VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

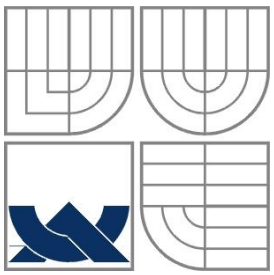
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ROZHRANÍ IS FIT PRO ANDROID

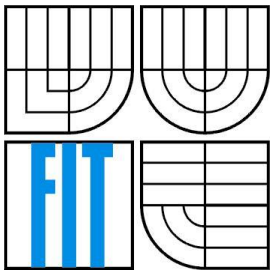
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE PETR KABÁT
AUTHOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ROZHRANÍ IS FIT PRO ANDROID

IS FIT INTERFACE FOR ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PETR KABÁT
AUTHOR

VEDOUCÍ PRÁCE — ING. JAN KOUŘIL
SUPERVISOR

BRNO 2013

Abstrakt

Tato bakalářská práce se zabývá podstatnými aspekty vývoje aplikace zpřístupňující rozhraní IS FIT na mobilní telefony. Aplikace se dělí na dvě základní části, a to serverovou část získávající data z IS FIT a část klientskou, která se stará o přehledné a komfortní ovládání vybraných funkcí informačního systému. V rámci této práce byly implementovány dvě verze klientské aplikace. Primárně nativní aplikace pro operační systém Android a také v podobě multiplatformní webové stránky.

Abstract

This bachelor thesis is dedicated to the process of development of application which provides access to the IS FIT for mobile devices. The application consists of two main parts. The server part requests data from IS FIT and the client part provides easy and intuitive way to use selected functions of information system. The development of two versions of the client application is described in this thesis. First one is the native Android software and the second one is the multi-platform web application.

Klíčová slova

IS FIT rozhraní, Android, Java, syntaktická analýza HTML, DOM, PHP, klient-server aplikace, JSON, mobilní web, jQuery Mobile

Keywords

IS FIT interface, Android, Java, HTML parsing, DOM, PHP, client-server application, JSON, mobile web, jQuery Mobile

Citace

Kabát Petr: Rozhraní IS FIT pro Android, bakalářská práce, Brno, FIT VUT v Brně, 2013

Rozhraní IS FIT pro Android

Prohlášení

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

……

Petr Kabát

15. 5. 2013

Poděkování

Chtěl bych tímto poděkovat vedoucímu mé bakalářské práce Ing. Janu Kouřilovi za připomínky a odborné rady, kterými přispěl k vypracování této bakalářské práce.

Obsah

1 Úvod

Cílem této bakalářské práce je popsat všechny důležité aspekty vývoje aplikace zpřístupňující rozhraní IS FIT na mobilní telefony, od základní analýzy problému, přes návrh řešení, přípravu vývojového prostředí, samotnou implementaci, testování, až k nasazení aplikace a zkoušení funkčnosti a použitelnosti na reálných uživatelích. V jednotlivých kapitolách jsou popsána specifika některých programovacích jazyků, formátů či zavedených principů, které byly k řešení dané problematiky využity. Kapitoly jsou koncipovány tak, aby se v nich čtenář dozvěděl vždy praktické informace o konkrétní problematice, které by potřeboval, kdyby se rozhodl projekt sám implementovat.

1.1 Analýza problému

Informační systém FIT (dále jen IS FIT) slouží studentům Fakulty informačních technologií Vysokého učení technického v Brně k přístupu k důležitým informacím týkajících se jejich studia. Naleznou zde různé školní novinky, aktuality z předmětů, přehled studia či osobní nastavení. Nejdůležitější součástí IS FIT je elektronický index, což je zjednodušeně řečeno přehled získaných bodů a známek v jednotlivých zapsaných předmětech.

V posledních několika letech se výrazně zvýšila obliba mobilních telefonů a tabletů s možností připojení na internet, což se projevilo i na poptávce po přístupu k mnoha webovým službám přes tyto zařízení. Většina webových aplikací je dostupná přes mobilní prohlížeč, avšak mnohdy to přináší nemalé ústupky a kompromisy v uživatelské přístupnosti a použitelnosti. Dále jsou tyto aplikace zpravidla méně optimalizované pro přenos dat přes mobilní síť, kde jsou datové přenosy drahé a často pomalé. Těmito a mnoha dalšími nedostatky trpí v prohlížeči mobilního telefonu i IS FIT.

První překážkou, kterou je potřeba k řešení tohoto problému překonat, je absence veřejného rozhraní pro přístup k datům z tohoto systému (například nějaké API). Tomu se práce věnuje v kapitole **Zpracování dat z IS FIT**. Dále pak je potřeba vyřešit, jak data do mobilního telefonu přenést a vhodně zobrazit tak, aby byla pohodlně použitelná i na malé dotykové obrazovce mobilního zařízení.

1.2 Navrhované řešení

Při návrhu aplikace vyšlo najevo, že nejlepší přístup k řešení problematiky zpřístupnění IS FIT na mobilní zařízení bude aplikaci rozdělit na dvě části - serverovou a klientskou.

Serverová část aplikace umožňuje komunikaci s webovým rozhraním IS FIT pomocí **HTTP protokolu**, stará se o autentizace uživatele, zadávání požadavků a syntaktickou analýzu získaných HTML dokumentů pomocí **DOM** (Document Object Model). Relevantní data poté na základě požadavků klientské části formátuje do objektového zápisu **JSON**.

Implementaci serverové části se bude věnovat kapitola **Zpracování dat z IS FIT**, zaměřená na popis architektury, způsob komunikace se serverem a problematiku syntaktické analýzy dat získaných z IS FIT.

1.3 Implementace aplikace

Cílem samotné aplikace, jejíž tvorbě jsou věnovány kapitoly **Implementace klientské aplikace** a **Multiplatformní webová aplikace**, je prezentovat data získaná ze serveru v uživatelském rozhraní, které by mělo být navrženo s důrazem na vysoký uživatelský komfort, přehlednost a jednoduché ovládání na mobilním zařízení. Klientská část byla v rámci této bakalářské práce implementována primárně jako nativní aplikace pro operační systém Android. Byla naprogramována v jazyce **Java** za pomoci **Android SDK**. Dále také v podobě multiplatformní webové stránky vyvinuté pomocí **HTML, PHP a jQuery Mobile frameworku**.

