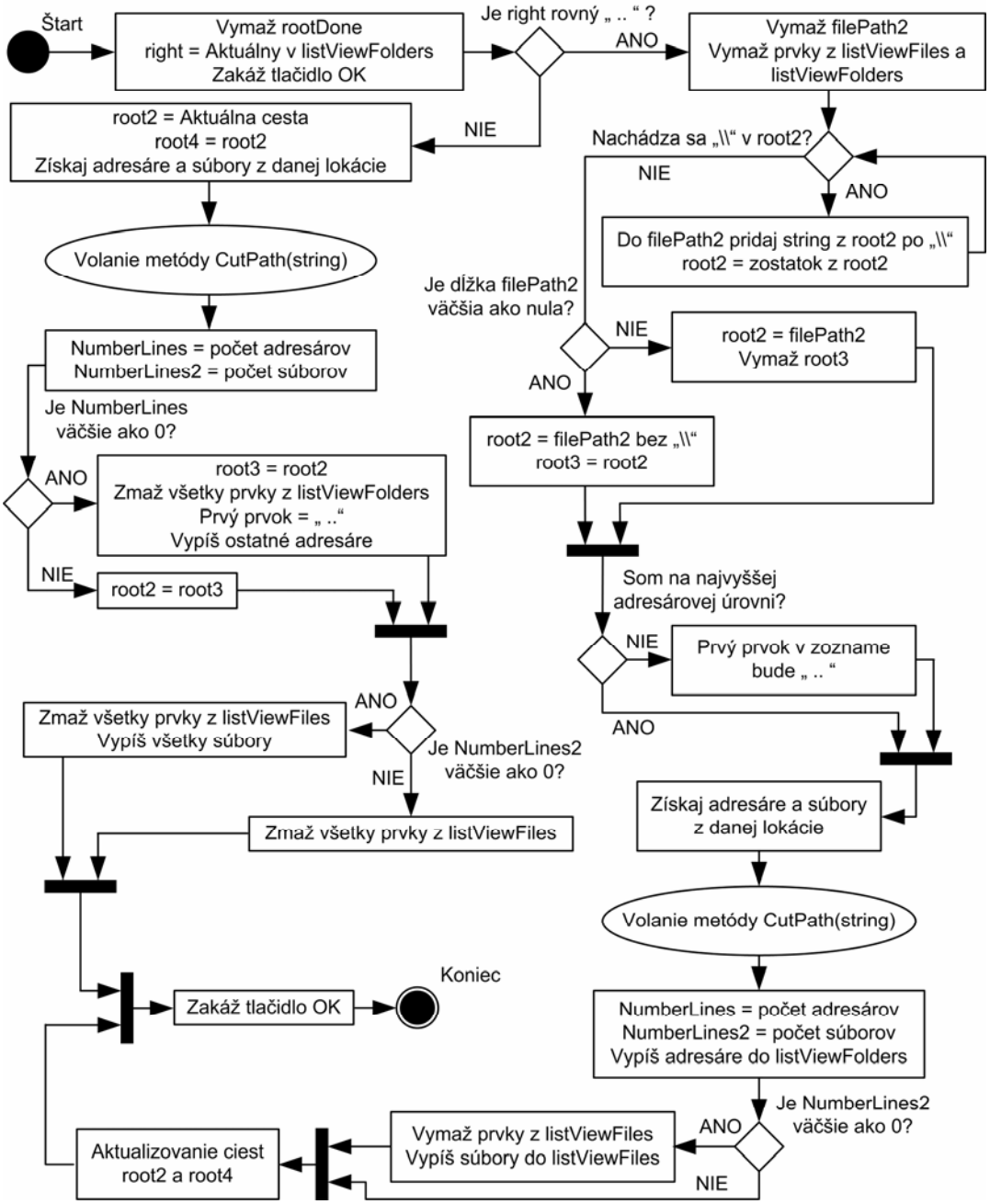
Obr. 2.20. Vývojový diagram komponenty listViewFolder

2.3.7 Trieda FormExplorer2

Trieda FormExplorer2 je z hľadiska funkčnosti veľmi podobná ako trieda FormExplorer. Hlavný rozdiel je v tom, že vo FormExplorer sa vyberá len cesta (adresár), kde sa budú skladby ukladať a vo FormExplorer2 sa vyberá konkrétny súbor k odoslaniu alebo súbor, ktorý sa má pridať do playlistu prehrávača. Po vytvorení objektu triedy FormExplorer2 sa vytvorí nový formulár, ktorý obsahuje dve tlačidlá a dve komponenty listView. Jedna slúži pre výpis adresárov a druhá pre výpis súborov. Trieda obsahuje päť metód.

Prvou z nich je opäť konštruktor, ktorý má presne tú istú funkciu ako konštruktor v triede FormExplorer (kapitola 2.3.6). V poradí nesledujúcou metódou je `listViewFolders_SelectedChanged`. Tá je volaná po zmene indexu v komponente listViewFolders. Ako prvé zakáže tlačidlo OK a vloží do atribútu *right* aktuálny prvok zoznamu. Ďalej sa podmienkou testuje, či sa vybral nejaký adresár alebo ide o vybratie reťazca „ .. “, čo by znamenalo, že sa máme posunúť v adresárovej štruktúre o úroveň vyššie. Ostatné kroky sú podobné ako v metóde `listViewFolder_SelectedIndexChanged` z triedy FormExplorer (kapitola 2.3.6). Rozdiel je len v tom, že sa vypisujú aj samotné súbory (len s príponou mp3) do komponenty listViewFiles, ktoré sa prípadne nachádzajú vo vybratom adresári. Vývojový diagram metódy je vidieť na obr. 2.21.

V predchádzajúcej metóde sa vždy pri zmene indexu zakázalo tlačidlo OK. Je to z toho dôvodu, aby užívateľ nemohol stlačiť toto tlačidlo pokiaľ nie je vybratá nejaká skladba. O povoľovanie tlačidla OK sa stará metóda `listViewFiles_SelectedIndexChanged`, ktorá sa volá po zmene indexu v komponente listViewFiles. To znamená, že vždy pri vybratí konkrétnej skladby sa tlačidlo OK povolí a tak je užívateľovi umožnené potvrdenie výberu danej skladby.



Obr. 2.21. Vývojový diagram komponenty listViewFolders

Posledné dve metódy `buttonAddFile_Click` a `buttonCan_Click` sa volajú po stlačení tlačidiel OK a Cancel. Podstata obidvoch je skryť aktuálny formulár. Rozdiel je, že po stlačení OK sa nastaví DialogResult na hodnotu `OK` a po stlačení tlačidla Cancel na hodnotu `Cancel`. Takto dáme vedieť rodičovskému formuláru, z ktorého je dialóg volaný, s akým výsledkom prebehol. Podľa toho sa bude pokračovať vo vykonávaní kódu.

2.3.8 Trieda Browser

Trieda Browser obsahuje len dve jednoduché metódy, ktoré sú volané z iných tried. Aj keď sú metódy jednoduché, ich funkcia je v aplikácií je veľmi dôležitá.

Prvou z nich je metóda `CutPath`, ktorej vstupnou aj výstupnou hodnotou je matica reťazcov. Jej úlohou je skrátiť všetky vstupné reťazce, ktoré v našom prípade predstavujú rôzne cesty k súborom a adresárom a poslať späť len samotné názvy. To sa využíva pri vypisovaní adresárovej štruktúry v triedach FormExplorer a FormExplorer2.

Druhou metódou je `IsFilePresent`, ktorá má za úlohu zistiť, či sa prijímaný súbor (skladba) nenachádza s rovnakým názvom vo vybratom adresári určeným pre sťahovanie súborov. Metóda má dve vstupné hodnoty a to názov cesty adresára určeného pre prijímanie súborov a názov skladby, ktorú v ňom bude metóda hľadať. Ak sa požadovaná skladba v tomto adresári nájde, výstupnou hodnotou metódy bude `true`, v opačnom prípade `false`.

2.4 Sieťová komunikácia

Predstavme si situáciu, že užívateľ našej multimediálnej aplikácie chce okrem prehrávania audio súborov, ktoré sa nachádzajú v pamäti PDA alebo na pamäťovej karte, prehrávať aj iné skladby, ktoré má umiestnené niekde na svojom stolovom (desktopovom) počítači doma alebo v kancelárií. Samozrejme je možné, aby si požadované súbory nahral z takéhoto počítača do svojho mobilného zariadenia pomocou USB káblu alebo pomocou pamäťovej karty. No všetky tieto možnosti je schopný využiť len v prípade, že sa nachádza v blízkosti daného počítaču. Ak sa však v blízkosti počítača nenachádza, jednou z alternatív, akou sa môže na daný počítač pripojiť, je práve sieťová komunikácia.

Aplikácie, ktoré bežia na mobilných zariadeniach a komunikujú z druhým zariadením (mobilné zariadenie, stolový počítač) prostredníctvom sieťovej komunikácie sú z pravidla založené na centralizovanom modeli klient-server, kde klient využíva služby poskytované serverom alebo na distribuovanom modeli, kde sú aplikácie rovnocenné. Každý z týchto modelov má svoje výhody a nevýhody. V našej mobilnej aplikácií, v ktorej budeme chcieť prijímať, poprípade odosielať audio súbory použijeme práve model klient-server. Ako komunikácia medzi klientom a serverom prebieha si popíšeme v nasledujúcich podkapitolách.

