VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ROZHRANÍ NETBOX MOJE KONTO PRO ANDROID

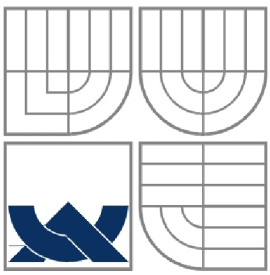
ANDROID NETBOX ”MOJE KONTO”INTERFACE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

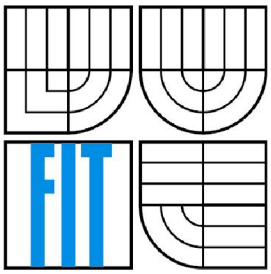
AUTOR PRÁCE
AUTHOR

RADIM VACULÍK

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ROZHRANÍ NETBOX MOJE KONTO PRO ANDROID

ANDROID NETBOX ”MOJE KONTO”INTERFACE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — RADIM VACULÍK
AUTHOR

VEDOUCÍ PRÁCE — Ing. RADEK KUBÍČEK
SUPERVISOR

BRNO 2011

Abstrakt

Tato bakalářská práce se zabývá uživatelským rozhraním pro webovou aplikaci NETBOX Moje Konto pro Android. Cílem práce je vytvořit jednoduché, přehledné a rychlé rozhraní pro zákazníky využívající služby NETBOX. Aplikaci, která nabídne uživateli souhrn svých služeb, mobilní rozhraní pro čtění zpráv, stav vyúčtování nebo přístup ke vzdálenému ovládání televize. Celá mobilní aplikace komunikuje on-line s webovým portálem NETBOX Moje Konto. Design aplikace používá návrhové vzory.

Abstract

This bachelor's thesis describes user interface of web application NETBOX Moje konto for Android. Purpose of this thesis is create simple, well-arranged and fast interface for customers, who used NETBOX's services. The application, which offers summary of components of service, mobile interface for reading messages, state of debts or remote control of TV. The whole application communicates on-line with web portal NETBOX Moje Konto. Design of appliacation uses Android design patterns.

Klíčová slova

Android, SOAP, XML, Java, Uživatelské rozhraní, Mobilní aplikace, Návrhové vzory

Keywords

Android, SOAP, XML, Java, User interface, Mobile application, Design patterns

Citace

Radim Vaculík: Rozhraní NETBOX Moje Konto pro Android, bakalářská práce, Brno, FIT VUT v Brně, 2011

Rozhraní NETBOX Moje Konto pro Android

Prohlášení

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

……

Radim Vaculík

16. května 2011

Poděkování

Rád bych poděkoval svému vedoucímu Radku Kubíčkovi za konzultace a dobré nápady při implementaci aplikace. Také bych rád poděkoval firmě SMART Comp. a.s. za možnost se podílet na tomto projektu.

Obsah

Obsah....1
1 Úvod....3
2 Android....4
2.1 Historie Android....4
2.2 Vývoj aplikací....5
2.3 Architektura systému Android....6
2.4 Struktura obecné aplikace....7
2.4.1 Zdrojové soubory aplikace....7
2.4.2 Resources....8
2.4.3 Externí knihovny....8
2.4.4 AndroidManifest....9
2.5 Chování aplikace a aktivit....10
3 UI design patterns....11
3.1 Dashboard....11
3.2 Action Bar....11
3.3 Quick Actions....12
3.4 Notification....12
3.4.1 Status Bar....12
3.4.2 ProgressDialog....13
3.4.3 Toast....13
3.4.4 Progress wheel....13
4 Návrh a analýza....15
4.1 Struktura aplikace....15
4.1.1 Koncept....15
4.1.2 Ovládání a design....16
4.2 Zdroj dat, API....16
4.2.1 SOAP....16
4.2.2 XML....17
4.2.3 JSON....17
4.3 Sezení....18
5 Implementace....20
5.1 Komunikační rozhraní – API....20
5.1.1 Server....20

5.1.2 Klientská aplikace....21
5.2 Zpracování XML....22
5.3 Zabezpečení....23
5.4 Dynamické načítaní v ListView....24
5.5 Design aplikace....24
5.6 Významné použité třídy....25
5.6.1 Downloader....25
5.6.2 MojeKonto....26
5.6.3 Preferences....26
5.6.4 R.java....27
5.6.5 Storage....27
6 Testování....28
7 Závěr....29
Použitá literatura....30
Seznam příloh....31
Příloha A – CD se zdrojovými kódy....32
Příloha B – obrazovky aplikace....33

1 Úvod

Mobilní telefony se v dnešní době staly nedílnou součástí našeho života. Za poslední desetiletí se trh s mobilními telefony velmi rozšířil. Figuruje v něm bezmála několik desítek firem, které se předhánějí ve snaze vyrobit a prodat to nejlepší zařízení. Výrobce láká potenciální zákazníky nejen na hardwarovou výbavu zařízení, ale také na softwarové vybavení. V posledních letech se rozmohl trend nasazovat do mobilních zařízení operační systémy, které uživateli dávají obrovskou možnost a volnost v práci. Telefony neslouží pouze k telefonování a psaní textových zpráv. Můžeme hrát hry, psát dokumenty, číst elektronické knihy, psát e-mailové zprávy, zapojovat se do konverzací na sociálních sítích a další. S rozvojem mobilního internetu se stává z mobilního zařízení neomezená brána informací. Hlavním důvodem je pohodlnost člověka, snaha usnadnit si práci a ušetřit čas. Za chvíli může přijít doba, kdy mobilním zařízením začneme platit – všechny nákupy budou bezhotovostní; začneme používat ve velké míře videohovory; svůj prostor dostane také například medicína, kdy se sami pomocí mobilního zařízení vyšetříme a rovnou výsledky zkonzultujeme s lékařem online; sledování online TV kdekoliv, kde je mobilní internet. Není to pěkná představa? A to vše pouze pomocí mobilního zařízení, které je již dnes dostupné pro každého.

Bakalářská práce se věnuje mobilnímu rozhraní NETBOX® Moje Konto běžícím na mobilním operačním systému Android OS[1]. Cílem práce je uživateli příjemnou, přehlednou a rychlou formou prezentovat služby, které jsou jinak dostupné pouze z webového prohlížeče. Aplikace nabízí uživateli přehled o svých službách a účtu, případně změnu nastavení služeb nebo služeb samotných.

Následující kapitola se věnuje operačnímu systému Android, jeho historii a architektuře. Také je zde vysvětlena struktura obecného projektu aplikace. Kapitola 3 se zabývá návrhovými vzory (design patterns) pro tvorbu aplikací. Jde o tzv. „best practice“, jak vytvářet uživatelsky přívětivé a líbivé aplikace. V další kapitole je diskutována problematika návrhu celé aplikace, komunikace s webovým portálem a celkové funkčnosti aplikace. Pátá kapitola seznamuje čtenáře s kompletní implementací aplikace a navazuje na předchozí kapitolu. Jsou zde rozebrány jednotlivé třídy a popsány jejich hlavní prvky. Předposlední kapitola popisuje testování aplikace na uživatelích a jejich hodnocení. Závěr práce je věnován zhodnocení celé aplikace, zpětné vazbě od uživatelů aplikace a diskuzím nad případným rozšířením aplikace o další funkcionalitu.

1 operační systém

2 Android

Android je operační systém založený na Linuxu určený převážně pro mobilní zařízení vyvíjen sdružením firem Open Handset Alliance (Google, HTC, Sony, Dell, Intel, Motorola, Qualcomm, Texas Instruments, Samsung, LG, T-Mobile, Nvidia a Wind River Systems).