1.4 Testování aplikace

Každá aplikace by měla být před nasazením do provozu podrobena důkladnému testování. Kapitola **Testování a nasazení do provozu** probere tématiku odhalování chyb v programu a bude se věnovat i ověření použitelnosti na reálných uživatelích. Testování serverové části se zaměří na schopnost aplikace reagovat na různé varianty vstupů od IS FIT. Cílem testování klientských aplikací bude ověření korektní funkčnosti na různých mobilních telefonech s různými verzemi operačních systémů a v případě webové aplikace navíc i s různými prohlížeči.

2 Programování pro Android

Tato kapitola se věnuje teoretickým aspektům programování aplikací pro operační systém Android. Popisuje vývojové prostředí a nástroje používané při tvorbě aplikací, zabývá se základními stavebními prvky programů a jejich strukturou. Dále se věnuje některým specifikům programování aplikací v programovacím jazyce Java pro mobilní telefony, ve srovnání s vývojem pro desktopové počítače.

2.1 Vývojové prostředí

Pro programování aplikací fungujících na operačním systému Android je potřeba mít nainstalovány nástroje pro vývoj na platformě Java - **Java Development Kit (JDK)** od společnosti Oracle Corporation. Tato sada mimo jiné zahrnuje i běhové prostředí **Java Runtime Environment (JRE)**, které obsahuje virtuální stroj a sadu základních knihoven. JDK je dostupné ke stažení z webu společnosti Oracle Corporation.

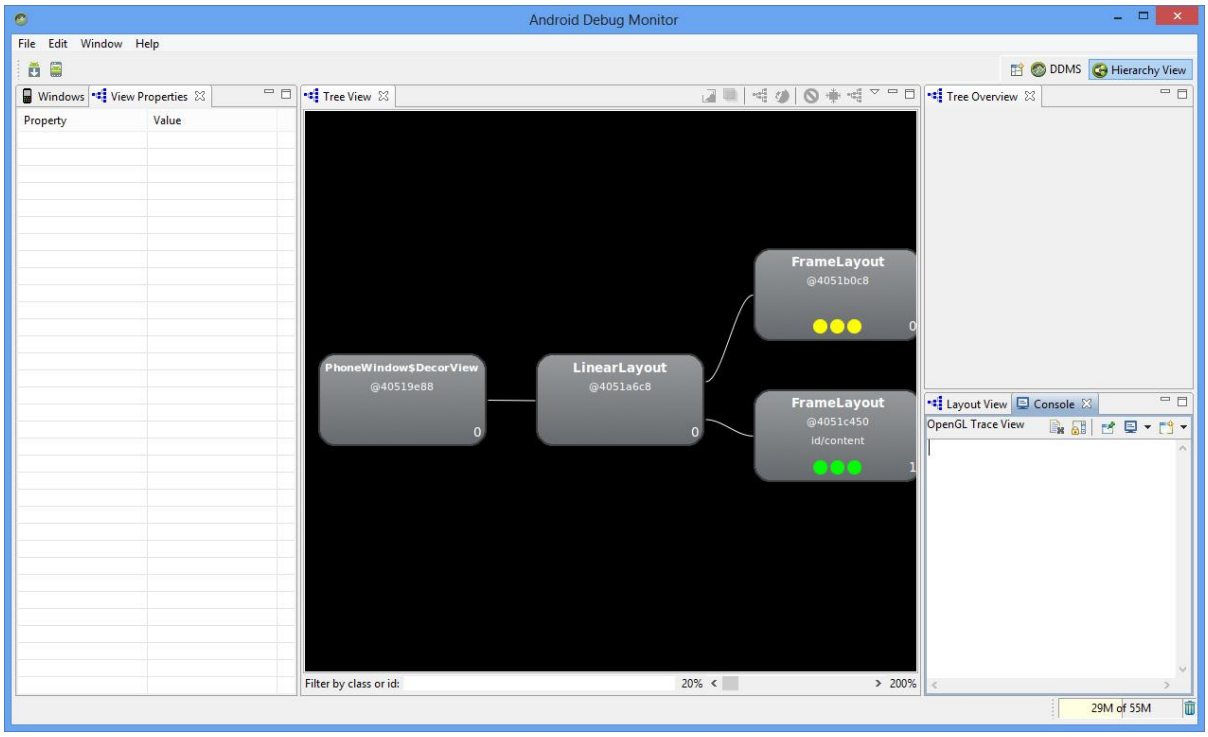
Společnost Google pro programátory připravila balíček **ADT Bundle**, jehož hlavní součástí je Android SDK, obsahující API knihovny a vývojové nástroje pro sestavování, testování a debugování vyvíjených aplikací. Pak také samotnou platformu Android, díky které je možné aplikace emulovat na počítači. Dále propracované vývojové prostředí Eclipse s pluginem Android Development Tools (ADT), který rozšíří toto IDE (Integrated Development Environment - vývojové prostředí) o všechny potřebné prvky pro vývoj aplikace pro Android, jako například o editor pro tvorbu grafického uživatelského rozhraní, apod.

2.2 Potřebné nástroje

Android SDK obsahuje celou řadu různých nástrojů a pomůcek, které usnadňují programátorům práci. Jedním z nich, se kterým se patrně setká každý vývojář aplikací pro Android, je **Android Debug Monitor** (Obrázek 2.1). Tato utilita usnadňuje práci hlavně při hledání problémů v grafickém uživatelském rozhraní. Například, pokud se některý element rozhraní zobrazuje nekorektně nebo vůbec. Tento nástroj je vhodný na hledání chyb při generování nebo editaci prvků uživatelského rozhraní přímo ze zdrojového kódu programu. Ihned je totiž vidět, jestli se element přidal do stromu obsahujícího celou hierarchii prvků uživatelského rozhraní, přesně jak bylo zamýšleno.

Android Debug Monitor umožňuje zobrazení programu přímo tak, jak je v daný okamžik vidět v mobilním telefonu. Díky tomu je možné pomocí zvětšení náhledu zjistit, jestli zamýšlené rozložení prvků sedí na pixely přesně. K dispozici je i možnost uložit aktuální snímek obrazovky pro pozdější využití.

SDK Manager použijeme pro stažení a doinstalování dalších nástrojů a jiných součástí. Jedná se například o kompletní dokumentaci SDK přístupnou bez připojení k internetu, zdrojové kódy příkladů, nebo starší verze operačního systému Android, které můžeme použít v emulátoru pro testování.



Obrázek 2.1

2.3 Aktivity a jejich životní cyklus

Programování pro Android má ve srovnání s programováním pro desktop určitá specifika. Tato kapitola se bude věnovat některým z nich.

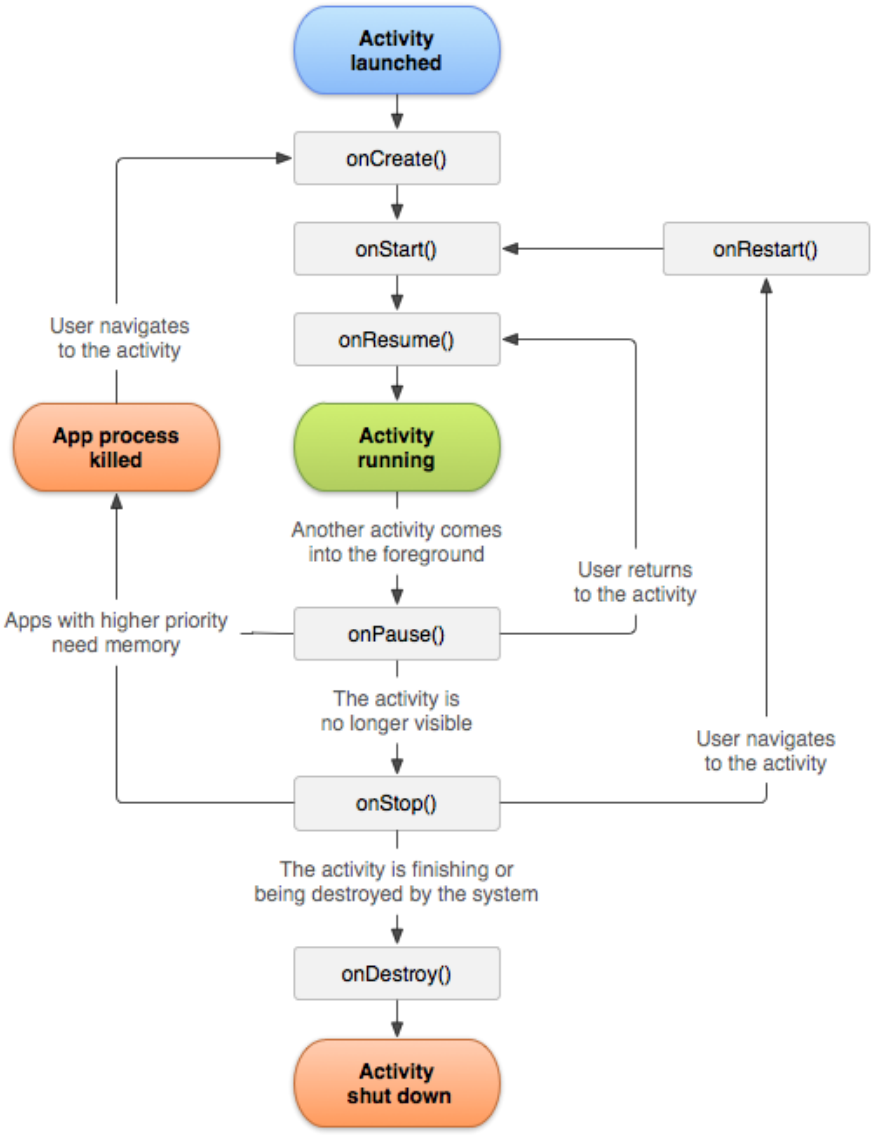
Nejdříve je potřeba uvést, z čeho se aplikace pro operační systém Android skládá. Základním stavebním prvkem při tvorbě rozhraní je u běžných desktopových aplikací okno, do kterého přidáváme další prvky, jako například tlačítka, textová pole, obrázky, apod. Android má základní stavební prvky aplikace trochu odlišné. Například výchozím prvkem pro tvorbu uživatelského rozhraní není okno, ale **Aktivita (Activity)**.

Aktivita je zjednodušeně řečeno ekvivalent controlleru či presenteru z architektury MVC/P [15]. Jejím cílem je všechna data, získaná od nižších vrstev, správně zobrazit v grafickém uživatelském rozhraní. Více k tématu na [1].

Životní cyklus aktivity (Obrázek 2.2 převzatý z [2]) se dělí na tři základní stavy. Prvním z nich je stav, kdy je aplikace na popředí a zároveň může přijímat uživatelský vstup, v druhém stavu se aplikace nachází tehdy, pokud je viditelná, ale není aktivní pro vstup. V tomto stavu, ale může být aktivita vypnuta, pokud dojde k úplnému vyčerpání operační paměti. Pokud je aktivita na pozadí, je pozastavená a kdykoliv může dojít k jejímu ukončení z důvodu uvolnění potřebné paměti. Pokud se aktivita má obnovit na popředí, je potřeba ji znovu načíst a navrátit do původního stavu.

Ve chvíli, kdy aktivita přechází mezi stavy je voláno systémové zpětné volání (callback). Callback vymezuje stavy aktivity a definuje její životní cyklus. Aktivita vzniká voláním funkce `onCreate()` a její životnost končí voláním `onDestroy()`. Aktivita je ve stavu viditelnosti mezi voláními funkcí `onStart()` a `onStop()` a uživatelský vstup má k dispozici mezi voláními funkcí `onResume()` a `onPause()`. Životní cyklus aktivity nemusí být vždy kompletní. Pokud například byla volána funkce `onPause()`, může systém při nedostatku paměti aktivitu ukončit. Ve funkci

`onPause()` je vhodné uložit aktuální stav aktivity. Jedná se o blokující funkci, takže není ideální provádět zdlouhavé operace. Spuštění nové aktivity proběhne až po dokončení této funkce. Bližší informace o životním cyklu aktivity jsou na [3].