2.4.1 Server

Serverová aplikácia bude bežať na stolovom (desktopovom) počítači pod operačným systémom Microsoft Windows (testované vo verzii XP). Prvým dôležitým krokom je vytvoriť inštanciu triedy `IPEndPoint`, ktorá obsahuje kombináciu IP adresy a portu z ktorej sa je možné na server pripojiť.

```
IPEndPoint ipEnd = new IPEndPoint(IPAddress.Any, 5656);
```

V našom prípade použitím `IPAddress.Any` bude server akceptovať pripojenie z ktorejkoľvek IP adresy na porte 5656.

Ďalším dôležitým krokom je nastavenie parametrov socketu. Nastavenie má tri základné parametre. Prvým je *AddressFamily*, ktorý definuje typ siete, druhým je *SocketType*, ktorý definuje typ dátového pripojenia a posledným je *ProtocolType*, ktorý definuje typ použitého protokolu.

```
Socket socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
                ProtocolType.Tcp);
```

Pre normálnu IP komunikáciu na sieti sa pre *AddressFamily* používa *InterNetwork*. Keďže chceme použiť v našej komunikácií TCP protokol, musíme nastaviť *SocketType* na *Stream* a *ProtokolType* na *Tcp*. Týmto definujeme, že pôjde o spoľahlivú komunikáciu medzi klientom a serverom.

Protokol TCP je spoľahlivá doručovacia spojovaná služba. Dáta sú prenášané v segmentoch. Spojovaná služba znamená, že pred samotnou výmenou dát medzi hostiteľmi musí byť ustanovené spojenie. Spoľahlivosť je dosiahnutá priradením poradového čísla každému prenášanému segmentu, pričom prijatie všetkých segmentov ďalším hostiteľom sa overuje potvrdením ich prijatím. U každého odosielaného segmentu musí do určitej doby prijímací hostiteľ vrátiť potvrdenie (ACK) prijatých bajtov. Ak k potvrdeniu nedôjde, sú dáta prenášané znovu. Protokol TCP používa komunikáciu na báze bajtových prúdov, v ktorých sú dáta TCP segmentu považované za sled bajtov bez hraníc záznamov. Odosielateľ na jednom uzlovom počítači postupne predáva protokolu TCP jednotlivé byty (oktety), a obdobne postupuje aj príjemca, ktorý ich na druhej strane odoberá. Najdôležitejšie polia hlavičky TCP paketu môžeme vidieť v tab. 2.1 [8].

Tab. 2.1. Prehľad kľúčových polí v TCP hlavičke

Pole	Funkcia
Zdrojový port	TCP port odosielateľa.
Port miesta určenia	TCP port príjemcu.
Poradové číslo	Poradové číslo prvého bajtu v TCP segmente.
Číslo potvrdenia	Poradové číslo bajtu, ktorý odosielateľ očakáva ako ďalší od druhej komunikujúcej strany.
Okno	Aktuálna veľkosť TCP vyrovnávacej pamäti na hostiteľovi odosielajúcom TCP segment určené k ukladaniu prichádzajúcich segmentov.
TCP kontrola	Overuje integritu TPC hlavičky a TCP dát.

Ďalším krokom je pomenovanie vytvoreného socketu. O to sa stará metóda `bind`. Tá priraďuje adresu nepomenovanému socketu (pripojí socketu meno). Metóda `listen` počúva, respektíve čaká na pripojenia k socketu. Tento príkaz identifikuje socket, ktorý „prijíma“ spojenie a limituje „backlog“, čiže množstvo prichádzajúcich požiadaviek na spojenie. Server môže byť schopný obslúžiť viacej klientov. Keďže však v našej aplikácií predpokladáme, že sa bude pripájať vždy len jeden klient, nastavíme hodnotu „backlog“ na jedna.

```
socket.Bind(ipEnd);
socket.Listen(1);
clientSock = socket.Accept();
```

Metóda `accept` akceptuje nové spojenie na sockete z prichádzajúceho pokusu o spojenie od klienta. Metóda vyberie prvé spojenie z fronty čakajúcich spojení, vytvorí nový socket s rovnakými vlastnosťami ako špecifikovaný socket a vytvorí pre neho nový file deskriptor. Potom toto nové spojenie prebieha cez novovytvorený socket. Tento sám o sebe nemôže prijať ďalšie spojenia, ale pôvodný (originálny) socket je otvorený a ten nové spojenia prijať môže. Ak je fronta prázdna, bez ďalších požiadaviek na spojenie, metóda `accept` blokuje socket, až pokiaľ nie je prítomná požiadavka na spojenie.

Takýmto spôsobom je všetko pripravené na spojenie. Server je pripravený prijať požiadavku o spojenie z akejkoľvek IP adresy na porte 5656. Po prijatí klienta môže začať samotná komunikácia pomocou metód `send` a `receive`.

2.4.2 Klient

Klientska časť modelu klient-server bude bežať na mobilnom zariadení. Je to tá časť aplikácie, ktorá sa snaží pripojiť k serveru. Server bude bežať niekde na vzdialenom počítači, z ktorého si užívateľ bude chcieť stiahnuť požadované skladby.

Prvým dôležitým krokom, podobne ako aj na servery je vytvoriť inštanciu triedy `IPEndPoint`. Tá sa bude opäť skladať z IP adresy a portu. Port je rovnaký ako na servery, teda 5656, ale IP adresa sa zadáva konkrétna pomocou metódy `IPAddress.Parse`. Ide o IP adresu serveru.

```
IPEndPoint ipEnd = new IPEndPoint(IPAddress.Parse(IP), 5656);
Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
                ProtocolType.Tcp);    //vytvorenie noveho socketu
server.Connect(ipEnd);                //pripojenie sa na server
```

Ďalším krokom je vytvorenie socketu. Ten je rovnaký ako tomu bolo na servery, teda ide opäť o TCP komunikáciu. Nemusíme však už použiť metódu `bind`, pretože na strane klienta sa používajú anonymné sockety. Ak máme takto vytvorený jeden koniec spojenia na strane klienta, musíme ho spojiť s druhým koncom na strane serveru. O to sa stará metóda `connect`. Tá vykoná pripojenie klienta na server. Pokiaľ sa spojenie nepodarí nadviazať, vyhodí sa výnimka o zlyhaní spojenia. To by mohlo viesť až k zlyhaniu aplikácie. Preto je dobré mať metódu `connect` v bloku try-catch, ktorý nám prípadné zlyhanie pripájania zachytí. Po pripojení klienta na server sa môžu začať odosielať a prijímať dáta pomocou metód `send` a `receive`.

2.4.3 Nadväzovanie a ukončenie spojenia

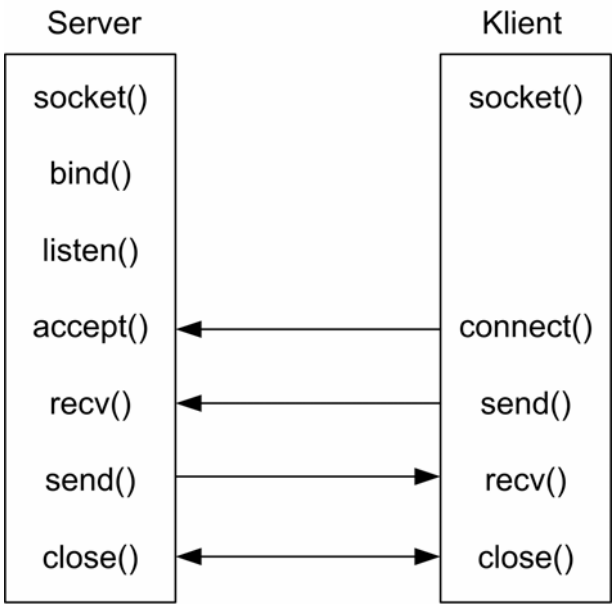
Táto kapitola slúži pre lepšie pochopenie ako sa nadväzuje a ukončuje TCP spojenie medzi klientom a serverom.

Vieme už, že protokol TCP je spoľahlivý, spojovo orientovaný protokol pomocou ktorého sa zriadi komunikačná cesta medzi dvomi IP adresami (endpointami). Protokol TCP umožňuje jednej strane nadväzovať spojenie. Druhá strana spojenie buď akceptuje alebo odmietne. V TCP komunikácií hovoríme o trojcestnom handshake. Z hľadiska aplikačnej vrstvy bude stranou nadväzujúcou spojenie klient a server tá strana, ktorá spojenie očakáva. Náš server beží na porte 5656. Ak teda chce klient nadviazať spojenie so serverom musí sa na tento port pripojiť. Je prakticky jedno na akom porte bude aplikácia bežať, no port musí byť väčší ako 1023. O porty pod hodnotou 1024 môže žiadať len privilegovaný užívateľ. Porty nad hodnotou 1023 sa označujú klientske alebo aj neprivilegované.

Zatiaľ čo spojenie nadväzoval v architektúre klient-server spravidla klient, tak ukončiť spojenie môže ľubovoľná strana. Strana, ktorá prvá odošle TCP segment s príznakom FIN (ukončenie spojenia), urobí takzvané aktívne ukončenie spojenia. Druhej strane neostáva nič

iné len urobiť pasívne ukončenie spojenia. V našom programe to znamená uzatvorenie socketu na serverovej alebo klientskej časti metódou `Close`.