2.1 Historie Android

Operační systém Android, původně vyvíjen firmou Android Inc., byl roku 2005 převzat společností Google. Ta celý projekt předala sdružení firem Open Handset Alliance (sdružení výrobců mobilních zařízení, telekomunikačních operátorů a technologických firem), která dodnes stojí za jeho vývojem. Roku 2007 byl Android oficiálně představen veřejnosti a v září dalšího roku byla vydána první veřejná verze Android 1.0 pod licencí Apache 2 a GPLv2. Jedná se tedy o open source software. [1]

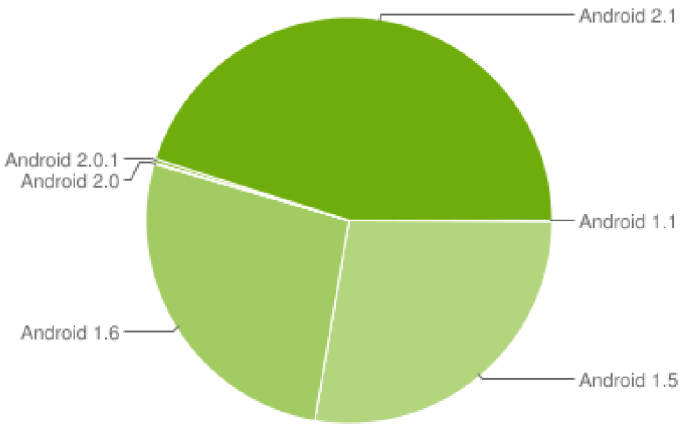
K dnešnímu dni existuje již 5 oficiálních verzí Android OS:

- První nejrozšířejší verzí byl **Android 1.5** (Cupcake) podstavený na linuxovém jádře 2.6.27 a vydán v první polovině roku 2009. Touto verzí se začal Android masivně rozšiřovat do mobilních zařízení.
- Druhou verzí se stal **Android 1.6** (Donut), který byl zveřejněn v druhé polovině roku 2009. Tato verze přinesla hlavně lepší podporu vyhledávání, nové uživatelské rozhraní kamery a galerii nebo podporu VPN.
- **Android 2.0/2.1** (Eclair) byl vydán necelé dva měsíce po Android 1.6. Během dalšího půlroku byly vydány minor verze 2.0.1 a 2.1, které opravovaly chyby. Přibyla optimalizace rychlosti hardware. S touto verzí společnost Google vydává svůj devoleper phone Google Nexus One, který má sloužit pro Android vývojáře. Tímto krokem a uvolněním verze Android 2.0 přišel největší boom v prodeji mobilních zařízení s OS Android. Podle obrázku č. 1 aktuálně nejrozšířenější verze.
- Předposlední verzí je **Android 2.2** (Froyo), který přišel s A2SD (instalace aplikací na SD kartu), JIT neboli Just-in-time kompilátor zvyšující až 5× výkon. Nesmíme opomenout také tzv. „Push notification".
- Aktuální verzí je **Android 2.3/2.4** (Gingerbread) orientovaný hlavně na podporu SIP protokolu, WebM video formátu pro HTML5 nebo NFC[2] – technologie, která dokáže kombinovat bezdrátový přenos na krátkou vzdálenost a rozhraní čipových karet. Díky této technologii lze například platit v obchodech mobilním zařízením.

2 Near Field Communication

Společnost Google již oznámila další verzi **Android 3.0** (Honeycomb), která přinese podporu více-jádrových procesorů nebo podporu pro tablety. Také je již vybráno kodové označení pro **Android 4.0** – Ice Cream. [2]

Za poznámku určitě stojí kódové označení jednotlivých verzí Androidu. Pozorný čtenář jistě zjistí, že všechna kódová označení jsou sladkosti, popř. sladká jídla. Aby se nepletlo pořadí jednotlivých verzí, určuje ho první počáteční písmeno. [3]



Obrázek 1: Podíl jednotlivých verzí Android OS. [4]

2.2 Vývoj aplikací

S každou vydanou verzí je zároveň uvolněn vývojářský kit pro Android (SDK), který obsahuje aktuální verzi, emulátor, debugger, dokumentaci, knihovny a další nástroje pro vývoj aplikací. Tyto nástroje jsou dostupné pro všechny desktopové operační systémy – Linux, Mac OS X i Microsoft Windows. Aplikace se nejčastěji vyvíjí v podporovaném prostředí Eclipse nebo Netbeans. Pomocí *Android Developement Tools Plugin* (ADT) získá vývojové prostředí všechny potřebné knihovny pro vývoj aplikace pro Android. Aplikace se píší v jazyce Java, nejedná se však o standardy typu Java SE nebo Java ME. Pomocí virtuálního stroje Dalvik se aplikace převedou z Javy do speciálního formátu Dex[3], který je paměťově i výkonově optimalizován a pak spuštěn. [5] Takto zkompilovaný kód je přibližně 3× menší a 2× rychlejší. Systém Android nabízí v základu databázi SQLite, podporu OpenGL, WebKit, SSL, grafický engine SGL a další.

Aplikace pro Android OS lze vyvíjet také na mobilním zařízení, které máme aktuálně u sebe. Pomocí ADT je možné připojit a nahrát hotovou aplikaci do zařízení. Distribuce aplikací pro ostatní uživatele zajišťuje Android Market, který je připraven ve většině Android OS. Pomocí této aplikace

3 Dalvik Executable

najdete, stáhnete a nainstalujete aplikaci. Na Android Market lze také prodávat aplikace – stačí mít platný Google Account a povolené platby. Distribuce vlastní aplikace na Android Market je velmi jednoduché – stačí vyplnit registraci a zaplatit registrační poplatek (~$20). V České republice bohužel nelze aplikace prodávat, pouze nabízet zdarma ke stáhnutí. Aktuálně Android Market nabízí přes 175 tisíc aplikací a přes 4 miliadry stáhnutí. [6]

Každá aplikace může vyžadovat pro svůj korektní běh určitá oprávnění, jejichž seznam se před instalací aplikace zobrazí uživateli, a ten musí udělit povolení. Proto je důležité tomuto kroku věnovat více pozornosti.

Pro samotné vývojáře aplikaci Android OS je také připravena skvělá dokumentace na webových stránkach projektu Android. Dokumentace obsahuje popis všech tříd, které lze použít v aplikaci. Nechybí ani názorné příklady použití v určitých situací nebo popis designu jednotlivých aktivit aplikace.

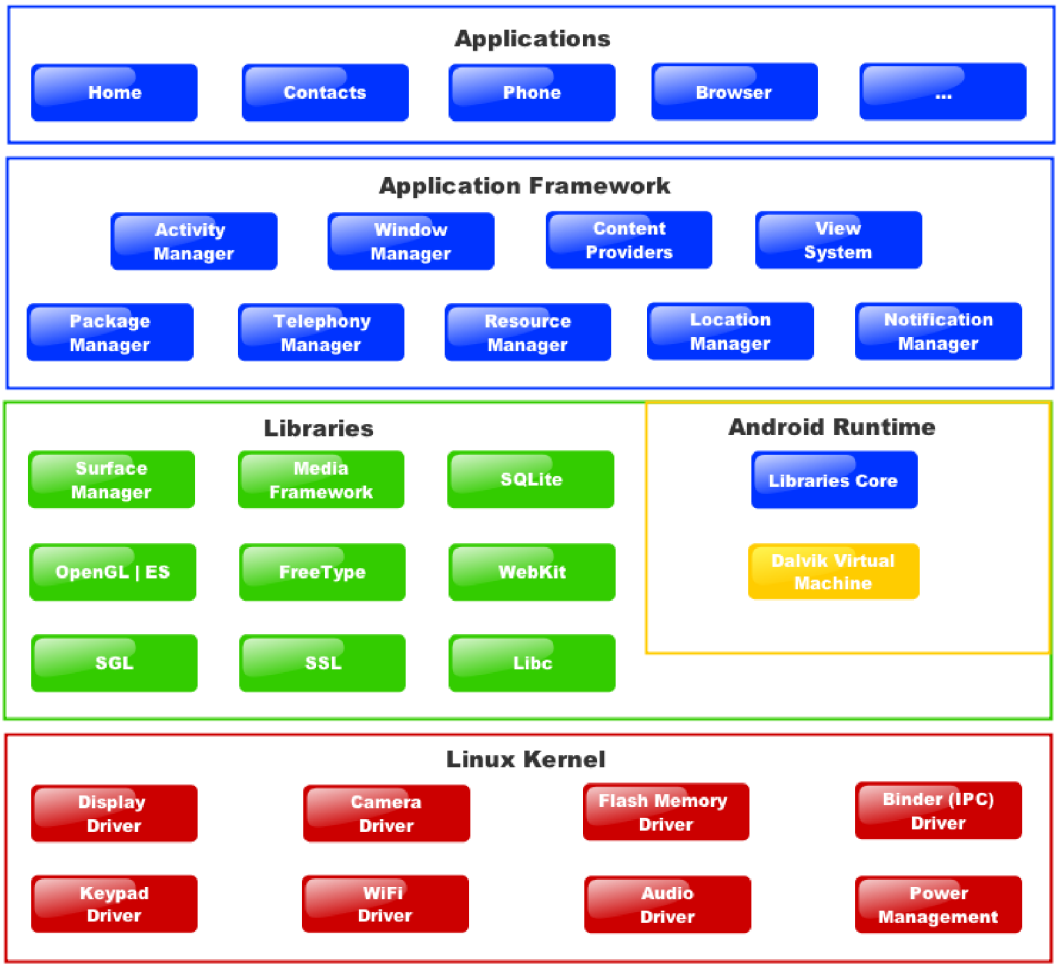
Společnost Google pořádá každoročně soutěž s názvem *Android Developer Challenge*, kde se snaží motivovat potencionální vývojáře aplikací k psaní aplikací pro Android.

