VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

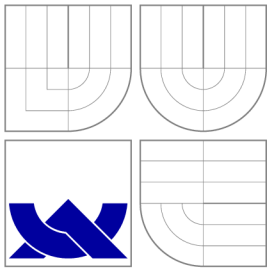
DEMONSTRAČNÍ APLIKACE MAEMO

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

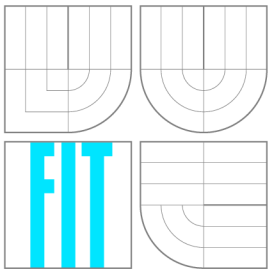
AUTOR PRÁCE
AUTHOR

JAKUB ŠPLÍCHAL

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DEMONSTRAČNÍ APLIKACE MAEMO

MAEMO DEMONSTRATION APPLICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAKUB ŠPLÍCHAL
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAKUB FILÁK
SUPERVISOR

BRNO 2010

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2009/2010

Zadání bakalářské práce

Řešitel: **Šplíchal Jakub**
Obor: Informační technologie
Téma: **Demonstrační aplikace Maemo**
Maemo Demonstration Application
Kategorie: Alg. a datové struktury

Pokyny:

1. Seznamte se s prostředky operačního systému Linux na ultrapřenosných zařízeních a zaměřte se na systém Maemo.
2. Prozkoumejte práci s běžnými typy dat a událostí v mobilních telefonech, jako jsou hovory, zprávy, kontakty, jejich skupiny, profily nebo multimediální soubory a události při jejich přijetí, odesílání, respektive modifikaci.
3. Dle pokynů vedoucího navrhněte aplikaci, která bude demonstrovat výše uvedené možnosti operačního systému Maemo.
4. Aplikaci implementujte a ukažte její funkčnost na příkladě možnosti nastavení inteligentního individuálního vyzvánění.
5. Diskutujte implementované řešení a jeho případné rozšíření.

Literatura:

- Mobile Phone Programming: and its Application to Wireless Networking. F. H.P. Fitzek, F. Reichert (editors). Springer, 2007. 473 p. ISBN 978-1402059681.
- Andrew Krause. Foundations of GTK+ Development (Expert's Voice in Open Source). Apress, 2007. 630 p. ISBN 978-1590597934.

Při obhajobě semestrální části projektu je požadováno:

- 1. a 2. bod zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Filák Jakub, Ing.**, UIFS FIT VUT
Konzultant: Chmelař Petr, Ing., UIFS FIT VUT
Datum zadání: 1. listopadu 2009
Datum odevzdání: 19. května 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Práce se zabývá mobilními operačními systémy, využívajích jádro Linux. Hlavní část práce je zaměřena na vytvoření aplikace na inteligentní nastavení vyzváněcích profilů, pro mobilní operační systém Maemo. Ta umožňuje automaticky nastavit vybraný profil na základě různých pravidel, například podle jména WiFi sítě. Samotná aplikace byla implementována v jazyce C++ za použití knihoven Qt 4. Na příkladu této aplikace jsou demonstrovány prostředky a nástroje pro přístup k relevantním datům a také potřebné související operace.

Abstract

This work deals with mobile operating systems using the Linux kernel. The main part is focused on creating applications for smart ringing profiles for Maemo operating system. This allows to automatically set the profile automatically on the basis of various rules, such as WiFi network name. The application itself has been implemented in C++ using Qt libraries. The application demonstrates appliances and tools to access relevant data and other necessary operations.

Klíčová slova

Maemo, vyzváněcí profily, Qt framework, C++

Keywords

Maemo, ring profiles, Qt framework, C++

Citace

Jakub Šplíchal: Demonstrační aplikace Maemo, bakalářská práce, Brno, FIT VUT v Brně, 2010

Demonstrační aplikace Maemo

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jakuba Filáka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Šplíchal
16. května 2010

Poděkování

Chtěl bych poděkovat panu Ing. Jakubu Filákovi a panu Ing. Petru Chmelařovi, za jejich odbornou pomoc, cenné připomínky a rady při vytváření této práce.

Obsah

Kapitola 1

Úvod

V dnešní době jsou osobní počítače stále více osobnější a stávají se dostupnými kdykoliv a kdekoliv. V čele tohoto pokroku jsou kapesní zařízení, která se mění v počítačové platformy. Mobilní telefony nejsou pouze určeny pro telefonování, ale staly se mnoho účelnými zařízeními, které jsou předurčeny stát se dalšími osobními počítači. Trh s mobilními zařízeními je jednen z nejvíce rozvíjejících se. Většina z těchto zařízení je dostupná pro velkou část populace. Většinou obsahují mnoho aplikací pro denní použití, hlavně ve spojení s internetovou sítí.

Cílem mé bakalářské práce je demonstrovat možnosti operačního systému Maemo na konkrétní aplikaci, která umožní inteligentní nastavení vyzváněcích profilů. Samotná aplikace by měla umožnit vytváření, mazaní a nastavení jednotlivých profilů. Také nastavit určitý profil na zvolenou dobu, nastavit profil na základě popisu událostí obsažených v kalendáři a na označení lokality, ve které se mobilní zařízení nachází. To jak za pomocí telefonní sítě nebo WiFi připojení a také na základě čísla příchozího hovoru. Aplikace by měla mít jednoduché uživatelské rozhraní.

Na začátku práce je čtenář seznámen s nejznámějšími operačními systémy založených na jádru Linux a také s jejich historií, vlastnostmi a možnostmi vývoje aplikací pro daný systém. Kapitola také popisuje použité technologie a knihovny pro vývoj demonstrační aplikace, jako jsou například knihovny Qt 4, D-Bus a další. V následující kapitole budeme rozebírat základní specifikaci aplikace, bude zde diskutována realizace změn profilu, ukládání profilů a pravidel do jednoho z navrhovaných úložišť. Komunikace aplikace s kalendářem a získávání informací o událostech obsažené v jednotlivých kalendářích a také problematika zjištění informací o sítích a příchozích hovorech. Na konci práce je popsána samotná implementace jednotlivých komponent, využití frameworků a také je zde ukázáno uživatelské rozhraní aplikace. Závěr obsahuje zhodnocení celé práce a jejího přínosu pro uživatele.

Kapitola 2

Základní teorie a pojmy

2.1 Přehled mobilních operačních systémů Linux

V dnešní době existuje velké množství multifunkčních mobilních zařízení od velkého počtu výrobců. Mnoho z nich si vytváří vlastní mobilní operační systémy. Výrobci využívají často k tvorbě nových mobilních systémů jádro *Linux*, které je vedeno pod licencí *GNU General Public License* [6]. Jádro *Linux* lze modifikovat pro vlastní potřebu, což je výhodou pro další vývoj mobilních systémů. Na druhou stranu si společnosti většinou vytváří vlastní knihovny, *API* a využívají jiné programovací jazyky, což vede k nekompatibilitě programů na mobilních zařízeních různých výrobců. Nyní následuje popis nejznámějších mobilních operačních systémů, využívajících jádro *Linux*.

2.1.1 Google Android

Firma *Android Inc.* započala vývoj systému *Android* a v roce 2005 tuto firmu koupila společnost *Google*. V roce 2007 *Google* vydal *Android SDK* jako takzvaný „early look“. Dalšího roku *Google* oznámil dostupnost *Android SDK Release Candidate* 1.0 a v říjnu téhož roku byla platforma *Google Android* dostupná pod licencí *Apache's open source*[1]. Jádro systému je založeno na jádru *Linux*. Většina aplikací je vytvořena v programovacím jazyce Java a spuštěna pomocí *Dalvik Virtual Machine*. Tento operační systém bude také využit jako operační systém pro netbooky výrobců jako jsou *ASUS*, *HP* a nebo *DELL*.

Obrázek 2.1 zobrazuje základní architekturu systému *Android*. Operační systém Android je rozdělen do pěti úrovní. Zde bude následovat jejich popis:

Linux Kernel – *Google Android* využívá jádro *Linux* ke správě paměti, procesů a dalších služeb operačního systému. Obsahuje ovladače pro displej, kameru, klávesnici, *WiFi*, flash paměť, zvuk, *IPC* a správu napájení.

Native Libraries – Je soubor *C/C++* knihoven, které využívají nejrůznější komponenty systému. Obsahuje knihovny jako například *OpenGL ES* pro akceleraci grafiky, *WebKit* pro renderování html stránek, *SQLite* pro ukládání dat do databáze. Přístup ke knihovnám není přímý, ale pomocí aplikačního frameworku.

Android Runtime – Je soubor knihoven implementovaných v jazyce *Java* a *Dalvik Virtual Machine* (*Dalvik VM*). Dalvik je virtuální stroj (*VM*), navržený a napsaný *Danen Bornstein* v *Googlu*. Kód se zkompiluje do platformě nezávislého kódu tzv. bytekódu,

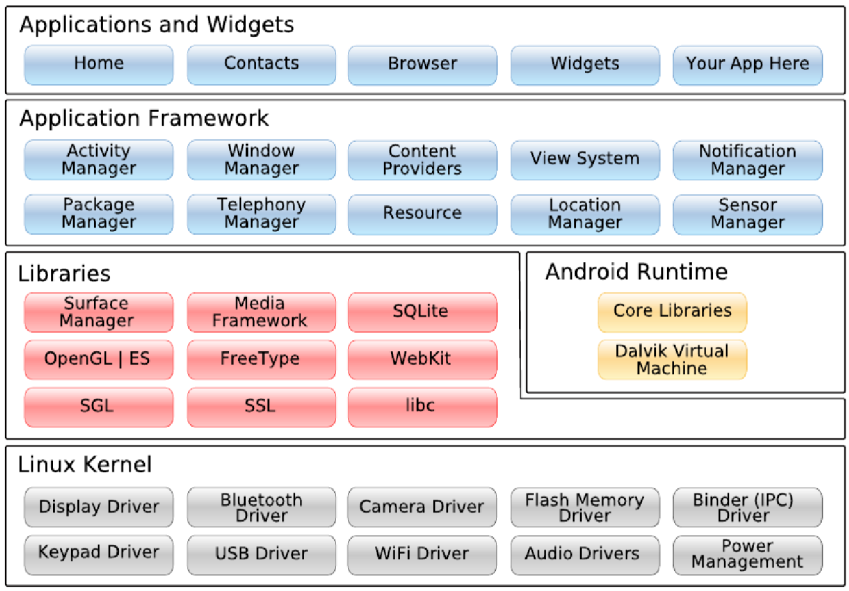
[1] viz. http://www.apache.org/licenses/LICENSE-2.0

který je pak prováděn *Dalvik VM* na mobilním zařízení. Přestože bytecode formáty jsou trochu odlišné, *Dalvik* je v podstatě *Java Virtual Machine*, který je optimalizován pro nízké nároky na paměť. Je optimalizovaný pro běh na mobilních zařízeních a každý proces má svůj *VM*. Umožňuje spustit více *VM* instancí souběžne a využívá základní operační systém pro bezpečnost a izolaci procesů.

Application framework – Je soubor hlavních knihoven, které jsou nad nativními knihovnami a *Android Runtime*. Tuto vrstvu využívají programátoři při vývoji programů a přístupu k nižších vrstvám. Framework je předinstalován na mobilním zařízení, ale můžeme ho také rozšířit vlastními komponenty.
Nejdůležitější částí frameworku jsou:

- *Activity manager*: Spravuje životní cyklus jednotlivých aplikací
- *Content providers*: Tyto objekty zapouzdřují data, která jsou potřeba sdílet mezi aplikacemi, jako například kontakty.
- *Resource manager*: Správce zdrojů zajišťuje programům přístup k neprogramovatelným datům, například obrázky.
- *Location manager*: Slouží ke zjištění pozice mobilního zařízení, například z GPS.
- *Notification manager*: Zobrazuje události, jako příchozí zprávy a schůzky uživateli.

V této části jsem vycházel z těchto knih [22, 23]



Obrázek 2.1: Architektura systému Google Android, obrázek byl převzat z [22]

2.1.2 WebOS

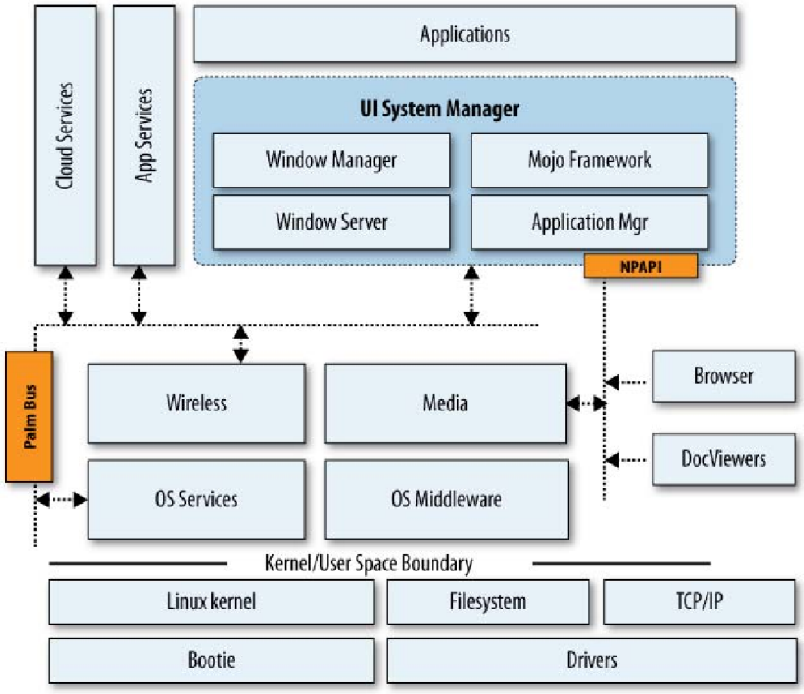
Operační systém *WebOS* byl představen v lednu roku 2009 firmou *Palm*. *Palm WebOS* je založeno na inovativním návrhu, které integruje moderní grafický operační systém spolu s webovými technologiemi, které umožňují vytvářet aplikace s využitím běžných webových jazyků a nástrojů, bez omezení pracovat ve webovém prohlížeči.

Aplikace jsou postaveny s využitím jazyků *JavaScript*, *HTML* a *CSS*. Tento aplikační model tedy umožňuje používat stejné jazyky a nástroje, které slouží k tvorbě webových aplikací.

WebOS se dělí na dvě hlavní části a to *Core OS*, což je jádro *Linux* verze 2.6 a *application environment*. Kombinuje tak *open source* a *Palm* komponenty pomocí kterých se přistupuje k jádru. Architekturu operačního systému zobrazuje obrázek 2.2.

Aplikační prostředí – Je spravováno pomocí *UI system Manager*. Framework umožňuje přístup k *UI widgets* a *Palm webOS* službám. Podporu tohoto prostředí zajišťuje *Core OS*, což je jádro *Linux* s přidanými komponentami. Uživatel ani programátor nemá přímý přístup k jádru, ale musí využít *Mojo API*.

UI System Manager – Je zodpovědný skoro za vše co vidí uživatel. Aplikační manažer je postaven na open source *WebKit* renderovacím jádře, který nahrává individuální aplikace a také systémové aplikace jako je status bar atd. Aplikační manager běží v jediném procesu, plánuje a spravuje každou běžící aplikaci. Stará se o vykreslování pomocí grafického subsystému a také řídí ukládání dat na zařízení, pomocí databázového systému.



Obrázek 2.2: Architektura systému WebOS, obrázek byl převzat z [20]

Core OS – Zahrnuje jádro *Linux* , ovladače a služby systému, *wireless* systém a media. Wireless systém poskytuje přístup k WAN a WiFi sítím a umožňuje si vybrat mezi

nimi, ale prioritně vybírá WiFi připojení. Dále podporuje standardní *Bluetooth* profily a zajišťuje jednoduché párování přístrojů. *Media server* v systému je postaven na *GStreamer*. Zahrnuje podporu mnoha audio a video kodeků a všechny hlavní obrázkové formáty. Podporuje také snímání obrázků přes vestavěnou kameru.

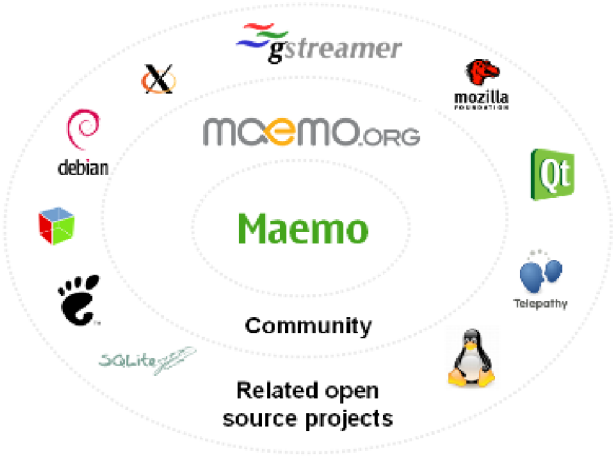
Vycházel jsem z těchto knih [20, 24].

2.1.3 Maemo 5

Meamo je operační systém vytvořený firmou Nokia pro chytré telefony a internetové tablety. Samotný systém je založen na jádru *Linux* a vychází z distribuce *Debian*. Uživatelské rozhraní architektury *Maemo* 5 je založeno na knihovnách *GNOME*. *Maemo* platforma zdědila hlavně komponenty jako *GTK+*, *GStreamer* multimediální framework, konfigurační systém *GConf* a knihovnu pro zpracování *XML*. Platforma *Maemo* tak rozšiřuje *GTK+* a *GNOME* na mobilní zařízení.

Díky tomu, že je *Maemo* založeno z velké části na open source projektech viz. obrázek 2.3, tak portovaní aplikaci z desktopu se stává jednoduchou záležitostí a to je hlavní rozdíl oproti *Google Android* a *WebOS*, kde toto možné není. Aplikace mohou být přímo napsané pro systém *Maemo*, ale některé jsou porty[2] existujících aplikací jako *Firefox*, *Skype*, *Pidgin*, atd. Instalace aplikací je řešena pomocí *Debian software balíčků*.

Meamo SDK (Software Development Kit) je složeno z programů a nástrojů pro vývojáře k tvorbě aplikací pro systém *Maemo*. *SDK* obsahuje nástroj *scratchbox* pro multiplatformní kompilaci. Hlavní vývoj aplikace probíhá na stolním počítači a finální verze je zabalena pro mobilní architekturu *ARM*. Dále obsahuje potřebné knihovny a software pro vývoj, který slouží programátorovi k testování aplikací aniž by vlastnil zařízení s *Maemo* operačním systémem. Programovat aplikace lze v jazyce *C*, *C++*, *Python*. Programátor může využít při tvorbě *GUI* programu *GTK+*, ale také *QT* 4 knihovny.



Obrázek 2.3: Přehled opensource projektů v *Maemo*, obrázek převzat z [4]

Architektura systému Maemo 5

V následujících bodech je popsána architektůra systému *Maemo* 5 , viz. obrázek 2.4

[2] viz. http://en.wikipedia.org/wiki/Porting

Linux kernel – Jádro Linux je centrální součástí systému. Poskytuje *Hardware Abstraction Layer* (HAL) pro zařízení systému, správu paměti, řízení procesů, síťové služby, správu souborů včetně souborového systému a různé další služby. *Maemo* 5 je založeno na jádru *Linux* verze 2.6. Jádro je zkompilováno pro *ARM* architekturu procesorů. Mezi ovladače zařízení patří například *USB*, *LCD*, *WLAN*, kamera a zvuk.

System Libraries – Platforma *Maemo* využívá standardní knihovny *GNU C*. Pro vytváření zabezpečených sítí, *Maemo* platforma používá *OpenSSL* knihovny, které poskytují síťové zabezpečení a knihovnu *libcurl*, která poskytuje mnoho protokolů pro aplikace jako například *HTTP*, *FTP*, *SCP* a další. Dále také obsahuje *X Windows System*, *POSIX* a *BusyBox*, který kombinuje malé verze mnoha běžných nástrojů *UNIX* do jednoho malého spustitelného souboru, hodícího se do mobilních operačních systémů.

Debian Package Management – Balíčkovací systém, který slouží k instalaci aplikací.

System Services – Hlavním komunikačním kanálem mezi aplikacemi je *D-Bus*. *D-Bus* také slouží pro interakci mezi systémem a aplikacemi. Systémové služby poskytují relační databázi *SQLite 3*, která může být použita k ukládání uživatelských dat. *SQLite* databáze je přístupná přes knihovní rozhraní. Systémové služby poskytují také velké množství služeb pro aplikace a koncové uživatele. Tyto služby zahrnují stav zařízení (*DSM*), režim kontroly (*MCE*), správce baterie (*BME*) a několik grafických prvků uživatelského rozhraní pro správu chování služeb.

Multimedia Framework – Obsahuje multimediální rozhraní včetně *GStreamer*, *ALSA*, *PulseAudio* a *API* nižší úrovně, jako například kodeky *DSP*. Rovněž je odpovědný za audio-směrování a zvuky událostí.

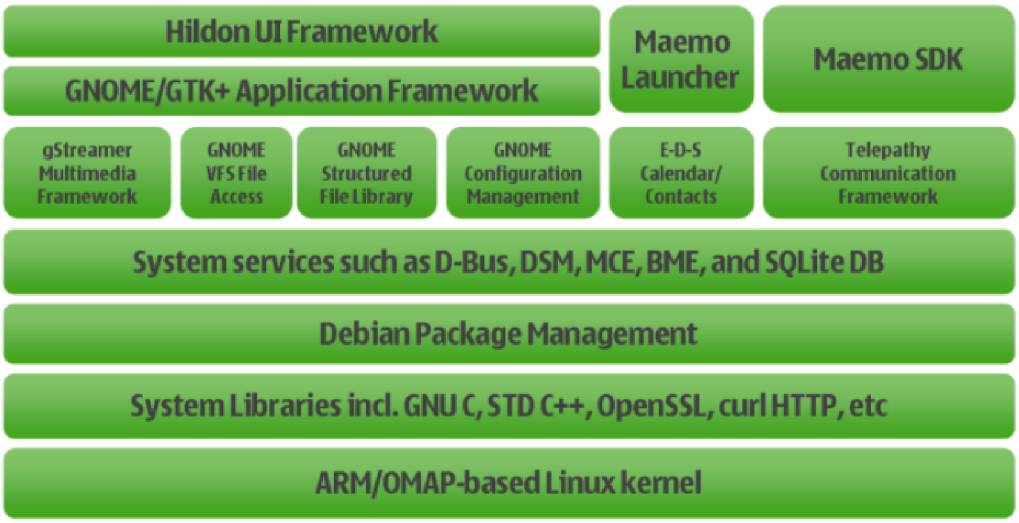
GNOME – Obsahuje knihovny *GNOME* jako *gconf2*, která slouží pro ukládání nastavení aplikací, gtk+, glib a dalších z GNOME projektu.

Communication Framework – Poskytuje i služby a aplikace pro realtime komunikaci přes Internet (Voice Over IP volání a Instant Messaging) na platformě *Maemo*. Poskytuje také a distribuuje informace o přítomnosti v rámci systému a je také zodpovědný za hladkou integraci externě vytvořené aplikace *Skype* do systému.

Hildon UI Framework – Uživatelské rozhraní je založeno na *X Window System*, které používá správce oken Matchbox a grafické prostředí *Hildon* založené na *GTK+* .

Applications – *Maemo* platforma obsahuje aplikace jako je prohlížeč, e-mail klient, kalendář, hry atd. Ne všechny tyto aplikace, které jsou předinstalovány na systému, jsou open source, ale existují adekvátní náhrady za proprietární aplikace.

Vycházel jsem z těchto zdrojů [7, 17]



Obrázek 2.4: Architektura systému *Maemo*, obrázek byl převzat z [17]

2.1.4 Shrnutí

Operační systémy využívající jádro *Linux* se od sebe liší, jak podporou programovacích jazyků, tak velikostí uživatelské komunity a podporou vývojových prostředí. Každý systém má své výhody a nevýhody. *Google Android* je založený na jazyku *Java* a tak programátor pro tuto platformu si nemůže vybrat jiný programovací jazyk. Ale na druhou stranu obsahuje kvalitní vývojové prostředí, dokumentaci a literaturu. *WebOS* využívá webové jazyky, které jsou v dnešní době velmi populární. Pro platformu *Maemo* lze využít největší množství jazyků jako například *C*, *C++*, *Python*, ale nemá tak kvalitní dokumentaci a vývojové prostředí není zcela uživatelsky přívětivé.

2.2 Důležité části platformy Maemo

Jelikož je tato práce zaměřena na platformu *Maemo*, následuje nyní popis důležitých částí, které jsou využity k tvorbě demonstrační aplikace.

2.2.1 Maemo SDK

Maemo SDK vývojové prostředí je z velké části postaveno na nástroji zvaném *Scratchbox*, jak již bylo řečeno v 2.1.3. Toto prostředí se chová jako operační systém na zařízení, ale s přidanými nástroji pro vývoj. Vývojový proces je tedy velmi podobný vývoji na desktop počítači, ale s trochou křížové kompilace řešené transparentně v *Scratchbox*.

Maemo SDK podporuje dvě architektury a to *x86* a *ARMEL*. Cílová architekrura *x86* je použita pro aktivní vývoj a má lepší nástroje pro běh aplikací, protože nepotřebuje emulaci. Obsahuje *UI framework* pro spuštění a zobrazení aplikací. Cíl *ARMEL* slouží pouze pro kompilaci pro *ARM* architekturu. Aplikace, které jsou zkompilovány pro tento cíl, lze spustit přímo v mobilním zařízení bez dalších úprav. Toto je možné díky emulaci poskytnuté *Maemo SDK*, ale skutečné testování aplikace se musí provést na samotném zařízení. Vycházel jsem z tohoto zdroje [8].

2.2.2 D-Bus

V meziprocesové komunikaci (*IPC*) *Maemo* do značné míry závisí na systému *D-Bus*. *D-Bus* umožňuje, aby aplikace vystavily své programové rozhraní, takže další procesy je mohou volat jim konzistentním způsobem, aniž by bylo nutné definovat vlastní *IPC* protokol.

Systém Maemo obsahuje knihovnu *libOSSO*, která užitečně zaobaluje *D-Bus* komunikaci. To také obsahuje požadovanou funkčnost pro všechny aplikace *Maemo*. Díky této knihovně se může aplikace připojit na poslech systémových zpráv jako změny v hardware, například nízký stav baterie.

V systému *D-Bus* je sběrnice důležitý pojem. Aplikace může volat metody, posílat signály a nebo jim přes sběrnici naslouchat. Následuje popis dvou hlavních sběrnic:

Session bus – Je určena pro komunikaci mezi aplikacemi, které jsou připojeny ke stejné relaci a jsou spuštěny jedním uživatelem (užívajícího stejného uživatelského identifikátoru tedy *UID*).

System bus – Je určena pro komunikaci, kdy aplikace (nebo služby), běží v samostatných relacích a chtějí komunikovat s ostatními. Nejčastější použití pro tuto sběrnici je posílání systémových událostí například: přidání nového zařízení, síťové události a dalších podobných událostí.

Kromě jediné systémové sběrnice, může existovat více *session* sběrnic. V systému *Maemo* se všechny aplikace spouštějí pod stejným *UID*, takže přístroj má pouze jednu *session* sběrnici.

Mechanismus *IPC* musí podporovat nějakou formu adresování, pro doručení zpráv příjemci. Systém *D-Bus* je založen na objektech. Aplikace mají objekty a ty implementují určité metody. Komunikace probíhá tak, že aplikace zavolá metodu objektu, která patří jiné aplikaci.

V následujících bodech je popsáno adresování v systému *D-Bus* viz. tabulka 2.1

Jméno připojení – Jméno sběrnice, která slouží pro příjem nebo posílání zpráv. Nejedná se o stejné jméno jako je *System* nebo *Session* sběrnice, ale o označení jedinečného připojení na dané sběrnici. Spojení je pak použito pro odesílání a přijímání zpráv tak dlouho, jak je třeba. Aplikaci je přiděleno jméno automaticky systémem *D-Bus* nebo si jej služba sama přidělí. Jméno připojení vytvořené aplikací se nazývá *dobře známé jméno*.

Cesta – Služby mohou obsahovat více různých objektů, z nichž každý poskytuje jiné (nebo stejné) rozhraní. Aby bylo možné oddělit jeden objekt od druhého, jsou používány cesty k objektům. Cesty k objektům vypadají jako cesty k souborům (elementy odděleny znakem '/').

Rozhraní – Slouží pro podporu objektově orientovaného mapování, kdy jednotlivé objekty implementují dané rozhraní. Pro jednoduché služby, je *dobře známé jméno* často opakováno v názvu rozhraní. Toto je nejčastější scénář s existujícími službami.

Člen – Jedná se o jméno metody nebo signálu, který je implentován objektem. Podle jména se vybere procedura k zavolání nebo signál pro odeslání. Jméno musí být jedinečné pouze v rámci rozhraní, které objekt implementuje.

Vycházel jsem z [2].

Tabulka 2.1: Příklad adresace v D-Bus

	Identifikace	**Příklad**
Připojení	jméno připojení	com.nokia.calendar
Objekt	cesta	/com/nokia/calendar
Rozhraní	jméno rozhraní	com.nokia.calendar
Člen	jméno člena	dbChange

2.3 Qt 4 framework

Qt je multy-platformní aplikační a *UI framework* viz. obrázek 2.5. Slouží jak k vytváření aplikací s grafickým tak i negrafickým rozhraním.

Obrázek 2.5: Platformy kde Qt běží, převzato z [12]

Qt používá jazyk *C++*, ale velmi využívá speciálního pre-procesoru k obohacení jazyka. *Qt* může být využito i v dalších programovacích jazycích.

2.3.1 Moduly knihovny Qt

Samotný framework je uspořádán do několika modulů. Zde bude následovat jejich popis viz. obrázek 2.6:

Core – Tento modul tvoří základ všech *Qt* aplikací. Obsahuje objekty pro vstupní a výstupní operace, pro správu vláken, zásuvných modulů, nastavení a komunikaci pomocí *signálů* a *slotů*.

GUI – Modul obsahuje potřebné funkce pro vytváření grafických aplikací s uživatelským rozhraním. *Qt* se snaží pro každou platformu plně využít systémových prostředků. Uživatelské rozhraní lze také modifikovat a vytvářet tak unikátní vzhled aplikací.

OpenGL – Modul nabízí třídy pro práci s 3D grafikou s podporou *OpenGL* a *OpenGL ES*. *OpenGL* je grafická knihovna, která umožňuje vytvářet multi-platformní aplikace s podporou grafického hardweru.

WebKit – Tento modul obsahuje open source webový prohlížeč *WebKit*. *Qt WebKit* poskytuje prohlížeč *HTML*, který umožňuje snadno vložit obsah webových stránek do aplikací.

Scripting – Obsahuje plně integrovaný, *ECMA* standard skriptovací engine založený na *JavaScriptCore*. *Qt Script* poskytuje integraci skriptování mezi *C++* a *JavaScript* za pomocí signálů a slotů.

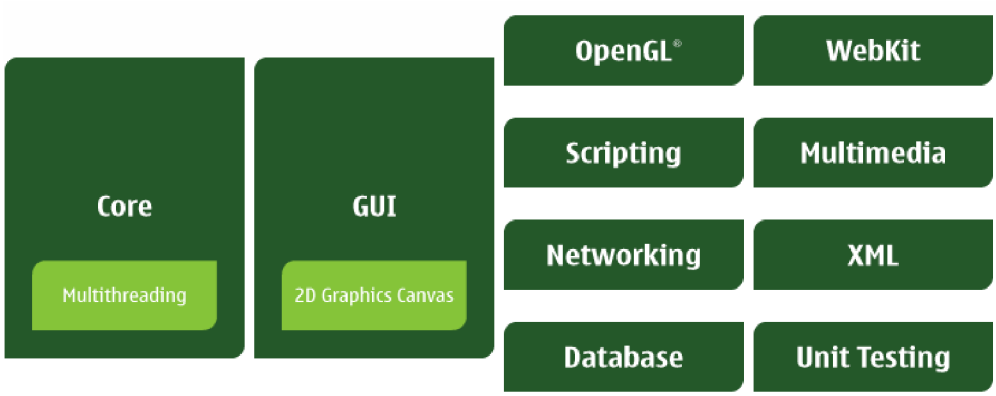
Multimedia – Modul obsahuje multy-platformní multimediální framework *Phonon*, který umožňuje používání audio a video obsahu v aplikacích.

Networking – Obsahuje funkce a objekty pro jednodušší síťové programování. Implementuje známé protokoly jako *HTTP*, *FTP* a *DNS*, včetně podpory pro asynchronní *HTTP 1.1*.

XML – Modul obsahuje objekty pro čtení a zápis *XML* dokumentů, *C++* implementace *SAX*, *DOM*, *XPath* a *XQuery engine*.

Database – Obsahuje objekty pro integraci databáze do aplikace. Podporuje známe databázové systémy a umožňuje posílat *SQL* dotazy nebo vytvářet třídy, které tyto dotazy vytvářejí automaticky.

Unit Testing – Obsahuje funkce pro běžné Unit testování a také rozšíření pro testování grafických uživatelských rozhraní.

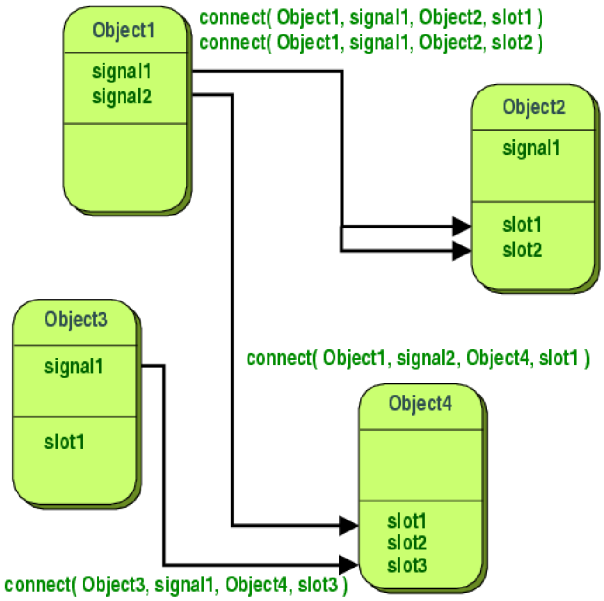


Obrázek 2.6: Moduly knihovny *Qt*, převzato z [9]

2.3.2 Signály a sloty

Mechanismus signálu a slotů je jedna z vlastností, které *Qt framework* odlišuje od ostatních frameworků. V *GUI* programování potřebujeme zařídit komunikaci mezi objekty, aby si mohly předávat zprávy a dát ostatním objektům vědět, že nastala změna. K tomu slouží *signály*, které jsou emitovány při konkrétní události. *Sloty* slouží pro příjem *signálů*, ale také jako běžné metody objektu.

Mechanismus *signálu* a *slotů* je typově bezpečný. Popis signálu se musí shodovat s popisem přijímajícího *slotu* a sám překladač nám může dát vědět, kdy se typy nebudou shodovat. *Signály* a *sloty* mohou mít libovolný počet argumentů jakéhokoliv typu. Na jeden slot lze připojít více *signálů* a jeden signál může být připojený k více *slotům*, jak znázorňuje obrázek 2.7.



Obrázek 2.7: Propojení signálu a slotů, obrázek převzat z [16]

2.3.3 Resource System

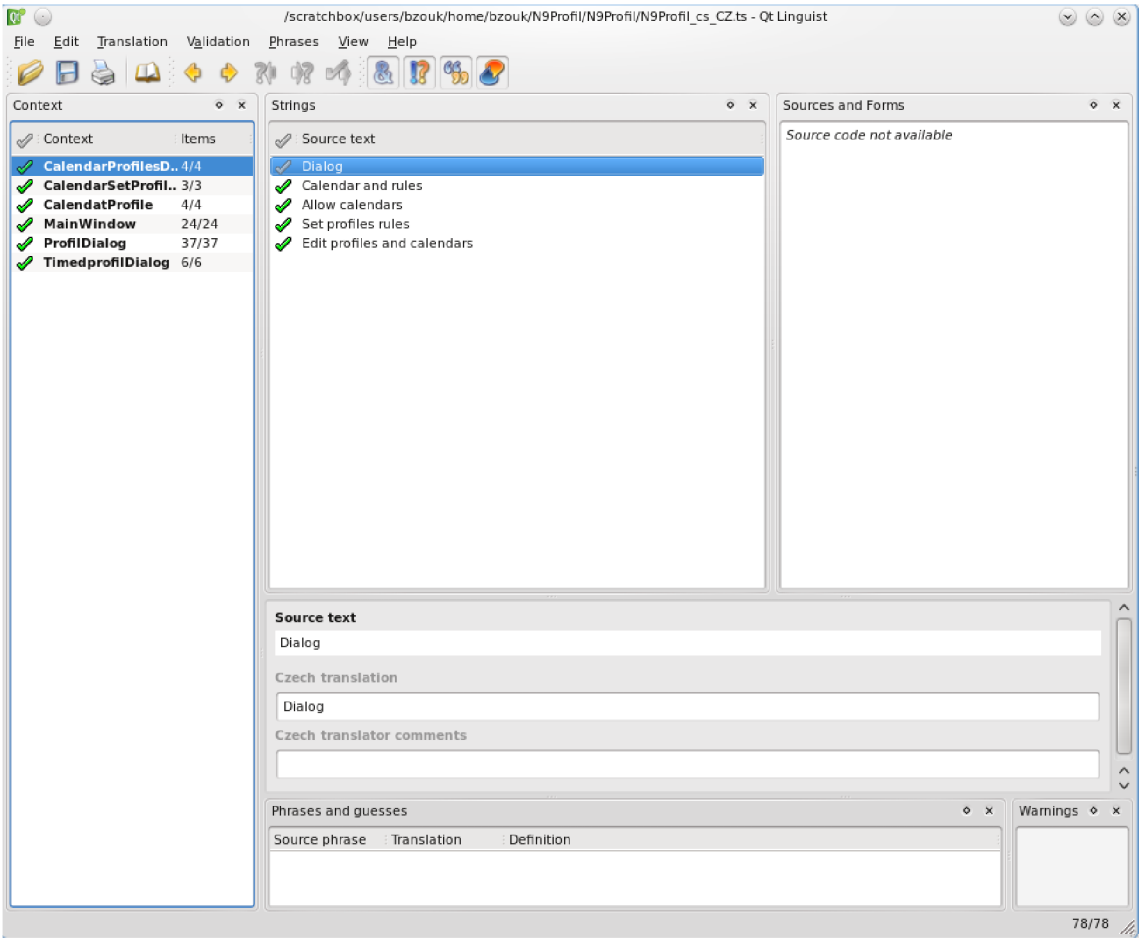
Qt obsahuje platformě nezávislý systém pro ukládání binárních nebo textových souborů do spustitelných aplikací. Toto je užitečné, pokud aplikace vždy potřebuje přístup k určitým souborům jako jsou například ikony, soubory s překladem, hudební soubory atd. K takto vloženým souborům má aplikace přístup přes třídy knihovny *Qt*. Překladač *rcc* načte soubor s příponou *.qrc,* který obsahuje seznam souborů k převedení, příklad obsahu souboru je zobrazen v kódu 2.1. Výhodou tohoto systému je menší potřeba instalace souborů, ale na druhou stranu se tímto zvětšuje velikost spustitelného souboru. Také je potřeba znovu zkompilovat aplikaci při změně jednoho ze souborů.

Kód 2.1: Obsah souboru s příponou .qrc

```
<RCC>
    <qresource prefix="/">
        <file>translations/N9Profil_cs_CZ.qm</file>
    </qresource>
</RCC>
```

2.3.4 Internacionalizace

Qt framework umožňuje lokalizaci aplikací a podporuje mnoho jazyků. *Qt* využívá *Unicode*, takže nezáleží na tom, jaký jazyk používáme pro uživatelské rozhraní, aplikace bude podporovat všechny uživatele. Nestačí pouze aby uživatel mohl psát ve svém jazyce, ale také aby aplikace byla přeložena. *Qt* toto ulehčuje, stačí všechny textové řetězce pro překlad zabalit do metody `tr()` a pomocí nástrojů *lupdate* vytvořit soubor obsahující text pro překlad. *GUI* aplikace *Qt Linguist* na obrázku 2.8 umožňuje daný text přeložit a za pomocí nástroje *lrelease* vytvořit binární soubor, který je aplikací načítán.



Obrázek 2.8: Qt Linguist

V této části jsem vycházel z knihy[21] a dokumentace [15].

Kapitola 3

Návrh aplikace

V této kapitole jsou nejprve upřesněny požadavky na samotnou aplikaci, dále na využití technologií a *API*, které budou sloužit ke splnění těchto požadavků. Zaměřím se na rozdíl mezi nimi, na jejich klady a zápory s ohledem na využití v demonstrační aplikaci.

3.1 Specifikace požadavků

Aplikace si klade za cíl inteligentně nastavovat individuální vyzvánění na platformě *Maemo*. Vyzvánění na této platformě je řešeno pomocí profilů. V systému však zcela chybí možnost přidání nových profilů a i samotná úprava stávajících je omezena.

Aplikace tedy bude umožňovat vytváření nových profilů, jejich úpravu a mazání s jednoduchým ovládáním pomocí dotykové obrazovky a také jejich jednoduchou změnu. Automatická změna profilů bude realizována těmito způsoby:

Na základě událostí obsažených v kalendáři – Aplikace bude obsahovat seznam slov pro každý profil, které bude vyhledávat v popisu jednotlivých událostí. Pokud nalezne shodu v popisu události, tak nastaví příslušící profil. Aplikace bude preferovat profil s více nalezenými slovy, nebo s vyšší prioritou.

Nastavení profilu na základě označení sítě nebo lokace – Nastavení profilu na základě jména WiFi sítě, na které je zařízení připojeno, nebo číselného označení lokace, kde se zařízení nachází. Zde se při výběru profilu, bude preferovat pravidlo se jménem WiFi sítě.

Manuální nastavení profilu na určitý čas – Vybraný profil bude nastaven na čas vybraný uživatelem.

Změna na základě čísla příchozího profilu – Uživatel si vytvoří pravidla pro telefonní čísla. Při příchozím hovoru aplikace vyhledá pravidla a nastaví příslušný profil. Po ukončení hovoru se nastaví původní profil.

Tyto čtyři způsoby nastavení mohou způsobit kolize. Mohla by nastat situace, při které je mobilní zařízení připojeno k WiFi síti, pro které má vytvořeno pravidlo a zároveň by v kalendáři nastala událost, pro kterou by měl vytvořeno jiné pravidlo. Nejvyšší prioritu tedy bude mít profil na základě příchozího hovoru, dále časový profil, menší prioritu bude mít pak profil na základě sítě, dále profil na základě popisu událostí v kalendáři a nejnižší základní profil. Aplikace by měla být uživatelsky přívětivá a bude ji možno schovat na pozadí.

3.2 Programovací jazyk

Platforma *Maemo* podporuje různé druhy programovacích jazyků a výběr příslušného jazyka pro implementaci je jedna z důležitých částí návrhu naší aplikace. Platforma *Maemo* podporuje tyto jazyky:

- jazyk C
- jazyk *Python*
- jazyk *C++*

Jazyk *C* je primárně určen k tvorbě knihoven, kdy se využije rychlost tohoto jazyka, ale není určen pro tvorbu grafických uživatelských rozhraní.

Maemo také umožňuje použití skriptovacího jazyka *Python*. *Python* je dynamický objektově orientovaný programovací jazyk. Silně podporuje integraci s jinými jazyky a nástroji. *Maemo* obsahuje speciální balík *PyMaemo*, což je distribuce jazyka *Python* a jeho knihoven na *Maemo* platformě. Balík *PyMaemo* je komunitně podporován a není oficiální částí Maemo *SDK* a také neobsahuje všechny balíčky, jako jsou verze pro personální počítače. Více na [13].

Dalším programovacím jazykem je jazyk *C++*. S využitím programovacího jazyka *C++*, lze vytvořit bohaté GUI aplikace za použití knihoven *Qt*, pro systém *Maemo*. Větší část rozhraní a knihoven tohoto systému je postavena také na jazyce *C++*. Na základě těchto podkladů jsem se rozhodl pro implementaci zvolit právě programovací jazyk *C++*.

3.3 Nastavení profilu

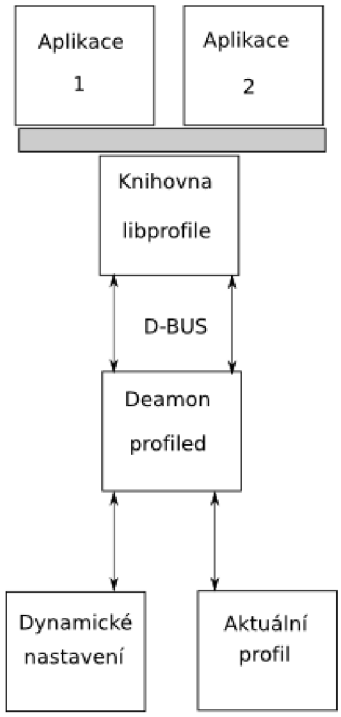
Hlavní funkcí aplikace je měnit nastavení vyzváněcích profilů v systému *Maemo* 5. K tomuto účelu slouží *profiled* démon, kterému lze posílat zprávy za pomocí knihovny *libprofile* [5]. Aplikace díky knihovně, která komunikuje přes *D-Bus* s démonem *profiled*, mohou měnit nastavení profilu. Komunikaci zobrazuje obrázek 3.1. Démon *profiled* neumožňuje vytvářet nové profily, pouze měnit nastavení již obsažených profilů. Systém *Maemo* obsahuje dva základní profily *general* a *silent*. Pro dosažení změny v nastavení profilu, bude aplikace měnit hodnoty profilu *general*, který musí být nastaven jako aktivní profil v mobilním zařízení.

Knihovna *libprofile* obsahuje funkci pro získání jmen všech profilů `profile_get_profiles()`, které jsou obsaženy v mobilních zařízení. Voláním funkce `profile_get_values()` lze získat seznam patnácti hodnot pro každý profil. Samotné hodnoty jsou uloženy ve struktuře `profileval_t`, která je zobrazena v tabulce 3.1.

Tabulka 3.1: Struktura profival_t

Jméno hodnoty	Popis
pv_key	Klíč hodnoty
pv_val	Hodnota
pv_type	Typ hodnoty

U profilu je možno nastavit zapnutí vibrací, alarmu u budíku a alarmu u událostí v kalendáři. Pro příchozí hovory, SMS, IM a E-mail zprávy, lze nastavit hlasitost vyzvánění a audio soubor pro přehrání. Také umožňuje nastavit tři úrovně zvuků pro systém, klávesnici



Obrázek 3.1: Knihovna libprofile a deamon profiled, inspirováno z [5]

a dotykový displej. Profily ještě obsahují klíč hodnoty `ringing.alert.type`, který může obsahovat hodnoty „Ringing“ a „Silent“. Pomocí hodnoty „Silent“, lze dosáhnout úplného vypnutí zvuku, nezávisle na nastavení ostatních hodnot. Příklad dat získaných z *profiled* démonu pro profil *general*, viz tabulka 3.2.

Tabulka 3.2: Profil získaný z profiled deamon

Klíč hodnoty	Hodnota	Typ hodnoty
calendar.alarm.enabled	On	BOOLEAN
clock.alarm.enabled	On	BOOLEAN
email.alert.tone	/usr/share/sounds/Message3.acc	SOUNDFILE
email.alert.volume	50	INTEGER 0-100
im.alert.tone	/usr/share/sounds/Message4.acc	SOUNDFILE
im.alert.volume	91	SOUNDFILE
keypad.sound.level	0	INTEGER 0-2
ringing.alert.tone	/usr/share/sounds/NokiaTune.acc	SOUNDFILE
ringing.alert.type	ringing	STRING „Ringing“ „Silent“
ringing.alert.volume	37	INTEGER 0-100
sms.alert.tone	/usr/share/sounds/Message3.acc	SOUNDFILE
sms.alert.volume	91	INTEGER 0-100
system.sound.level	0	INTEGER 0-2
touchscreen.sound.level	0	INTEGER 0-2
vibrating.alert.enabled	On	BOOLEAN

K nastavení hodnot slouží funkce `profile_set_value(profile, key, value)`, která

potřebuje jméno profilu, klíč hodnoty a samotnou hodnotu.

3.4 Uložení profilů a pravidel

Aplikace musí ukládat své profily a pravidla do některého z úložišť. Není potřeba ukládat velké množství dat a samotná data mají textovou formu. Proto budeme uvažovat o těchto dvou úložištích:

1. Databáze
2. XML dokument

Maemo 5 obsahuje databázi *SQLite* (viz. podsekce 2.1.3). Databáze by dostatečně splňovala požadavky na uložení profilů, ale vzhledem k malému množství dat není výhodné použít databázi pro potřeby naší aplikace.

Další ze způsobů je možnost využít značkovací jazyk, který je založen na jazyce XML. Tento jazyk umožňuje tvorbu jednoduchých dokumentů k ukládání textových dat. Tento způsob jsem nakonec zvolil. Profily jsou uloženy v souboru *profiles.xml*, který je v domovském adresáři ve složce *NProfile*. Tento soubor lze zkopírovat a využít profily na jiném mobilním zařízení. Dokument bude obsahovat elementy, které ponesou jméno profilu a budou obsahovat zanořené elementy se jménem hodnoty, atributem s typem hodnoty a textem s obsahem hodnoty. Příklad profilu můžete vidět v kódu 3.1.

V hlavičce tohoto dokumentu je obsažena informace o verzi standardu XML. V tomto dokumentu je to verze 1.0. Dále je zde informace o kódování UTF-8, kdy každý znak je uložen na 8-bitech. DOCTYPE říká jakého je dokument typu, zde je to *NProf*. Kořenový element má název *NProf* a má atribut version, jehož hodnota slouží aplikaci ke kontrole dokumentu. Také obsahuje další vnořené elementy. Tyto elementy nesou jména profilů, jako například *general* a *silent*. Každý z těchto elementů obsahuje 15 dalších zanořených elementů. Každý je pojmenován podle jména hodnoty z *profiled* démonu, má atribut value, který označuje typ hodnoty a každý z nich obsahuje textovou hodnotu. Tento dokument jsem vytvořil na základě struktury dat získané z *profiled* démona, viz tabulka 3.2.

Kód 3.1: Dokument pro ukládání profilů

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE NProf>
<NProf version="1.0">
 <general>
  <calendar.alarm.enabled type="BOOLEAN">1</calendar.alarm.enabled>
  <clock.alarm.enabled type="BOOLEAN">1</clock.alarm.enabled>
  <email.alert.tone type="SOUNDFILE">NokiaTune.aac</email.alert.tone>
  <email.alert.volume type="INTEGER 0-100">100</email.alert.volume>
  <im.alert.tone type="SOUNDFILE">NokiaTune.aac</im.alert.tone>
  <im.alert.volume type="INTEGER 0-100">100</im.alert.volume>
  <keypad.sound.level type="INTEGER 0-2">2</keypad.sound.level>
  <ringing.alert.tone type="SOUNDFILE">NokiaTune.aac</ringing.alert.tone>
  <ringing.alert.type type="STRING 'Ringing' 'Silent'">
      ringing</ringing.alert.type>
  <ringing.alert.volume type="INTEGER 0-100">100</ringing.alert.volume>
  <sms.alert.tone type="SOUNDFILE">NokiaTune.aac</sms.alert.tone>
  <sms.alert.volume type="INTEGER 0-100">100</sms.alert.volume>
  <system.sound.level type="INTEGER 0-2">2</system.sound.level>
  <touchscreen.sound.level type="INTEGER 0-2">2</touchscreen.sound.level>
  <vibrating.alert.enabled type="BOOLEAN">1</vibrating.alert.enabled>
```

```
 </general>
</NProf>
```

Pro uložení pravidel jsem také zvolil XML dokument, který je vidět v kódu 3.2. Dokument je uložen v souboru *cellidwifinum.xml*, který se nachází ve stejné složce, jako dokument pro uložení profilů. Stejně jako v předešlém dokumentu se jedná o standard XML verze 1.0 s kódováním UTF-8. Obsahuje tři hlavní elementy, které mají vnořeny elementy pro uložení jednotlivých pravidel. Tyto tři elementy nesou název podle druhu pravidel, které obsahují. *WIFI* slouží pro uložení pravidel na základě jména WiFi sítě, *LOCID* pro identifikační číslo lokace a *TELENUM* pro telefonní číslo. Jednotlivé podelementy reprezentují jednotlivá pravidla. Každý z těchto elementů má tyto tři synovské elementy. *Name* obsahuje jméno pravidla. Element *id* slouží k uložení hodnoty pravidla, podle kterého se nastaví profil uložený v elementu *profile*.

Kód 3.2: Dokument pro ukládání pravidel

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE NProfIDWifi>
<NProfIDWifi version="1.0">
 <WIFI>
  <rule0>
   <name>Moje sit</name>
   <id>BZ</id>
   <profile>general</profile>
  </rule0>
 </WIFI>
 <LOCID>
  <rule0>
   <name>Doma</name>
   <id>1719</id>
   <profile>silent</profile>
  </rule0>
 </LOCID>
 <TELENUM>
  <rule0>
   <name>Prvni cislo</name>
   <id>+420111111111</id>
   <profile>silent</profile>
  </rule0>
  <rule1>
   <name>Druhe cislo</name>
   <id>+420666666666</id>
   <profile>silent</profile>
  </rule1>
 </TELENUM>
</NProfIDWifi>
```

3.5 Uložení nastavení programu

Pro účel ukládání nastavení aplikace není použit XML dokument, ale knihovny *Qt*. Ty obsahují třídu `QSettings` pro ukládání nastavení. Třída umožňuje ukládat data pomocí metody `setValue(key, value)` a metodou `value( key, defaultValue)` je zpět získat. Metody využívají textového klíče `key` k přístupu a ukládání hodnot.

Aplikace si pomocí těchto metod také ukládá jméno základního profilu, který je aktuálně nastaven v aplikaci. Tento přístup je také využit pro ukládání slov a priorit k nastavení

profilů za pomocí událostí z kalendáře. Toto řešení umožňuje jednoduchý přístup k hodnotám a jejich ukládání díky tomu, že slova a priorita jsou vázány na jméno profilu. Klíč se tedy bude skládat ze jména profilu a slova *words* nebo *priority* pro přístup k jednotlivým hodnotám pro každý profil.

3.6 Získání informací z kalendáře

Zaměřme se teď na nastavení profilů za pomocí událostí z kalendáře. K získání informací o kalendářích je zde knihovna *calendar-backend* [1]. Tato knihovna poskytuje *C++ API* pro přístup externích aplikací k datům kalendáře. *API* umožňuje získat nebo uložit data do databáze kalendáře. Při změně provedené v databázi kalendáře knihovna vyšle signál pomocí rozhraní *D-Bus*, informující o změně v databázi. V tabulce 3.3 je vidět, že se jedná o signál `dbChange`. Aplikace tedy bude naslouchat tomuto signálu, aby si mohla stáhnout nová data.

Tabulka 3.3: Signál o změně v databází kalendáře.

	Hodnota
Cesta	/com/nokia/calendar
Jméno rozhraní	com.nokia.calendar
Signál	dbChange

Knihovna poskytuje třídu `CMulticalendar`. Voláním metody `getListCalFromMc()` lze získat seznam všech kalendářů, které jsou obsaženy v databázi. Třída `CMulticalendar` také obsahuje metodu `getComponents()`, pro získání událostí jednotlivých kalendářů. Aplikace si bude ukládat informace o jednotlivých událostech jako je identifikační číslo, její popis a začátek a konec samotné události.

3.7 Telefonní síť a WiFi

Informace o telefonní síti lze získat pomocí těchto způsobů.

1. Pomocí knihovny *liblocation* [18]
2. Pomocí dotazu přes *D-Bus*

Liblocation primárně slouží k získávání pozice z *GPS* modulu nebo pomocí telefonní sítě. Knihovna poskytuje strukturu `gsm_cell_info`, která obsahuje informace o gsm buňce, ve které je obsaženo identifikační číslo vysílače, aktuální mobilní kód země, aktuální kód mobilní sítě a kód oblasti. *Liblocation* je robustní řešení, které se pro naši aplikaci přímo nehodí.

Získat informace o telefonní síti, lze také pomocí systému *D-Bus*. K tomuto účelu slouží metoda *get_registration_status* [11]. Tabulka 3.4 ukazuje všechny potřebné údaje pro získání informací přes systém *D-Bus*. Metoda *get_registration_status* vrací informace o stavu mobilního připojení, kód oblasti, čísla vysílače, aktuální mobilní kód země a aktuální kód mobilní sítě.

Tabulka 3.4: Rozhraní pro informace o telefoní síti

	Hodnota
Služba	com.nokia.phone.net
Cesta	/com/nokia/phone/net
Rozhraní	Phone.Net
Jméno metody	get_registration_status

Služba `com.nokia.phone.net` také obsahuje signál `registration_status_change`[3], který se přes *D-Bus* vyšle pokud nastala změna. Signál obsahuje stejně údaje jako metoda `get_registration_status`. Rozhraní pro signál o změně informací o telefonní síti je uvedeno v tabulce 3.5.

Tabulka 3.5: Rozhraní a signál pro změnu v telefonní síti

	Hodnota
Služba	com.nokia.phone.net
Cesta	/com/nokia/phone/net
Rozhraní	Phone.Net
Signál	registration_status_change

Pro zjištění názvu WiFi sítě je možno využit systém *Maemo*, který je založený na jádru *Linux* (viz. podsekce 2.1.3. Systém v cestě */sys/class/net* vytváří složky, jenž nesou jména síťových rozhraní, obsažených v systému. Pokud složka obsahuje podsložku *wireless*, jedná se o bezdrátové připojení. Pokud je jméno tohoto připojení obsaženo v routovací tabulce je systém připojen na WiFi síť.

3.8 Příchozí hovor

K nastavení profilu na základě čísla musí být aplikace upozorněna o příchozím hovoru. K tomuto účelu slouží *D-Bus* signál `Coming` [10] pro upozornění na příchozí hovor, který nese jeho telefonní číslo. Na tento signál aplikace bude reagovat a pokusí se vyhledat příslušné pravidlo. Aplikace se také musí dozvědět, zda byl telefonní hovor ukončen, pro nastavení předešlého profilu. Pomocí metody `GetStatus` [19] aplikace získá informaci zda je hovor aktivní. V tabulce 3.6 je vidět rozhraní pro signál `Coming` a v tabulce 3.6 pro metodu `GetStatus`.

Tabulka 3.6: Rozhraní pro signál Coming a metodu GetStatus

	Hodnota
Služba	com.nokia.csd.Call
Cesta	/com/nokia/csd/call
Rozhraní	com.nokia.csd.Call
Signál	Coming

	Hodnota
Služba	com.nokia.csd.Call
Cesta	/com/nokia/csd/call/1
Rozhraní	com.nokia.csd.Call.Instance
Jméno metody	GetStatus

Kapitola 4

Implementace

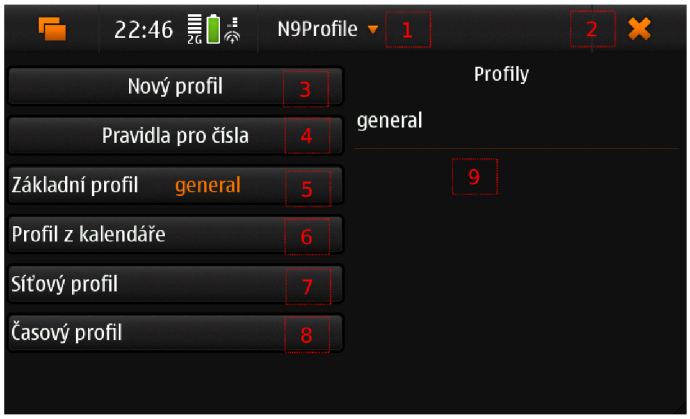
V této kapitole jsou shrnuty vývojové prostředky, dále popis grafického rozhraní aplikace a také implementace aplikace.

4.1 Vývojové prostředky

Samotná aplikace je naprogramována za použití jazyka *C++* s využitím knihoven *Qt* 4. Vývoj aplikace probíhal za použití vývojového prostředí *Qt Creator*. Toto prostředí neumožňuje překlad aplikací pro systém *Maemo*. Překlad tedy probíhal v prostředí *Scratch*box (viz. podsekce 2.2.1). Jeden z požadavků vycházející ze specifikace byla uživatelská přívětivost aplikace. Knihovny *Qt* pro systém *Maemo* obsahují modul pro speciální uživatelské rozhraní, které například ulehčí výběr ze seznamu prvků, data nebo času. Tento modul je využit pro zobrazení seznamu profilů a jeho následného výběru. Samotné testování aplikace probíhalo v prostředí *Scratchbox* a také na referenčním mobilním telefonu *Nokia N900*.

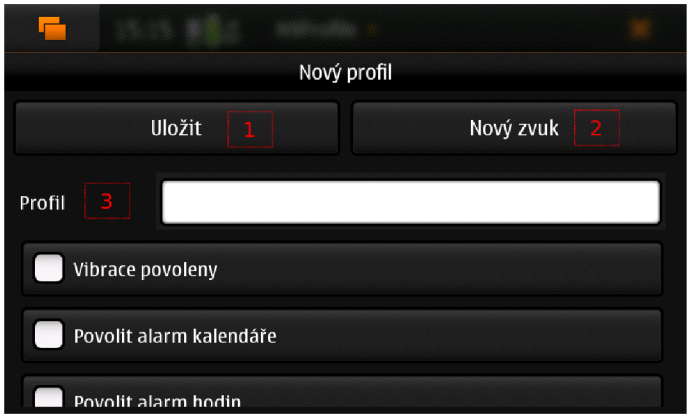
4.2 Grafické rozhraní aplikace

Hlavní okno aplikace je zobrazeno pomocí třídy `MainWindow`. Grafické rozhraní můžete vidět na obrázku 4.1. V horní části je obsaženo menu (č. 1), které obsahuje tlačítka pro ukončení programu a zobrazení informací o programu. Tlačítko (č. 2) v pravém horním rohu, umožňuje schovat aplikaci na pozadí.



Obrázek 4.1: Hlavní rozhraní aplikace

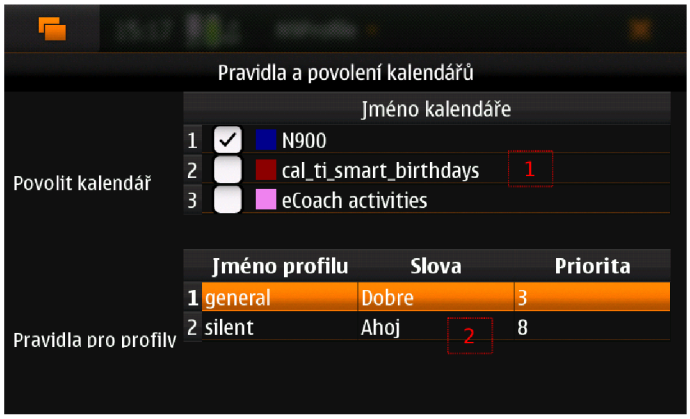
Tlačítka v levé části rozhraní aplikace slouží k tvorbě nového profilu (č. 3) a také pro zobrazení jednotlivých dialogů. Tlačítka pro nastavení profilu (č. 5, 6, 7, 8) , kromě vytvoření nového profilu (č. 3) a pravidla pro telefonní čísla (č. 4), jsou založena na třídě `QMaemo5ValueButton`, která umožňuje zobrazit název vybraného profilu v textu tlačítka. V pravé části se nachází seznam všech profilů (č. 9). Po kliknutí na jméno profilu se zobrazí dialog s volbou pro smazání profilu nebo jeho úpravy.



Obrázek 4.2: Vytvoření nebo modifikace profilu

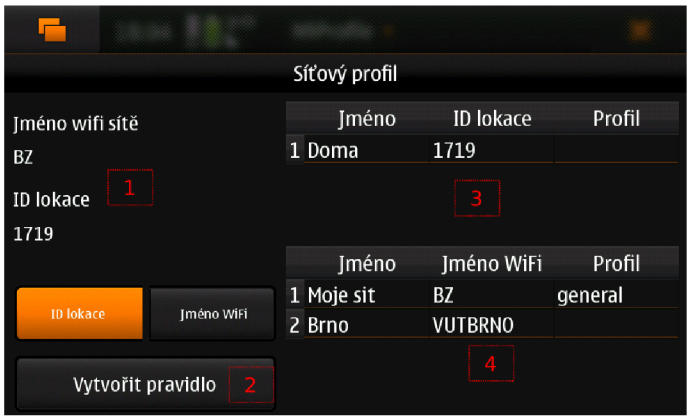
Pro vytvoření nového profilu slouží dialog *Nový profil*, který je k vidění na obrázku 4.2. Obsahuje ovládací prvky pro nastavení jednotlivých hodnot profilu (č. 3). Uživatel musí vyplnit název profilu a také vybrat audio soubory, aby mohl být nový profil vytvořen. Tlačítko *Nový zvuk* (č. 2) slouží pro přidání nových audio souborů, které si pak uživatel může přidat do profilu. Tento dialog také slouží k úpravě již vytvořeného profilu. Po stisku tlačítka *Uložit* (č. 1) je nový profil vytvořen. Samozřejmostí je kontrola, zda se uživatel nesnaží přidat již vytvořený profil.

Dialog *Pravidla a povolení kalendářů* slouží k zobrazení jmen kalendářů a seznamu profilu a adekvátních slov k nim příslušícím viz. obrázek 4.3. Dialog umožňuje vybrat kalendáře (č. 1), ze kterých aplikace načte jednotlivé události. Po kliknutí na jméno profilu (č. 2) se zobrazí dialog s možností úpravy vybraného profilu. V něm lze změnit slova k vyhledání v událostech kalendáře a jeho prioritu, která má rozsah od nuly do deseti.



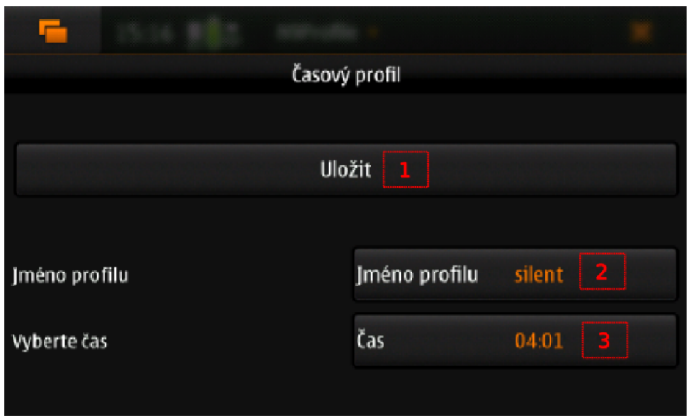
Obrázek 4.3: Nastavení profilů pro události v kalendáři

Dialog s názvem *Síťový profil*, zobrazený na obrázku 4.4 umožňuje vytvářet pravidla pro nastavení profilu na základě jména WiFi sítě (č. 4) nebo označení lokace (č. 3), kde se mobilní zařízení nachází. V levé části dialogu je zobrazen aktuální název WiFi sítě a označení lokace (č. 1). Tlačítko *Vytvořit pravidlo* (č. 2) zobrazí dialog, pro tvorbu nového pravidla. Po kliknutí na pravidlo v tabulce (č. 3, 4), lze dané pravidlo smazat nebo upravit.



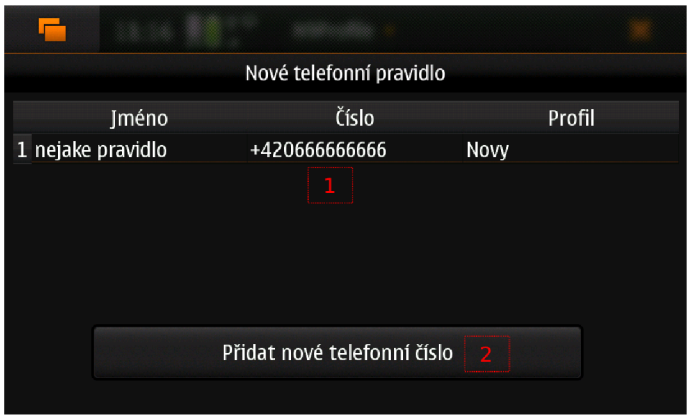
Obrázek 4.4: Dialog pro vytváření pravidel

Pro nastavení profilu na určitý čas umožňuje dialog *Časový profil*. Uživatel si může vybrat určitý profil (č. 2), který se nastaví na zadaný čas (č. 3). Po stisknutí tlačítka *Uložit* (č. 1) se profil nastaví. Při nastavení nulového času se profil nenastaví, toto může sloužit k vypnutí předchozího nastaveného profilu.



Obrázek 4.5: Nastavení časového profilu

Dialog *Nové telefonní pravidlo*, který je k vidění na obrázku 4.6, umožňuje vytvářet pravidla pro telefonní čísla příchozích hovorů. Tabulka (č. 1) zobrazuje seznam s názvy pravidel a jednotlivými telefonními čísly s přiřazenými profily. Po stisku pravidla v tabulce se zobrazí dialog pro výběr, zda dané pravidlo smazat nebo upravit. Pomocí tlačítka *Přidat nové telefonní číslo* (č. 2), lze vytvořit nové pravidlo.



Obrázek 4.6: Dialog zobrazující pravidla pro telefonní čísla

4.3 Aplikační část

Aplikaci je rozdělena do několika částí, jak naznačuje jednoduchý diagram tříd na obrázku 4.7:

- Hlavní rozhraní aplikace a vytváření profilů a jejich modifikace
- Vytváření pravidel pro události v kalendáři
- Pro nastavení profilu na určitý čas
- Profil nastavený na základě jména WiFi sítě nebo čísla lokace
- Kontrolní třída pro výběr profilu, který se bude preferovat
- Profil na základě telefonního čísla příchozího hovoru
- Část pro ukládání dat a samotné nastavení profilu do mobilního zařízení

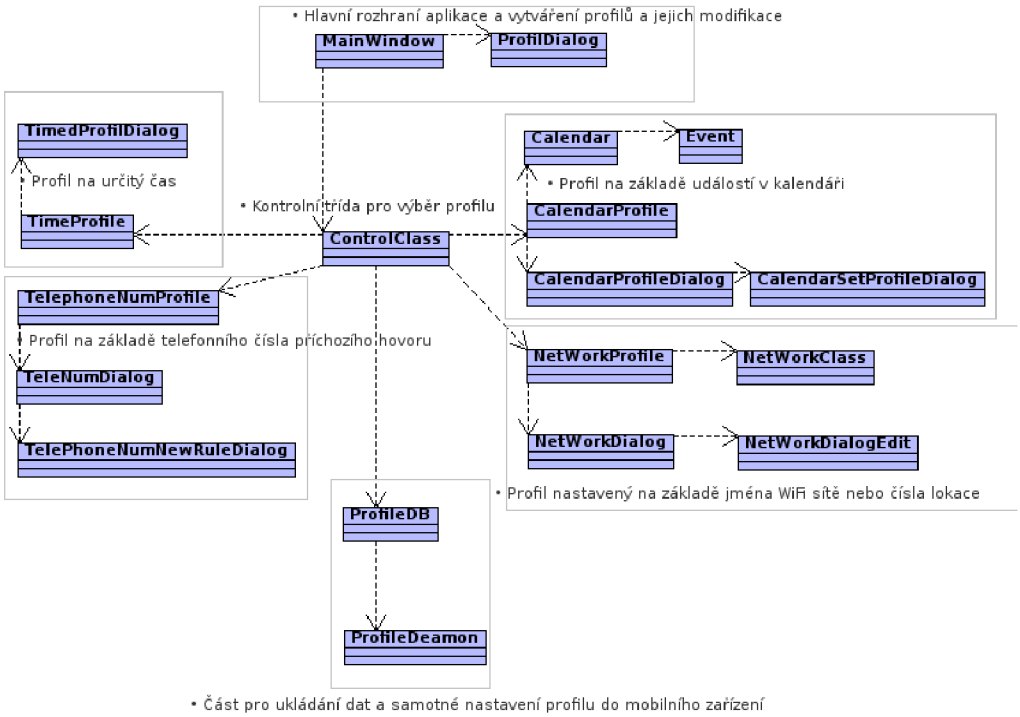
Nastavení a vytváření profilů

Třídy, které vybírají profily se jmenují `CalendarProfile`, `NetWorkProfile`, `TimeProfile` a `TelephoneNumProfile`. Tyto třídy vysílají signály nesoucí názvy profilů, které jsou určeny třídě `ControlClass`. Třída prioritně vybírá jeden z těchto profilů. Zde následuje popis hlavních tříd v aplikaci, jejich použití a důležitých metod.

`MainWindow` – Jak již bylo řečeno v sekci 4.2 pro zobrazování hlavního rozhraní slouží třída `MainWindow`. Tato třída zobrazuje seznam všech profilů v aplikaci a umožňuje jejich vytváření a modifikaci pomocí třídy `ProfilDialog`. Třída také za pomocí signálů získává jména profilů z ostatních částí aplikace, které pak zobrazuje v obsahu odpovídající tlačítek.

`CalendarProfile` – Slouží k nastavení profilu na základě popisu událostí obsažených v kalendářích . Třída načte data za pomocí metody `loadCalendarsAndEvents()` (viz. sekce 3.6), která vytvoří seznam tříd typu `Calendar` pro uložení informací o kalendářích a třídy typu `Event`, které slouží pro uložení událostí daného kalendáře. Třída `Event` využívá časovače pro upozornění, že nastala událost. Když událost nastane,

třída `Event` vyšle signál, který zachytí třída `CalendarProfile` a vybere nový profil pomocí metody `SetProfile()`. Metoda vybere všechny události, které nyní probíhají a vybere nejlepší profil na základě shodujících se slov a priority. Nakonec vyšle signál se jménem vybraného profilu. Pro profily, které mají stejné priority, se metoda snaží vybrat ten, který má více shodujících se slov. Třída také pomocí slotu `calendarTrack()` naslouchá přes rozhraní *D-Bus* o změnách v databázi kalendáře a při změně znovu načte všechna data z kalendáře.



Obrázek 4.7: Diagram tříd aplikace

`NetWorkProfile` – Pro výběr profilu na základě jména sítě slouží třída `NetWorkProfile`. Obsahuje metodu `SetProfile()`, která vybere profil na základě shodujícího se pravidla. Metoda preferuje pravidla pro WiFi připojení a po té pravidla pro označení lokace. Při změně v seznamu pravidel nebo při změně sítě je tato metoda znovu zavolána. Pro sledování změn v připojení mobilního zařízení slouží třída `NetworkClass`. Tato třída využívá *D-Bus* k zjištění změn v připojení k telefonní síti (viz. sekce 3.7). Slot `registrationStatusChanged()` umožňuje příjem těchto změn. Třída poté vyšle signál, informující o změně v označení lokace. Také je spuštěn časovač, který periodicky volá metodu `lanName()`, která vyhledá název bezdrátového rozhraní v systému a za pomocí funkce `ioctl()` zjistí název sítě. Při změně v názvu sítě, třída vyšle signál s novým jménem WiFi sítě.

`TimeProfile` – Pro nastavení profilu na určitý čas, slouží třída `TimeProfile`. Využívá třídu `QTimer`, která slouží jako časovač pro sledování času, po který je vybraný profil aktivní. Hlavní metoda pro nastavení časovače je `SetTimer()`, která zobrazí dialog pro výběr času a názvu profilu. Po nastavení času a profilu je vyslán signál se jménem vybraného profilu a nastaven časovač. Po uplynutí zadaného času je zavolán slot `TimerTimeout()`, který vyšle signál, že vybraný profil skončil.

`ControlClass` – Třída, která rozhoduje o samotném výběru profilu se jmenuje `ControlClass`. Třída přijímá signály z tříd pro nastavení profilů a vybírá profil na základě pevně dané priority. Při změně v profilu v některé z tříd je zavolána metoda `ChooseProfile()`, která vybere jediný profil a ten se následně nastaví do mobilního zařízení. Třída také ukládá název základního profilu pomocí třídy `QSettings`, jak je ukázáno v kódu 4.1.

Kód 4.1: Příklad na použití třídy `QSettings`

```
settings->beginGroup("ControlClass"); //začátek skupiny
//načtení jména základního profilu
default_profile = settings->value("DefaultProfile", "general").toString();
//konec skupiny
settings->endGroup();
```

Uložení profilů a jejich nastavení do zařízení

Ukládání a správu dat zajišťuje třída `ProfileDB`. Třída umožňuje načítat soubory s daty o jednotlivých profilech a pravidlech. Každý soubor je reprezentován třídou `QDomDocument`. Tato třída umožňuje jednoduché procházení a získávání dat z dokumentu. V kódu 4.2 je uveden příklad načtení takového souboru. Z tabulky lze vidět, že třída `QDomDocument` sama kontroluje, zda je XML dokument v pořádku. Třída `ProfileDB` také vytváří model, který obsahuje informace o audio souborech. Tento model je využit k výběru zvuku při vytváření nového profilu.

Kód 4.2: Načítání dokumentu

```
//otevření xml souboru
file.open(QIODevice::ReadWrite | QIODevice::Text)

//proměnné pro zjistění chyby v XML dokumentu
QString errorStr;
int errorLine;
int errorColumn;

//načtení dokumentu
domDocument.setContent((QIODevice *)&file, false, &errorStr, &errorLine,
    &errorColumn);
```

Důležitou metodou je `CreateProfile()`, která vytvoří nový profil, který se uloží do třídy `QDomDocument`. Pro aktualizaci profilu slouží metoda `UpdateProfile()`, pomocí níž se vyhledá v dokumentu starý profil, který se nahradí novým. Také je zde metoda `DeleteProfile()` pro smazání určitého profilu. Při ukončení aplikace se dokumenty uloží zpět do souborů. Pro nastavení profilu do mobilního zařízení slouží metoda `SetProfile()`. Metoda vyhledá profil v dokumentu a vytvoří objekt typu `Profil`, který obsahuje všechny údaje o profilu a předá jej třídě `ProfileDeamon`.

Třída `ProfileDeamon` slouží ke komunikaci s *profiled* démonem. Obsahuje důležitou metodu `SetProfile()`, která má jediný parametr a to objekt typu `Profil`, který obsahuje všechny potřebné hodnoty profilu. Metoda všechny tyto hodnoty nastaví za pomocí funkcí knihovny *libprofile* (viz. sekce 3.3). Obsahuje také metodu `GetProfile()` pro načtení hodnot profilu ze zařízení.

Lokalizace aplikace

Aplikace je lokalizována pomocí knihoven *Qt* 4 viz. podsekce 2.3.4. Pomocí knihovny, lze poměrně jednoduše lokalizovat potřebné texty, které si pak aplikace sama načítá v závislosti na aktuálním jazyku prostředí. V kódu 4.3 je zobrazen příklad načtení lokalizace, která je přibalena do spustitelnému souboru aplikace.

Kód 4.3: Překlad aplikace

```
QApplication a(argc, argv);

//Třída, která obsahuje přeložené texty
QTranslator t;

//načtení překladu
t.load(":/translations/" + a.applicationName() + "_" +
    QLocale::system().name());

//instalace překladu do aplikace
a.installTranslator(&t);
```

Zaslání aplikace na pozadí

Pro svoji funkci nemusí být aplikace neustále zobrazena, ale svoji funkci může plnit i na pozadí. K tomuto účelu je využito tlačítko v hlavním rozhraní aplikace (viz. sekce 4.2). Implementačně je to provedeno pomocí třídy `MainWindow`, která reimplementuje metodu `closeEvent()`, která primárně slouží k zavření aplikace. Metoda přepne aplikaci na pozadí. Pro opětovné zobrazení aplikace slouží metoda `top_application()`, která je vystavena na rozhraní *D-Bus*. Metoda je volána při stisku ikony aplikace v menu zařízení. Vycházel jsem z návodu z [14].

Instalace aplikace na mobilní zařízení

Aplikace je nainstalována na zařízení pomocí balíčkovacího systému, který systém *Maemo* 5 využívá (viz. podsekce 2.1.3). Tento přístup usnadní potřebnou instalaci jak samotné aplikace, tak dalších potřebných souborů pro správnou funkci aplikace. Balíček obsahuje ikony aplikace, soubor *N9Profil.desktop*, který obsahuje informace o jménu, ikoně a popisu aplikace pro systém Maemo. Aplikace bude tedy reprezentována ikonou v menu, pomocí které se bude spouštět. Dále soubor *org.indt.N9Profil.service*, který slouží systému *D-Bus* pro identifikaci služby [14]. Balíček v sobě obsahuje všechny potřebné závislosti, které je potřeba splnit, aby aplikace byla správně nainstalována.

Kapitola 5

Závěr

Cílem této bakalářské práce je vytvořit aplikaci pro inteligentní správu vyzvánění pro mobilní operační systém Maemo. Tento cíl se mi podařilo dosáhnout vytvořením aplikace nazvanou N9Profile, která splňuje na ni kladené požadavky. Výsledná aplikace poskytuje jednoduchou změnu nastavení vyzváněcích profilů, vytváření nových profilů a také dovoluje vytvářet nejrůznější pravidla pro automatizovanou změnu vyzváněcího profilu. Tyto změny jsou založeny na základě popisu událostí obsažených v kalendáři, kdy se nastaví vybraný vyzváněcí profil na základě kritérií daných uživatelem. Dále také na základě pravidel obsahující specifický název WiFi sítě, nebo označení místa, kde se mobilní zařízení nachází. Další možností je nastavení profilu na základě čísla příchozího hovoru. Aplikace také umožňuje nastavit vybraný profil na určitý čas. Aplikace tedy přináší nové možnosti změny profilů, které systém Maemo neobsahuje.

Při vývoji aplikace jsem musel nastudovat různé materiály, které mi pomohly splnit všechny požadavky kladené na aplikaci. Většina těchto materiálů pochází z internetových zdrojů. Měl jsem možnost se přesvědčit, že dokumentace pro systém Maemo a jeho knihovny není velmi pečlivě zpracována a některé informace jsem nalezl za pomocí stránek uživatelské komunity. Celá práce pro mne byla velmi přínosná. Pronikl jsem hlouběji do vývoje aplikací pro mobilní operační systémy s využitím knihoven Qt. Tyto cenné zkušenosti jsou pro mě velkým přínosem do budoucna. Trh s mobilními aplikacemi je jeden z nejrychleji rostoucích na světě a je zde mnoho možností vývoje dalších aplikací. Také jsem se blíže seznámil s knihovnami Qt, které umožňují snadnou tvorbu multiplatformních aplikací. Jejich znalost bude při řešení dalších prací jistě velkým přínosem. Samotnou aplikaci jsem přihlásil do soutěže „Vyhrajte linuxový telefon Nokia N900“ na serveru zdrojak.root.cz, kde se umístila mezi nejlepšími třemi příspěvky.

Aplikace nabízí mnohá vylepšení pro budoucí vývoj. Jednou z věci pro zlepšení uživatelského prostředí je možnost zobrazit aplikaci také na výšku. Další z věcích ke zlepšení aplikace, je přidání dalších pravidel pro nastavení profilů například na základě údajů ze senzorů obsažených v mobilním zařízení.

Literatura

[1] Data Structures. [online]. Poslední modifikace: 13.4.2010 [cit. 2010-04-20]. Dostupné na.
URL
http://maemo.org/api_refs/5.0/5.0-final/calendar-backend/index.html

[2] DBus Basics. [online]. Poslední modifikace: 12.4.2010 [cit. 2010-04-15]. Dostupné na.
URL http:
//wiki.maemo.org/Documentation/Maemo_5_Developer_Guide/DBus/DBus_Basics

[3] Dbus method call. [online]. Poslední modifikace: 21.1.2010 [cit. 2010-04-17]. Dostupné na.
URL http://talk.maemo.org/showthread.php?p=386188

[4] The Home of the Maemo Community. [online]. Poslední modifikace: 2010 [cit. 2010-04-15]. Dostupné na.
URL http://maemo.org/intro/

[5] libprofile Documentation. [online]. Poslední modifikace: 21.10.2009 [cit. 2010-04-17]. Dostupné na.
URL http://maemo.org/api_refs/5.0/5.0-final/libprofile/

[6] Linux. [online]. Poslední modifikace: 9. 4. 2010 [cit. 2010-04-20]. Dostupné na.
URL http://cs.wikipedia.org/wiki/Linux

[7] Maemo 5 Software Architecture. [online]. Poslední modifikace: 2010 [cit. 2010-04-15]. Dostupné na.
URL http://wiki.maemo.org/Documentation/Maemo_5_Developer_Guide/
Architecture/Top_Level_Architecture

[8] Maemo SDK. [online]. Poslední modifikace: 8.4.2010 [cit. 2010-04-15]. Dostupné na.
URL http://wiki.maemo.org/Documentation/Maemo_5_Developer_Guide/
Development_Environment/Maemo_SDK

[9] Modular Class Library. [online]. Poslední modifikace: 2010 [cit. 2010-04-15]. Dostupné na.
URL http://qt.nokia.com/products/library

[10] Phone call and SMS examples. [online]. Poslední modifikace: 11.02.2010 [cit. 2010-04-28]. Dostupné na.
URL http://wiki.maemo.org/Documentation/Maemo_5_Developer_Guide/DBus/
DBus_in_Freemantle

[11] Phone control. [online]. Poslední modifikace: 22.4.2010 [cit. 2010-04-27]. Dostupné na. URL http://wiki.maemo.org/Phone_control

[12] Products. [online]. Poslední modifikace: 2010 [cit. 2010-04-15]. Dostupné na. URL http://qt.nokia.com/products

[13] PyMaemo/FAQ. [online]. Poslední modifikace: 13.4.2010 [cit. 2010-04-20]. Dostupné na. URL http://wiki.maemo.org/PyMaemo/FAQ

[14] Qt application for Maemo with DBus support. [online]. Poslední modifikace: 2010 [cit. 2010-04-20]. Dostupné na. URL http://wiki.forum.nokia.com/index.php/Qt_application_for_Maemo_with_DBus_support

[15] Qt Reference Documentation. [online]. Poslední modifikace: 2010 [cit. 2010-04-17]. Dostupné na. URL http://doc.qt.nokia.com/4.6/index.html

[16] Signals and Slots. [online]. Poslední modifikace: 2010 [cit. 2010-04-15]. Dostupné na. URL http://doc.qt.nokia.com/4.6/signalsandslots.html

[17] Software Platform. [online]. Poslední modifikace: 2010 [cit. 2010-04-15]. Dostupné na. URL http://maemo.org/intro/platform/

[18] Using Location API. [online]. Poslední modifikace: 29.3.2010 [cit. 2010-04-20]. Dostupné na. URL http://wiki.maemo.org/Documentation/Maemo_5_Developer_Guide/Using_Connectivity_Components/Using_Location_API

[19] W32g. [online]. Poslední modifikace: 9.4.2010 [cit. 2010-04-28]. Dostupné na. URL http://wiki.maemo.org/User:Nbc

[20] Allen, M.: *Palm webOS*. O'Reilly Media, 2009, ISBN 0596155255, 456 s.

[21] Blanchette, J.; Summerfield, M.: *C++ GUI Programming with Qt 4*. Prentice Hall, druhé vydání, February 2008, 752 s.

[22] Burnette, E.: *Hello, Android: Introducing Google's Mobile Development Platform*. Pragmatic Bookshelf, pragmatic programmers vydání, 2009, ISBN 1934356492, 250 s.

[23] Hashimi, S. Y.; Komatineni, S.: *Pro Android*. Apress, 2009, ISBN 1430215968, 464 s.

[24] Zammetti, F. W.: *Practical Palm Pre WebOS Projects*. Apress, 2009, ISBN 1430226749, 400 s.

Dodatek A

Obsah CD

Nosič CD obsahuje:

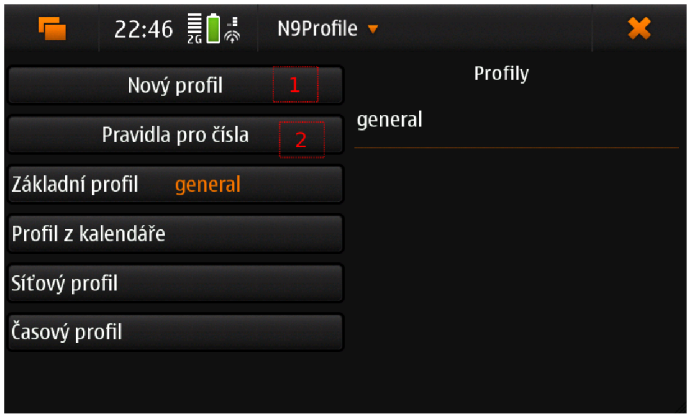
- Adresář *n9profil-0.1*, který obsahuje adresář se zdrojovými kódy aplikace a dalšími adresáři potřebných k vytvoření deb balíčku.
  Nosič cd neobsahuje žádné SDK, IDE ani knihovny, získat je lze z umístění v souboru README.txt
- Adresář *doc* obsahuje dokumentaci zdrojového kódu, která je automaticky vygenerována za pomocí nástroje Doxygen
- Adresář *text* obsahuje tuto zprávu ve formě PDF
- Adresář *latex* obsahuje zdrojové kódy pro systém LaTeX s potřebnými obrázky a kódy.
- Adresář *debian*, který obsahuje debian balíček pro instalaci programu na zařízení s Nokia N900 se systémem Maemo 5 verze PR 1.1
- Soubor *README.txt*, který obsahuje informace s odkazy ke stažení všech potřebných nástrojů a knihoven.

Dodatek B

Manual

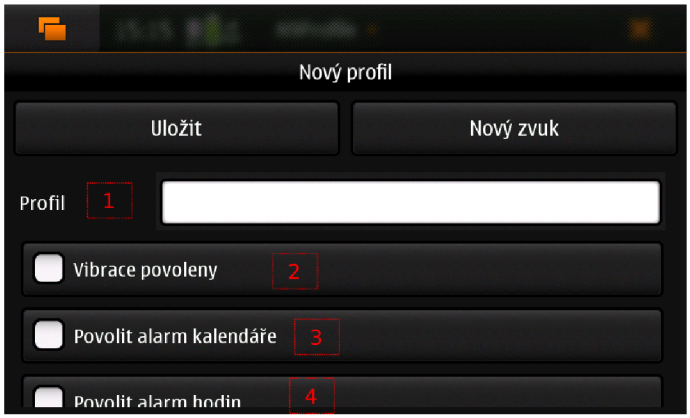
Vytvoření nového profilu na mobilním zařízení Nokia N900

K vytvoření nového profilu stiskněte tlačítko č. 1 na obrázku B.1.



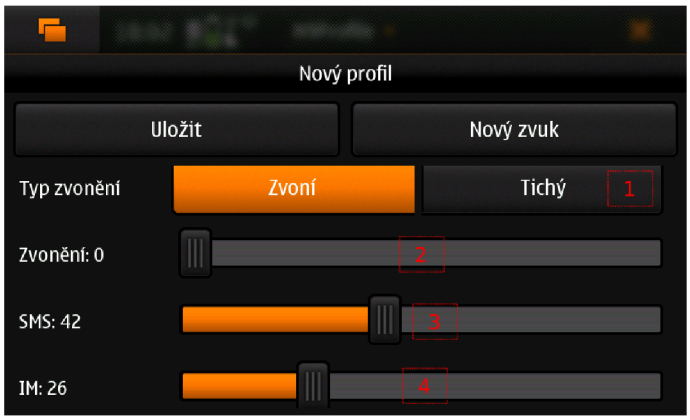
Obrázek B.1: Hlavní obrazovka aplikace

Nově otevřený dialog na obrázku B.2, obsahuje pole (č. 1) pro jméno profilu. Uživatel může zadat pouze jednoslovný název profilu. Dále obsahuje checkboxy (č. 2 3 4) pro povolení vybrací a budíku z kalendáře nebo hodin.



Obrázek B.2: Dialog pro vytvoření nového profilu, část první

V další části dialogu na obrázku B.3, si může uživatel vybrat zda bude typ profilu Zvonící nebo Tichý. Při nastavení tichého profilu mobilní zařízení nepřehraje zvuk při příchozím hovoru nebo zprávě. Pro nastavení hlasitosti slouží posuvníky (č. 2, 3, 4) pro nastavení hlasitosti při příchozí zprávě nebo volání a také na obrázku B.4 (č.1).

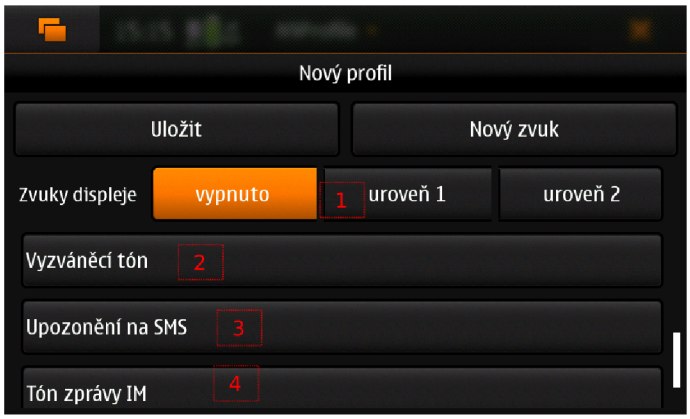


Obrázek B.3: Dialog pro vytvoření nového profilu, část druhá

Další část dialogu je zobrazena na obrázku B.4, obsahuje možnost zvolit úroveň systémových zvuků (č. 2) a úroveň zvuků při stisku klávesnice (č. 3).

Obrázek B.4: Dialog pro vytvoření nového profilu, část třetí

Dále v dialogu na obrázku B.5, lze nastavit úroveň zvuku při stisku displeje (č. 1). Tlačítka (č. 2, 3, 4) umožňují výběr audio souboru, který se přehraje při příchozím hovoru nebo zprávě.



Obrázek B.5: Dialog pro vytvoření nového profilu, část čtvrtá

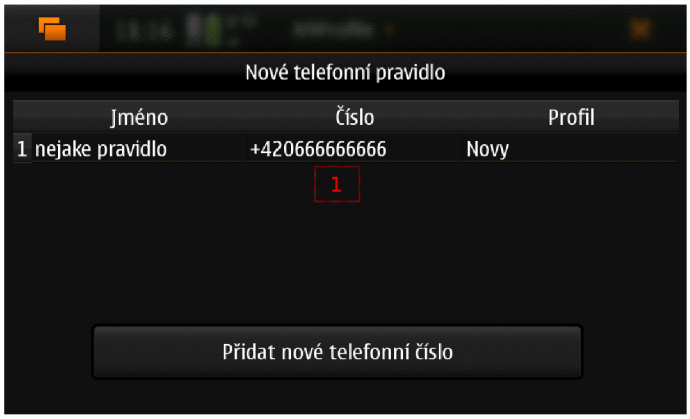
Vytvoření nového pravidla pro telefonní číslo

K zobrazení dialogu pro tvorbu pravidel na základě telefonních čísel slouží tlačítko (č. 2) na obrázku B.1. Dialog na obrázku B.6 obsahuje prázdný seznam pravidel (č. 1) a tlačítko pro tvorbu nového pravidla (č. 2). Po stisku tlačítka se zobrazí dialog pro vytvoření nového pravidla.

Obrázek B.6: Vytvoření pravidla pro telefonní číslo, část první

Dialog je zobrazen na obrázku B.7, obsahuje pole (č. 1) pro jméno pravidla a pro samotné telefonní číslo (č. 2). Tlačítko č. 3 slouží k výběru profilu. Na obrázku lze vidět, že je vybrán profil *Novy* (č. 3). Po stisku tlačítka *Uložit* (č. 4), se vybrané pravidlo uloží. Nově vytvořené pravidlo zobrazuje obrázek B.8.

Obrázek B.7: Vytvoření pravidla pro telefonní číslo, část druhá



Obrázek B.8: Vytvoření pravidla pro telefonní číslo, část třetí