Obrázek 2.2 (převzato z [2])

2.4 Další stavební prvky Android aplikací

Důležitým prvkem aplikací pro systém Android je **Dodavatel obsahu (Content provider)**. Content providery zpřístupňují data uložená v zařízení. Díky nim může program ukládat data do souborového systému, SQL databáze, apod.

Aktivity mají velmi krátkou životnost, a proto se nehodí pro situace, kdy je potřeba provádět nějakou činnost v pravidelných intervalech. Na rozdíl od nich jsou **Služby (Services)** navržené tak, aby mohly ve své činnosti pokračovat nezávisle na libovolné aktivitě. Služby se na rozdíl od aktivit

mohou nacházet pouze ve dvou stavech. Prvním z nich je stav, kdy je služba spuštěná (`Started`) voláním `startService()`. Služba může běžet neomezeně dlouhou dobu i v případě, že mezitím byla zrušena komponenta, která službu spustila. Libovolná komponenta se může ke službě přivázat voláním `bindService()`, tímto služba přejde do stavu `Binded`. Základní metody, které mohou být spouštěny callbacky jsou `onCreate()` (vytvoření procesu služby), `onStartCommand()` (spuštení služby metodou `startService()`), `onDestroy()` (ukončení služby) a `onBind()` (přivázání služby komponentou).

Posledním stavebním prvkem aplikací pro Android jsou **Příjemci broadcastů (Broadcast receivers)**. Ti se používají v případě, pokud má aplikace reagovat na nějakou systémovou událost, typicky například příchozí SMS, vybitá baterie, vložená paměťová karta, stažený soubor, atd.

2.5 Struktura projektu

Projekt pro operační systém Android má specifickou stromovou strukturu adresářů, ve kterých se nachází soubory potřebné pro úspěšné sestavení programu. Až na určitá specifika, vychází ze struktury jiných Java projektů.

Složka, ve které jsou uložené zdrojové soubory projektu, se jmenuje `src`. Když programátor například vytváří novou aktivitu, automaticky se tam vytvoří příslušný soubor `NazevActivity.java`.

Adresář `gen` obsahuje generované `java` soubory s nastavením potřebným pro úspěšné spuštění aplikace. V soubor `BuildConfig.java` je konfigurace potřebná pro sestavení projektu a soubor `R.java` nese informace o identifikátorech prvků uživatelského rozhraní.

Knihovny v podobě Java archivů se nachází v adresáři `libs` a zkompilované soubory nalezneme ve složce `bin`.

Dalším velice důležitým adresářem je `res`. Tento adresář obsahuje všechny důležité prostředky (resources) aplikace. V následující tabulce (Tabulka 2.1) je uvedeno, k jakému účelu slouží jednotlivé podsložky.

`res/drawable/`	Ikony, obrázky a další grafické soubory.
`res/layout/`	Specifikace grafického uživatelského rozhraní založeného na XML.
`res/menu/`	Specifikace menu založené na XML.
`res/raw/`	Obecné soubory.
`res/values/`	Pro textové řetězce, barvy, rozměry, apod.
`res/xml/`	Další přibalené XML soubory.

Tabulka 2.1

Velice důležitý je podadresář `values`, ve kterém se nachází soubory ve formátu XML, obsahující různé typy dat. Například v souboru `strings.xml` jsou uloženy textové řetězce použité v aplikaci. Díky tomu je možné například při změně jazyka aplikace, všechny texty změnit na jednom místě. Stejně tak je možné uložit i styly do souboru `styles.xml` nebo rozměry do souboru `dimens.xml`.

2.6 Modifikátory a hustota obrazových bodů

Pokud je potřeba použít odlišné sady prostředků pro různé situace, je vhodné toho docílit pomocí velmi užitečných modifikátorů (`qualifiers`). Názorně je to možné vysvětlit na přikladu lokalizace aplikace do různých jazyků, kdy je jedna složka `values` s názvem `values-cs` vyhrazena pro český jazyk a druhá složku `values-en` pro překlad do angličtiny. Modifikátorů je samozřejmě více, kromě těch pro rozlišení zmíněného jazyka a regionu, například také pro přizpůsobení se aktuální orientaci zařízení, hustotě pixelů a mnohé další. Modifikátor se spolu dají různě kombinovat.

Další důležitou složkou, kde je praktické použít modifikátory, je drawable, kam se ukládají všechny obrázky, ikony, apod. Obrázky je vhodné nahrávat v různém rozlišení podle hustoty obrazových bodů na obrazovce. Složky se jmenují podle modifikátorů ldpi (low-density – průměrně 120dpi), mdpi (medium-density – průměrně 160dpi), hdpi (high-density – průměrně 240dpi), xhdpi (extra high-density – průměrně 320dpi), nodpi pro bitmapy, které není možné přizpůsobit a tvdpi pro televizní obrazovky.

Ideální poměr velikostí obrázků se udává 3:4:6:8 pro první čtyři typy hustoty. To znamená, že ikona velká 15x15 pixelů pro ldpi, by měla být pro hustotu mdpi velká 20x20 obrazových bodů.

2.7 Android Manifest

Přímo v kořenovém adresáři projektu je umístěn soubor `AndroidManifest.xml`, ve kterém se nacházejí všechny důležité informace o vyvíjené aplikaci a jejich komponentách, jako jsou například **Aktivity**, **Služby** nebo **Dodavatelé obsahu**.

Kořenovým elementem v souboru je `manifest`, jehož atributem je deklarace jmenného prostoru dokumentu. `AndroidManifest` může dále obsahovat tyto elementy (Tabulka 2.2):

`application`	Tento element popisuje komponenty (aktivity, služby, apod.) tvořící svazek aplikace.
`uses-sdk`	Atributy tohoto elementu obsahují údaje o podporovaných verzích Android SDK.
`uses-library`	Umožňuje k aplikaci připojit volitelné komponenty (například mapovací služby).
`uses-permission`	Nastavuje oprávnění aplikaci využívat zdroje operačního systému nebo jiných aplikací.
`permission`	Deklarace práv, které je možno přidělit jiným programům, za účelem přístupu ke zdrojům aplikace.
`instrumentation`	Tento element deklaruje zdrojový kód spouštěný při systémových událostech (využití například pro logování).

Tabulka 2.2

Částečně se o obsah tohoto souboru stará přímo vývojové prostředí, takže když je vytvářena aktivita, automaticky je do souboru s manifestem přidána, pokud je potřeba vytvořit nějaké oprávnění, není potřeba znát jeho přesný název, ale je možné ho vybrat ze seznamu.

2.8 Bezpečnost a oprávnění

Aplikace jsou spouštěné na operačním systému Android pod separátními uživatelskými účty (UID i GID). Bezpečnostní architektura je navržena tak, aby ve výchozím nastavení aplikace nemohly provádět žádné operace ovlivňující ostatní programy a operační systém. To zahrnuje přístup k uživatelským datům, systémovým zdrojům, práci v síti, apod.

Pokud je potřeba přidělit aplikaci nějaká standardní oprávnění, je možné použít jedno z `android.permission`, kterých je přibližně 130, například povolení přístupu k datům o poloze telefonu (`ACCESS_FINE_LOCATION`), přístup k informacím o WIFI sítích (`ACCESS_WIFI_STATE`), možnost aplikací pořizovat snímky (`CAMERA`), číst SMS zprávy (`READ_SMS`), apod. Nastavení práva pro přístup k internetu v souboru `AndroidManifest.xml` vypadá následovně:

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

2.9 Grafické uživatelské rozhraní

Pro tvorbu grafického uživatelského rozhraní aplikace pro operační systém Android se používá jazyk XML. Samozřejmostí je potom možnost vytvářet instance widgetů prostřednictvím zdrojového kódu jazyku Java. To se používá v případě, kdy je potřeba dodatečně přidávat nové prvky rozhraní nebo měnit jejich vzhled.

Sady nástrojů pro tvorbu grafického uživatelského rozhraní (GUI) vždy nabízí nějaké základní prvky, jako jsou například textová pole, popisky, tlačítka atd. Těmto prvkům se říká widgety a v sadě nástrojů pro tvorbu uživatelského rozhraní pro Android jsou tyto widgety odvozeny od třídy `View`. Více informací o prvcích uživatelského rozhraní nalezneme v [4].

Návrhy založené na XML jsou uloženy v projektu ve složce `/res/layout` a je možné je editovat ručně, nebo pomocí **GUI Builderu** prostředí Eclipse. Dále existují některé specializované nástroje pro tvorbu a úpravu návrhů, jedním z nich je například webový nástroj **DroidDraw** (Obrázek 2.3).

Obrázek 2.3

3 Návrh aplikace

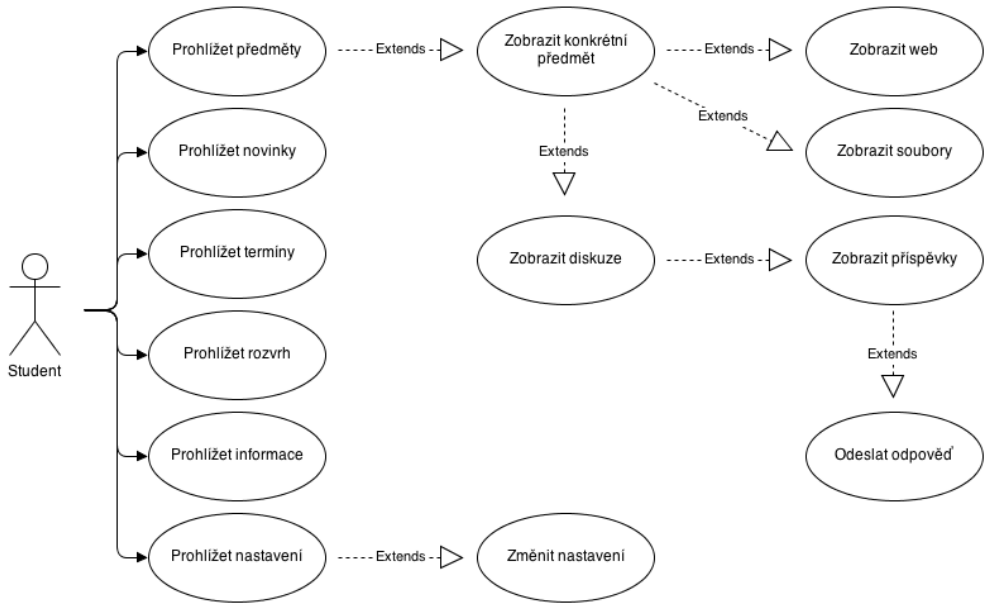
Při návrhu aplikace je nutné vzít v úvahu veškerá specifika vývoje software pro mobilní platformu. Dále je potřeba pečlivě zvážit výběr funkcí IS FIT, které budou do implementovaného rozhraní zahrnuty. Na základě těchto informací potom vybrat správné programovací jazyky a použité standardy a protokoly.

3.1 Požadavky na systém

Prvním krokem při návrhu systému bylo určit, které funkce systému IS FIT budou dostupné. Pochopitelně by to měly být ty nejčastěji používané. Po provedení jednoduchého průzkumu mezi studenty bylo zjištěno následující pořadí četnosti používání jednotlivých funkcí:

1. **Zapsané předměty v aktuálním roce**
2. **Detail předmětu**
3. **Rozvrh na semestr**
4. **Aktuální termíny**
5. **Soubory k předmětu**
6. **Novinky (nástěnky)**
7. **Diskuzní fóra k předmětu**

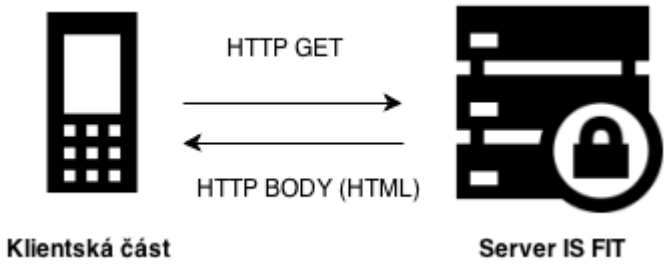
Z tohoto pořadí budeme při návrhu vycházet. Pro lepší představu je vhodné systém nějak vizualizovat. Pro zachycení požadavků na systém se používá součást jazyka UML: **diagramy případů užití (use cases)**. Diagramy případů užití (Obrázek 3.1) znázorňují akce, které je možné v systému vykonávat. Více o UML a use case diagramech nalezneme na [5].