Jelikož je Android open-source, začaly vznikat skupiny lidí přispívajících do projektu a jeho vývoje. Z důvodu dlouhého schvalování a nepřijetí patchů ze strany OHA začaly vznikat open-community Android projekty, které staví na základě OS Android, ale jsou značně upraveny či opraveny. Mezi velké hráče na tomto poli patří skupina *CyanogenMod* podporující přes 20 zařízení a mnohdy přichází s aktuální verzí Android OS rychleji než oficiální výrobce telefonu. Některým uživatelům starších telefonů tak nezbývá nic jiného, než využít této možnosti, protože zařízení přestane být ze strany výrobce podporováno.

2.3 Architektura systému Android

Architekturu systému Android lze rozdělit do 4 vrstev (viz. obrázek 2):

- **Linux Kernel** – nejspodnější vrstva; obsahuje upravené linuxové jádro a ovladače pro hardware (Bluetooth, Wi-Fi, GSM a další).
- **Knihovny a Dalvik VM** – knihovny komunikující s hardware a poskytující API pro další vrstvy. Patří sem také Dalvik Virtual Machine.
- **Aplikační framework** – hotové částí frameworku, které lze použít ve vlastních aplikacích. Tvoří vrstvu mezi hardware a aplikací.
- **Aplikace** – samotné aplikace, které využívají aplikačního frameworku. [7]



Obrázek 2: Architektura systému Android. [7]

2.4 Struktura obecné aplikace

Při vývoji aplikace pro OS Android se setkáme se strukturou, která nás bude provázet po celou dobu implementace aplikace. Strukturu můžeme rozdělit do 3 + 1 kategorií.

2.4.1 Zdrojové soubory aplikace

Zdrojové soubory aplikace jsou klíčové soubory obsahující definici tříd psané v jazyce Java. Tyto třídy přestavují nejen jednotlivé obrazovky (aktivity) aplikace, ale také zde lze najít helpery, wrappery[4] nebo jednoduché objekty reprezentující ucelené části kódu. Nachází se zde také speciální

4 Wrapper je objekt, který dělá prostředníka pro I/O operace. Poté lze jednoduše vyměnit za jiný, aniž by rozbil jakkoliv funkčnost programu. Helper je objekt, který zaobaluje určitou funkčnost. Používá se například při práci s databází.

soubor *R.java* obsahující reference na grafické části aplikace. Pomocí těchto referencí lze následně přistupovat k designu aplikace a ovládat jej.

2.4.2 Resources

Resources, neboli zdroje, obsahují ve stromové struktuře soubory, které nám zajišťují kompletní správu vzhledu aplikace. Lze zde nadefinovat, jak má aplikace vypadat ve vertikální nebo horizontální poloze, můžeme zde nadefinovat vzhled pro jednotlivá rozlišení mobilních zařízení. Všechny tyto vlastnosti najdeme ve složce *layout*. Nachází se zde také grafické prvky aplikace. Android dokáže zobrazovat většinu standardních typů obrázků, tyto soubory najdeme v adresáři *drawable*. Pomocí souborů z *values* lze bezproblémově lokalizovat aplikaci do jakéhokoliv jazyku, aplikace následně vybere správnou lokalizaci na základě lokalizace mobilního zařízení. Do adresáře *xml* si lze nadefinovat vlastní XML soubory, které lze použít v aplikaci. Tento typ zdroje lze chápat například jako předem nadefinované konstanty. Ve složce *styles* si může vývojář definovat vlastní styly, které lze následně uplatnit v definici designu aplikace. Funguje to na stejném principu jako CSS styly v kombinaci s HTML, styly lze v definici layoutu následně předefinovat. Tuto vlastnost lze například pěkně použít při vytváření menu, které zobrazujeme v každé aktivitě aplikace. [8]

Pokud potřebujeme vytvořit menu, lze využít adresář *menu*, kde se nadefinuje jeho struktura. Stačí do elementu *menu* naskládat elementy *item*, jak je ukázáno v kódu 1.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/delete"
          android:icon="@drawable/delete"
          android:title="Smazat" />
</menu>
```

Kód 1: Použití menu přes resource.

2.4.3 Externí knihovny

Do projektu lze také přidat externí knihovny, které jsou následně přibaleny při kompilaci k aplikaci. Lze přidat jak hotový JAR soubor, tak rozpracovaný jiný projekt. Následně lze v projektu používat objekty dané knihovny. Výhodou této možnosti je použití jiných kódů v projektech. Odpadá tak složitá implementace dalších objektů. Nevýhodou linkování externích knihoven do projektu je jejich velikost. I když použijeme pouze jednu funkcionalitu knihovny, je nutné přibalit celou knihovnu.

2.4.4 AndroidManifest

AndroidManifest je zvláštní soubor v projektu aplikace Android. Obsahuje kompletní definici, jak bude výsledná aplikace vypadat (viz Kód 2). V tomto souboru se určuje jméno aplikace, přesný název balíku, interní verze nebo číslo verze, která se zobrazuje uživatelům. Můžeme zde také definovat seznam podporovaných rozlišení mobilních zařízení. Nezbytnou součástí je seznam všech aktivit a jejich název. Dále je nutné specifikovat, která aktivita je hlavní a aplikace ji má spustit jako první. Nesmí chybět seznam práv (povolení) sloužící k přístupu k soukromým datům v mobilním zařízení, který během instalace aplikace uživatel schvaluje. Patří sem například přístup k internetu, přístup na SD kartu zařízení, přístup do seznamu kontaktů a další. [8] [9]

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="cz.netbox.mojekonto"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="7" />
    <uses-permission android:name="android.permission.INTERNET" />
    <supports-screens
        android:largeScreens="true"
        android:normalScreens="true"
        android:smallScreens="true"
        android:anyDensity="true" />
    <application android:label="NETBOX Moje Konto"
                 android:icon="@drawable/icon"
                 android:name=".MojeKonto"
                 android:theme="@android:style/Theme.NoTitleBar">
        <activity android:name=".Login" android:label="Login">
          <intent-filter>
          <action android:name="android.intent.action.MAIN"/>
          <category android:name="android.intent.category.LAUNCHER"/>
          </intent-filter>
        </activity>
        ...
    </application>
</manifest>
```

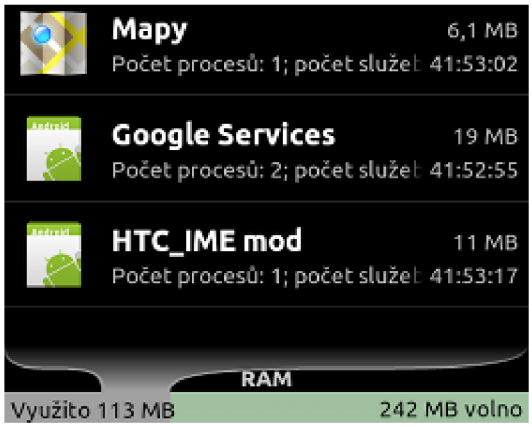
Kód 2: Příklad souboru AndroidManifest.

2.5 Chování aplikace a aktivit

Jak celá aplikace, tak i aktivity, neboli také volně přeloženo jako obrazovky aplikace, mají velmi specifické chování. Oproti ostatním mobilním operačním systémům Android již v základu podporuje multitasking[5]. S tím souvisí i vysoká režie operačního systému držet všechny aplikace spuštěné a aktivní. Android tuto situaci řeší velmi elegantně – nepoužívané běžící aplikace uspává do paměti telefonu, takže nepotřebují strojový čas a tím šetří výkon zařízení. Paměť mobilního zařízení také není nekonečná, proto operační systém v případě nedostatku paměti aplikaci ukončí. Pokud mobilní zařízení disponuje velkou operační pamětí, dokáže držet v paměti všechny aplikace od spuštění mobilního zařízení. V případě vyvolání aplikace zpět na popředí se obnoví na místě, kde uživatel skončil. [10]

Aktivity aplikace jsou obrazovky, mezi kterými uživatel přepíná. Při startu aplikace se vybere aktivita, která je vybrána v AndroidManifest a spustí se. Dále uživatel pomocí rozhraní aplikace aktivity přepíná. Při přepínání těchto aktivit se původní aktivita uloží a při návratu na tuto aktivitu se vyvolá přesně ve stavu, ve kterém jsme ji původně nechali.

Speciální kategorii tvoří tzv. services neboli služby. Stejně jak v linuxové distribuci, tak i v mobilním zařízení s Android OS běží několik služeb, které zajišťují chod určité části systému. Android Development umožňuje naimplementovat vlastní službu jako součást aplikace, která běží vždy na pozadí telefonu a stále se drží v paměti. I v případě nedostatku paměti mobilního zařízení jsou dříve ukončeny aplikace na pozadí a až poté služby (ale tato situace je velmi výjimečná). Využívá se například pro dlouho trvající operace běžící na pozadí – stahování dat z internetu, výpočet nebo jednoduchý daemon[6]. Ukázka spuštěných aplikací v mobilním zařízení na obrázku 3.



Obrázek 3: Seznam spuštěných služeb.

5 Schopnost operačního systému provádět několik procesů současně.

6 Aplikace, která je spuštěna dlouhodobě a není v přímém kontaktu s uživatelem.

3 UI design patterns

Design patterns, neboli návrhové vzory, jsou obecné best practise, které by programátor aplikace měl dodržet. Nejedná se tedy o žádně knihovny nebo předem připravené kusy kódu. Níže uvedené návrhové vzory představila společnost Google na konferenci Google I/O 2010 na aplikaci Twitter. Od této doby jsou tyto vzory brány jako nepsaný standard pro aplikace. [11]

3.1 Dashboard

Dashboard, nebo také volně přeloženo jako nástěnka, slouží jako rychlý a přehledný rozcestník pro celou aplikaci. Dashboard je zobrazen přes celou obrazovku mobilního zařízení, tvořen ikonami s popisem, které po kliknutí nabízí další funkce, jak je vidět na obrázku 4. Při natočení mobilního zařízení na bok se ikony přeskládají tak, aby uživateli nabídly přehledný rozcestník. Dashboard by se měl měnit podle chování uživatele aplikace. Nejčastěji používané akce jsou zobrazeny výše než méně používané. Toto chování nelze použít vždy, aby uživatele nemátlo náhodné měnění pozice. Tento UI vzor se rychle uchytil, lze ho spatřit v mnoha aplikacích jako je Facebook, Twitter, Evernote a další.

Obrázek 4: Dashboard v praxi.

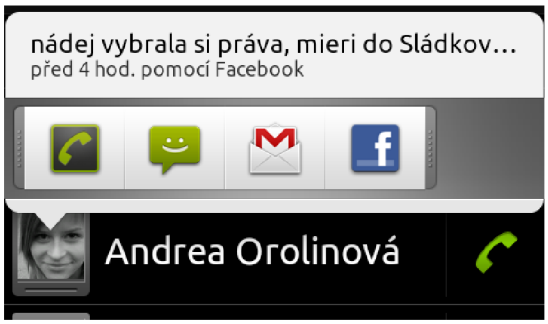
3.2 Action Bar

Action Bar je velmi oblíbený vzor, kdy se v hlavičce aktivity nachází navigační proužek aplikace obsahující nejčastěji logo a název aktuálně vybrané sekce. Na pravé straně navigačního proužku najdeme ikony pro vyhledávání, obecné informace nebo prokliky na další aktivity, jak je vidět na obrázku 5. Tento vzor lze kombinovat s vyhledáváním (Search Bar).

Obrázek 5: Ukázka použití Action Bar.

3.3 Quick Actions

Quick Actions, neboli Rychlé akce, najdete nejčastěji v kombinaci se seznamem položek. Po kliknutí na položku v obecném seznamu se otevře pop-up bublina se seznamem akcí, které lze aplikovat na položku. Například v seznamu uživatelů po kliknutí nabídne Quick Action bublina akce typu přidat uživatele do seznamu kontaktů, napsat uživateli zprávu nebo zobrazit uživatelovu fotografii. Příklad použití zobrazuje obrázek 6.



Obrázek 6: Ukázka Quick Actions.

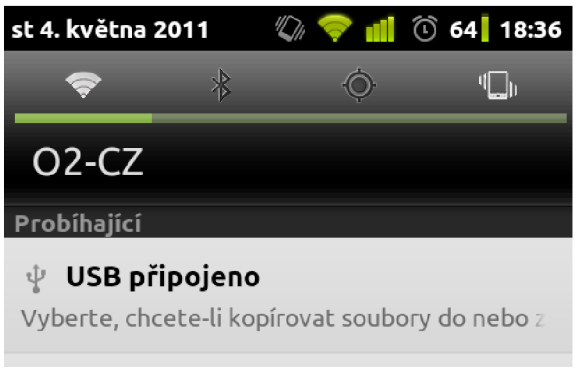
3.4 Notification

Mezi další vzory patří i vzor Notification neboli v překladu Oznámení. Smyslem tohoto vzoru je uživatele informovat, co se s aplikací děje. I sebemenší lenost aplikace uživatele otráví při jejím používání. Proto je dobré uživatele informovat o stavu. Pro tyto situace nabízí Android různé druhy Dialogu – od ProgressDialogu, přes tzv. Toasty až po Progress wheel.

Všechny typy notifikace používají hlavní UI vlákno. Abychom mohli tyto notifikace řídit a ovládat a zároveň nenechat UI zmrznout, je nutné všechny akce provádět v jiných vláknech.

3.4.1 Status Bar

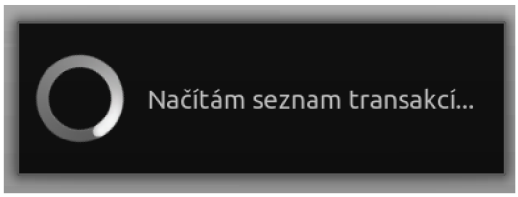
Tento typ notifikace využívá globální Status Bar, který sdílí všechny aplikace dohromady a zobrazují se vždy po celou dobu běhu mobilního zařízení. Lze použít například při oznámení nové textové zprávy. Příklad notifikace lze vidět na obrázku 7.



Obrázek 7: Status Bar.

3.4.2 ProgressDialog

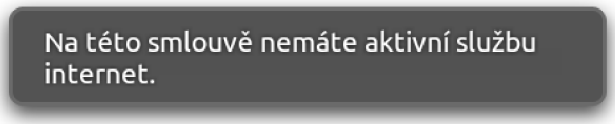
ProgressDialog je typ dialogu, kdy uživateli lze zobrazit stav akce v procentech nebo v jednotkách. Lze využít například při stahování informací z internetu, jak je vidět na obrázku 8. Nevýhodou tohoto řešení je sebrání uživateli focus z aplikace.



Obrázek 8: Příklad ProgressDialogu.

3.4.3 Toast

Toast je rychlá a nenápadná bublina, která se zobrazí uživateli na krátký moment ve spodní části displeje. Tento typ notifikace by měl zobrazovat pouze krátké a informativní sdělení uživateli – např. „Budík byl nastaven na 7:00“ nebo obrázek 9.



Obrázek 9: Ukázka Toast-u.

3.4.4 Progress wheel

Tento typ notifikace je shodný s ProgressDialogem, ale nebere uživateli focus z aplikace, takže uživatel může dále pracovat s aplikací. Příklad použití s ListView lze vidět na obrázku 10.

Obrázek 10: Použití Progress wheel v kombinaci s ListView.

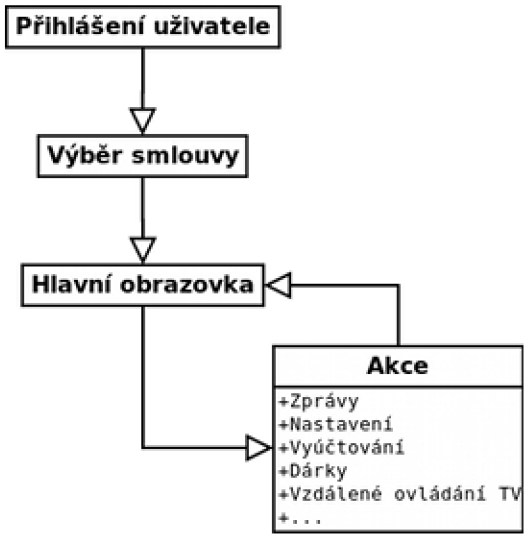
4 Návrh a analýza

Aplikace bude napsána v programovacím jazyce Java pod operačním systémem Linux. Jako vývojové prostředí bude sloužit aplikace NetBeans 6.8. Aplikaci lze také psát i v jiných jazycích (PHP, Python, C#, …), ale ty slouží pouze jako wrapper a ve výsledku jsou stejně přeloženy do jazyku Java a následně zkompilovány.

Aplikace bude napsána čistě objektově, každý ucelený kus kódu bude tvořit vlastní objekt ve vlastním souboru. Bude hojně použita dědičnost, zapouzdření a další vlastnosti objektově-orientovaného programování.

4.1 Struktura aplikace

4.1.1 Koncept



Obrázek 11: Struktura aplikace.

Na přiloženém obrázku 11 je vidět stručná struktura aplikace. Při prvním spuštění aplikace si vyžádá uživatelské jméno a heslo. Aplikace si automaticky pamatují zadané uživatelské jméno a heslo. Také lze vybrat, jestli příští přihlášení má být automatické. Všechna tato nastavení je možné editovat přes hlavní obrazovku v sekci Nastavení. Po úspěšné autorizaci uživatele aplikace nabídne uživateli jeho platné a aktivní smlouvy, se kterými lze pracovat. Aplikace si i tentokrát pamatuje naposledy vybranou smlouvu a uloží si ji. V Nastavení lze zapnout automatické vybírání smlouvy. Po výběru se zobrazí hlavní obrazovka. Ta slouží jako hlavní rozcestník pro další akce a podakce.

4.1.2 Ovládání a design

Ovládání aplikace by mělo být co nejvíce intuitivní, pochopitelné a příjemné. Celá aplikace bude ovládána pouze dotykem, takže je dobré mít všechny prvky správně rozmístěné – proti nechtěnému přehmátnutí.

Design aplikace bude ve velké míře využívat návrhové vzory. Celá aplikace bude laděna do světlých barev vycházejících z webového portálu NETBOX® Moje Konto.

4.2 Zdroj dat, API

Aplikace bude tvořit mobilní verzi webového portálu NETBOX® Moje Konto, který se aktuálně nachází na internetové adrese https://konto.netbox.cz. Proto je nutné vytvořit API[7], které bude kompatibilní s oběma verzemi portálu a budou umět mezi sebou komunikovat.

4.2.1 SOAP

SOAP neboli Simple Object Access Protocol je protokol na výměnu dat založený na XML pomocí HTTP. SOAP funguje na základě výměny zpráv, kdy každá zpráva má pevně daný formát dle šablony. SOAP zpráva (Envelope) obsahuje hlavičku (Header) a tělo (Body), jak je vidět v kódu 3 a 4. V hlavičkách se definují specifické informace aplikace – například identifikace. Samotné tělo požadavku obsahuje název funkce s parametry, která se zavolá na straně serveru. Požadavek obsahuje v atributu také definici schéma – tzv. namespace[8]. Pomocí tohoto schématu se validuje odesílaný požadavek.

Výhodou SOAP je implementace a podpora v mnoha programovacích jazycích. SOAP protokol je standardizován, takže ho lze využít i pro jiné projekty. Nevýhodou je možná složitější komunikace a syntaxe protokolu.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="...">
    <SOAP-ENV:Header />
    <SOAP-ENV:Body>
        <ns1:soapGetRealName xmlns:ns1="..." />
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Kód 3: SOAP Požadavek.

7 Application Programming Interface

8 Jmenný prostor

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="..." xmlns:ns1="...">
    <SOAP-ENV:Body>
        <ns1:soapGetRealNameResponse>
            <soapGetRealNameReturn>
                Zabaci Zaba
            </soapGetRealNameReturn>
        </ns1:soapGetRealNameResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Kód 4: SOAP Odpověď.

4.2.2 XML

Další možností pro přístup k datům pro aplikaci je vytvoření vlastního XML API. Pracovalo by na stejném principu jako SOAP, ale nebylo by potřeba vysoké a možná i zbytečné režie kolem komunikace (validace pomocí XSD[9]). Webová aplikace by jednoduše nabízela validní XML, které by si aplikace načetla a zpracovala na základě požadavku. Požadavky by se tvořily jako normální HTTP dotazy, které by po zadání GET a POST dat zobrazovaly výsledky. Příklad takového přístup lze vidět v kódu 5.

```
<ContractSet>
    <Contract contractId="1" contractNum="1234" sealed="true" />
    <Contract contractId="2" contractNum="4685" sealed="false" />
    <Contract contractId="3" contractNum="3456" sealed="true" />
</ContractSet>
```

Kód 5: Ukázka XML přístupu.

4.2.3 JSON

Mezi další možnosti se řadí také poměrně mladá technologie – JSON (JavaScript Object Notation). JSON má pevně danou strukturu a lehce připomíná strukturu XML. S tímto formátem se dá pracovat stejně jako s XML. Jedná se pouze o jiný formát prezentace dat, viz kód 6. Tato možnost má výhodu, že tento formát dokáže zpracovat i JavaScript – požadavky lze volat přes AJAX[10] a dále zpracovat.

9 XML Scheme Definition

10 Asynchronous JavaScript and XML

```
{ "menu": "File",
  "commands": [
      { "title": "New" }
   ]
}
```

Kód 6: Ukázka formátu JSON.

4.3 Sezení

Sezení, nebo také session, představuje spojení mezi serverem a klientem, kdy si vyměňují informace. V protokolu HTTP se session využívá k identifikaci uživatelů. Server na základě této informace dokáže podávat správná data pro správné uživatele. Session bývá nejčastěji ve tvaru, kterému uživatel nerozumí – například náhodný MD5 hash o minimální délce 30 znaků. Důležité je také session po určité době neaktivity mazat na straně serveru z důvodu bezpečnosti – tzv. garbage collection. Jako bezpečný interval lze považovat do 30-ti minut neaktivity – defaultní hodnota pro webový server Apache s PHP je 1440 vteřin – direktiva `session.gc_maxlifetime`. Po této době server sezení ukončuje a klient je nucen provést znovu identifikaci (přihlášení jménem a heslem). Po úspěšné autorizaci uživatel dostane novou session, se kterou lze dále pracovat.

Session lze implementovat dvěma nejrozšířenějšími způsoby – **URL**, uvedením identifikace přímo v URL požadavku (viz kód 7) nebo **Cookie**, společně s URL posílat identifikaci v cookie (viz kód 8).

```
https://konto.netbox.cz/43e417369796c8556fed305577fe31f9/
```

Kód 7: Session v URL.

```
Set-Cookie: session=53b6b83b9395cea18cf4ab94583d6a7f; expires=Sun, 17-Apr-2011
15:47:48 GMT; path=/
```

Kód 8: Session v cookie odesílaná společně s HTTP požadavkem.

Vkládat session do URL má stejný bezpečnostní efekt jako cookie – pokud někdo HTTP požadavek odposlechne (packet sniffing), session bude v požadavku vidět v obou případech. Cookie má ale oproti URL výhodu, že session není přímo viditelná pro uživatele. Pokud by uživatel URL zkopíroval a poslal někomu jinému, dává druhé osobě přímý přístup do aplikace. Jestli chceme být

velmi opatrní, lze na session vázat také IP adresu klientského počítače nebo tzv. User-Agent[11] a kontrolovat tyto hodnoty při každém dalším požadavku. [12]

O typu umístění session vždy rozhoduje server – například u Apache cookie vynutíme pomocí direktivy `session.use_cookies`.

11 Soubor informací, které prohlížeč přibaluje k požadavku a odesílá jej společně. Obsahuje například infornace o webovém prohlížeči, informace o operačním systému a další.

5 Implementace

Po návrhu a analýze celého projektu následuje implementace samotné aplikace. Aplikace je naimplementována v jazyce Java s využitím OOP.

5.1 Komunikační rozhraní – API

V kapitole 4.2 jsme prodiskutovali možná propojení s webovým portálem. Z důvodu možného dalšího rozšiřování webového portálu a snahy udělat API univerzální byla implementována možnost komunikace přes SOAP. Ten nabízí pevně definovaný standard od asociace W3C[12] [13], který se aktuálně nachází ve verzi 1.2, podle kterého je projekt implementován. Pro správnou komunikaci je nutné vytvořit SOAP rozhraní, přes které budou aplikace komunikovat. Toto rozhraní se definuje v souboru WSDL (Web Service Definition Language).

5.1.1 Server

Webový portál NETBOX® Moje Konto běží na webovém serveru Apache, naprogramován v jazyce PHP. Jazyk PHP podporuje plugin SOAP, proto vytvoření serveru nebylo obtížné. Na straně serveru se nachází PHP skript, který je navázán na WSDL soubor – metody definované v WSDL (viz kód 10) souboru odpovídají PHP skriptu, jak je vidět v kódu 9.

```php
public function soapGetVideoRemoteUrl($contractId) { ... }
```

Kód 9: Volání požadavku na straně serveru.

```
  <message name="soapGetVideoRemoteUrlRequest">
    <part name="contractId" type="xsd:integer"/>
  </message>
  <message name="soapGetVideoRemoteUrlResponse">
    <part name="url" type="xsd:string"/>
  </message>
  ...
  <operation name="soapGetVideoRemoteUrl">
    <input message="typens:soapGetVideoRemoteUrlRequest"/>
    <output message="typens:soapGetVideoRemoteUrlResponse"/>
  </operation>
```

12 World Wide Web Consortium

```
<operation name="soapGetVideoRemoteUrl">
  <operation soapAction="urn:MyPluginDefaultAction"/>
  <input>
    <body namespace="urn:pluginDefault" use="literal"/>
  </input>
  <output>
    <body namespace="urn:pluginDefault" use="literal"/>
  </output>
</operation>
```

Kód 10: Definice požadavku ve WSDL souboru.

Server podává jiný WSDL, pokud uživatel je nebo není přihlášen – jde hlavně o bezpečnostní opatření, aby uživatelé bez správné autentifikace nemohli vidět seznam dostupných metod. Po přihlášení server nabídne uživateli správný WSDL soubor.

5.1.2 Klientská aplikace

Pro aplikaci bylo potřeba sehnat kvalitní, rychlou a jednoduchou externí knihovnu napsanou v jazyce Java pro práci se SOAP požadavky, která bude správně fungovat na mobilních zařízeních s Android OS. Na internetu lze najít projekt *kSOAP 2*[13], ale odradila nás stará verze a pozastavení vývoje této knihovny. Bohužel se nepodařilo najít vhodnou knihovnu, která by odpovídala požadavkům, proto byla vytvořena třída `SoapWrapper`, která poskytuje základní SOAP rozhraní. Třída obsahuje speciální metodu `doRequest`, která zpracuje předané parametry a zavolá požadavek. Jelikož SOAP vždy vrací validní XML, výsledek požadavku se uloží do DOM dokumentu a předá se dalším metodám, které požadavek zpracují a vrátí data, o které bylo žádáno, jak je vidět v kódu 11. Ve výsledku tato pomocná třída má pouze několik set řádků oproti knihovnám, které čítají několik stovek kB. O samotný přenos dat se stará třída `HTTPUrlConnection`, která je součástí Android API. Požadavek je tedy obyčejný HTTP request společně s cookie. Pokud z nějakého důvodu požadavek selže, třída vrací `SOAPException`.

```
public Integer soapGetLoyalPoints() SoapException {
    Element soapGetLoyalPoints = new Element("soapGetLoyalPoints");
    Document doc = doRequest(soapGetLoyalPoints, SCO);
    ...
```

Kód 11: Příklad volání požadavku v SoapWrapper-u.

13 http://ksoap2.sourceforce.net/

5.2 Zpracování XML

V přechozí kapitole jsme si ukázali, jak správně získat data z webového portálu. Nyní je na řadě tato data zpracovat, rozdělit do objektů a předat dále ke zpracování. Jelikož data, která přichází na vstup, jsou ve tvaru XML v objektu Document, je potřeba tento DOM dokument projít a zpracovat. XML parserů je naprostý nespočet – od jednoduchých tříd až po celé rozsáhlé externí knihovny. Také se zde naskýtá možnost si XML parser napsat vlastní. Je nutno brát v potaz, že data přicházející od API obsahují SOAP strukturu včetně namespace a ne všechny knihovny umí s namespacy pracovat. Bylo tedy potřeba vybrat vhodný nástroj, který zvládne i toto rozšíření.

Po delších úvahách padlo rozhodnutí pro knihovnu JDom. Tato knihovna poskytuje všechny možnosti a funkce, které jsou potřeba. Knihovna patří mezi knihovny s větší režií, ale vzhledem k minimální velikosti dat k parsování můžeme tento nedostatek přehlédnout. Mezi další knihovny lze zařadit například DOM4J nebo Apache AXIOM. Bohužel Android API neobsahuji žádný předpřipravený XML parser.

V aplikaci společně se `SoapWrapper`-em lze tedy najít také metody, které využívají jednoduše JDom a zpracovávají data, která jsou potřeba. Jako vstup je předán `DOMDocument`, na výstupu lze očekávat `List` objektů, které jsou předány dál – například do `ListView`, který data zobrazuje v aktivitě na obrazovku mobilního zařízení.

```
public List<ContractGift> soapGetContractGifts(Integer contractId) throws {
    ...
    Document doc = doRequest(getContractGifts, SCO);
    ...
    for (int i = 0; i < doc.getRootElement().getChildren().size(); i++) {
        Element e = (Element) doc.getRootElement().getChildren().get(i);
        ContractGift cg = new ContractGift();
            cg.giftLabel = ((Element) e.getChildren().get(0)).getText();
            cg.validFrom = ((Element) e.getChildren().get(1)).getText();
            cg.validTo = ((Element) e.getChildren().get(2)).getText();
        contractGifts.add(cg);
    }
    return contractGifts;
}
```

Kód 12: Příklad parsování SOAP požadavku pomocí JDom.

Z kódu 12 lze vidět, že data z `DOMDocument` jsou zpracována, ve smyčce procházena a postupně plní `List` objekty.

Pokud chceme provést nějakou akci, například označení zprávy jako přečtenou, je nutné dát vědět webovému portálu, že se tato akce uskutečnila. Princip je naprosto stejný jako pro zpracování příchozí zprávy, v tomto případě budeme zprávu skládat a provedeme požadavek přes naši komunikační třídu `SoapWrapper`. Pomocí knihovny JDom vytvoříme XML dokument, naplníme daty a provedeme. Je nutné čekat na odezvu serveru potvrzující správná data a provedení.

5.3 Zabezpečení

Důležitou kategorií je také bezpečnost celé aplikace, práva a SOAP dotazy, které se posílají. Existují 2 možnosti, jak autorizovat uživatele a udržovat informaci o jeho přihlášení. Jelikož webový portál používá možnost session v cookie, je nutné se tímto krokem řídit i v této aplikaci. Posílání session v URL je více nebezpečné než v cookie, proto je nutné společně s HTTP požadavkem posílat i informaci cookie se session. Pokud uživatel zašle špatnou cookie nebo cookie nezašle vůbec, systém uživateli nabídne přihlašovací WSDL, kde lze provést autorizaci. Aplikace si informaci o aktuální cookie drží v globálním objektu `MojeKonto`, který rozšiřuje objekt `Application`. Tento objekt „žije“ po celou dobu životnosti aplikace, proto není nutné si session posílat mezi aktivitami. V tomto objektu lze ukládat globální informace o aktuálně přihlášeném uživateli – login, jméno, aktuálně vybraná smlouva a další.

Po přihlášení jsou uživateli nabídnuty funkce prostřednictvím WSDL. Tyto funkce jsou buď bez parametru (a informace o uživateli se berou ze session) nebo se jako parametr posílá číslo vybrané smlouvy, zprávy nebo jiný identifikátor. Webový portál interně kontroluje konzistenci volaných parametrů – například že přihlášený uživatel vybírá smlouvu, která mu patří. V opačném případě systém vyvolává výji mku, kterou posílá přes SOAP. Veškerou konzistenci dat tedy zajišťuje webový portál sám a není potřeba se tímto problémem zabývat.

Aplikace vyžaduje práva pro komunikaci s internetem a možnost pracovat s externím úložištěm v mobilním zařízení – nejčastěji SD karta. Tato práva se definuji v AndroidManifest, jak je vidět v kódu 13.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Kód 13: Ukázka definice oprávnění v AndroidMafinestu.

5.4 Dynamické načítaní v ListView

Chceme-li zobrazovat seznam položek, není nic jednoduššího než využít `ListView`. Toto předem připravené `View` lze kombinovat s `ListActivity` – aktivita podporující `ListView` v základu. Mnohem jednodušeji se zde používají různé metody při práci s `ListView`. Pomocí metody `setAdapter` nastavíme data, která se mají zobrazovat. Můžeme předat statické pole textových řetězců – `ArrayAdapter`, kurzor do připojené databáze – `SimpleCursorAdapter` nebo si můžeme adaptér vytvořit vlastní `BaseAdapter`. V tomto adaptéru lze nadefinovat kompletně vlastní zobrazení, binding na GUI a další. Pro dynamické načítání je nutné použít tento adaptér, jak je vidět v kódu 14.

```
public class MsgListAdapter extends BaseAdapter {
    private static final String TAG = "MojeKonto|MsgListAdapter";
    private Context context;
    private List<ScoMsg> scoMsg;
    private LayoutInflater mInflater;
```

Kód 14: Definice MsgListAdapteru.

Pro správnou funkčnost dynamického načítání je nutné nastavit `OnScrollListener`, který kontroluje stav posunutí `ListView`. Pokud se tedy `ListView` posune, je zavolána metoda `OnScroll`, kde jako parametry jsou poslány pozice jednotlivých prvků. Na základě této informace můžeme donačítat další a další data. Vhodné je také na konec `ListView` vždy připojit patičku, která informuje o stavu, jestli budou ještě nějaká data načtena (Kód 15).

```
footerView = (View) getLayoutInflater().inflate(R.layout.last_item, null);
getListView().addFooterView(footerView);
...
getListView().removeFooterView(footerView);
```

Kód 15: Přidání a odstranění patičky z ListView.

Také je nezbytně nutné upravit funkci na zasílání požadavků, kde musí být určeny meze, která data se mají donačíst a připojit do ListView.

5.5 Design aplikace

Aplikace byla vytvořena, aby uživateli nabídla co nejjednodušší ovládání, které bude profesionálně vypadat. Pomocí návrhových vzorů, které jsme si představili v kapitole 3, byla implementována

většina designu aplikace. Hned po přihlášení je uživateli nabídnut ActionBar (3.2) nabízející název aktuální aktivity. Po výběru smlouvy pro práci je uživateli nabídnut Dashboard (3.1) obsahující 9 ikon. Hlavní obrazovka je rozcestník pro celou celou aplikaci. Pod ActionBar-em lze spatřit proužek s aktuálně vybranou smlouvou. Kterákoliv časově náročná část aplikace je opatřena ProgressDialogem (3.4.2), který informuje uživatele o stavu operace. Tyto operace je nutné provádět v jiném vlákně než je samotná aplikace. Pokud provádíme tyto operace ve stejném threadu, uživateli zamrzne celá aplikace a nelze cokoliv jiného ovládat. Výjimky a ostatní chybová hlášení jsou oznamovány uživateli pomocí Toastu (3.4.3).

Většina designu aplikace je tvořena styly – definice UI objektu se naimplementuje jednou a pak se pouze používá, případně se přetěžují některé vlastnosti. Tento způsob umožňuje globálně měnit nastavení na jednom místě v jednom souboru. Ukázku definice a použití lze vidět v kódu 16 a 17.

```
<style name="action_bar_separator">
    <item name="android:layout_width">@dimen/actionbar_separator_width</item>
    <item name="android:layout_height">@dimen/actionbar_height</item>
    <item name="android:layout_marginLeft">4dip</item>
    <item name="android:layout_marginRight">4dip</item>
    <item name="android:gravity">center</item>
    <item name="android:background">@drawable/actionbar_separator</item>
</style>
```

Kód 16: Definice stylu.

```
<View style="@style/action_bar_separator" />
```

Kód 17: Použití předem nadefinovaného stylu.

5.6 Významné použité třídy

Níže popsané třídy hrají v aplikaci důležitou roli a jsou nezbytné pro fungování aplikace.

5.6.1 Downloader

Speciálně upravená třída, pomocí které lze stahovat libovolné soubory z internetu. Stačí pouze zavolat statickou metodu `getContent`, kde parametry jsou `URL` a `File` – požadovaná stránka ke stáhnutí a umístění pro výsledný soubor. Tato metoda se používá například při stahování PDF s výpisem vyúčtování. Samotné stahování používá `URLConnection` s využitím `FileOutputStream` pro ukládání do souboru.

5.6.2 MojeKonto

Tato třída rozšiřuje třídu `Application`. Tento speciální objekt může v celé aplikaci existovat právě jednou. Objekt se inicializuje při spuštění celé aplikace a žije po celou dobu aplikace. Všechny aktivity tedy mají přístup k tomuto objektu a lze tuto instanci objektu brát za globální. U této aplikace zde najdeme například globální definice některých prvků GUI, uložení aktuálně přihlášeného uživatele nebo předpřipravené seznamy pro cachování některých požadavků. Také se zde inicializuje `SoapWrapper` pro komunikaci s webovým portálem.

5.6.3 Preferences

Android OS má elegantně vyřešeno nastavení aplikací. Samotný programátor se nemusí starat o ukládání údajů nebo dalších věcí. Stačí si nadefinovat novou třídu rozšiřující PreferenceActivity, nadefinovat spravný resource XML s daty a vše správně funguje (viz kód 18). Pak pouze stačí naimplementovat metody `getString` a `setString` pro ukládání nebo zobrazení dat z nastavení. V aplikaci je použito nastavení pro ukládání informací o přihlášení nebo o automatickém výběru smlouvy. V XML části lze také určit závislosti jednotlivých položek na sobě – pomocí `android:dependency` (viz kód 19).

```
public class Preferences extends PreferenceActivity {
    ...
    addPreferencesFromResource(R.xml.settings);
    ...
}
```

Kód 18: Ukázka definice PreferenceActivity.

```
<PreferenceScreen>
    <PreferenceCategory android:title="Přihlášení">
        <EditTextPreference
            android:title="Přihlašovací jméno"
            android:key="login">
        ...
```

Kód 19: Denifice XML pro PreferenceActivity.

5.6.4 R.java

Automaticky generovaná třída starající se o binding mezi kódem a samotným UI. Generuje se při každé kompilaci projektu. Ukázka vygenerovaného souboru a chytání objektu v aplikaci vidíme v kódu 20.

```
R.java
  public static final int amount=0x7f090002;
  public static final int autoLogin=0x7f090011;
  public static final int category=0x7f090008;
  public static final int contractList=0x7f090007;

Activity.java
  (ListView) findViewById(R.id.contractList);
```

Kód 20: Příklad R souboru a chytání objektu v aplikaci.

5.6.5 Storage

Jednoduchá třída využívající externí úložiště pro ukládání dat. V kombinaci s třídou `Downloader` je toto ideální nástroj pro stahování a následné ukládání souborů z internetu. Tento objekt také kontroluje přítomnost externího úložiště a jestli lze na něj zapisovat. Pro přístup k externímu zařízení je nutné definovat práva v AndroidManifest.

6 Testování

Aplikace byla testována v emulátoru Android na verzích 1.6, 2.0, 2.1 a 2.2. Na všech verzích aplikace fungovala bez problému. Také byla testována různá rozlišení displejů – QVGA (320×240), HVGA (480×320) a WVGA (800×480). Díky definici všech GUI prvků a stylů v jednotkách DIP[14] nebyl očekáván problém. Aplikace byla také testována na mobilních zařízeních s Android OS – na všech testovaných zařízeních aplikace fungovala korektně. Pouze byly upraveny malé drobnosti v UI. [14]

Problémy byly s knihovnou JDom, která nekorektně pracovala na zařízeních s Android 1.6. Tento problém byl vyřešen nasazením jiné, trochu odlišné verze, která je přímo určena pro Android zařízení a řeší tento problém. Po nasazení nové verze knihovny se problémy již nevyskytovaly.

Program byl po dobu vývoje testován pracovníky technického oddělení SMART Comp. a.s., kteří reportovali chyby. Ty byly úspěšně odladěny.

14 Device Independent Pixel

7 Závěr

Tato bakalářská práce se zabývá analýzou a implementací rozhraní pro NETBOX® Moje Konto pro mobilní zařízení s Android OS. NETBOX® Moje Konto byl doposud pouze webovým portálem. Tento projekt měl za cíl vytvořit uživatelsky profesionální, jednoduchou a rychlou nativní aplikaci pro správu vlastních služeb od firmy SMART Comp. a.s.

Práce byla již od začátku konzultována a analyzována společně se zaměstnanci technického oddělení firmy SMART Comp. a.s. Na základě těchto požadavků byla vytvořena analýza aplikace včetně schéma jednotlivých obrazovek a hlavního menu. Během implementace byla aplikace průběžně testována, doplněna o další drobné funkcionality. Také díky testování zaměstnanců bylo odladěno několik chyb způsobujících pád celé aplikace. Následovalo celkové testování aplikace, které dopadlo úspěšně.

Výsledná aplikace splňuje základní požadavky, které byly kladeny a v dalších měsících bude nasazena do ostrého provozu a nabídnuta zákazníkům. V budoucnu by mohla být aplikace rozšířena o možnost změn a zobrazení širšího spektra informací. Bohužel pro přidání těchto rozšíření je nutné upravit stávající webový portál, tudíž změny budou velmi komplikované. Také je možné doplnit aplikace o více jazykových mutací nebo zobrazení aplikace v horizontální poloze.

Mezi další vhodné rozšíření lze zařadit například cachování všech výsledků a načítání reálných dat na pozadí. S tímto je spojené také ukládání aktuálního stavu do databáze telefonu, mergování dat a garbage collecting dat stávájících.

Použitá literatura

[1] Android (operating system). 2011, [online, cit. 2011-05-03]. URL: <http://en.wikipedia.org/wiki/Android_(operating_system)>

[2] Android 4.0 se bude jmenovat "Ice Cream". 2011, [online, cit. 2011-05-03]. URL: <http://www.root.cz/zpravicky/android-4-0-se-bude-jmenovat-ice-cream/>

[3] Reasons why the code name of Android OS is a dessert. 2011, [online, cit. 2011-05-03]. URL: <http://drexterslab.blogspot.com/2011/02/reasons-why-code-name-of-android-os-is.html>

[4] Aktuální podíl různých verzí Androidu na trhu. 2011, [online, cit. 2011-05-03]. URL: <http://www.androvinky.cz/2010/06/02/aktualni-podil-ruznych-verzi-androidu-na-trhu.html>

[5] Dalvik Virtual Machine. 2011, [online, cit. 2011-05-05]. URL: <http://www.dalvikvm.com/>

[6] Accumulated number of Application in the Android Market. 2011, [online, cit. 2011-05-03]. URL: <http://www.androlib.com/appstats.aspx>

[7] Android Architecture. 2011, [online, cit. 2011-05-02]. URL: <http://developer.android.com/guide/basics/what-is-android.html>

[8] MEIER, R..: Professional Android 2 Application Development. Wiley Publishing, Inc, 2010, ISBN 978-0-470-56552-0

[9] The AndroidManifest.xml File. 2011, [online, cit. 2011-05-03]. URL: <http://developer.android.com/guide/topics/manifest/manifest-intro.html>

[10] Android Developers: Multitasking the Android Way. 2011, [online, cit. 2011-05-04]. URL: <http://android-developers.blogspot.com/2010/04/multitasking-android-way.html>

[11] Google I/O 2011, San Francisco. 2010, [online, cit. 2011-04-20]. URL: <http://www.google.com/events/io/2010/>

[12] Session. 2011, [online, cit. 2011-05-04]. URL: <http://cs.wikipedia.org/wiki/Session>

[13] Simple Object Access Protocol (SOAP) 1.1. 2011, [online, cit. 2011-05-06]. URL: <http://www.w3.org/TR/soap/>

[14] Graphic display resolutions. 2011, [online, cit. 2011-05-09]. URL: <http://en.wikipedia.org/wiki/Graphic_display_resolutions>

Seznam příloh

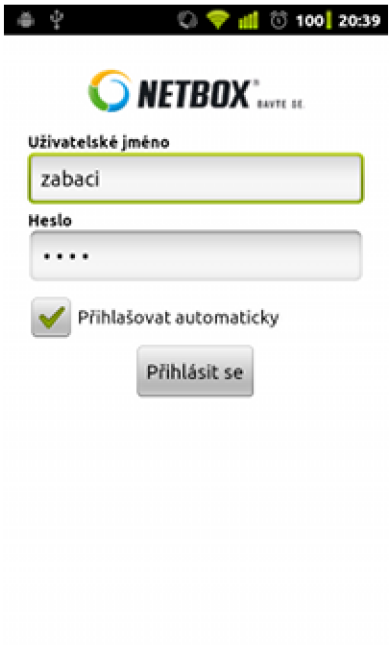
- Příloha A – CD s kompletními zdrojovými kódy aplikace.
- Příloha B – obrazovky aplikace.

Příloha A – CD se zdrojovými kódy

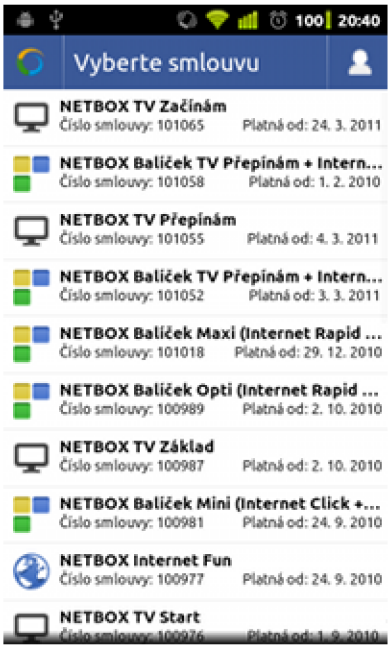
Přiložené CD obsahuje následující soubory a adresáře:

- src – adresář se zdrojovými kódy aplikace
- dist – adresář s APK verzí aplikace
- bp.pdf – PDF verze textu bakalářské práce
- README – soubor s informacemi

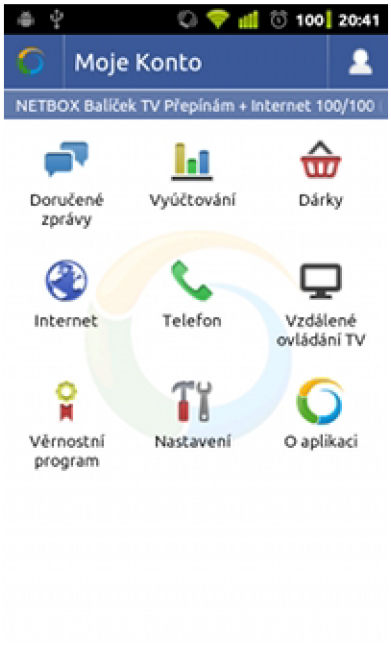
Příloha B – obrazovky aplikace



Obrázek 12: Přihlášení do aplikace.



Obrázek 13: Výběr smlouvy.



Obrázek 14: Hlavní obrazovka.

Obrázek 15: Seznam dárku u smlouvy.