V kapitole 2.4.1 a 2.4.2 sú popísané všetky metódy, ktoré sú potrebné pre spojenie klienta so serverom. Pre lepšie pochopenie je naznačená komunikácia na obr. 2.22.



Obr. 2.22. Naznačenie komunikácie medzi klientom a serverom

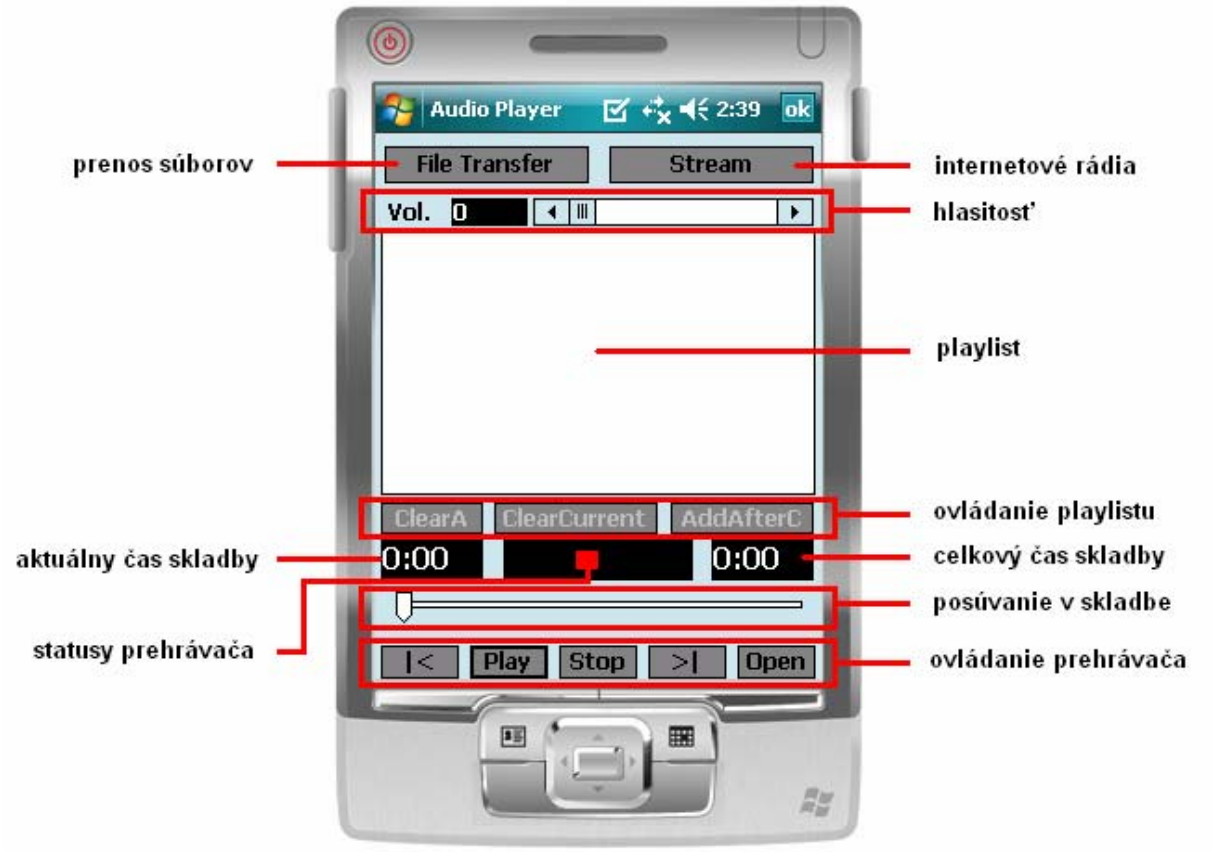
2.5 Popis ovládania programov

V tejto kapitole si popíšeme ovládanie obidvoch vytvorených aplikácií. Najprv sa zameriame na ovládanie mobilnej multimediálnej aplikácie, ktorá bude bežať na PDA pod operačným systémom Microsoft Windows Mobile 6.0 a potom si popíšeme serverovú aplikáciu, ktorá bude bežať niekde na stolovom (desktopovom počítači) pod operačným systémom Microsoft Windows XP.

2.5.1 Mobilná multimediálna aplikácia

Po spustení aplikácie sa zobrazí užívateľské prostredie, ktoré je vidieť na obr. 2.23. Užívateľ si tu má možnosť vybrať z viacej možností, ktoré by chcel previesť. Ak by sa rozhodol len pre prehrávanie skladieb, ktoré má uložené v pamäti telefónu alebo na pamäťovej karte, ako prvé čo musí urobiť je stlačiť tlačidlo Open. Po stlačení sa mu zobrazí dialógové okno, kde si môže vybrať skladbu, ktorú bude chcieť prehrávať. Po vybratí sa táto skladba pridá do playlistu. Vždy po ďalšom stlačení sa ostatné skladby budú postupne

pridávať na koniec playlistu. Takto si môže užívateľ vytvoriť playlist, presne podľa svojich predstáv. Hneď ako sa skladba pridá do playlistu je pripravená na prehrávanie. Stačí ju len v playliste označiť a stlačiť tlačidlo Play. Počas prehrávania si užívateľ môže skladbu zastaviť tlačidlom Stop, poprípade sa v playliste pohybovať pomocou šípok (tlačidlá Next a Prev), ktorými sa prepína na nasledujúcu alebo predchádzajúcu skladbu. Pri každom prehrávaní sa automaticky ukazuje celkový čas vybratej skladby a tiež aktuálny čas, kde sa práve prehrávanie nachádza. Na posúvanie sa v skladbe slúži posuvný bežec, ktorý môžeme nájsť hneď nad ovládaním prehrávača (obr. 2.23). Pomocou neho si užívateľ môže presne nastaviť čas, od ktorého chce počúvať určitý úsek piesne. Nad oknom playlistu je nastavovanie hlasitosti, kde si kedykoľvek užívateľ môže nastaviť úroveň hlasitosti prehrávania. Hneď vedľa je ukazovateľ, ktorý ukazuje túto úroveň v percentách. Slúži hlavne pre lepšiu orientáciu.



Obr. 2.23. Popis častí ovládania multimediálneho prehrávača

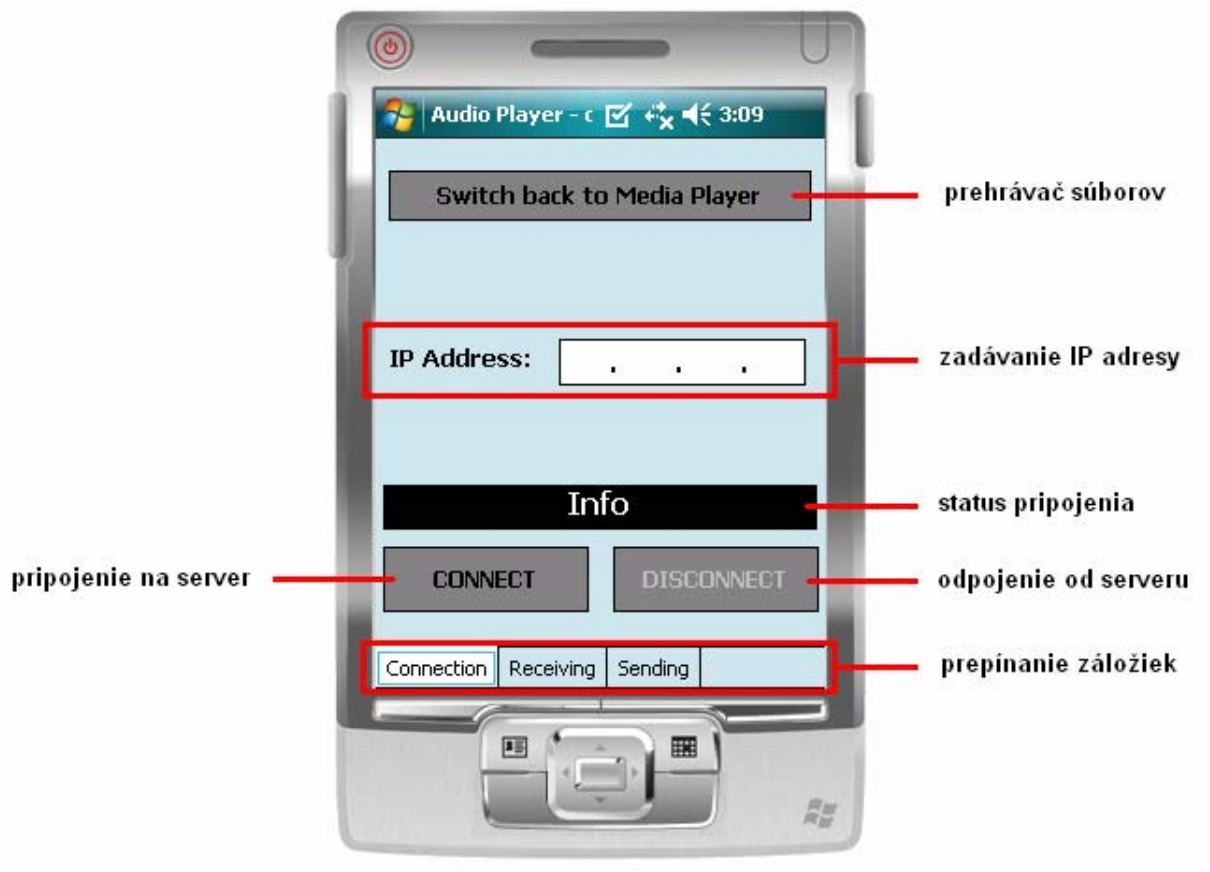
Ak by sa však užívateľ rozhodol, že si chce vypočuť skladbu, ktorá sa nenachádza v pamäti telefónu, ale na počítači doma, môže si túto skladbu stiahnuť a prehrať. Je to veľká výhoda, lebo nemusíme zo sebou nosiť obrovské množstvo piesní. Ak si bude chcieť užívateľ skladbu prehrať a nebude momentálne nahratá na PDA, môže si ju kedykoľvek stiahnuť.