Obrázek 3.1

3.2 Návrh architektury

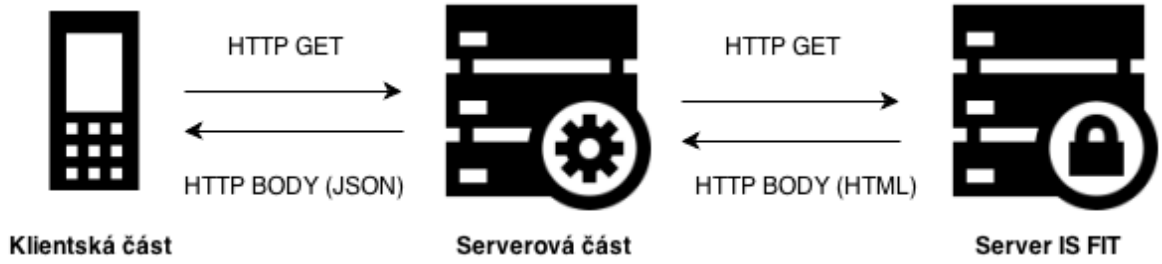
Pro implementaci mobilního rozhraní pro IS FIT, které je závislé na datech z tohoto informačního systému, je nutné zvolit vhodnou architekturu. Prakticky se nabízí dvě možné varianty. První variantou je samostatná klientská aplikace, která komunikuje přímo se systémem IS FIT (obrázek 3.2), sama ze získaných dokumentů získává syntaktickou analýzou potřebná data a přímo je zobrazuje v uživatelském rozhraní.



Obrázek 3.2

Tento přístup má své klady, ale zároveň i velice podstatné zápory. Jednou z hlavních nevýhod je nutnost zpracovávat data přímo na cílové platformě, což je nepraktické hned z několika důvodů. V první řadě **syntaktická analýza HTML** dat může být u složitějších dokumentů výpočetně i paměťově náročná operace a zatěžovat tímto mobilní zařízení s omezeným výkonem a baterií není nejlepší řešení. Dále je také přenos celých HTML dokumentů nesrovnatelně datově náročnější než přenos pouze požadovaných dat, což může nepříznivě ovlivnit rychlost reakcí aplikace. Jeden z nejzávažnějších problémů tohoto řešení však plyne ze skutečnosti, že IS FIT může změnit svoji prezentační logiku a v tom případě by pevně nastavená analýza pro získávání konkrétních dat přestala fungovat na všech klientských zařízeních do doby, než by se provedl update naší aplikace.

Proto ze srovnání vychází lépe **třívrstvá architektura** (obrázek 3.3), kdy mezi IS FIT a klientské zařízení přidáme mezičlánek v podobě serveru, který bude přijímat požadavky od klienta, předávat je IS FIT a formátovat odpověď do optimální podoby pro cílové zařízení.



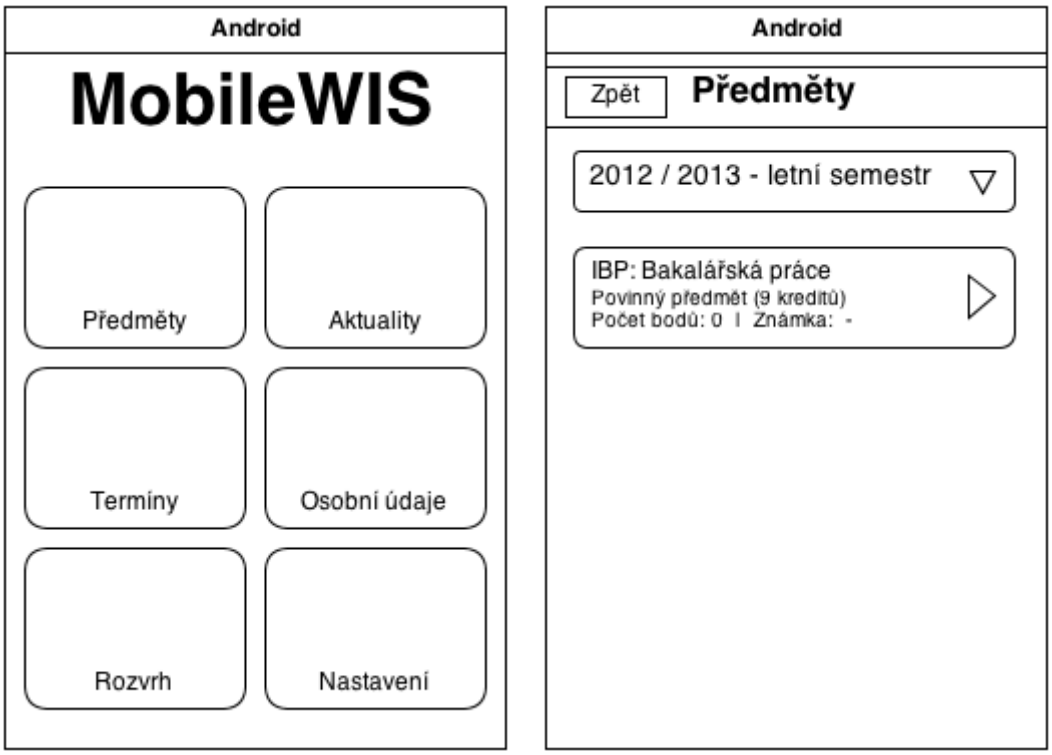
Obrázek 3.3

Tento přístup řeší hlavní nevýhody první varianty, avšak některé další problémy z něj naopak plynou. Jedním z nich je, že díky tomuto přístupu může utrpět důvěryhodnost aplikace, která musí odesílat autentizační informace na jiný server než je IS FIT. (Autentizace je proces, kterým informační systém zjišťuje, že uživatel je oprávněn pro přístup do systému. Více k tématu v [6])

Druhá nevýhoda spočívá v nutnosti udržovat v chodu server, na kterém běží serverová aplikace. Klady tohoto řešení však převažují nad zápory, proto se práce bude nadále věnovat pouze této variantě.

3.3 Návrh uživatelského rozhraní

Uživatelské rozhraní je velice důležitá součást každé aplikace a obzvláště to platí pro ty mobilní. Je nutné dbát na to, aby ovládání bylo pro uživatele na malé dotykové obrazovce komfortní. Rozhraní by nemělo být zbytečně složité, nepřehledné a kvůli tomu pro uživatele špatně ovladatelné. Podle diagramu případu užití je tedy potřeba navrhnout základní rozložení ovládacích prvků a umístění prezentovaných dat (Obrázek 3.4). Existuje velké množství programů, které umožňují vizualizovat návrh uživatelského rozhraní. Tyto programy mnohdy obsahují stejné prvky uživatelského rozhraní, jako widgety dostupné v sadě nástrojů pro tvorbu GUI pro Android.



Obrázek 3.4

3.4 Volba programovacích jazyků

V rámci návrhu aplikace je potřeba zvolit programovací jazyky, které budou pro implementaci jednotlivých částí použity. Protože má komunikace s IS FIT serverem probíhat protokolem HTTP (internetový protokol určený pro výměnu hypertextových dokumentů ve formátu HTML), je nezbytné, aby všechny jazyky měly v sobě zabudovanou dobrou podporu tohoto protokolu, kvůli bezproblémové komunikaci jednotlivých částí aplikace.

U serverové části je navíc kladen požadavek na jednoduchou instalaci na co nejširší spektrum serverů. Proto je nutné vybrat takový programovací jazyk, aby bylo možné aplikaci instalovat na libovolný webhosting a vyhnout se tak nutnosti vyhradit pro ni zvláštní server. Uvedené skutečnosti vedly k výběru programovacího jazyka PHP.

Open-source jazyk PHP byl vytvořen před více než 10 lety Rasmusem Lerdorfem. Splňuje všechny parametry úspěšného open-source projektu. Neustále se upgraduje, má robustní základnu

vývojářů a Rasmus Lerdorf se stále aktivně podílí na jeho vývoji. PHP je zdaleka nejoblíbenějším jazykem pro tvorbu webových aplikací a hlavním důvodem jeho úspěchu je jednoduchost jeho používání. Více o PHP naleznete v [7].

Hlavními důvody pro volbu PHP byla jednak výborná knihovna pro implementaci HTML parseru **PHP Simple HTML DOM Parser**, bezproblémová HTTP komunikace pomocí rozšíření **CURL** a možnost využití i pro aplikační logiku webové verze klientské části.

Programovací jazyk klientské části pro Android byl stanoven zadáním bakalářské práce, tzn. byl použit programovací jazyk Java s Android SDK (software development kit).

Pro webovou verzi klientské části bylo zvoleno výše zmíněné PHP pro aplikační logiku, dále HTML jako značkovací jazyk pro tvorbu grafického uživatelského rozhraní a JavaScriptový framework jQuery Mobile pro usnadnění práce při tvorbě rozhraní a implementaci jeho základní funkčnosti.

4 Zpracování dat z IS FIT

Tato kapitola je věnována především serverové aplikaci, která zprostředkovává komunikaci mezi klientem a serverem IS FIT, zajišťuje autentizace na serveru IS FIT, předává mu požadavky od klienta, zpracovává vrácené dokumenty a požadovaná data posílá ve formátu JSON zpět klientské části. Jednotlivé součásti jsou detailně probrané v následujících podkapitolách.

4.1 Čekání na požadavek

Jak již bylo zmíněno, programovací jazyk pro implementaci serverové části projektu bude PHP. Díky tomu můžeme využít skutečnosti, že primární použití pro PHP je dynamické generování webových stránek. Z tohoto důvodu PHP většinou funguje jako modul webového serveru. To se nám bude hodit při komunikaci mezi klientskou a serverovou částí aplikace, která bude zajištěna pomocí protokolu HTTP.

Protokol HTTP funguje na principu dotaz - odpověď. Požadavek musí být vyvolán klientem. Komunikace se serverem probíhá přes protokol TCP (server standardně naslouchá na portu 80). Požadavek musí mít specifikovánu metodu, URI (absolutní nebo relativní cesta k souboru nebo úplné URL dokumentu), verzi a hlavičky. V některých případech následuje po hlavičkách i tělo dotazu oddělené jedním prázdným řádkem. Protokol HTTP je detailně rozebrán na [8].

Vzhledem k návrhu aplikace, budou požadavky realizovány ve většině případů jako dvojice akce a parametr. Díky jednoduchosti přenášených dat může být pro zadávání požadavků serveru použito jednoduché metody GET HTTP protokolu. Spojení se serverem je tedy v případě potřeby možné provést HTTP požadavkem na adresu v podobném formátu:

```
http://server/skript.php?action=COURSE&id=1000
```

V tabulce (Tabulka 4.1) jsou podrobně rozebrány jednotlivé fragmenty URI. Reálné požadavky budou vypadat složitěji, příklad slouží pouze pro demonstraci podoby požadavku.

`http`	Schéma - určuje, o jaký druh URI se jedná a jaká syntaxe platí pro zbytek URI.
`:`	Oddělovač schématu a hierarchické části.
`//server`	Podle formátu následuje po dvou lomítkách název nebo IP adresa serveru.
`/skript.php`	V hierarchické části následuje cesta ke zdroji.
`?`	Otazník odděluje hierarchickou část a dotaz.
`action=COURSE`	Následuje posloupnost dvojic klíč = hodnota oddělená ampersandy.

Tabulka 4.1

Po přijetí HTTP požadavku webový server spustí interpret jazyka PHP a to všechny předané parametry zpřístupní v poli $_GET. Umožní tak jejich další použití.

4.2 Získání požadovaných dat

Komunikace se serverem IS FIT je realizována pomocí protokolu HTTP. Serverová část projektu se nyní nachází v roli klienta a spojení bude sama inicializovat. PHP umožňuje více možných řešení vytvoření HTTP požadavku, pro účely projektu nejlépe poslouží knihovna CURL.

PHP podporuje knihovnu libcurl vytvořenou Danielem Stenbergem, která umožňuje spojení a komunikaci s mnoha typy serverů na mnoha různých typech protokolů. Libcurl podporuje protokoly http, https, ftp, gopher, telnet, dict, file a ldap. Libcurl umožňuje použití HTTPS certifikátů, cookies, uživatelské autentizace, atd. O dalších funkcích CURL je možné se dočíst na [9].

Pro spojení se serverem IS FIT byla v projektu vytvořena zvláštní třída. Za zmínku stojí hlavně metoda, která provádí pomocí CURL požadavek na server.

V ní se provede inicializace CURL session, parametrem je URI požadavku, které se získá z třídy `types`, rozšířené o parametry metody GET z atributu metody `param`.

```
$curl = curl_init($this->types->getAddress($type).$param);
```

Pomocí `curl_setopt()` jsou nastaveny základní informace o spojení, jako například nastavení maximální délky čekání na odpověď pomocí `CURLOPT_CONNECTTIMEOUT` nebo určení jména, kterým se bude náš klient identifikovat, pomocí `CURLOPT_USERAGENT`.

Dále se volá funkce `curl_exec()` pro provedení spojení, která do proměnné `$result` vrátí výslednou odpověď. Z ní je pomocí `curl_getinfo()` získána informace o spojení, kde hodnota `http_code` rovna 200 znamená úspěšné provedení požadavku.

Takto probíhá většina požadavků na IS FIT s výjimkou, kdy je potřeba odeslat zprávu do diskuzního fóra k předmětu a server tak musí simulovat chování formuláře IS FIT, který data posílá metodou POST. V uvedeném případě jsou data k požadavku připojena pomocí `CURLOPT_POSTFIELDS`.

4.3 Zpracování získaných dat

Data, která server obdrží od IS FIT, jsou formátovaná jako HTML 4.01 dokument. Tato data je potřeba zpracovat tak, aby z nich server získal pouze relevantní informace. Pro parsování HTML dokumentu je použita třída Simple HTML DOM Parser (`simple_html_dom`), která zpracovává HTML dokument pomocí DOM.

Objektový model dokumentu (Document object model, dále pouze DOM) je API pro přístup k dokumentům XML. Umožňuje aplikaci načítat XML dokumenty bez nutnosti znalosti použité syntaxe. Konsorcium W3C vyvinulo DOM původně pro prohlížeče. Doporučení DOM podporuje jak dokumenty XML, tak HTML. Formální informace o DOM jsou blíže popsány v [10].

Základním objektem v DOM je uzel (`node`). Většina objektů v DOM stromu je odvozena z uzlů. Uzel obsahuje vlastnosti, které umožňují jednodušší průchod stromem, jako je `parentNode` (rodič), `childNode` (seznam potomků), `firstChild` (první potomek), `lastChild` (poslední potomek), `previousSibling` (předchozí uzel), `nextSibling` (následující uzel), `nodeType` (typ objektu), `attributes` (seznam atributů). Na nejvyšší úrovni stromu je `Document`, objekt odvozený z `node`, který navíc obsahuje `doctype` a `documentElement` – element nejvyšší úrovně.

Simple HTML DOM Parser funguje tak, že předá objekt `Document` a pomocí implementovaných metod (např. `find()`) je možné procházet stromem.

V ideálním případě je tak možné pomocí metody `find()` najít element `table` (tabulka) s identifikátorem například `predmety` a v něm následně hledat element `tr` (řádek tabulky) s třídou `predmet`. V tomto objektu je potom možné nalézt v jednotlivých sloupcích tabulky požadovaná data. Celý postup včetně nastavení parseru vypadá takto:

```
$document = str_get_html($inputHTML);
$table = $ document->find('table#predmety');
$trs = $tables->find('tr.predmet');
```

IS FIT v HTML kódu neoznačuje elementy identifikátory ani třídami, proto nalezení požadovaného elementu je nutné realizovat podle předpokládaného umístění objektu ve stromu, nebo podle jeho obsahu. Tímto se kód zpracovávající data velice znepřehledňuje. Toto řešení je navíc náchylné na chyby způsobené změnou umístění jednotlivých prvků v dokumentu. Z tohoto důvodu je tedy velká výhoda tento analyzátor měnit pouze na straně serveru a neprovádět aktualizace na všech klientských zařízeních.

4.4 Zaslání odpovědi

Získaná data ze zpracovaného dokumentu je nutné nějakým způsobem předat klientské aplikaci. Nejčastěji využívanými formáty pro přenos podobných dat jsou XML, CSV a JSON.

XML (extensible markup language) je velice dobře čitelný a přehledný formát pro přenos dat, navíc díky neomezené míře zanořování elementů se hodí pro dobře strukturovaná data.

Příklad formátu XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<predmet>
     <nazev>Bakalářská práce</nazev>
</predmet>
```

Nevýhodou tohoto formátu, jak je zřejmé hned na první pohled, je jeho velká datová náročnost. Pro mobilní aplikaci, která potřebuje ušetřit každý kilobyte dat přenášených po mobilní sítí, se tento zápis nehodí.

Dále lze použít formát **CSV (comma separated values)**. Co se týče datové náročnosti, tak je určitě ze zmíněné trojice nejúspornější, zaostává ovšem v možnosti data do sebe zanořovat a strukturovat.

Posledním z trojice formátů je **JSON** (JavaScript Object Notation) - odlehčený formát pro výměnu dat, který je jednoduše čitelný člověkem a lehce analyzovatelný strojově. JSON je textový formát, využívající konvence dobře známé programátorům jazyků rodiny C (C, C++, C#, Java, JavaScript, Perl, