VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

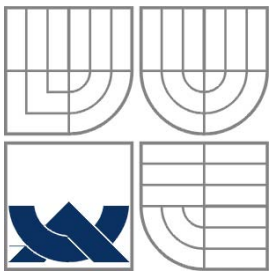
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

IS PRO EVIDENCI TELEFONNÍCH HOVORŮ

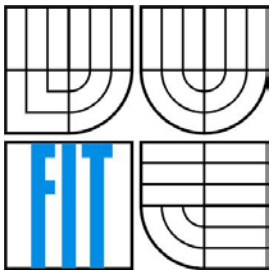
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. PETR ŠITKA
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

IS PRO EVIDENCI TELEFONNÍCH HOVORŮ

IS FOR PHONE CALLS ACCOUNTING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PETR ŠITKA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. FRANTIŠEK ŠČUGLÍK

BRNO 2009

Abstrakt

Tato práce se zabývá vývojem informačního systému spolupracujícího s mobilními zařízeními. Informační systém je zaměřen na shromažďování a vyhodnocování údajů o provedených hovorech na mobilních zařízeních. Jsou zde uvedeny možnosti spojení osobního počítače a mobilního zařízení, dále jsou popsány nejpoužívanější operační systémy pro mobilní zařízení a nástroje pro vývoj aplikací. Na základě této studie je navržen a implementován informační systém.

Abstract

This contribution refers to development of information system working with mobile communication devices. The information system is mainly for collecting and analyzing data made through the mobile devices. In the contribution there are described possibilities of connection between PC and mobile device, also there are described the most common information system for mobile devices, and tools for application development. Design and implementation of information system is based on this study.

Klíčová slova

Informační systém, mobilní zařízení, datový kabel, WiFi, Bluetooth, IrDA, Windows Mobile, Symbian, .NET, LINQ.

Keywords

Information system, mobile device, data cable, WiFi, Bluetooth, IrDA, Windows Mobile, Symbian, .NET, LINQ.

Citace

Šitka Petr: IS pro evidenci telefonních hovorů, diplomová práce, Brno, FIT VUT v Brně, 2009

IS pro evidenci telefonních hovorů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Františka Ščuglíka.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Petr Šitka
20.5.2009

Poděkování

Děkuji Ing. Františku Ščuglíkovi za čas věnovaný konzultacím, užitečné připomínky a rady.

Obsah

1 Úvod

Mobilním zařízením jako je GSM telefon, Smartphone nebo PDA disponuje v dnešní době takřka každý člověk. Potřeba vzájemné komunikace je stále větší, obzvláště v pracovní sféře je pak důležité být ve spojení s okolním světem. Mobilní operátoři urputně bojují o každého zákazníka a snaží se nabízet co nejvýhodnější služby. Operátoři neustále uvádějí na trh různé balíčky a zvýhodnění zaměřené na určité cílové skupiny. Nesprávná volba mobilního operátora a služeb jím poskytovaných může dnes znamenat nemalé finanční výdaje, zejména když se účastnické smlouvy podepisují na roky dopředu. Účelem této práce je navrhnout a implementovat systém, který by sloužil k evidenci telefonních hovorů. Na základě sběru dat o uskutečněných hovorech z mobilních zařízení je možno vytvářet statistiky vytížení jednotlivých přístrojů. Využití systému by pak mohlo vést k úspoře prostředků. Uživatel na základě získaných informací může posoudit, zda tarif, který právě využívá je pro něj optimální. Mohou se pak zavádět opatření jako je změna tarifu nebo mobilního operátora. Práce je implementačního charakteru a za cíl si klade realizovat výše zmíněný systém.

Zpráva je rozdělena do několika částí. V první části je zaveden pojem mobilní zařízení a prozkoumávají se zde možnosti propojení mobilního zařízení s osobním počítačem. Popsány jsou obecné principy přenosu dat pro jednotlivé druhy spojení. Uveden je způsob využití dané technologie v mobilních zařízeních. Pojednáno je také o operačních systémech pracujících na mobilních zařízeních, jsou uvedeny základní vlastnosti, přehled verzí a nejčastěji používané vývojové prostředí pro tvorbu aplikací na daném operačním systému. Nastíněny jsou základní principy technologií využívaných v implementační části práce.

Druhá část technické zprávy se zabývá analýzou a návrhem řešení. Využívá poznatků z první části zprávy a na jejich základě je zvolena cílová platforma a vhodné vývojové prostředí. Navržen je způsob získávání a ukládání informací o provedených hovorech na mobilním zařízení. Je vytvořen komunikační protokol pro spojení a přenos dat mezi mobilním zařízením a osobním počítačem. Nadefinován je způsob ukládání dat a struktura databáze.

Ve třetí části je popsána vlastní aplikace. Zaměřujeme se především na vzhled části pracující na osobním počítači. Shrnuty jsou veškeré vlastnosti aplikace a je uvedeno další možné řešení.

2 Mobilní zařízení

Mobilním zařízením nejčastěji rozumíme mobilní telefon, smartphone, PDA, či notebook. Odvětví mobilních zařízení se neustále vyvíjí. Postupem času docházelo k zmenšování osobních počítačů, až na podobu dnešních přenositelných notebooků. Běžně používané digitální asistenti, označovaní jako PDA, již dosahují výkonnosti blížící se osobnímu počítači. S příchodem operačních systémů do mobilních telefonů je již velmi těžké rozlišit, zda je zařízení smartphone (chytrý telefon), PDA, nebo komunikátor. Pro účely této práce budeme ve zbytku tohoto textu mobilní zařízení chápat jako přístroj řízený operačním systémem, komunikující v GSM sítích.

2.1 Komunikace mezi mobilním zařízením a PC

Způsoby komunikace mobilních zařízení s osobním počítačem se neustále rozšiřují, zejména v posledních letech došlo v tomto směru k velkému vývoji. Postupem času nahrazují kabelové spojení po sériových linkách, jako jsou RS232 a USB bezdrátové technologie. Mezi bezdrátové spojení patří IrDa rozhraní a radiové Bluetooth, WiFi a Zigbee. Zbytek kapitoly popisuje základní principy jednotlivých technologií a jejich využití v mobilních zařízeních.

2.1.1 Datový kabel

Nejzákladnějším způsobem komunikace mobilního zařízení a PC je pomocí metalického spojení.
Od tohoto způsobu přenosu dat se přechází k uživatelsky přívětivější bezdrátové komunikaci, i přesto je rozhraním pro komunikaci po datovém kabelu vybavena drtivá většina zařízení. Používá se sériová komunikace na linkách RS232 a USB.

2.1.1.1 RS232

RS232 je standardem definovaným již v 60. letech Electronic Industries Association. Jedná se o sériový asynchronní přenos informací mezi dvěma zařízeními [6].

Fyzickou vrstvu reprezentuje přenos dvou logických úrovní po vedení, log. 0 je přenášena napětím +10 V, log. 1 pak -10 V. Mobilní zařízení pracují většinou s napětím okolo 3 V, datový kabel proto obsahuje přidanou logiku, jenž konvertuje napětí na požadované hladiny. Přenosová rychlost úzce souvisí s délkou vedení, standard RS232 uvádí maximální délku 15 m, při snížení úrovně napětí na 3 V se pro zachování přenosové rychlostí zkracuje i délka kabelu. Nejvyšší přenosová rychlost, které lze dosáhnout, je však 115 kb/s. Konektor na straně zařízení je specifický pro konkrétní model, na straně PC je kabel osazen nejčastěji konektorem Cannon 9 female (viz. Obr. 2.1). Pro základní komunikaci slouží zapojení následujících vodičů:

- Rxd (Recive Data) – tok dat příjem
- Txd (Transmit Data) – tok dat vysílání
- GND (System Ground) – referenční zemnění pro ostatní signály

Zapojení datových kabelů se liší v závislosti na typu zařízení a výrobci.

Data jsou zpracovávaná po slabikách, slabika je rozdělena na jednotlivé bity a sériově přenesena. Přenosový rámec tvoří synchronizační start bit, následují datové bity, které mohou být doplněny paritním bitem, rámec ukončuje stop bit. Pro správný průběh synchronizace musí být na obou stranách nastaveny totožné hodnoty rychlosti, délky slova a parity.

Obr. 2.1: Příklad datového kabelu s rozhraním RS232 od společnosti NOKIA

Výhody použití RS232:

- Vysoká bezpečnost

Nevýhody použití RS232:

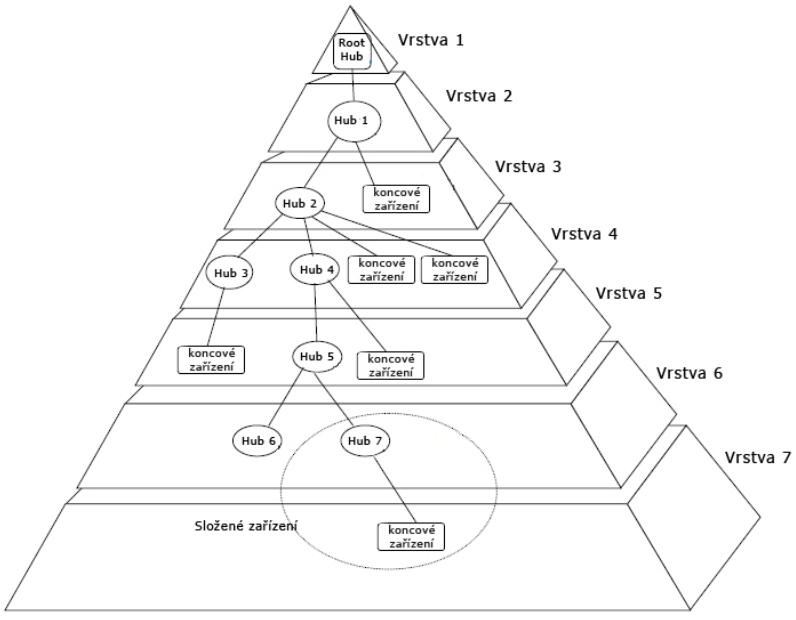
- Nízká přenosová rychlost – 115 kb/s
- Nutnost nastavení parametrů přenosu
- Nutnost vlastnit přenosové médium (kabel)

RS232 je v oblasti mobilních zařízení již zcela nahrazena výkonnějším rozhraním USB.

2.1.1.2 USB

Universal Serial Buss (Univerzální sériová sběrnice) byla původně vyvinuta společnostmi Compaq, Digital, IBM, Intel, Microsoft, NEC a Northern. Dnes skupina vyvíjející USB standard vystupuje pod názvem USB Implementers Forum. První standard USB 1.0 byl vydán v listopadu 1995, v roce 2000 vyšel standard USB 2.0 a v současné době (listopad 2008) vychází norma pro USB 3.0.

Topologie systému USB je vrstvená hvězdicová. První vrstvu topologie tvoří kořenový rozbočovač – root hub s host controllerem, středem každé hvězdice je USB hub, na který je v další úrovni připojen buďto další hub nebo koncové zařízení. Průchodem signálu huby a vedením vzniká zpoždění, proto norma povoluje nejvýše sedm vrstev. Počet adresovaných zařízení je limitován na 127. Topologie USB sběrnice je znázorněna na obrázku 2.2.



Obr. 2.2: Topologie USB sběrnice (převzato a upraveno z [7])

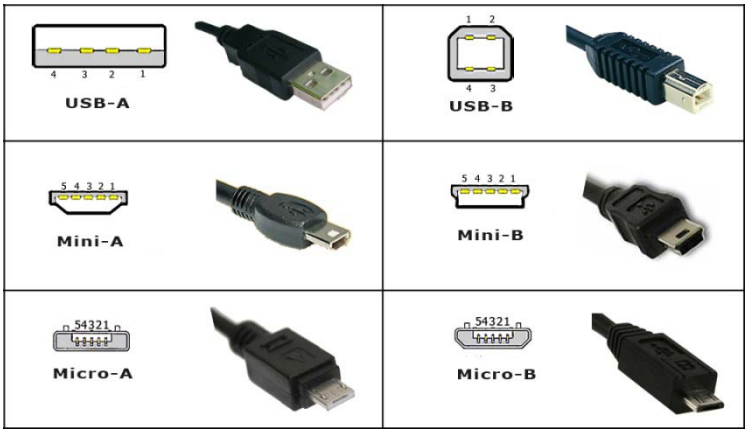
USB sběrnice pracuje ve třech režimech v závislosti na rychlosti toku dat, jednotlivé režimy a jejich vlastnosti jsou popsány v tabulce 2.1.

Režim	Využití	Vlastnosti
LOW-SPEED • 10 – 110 kb/s	klávesnice, myš stylus hrací zařízení zařízení pro virtuální realitu	nejnižší cena jednoduché použití dynamické připojování/odpojování více zařízení
FULL-SPEED • 500 kb/s – 10 Mb/s	Modemy audio zařízení	nižší cena jednoduché použití dynamické připojování/odpojování více zařízení garantovaná šířka pásma garantovaná doba čekání
HIGH-SPEED • 25 – 400 Mb/s	video zařízení přenos dat z paměti	nízká cena jednoduché použití dynamické připojování/odpojování více zařízení garantovaná šířka pásma garantovaná doba čekání vysoká šířka pásma

Tab. 2.1: Vlastnosti a využití režimů USB sběrnice (převzato z [7])

Huby high speed jsou kompatibilní s low i full speed zařízeními. Zařízení high speed je možno připojit k low nebo full speed hubu, potom je rychlost provozu omezena rychlostí použitého hubu.

Standardní USB kabel obsahuje čtyři vodiče, dva jsou určeny pro napájecí napětí a zemnění, další dva pak pro přenos dat. Osazen je na straně kontroléru (PC) konektorem USB-A, na straně zařízení pak konektorem USB-B. Postupně se konektory na straně zařízení zmenšovaly a tak vyšel standard pro Mini a Micro konektory. Kabel s Mini a Micro konektory je rozšířen o další vodič ID. Jednotlivé druhy konektorů jsou znázorněny na obrázku 2.3. Maximální délka kabelu je u standardu USB 1.0 stanovena na 3m, pro standard USB 2.0 pak 5m.



Obr. 2.3: Standardizované konektory USB kabelů

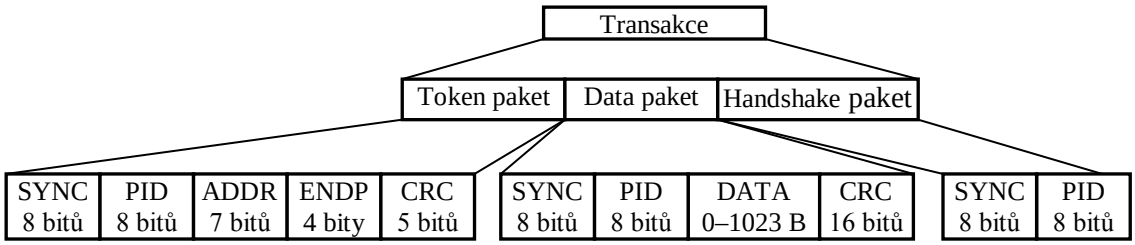
USB je řízená sběrnice, kde veškeré datové přenosy inicializuje host controller, data jsou přenášena pomocí paketů. Sběrnicové transakce se sestávají ze tří paketů. První paket (token paket) obsahuje typ transakce, číslo USB zařízení a adresu koncového bodu. Token paket určuje, zda se jedná o komunikaci host controller – zařízení (typ OUT), nebo zařízení – host controller (typ IN).

Pakety jsou šířeny do všech uzlů sběrnice, tudíž zařízení musí umět podle adresy rozpoznat, že je komunikace určena právě pro něj. Zařízení, které rozpozná svoji adresu, se připraví k přenosu a vyšle datový paket, nebo oznámí, že nemá žádná data pro komunikaci. Ukončení transakce provádí příjemce dat vysláním tzv. handshake paketu, kterým potvrdí úspěšnost přenosu.

USB protokol zahrnuje celkem 16 typů paketů, které se dělí do následujících skupin:

- pověřovací (token) - IN, OUT, SETUP, SOF
- datové (data) - DATA0, DATA1, DATA2, MDATA
- potvrzovací (handshake) - ACK, NAK, STALL, NYET
- speciální (special) - ERR, SPLIT, PING

Obrázek 2.4 ilustruje strukturu základních paketů transakce na sběrnici USB, podrobnější popis ostatních typů paketů viz [7].



Obr. 2.4: Struktura paketů transakce na USB sběrnici

Formát paketu je závislý na jeho typu. Společné pro všechny pakety jsou úvodní synchronizace (SYNC) a identifikace paketu (PID). Jak již bylo uvedeno výše token paket obsahuje adresu zařízení, s nímž se bude komunikovat (ADDR) a adresu koncového bodu (ENDP), zakončen je 5 bitovým kontrolním součtem (CRC). Datový paket rovněž obsahuje synchronizaci a typ paketu, následují samotná data. Velikost datové části paketu (DATA) se liší v závislosti na koncovém zařízení. Jedná-li se o low-speed zařízení obsahuje datová část paketu nejvýše 8 Bytů, pro full-speed zařízení je velikost 1023 Bytů a pro high-speed 1024 Bytů. Paket je zakončen 16 bitovým kontrolním součtem (CRC). Handshake paket obsahuje pouze synchronizaci (SYNC) a identifikátor paketu (PID).

Specifikace USB obsahuje čtyři základní typy datových přenosů [8]:

Řídící (control) přenosy jsou používány ke konfiguraci jednotlivých zařízení. Řídící přenosy jsou plánovány do rámců tak, aby se periodicky detekovaly nově připojené zařízení.

Hromadné (bulk) přenosy obsahují velká množství dat, např. data pro tiskárny nebo získaná ze scannerů. Hromadná data jsou přenášena sekvenčně a spolehlivost jejich přenosu je zajišťována detekcí chyb na hardwarové úrovni a omezeným počtem opakovaných pokusů. Šířka pásma, využitá hromadným přenosem, může být různá a záleží na ostatním provozu na sběrnici.

Přerušovací (interrupt) přenosy přenášejí data do nebo ze zařízení. Zařízení může požádat o přenos těchto dat v kterýkoliv okamžik a tato data jsou doručena USB sběrnicí v nejkratším možném čase. Nejčastěji se jedná o upozornění na nějakou událost, například na stisk klávesy na klávesnici nebo změnu pozice myši, tedy taková data, která zaberou jen několik bytů.

Izochronní (isochronous) přenosy jsou trvalé přenosy, u nichž probíhá vytváření, přenos a zpracování dat v reálném čase. Přesné časování přenosu je zajištěno rovnoměrným rozložením úseků, ve kterých jsou izochronní data přijímána a odesílána, v čase. Aby bylo dodrženo časování,

musí být izochronní data předávána hubem se stejnou frekvencí, s jakou jsou přijímána. Tato data mohou být také citlivá na zpoždění při přenosu. Izochronní roury mívají šířku přenosového pásma odvozenou nejčastěji od vzorkovací frekvence daného zařízení. Požadavky na zpoždění jsou dány velikostí vyrovnávací paměti daného koncového bodu.

Nově přidaná specifikace do standardu USB je OTG (USB On-The-Go). OTG do klasického formátu USB komunikace typu Master - Slave (Host-Device) přináší možnost přímého přenosu dat typu bod-bod (point-to-point). Tento standard umožňuje komunikaci dvou periferních zařízení bez nutnosti přítomnosti nadřazeného systému (PC) [9]. Způsob komunikace je zachován, zařízení pracují stále v režimech Host/Device, je však rozšířen o DRD (Dual-Role Device), což je schopnost provozovat zařízení jak v režimu Host, tak i v režimu Device. Při komunikaci si zařízení po dohodě mohou režimy přepínat, jedno ze zařízení vystupuje tedy jako Host a druhé jako Device, po domluvě se první zařízení přepne do režimu Device a druhé do Host. Během inicializace zařízení, tj. po zapnutí napájení, však musí být přesně dáno, které ze zařízení je Host a které Device. K odlišení zařízení při inicializací slouží zavedení nového kabelu s konektory Mini a Micro (viz výše.). Zařízení, k němuž se připojí kabel konektorem Mini-A (Micro-A) se po inicializaci chová jako Host, zařízení s konektorem Mini-B (Micro-B) je v režimu Device. Obě zařízení, která chtějí provozovat komunikaci bod-bod musejí podporovat OTG.

Pro komunikaci mobilních zařízení po vedení je USB v dnešní době nejrozšířenějším standardem. Jednotliví výrobci používají buďto konektory specifické pro daný model zařízení, podobně jako u RS232, nejčastěji se však setkáme s konektory typu Mini a Micro. Skupina výrobců mobilních zařízení jako Sony Ericsson, Samsung, Motorola, LG uvažují o jednotném univerzálním USB konektoru s názvem USB-IF. S nástupem standardu USB 3.0 dojde k vývoji nových konektorů pro mobilní zařízení, specifikace hovoří o rozšíření kabelu o dalších pět vodičů.

Výhody použití USB:

- Vysoká bezpečnost
- Vysoká přenosová rychlost
- Rozšířená specifikace

Nevýhody použití USB:

- Nutnost vlastnit přenosové médium (kabel)

2.1.2 IrDA

IrDA je standard vytvořený IrDA konsorciem (Infrared Data Association), který definuje jak bezdrátově přenášet digitální data pomocí infračerveného zařízení. IrDA ve svých specifikacích definuje standardy jak fyzických koncových zařízení, tak protokolů jimiž komunikují IrDA zařízení [10]. Standard byl představen roku 1993 firmami HP, IBM a Sharp.

IrDA zařízení komunikují pomocí modulovaného infračerveného světla o vlnové délce 875 nm. Vysílacím prvkem jsou infračervené LED diody, příjem zajišťují fotodiody. Šíření infračerveného záření je obdobné jako u viditelného světla. Přímá viditelnost není díky schopnosti odrazů v uzavřeném prostoru obecně přísnou podmínkou pro infračervené sítě, ale IrDA ji vyžaduje [11]. IrDa zařízení podle normy IrDA 1.0 a 1.1 jsou schopny komunikovat do vzdálenosti 1 m (přenos je citlivý na okolní záření a může docházet k rušení), při bitové chybovosti BER = 10^{-9}. První verze IrDA 1.0 podporovala sériovou asynchronní komunikaci s přenosovou rychlostí 2,4 kbit/s až 115,2 kbit/s. Ve verzi 1.1 je maximální dosahovaná rychlost 4 Mbit/s. Formát přenosu dat je jako u klasického sériového přenosu. Jedná se o asynchronní přenos se start a stop bitem, doplněný o 16 bitový CRC

pro detekci chyb. Při vyšších přenosových rychlostech se nepřidává žádný start a stop bit a data jsou přenášena v synchronním formátu. Aby byla zajištěna synchronizace hodinových impulsů při přenosu většího bloku logických jedniček, používá se bit stuffing, to znamená vložení nulového bitu vždy po sekvenci určitého počtu jedničkových bitů. Při přenosové rychlosti 4Mbps není opět zapotřebí start a stop bitu a ani bit stuffing. Pro kontrolu chyb je použit 32bitový CRC.

Protokoly v rámci specifikace jsou následující [10, 11] (viz. Obr. 2.5):

IrLAP (Infrared Link Access Protocol) – slouží ke zjišťování zařízení v síti, řešení konfliktů adres, ke zjišťování, zda jiné zařízení je dostupné pro navázání spojení, pro navazování spojení a pro přenos dat.

IrLMP (Infrared Link Management Protocol) – detekuje přítomnost zařízení, která nabízejí nějakou službu, kontroluje tok dat a multiplexuje konfigurace pro účast více stanic s různými schopnostmi. Aplikace potom využívají vrstvy IrLMP k dotazům, zda je v dosahu požadované zařízení.

IrLM-IAS (Infrared Link Management – Information Access Service) – spravuje informace o vlastních službách a také umožňuje vzdálený přístup k informacím o službách partnerského zařízení.

Tiny-TP (Tiny Transport Protocol) – vrstva udržující virtuální kanál mezi zařízeními, sama opravuje chyby na lince, provádí rozdělení dat do paketu a jejich znovusestavení.

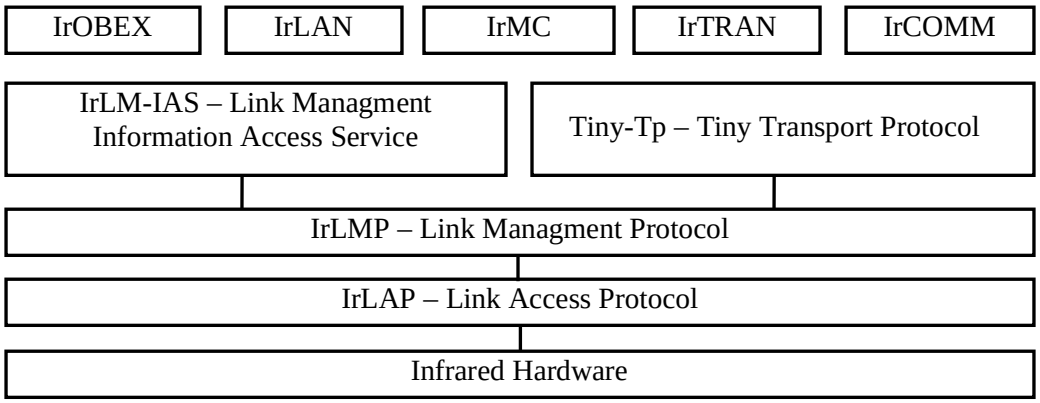
IrOBEX (Infrared Object Exchange Protocol) – je jednoduchý protokol s příkazy PUT a GET, který dovoluje přenášet binární data mezi stanicemi.

IrLAN (Infrared LAN Access Extensions for Link Management) – umožňuje připojení počítače k bezdrátové LAN prostřednictvím přístupového bodu (IrLAN adaptéru), nebo umožňuje dvoubodovou komunikaci mezi počítači, případně umožňuje připojení počítače k LAN prostřednictvím jiného již připojeného PC.

IrMC (Infrared Mobile Communications) – rozšíření IrOBEX pro mobilní zařízení, definuje jak přenášet informace vztahující se k GSM síti (adresáře kontaktů, sms zprávy, kalendář)

IrTRAN (Infrared Transfer Picture) – specifikace transportu obrazových dat z digitálních kamer a fotoaparátů

IrCOMM (Infrared Communications Protocol) – emulace sériového a paralelního portu.



Obr. 2.5: Protokolová architektura IrDA [2]

Na straně mobilního zařízení bývá IrDA modul vždy zabudován, u PC může být vestavěný (notebooky) nebo jako externí modul připojený přes USB rozhraní (stolní počítače).

Výhody použití IrDA:

- Vysoká bezpečnost – záření se nedostane přes stěny místnosti
- Dostupnost kmitočtového spektra
- Minimální spotřeba energie
- Poměrně vysoká přenosová rychlost – až 4 Mbit/s

Nevýhody použití IrDA:

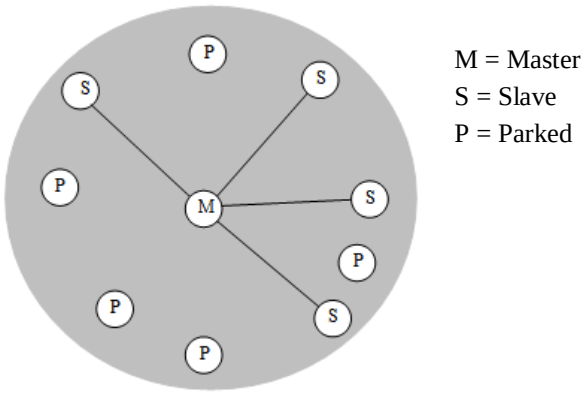
- Přenos na malou vzdálenost – jednotky metrů
- Nutnost fyzické viditelnosti zařízení
- Citlivost na okolní rušení – elektromagnetické, světelné rušení (sluneční svit, žárovka)

2.1.3 Bluetooth

Specifikace Bluetooth je charakteristická nízkými nároky na napájení a spoluprací s malými koncovými zařízeními. Rychlost na fyzické vrstvě dosahuje 1 Mbit/s, přičemž skutečná propustnost dat se pohybuje maximálně kolem 720 kbit/s [11].

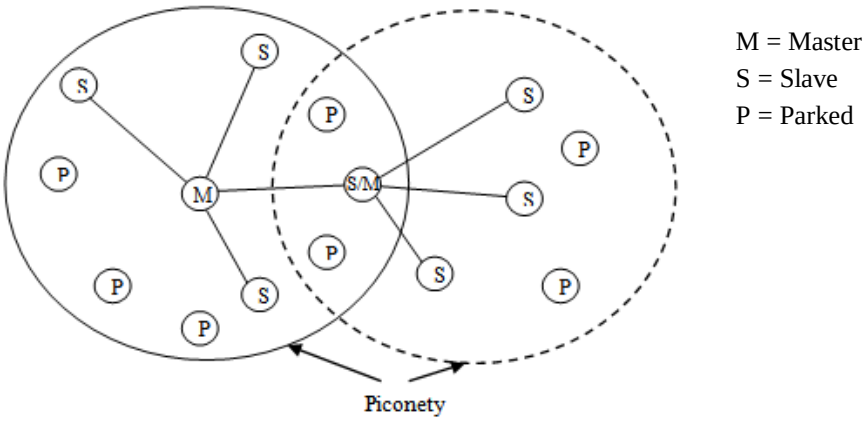
U zrodu technologie Bluetooth stála švédská společnost Ericsson, která v roce 1994 vyvíjela systém bezdrátového spojení mobilních zařízení s jejich příslušenstvím, tento projekt se jmenoval MC-Link (multi-mommunicator link). V roce 1998 se k Ericssonu přidávají společnosti Intel, IBM, Nokia a Toshiba, tato skupina vystupovala pod názvem Bluetooth SIG (Special Interest Group). Sdružení Bluetooth SIG má v současné době přes 2500 členů. Název Bluetooth pochází od přízviska dánského krále žijícího v desátém století Haralda Gromse. Tento král za své vlády sjednotil Dánsko a Norsko, díky jeho schopnosti sjednocovat je i technologie pojmenována podle něho. Gromsovi říkali podle jeho tmavé pleti Harald Blåtand, tudíž původní název neměl znamenat „modrý zub“ (blue tooth). Bluetooth patří do skupiny bezdrátových sítí s malým dosahem IEEE 802.15 WPAN (Wireless Personal Area Network).

Bluetooth pracuje v bezlicenčním pásmu 2,4 GHz. Využívá metody rozprostřeného spektra s přeskakováním mezi kmitočty FHSS (Freyuency Hopping Spread Spectrum). Radiový signál náhodně přeskakuje mezi 79 kanály o šířce 1 MHz, těchto hoppů je provedeno 1600 za sekundu. Komunikace v Bluetooth síti je řízena hlavní stanicí (master), která vyzývá podřízené stanice (slave) a následně od nich přijímá informaci o jejich stavu. Slave stanice komunikuje s ostatními stanicemi pouze prostřednictvím master stanice. Komunikace může být jednobodová i mnohobodová. Jsou-li stanice propojeny do ad-hoc, vzniká tzv. piconet, kde jedna stanice působí jako master a obsluhuje ostatní slave stanice. Znázornění piconetu je na obrázku 2.6.



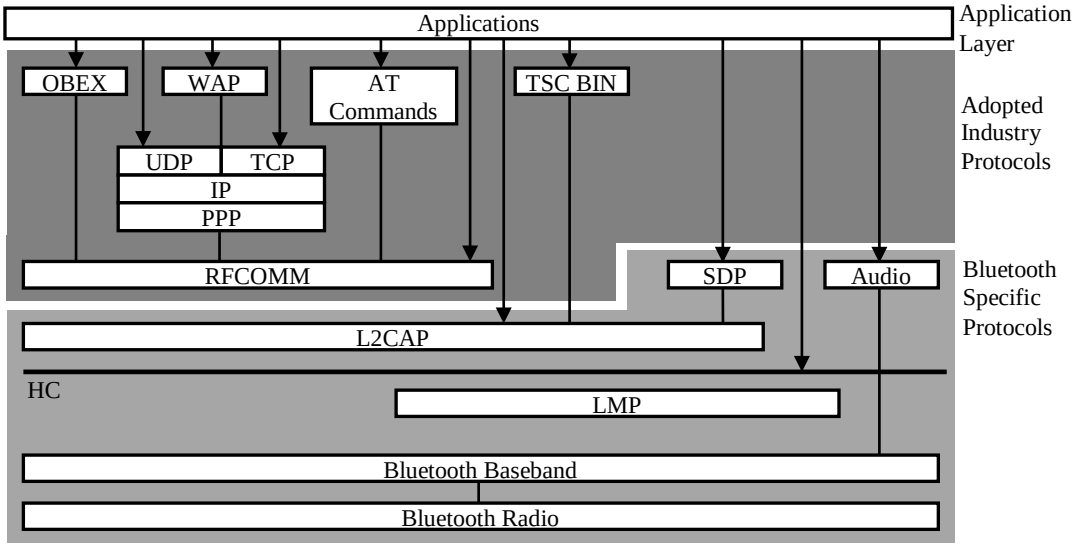
Obr. 2.6: Bluetooth piconet (upraveno z [1])

Piconet tvoří dvě a více zařízení, kde vždy jedno vystupuje v rámci tohoto piconetu jako master (M), dalších 7 zařízení může být v režimu slave (S) a více jak 200 zařízení může být součásti piconetu ve stavu zaparkovaný (P). Všechna zařízení v piconetu mají stejnou hopping sekvenci. Zařízení se tedy stane součástí piconetu po synchronizaci hopping sekvenci. Hopping sekvenci určuje v piconetu master zařízení pomocí 48 bitového ID, které je celosvětově unikátní. Všem aktivním zařízením je následně přidělena 3 bitová adresa (AMA - active member address), zaparkovaná zařízení jsou adresovány 8 bitovou adresou (PMA parked member address). Několik blízkých piconetů se může sloučit do uskupení zvaného scatternet a umožnit tak pružnější komunikaci. Spojení je realizováno sdílením zařízení master nebo slave. V jednom piconetu může být zarízení v režimu slave a v druhém piconetu může být to stejné zařízení v režimu master, jak je znázorněno na obrázku 2.7



Obr. 2.7: Bluetooth scatternet

Protokolová architektura je organizovaná do vrstev, jak znázorňuje obrázek 2.8. Nejnižší vrstva definuje komponenty specifické pro Bluetooth. Prostřední vrstva se skládá ze standardních průmyslových protokolů, které byly přizpůsobeny k využití Bluetooth technologie. Využití existujících protokolů umožňuje snadnější portování aplikací. Nejvyšší vrstvou je aplikační vrstva [1, 2, 12].



Obr. 2.8: Protokolová architektura Bluetooth [2]

Bluetooth Radio – specifikace rozhraní, frekvence, modulace a vysílacího výkonu.

Bluetooth Baseband – popis základní organizace spojení, formát paketu a časování

LMP (Link Manager Protocol) – zodpovídá za navázání spojení mezi zařízeními, za řízení a sestavení komunikace, dohodnutí délky paketů používaných ke komunikaci a řízení spotřeby. Další důležitou činností je generování, výměna a řízení kanálu a výměna šifrovacích klíčů pro autentifikaci a šifrování.

HCL (**Host Controller interface**) – jednotné rozhraní a metoda přístupu k hardwaru Bluetooth. Obsahuje příkazové rozhraní, správu kanálu a monitor stavu hardwaru.

L2CAP (Logical Link Control and Adaption Protocol) – poskytuje služby vyšším vrstvám pro spojované (connection-oriented), nespojované (connectionless-oriented) datové přenosy a signalizaci (signaling command).

SDP (Service Discovery Protocol) – definice vyhledávání služeb serverů klientským Bluetooth zařízením.

Audio - definuje služby pro přenos zvuku mezi jedním nebo více Bluetooth zařízením. Přenosu zvuku se neúčastní L2CAP vrstva.

RFCOOM (Radio Frequency Communications Port) – emulace sériového portu. Protokol poskytuje služby vyšším vrstvám, které používají pro přenos dat sériovou linku.

PPP (Point-to-Point) – protokol pro přenos paketů.

IP (Internet Protocol) – propojení zařízení Bluetooth se zařízeními na internetu. Aplikace využívající k přenosu dat protokol IP vytvářejí **UDP** nebo **TCP** spojení přes protokol PPP.

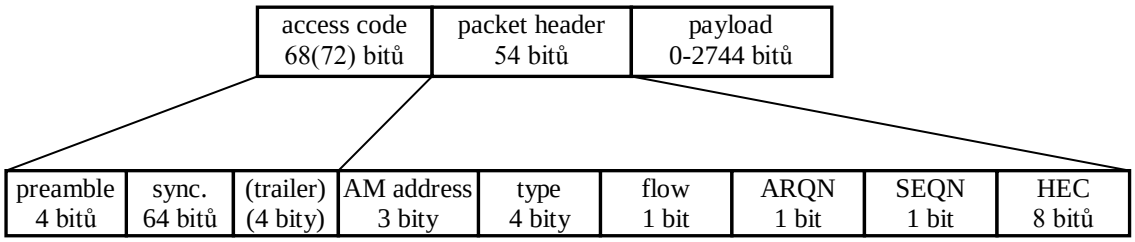
OBEX (Object Exchange) – protokol pro výměnu objektů. Vychází ze specifikace IrDA, kde sloužil k výměně souborů.

WAP (Wireless Application Protocol) – protokol pro bezdrátovou komunikaci, určený ke zpřístupnění internetových služeb.

TCS BIN (Telephony Control – Binary) – definuje sestavení a řízení linky pro bitový přenos hlasu a dat mezi jednotlivými Bluetooth zařízeními.

AT Commands (Advaced Telephony) – servisní příkazy pro telefonii. Primární využití je pro řízení modemu, ale podporuje i faxové transfery.

Struktura paketu na nejnižší úrovni (baseband) je následující (Obr. 2.9):



Obr. 2.9: Struktura paketu na nejnižší úrovni (upraveno z [1])

Access code slouží pro synchronizaci a identifikaci. Rozlišujeme tyto tři základní typy – CAC (Channel Access Code) pro identifikaci paketů v piconetu, DAC (Device Access Code) sloužící k signalizaci a IAC (Inquiry Access Code) pro vyhledávání okolních zařízení.

Packet Header obsahuje informace o cílové adrese (AM address), typu paketu (type), příznak řízení toku (flow), potvrzovací příznak (ARQN), sekvenční číslo paketu (SEQN) a zabezpečení hlavičky (HEC).

Payload jsou přenášená užitečná data.

Vývojové řady Bluetooth:

Bluetooth 1.0 a 1.0B
- značné množství problémů
- produkty od různých výrobců nebyly schopné komunikovat

Bluetooth 1.1
- vytvořen standard IEEE 802.15.1-2002
- odstraněny nedostatky z verze 1.0B
- přidáno RSSI – indikátor síly přijímaného signálu

Bluetooth 1.2
- vytvořen standard IEEE 802.15.1-2005
- zpětně kompatibilní s verzí Bluetooth 1.1
- rychlejší spojení jednotlivých zařízení
- přidáno AFH (Adaptive FHSS) – zvýšení odolnosti proti nežádoucím interferencím rádiových frekvencí
- přidáno eSCO (Extended Synchronous Connections) – zvýšení kvality audia spojení, opětovným přenosem poškozených dat.
- vyšší přenosové rychlosti 721 kb/s oproti verzi 1.0

Bluetooth 2.0
- zpětně kompatibilní s verzemi 1.1 a 1.2
- přidáno EDR (Enhan-ced Data Rate) – zvýšení přenosové rychlosti až na 2,1 Mbit/s
- nižší energetické nároky

Bluetooth zařízení se dále dělí do tříd podle vysílacího výkonu, s čímž souvisí i vzdálenost na kterou jsou schopny komunikovat. Rozdělení je platné pro všechny vývojové řady.

Třída	Max. výstupní výkon	Komunikační dosah	Využití
1	100 mW	100 m	Bluetooth sítě
2	2,5 mW	10 m	Mobilní zařízení (PDA, telefon)
3	1 mW	1 m	Handsfree zařízení

Tab. 2.2: Třídy Bluetooth zařízení

Technologie Bluetooth se dnes vyskytuje v drtivé většině vyráběných mobilních zařízení. Osobní počítače mají potřebné součásti osazeny přímo na základní desce nebo se připojuje Bluetooth modul pomocí USB rozhraní.

Výhody použití Bluetooth:
- Bezlicenční pásmo
- Velmi rozšířený standard
- Podpora mnoha zařízení a výrobců
- Nízká spotřeba energie

Nevýhody použití Bluetooth:
- Rušení na bezlicenčním pásmu
- Omezení komunikační vzdálenosti

2.1.4 WiFi - IEEE 802.11

WiFi je označován standard WLAN (Wireless Local Area Network) sítě podle specifikace IEEE. Název WiFi vznikl jako analogie k systémům HiFi (High Fidelity), ale WiFi nemá význam zkratky, je to pouze označení standardu – 802.11b.

Podvýbor IEEE 802.11 zahájil svoji práci na funkčních požadavcích, specifikaci požadované šíře pásma a protokolu řízení přístupu k bezdrátovému přenosovému prostředku v roce 1990. V červnu 1997 byl schválen návrh normy IEEE 802.11 specifikující příslušnou MAC (Media Acces Control) a fyzickou vrstvu bezdrátových lokálních sítí [11].

Zařízení Wi-Fi lze provozovat v bezlicenčních frekvenčních pásmech, která se nacházejí na frekvencích 2,4 až 2,48 GHz a 5,1 až 5,8 GHz. Standardy 802.11 jsou definovány na prvních dvou vrstvách ISO/OSI modelu. Spojová vrstva je reprezentovaná podvrstvou LLC (Logical Link Control) a podvrstvou MAC (Media Access Control). Tato vrstva se v rámci specifikace 802.11 neliší, odlišnosti jsou pouze v řešení fyzických vrstev, jak je znázorněno na obrázku 2.10.

<table>
<tr><td colspan="6">Řízení logického spoje – podvrstva LLC</td><td rowspan="2">Spojová vrstva</td></tr>
<tr><td colspan="6">Řízení přístupu k médiu – podvrstva MAC</td></tr>
<tr><td colspan="5">2,4 GHz</td><td>5 GHz</td><td rowspan="3">Fyzická vrstva</td></tr>
<tr><td>IEEE 802.11 FHSS</td><td>IEEE 802.11 DFIr</td><td>IEEE 802.11 DSSS</td><td>IEEE 802.11b DSSS</td><td>IEEE 802.11g OFDM/DSSS</td><td>IEEE 802.11a OFDM</td></tr>
<tr><td colspan="3">1,2 Mbit/s</td><td>11 Mbit/s</td><td colspan="2">54 Mbit/s</td></tr>
</table>

Obr. 2.10: Referenční model ISO/OSI pro 802.11

Vrstva přístupu k médiu (MAC) využívá ve standardu 800.11 protokol CSMA/CA (Carrier-sense, multiple-access, collision avoidance). Tento protokol používá techniku předcházení kolizi, neboť v sítích používajících bezdrátové médium při vysokofrekvenčním přenosu je detekce chyb obtížná. Protokol CSMA/CA zajišťuje minimum kolizí použitím čtyř rámců [13]:

- RTS (Ready to send) – připraven ke komunikaci
- CTS (Clear to send) – volná komunikační cesta
- ACK (Acknowledge) – potvrzení přijetí
- NAV (Network allocation vector) – doba rezervování sítě

Před vlastním vysíláním nejprve stanice sítě vyšle požadavek na komunikaci zasláním rámce RTS s udáním adresy příjemce a hodnotou rezervačního parametru. Na základě přijatého RTS se v každém z uzlů vypočítá NAV, čímž se ostatní uzly upozorní na vytížení sítě. Příjemce odpoví pomocí zaslání CTS, tím potvrdí, že je schopen přijímat. Neobdrží-li vysílací uzel CTS, je to považováno za kolizi a celý proces začíná znovu. Po úspěšném přijetí dat zasílá přijímací stanice potvrzení o přijetí (ACK).

Vrstva řízení logického spoje (LLC) poskytuje uživatelům tři typy služeb [11]:

- **Nepotvrzovaná služba bez spojení**

  Vysílání přijímání rámců bez navazování spojení, potvrzování, bez chybového řízení a řízení toku. Detekce chyb a zničení rámce řeší úroveň MAC. Každý paket se přenáší samostatně, různými cestami v síti a musí být nutně vybaven všemi potřebnými informacemi, jako jsou adresa a pořadí paketu ve zprávě.

- **Služba se spojením**
  mezi přístupovými body se navazuje logické spojení. Je zajištěno spolehlivé doručení rámců pomocí chybového řízení, řízení toku, potvrzování a pořadových čísel. Pouze první paket obsahuje adresové informace, ostatní pakety jsou již doručovány po stejné cestě.
- **Potvrzovaná služba bez spojení**
  nedochází k navazování spojení, potvrzení doručení dat se provádí.

Z hlediska využití WiFi v mobilních zařízeních se uplatňují zejména standardy 802.11b a 802.11g.

802.11b na fyzické vrstvě využívá metodu rozprostřeného spektra DSSS s klíčováním CCK. DSSS využívá k rozprostření dat před přenosem Barkerovu sekvenci, každý přenášený bit je zpracován touto sekvencí. Do přenášených dat se zavádí velká redundance, která ovšem na straně přijímače zlepšuje rekonstrukci dat. Na fyzické vrstvě je dosahováno celkem čtyř přenosových rychlostí: 11 Mbit/s, 5,5 Mbit/s, 2 Mbit/s, 1 Mbit/s. Maximální rychlosti se dosahuje pouze na krátkou vzdálenost s rostoucí vzdáleností přijímače a vysílače roste chybovost přenosu a tím i rychlost.

802.11g dosahuje vyšší rychlosti než 802.11b a to až 54 Mbit/s. Využívá ortogonálního multiplexu s kmitočtovým dělením OFDM. Přenos u OFDM (Orthogonal Frequency Division Multiplexing) je rozložen do mnoha úzkých subkanálů, které jsou přenášeny paralelně. OFDM využívá čtyři druhy modulace: BPSK, QPSK, 16-QAM a 64-QAM. Rychlosti 802.11g na bázi OFDM jsou: 54, 48, 36, 24 Mbit/s s využitím modulace 16-QAM, 18, 12 Mbit/s s využitím modulace QPSK a 9, 6 Mbit/s s modulací BPSK. 802.11g využívá i metodu DSSS a tak je zajištěna zpětná kompatibilita se systémy 802.11b na rychlostech 11 Mbit/s, 5,5 Mbit/s, 2 Mbit/s, 1 Mbit/s.

Rozhraní WiFi se v mobilních zařízeních objevuje stále častěji, modely disponující WiFi jsou z vyšších cenových kategorií. Osobní počítače mají WiFi modul přímo na základní desce, další variantou je připojení pomocí USB rozhraní.

Výhody použití WiFi:

- Bezlicenční pásmo
- Vysoké přenosové rychlosti
- Dobrá komunikační vzdálenost

Nevýhody použití WiFi:

- Energetická náročnost

2.1.5 Zigbee, Wibree

Standar 802.15.4 označován jako Zigbee patří do skupiny WPAN sítí. Jedná se o pomalou síť s minimální spotřebou energie. Pracuje v pásmu 858 MHz na rychlosti 20 kbit/s, v pásmu 902 – 928 MHz rychlostí 40 kbit/s a v pásmu 2,4 GHz rychlostí 250 kbit/s. V oblasti mobilních zařízeních není tento standard zatím moc rozšířen. Z hlediska jeho nízké energetické náročnosti se využívá zejména k připojení periferií (handsfree). Standard vznikl v roce 2003 a o jeho vývoj se stará Zigbee Alliance viz. [14].

Wibree je technologie vyvinutá Nokia Research Center. Wibree je založeno na podobných principech jako Bluetooth, mají stejné pracovní pásmo a přenosové rychlosti jsou rovněž podobné. Oproti Bluetooth má být snížena spotřeba a rozměry zařízení na minimum při zachování příznivé ceny. Wibree lze implementovat jako samostatný čip, nebo v kombinaci s Bluetooth jako duálmódový typ. Stejně jako Zigbee slouží Wibree především k připojení periferií mobilního zařízení.

2.2 OS a vývojové prostředí pro mobilní zařízení

Operační systémy pro mobilní zařízení plní obdobnou funkci jako u osobních počítačů, zprostředkovávají vzájemnou komunikace hardwaru a softwaru. OS u mobilního zařízení rovněž obstarává veškeré funkce telefonie – navazování hovorů, spojování, evidenci uskutečněných hovorů, odesílání SMS zpráv. Existuje celá řada operačních systémů, výrobce mobilních zařízení se nejčastěji zaměří na jeden druh, operační systém proto bývá často svázán se značkou pořizovaného zařízení. Na našem trhu se nejčastěji setkáváme s operačním systémem Windows mobile a Symbian.

2.2.1 Windows mobile

Windows mobile je založeno na svém předchůdci Windows CE (Windows Embedded Compact) společnosti Microsoft. První verze systému Windows CE se objevila v roce 1996, technologie jádra byla podobná systému Windows NT. Postupně docházelo k vývoji dalších verzí, přibývala podpora nejrůznějších periferií a aplikací. Následuje přehled základních verzí, jejich označení a vlastnosti:

Pocket PC 2000
Uveden v lednu roku 2000, jedná se o Windows CE 3.0 s pracovním označením Rapier. Pocket PC 2000 nebyl standardizován s určitou architekturou. Pocket PC 2000 pracoval na CPU architekturách: SH3[1], MIPS[2] a ARM[3]. Podporoval paměťové karty a přenos po rozhraní IrDA, rozlišení obrazovky bylo 240 x 320 obrazových bodů ve standardu QVGA. Mezi programovou výbavu patřil např.: Pocket Office, Pocket Internet Explorer a WM Player.

Pocket PC 2002
Uveden v říjnu roku 2001, na jádře Windows CE 3.0 s označením Merlin. Uživatelské rozhraní bylo přívětivější, zavedena byla možnost měnění témat prostředí.

SmartPhone 2002
Stejná verze jako Pocket PC 2002, určená především pro smartphone. Výrazným rysem byla podpora GSM telefonie.

Windows Mobile 2003
Vydáno v červnu 2003, postaveno na jádře Windows CE 4.2 pod označením Ozone. Verze byla vydána ve čtyřech edicích v závislosti na cílové platformě (Pocket PC, Smartphone) a programovém vybavení. Rozšířena byla o schopnost komunikovat na rozhraní Bluetooth. Aplikace ze starších verzí byly rovněž vylepšeny.

[1] SH3 – verze procesoru pro mobilní zařízení z rodiny SuperH procesorů. 32 bitové procesory s RISC instrukční sadou společnosti Hitachi.
[2] MIPS – procesory load/store architektury s RISC instrukční sadou.
[3] ARM – 32 bitové procesory s RISC instrukční sadou, používané především v mobilním odvětví

Windows Mobile 2003 Second Edition
Na trh uvedeno v březnu 2004 a označeno jako Ozone update. Výrazným zlepšením oproti předešlé verzi bylo rozšíření komunikace o WiFi.

Windows Mobile 5
Uveden v květnu 2005, postaveno na jádře Windows CE 5.0 s podporou. NET Compact Framework 1.0. Systém nesl označení Magneto. Nejvýraznější zlepšení bylo v oblasti šetření energie, nárůst pohotovostí doby přístrojů byl až o polovinu. Podpora lokalizačního systému GPS.

Windows Mobile 6
Uveden v prosinci 2007, postaveno na jádře Windows CE 5.2, nese označení Crossbow. Vylepšení zaměřena především na online práci. Obrazovky o rozlišení až 800 x 480 bodů ve standardu WVGA.

Windows Mobile 6.1
Uveden v dubnu 2008 jako upgradovaný WM 6. Přidána domácí obrazovka s rychlým přístupem k aplikacím, podporuje rovněž formáty sady MS Office 2007.

Další plánovanou verzí Windows Mobile je 6.5. Vylepšení se dočkají především aplikace (přechod na vyšší verze). Do budoucnosti se počítá i s verzemi 7 a 8, které budou zaměřeny především na zlepšení ovládání přístroje pomocí pohybu přístroje (akcelerometry) a gest rukou.

Zařízení pracující na systému Windows Mobile lze spojit s PC pomocí rozhraní USB, bezdrátově pomocí Bluetooth, WiFi nebo pomocí infračerveného rozhraní. K usnadnění komunikace vyvinula společnost Microsoft aplikaci ActiveSync. Hlavním účelem ActiveSync je synchronizace zařízení s počítačem. Po spojení zařízení s PC (USB, Bluetooth) dojde k automatické synchronizaci složek, kontaktů, pošty a dalších položek. ActiveSync pracuje na operačních systémech Windows 2000/XP/2003, ekvivalentem pro Windows Vista je aplikace Windows Mobile Device Center, která je rozšířena o další funkce (správa programů, nastavení mobilního zařízení).

Aplikace na straně zařízení se rozdělují podle způsobu běhu programového kódu. Nativní kód běží přímo v procesoru mobilního zařízení, zatímco řízený kód (managed) je společný pro všechny typy procesorů a kompiluje se přímo na mobilním zařízení až v okamžiku spuštění (just in time - JIT). Instrukční sada procesorů zařízení nebyla vždy stejná, Windows Mobile pracuje nejčastěji na procesorech typu SH3, MIPS a ARM, proto se musí aplikace běžící přímo na procesoru (nativní kód) překládat a linkovat pro každý typ procesoru zvlášť. Výhodou nativního kódu oproti řízenému je jeho rychlost, nutnost překladu kódu přímo pro daný procesor zajišťuje jeho optimalizaci. Řízený kód disponuje velkou kompatibilitou a vývoj aplikace není tak časově náročný jako u nativního kódu. Trend vývoje aplikací v dnešní době ukazuje na ústup nativního kódu a upřednostňuje se kód řízený [4].

2.2.1.1 .NET Framework

.NET Framework je platforma pro vytváření a provozování aplikací na stolních počítačích. Primárním účelem .NETu je vývoj aplikací v řízeném kódu. Platforma byla poprvé přestavena v roce 2001. V roce 2002 byla oficiálně uvolněna její první verze 1.0, spolu s ní také vyšlo vývojové prostředí Visual Studio .NET a specielně vyvinutý jazyk C# (1.0). Tato verze frameworku byla k dispozici pro Windows 98, Me, NT 4.0, 2000 a XP. Postupem času přicházely nové verze .NETu a s nimi se také objevovala nová vývojová prostředí:

.NET 1.1 v roce 2003, tato verze je standardně součástí operačního systému Windows Server 2003. Vývojové prostředí MS Visual Studio .NET 2003.

.NET 2.0 v roce 2005, lze volně stáhnout od společnosti Microsoft, byla také součástí Service Packu 2 pro Windows XP. Vývojové prostředí MS Visual Studio 2005. Společně vychází také nová verze jazyku C# 2.0.

.NET 3.0 v roce 2006, stažitelný pro operační systémy Windows XP a Server 2003, součástí Windows Vista a Server 2008.

.NET 3.5 v roce 2007. Vývojové prostředí MS Visual Studio 2008.

V současné době je v distribuci Service Pack 1 pro verzi 3.5, jenž doplňuje některé funkce verze 3.5. Dalším nástupcem je .NET 4.0 a připravuje se také MS Visual Studio 2010.

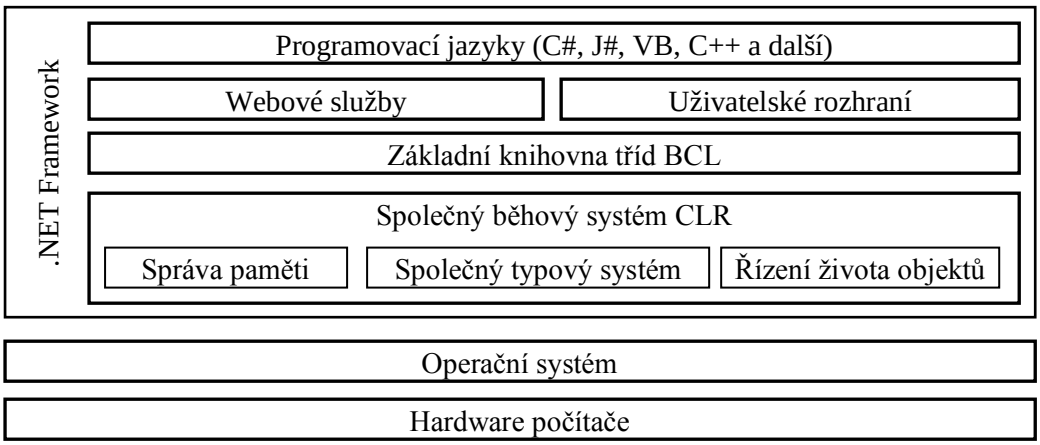
Struktura prostředí .NET je znázorněna na obrázku 2.11. Základní vrstvu nad hardwarem počítače tvoří operační systém, nad ním se pak nachází framework, jehož základními komponenty jsou společný jazykový běhový modul CLR (Common Language Runtime) a základní knihovna tříd BCL (Base Class Library).

CLR se nachází nad operačním systémem a poskytuje virtuální prostředí pro hostování řízených aplikací. Po spuštění aplikace v řízeném kódu, CLR nahraje modul obsahující spustitelný soubor a vykoná v něm obsažený kód. Kód zacílený na CLR se skládá z instrukcí zapsaných v pseudo-strojovém kódu označovaném za společný zprostředkovací jazyk CIL (Common Intermediate Language). Instrukce CIL se na požádání JIT (viz. výše) zkompilují do nativního strojového kódu za běhu. Ve většině případů se určitá metoda zkompiluje technikou JIT jen jednou (při svém prvním volání) a následně se uloží do paměti, aby ji příště bylo možné vykonat bez zpoždění. Kód, který se nikdy nezavolá, se ani nezkompiluje. Kompilace JIT nepochybně snižuje výkonnost, její negativní efekty jsou minimalizovány skutečností, že daná metoda se kompiluje jen jednou během celého života aplikace. Teoreticky dokáže být kód zkompilovaný JIT výkonnější než obvyklý kód, protože kompilátor JIT dokáže optimalizovat nativní kód vytvářený pro určitou verzi hostitelského procesoru, na kterém právě pracuje. Při převodu instrukcí CIL na nativní kód, používá CLR proces ověřování kódu zajišťující typovou bezpečnost daného kódu, proto není možné vykonat nějakou instrukci, která přistupuje k paměti, k níž nemá oprávnění, rovněž nelze převést typ na něco, čím není, neboť taková operace není typově bezpečná.

Silnou stránkou řízeného prostředí je také inteligentní správa paměti. Prostředky alokované řízeným kódem se samočinně uvolňují z paměti, tj. programátor paměť alokuje, ale už ji neuvolňuje. CLR zahrnuje nástroj uvolňování paměti, který sleduje odkazy na objekty vytvářené programátorovým kódem a tyto objekty ničí, když je jimi obsazená paměť zapotřebí jinde. Algoritmům obstarávajícím uvolňování paměti se říká Sběr odpadků (Garbage collection). Algoritmus alokování paměti používaný v CLR je natolik výkonný, že jeho rychlost převyšuje odpovídající rutiny alokování paměti v runtimovém modulu jazyka C [17].

Další nespornou výhodou běhu aplikací v hostitelském prostředí CLR je to, že veškerý kód je omezen na CIL, tudíž je čistě na programátorovi v jakém jazyce svoji aplikaci napíše. Všechny řízené aplikace používají stejné API z knihovny tříd rámce .NET. Microsoft nabízí kompilátory CIL pro pět jazyků: C#, J#, C++, Visual Basic a Jscript. Dalšími jazyky jsou např.: Perl, Python, Eiffel, COBOL. Obecně se může jednat o jakýkoliv jazyk, jenž splňuje specifikaci CLS (Commnon Language Specification), což je sada pravidel, která musejí být splněna, aby se kód z daného jazyka dal překompilovat do kódu jazyka CIL.

Základní knihovna tříd BCL poskytuje základní stavební kameny pro vývoj aplikací. Jedná se o knihovnu dostupnou všem jazykům využívajících .NET Framework. BCL obsahuje třídy, které zapouzdřují nejčastější pomocné funkce, jako je práce se soubory, interakce s databázemi, zobrazování grafiky, třídění, síťovou komunikaci, implementaci front, textové výstupy a další. Konkrétní seznam a popis jednotlivých tříd je dostupný na [19].



Obr. 2.11: Základní struktura prostředí .NET (převzato z [18])

2.2.1.2 .NET Compact Framework

Pro mobilní zařízení se využívá .NET Compact Framework (dále CF). Jedná se o hardwarově nezávislé prostředí, ve kterém mohou běžet řízené aplikace, s přihlédnutím k omezením daného zařízení. CF nese vlastnosti plnohodnotné platformy, struktura je obdobná (Obr. 2.11), jen první dvě vrstvy jsou rozdílné. Nejnižší vrstva je tvořena fyzicky mobilním zařízením, druhou vrstvu pak představuje operační systém Windows Mobile. Skutečnost, že mobilní zařízení jsou hardwarově méně výkonné, než desktopové počítače, vede k určitým omezením CF oproti standardní verzi. Nejvýznamnější omezení se týkají paměti a zobrazovacích schopností zařízení. Kompaktní verzi Frameworku postačují 2 MB paměti, plnohodnotný .NET spotřebuje více než 43 MB [4]. Z hlediska knihovny tříd je CF podmnožinou standardního Frameworku. Microsoft uvádí, že je pokryto přibližně 30 % všech funkcí. Vývojové verze jsou taktéž obdobné, v současné době je v distribuci CF ve verzi 3.5.

2.2.1.3 Vývojové nástroje pro Windows Mobile

V současné době je pro vývojáře k dispozici několik vývojových nástrojů. Společnost Microsoft dává k dispozici nástroje pokrývající v podstatě všechny mobilní zařízení založené na architektuře Windows Mobile. Microsoft se snaží sjednotit nabízené nástroje a poskytnout tím vývojáři nástroj univerzální, založený na jednotné platformě .NET.

MS eMbedded Visual Tools je vývojářské prostředí s intuitivním GUI, integrovaným kompilátorem a ladícími prostředky. Součástí MS eMbedded Visual Tools je eMbedded Visual C++ a eMbedded Visual Basic.

eMbedded Visual C++

- Umožňuje vytvářet nativní aplikace pro mobilní zařízení
- Vhodné pro aplikace, u nichž je prioritou rychlost

- Vhodné pro hry a aplikace, které vyžadují rychlou grafiku
- Obsahuje softwarový emulátor

eMbedded Visual Basic

- Vhodné pro aplikace s důrazem na rychlost vývoje
- Vhodné pro aplikace typu uživatelského rozhraní složeného z komponent
- Umožňuje tvorbu prototypů aplikací RAD (Rapid Application Development)
- Vhodné pro tvorbu jednoduchých nástrojů

Visual Studio .NET (2005/2008) využívá možností, které přináší MS .NET Compact Framework, umožňuje vývoj distribuovaných aplikací. Vývojáři poskytuje knihovnu tříd pro pohodlný a rychlý vývoj aplikací, stejně jako i tvorbu tříd nových a opětovně použitelných. Vývojář vyvíjí aplikace, které používají stejné nástroje jak na mobilním zařízení, tak i na stolním počítači. Celý vývojový proces je sjednocen a poskytuje přenositelný a universální zdrojový kód nezávislý na procesoru i verzi operačního systému Windows. Vývoj, případně migrace aplikací probíhá v programovacích jazycích Visual Basic a Visual C#.

- Tvorba aplikací využívající XML a webové služby.
- Tvorba aplikací, které musí pracovat v on-line a off-line režimu
- Vhodné pro aplikace vyžadující spolehlivé a bezpečné prostředí
- Vhodné pro aplikace ležící na serveru, ke kterým se přistupuje pomocí webového prohlížeče

Visual studio (v poslední verzi 2008) je velice silným vývojovým nástrojem, jak pro programování desktopových, tak i mobilních aplikací. Vývoj mobilní aplikace ve Visual Studiu velmi usnadňuje použití SDK (Software Development Kit). SDK je sada nástrojů specifických pro danou platformu a verzi operačního systému, nedílnou součástí jednotlivých SDK pro Windows Mobile je emulátor, jenž umožňuje rychlejší a pohodlnější tvorbu některých programů, bez nutnosti vlastnit mobilní zařízení. Obrázek 2.12 zobrazuje emulátor mobilního zařízení s operačním systémem Windows Mobile 6. Kompilace a nahrání programu přímo do mobilního zařízení probíhá za pomocí aplikace ActiveSync (viz. výše). Program se zkompiluje na straně PC, je-li zařízení připojeno k PC prostřednictvím ActiveSync, dojde k automatickému nahrání programu do zařízení a následně k jeho spuštění.

Obr. 2.12: Emulátor Windows Mobile 6

2.2.2 Symbian

Symbian je operační systém navržený specielně pro mobilní zařízení. Vznik systému je spojen se vznikem sdružení společností Motorola, Ericsson, Nokia a Psion. V současné době je již výhradním vlastníkem společnost Nokia a do budoucna by měla vytvořit ze Symbianu otevřený operační systém.

Symbian vychází z operačního systému EPOC, pracujícím na zařízeních společnosti Psion. Je provozován zejména na procesorech s ARM architekturou. Používá objektovou orientaci a je napsán v jazyce C++, s malým množstvím assembleru, nacházejícím se na nejnižších úrovních. Většina aplikací vyvíjených pro tento operační systém je napsána taktéž v jazyce C++.

Zařízení se systémem Symbian jsou momentálně založeny na následujících uživatelských rozhraních:

Platforma Series 60 (označována také jako S60) – Nejrozšířenější platforma. Vyvinuta pro mobilní zařízení klasické konstrukce, s ohledem na snadné ovládání jednou rukou. Dále je rozdělena do dalších edicí podle využité verze OS.

1st Edition – Symbian verze 6.1
2nd Edition – Symbian ve verzích 7.0 – 8.1
3rd Editon – Symbian verze 9.1, 9.2 (Feature Pack 1), 9.3 (Feature Pack 2)

Platforma Series 80 (S80) – Provozováno na zařízeních vybavených plnohodnotnou klávesnicí (komunikátory). Vyvíjeno na Symbianu ve verzích 6.0 a 7.0s

Platforma Series 90 (S90) **–** V současné době využito jediným zařízením (Nokia 7710). Symbian 7.0

UIQ – vyvinuto dceřinou společností Symbianu. Zařízení UIQ jsou vybaveny velikou dotykovou obrazovkou. Hlavním ovládacím prvkem je přiložený stylus.

Z výše uvedeného je zřejmé, že Symbian je distribuován ve spoustě verzí. Není vždy zaručena kompatibilita mezi jednotlivými verzemi, někdy se jedná i o poměrně velké rozdíly při psaní kódu ve dvou po sobě jdoucích verzích.

Aplikace pro systém Symbian se nejčastěji tvoří v jazyce C++. Programování aplikací pro Symbian nepatří k nejsnadnějším. Je zcela na programátorovi, aby správně uvolňoval přidělené prostředky. K tomu slouží techniky jako dvoufázová konstrukce nebo využívání úklidového zásobníku.

Nástrojů pro vývoj aplikaci je několik. Jedním ze způsobů je psaní kódu v editoru a následné sestavení v příkazové řádce. Společnost Nokia vyvinula IDE založené na Eclipse pod názvem Carbide.c++, je to prostředí pro psaní a ladění kódu, doplněné o emulátor přístroje (viz. Obr. 2.13). Pro MS Visual studio je dostupný modul Carbide.vs, který podporuje vývoj aplikací ve Visual studiu. Pomocí tohoto modulu a doinstalovaného SDK pro konkrétní verzi operačního systému lze pohodlně vytvářet a testovat aplikace.

Obr. 2.13: Emulátor Symbianu v prostředí Carbide (Series 60)

2.3 MS SQL Server

Microsoft SQL Server je relační databázový systém, jehož hlavními dotazovými jazyky jsou SQL (Structured Query Language) a T-SQL (Transact-SQL). SQL server se zrodil na základě obchodu, jejž uzavřely firmy Microsoft a Sysbase v roce 1987. O dva roky později byla vydána první verze MS SQL Srver 1.0 určená pro systémy OS/2[4]. Podpora systému MS Windows NT se objevila až s verzí 4.21, která byla vydána v roce 1993 [22]. V současné době je na trhu verze MS SQL Server 2008.

MS SQL Server je složen z následujících komponent (MS SQL Server 2005):

Database Engine (Databázový stroj) – základní služba pro ukládání, zpracování a zabezpečení dat. Poskytuje kontrolovaný přístup a rychlé zpracování transakcí, které zahrnuje vytvoření tabulek pro ukládání dat a databázových objektů, jako jsou indexy, pohledy a uložené procedury.

Reporting Services (Reportovací služby) – poskytují kompletní platformu pro vytváření, správu a distribuci databázových sestav.

Analysis Services (Analytické sužby) – služby pro obchodní systémy, aplikace online analytického zpracování (OLAP) a dolování dat (data mining). Analytické služby umožňují agregovat data z více zdrojů, například relačních databází, a zpracovat tato data různými způsoby [23].

Notification Services (Oznamovací služby) – zahrnují stroj pro rozesílání oznámení a klientské komponenty pro vytváření a včasné odesílání zpráv uživatelům.

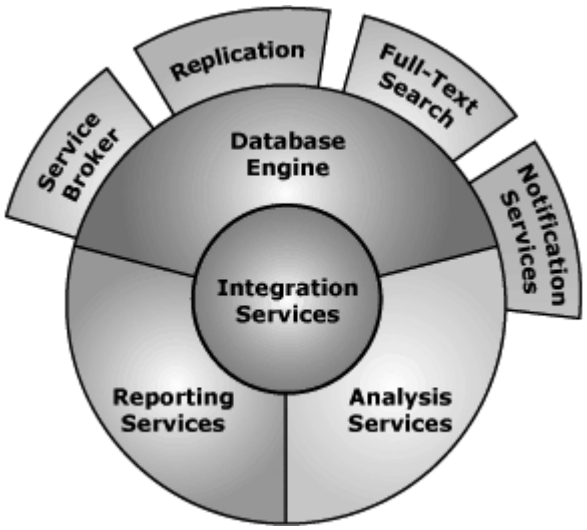
Integration Services (Integrační služby) – služby pro extrakci a transformaci dat z více datových zdrojů a jejich přesun na jeden nebo více cílových zdrojů dat.

[4] OS/2 – operační systém vyvíjený především společností IBM, v dnešní době je za jeho nástupce považován systém eComStation

Full-Text Search (Fulltextové vyhledávání) - zprostředkovává fulltextové vyhledávání umožňující aplikaci jazykových dotazů na data uložená v tabulkách SQL Serveru.

Replication (Replikace) – zvyšuje dostupnost dat díky distribuci dat do více databází.

Service Broker (Zprostředkovatel služeb) – zajišťuje spolehlivé řazení požadavků do fronty a jejich rozesílání. Požadavkem je například databázový dotaz.



Obr. 2.14 : Komponenty MS SQL Serveru 2005 (převzato z [24])

MS SQL Server 2005 je distribuován v několika edicích, které jsou rozděleny podle systémových požadavků a poskytovaných služeb:

Workgroup – edice pro menší firmy pracující na systémech Windows 2000, XP Professional a Windows Server 2003. Podporuje až 3 GB RAM a umí využít dva procesory pro symetrický multiprocesing.

Standard – určena pro organizace střední velikosti. Pracuje na stejných systémech jako edice Workgroup. Využívá veškerou dostupnou RAM a až čtyři procesory pro symetrický multiprocesing.

Enterprise – edice pro velké podniky. Umožňuje vytváření záložních clusterů. Tato vlastnost chrání server před poruchami hardwaru a napomáhá opětovnému zotavení při výpadku.

Express – odlehčená verze Serveru je distribuována zcela zdarma. Využívá maximálně jeden procesor a 1 GB RAM, velikost databáze je omezena na 4 GB.

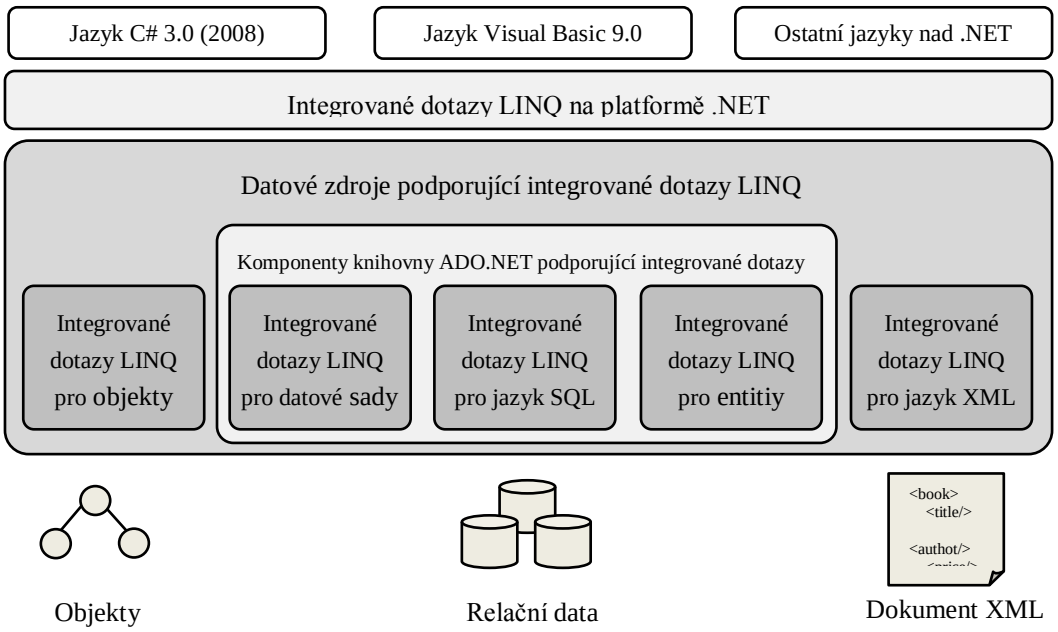
Mobile – kompletní databázové řešení pro mobilní zařízení.

K administrativě SQL severu se využívá grafický nástroj Management Studio pracující s komponentou Database Engine. Jedná se o silný nástroj zprostředkovávající kompletní správu serveru. Pomocí Management Studia je možno registrovat více serverů, na kterých se provádí operace jako vytvoření databáze a tabulek, replikace databází, export a import dat, generování skriptů databáze, tvorba dotazů a uložených procedur.

2.4 LINQ technologie

Technologie LINQ (Lanuage Integrated Query – dotazy integrované do jazyka) je novinkou od společnosti Microsoft uvedenou společně s Visual Studiem 2008 a rozhraním .NET Framework 3.5, která slibuje revoluční změnu způsobu, jakým vývojáři pracovali s daty před vydáním platformy 3.5 [20]. LINQ unifikuje přístup k datům pocházející z velmi různorodých a logicky odlišných datových zdrojů. Umožňuje tedy vytváření dotazů na různé datové zdroje stejným způsobem. Dotazy mohou směřovat na data uložená na SQL serveru, v XML dokumentu, na jména souborů v adresářích atd. Architektura LINQ je znázorněna na obrázku 2.15 a skládá se ze tří základních komponent.

- Integrované dotazy LINQ pro objekty (LINQ to Objects)
- Integrované dotazy LINQ pro knihovnu ADO.NET (LINQ to ADO.NET), zahrnující:
  - Integrované dotazy LINQ pro jazyk SQL (LINQ to SQL)
  - Integrované dotazy LINQ pro datové sady (LINQ to DataSets)
  - Integrované dotazy LINQ pro entitiy (LINQ to Entities)
- Integrované dotazy LINQ pro jazyk XML (LINQ to XML)



Obr. 2.15: Architektura technologie LINQ

Integrované dotazy LINQ pracují s daty pomocí operátorů SQO (Standard Query Operators). Jedná se o množinu metod obsahující např.: metodu *Where* pro definici omezení prvků ve výsledkové sadě, metodu *Select* pro implementaci projekce, nebo metodu *OrderBy* pro seřazení výsledku. Práce s LINQ je po syntaktické stránce vždy stejná, interní funkčnost se může lišit podle umístění dat, s nimiž se pracuje. Z tohoto hlediska rozlišujeme dvě základní varianty:

- LINQ pracující s objekty, které má k dispozici v paměti a přístup k nim je tedy bez režie, v tomto případě je implementováno rozhraní *IEnumerable<T>*. Jedná se o generické rozhraní implementováno kolekcemi ve jmenném prostoru *System.Collections.Generic*.

- LINQ pracující s objekty, které nejsou umístěny přímo v paměti a je tedy nutné tyto objekty zpřístupnit. Typicky se jedná o data umístěná na SQL serveru. Implementace je prováděna rozhraním *IQueryable<T>*, které je odvozeno od *IEnumerable<T>* a přidává především vlastnost *Provider* typu *IQueryProvider*.

2.5 TCP/IP komunikace

TCP/IP (Transmission Control Protocol / Internet Protocol) je standardní rodina protokolů pro komunikaci v počítačových sítích. Využívá se zejména pro komunikaci v Internetu. Architektura TCP/IP je rozdělena do čtyř vrstev [25].

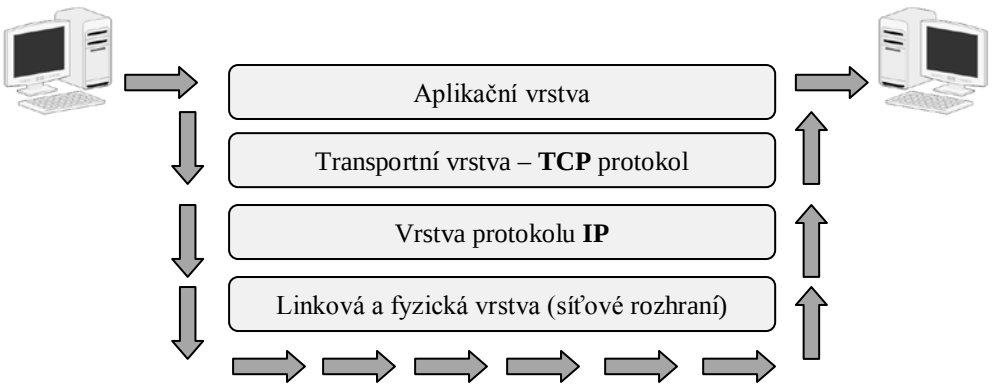
Vrstva síťového rozhraní – Nejnižší vrstva umožňující přístup k přenosovému médiu, odpovídá fyzické a linkové vrstvě ISO OSI modelu.

Vrstva IP protokolu – odpovídá síťové vrstvě ISO OSI modelu. IP-protokol přenáší tzv. IP-datagramy mezi vzdálenými počítači. Každá IP-datagram ve svém záhlaví nese adresu příjemce, což je úplná směrovací informace pro dopravu k adresátovi. Síť může přenášet každý IP-datagram samostatně a mohou k adresátovi dorazit v jiném pořadí, než byly odeslány. Síťové rozhraní v IP-protokolu má přiděleno IP adresu, která je v případě IP verze 4 čtyřbajtová, v případě IP verze 6 šestnáctibajtová.

Transportní vrstva – vrstva protokolu TCP. Protokol IP adresuje pouze síťové rozhraní počítače, pomocí protokolu TCP je určeno spojení dvou konkrétních aplikací. Adresování se provádí pomocí čísla portu, což je dvoubajtové číslo v rozsahu hodnot 0 až 65535. Protokol TCP je spojovanou službou, tj. službou, která mezi dvěma aplikacemi vytvoří virtuální plně duplexní okruh. Přenášené bajty se číslují, v případě ztráty nebo poškození se data znovu vyžádají a jsou opětovně přenesena. Pro přenos se data zabezpečují kontrolním součtem.

Aplikační vrstva – představuje programy využívající síťovou komunikaci. Do této vrstvy patří protokoly jako: HTTP, SMTP, Telnet, FTP, IMAP, POP3 atd.

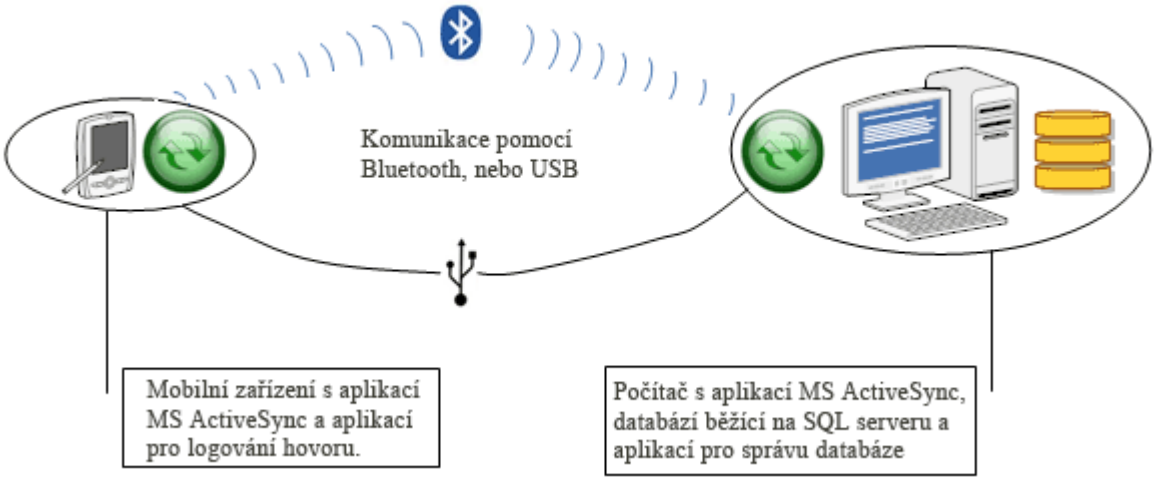
Postup výměny dat mezi vrstvami je přesně definován, každá vrstva využívá služby vrstvy nižší a poskytuje svoje služby vrstvě vyšší. Vrstvový model TCP/IP je znázorněn na následujícím obrázku (Obr. 2.16).



Obr. 2.16: Vrstvový model komunikace TCP/IP

3 Návrh

Po prostudování možností vývoje aplikací pro jednotlivé platformy, zvážení všech úskalí a s ohledem na dostupné prostředky, jsem zvolil operační systém Microsoft Windows Mobile s využitím platformy .NET. Na straně mobilního zařízení poběží aplikace zaznamenávající údaje o uskutečněných hovorech. Aplikace bude realizována jako démon, tudíž bude uživateli nepřístupná a nebude přímo zobrazovat žádné informace. Na straně PC poběží aplikace, která bude spravovat SQL databázi a pomocí GUI bude uživateli zobrazovat statistiky o hovorech. Obě aplikace spolu budou komunikovat pomocí protokolu TCP. Na fyzické vrstvě bude spojení realizováno buďto pomocí USB kabelu, nebo bezdrátově technologií Bluetooth. Jako prostředník pro navázání komunikace poslouží synchronizační program ActiveSync společnosti Microsoft, který je standardně využíván pro přenos informací mezi PC a mobilním zařízením. Funkci celého systému ilustruje následující obrázek (Obr. 3.1). Návrh se zabývá způsobem, jak budou uchovávány informace o hovorech na mobilních zařízeních, komunikačním protokolem pro přenos dat ze zařízení do PC a návrhem struktury databáze.



Obr. 3.1: Schéma navrženého systému

3.1 Ukládání údajů o hovorech na mobilním zařízení

Pro uložení informací o uskutečněných hovorech využijeme XML dokument. Značnou výhodou tohoto přístupu je fakt, že .NET Famework obsahuje metody pro práci s XML dokumenty. Práce s dokumenty bude tedy poměrně snadná, jak na straně PC, tak i na mobilním zařízení. Pro každý den, v němž bude proveden alespoň jeden telefonní hovor, vytvoříme dokument pojmenovaný podle aktuálního kalendářního data.

Do XML dokumentu budeme ukládat následující informace:

- **Telefonní číslo druhého účastníka** – řetězec devíti číselných znaků. V rámci našeho návrhu budeme pracovat s čísly bez předvoleb.

- **Datum a čas uskutečnění hovoru** - řetězec obsahující časové informace ve formátu datového typu DateTime jazyka SQL

- **Typ hovoru** - řetězec, který bude jasně specifikovat, jedná-li se o příchozí, nebo odchozí hovor

- **Délka hovoru** - řetězec obsahující informaci o délce provedeného hovoru

Dokument bude uvozen standardní hlavičkou obsahující informace o verzi a použitém kódování. Jednotlivé záznamy o hovorech budou potomci kořenového elementu pojmenovaného *calls*, tyto elementy budou označeny jako *cl*. Každý z elementů *cl* bude obsahovat atributy, jejichž hodnoty ponesou informace o hovoru.

Jména atributů elementu *cl* a jejich význam:

- *nbr* – telefonní číslo druhého účastníka
- *dt* – datum a čas uskutečnění hovoru
- *typ* – typ hovoru
- *dur* – délka hovoru

Struktura XML dokumentu bude tedy následující (ve formátovaném tvaru):

```
<?xml version="1.0" encoding="utf-8"?>
<calls>
    <cl>
        <nbr>tel_cislo_druheho_ucastnika</nbr>
        <dt>datum_a_cas_uskutecneni_hovoru</dt>
        <typ>typ_hovoru</typ>
        <dur>delka_hovoru</dur>
    </cl>
      .
      .
</calls>
```

Dokument popsaný pomocí DTD[5]:

```
<!ELEMENT    calls    (cl*)>
<!ELEMENT     cl      (nbr, dt, typ, dur)>
<!ELEMENT    nbr      (#PCDATA)>
<!ELEMENT    dt       (#PCDATA)>
<!ELEMENT    typ      (#PCDATA)>
<!ELEMENT    dur       (#PCDATA)>
```

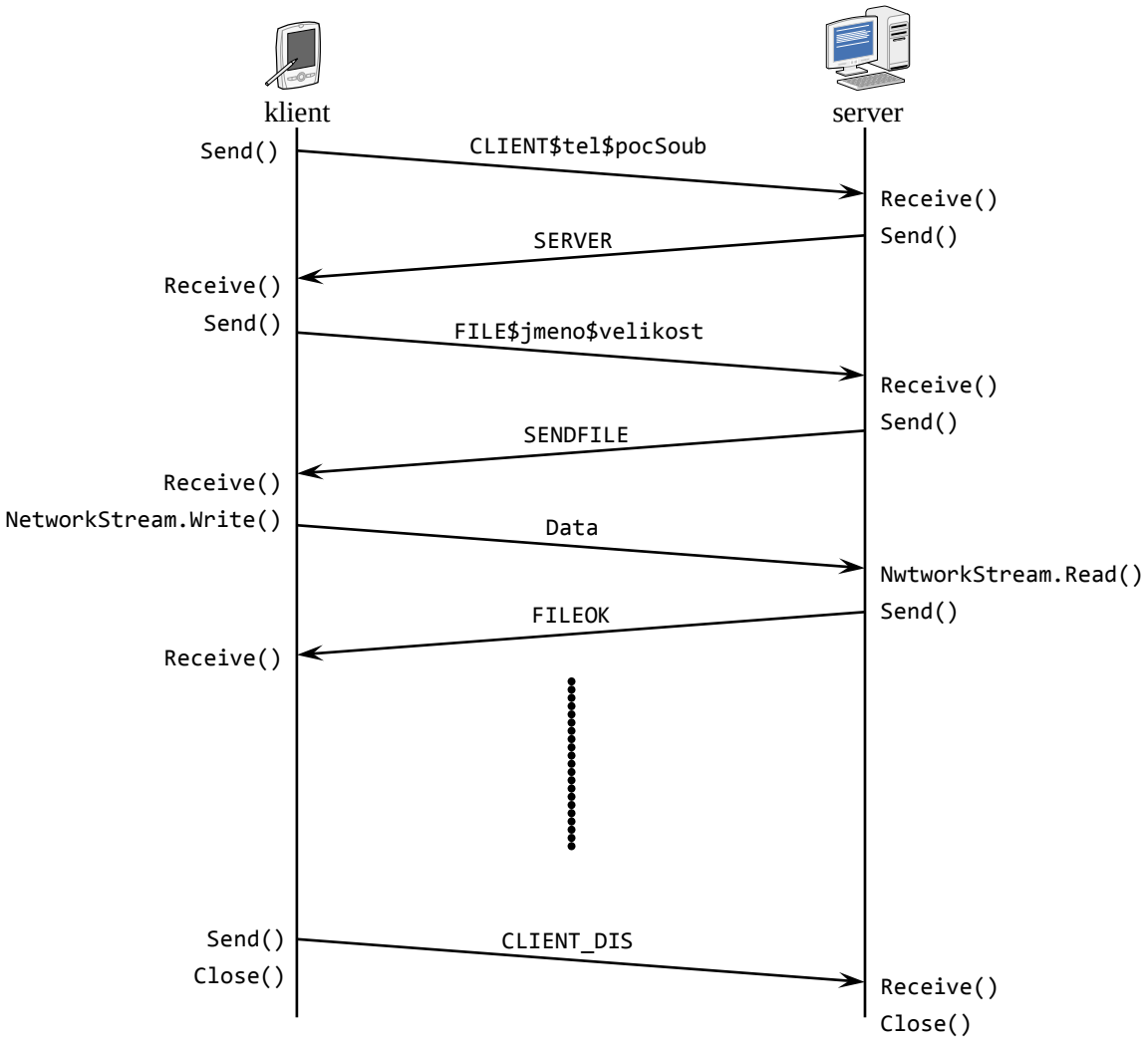
[5] DTD (Document Type Definition) – Definuje použité elementy a atributy dokumentu, určuje vzájemné vztahy mezi elementy.

3.2 Komunikace

Jak již bylo uvedeno výše, komunikovat budeme pomocí protokolu TCP a jako prostředník mezi oběma body poslouží synchronizační program MS ActiveSync (dále AS). Komunikace bude postavena na modelu klient-server, v roli klienta bude mobilní zařízení, PC pak bude představovat server.

Komunikaci navážeme ve chvíli, kdy budou oba body spojeny přes AS. Celá komunikace bude probíhat pomocí zasílání krátkých zpráv, podle typu zprávy bude klient i server postupně procházet stavy, které komunikaci řídí. Bude-li zpráva obsahovat, kromě identifikace stavu, i další informace, budeme je od sebe oddělovat pomocí znaku $.

Klient se přihlásí na server a oznámí mu, že má jeden nebo více souboru k zaslání. Server akceptuje klientův požadavek a příjem zprávy potvrdí. Následně nabídne klient serveru soubor k zaslání, server opět požadavek akceptuje. Klient zasílá soubor serveru, ten ho přijímá a ukládá pro další zpracování. Po příjmu souboru server zašle potvrzení. Tento cyklus se opakuje do chvíle, kdy jsou všechny soubory od klienta odeslány serveru. Komunikace je ukončena odpojením klienta od serveru. Návrh průběhu komunikace je zobrazen na obrázku 3.2.



Obr. 3.2: Průběh komunikace

Podrobný popis průběhu komunikace:

1. Klient odešle serveru uvítací zprávu, obsahující informaci o počtu souborů a identifikaci klienta (telefonní číslo).
   zpráva: `CLIENT$telefonní_číslo$počet_souborů`

2. Server přijatou zprávu zpracuje a odešle svoji uvítací zprávu směrem ke klientovi.
   zpráva: `SERVER`

3. V této chvíli, již máme potvrzeno spojeni z obou stran.

4. Klient odešle zprávu, ve které serveru nabízí soubor. Zpráva obsahuje informaci o jméně a velikosti souboru.
   zpráva: `FILE$jméno_souboru$velikost_souboru`

5. Server potvrdí klientovi, že je schopen přijmout nabízený soubor.
   zpráva: `SENDFILE`

6. Klient v cyklu odesílá soubor směrem k serveru, ten v cyklu soubor přijímá.

7. Server odešle potvrzení, že přijal všechna data.
   zpráva: `FILEOK`

8. Jestliže má klient další soubory k odeslání, opakují se kroky 4 až 7.

9. Klient odešle serveru zprávu, která ukončuje komunikaci. Po odeslání zprávy klient může uzavřít socket. Server uzavře socket, jakmile v pořádku obdrží tuto zprávu od klienta.
   zpráva: `CLIENT_DIS`

3.3 Návrh databáze

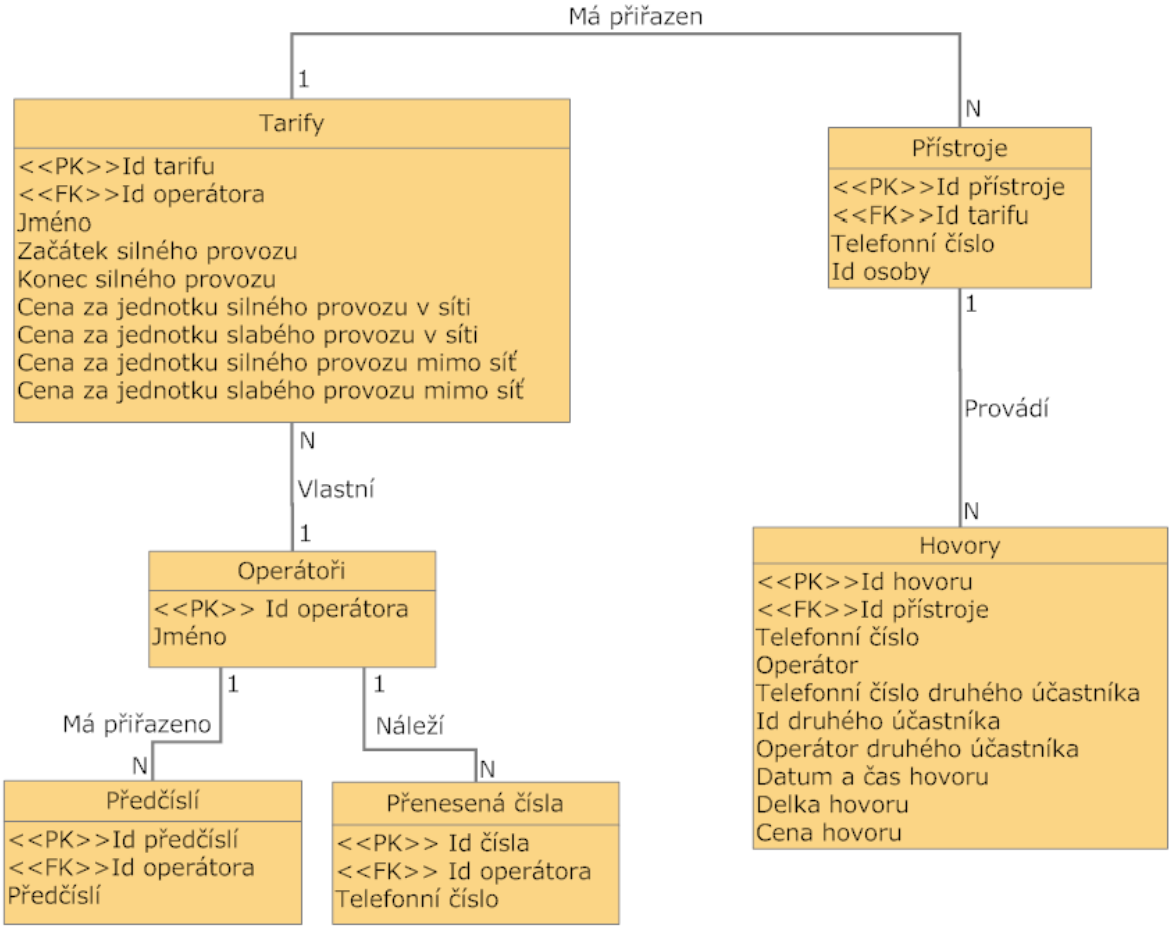
Databáze nám bude sloužit k uchovávání údajů o provedených hovorech, jako jsou: telefonní číslo přístroje, telefonní číslo volaného, datum a čas hovoru, délka trvání, cena hovoru. Obsaženy budou dále informace o zvoleném operátorovi a tarifu, které jsou zapotřebí ke kalkulaci cen hovorů.

V dnešní době existuje velké množství tarifů, přičemž každý disponuje jinými vlastnostmi. Dají se rozdělit do skupin, podle nejčastěji využívaných služeb. Jsou tarify specializované na časté posílání SMS (MMS) zpráv, tarify pro časté volání do vlastní nebo cizí sítě, volání zdarma v rámci skupiny lidí, zvýhodněné volání mezi zaměstnanci firmy, tarify pro častý přístup na internet atd. Pokrytí všech vlastností aktuálně nabízených tarifů by do databáze vnášelo redundanci, neboť nově nadefinovaná vlastnost může být v krajním případě specifickou vlastností pouze jediného tarifu. Pro naše účely použijeme standardní přístup k tarifikaci, který rozlišuje pouze volání do vlastní, nebo cizí sítě, dále rozdělené na tzv. silný a slabý provoz, kdy je den rozdělen na dvě části, v nichž jsou ceny hovorů rozdílné. Zpravidla jsou hovory v nočních a brzkých ranních hodinách (slabý provoz) výrazně levnější, než v pracovní době (silný provoz), kdy bývají sítě vytíženější. Tento model je znám také jako volání ve špičce a mimo špičku. Při analýze projektu vycházíme z předpokladu, že klíčovou informací pro uživatele systému je cena za uskutečněné hovory. Databáze tedy bude obsahovat údaje

pouze o odchozích hovorech, uskutečněných na daném zařízení. Na zařízení budou logovány veškeré hovory, při zpracování údajů však hovory označené jako příchozí vynecháme. Umožníme tím budoucí rozšíření systému o kompletní přehled všech uskutečněných hovorů. Výsledný návrh databáze znázorňuje ER diagram (viz. Obr. 3.3).

Popis entit ER diagramu

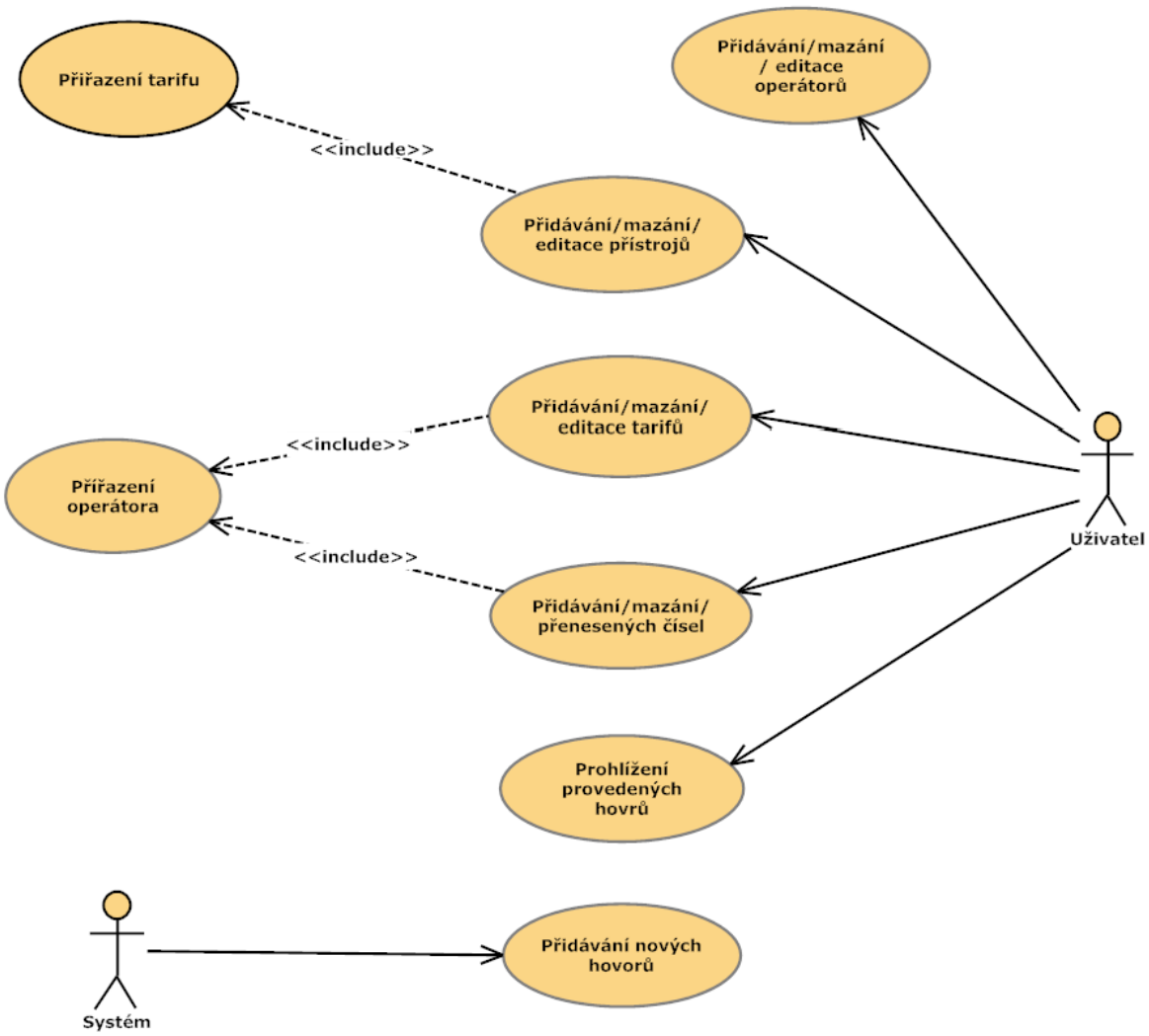
Operátoři	Tato entita obsahuje jména evidovaných operátorů a jejich identifikační číslo, kterým jsou v systému reprezentováni.
Předčíslí	Údaje o předčíslích operátora. Předčíslí jsou reprezentována třemi číselnými znaky. Každé předčíslí je přiřazeno jednomu konkrétnímu operátorovi z entity Operátoři. Operátor má přiřazeno alespoň jedno ze seznamu předčíslí.
Tarify	Obsahuje informace o tarifu, jenž je přidělen konkrétnímu telefonnímu přístroji (hovoříme-li o přiřazení k přístroji, máme na mysli přiřazení k telefonnímu číslu, resp. k SIM kartě využívané daným přístrojem). Na základě navrženého řešení obsahuje: *Jméno* – Jméno tarifu, nemusí být však vždy shodné se jmény tarifů nabízenými operátory. V databázi budou moci být uloženy i tarify jiné než standardní, půjdou-li je prostřednictvím systému nadefinovat. Tento atribut slouží především ke snadné identifikaci tarifu uživatelem. *Začátek silného provozu, Konec silného provozu* – Tyto dva atributy vymezují dobu silného provozu. V tomto čase je hovor účtován sazbou odpovídající hodnotám atributu *Cena za jednotku silného provozu v síti*, nebo *Cena za jednotku silného provozu mimo síť*. Zároveň vymezuje dobu slabého provozu, která vyplňuje zbylý čas dne mimo silný provoz. Hovor je účtován vždy cenou pásma, v němž byl započat. Pomocí hodnot atributů *Cena za jednotku* bude kalkulována cena hovoru.
Přenesená čísla	Představuje množinu čísel, která byla převedena pod jiného operátora. Přenesením čísla, již nebude možné podle předčíslí rozlišit o jakého operátora se jedná. Entita obsahuje celá telefonní čísla a odkazy na příslušné operátory.
Přístroje	Obsahuje informaci o telefonním čísle přístroje a odkaz na tarif, který přístroj využívá. Atribut *Id osoby* je určen pro možné budoucí rozšíření systému, kde se povede záznam o osobě využívající přístroj.
Hovory	Entita obsahuje veškeré potřebné informace o provedeném hovoru. Na základě informací obsažených ve výše uvedených entitách a údaji o délce hovoru bude systémem vypočtena cena uskutečněného hovoru. Jako jediná bude tato entita sloužit k vytváření statistik o uskutečněných hovorech.



Obr. 3.3: Navržený ER digram

Navržená databáze bude obsahovat tabulky korespondující s výše uvedenými entitami ER diagramu. Pro dodržení referenční integrity databáze bude využito cizích klíčů. Nebude moci dojít k odstranění řádku tabulky, jehož hodnota primárního klíče je obsažena v jiné tabulce jako hodnota klíče cizího.

Činnosti, jenž budou vykonávány nad databází jsou znázorněny na diagramu případu použití (viz. Obr. 3.4). Z uživatelského hlediska není přístup k databázi omezen právy, proto je zaveden pouze jeden druh uživatele. Role Uživatel spravuje všechny údaje, které jsou nutné pro kalkulaci cen hovorů, zároveň si může prohlížet informace o uskutečněných hovorech. Nad databází budou prováděny i úkony nezávislé na uživateli. Jedná se o situaci, kdy budou přijata data z mobilního zařízení, následně aplikací zpracovány a uloženy do databáze. Tato činnost je plně v moci aplikace a uživatel do ní není schopen zasáhnout. Vytvořil jsem roli Systém, která bude vykonávat tuto autonomní činnost. Jednotlivé činnosti jsou zřejmé z diagramu a není zapotřebí hlouběji uvádět jejich význam. Pro upřesnění jen uvedu, že činnost *Přiřazení tarifu* zahrnuta do činnosti *Přidávání/mazání/editace přístrojů* se vztahuje pouze na přidávání a editaci, nikoliv na činnost mazání. Analogicky toto platí i pro činnost *Přiřazení operátora*.



Obr. 3.4: Diagram případů použití

4 Implementace

Pro implementaci aplikace pracující na PC jsem zvolil platformu .NET Framework 3.5, na mobilním zařízení pak kompaktní verzi 2.0. Jako programovací jazyk využijeme C# s podporou vývojového prostředí MS Visual Studio 2008 Professional Edition. Celá aplikace je tedy psána v řízeném (managed) kódu. Následující kapitola je rozdělena do částí, které odpovídají postupu vývoje projektu.

První část se věnuje straně mobilního zařízení, a to oblasti týkající se získávání údajů o hovorech. Jsou zde uvedeny problémy, které se při implementaci vyskytly a jejich řešení v podobě využití knihoven mimo rámec .NET Compact Frameworku. Popsán je způsob vytvoření a editace XML dokumentu pro uchovávání údajů.

V další části této kapitoly se zabývám realizací výměny dat mezi mobilním zařízením a PC. Zaměřuji se především na to, jakým způsobem jsou přidělovány IP adresy koncovým bodům, při komunikaci prostřednictvím programu ActiveSync.

Poslední část je zaměřena na stranu PC. Realizován je návrh SQL databáze a jednotlivé operace nad ní prováděné. Věnujeme se způsobu zpracování informací z XML dokumentu a jejich následné uložení do databáze. Závěrem je popsáno grafické prostředí sloužící k výstupu informací pro uživatele.

4.1 Logování hovorů

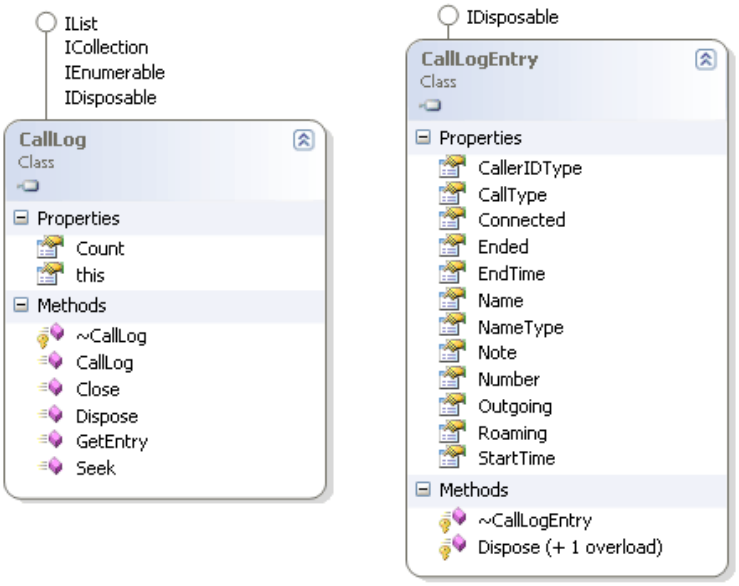
Logování provedených hovorů je rozděleno do dvou kroků. Prvním krokem je rozpoznání probíhajícího hovoru a získání informací o něm. Druhým krokem je uložení informací v požadovaném tvaru do XML dokumentu.

4.1.1 Získání informací o provedeném hovoru

K získání informace o provedeném hovoru, bylo v záměru využít prostředky standardně obsažené v zařízení. SDK pro Windows Mobile obsahuje třídu *SystemState* z jmenného prostoru *Microsoft.WindowsMobile.Status* pomocí níž lze sledovat aktuální stav přístroje. Tato třída obsahuje informace o prováděném hovoru, jako jsou: čas začátku hovoru, číslo volaného, jméno odpovídající číslu v seznamu kontaktů, apod. Problémem tohoto přístupu je fakt, že tato třída neobsahuje metodu, která navrací délku trvání provedeného hovoru. Ani stopování hovoru nelze pomocí standardních metod realizovat. Funkce *Time* pro zjištění aktuálního času přístroje vrací informaci, která je aktualizovaná po minutách, což je pro naše účely nedostačující.

Řešením je využití knihovny, vyvíjené sdružením OpenNETCF Consulting, Smart Device Framework (dále SDF). Knihovna rozšiřuje užitečné vlastnosti jádra .NET CF a je určena pro usnadnění práce vývojářů mobilních aplikací. Ve verzi Community Edition je volně k využití pod Shared Source[6] licencí. Z knihovny SDF využíváme třídy *CallLog* a *CallLogEntry* z jmenného prostoru *OpenNETCF.Phone*. Diagram těchto dvou tříd je znázorněn na obrázku 4.1.

[6] Shared Source licence - licence umožňující volné využití v nekomerčních i komerčních projektech, není dovoleno zasahovat do zdrojových kódu knihovny. Více informací na URL http://www.opennetcf.com/sharedsourcelicense.ocf

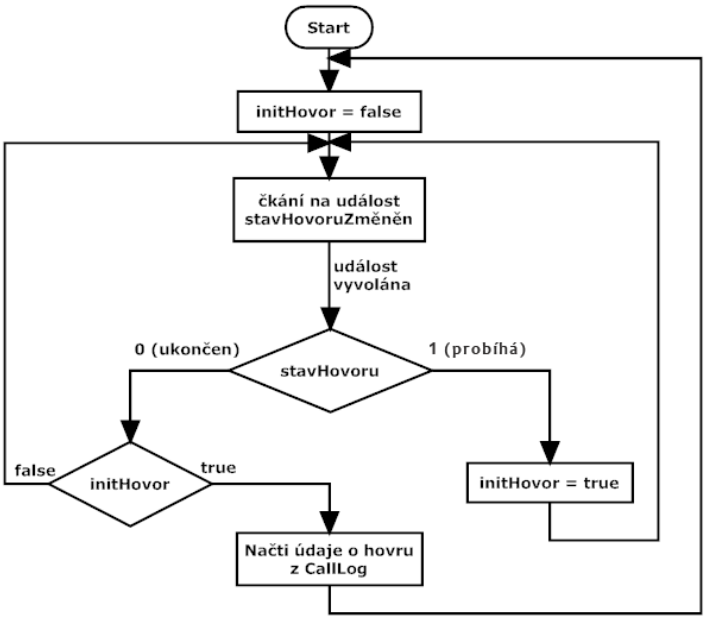


Obr. 4.1: Diagram tříd CallLog a CallLogEtry z knihovny Smart Device Framework

Třída *CallLog* definuje seznam hovorů evidovaných v přístroji, *CallLogEntry* pak jednotlivé položky tohoto seznamu.

Postup načtení informací o hovoru

Při startu aplikace je vytvořena událost reagující na změnu stavu systému. Tato událost je vázána na vlastnost *PhoneCallTalking* (z výše jmenované třídy *SystemState*), která indikuje probíhající hovor. Při navázání hovoru dojde k vyvolání události, hodnota vlastnosti je v tomto případě rovna log. 1. Další vyvolání události indikuje ukončení hovoru a hodnota vlastnosti je změněna na log. 0. Jakmile je hovor ukončen přistoupím pomocí třídy *CallLog* k seznamu hovorů a pomocí metody *GetEntry* načteme naposledy uskutečněný hovor. Takto získaná instance třídy *CallLogEntry* nese veškeré potřebné informace o hovoru (viz. Obr. 4.1 Properties). Vývojový graf znázorňující postup získání informací o provedeném hovoru je na obrázku 4.2.



Obr. 4.2: Vývojový graf vedoucí k odchytu informací o provedeném hovoru

4.1.2 Uložení informací o provedeném hovoru

Aplikace vytváří pro každý den, v němž byl vykonán alespoň jeden telefonní hovor vlastní XML dokument. Není-li uskutečněn žádný hovor dokument se nevytvoří. Jméno dokumentu je odvozeno od aktuálního kalendářního data daného dne. Jakmile jsou dostupné informace o novém hovoru (viz. kapitola 4.1.1), pomocí údaje *StartTime* se vyhledá příslušný XML dokument a zapíší se do něj informace. V případě, že dokument neexistuje, jedná se tedy o první hovor dne, je vytvořen nový dokument a následně jsou zapsána data.

Zpracování dokumentu se provádí prostřednictvím třídy *XmlDocument*, která nabízí programové rozhraní pro XML odpovídající specifikacím DOM[7]. Použití tohoto přístupu vyžaduje před každým zpracováním dokumentu jeho úplné načtení do paměti. U vývoje aplikací určených pro mobilní zařízení je snaha nezatěžovat příliš paměť přístroje. Podle našeho návrhu, kdy se evidují hovory pro každý den do zvláštního dokumentu, je však využití paměti zanedbatelné. Za předpokladu, že běžný uživatel uskuteční deset až sto hovorů denně, bude velikost dokumentu stále v jednotkách kB.

Pří ukládání nového záznamu do XML dokumentu pomocí třídy *XmlDocument* se provádí následující kroky:

1. Nahrání XML dokumentu do paměti.

2. Z dokumentu je načten kořenový element. V našem návrhu se jedná o element *calls*.

3. Vytvoří se nový element včetně atributů a jejich hodnot, který bude reprezentovat vkládaný záznam. Dle návrhu element *cl* s atributy: *nbr*, *dt*, *typ*, *dur*.

4. Nově vytvořený element je přiřazen jako synovský ke kořenovému elementu dokumentu.

5. Upravený dokument je uložen.

4.2 Komunikace

Podle návrhu jsem realizoval komunikaci typu klient-server prostřednictvím synchronizační aplikace MS ActiveSync. Využívám blokujícího TCP spojení, server (klient) je tedy zablokován na metodě příjmu, dokud nedorazí data od klienta (serveru).

Nevýhodou .NET CF oproti standardní verzi Frameworu je absence metody *Socket.SendFile*, která umožňuje odeslání souboru. Tato metoda se po zadání cesty k souboru postará o jeho odeslání. Naše implementace řeší odesílání souboru pomocí *NetworkStreamu*, kde jsou v cyklu odesílána data ve vhodně zvolených dávkách, na straně PC jsou pak data opět v cyklu přijímána a zapisována do souboru.

Při využití aplikace AS nelze nastavit libovolné IP adresy serveru a klientovi. AS má pevně definované IP adresy, na kterých zařízení a PC komunikují. V následující kapitole je popsáno, jakým způsobem informace o IP adresách získáváme.

[7] DOM(Document Object Model) – Objektově orientovaná reprezentace XML, HTML dokumentu.

4.2.1 Přiřazení IP adres

Po připojení zařízení pomocí USB kabelu se ve Windows automaticky vytvoří nové síťové připojení pojmenované *Windows Mobile-based Device #n*. Pomocí této sítě lze navázat komunikaci, jak ze strany zařízení, tak i ze strany počítače. Při spojení pomocí Bluetooth je situace zcela jiná. V síťových připojeních se nevytvoří nová síť a nelze tedy navázat spojení ze strany počítače. Komunikaci ze strany zařízení je možné navázat právě pomocí AS. IP adresy obou bodů spojení jsou pevně dány následujícím způsobem:

USB komunikace:	Bluetooth komunikace:
IP zařízení: 169.254.2.1	IP zařízení: 192.168.55.101
IP počítače: 169.254.2.2	IP počítače: 192.168.55.100

Jakmile je zařízení spojeno s počítačem pomocí AS, má přidělenou svoji IP adresu. Zařízení vlastní také informaci o adrese, která byla přidělena druhému bodu komunikace (počítač). Tato adresa je uvedena jako hodnota klíče v registrech systému. Klíče jsou umístěny v *HKEY_LOCAL_MACHINE\Comm\Tcpip\Hosts*, jejich jména, názvy hodnot a hodnoty jsou:

USB komunikace:
- jméno klíče: dtpt_peer
- jméno hodnoty: ippaddr
- údaj hodnoty: A9 FE 02 02 $(169.254.2.2)_{10}$

Bluetooth komunikace:
- jméno klíče: ppp_peer
- jméno hodnoty: ippaddr
- údaj hodnoty: C0 A8 37 64 $(192.168.55.100)_{10}$

Pomocí následující části kódu pak zjistíme, na jaké IP adrese bude komunikace probíhat.

```
IPHostEntry ipHostServer = Dns.GetHostEntry("PPP_PEER";);
IPAddress ipAdressServer = ipHostServer.AddressList[0];
```

Kód 4.1: Načtení IP adresy počítače pomocí fixního jména

4.2.2 Běh serveru

Server se spouští automaticky při načítání formuláře aplikace, která má dvě základní funkce. První funkcí je komunikace s databází, vkládání údajů, dotazování a zobrazování výsledků dotazů. Druhou funkcí je příjem dat od klienta. Jelikož je použita blokující komunikace, musí server běžet ve vlastním vlákně, jeho činnost tak neovlivňuje uživatelovu práci s formulářem.

4.2.3 Běh klienta

Jak bylo uvedeno výše, pokus o navázání komunikace se serverem začíná, jakmile je aktivní aplikace ActiveSync. K tomu slouží událost navázaná na změnu stavu systému

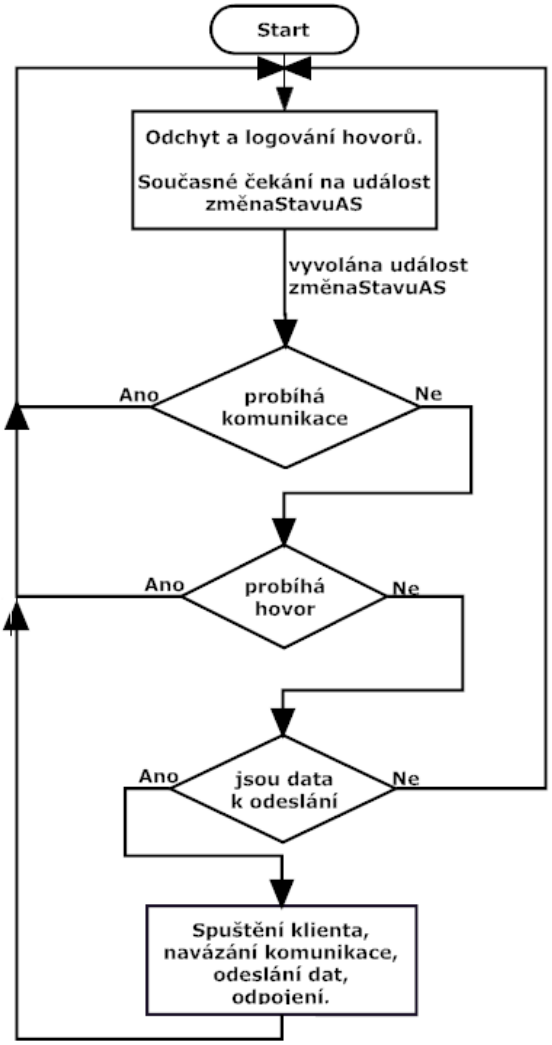
SystemProperty.ActiveSyncStatus. Událost však není vyvolána pouze při spuštění AS, ale pokaždé, když se spustí nějaký synchronizační cyklus. Tento jev je nepříznivý, neboť by docházelo o pokusy opětovného navázání komunikace se serverem, mnohdy i za právě probíhající komunikace. Následují podmínky, které musejí být splněny, aby bylo zamezeno možným konfliktům při komunikaci a sdílení dat.

1. Nesmí probíhat komunikace se serverem. Při předchozí komunikaci, jsou odeslána veškerá data, proto nemá smysl sestavovat nové spojení.

2. Nesmí probíhat telefonní hovor. Jakmile se XML soubor v pořádku doručí serveru, je smazán. Současně může docházet k zapisování informací o právě odchyceném hovoru a tím by vznikl konflikt sdílení dat.

3. Musí existovat nejméně jeden soubor s alespoň jedním záznamem určený k odeslání.

Kroky vedoucí k spuštění klienta jsou znázorněny na následujícím obrázku (Obr. 4.3).



Obr. 4.3: Vývojový diagram - kroky vedoucí ke spuštění klienta

4.3 Databáze uchovávající údaje o hovorech

Databáze je navržena pro Microsoft SQL Server 2005. Celá práce s databází je postavena na technologii LINQ, které jsem se věnoval v kapitole 2.4. Pro naše účely využíváme komponenty LINQ to XML ke zpracování XML dokumentu a LINQ to SQL pro vkládání a editaci údajů tabulek databáze. Při realizaci databáze jsem vycházel z návrhu, vytvořené tabulky jsou tedy téměř shodné s entitami navrženého ER diagramu. Byla doplněna pouze tabulka s údaji o hovorech (tbHovory), kde byl přidán sloupec obsahující délku provedeného hovoru v sekundách. Tento způsob uložení je vhodnější z hlediska porovnávání, protože celá čísla se porovnávají lépe než řetězce. V nové verzi SQL (2008) již je implementován datový typ *Time*, který by byl pro ukládání času nejvhodnější.

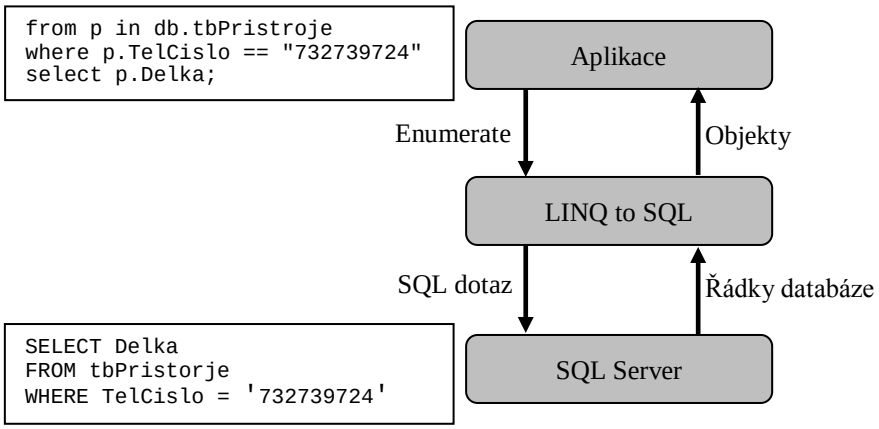
V této kapitole je popsán způsob přípravy aplikace pro práci s technologií LINQ, postup zpracování XML dokumentu a práce s databází pomocí dotazů integrovaných do jazyka.

4.3.1 Vytvoření mezivrstvy datových tříd pro LINQ

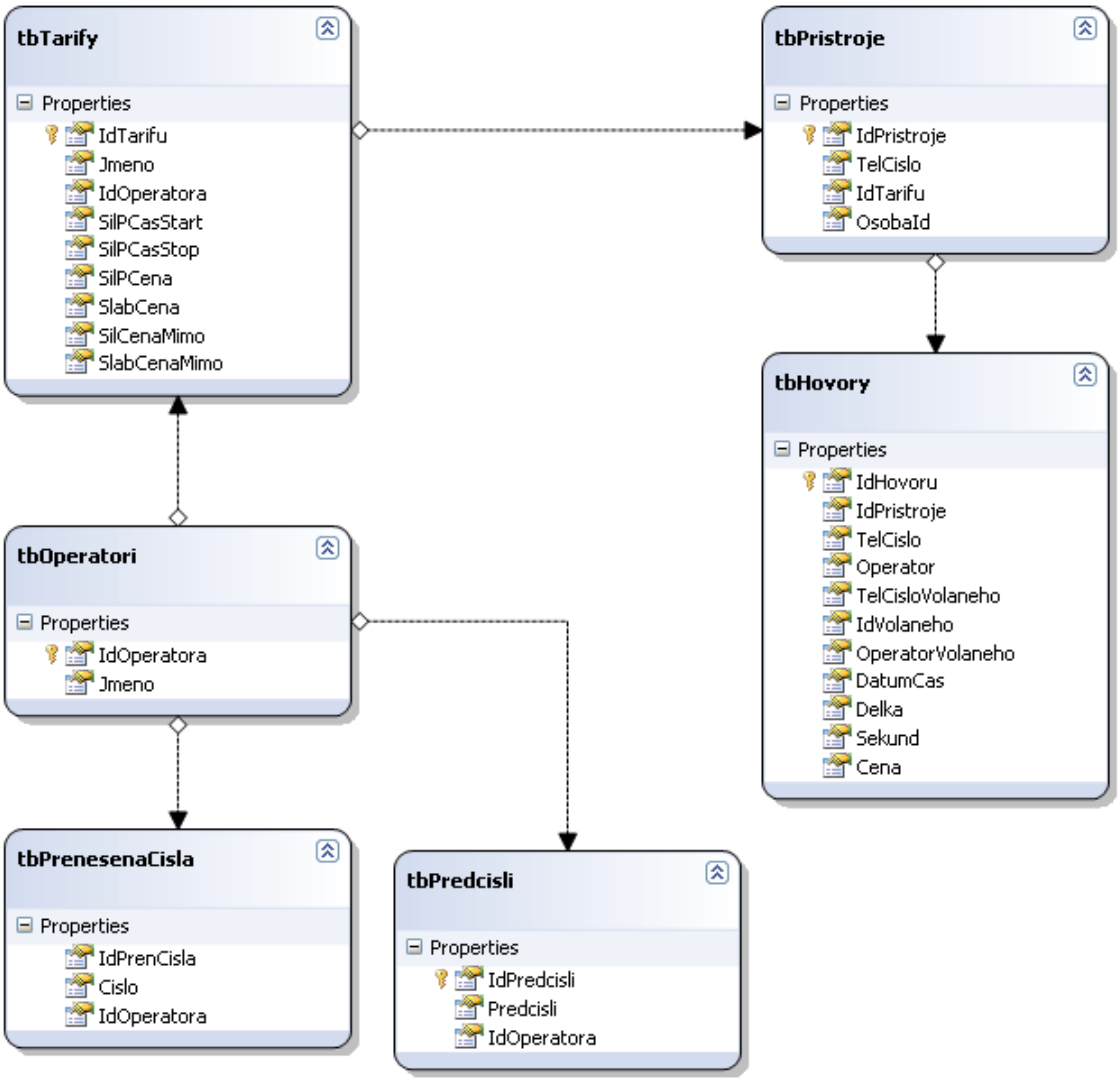
Činnost komponenty LINQ to SQL je rozdělena do tří vrstev: vrstva Aplikace, LINQ to SQL vrstva a vrstva SQL Serveru. V aplikační vrstvě píšeme dotazy databáze v syntaxi jazyka C#, prostřednictvím vrstvy LINQ to SQL jsou tyto dotazy přeloženy do standardního jazyka SQL a odeslány vrstvě SQL serveru. Výsledek dotazu v podobě řádků tabulky je předán z vrstvy SQL serveru vrstvě LINQ to SQL, zde jsou řádky převedeny na objekty a předány vrstvě aplikace. Vrstvy s příklady kódu jsou znázorněny na obrázku 4.4.

Vytvoření mezivrstvy datových tříd, která slouží jako prostředník mezi aplikační vrstvou a vrstvou databáze je v prostředí Visual Studio 2008 automatizováno pomocí nástroje ORM (Object Relational Mapping) designer. Do prostředí ORM designeru přeneseme tabulky z naší databáze, designer automaticky generuje třídy navázané na tyto tabulky. Ke každému atributu tabulky jsou vygenerovány příslušné metody reprezentující tento atribut, generovány jsou také vztahy primárních a cizích klíčů. Instance vygenerovaných tříd pak slouží k reprezentaci dat z databáze.

Designer automaticky generuje i tzv. Datový kontext (DataContext), který se v LINQ stará o zpřístupnění tříd odpovídající tabulkám SQL serveru. Umožňuje zadávání výběrových podmínek a sleduje změny v datech, které později odešle ve správném formátu SQL serveru. Třída datového kontextu v rámci naší implementace je pojmenována *LinqSql.cs*. Na obrázku 4.5 je pohled na ORM designer při realizaci návrhu databáze.



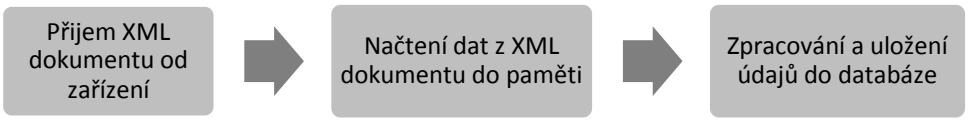
Obr. 4.4: Činnost komponenty LINQ to SQL



Obr. 4.5: Přenesené tabulky databáze do prostředí ORM designeru

4.3.2 Uložení údajů z XML do databáze

Proces uložení údajů o provedených hovorech, můžeme rozdělit do částí podle diagramu na obrázku 4.6. Přičemž prvním blokem diagramu jsem se zabýval v kapitole 4.1.2.



Obr. 4.6: Rozdělení procesu zpracování XML dokumentu

4.3.2.1 Načtení dat z XML dokumentu

Pro načítání dat z XML dokumentu je využita technologie LINQ, a to konkrétně komponenta LINQ to SQL. Jmenný prostor *Systém.XML.LINQ* poskytuje funkcionalitu pro přístup k dokumentům XML pomocí technologie LINQ. Jedná se o plně vybavené rozhraní pro jazyk XML s několika klíčovými funkcemi z jazyků XPath[8] a XSLT[9]. Integrované dotazy pro XML jazyk poskytují prostředky pro úpravu dokumentů XML a stromů s elementy v paměti počítače, stejně tak i prostředky pro zpracování datového proudu [20].

V prvním kroku dojde k načtení XML dokumentu do paměti, následně je sestaven výraz, pomocí něhož jsou načteny jednotlivé elementy dokumentu, výsledkem načítání je instance třídy *Enumerate*. Celý proces probíhá tak, že se prochází všechny elementy, které jsou potomky kořenového elementu, současně se načítají hodnoty jednotlivých atributů elementů. Následující úsek kódu (Kód 4.2) demonstruje načítání XML dokumentu do instance třídy *Enumerate* při implementaci naší aplikace. Použité názvy elementu a atributů byly definovány pomocí DTD v návrhu (viz. kapitola 3.1).

```
// načtení XML dokumentu
var xmlDokument = XElement.Load(xmlfile);

// pomoci LINQ pro xml načteme z dokumentu potřebná data
// hovory – výsledek načíání typu Enumerable
var hovory = from hovor in xmlDokument.Descendants("cl")
             // Descendants - zvolime kazdeho potomka
             // kořenového elementu - "cl"

             select new // vytvoření nového hovoru
             {   // nacteme spravne pretypovane hodnoty atributu
                 cislo = (string)hovor.Attribute("nbr"),
                 dateTime = (string)hovor.Attribute("dt"),
                 typ = (string)hovor.Attribute("typ"),
                 duration = (string)hovor.Attribute("dur"),
             };
```

Kód 4.2: Načtení XML dokumentu pomocí LINQ to XML

4.3.2.2 Zpracování a uložení dat do databáze

Jakmile jsou údaje z XML dokumentu načteny, nastává fáze jejich zpracování a uložení do databáze. Zpracováním máme na mysli výpočet ceny hovoru. Po operaci načítání dokumentu máme k dispozici instanci třídy *Enumerate*, ve které jsou všechny informace z dokumentu. Pomocí metody *ForEach* je postupně procházen vytvořený objekt, jednotlivé záznamy jsou zpracovány a uloženy do databáze. Tento proces probíhá v následujících krocích:

1. Ke zpracování jsou předány pouze odchozí hovory. Hodnota atributu *typ* musí být rovna „*Outgoing*“

[8] XPath (XML Path Language) – jazyk pro adresování částí dokumentu XML.
[9] XSLT (eXtensible Stylesheet Language Transformations) – jazyk pro převod zdrojových dat ve formátu XML do libovolného jiného požadovaného formátu, nejčastěji HTML.

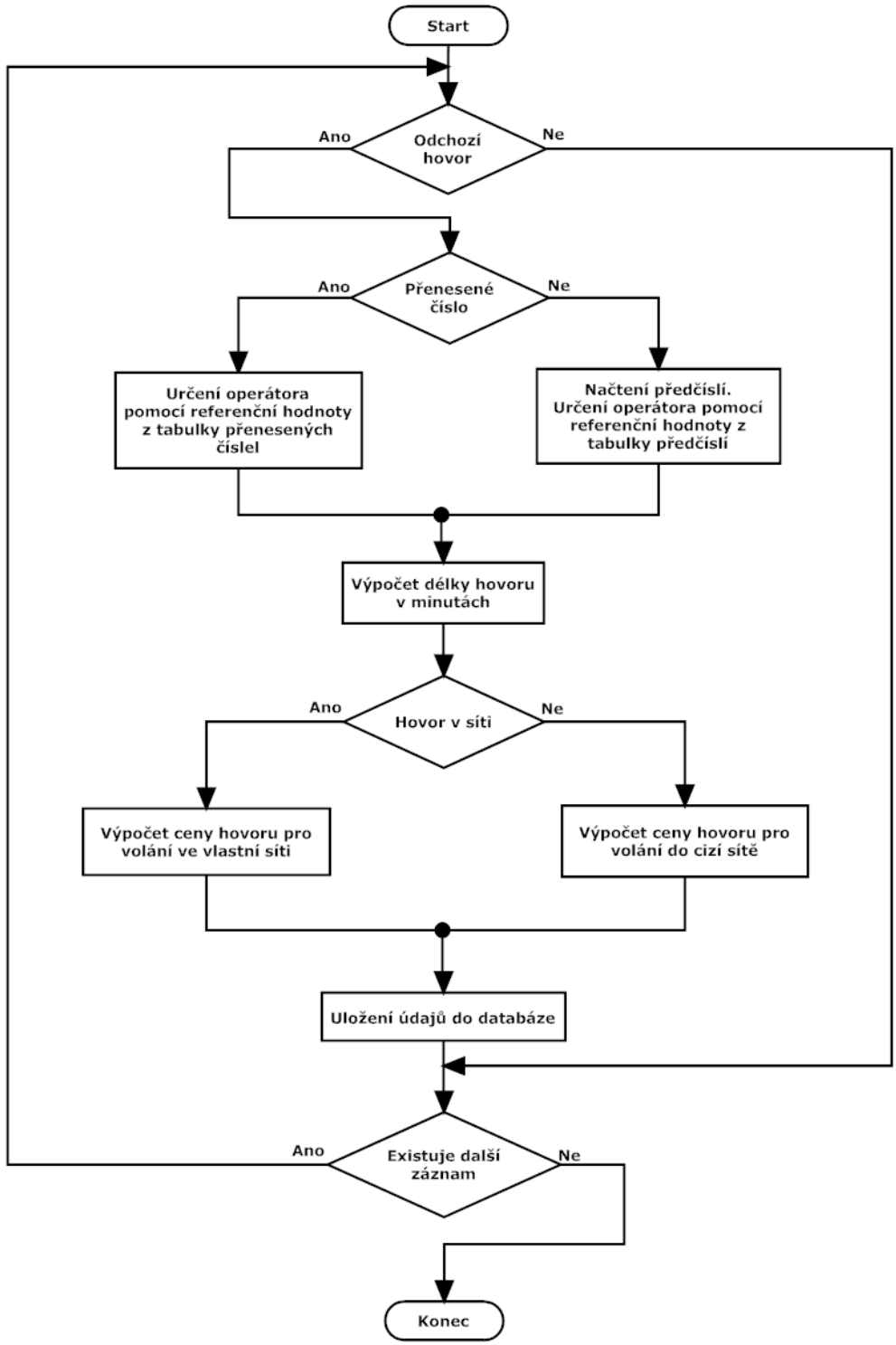
2. Pomocí dotazu je ověřeno, zda je volané telefonní číslo v seznamu přenesených čísel, tj. je obsaženo v tabulce přenesená čísla. Jestliže ano, pomocí id operátora je načten odpovídající operátor z tabulky operátorů. V opačném případě se určí pomocí čísla volaného, do jaké sítě byl hovor proveden. Prostřednictvím předčíslí volaného čísla je vyhledán odpovídající údaj v tabulce předčíslí operátorů, kde s každou položkou předčíslí koresponduje id operátora v tabulce operátorů.

3. Ze záznamu je načtena délka hovoru a převedena na počet minut. V rámci implementace předpokládám, že se účtuje každá započatá minuta hovoru. Ve skutečnosti někteří operátoři účtují hovory po sekundách, každý operátor však provádí účtování po svém. Někteří z operátorů účtují hovor po sekundách ihned od navázání hovoru, jiní zase po určitém počtu minut probíhajícího hovoru. V případě, že by všichni operátoři účtovali celý hovor po sekundách, lze použít k výpočtu ceny délku hovoru v sekundách. Tento údaje je také obsažen v databázi.

4. Porovnán je čas uskutečnění hovoru s údaji o časovém rozmezí, která určují, zda byl hovor proveden v silném, nebo slabém provozu. Je-li hovor uskutečněn přes hranici, která odděluje silný a slabý provoz, je účtován podle sazby odpovídajícího úseku, ve kterém byl hovor navázán.

5. Na základě výše zjištěných informací je vypočtena cena uskutečněného hovoru. Pomocí LINQ je sestaven dotaz pro přidání řádku do tabulky hovorů. Sestavený dotaz se odešle serveru, následuje zpracování dalších záznamů.

Vývojový diagram na obrázku 4.7 znázorňuje postup zpracování záznamů a následné uložení nově vytvořeného řádku do tabulky databáze.



Obr. 4.7: Postup výpočtu ceny hovoru a uložení údajů do databáze

4.3.3 Dotazy prováděné nad databází

V této části jsou podrobněji popsány dotazy prováděné nad databází. Dotazy jsou využity jednak pro vkládání, mazání a update řádků tabulky. Dalším využitím je získání informací o provedených hovorech za určité období. Pro lepší pochopení nejprve objasním dvě v příkladech často používané techniky.

Odvozené typy lokálních proměnných
Pro usnadnění práce programátorům je v nové verzi C# (3.0) zavedeno klíčové slovo *var*, které zapříčiní odvození typu lokální proměnné v době kompilace. Deklarujeme-li tedy pomocí tohoto klíčového slova proměnnou, do níž uložíme řetězec, bude proměnná v době překladu automaticky převedena na typ *string*.

Lambda výrazy
Pomocí lambda výrazů je možné tvořit anonymní metody, které obsahují jeden výraz nebo několik příkazů a použít je kdekoli, kde je očekávána instance delegáta. Je zapisován pomocí šipky =>, před šipkou je argument nebo seznam argumentů v závorkách, za šipkou je pak výraz specifický pro použitou metodu.

Pro přístup k databázi pomocí LINQ je nutné nadeklarovat datový kontext, jenž byl vytvořen pomocí OMR designeru v kapitole 4.3.1.

```
LinqSqlDataContext dataContext = new LinqSqlDataContext();
```

INSERT (vložení nového řádku do tabulky) dotazy

1. Vytvoření nového řádku tabulky a jeho naplnění daty.
2. Datový kontext se nastaví na přidávání prvků.
3. Pomocí datového kontextu jsou uloženy nové řádky do tabulky databáze.

Následující úsek kódu provede vložení nového řádku do tabulky tbTarify.

```
var qryInsert = new tbTarify    // vytvoření nového řádku tabulky
{
     Jmeno = "O2 NEON L",       // naplnění všech potřebných údajů
     SilCenaMimo = 5.4
      // ...
};
// nastavíme datový kontext pro přidávání
dataContext.tbTarifies.InsertOnSubmit(qry_kontrola);
dataContext.SubmitChanges(); // provedeme změny v datovém kontextu
```

SELECT dotazy
V dotazu se pracuje s tabulkou z datového kontextu, pro vymezení určitých dat se využívá klauzule where. Následující část kódu vybere z tabulky tbHovory záznamy provedené přístrojem s určitým id, současně je výběr omezen na určité časové období.

```
// sestavení dotazu pomocí datového kontextu
var qrySel = from hovor in dataContext.tbHovories
             where hovor.IdPristroje == 5  // výběr pomocí indexu
             where hovor.DatumCas > dtOd   // výběr pomocí data
             where hovor.DatumCas < dtDo   // dtOd, dtDo - DateTime
             select hovor;                 // provedení výběru
```

Výběrové dotazy lze dále rozšiřovat pomocí extension metod, při implementaci jsou využity nejčastěji *Single<>* a *FirstOrDefault<>*, které vybírají pouze jeden, nebo první prvek z výsledku dotazu. Další výhodou dotazování se pomocí LINQ, je možnost provádět nad výsledky dotazů další dotazování. Tímto způsobem selektuji z výše uvedeného dotazu hovory, které byly provedeny v rámci vlastní sítě a hovory provedené do sítí cizích. Za povšimnutí stojí pojmenování tabulky tbHovory jako tbHovories, což je dáno tím, že v datovém kontextu jsou tabulky reprezentovány jako entity (anglicky entities).

UPDATE dotazy se provádí na základě výsledku dotazu select, vybraný prvek je upraven a změny v datovém kontextu jsou potvrzeny pomocí metody *SubmitChanges*.

DELETE dotazy se opět aplikují na výsledky dotazu select, datový kontext je nastaven tak, aby po odeslání změn byly výsledky dotazu select odstraněny. Pro odstranění záznamů vybraných v předcházející části kódu se využije následující sekvence příkazů.

```
dataContext.tbHovories.DeleteOnSubmit(qrySel);   // metoda delete
dataContext.SubmitChanges();                     // odeslání dotazu
```

Jsou-li odstraňovány záznamy z tabulky, ve které jsou řádky v relaci primární-cizí klíč s řádky jiné tabulky a záznam vedený jako primární klíč má již vytvořen odkaz v druhé tabulce v podobě cizího klíče, je vyvolána výjimka typu konflikt referencí. Jestliže je při pokusu o odstranění záznamů z tabulky tbTarify vyvolána výjímka, obsluha pak spočívá v upozornění uživatele, že chce odstranit záznam, na nějž je odkazováno v jiných tabulkách (tbPřístoje) a odstranění je zamítnuto. Pokud je mazán záznam z tabulky tbPřístroje a dojde k vyvolání výjimky, obsluha uživateli oznámí, že odstraňuje položku, ke které již existují záznamy v tabulce tbHovory. Uživatel si zvolí, zda chce v odstraňování pokračovat. V tomto případě budou nejprve odstraněny všechny záznamy, které korespondují s odstraňovaným záznamem, z tabulky tbHovory a následně samotná položka z tabulky tbPřístroje. Další možností je mazání stornovat a položku ponechat.

Aby měl uživatel lepší představu o nákladech, které byly na telefonování vynaloženy za určité období, jsou vypočteny následující údaje:

- Délka uskutečněných hovorů
- Délka uskutečněných hovorů ve vlastní síti
- Délka uskutečněných hovorů do cizích sítí
- Celková cena uskutečněných hovorů
- Celková cena uskutečněných hovorů ve vlastní sít
- Celková cena uskutečněných hovorů do cizích sítí
- Nejdelší uskutečněný hovor
- Nejdražší uskutečněný hovor
- Celkový počet hovorů

K výpočtu těchto hodnot jsou využity extension metody aplikované na výsledky dotazů. K vyjádření požadované veličiny pak slouží lambda výrazy. Před výpočtem se ověří, zda výsledek dotazu není prázdný. V tomto případě by hodnota výsledku byla rovna nule, v opačném případě je navrácena požadovaná informace. Pracujeme-li stále s dotazem *qrySel*, který navrátil uskutečněné hovory na konkrétním přístroji za určité období, pak se pomocí následujících dvou podmíněných výrazů vypočítá celková cenu hovoru a nalezne se nejdelší uskutečněný hovor.

```
// celková cena ukutečněných hovorů
//            kontrola neprázdnosti    výpočet sumy z cen hovorů
decimal cena = (qrySel.Count() > 0) ? qrySel.Sum(x => x.Cena) : 0;


// nejdelší uskutečněný hovor v sekundách
//           kontrola neprázdnosti  nalezení maxima z délek hovorů
int delkaS = (qrySel.Count() > 0) ? qrySel.Max(x => x.Sekund) : 0;
```

5 Práce s aplikací

Na mobilním zařízení je práce aplikace nezávislá na uživateli. Uvádím tedy jen způsob spuštění aplikace a její uvedení na pozadí. Aplikace na straně PC je obsluhována uživatelem, proto se zde zabývám vzhledem grafického rozhraní a způsobem práce s aplikací.

5.1 Aplikace na Mobilním zařízení

Při implementaci návrhu aplikace pro mobilní zařízení jsem narazil na poměrně závažný problém, týkající se běhu aplikace na pozadí. V tomto ohledu je programování pomocí řízeného kódu slabší než nativní aplikace. Jednou z možných variant je vytvořit konzolovou aplikaci, ta ovšem vyžaduje pro běh neustálé cyklení, což by zařízení téměř zablokovalo. Implementoval jsem tedy aplikaci formulářového typu. Formulářové aplikace po spuštění nevytvářejí žádnou činnost, dokud není vyvolána nějaká událost formuláře, která může být interakcí na uživatelovu činnost nebo na změnu vnitřního stavu systému. Z našeho hlediska je problémem formulářových aplikací jejich viditelnost a možnost ukončení, tento problém řeším pomocí metody *hide*. Po spuštění se objeví formulář s informacemi o aplikaci a tlačítkem pro skrytí formuláře. Stisknutím tlačítka se formulář skryje a aplikaci již není možné běžným způsobem ukončit, běh aplikace je ukončen až po restartu zařízení. Nepodařilo se však zajistit, aby se aplikace automaticky spustila při startu zařízení. Toto je prozatím ponecháno na uživateli. Na následujícím obrázku je náhled na okno formuláře po spuštění aplikace (Obr. 5.1).

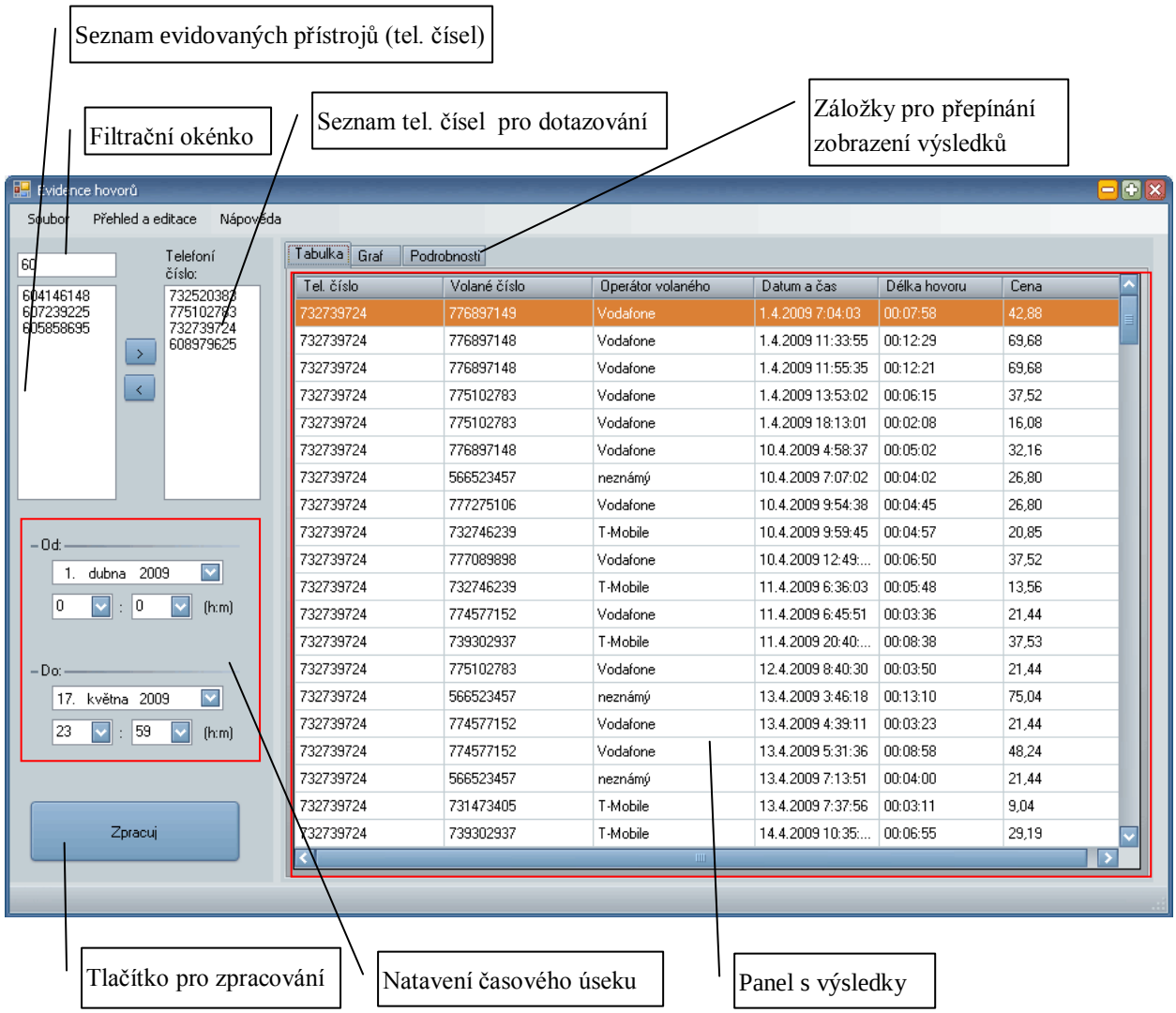
Obr. 5.1: Vzhled obrazovky mobilního zařízení po spuštění aplikace

5.2 Aplikace na PC

Po spuštění aplikace pro evidenci hovorů se zobrazí hlavní formulářové okno. V levé části okna je panel, ve kterém se nacházejí prvky pro nastavení dotazování. Pravá část okna slouží k zobrazování výsledků dotazů. Pomocí položek v menu Přehled a editace přistupujeme ke správě následujících záznamů:

- Evidované přístroje – přehled a editace evidovaných přístrojů
- Evidované tarify – přehled a editace evidovaných tarifů
- Přenesená čísla – přehled a editace telefonních čísel, která byla přenesena k jinému operátorovi
- Operátoři – přehled a editace evidovaných operátorů

Na obrázku 5.2 je zobrazeno hlavní okno aplikace s popisy nejdůležitějších částí.



Obr. 5.2: Vzhled hlavního okna aplikace pro evidenci hovorů

Postup práce

1. Ze seznamu evidovaných telefonních čísel vybereme čísla (minimálně jedno), s nimiž chceme pracovat. Výběr probíhá přenesením čísel pomocí šipek do seznamu čísel. K vyhledání určitého čísla můžeme využít filtrační okénko.

2. Pomocí volby data a času nastavíme rozmezí, ze kterého chceme získat informace.

3. Tlačítkem *Zpracuj* potvrdíme volby.

Zobrazované výsledky

- Tabulka – v tabulce jsou zobrazeny veškeré záznamy, které byly vybrány provedeným dotazem. Výpis tabulky je patrný na obrázku 5.2.

- Graf – pomocí grafu zobrazujeme pro lepší přehled zejména údaje o ceně a délce hovorů. K dispozici jsou dva druhy grafu
    - Graf časových údajů o hovorech
        - Celková délka hovorů
        - Délka hovorů v síti
        - Délka hovorů mimo síť
    - Graf cenových údajů o hovorech
        - Celková cena hovorů
        - Cena hovorů v síti
        - Cena hovoru mimo síť

- Podrobnosti – ke každému číslu ze seznamu vybraných čísel zobrazujeme nejdůležitější podrobnosti. Pro zvýšení přehlednosti je možné si zvolit, které údaje budou zobrazovány, k tomu slouží seznam zatrhávacích polí. Podrobnosti obsahují informace:
    - Operátor
    - Tarif
    - Počet hovorů
    - Délka hovorů
    - Délka hovorů v síti
    - Délka hovorů mimo síť
    - Nejdelší hovor
    - Cena hovorů
    - Cena hovorů v síti
    - Cena hovorů mimo síť
    - Nejdražší hovor

Spojení s mobilním zařízením, získání nových dat o hovorech

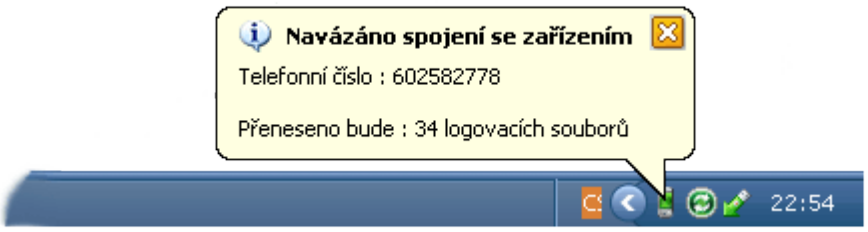
Po propojení mobilního zařízení a PC pomocí aplikace ActiveSync dojde automaticky k přenosu záznamů a jejich uložení do databáze. Celý proces je složen z následujících kroků.

1. Uživatel spustí na mobilním zařízení aplikaci ActiveSync a naváže spojení s počítačem. Spojení zařízení a PC pomocí USB kabelu je standardně rozpoznáno systémem a synchronizační aplikace se tedy spouští automaticky. Bluetooth připojení je zapotřebí iniciovat ze strany zařízení, z nabídky programu ActiveSync proto vybereme *Připojení pomocí Bluetooth*. Jestliže není zařízení s počítačem v synchronizační relaci, je nutné ji před prvním spojením vytvořit. Postup sestavení relace bývá obsažen v manuálech zařízení, která jsou dodávaná s operačním systémem WM.

2. Data jsou přenesena ze zařízení do počítače, tato událost je indikována zobrazením balloon tipu[10] od ikony aplikace v oznamovací oblasti hlavního panelu. Příklad zobrazení tipu je znázorněn na obrázku 5.3.

3. V případě, že zařízení není doposud evidováno v databázi, aplikace se uživatele dotáže, zda chce nový přístroj zaevidovat. Jestliže uživatel přístroj zaeviduje, jsou soubory zpracovány a záznamy o hovorech uloženy do databáze. Ukončení této události je opět doprovázeno zobrazením balloon tipu s informací o počtu uložených záznamů. Nezaeviduje-li uživatel nový přístroj, dojde k odstranění přenesených souborů.

4. S přenesenými informacemi může uživatel okamžitě pracovat.



Obr. 5.3: Zobrazení balloon tipu oznamujícího navázání spojení se zařízením

[10] Balloon tipu využíváme proto, že informace o připojení je vlastněna vláknem, ve kterém běží server. Pro zobrazování této informace přímo do formuláře, je nutné používat konstrukce Invokerequired a BeginInvoke. Jelikož se jedná o informativní zprávu, referující o automatické činnosti, je zobrazení balloon tipu dostačující.

6 Testování systému

Testování aplikací probíhalo postupně s vývojem, neboť klientská aplikace byla laděna přímo v zařízení. Použité mobilní zařízení, bylo PDA HTC TyTN II (viz. příloha) pracující na operačním systému Windows Mobile Professional 6.0. Aplikace pracující na PC je navržena a vyvíjena pro MS Windows XP, s využitím ActiveSync 4.5.0.

Komunikace mezi zařízením a počítačem probíhala bez větších problémů. Spojení pomocí USB kabelu bylo znatelně rychlejší než bezdrátové Bluetooth, to je dáno především menší přenosovou rychlostí a párováním zařízení. Když hovoříme o rychlosti přenosu, máme na mysli dobu, která uplynula od chvíle, kdy byl oznámen příjem dat až do potvrzení uložení do databáze (časový usek mezi zobrazením informačních balloon tipů). Rychlost byla rovněž mírně odlišná ve chvílích, kdy se prostřednictvím synchronizace přenášelo větší množství dat. Při vytvoření prázdné synchronizační relace, kdy jsou sice zařízení spojena, ale nejsou vybrány položky pro synchronizaci, byla rychlost poměrně větší, než při synchronizaci více položek (soubory, e-maily, zprávy). Nelze tedy přesně určit, která data mají při přenosu přednost nebo jak jsou řazena, což je dáno uzavřeností ActiveSync protokolu, který Microsoft neuvolňuje.

Pro testování databáze byla využita data získaná přímo ze zařízení. Dalším zdrojem dat byly programově vygenerované XML dokumenty, které odpovídaly struktuře dokumentů ze zařízení. Do tabulek operátorů, tarifů a přístrojů pak byly data zanesena manuálně. Nastavení parametrů dotazu je intuitivní a z výsledků dotazů se dají snadno určit vynaložené náklady na hovory a provolaný čas. Ladění a testování bylo podstatně usnadněno využitím technologie LINQ. Visual Studio za překladu kontrolovalo chyby v syntaxi dotazů, při vyvolání výjimky pak bylo jasně patrné, ve kterém dotazu a jakého typu se vyskytla chyba. Doposud používaný způsob, kdy jsou dotazy reprezentovány řetězcem a následně odesílány serveru toto neumožňují. Databáze se přihlašuje k localhostu a přístup k ní je chráněn windows autentizací, může být tedy přenesena na jiné PC s MS SQL serverem.

K dispozici bylo pouze jedno mobilní zařízení, nemohli jsme tedy provést více testů. Vycházíme-li ze skutečnosti, že aplikace na zařízení komunikuje s PC prostřednictvím ActiveSync, existuje velmi dobrá pravděpodobnost, že bude funkční i na ostatních zařízeních vybavených operačním systémem WM 6 a vyšším. Visual Studio umožňuje kód překompilovat i pro jiné platformy jako Pocket PC 2003, WM 5, Windows CE.

7 Závěr

Cílem diplomové práce bylo implementovat informační systém pro evidenci telefonních hovorů. Systém může být využit ve firmě pro sběr informací o provedených hovorech, výsledky jim poskytované by pak mohly uspořit vynaložené finanční náklady. Řešení práce spočívalo v teoretické přípravě, na základě které byl vytvořen návrh, jenž byl implementován a současně s implementací pak probíhalo testování a dokumentace práce.

V teoretické části práce byly nastudovány možnosti propojení a komunikace mobilního zařízení s osobním počítačem. Dále byl vytvořen přehled nejčastěji používaných operačních systémů pro mobilní zařízení. V části o komunikaci mobilního zařízení s osobním počítačem byly uvedeny základní principy přenosu dat po vedení, infračerveného a radiového přenosu. Jednotlivé technologie byly popsány od fyzické úrovně až po strukturu dat. Uvedeny byly přenosové rychlosti, energetická náročnost a komunikační dosah. Na základě těchto vlastností byly technologie zhodnoceny pro použití v mobilních zařízeních. Pojednání o operačních systémech bylo zaměřeno na Windows Mobile a Symbian. Popsány byly jednotlivé vývojové verze a základní vlastnosti každého ze systému. Porovnání se provádělo zejména s ohledem na vývoj aplikací a podporu vývojových prostředí pro danou platformu. Na základě této studie byl za cílovou platformu zvolen systém Microsoft Windows Mobile, za způsob komunikace pak USB nebo Bluetooth spojení prostřednictvím synchronizační aplikace ActiveSync. Teoretická část pak pokračovala v dostudování informací týkajících se technologií úzce spjatých s vybraným systémem.

Praktická část práce byla rozdělena do tří fází. První fáze se zabývala odchytem a získáním informací o provedeném hovoru na mobilním zařízení, řešen byl také způsob uložení záznamů s ohledem na paměťové vlastnosti zařízení. Další fáze spočívala v realizaci spojení a přenosu dat mezi zařízením a počítačem. V poslední fázi byla navržena databáze pro aplikaci pracující na osobním počítači. Vytvořeno bylo grafické rozhraní a rutiny pro práci s databází. Současně s vývojem aplikace probíhalo také její testování.

Při návrhu řešení bylo předpokládáno, že v systému se bude uchovávat velké množství dat, bylo proto zvoleno řešení v podobě SQL databáze. Značnou nevýhodou tohoto přístupu je poměrně zdlouhavá a uživatelsky nepříliš přívětivá instalace produktu, která zahrnuje vytvoření databáze s potřebnými tabulkami a v případě absence i instalaci SQL serveru. Jako řešení tohoto problému je možnost uchovávat data v XML dokumentech i na straně počítače. Zpracovávání záznamů by bylo díky technologii LINQ obdobné jako při využití databáze, nutností by však bylo hlídat velikosti dokumentů, aby jejich zpracování nebylo příliš paměťově náročné.

Další vývoj aplikace by se měl ubírat doplněním základních nedostatků, jako je absence instalačních balíčků a na straně zařízení automatický start programu. Dalším rozšířením by mohlo být přenesení desktopové aplikace do prostředí operačního systému Windows Vista.

Literatura

[1] Schiller, J.: Mobile Communications (2nd edition), Addison-Wesley 2003, ISBN 0-321-12381-6

[2] Hansman, U., Merk, L., Micklous, M., Strober, T.: Pervasive computing (2nd edition), Springer 2003, ISBN 3-540-00218-9

[3] Tanenbaum, A.: Computer network (4th edition), Prentice-Hall 2003, ISBN 0-13-066102-3

[4] Lacko, L.: Programujeme mobilní aplikace ve Visual Studiu .NET, Computer Press 2004, ISBN 80-2510176-2

[5] Harrison, R.: Programujeme aplikace Symbian OS v jazyce C++, Computer Press 2006, ISBN 80-251-1243-8

[6] Strangio, Ch.: The RS232 Standard 1993. Dokument dostupný na URL http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html (prosinec 2008)

[7] Universal Serial Buss Specification, USB Implementers Forum, 2000. Dokument dostupný na URL http://www.usb.org/developers/docs/usb_20_040908.zip (prosinec 2008)

[8] Malý, M.: USB 2.0, 2005. Dokument dostupný na URL http://hw.cz/Rozhrani/ART1232-USB-2.0---dil-1.html (prosinec 2008)

[9] Vojáček, A.: USB OTG, 2008. Dokument dostupný na URL http://hw.cz/teorieapraxe/dokumentace/art2361-co-se-skryva-pod-komunikaci-oznacenou-jako-usb-otg.html (prosinec 2008)

[10] Myslík, V., Řehák, J.: IrDA – Kompletní popis, 1998. Dokument dostupný na URL http://hw.cz/Teorie-a-praxe/Dokumentace/ART784-IrDa---Kompletni-popis.html (prosinec 2008)

[11] Pužmanová, R.: Moderní komunikační sítě od A do Z, Computer Press 2006, ISBN 80-251-1278-0

[12] Specification of the Bluetooth System. Volume 0, Bluetooth SIG 2007. Dokument dostupný na URL http://bluetooth.com/NR/rdonlyres/F8E8276A-3898-4EC6-B7DA-E5535258B056/6545/Core_V21__EDR.zip (prosinec 2008)

[13] Řehák, J.: Co je to WiFi, 2003. Dokument dostupný na URL http://hw.cz/Produkty/Ethernet/ART915-Co-je-to-WiFi---uvod-do-technologie.html (prosinec 2008)

[14] Stránky Zigbee Alliance, dostupné na URL http://www.zigbee.org (prosinec 2008)

[15] Stránky vývojového centra Microsoft, dostupné na URL http://msdn.microsoft.com/en-us/windowsmobile (prosinec 2008)

[16] Stránky Nokia Forum, dostupné na URL http://www.forum.nokia.com (prosinec 2008)

[17] Prosise, J.: Programování v Microsoft .NET : webové aplikace v .NET Framework, C# a ASP.NET, Computer Press 2006, ISBN 80-7226-879-1

[18] Virius, M.: C#: Hotová řešení, Computer Press 2006, ISBN 80-251-1084-2

[19] Stránky Base Class Libraries Community, dostupné na URL http://msdn.microsoft.com/en-us/netframework/aa569603.aspx (prosinec 2008)

[20] Agarwal, V. V., Huddleston, J.: Databáze v C# 2008 : průvodce programátora, Computer Press 2009, ISBN 978-80-251-2309-6

[21] Pialorsi, P., Ruso, M.: Introducing Microsoft® LINQ, Microsoft Corporation 2007, ISBN 0-7356-2391-0

[22] Knight, B.: Microsoft SQL Server 2000: Pokročilé techniky, Computer Press 2004, ISBN 80-251-0111-8

[23] Stanek, W. R.: Microsoft SQL Server 2005: Kapesní rádce administrátora, Computer Press 2006, ISBN 80-251-1211-X

[24] SQL Server Overview. Stránky společnosti Microsoft dostupné na URL http://msdn.microsoft.com/en-us/library/ms166352(SQL.90).aspx

[25] Dostálek, L., Kabelová, A.: Velký průvodce TCP/IP a systémem DNS 3. aktualizované a rozšířené vydání, Computer Press 2002, ISBN 80-7226-675-6

Seznam příloh

Příloha 1 – Instalační manuál

Aplikace na mobilním zařízení

1. Nainstalujte aplikaci ActiveSync, je standardně dodávána se zařízeními WM
2. Do mobilního zařízení nahrajte adresář s aplikací CallLoger .
   Adresář můžete nahrát pomocí synchronizační aplikace ActiveSync.
3. Aplikaci spustíte pomocí CallLoger.exe

Aplikace na osobním počítači

1. Ověřte, zda je na stroji nainstalovaný .NET Framework 3.5 nebo vyšší
   Zdroje ke stažení:

   Framework 3.5 : http://www.microsoft.com/downloads/details.aspx?FamilyId=333325FD-AE52-4E35-B531-508D977D32A6&displaylang=en

   Service Pack 1 : http://www.microsoft.com/downloads/details.aspx?FamilyID=AB99342F-5D1A-413D-8319-81DA479AB0D7&displaylang=en

2. Jestliže není na stroji nainstalovaný MS SQL Server, stáhněte a nainstalujte si např. Express edici a Microsoft Management Studio
   Zdroje ke stažení:

   SQL Server 2005 Express:
   http://www.microsoft.com/downloads/details.aspx?familyid=220549b5-0b07-4448-8848-dcc397514b41&displaylang=en

   MS SQL Server Management Studio Express:
   http://www.microsoft.com/downloads/details.aspx?FamilyId=C243A5AE-4BD1-4E3D-94B8-5A0F62BF7796&displaylang=en

3. Vytvoření databáze a tabulek.
   Pomocí Management Studia otevřete skript pro vytvoření databáze:

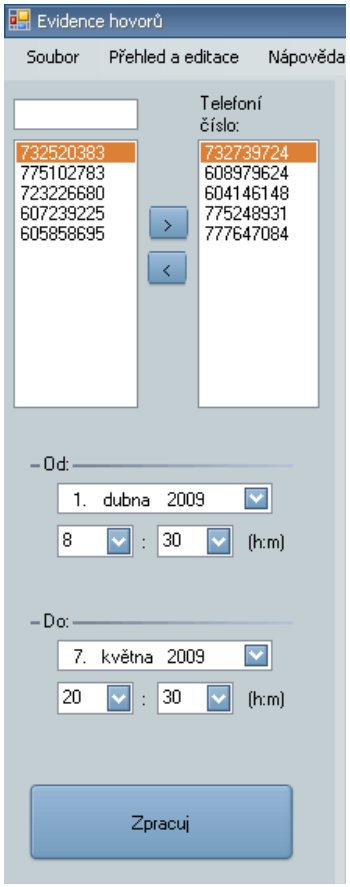
   - File > Open > File …
   - z instalační složky vyberte skript CreateDb.sql
   - skript spusťte pomocí tlačítka !Execute

   K nově vytvořené databázi EvidenceHovoru se připojte a otevřete skript pro vytvoření tabulek:

   - File > Open > File …
   - z instalační složky vyberte skript CreateTables.sql
   - skript spusťte pomocí tlačítka !Execute

4. Spuštění aplikace.
   Z instalační složky si zkopírujte složku EvidenceSQL, aplikaci spusťte pomocí EvidenceSQL.exe

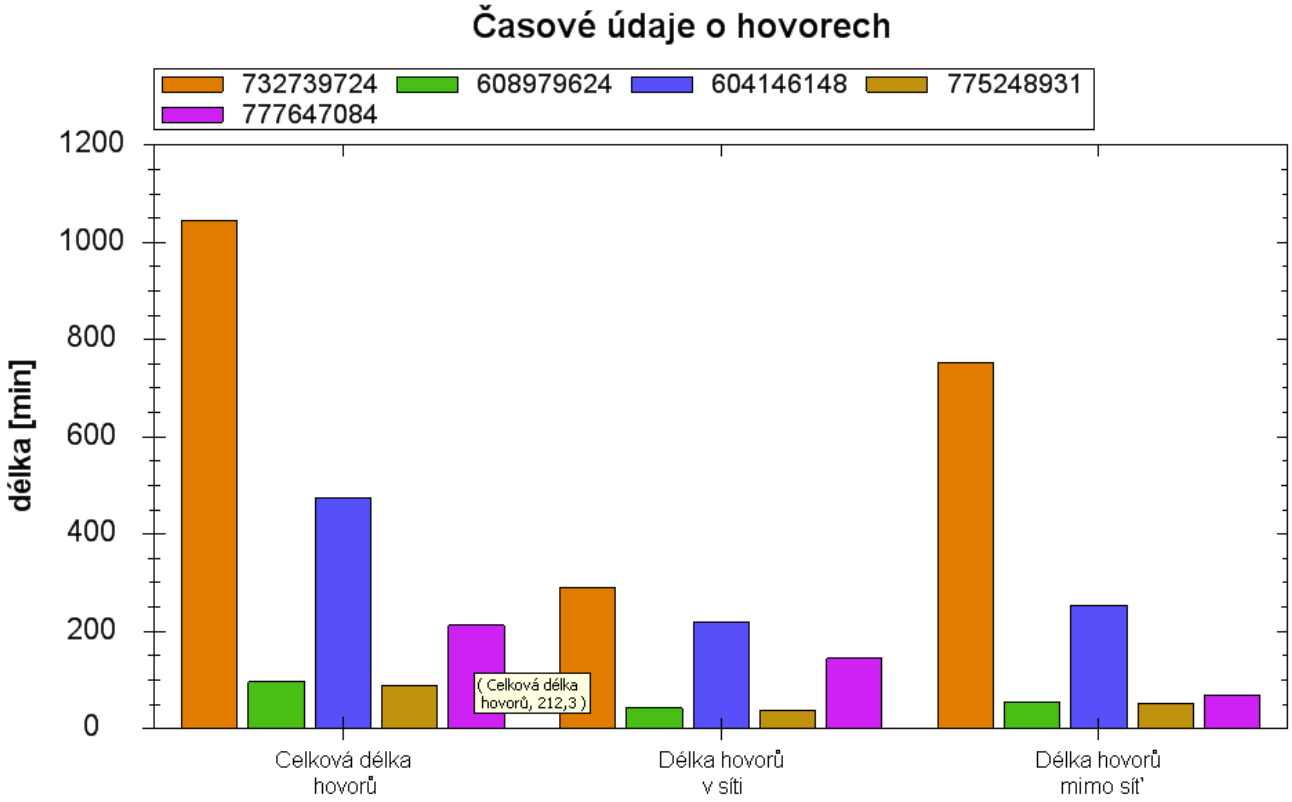
Příloha 2 - Příklady grafických výstupu desktopové aplikace



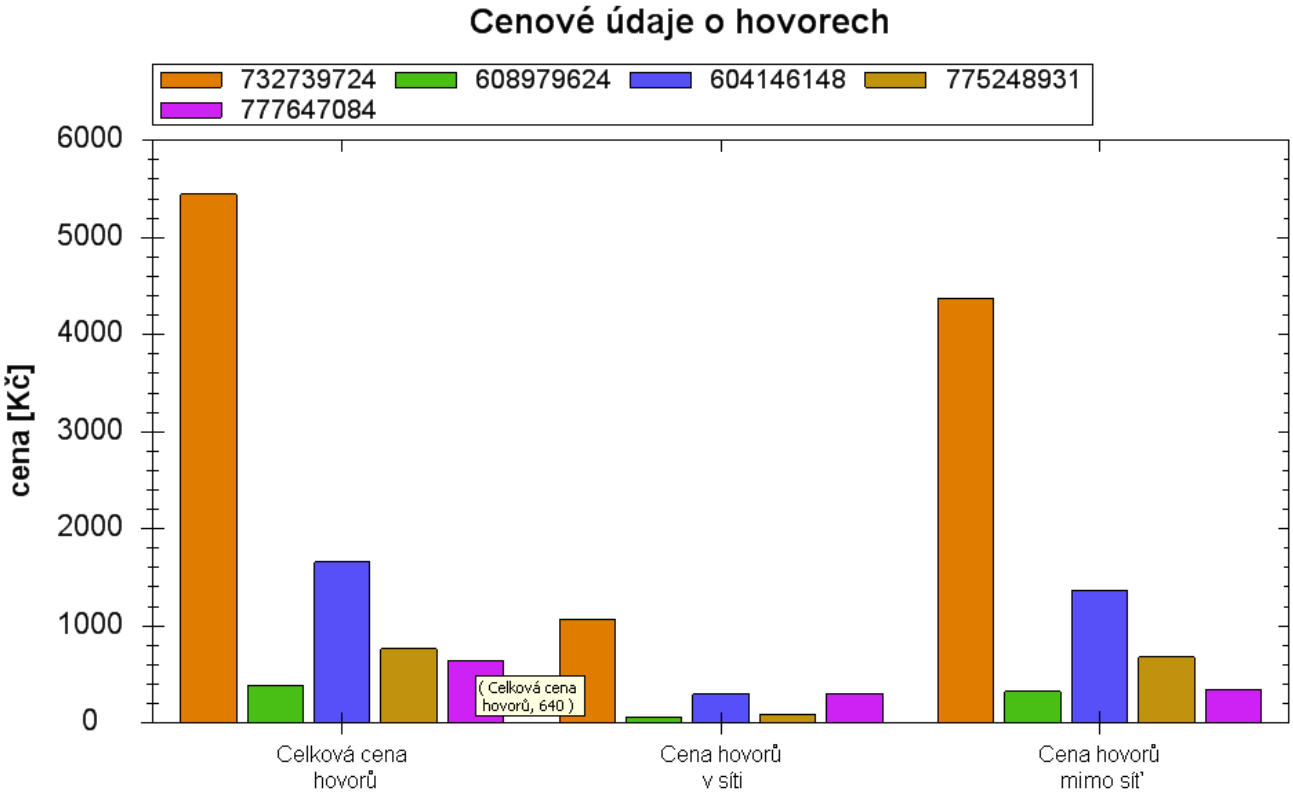
Příklad nastavení pro zjištění informací o hovorech za určité období pěti vybraných čísel

Telefonní číslo	Operátor	Jméno tarifu	Počet hovorů	Délka hovorů	Délka hovorů v síti	Délka hovorů mimo síť	Cena hovorů	Cena hovorů v síti	Cena hovorů mimo síť
604146148	T-Mobile	T-Mobile T80	176	7:53:02	3:39:01	4:14:01	1656,50	297,50	1359,00
732739724	T-Mobile	T-Mobile BAV SE	175	17:24:48	4:51:14	12:33:34	5446,65	1072,89	4373,76
777647084	Vodafone	Vodafone nabito 700	106	3:32:18	2:23:48	1:08:30	640,00	300,00	340,00
608979624	Vodafone	Vodafone nabito 350	34	1:36:20	0:42:50	0:53:30	390,00	65,00	325,00
775248931	Vodafone	Vodafone Odepiš	27	1:29:18	0:36:41	0:52:37	763,98	85,68	678,30

Výřez zobrazující podrobnosti o hovorech vybraných čísel. Pomocí zaškrtávacích tlačítek se volí informace, které se mají zobrazit.



Graf zobrazující informace o provolaných minutách, u kurzorem vybraného sloupce se zobrazí přesná hodnota.



Graf zobrazující informace o vynaložených nákladech.

Příloha 3 - Informace o použitém mobilním zařízení

Využito bylo PDA **HTC TyTN II (Kaiser)**

Systémové informace	
Procesor	Qualcomm® MSM 7200 400MHz
Paměť	- ROM: 256 MB - RAM: 128 MB SDRAM
Operační systém	Windows Mobile® 6 Professional
Displej	
Typ LCD	2,8 palce dotykový TFT-LCD
Rozlišení	240 x 320, 65 536 barev
Modul HSDPA / UMTS / GSM / GPRS / EDGE	
Funkčnost	HSDPA/UMTS: Trojpásmový (850, 1900 a 2100 MHz). HSDPA: Do 384 kb/s pro odesílání a 3,6 Mb/s pro přijímání dat. UMTS: Do 384 kb/s pro odesílání a přijímání dat. GSM/GPRS/EDGE: Quad-band (850, 900, 1800 a 1900 MHz)
Vnitřní anténa	Ano
Tělo	
Rozměry	112 mm (D) x 59 mm (Š) x 19 mm (V)
Hmotnost	190 g (s baterií)
Připojení	
Port I/O	HTC ExtUSB™ (11 pólový mini-USB a audio konektor v jednom, USB 2.0 plná rychlost).
Bezdrátová připojení	Wi-Fi (IEEE 802.11 b/g), Bluetooth 2.0 s EDR