Jedinou podmienkou je, aby bol pripojený na internet. Ak je zariadenie pripojené do siete, musí sa stlačiť tlačidlo File Transfer a objaví sa nové dialógové okno, ktoré je vidieť na obr. 2.24. Tu je treba zadať IP adresu počítača na ktorom beží serverová aplikácia. Ak je IP adresa zadaná správne a klientovi sa podarí pripojiť, vypíše sa správa „Server is connected“ do statusu pripojenia. Od tejto chvíle je povolené prepínanie medzi záložkami Receiving a Sending (obr. 2.24), z ktorých v záložke Receiving je možné sťahovať skladby zo serveru a naopak v záložke Sending ich na server posielať. Vždy ale pred samotným prijímaním súborov sa musí určiť adresár, do ktorého sa budú prijímané dáta ukladať. Bez toho by samotné prijímanie dát nebolo aplikáciou povolené. Ako prvý sa zo serveru stiahne obsah adresára, ktorý bol vybratý ako domovský adresár pre klienta na servery (kapitola 2.5.2). Potom sa už môžeme pohybovať v adresárovej štruktúre ako keby to bolo priamo na lokálnom pamäťovom médiu. Vždy sa označí príslušný adresár, ktorého obsah si chceme stiahnuť a stlačí sa tlačidlo Open/Download. Tlačidlom Level_UP sa dostaneme vždy o úroveň vyššie v adresárovej štruktúre. Po tom ako sa užívateľ dopracuje k požadovanej skladbe, označí ju a opäť sa stlačí tlačidlo Open/Download. Aplikácia automatický rozpozná, že ide o súbor a začne sa sťahovanie. Po ukončení prenosu sa vypíše správa o úspešnom prenesení celého súboru. Pre návrat do ovládania prehrávača sa musí stlačiť tlačidlo Switch back to Media Player v záložke Connection. Tu sa môže opäť nová prijatá skladba pridať do playlistu a prehrať.

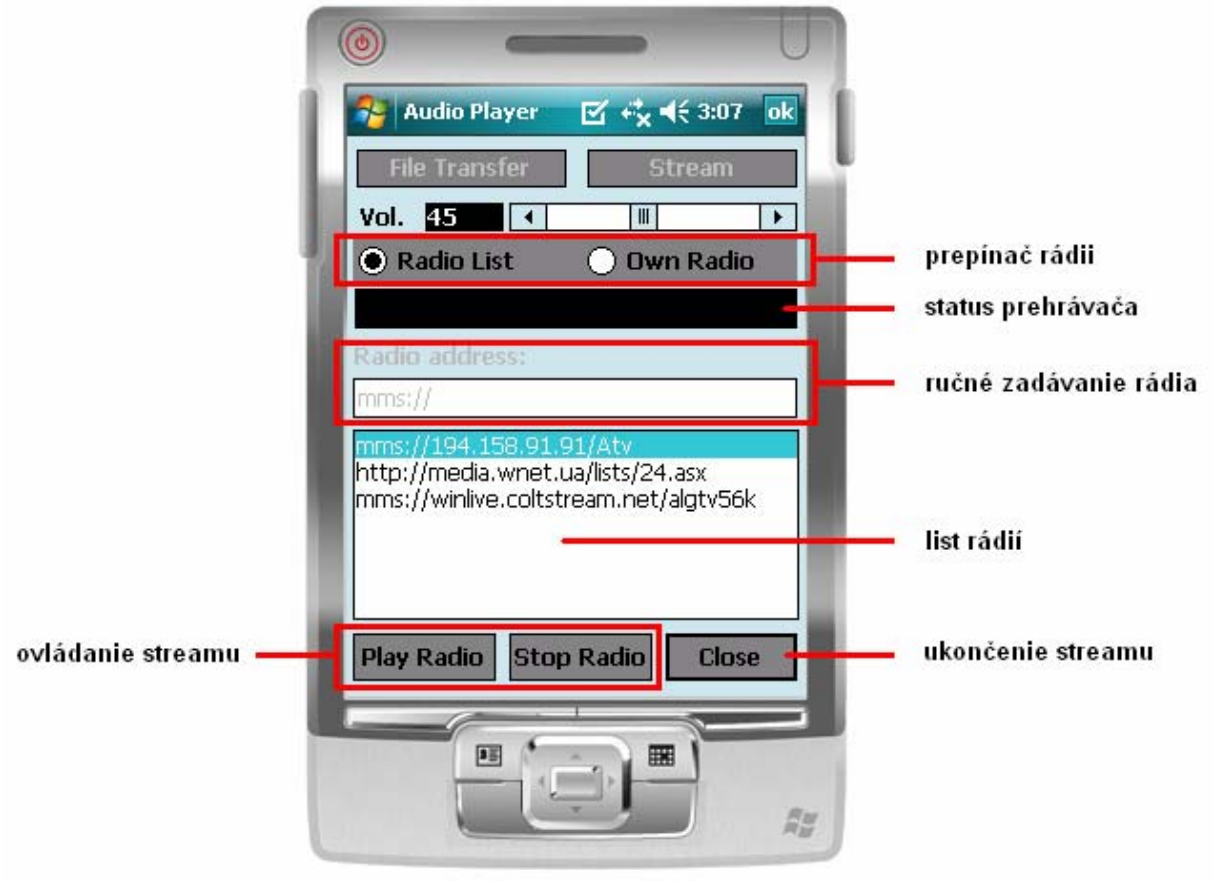
Ak sa užívateľ rozhodne pre spustenie internetového rádia, musí ako prvé stlačiť tlačidlo Stream (obr. 2.23). V novovytvorenom okne, ktoré je vidieť na obr. 2.25 sa môže rozhodnúť, či zadá adresu rádia ručne alebo si vyberie z už preddefinovaných staníc. Po zadaní cesty k rádiu už len stačí stlačiť tlačidlo Play Radio a prehrávanie sa za pár sekúnd spustí. Dĺžka čakania závisí od toho, akú má rádio práve dostupnosť. Ak by však na zadanej adrese žiadne rádio neexistovalo, užívateľovi sa o tom zobrazí správa. Pre zastavenie prehrávania rádia slúži tlačidlo Stop Radio. Tlačidlom Close tiež zastavíme prehrávanie rádia, no okrem toho sa aj zatvorí príslušný panel pre ovládanie rádia a dostaneme sa späť do ovládania prehrávača skladieb.

Poslednou časťou v aplikácií je ovládanie playlistu. Ide o veľmi užitočnú vec, ktorá sa hodí hlavne pre náročnejších užívateľov. Obsahuje tri tlačidlá a to ClearA, ClearCurrent a AddAfterC. Tlačidlo ClearA slúži na vymazanie celého playlistu, teda ak bude treba vymazať celý playlist, napríklad z dôvodu vytvorenia nového, tlačidlo ClearA je na tento účel presne určené. Tlačidlo ClearCurrent vymaže aktuálny, práve označený prvok z playlistu

a nastaví ako aktuálny prvok ďalší v poradí, poprípade ak sa nachádzame už na poslednom v playliste, nastaví ako aktuálny predchádzajúci v poradí. Posledné tlačidlo je AddAfterC. Jeho funkcia je pridať novú skladbu do playlistu. Ide o veľmi podobnú funkciu ako má tlačidlo Open. Je tu len jeden rozdiel a to taký, že po stlačení AddAfterC sa nepridá nová skladba vždy na koniec playlistu ako to bolo pri Open, ale vždy za aktuálnu, práve označenú skladbu. Pomocou týchto troch tlačidiel si môže užívateľ meniť playlist veľmi rýchlo a flexibilne. Grafický dizajn aplikácie a popis jej jednotlivých častí je vidieť na obr. 2.23 až obr. 2.25.



Obr. 2.24. Popis častí ovládania pripojenia na server

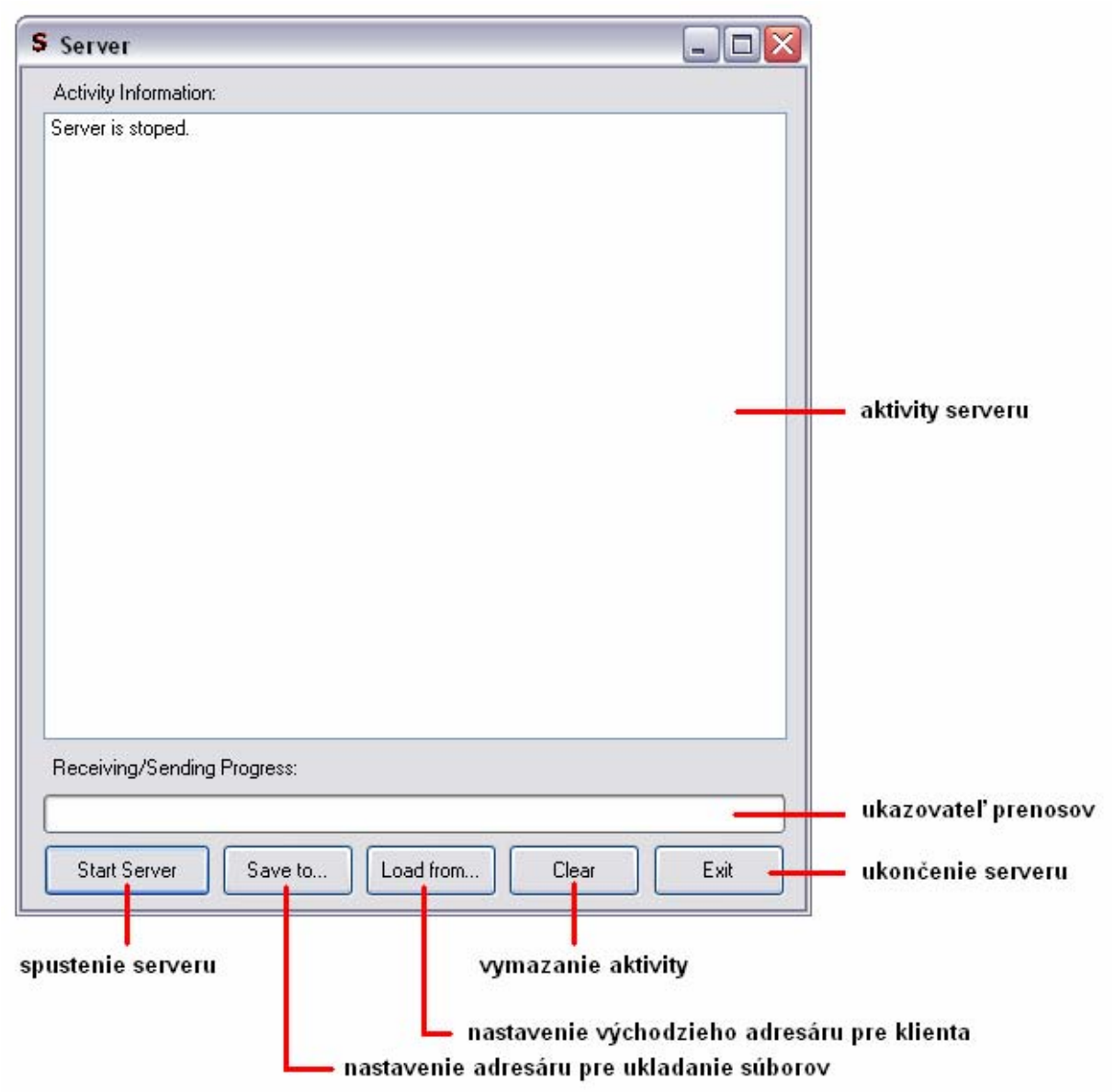


Obr. 2.25. Popis častí ovládania internetových rádií

2.5.2 Serverová aplikácia

Serverová aplikácia je nedielnou súčasťou našej multimediálnej mobilnej aplikácie. Pre mobilnú aplikáciu vytvára na vzdialenom počítači server, z ktorého chce užívateľ sťahovať súbory. Po spustení aplikácie sa zobrazí užívateľské prostredie, ktoré je vidieť na obr. 2.26. Pred samotným spustením serveru sa musia nastaviť dve cesty. Prvá sa nastaví pomocou tlačidla Save to, ktoré slúži na vybratie adresára, kde sa budú ukladať prípadné prijaté skladby z mobilného zariadenia. Druhá cesta sa nastaví po stlačení Load from. Ide o adresár, ktorý bude slúžiť ako domovský adresár pre pripájajúceho sa klienta. To znamená, že užívateľ si vyberie akýkoľvek adresár, ktorého obsah sa zobrazí ako prvý na mobilnom zariadení. Prakticky je však jedno aký adresár si účastník zvolí, pretože v tomto adresári nie je uzamknutý a voľne sa môže pohybovať v adresárovej štruktúre na servery. Po zapnutí aplikácie nie je nastavená ani jedna z týchto ciest, preto ich užívateľ podľa potreby musí zadať. Bez toho by sa mu server nepodarilo spustiť a zobrazovala by sa mu správa „Please set receiving and loading paths“. Ak však obidve cesty zadá správne, server sa spustí a v okne

Activity Information sa o tom zobrazí užívateľovi správa. Takto je server pripravený na prichádzajúce spojenie, ktoré očakáva na porte 5656. Všetky informácie, ktoré pri spojení prebehnú sa zobrazujú do okna Activity Information. Pomocou tlačidla Clear má užívateľ možnosť všetky informácie z tohto okna vymazať. Prebiehajúce prenosy súborov, či už prichádzajúce alebo odchádzajúce, môže sledovať pomocou ukazovateľa prenosov. O ostatnú funkčnosť a beh serveru sa nemusí už vôbec starať. Po spustení serveru všetko beží v takzvanom bez obslužnom režime. To znamená, že pri akomkoľvek výpadku na sieti sa server automaticky reštartuje a čaká opäť na prichádzajúcu požiadavku o spojenie. Užívateľ môže server ukončiť tlačidlom Exit, ktorým zároveň aj uzavrie aplikáciu.



Obr. 2.26. Užívateľské prostredie serverovej aplikácie

2.5.3 Výnimky aplikácií

Obidve aplikácie okrem správnej funkčnosti musia zabezpečovať a ošetrovať veľké množstvo výnimiek. Tieto časti kódov nie sú vždy z hľadiska používania aplikácie užívateľom viditeľné, ale predstavujú dôležité časti, ktoré riešia veľa situácií.

Prvou z nich je ošetrenie všetkých tlačidiel v aplikáciách. Tie sú zabezpečené proti tomu, aby neboli stlačené v nevhodnej chvíli, poprípade aby sa užívateľovi zobrazila príslušná informačná správa o tom, čo má urobiť. Príkladom využitia zakázania tlačidla je situácia, keď sa klient pripojí na server. V tejto chvíli je jasné, že tlačidlo Connect už druhý krát nemôže byť stlačené, preto sa zakáže. Tlačidlo sa opäť povolí, keď sa klient od serveru odpojí. Príkladom informačnej správy je, ak užívateľ stlačí hneď po otvorení serverovej aplikácie tlačidlo pre štart serveru. Zobrazí sa mu správa o tom, aby najprv zadal cestu pre príjem a odosielanie súborov. Bez zadaných ciest sa mu server nepodarí spustiť.

Ďalším ošetrením aplikácie je správna práca s vláknami. Pri každom spustení piesne sa vytvorí nové vlákno, ktoré slúži na ukazovanie časov prehrávanej skladby. Keď nastane situácia prechodu z práve prehrávanej skladby na ďalšiu alebo predchádzajúcu, musí sa najprv toto vlákno korektne ukončiť a až potom je aplikácií umožnené prehrávanie inej skladby. Je to z toho dôvodu, aby sa nevytváral stále väčší počet vlákien. Takto vieme, že je spustené vždy len jedno vlákno a máme nad ním kontrolu. Väčší počet vlákien by mohol spôsobovať nežiaduce náhodné situácie pri prehrávaní a väčšiu spotrebu batérií mobilného zariadenia. Taktiež pri zatváraní aplikácie sa pred jej samotným ukončením musí najprv ukončiť vlákno a až potom dôjde k samotnému vypnutiu.

Predstavme si situáciu, kedy komunikuje server s klientom a zrazu dôjde k výpadku spojenia niekde na prenose. Obidve aplikácie výpadok spojenia dokážu rozpoznať. Na klientskej časti sa užívateľovi vypíše správa o strate signálu a aplikácia sa prepne späť na záložku pripojovania, kde užívateľ môže opäť zadať IP adresu a pokúsiť sa pripojiť na server. Server po rozpoznaní akejkoľvek straty s klientom sa do sekundy reštartuje a tým je znova pripravený na prichádzajúce spojenie. Takto je zabezpečený bez obslužný chod serveru, kedy po prvotných nastaveniach ciest a spustení, už užívateľ nemusí robiť nič iné. Všetky popísané výnimky a ošetrenia sú dôležitou súčasťou pre správny chod aplikácií.

3 SYNCHRONIZÁCIA A PODPOROVANÉ FORMÁTY

3.1 ActiveSync 4.5

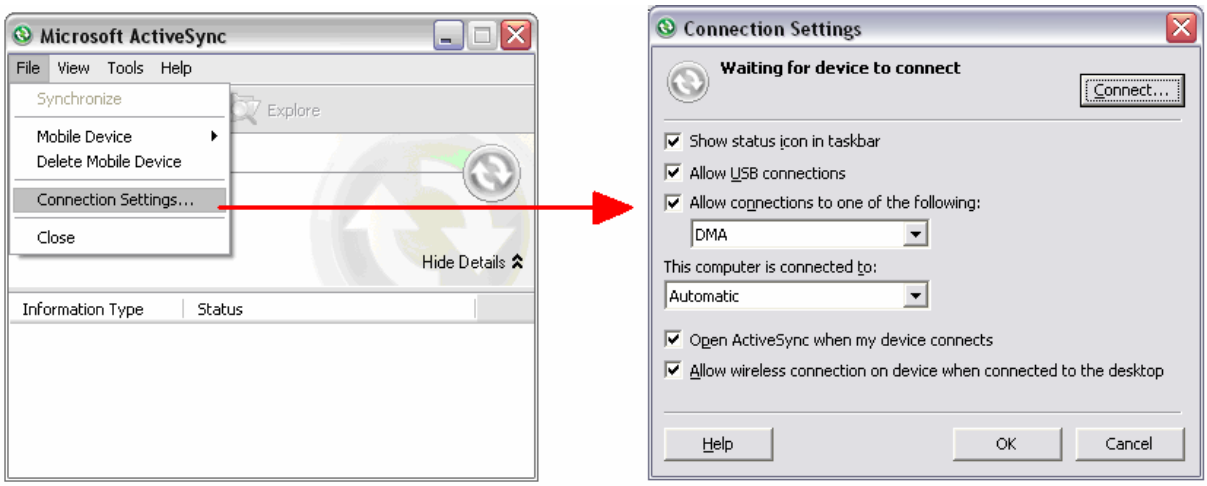
ActiveSync je dátový synchronizačný program vyvíjaný firmou Microsoft, ktorý sa používa pre synchronizáciu so všetkými verziami operačného systému Microsoft Windows. Prvá verzia programu vyšla v roku 1996 pod menom Handheld PC Explorer. V roku 1999 bol program premenovaný na ActiveSync a tento názov mu zotrval až do dnes. Najnovšia verzia ActiveSync 4.5 ponúka pre užívateľov jednoduchý a rýchli transport a synchronizáciu dokumentov, kontaktov, kalendáru a emailov medzi stolovým počítačom a mobilným zariadením.

3.2 Použitie ActiveSync 4.5 s Visual Studiom 2008 a emulátorom

Táto časť nám poskytuje stručný návod ako synchronizovať emulátor s počítačom. Ide o dôležitú časť, pretože pri rôznom testovaní multimediálnej aplikácie bude treba do telefónu nahrávať súbory s multimediálnym obsahom na prehrávanie. Synchronizácia emulátoru s počítačom má tiež využitie pri testovaní aplikácií, ktoré využívajú pri komunikáciu TCP/IP alebo iné protokoly.

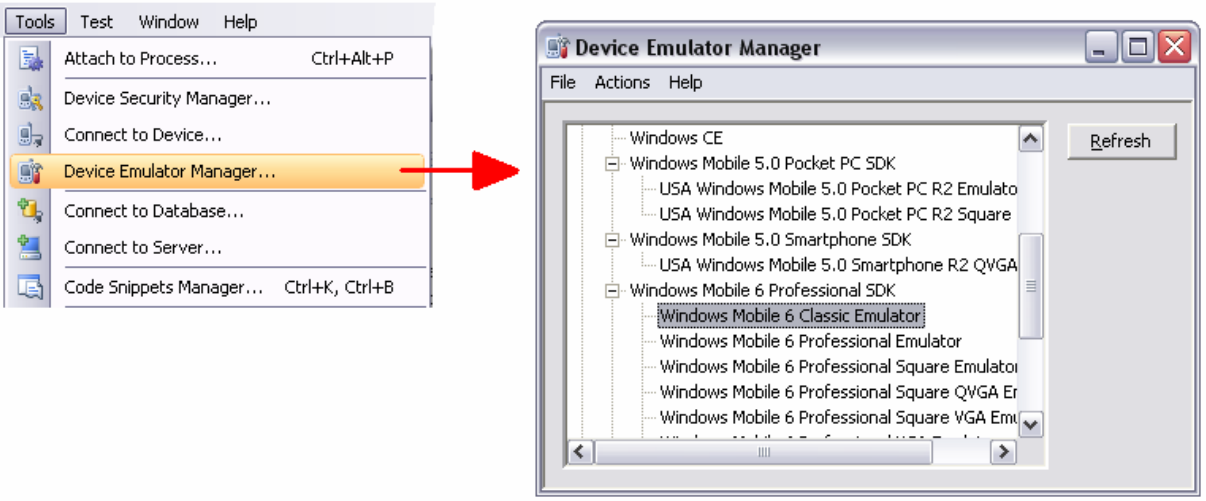
Ako prvé je dobré nainštalovať Visual Studio 2008, ktorého súčasťou už je emulátor Windows Mobile 5.0. Keďže aplikácia má bežať pod Windows Mobile 6.x, musí sa príslušný emulátor doinštalovať. Tento emulátor sa dá stiahnuť priamo na webových stránkach Microsoftu alebo ho nájdeme na priloženom médiu na konci práce. Ako posledný sa nainštaluje ActiveSync 4.5 (priložený taktiež na médiu) a tým je všetko pripravené k synchronizácií.

V prvom kroku sa otvorí program ActiveSync 4.5 a vyberie sa File→Connection Settings. Tu treba zatrhnúť položku „Allow connections to one of the following:“ a vybrať z ponuky „DMA“. Tiež keďže sa bude pracovať aj so sieťovou komunikáciou, zatrhne sa položka „Allow wireless connection on device when connected to the desktop“ (obr. 3.1).



Obr. 3.1. Nastavenie programu ActiveSync 4.5

V ďalšom kroku sa musí vybrať a nastaviť vhodný emulátor z ponuky vo Visual Studiu, pretože program ActiveSync 4.5 emulátor nevie inak rozpoznať. Vyberie sa Tools → Device Emulator Manager. V novootvorenom okne môžeme vidieť všetky nainštalované emulátory (obr. 3.2).

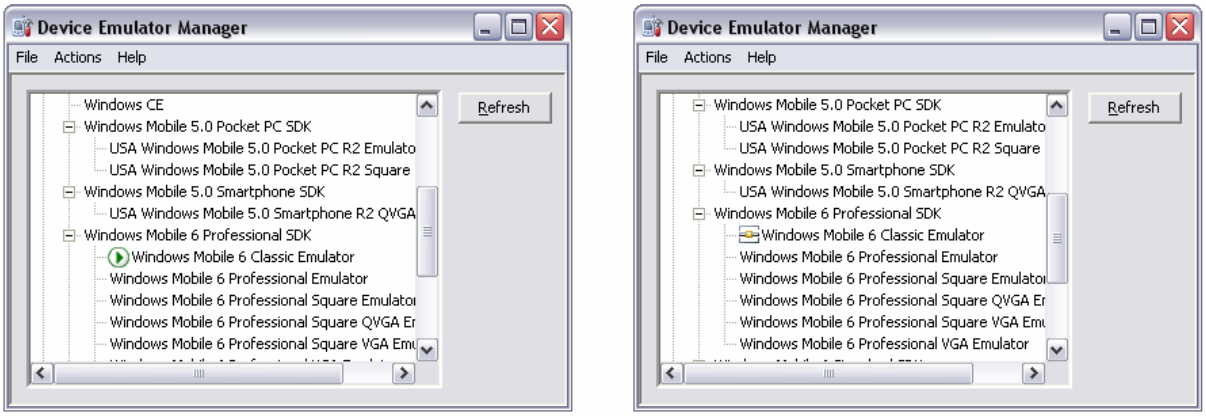


Obr. 3.2. Výber vhodného emulátora z ponuky vo Visual Studiu 2008

Po vybratí vhodného emulátora (Windows Mobile 6 Classic Emulator), sa po stlačení pravým tlačidlom myši vyberie z ponuky „Connect“ . Tým sa otvorí príslušný emulátor, ktorý sa chová ako presná kópia zariadenia s operačným systémom Microsoft Windows Mobile 6.0. Na obr. 3.3a môžeme vidieť, že pripojenie na emulátor je indikované malou zelenou ikonkou tvaru trojuholníka vedľa emulátoru.

Priamo v emulátore vyberieme program ActiveSync, ktorý sa nachádza medzi ostatnými programami. Tu sa nastaví z ponuky podobne ako to bolo v ActiveSync 4.5

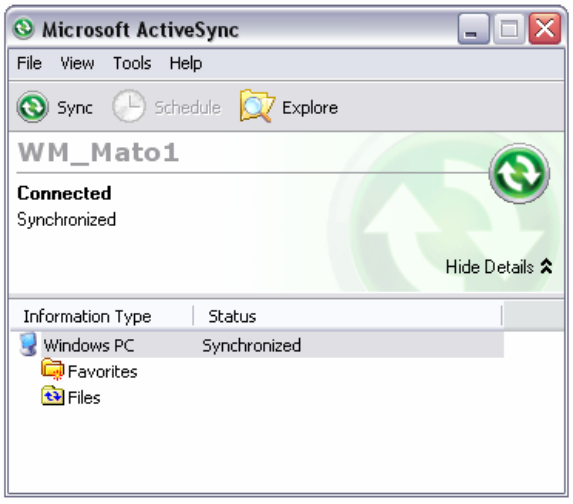
pripojenie pomocou DMA. Teraz je všetko pripravené k synchronizácií. Stačí už len v Device Emulator Manager prísť opäť na vybraný emulátor a po stlačení pravým tlačidlom myši vybrať z ponuky „Cradle“. Týmto nastavením si ActiveSync 4.5 emulátor rozpozná a spustí sa synchronizácia. Tiež sa zmení ikonka z tvaru trojuholníka na tvar pripojeného káblu (obr. 3.3b).



a) b)

Obr. 3.3. Naznačenie pripojenia emulátora (obrázok a) a naznačenie spojenia s ActiveSync (obrázok b)

Pred samotným spustením synchronizácie si môže užívateľ z ponuky vybrať, ktoré položky chce synchronizovať s počítačom. V ponuke môžeme nájsť položky ako kontakty, e-mail, kalendár, súbory, média a poznámky. Po vybratí príslušných položiek sa spustí synchronizácia. Tá trvá v závislosti od množstva dát, ktoré sa synchronizujú. Na obr. 3.4 je vidieť stav po úspešne vykonanej synchronizácií.



Obr. 3.4. Ukážka ActiveSync 4.5 po úspešnom ukončení synchronizácie

3.3 Podporované formáty v aplikácii

3.3.1 Formát WAV

WAV (alebo tiež WAVE) je skratka a bežne používaná prípona pre Waveform audio formát. Tento zvukový formát vytvorila firma IBM a Microsoft pre ukladanie zvuku na PC. Ide o formát nekomprimovaný, ktorý ukladá jednotlivé vzorky bez ďalších zvláštnych úprav pomocou pulzne kódovej modulácie PCM. Rovnakým spôsobom je uložený aj zvuk na Audio CD. Vzoriek, ktoré sa vytvoria pri PCM je veľmi veľa (CD má napr. vzorkovaciu frekvenciu 44,1kHz, teda 44 100 vzorkou za sekundu). Z natavení, ktoré sa dajú pri WAV formáte ovplyvňovať, sú to napr. typ záznamu, teda rozhodujeme či ide o MONO alebo STEREO signál, ďalej či sú vzorky ukladané ako 8, 16, 32 ... bitové. Čím viacej bitov, tým viacej informácií so sebou údaj nesie. Výhodou takéhoto formátu je, že sa s ním dobre manipuluje (úprava hlasitosti, redukcia šumu).

3.3.2 Formát MP3

Formát MP3 je formát stratovej kompresie zvukových súborov, založených na kompresnom algoritme MPEG (Motion Picture Experts Group). Rovnako ako pri formáte WAV je jeho veľkosť závislá na vzorkovacej frekvencií a rozsahu sledovaných hodnôt. Okrem toho je ale zvukový záznam ochudobnený o zvukové rozsahy a vlastnosti, ktoré ľudské ucho aj tak nepočuje, prípadne nie sú pre počutie dôležité. Kvalita záznamu sa potom určuje kompresným pomerom. Veľká kompresia = malý súbor a nízka kvalita zvuku, malá kompresia naopak. Ďalšie faktory, ktoré ovplyvňujú kompresiu sú:

- použitý psychoakustický model, ktorý (zjednodušene povedané) určuje čo je a čo nie je dôležité pre naše ucho.
- komprimačný algoritmus

Za najkvalitnejší kodek sa momentálne považuje kodek LAME, ktorý je vyvíjaný ako „open source“.

3.3.3 Formát WMA

Ide o formát, ktorý vyvíja spoločnosť Microsoft. WMA má podobné vlastnosti ako MP3, teda ide tiež o stratovú kompresiu zvukových súborov. Tento formát ponúka dvojnásobné množstvo hudby ako MP3 bez zníženia kvality. Tiež Windows Media je vraj najkvalitnejší, najbezpečnejší a najkomplexnejší spomedzi digitálnych formátov určených pre osobné počítače. Formát WMA obsahuje tiež podporu pre streamovanie hudby.

4 ZÁVER

Zadaním práce bolo zoznámiť sa s možnosťami platformy .NET Compact Framework a návrhom aplikácie pre mobilné zariadenia v jazyku C#. Po preštudovaní príslušnej literatúry bolo treba navrhnúť vhodnú realizáciu aplikácie do PDA, ktorá by bežala pod operačným systémom Windows Mobile 6.X a ktorá by dokázala prehrať multimediálny obsah.

Pre vývoj aplikácie bolo ideálne použiť prostredie Visual Studia 2008. Do prostredia stačilo len doinštalovať najnovšie SDK priamo pre vývoj pod operačným systémom Windows Mobile 6.0. SDK obsahuje aj PDA emulátor, v ktorom sa naprogramovaný program chová presne tak, ako na zariadení. Po importovaní príslušných knižníc sa podarilo nájsť vhodnú realizáciu aplikácie. Tá je schopná prehrávať audio súbory typu mp3, wav a wma. V aplikácii je navrhnutý playlist, ktorý funguje ako obojsmerný lineárny zoznam. Je to veľká výhoda, pretože nejde o statický prvok, ktorý by mal zadaný presný počet, ale naopak o prvok dynamický. Teda jeho veľkosť sa zmenšuje alebo zväčšuje podľa toho, koľko je v playliste skladieb. Rozdelenie tried a jednotlivých metód je popísaný v kapitole 2.3.

Začiatok práce je venovaný hlavne teoretickému úvodu o zariadeniach PDA, o .NET Frameworku a jeho verziách, ale tiež je tu zoznámenie sa s vývojovým prostredím Visual Studia 2008. Druhá kapitola sa venuje už konkrétnej multimediálnej aplikácií, ktorú sa podarilo zrealizovať. Je tu kompletný popis metód a atribútov jednotlivých tried, ale tiež je tu rozobratá komunikácia medzi klientom a serverom, ktorá sa využíva na príjem multimediálnych dát. Okrem samotného prenosu súborov a následného prehrávania, aplikácia umožňuje aj príjem v reálnom čase. V kapitole 2.5 sa nachádza podrobný popis ovládania programov z hľadiska užívateľa. Je tu tiež podkapitola venovaná výnimkám aplikácií, ktoré môžu nastať pri rôznych situáciách pri prenose alebo aj pri základnom používaní. Vyzdvihujú sa tu prednosti pri ich zachytávaní a ošetrení. Posledná časť práce sa venuje hlavne synchronizácií emulátoru s počítačom, ale tiež sú tu opísané jednotlivé podporované formáty, ktoré je aplikácia schopná prehrávať.

Výsledkom práce je teda kompletné riešenie a návrh aplikácie, ktorá dokáže prijať a pracovať s multimediálnym obsahom.

LITERATÚRA

[1] LACKO, Ľuboslav. *Vývoj aplikácií pre mobilné zariadenia.* Praha : Microsoft, s.r.o, [2003?]. 1, s. 3-20.

[2] WINGLEY, Andy, MOTH, Daniel, FOOT, Peter. *Microsoft mobile development handbook.* Redmond, Washington : Microsoft Press, 2007. 1, s. 3-35.

[3] PC.server.sk [online]. *TS-PCServer*, 2006 [cit. 2008-11-05]. Dostupný z WWW: <http://pc.server.sk/aktuality/detail/13415-windows-mobile-6/>.

[4] Linked list [online]. *Wikipedia*, 2003 [cit. 2008-12-05]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Linked_lists>.

[5] BURGET, Radim. *Abstraktní dátové typy 1*. Přednášky MTIN [online]. 2007 [cit. 2008-11-15].

[6] *Getting Started with Visual Studio .NET and the Microsoft .NET Compact Framework* [online]. MSDN, 2003 [cit. 2008-10-25]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/aa446534.aspx#netcfgetstarted_topic25>.

[7] PIRKL, Josef. *Řešené příklady v C# : C# skutečně prakticky*. České Budějovice : Kopp, 2005. 34, s. 227-229.

[8] HANSLIAN, Martin, KOCANOVÁ, Pavla. KOCAN, Marek. *Microsoft Windows 2000 Server Sítě TCP/IP*. Praha: Computer Press, 2000. s.15-16. ISBN 80-7226-291-2.

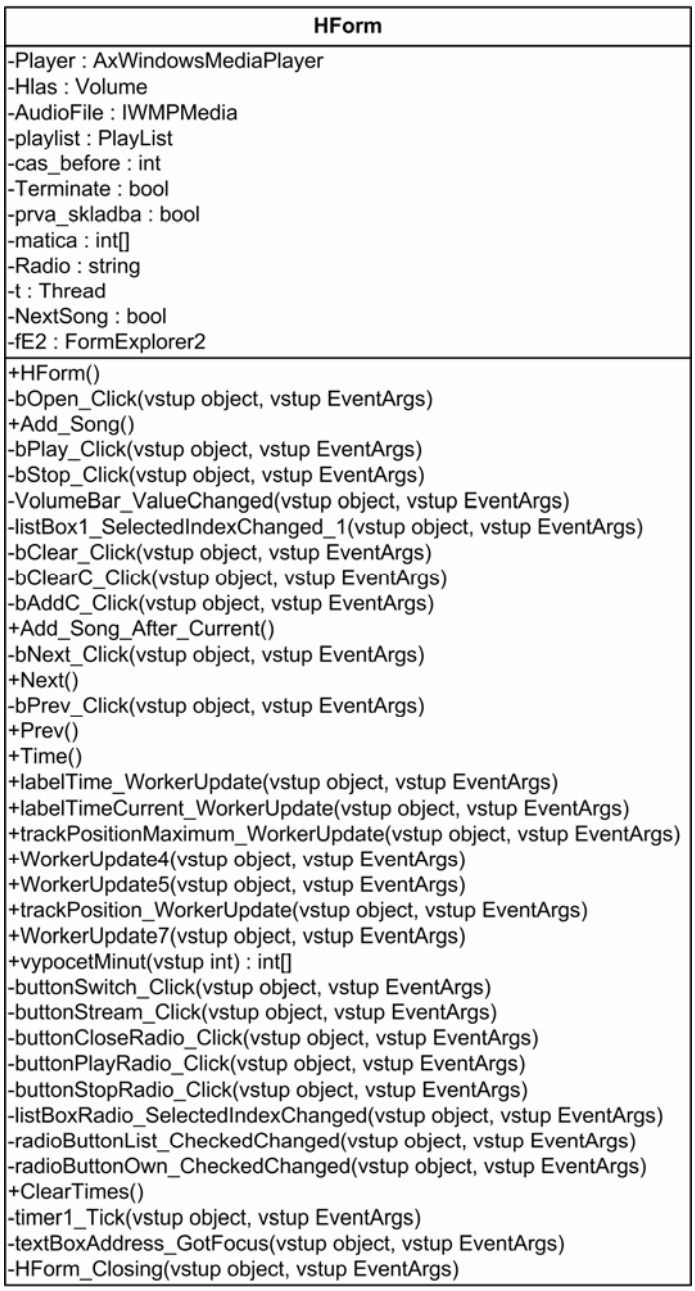
ZOZNAM SKRATIEK

Avi	AVI (Audio Video Interleave) je video formát vyvinutý primárne pre platformu Windows.
GUI	Graphical User Interface - je skratka pre grafické používateľské rozhranie.
IDE	Integrated Development Environment - integrované vývojové prostredie pre programovacie jazyky.
Jpg	Značenie súborov stratovej kompresie používanej pre ukladanie počítačových obrázkov vo fotorealistickej kvalite.
Mp3	Formát stratovej kompresie zvukových súborov, založený na kompresnom algoritme MPEG (Motion Picture Experts Group).
Palm	PDA, ktoré majú operačný systém PALM OS. V podstate sa jedná o konkurenciu Pocket PC.
PDA	Z anglickej skratky Personal Digital Assistant alebo osobný digitálny asistent.
Plugin	Je software, ktorý nepracuje samostatne, ale ako doplnkový modul inej aplikácie a rozširuje tak jej funkčnosť.
Pocket PC	Vreckový osobný počítač. Pojmom Pocket PC sa označuje v prvom rade operačný systém vyvinutý firmou Microsoft.
TCP	Transmission Control Protocol - Je spoľahlivý, spojovo orientovaný komunikačný protokol transportnej vrstvy.
VoIP	Voice over Internet Protocol - Internetová telefónia, je prenos komunikácie uskutočňovanej ľudským hlasom cez Internet.
Wav	Waveform audio formát. Ide o nekomprimovaný formát audia.

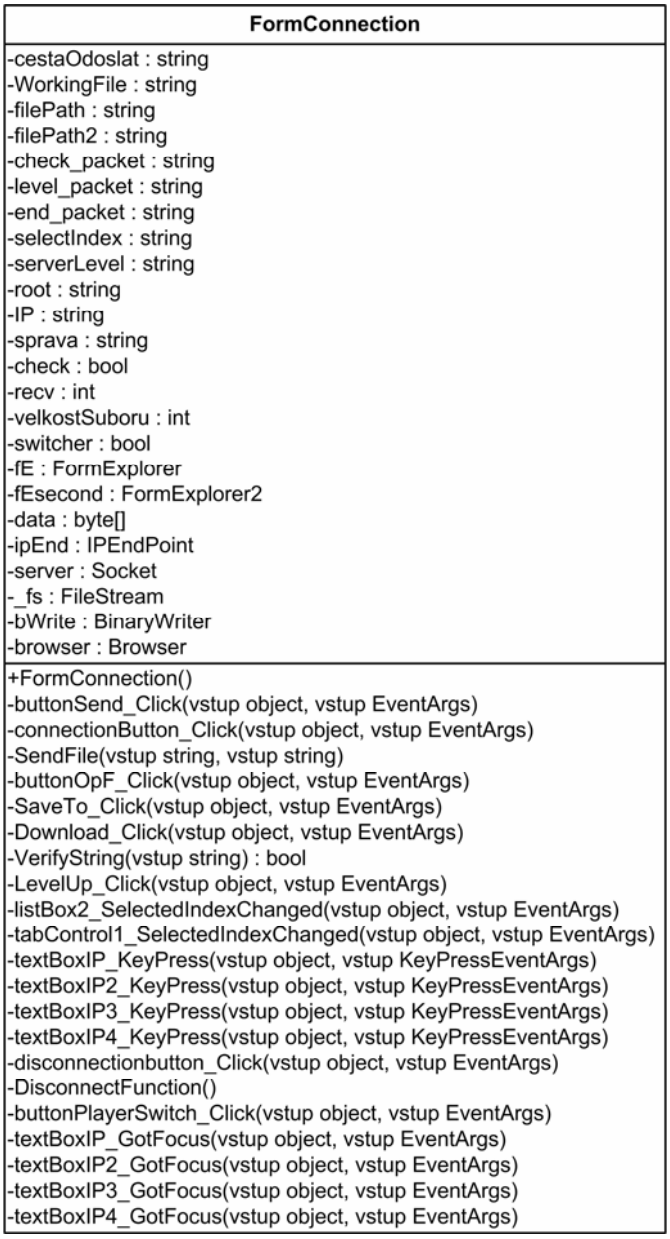
PRÍLOHY

A PODROBNÝ POPIS TRIED Z HĽADISKA UML

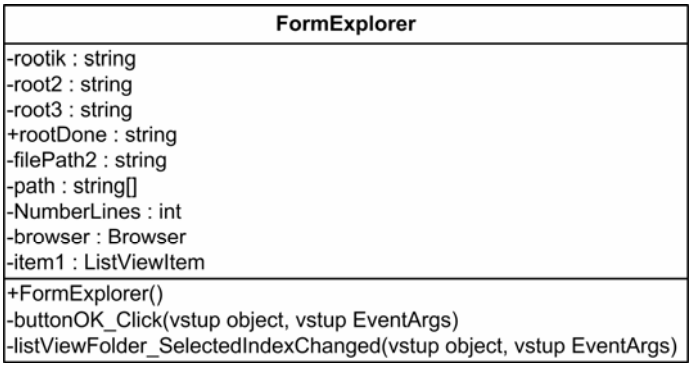
A.1 Trieda HForm



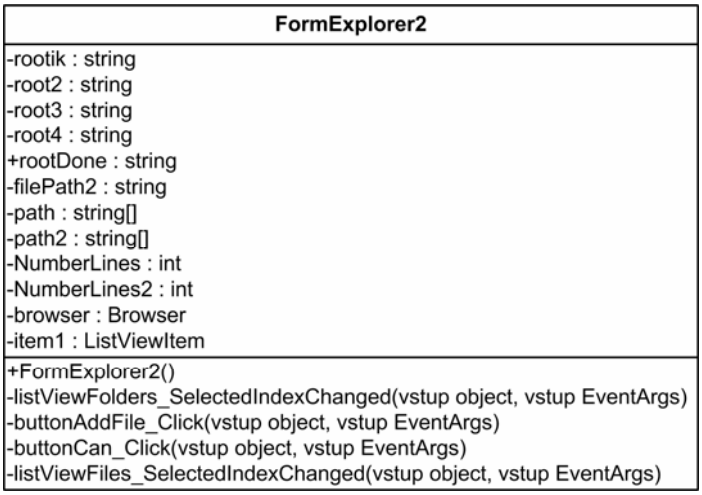
A.2 Trieda FormConnection



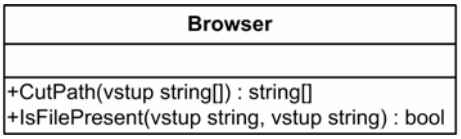
A.3 Trieda FormExplorer



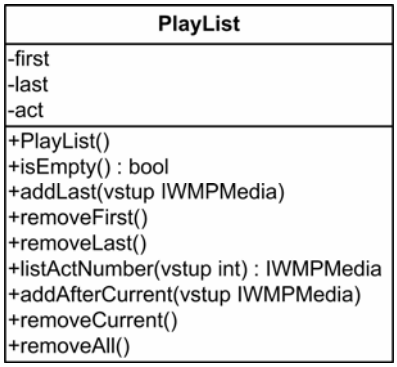
A.4 Trieda FormExplorer2



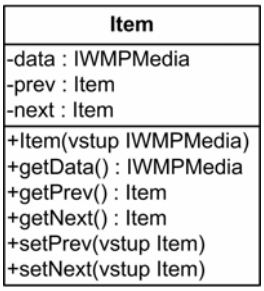
A.5 Trieda Browser



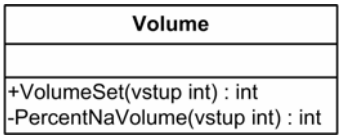
A.6 Trieda PlayList



A.7 Trieda Item



A.8 Trieda Volume



B OBSAH PRILOŽENÉHO CD

`Aplikácie`

- `Prehravač –` vlastná aplikácia pre mobilné zariadenie
- `Server –` vlastná aplikácia serveru pre stolový počítač

`Dokumenty`

- `Diplomová práca` - vlastná diplomová práca vo formáte pdf

`Ostatné programy`

- `Acrobat Reader 7.0` – program na prehliadanie pdf súborov
- `ActiveSynch 4.5` – program pre synchronizáciu emulátora s počítačom
- `Windows Mobile 6 Professional SDK` – obsahuje SDK a emulátor Windows Mobile 6.0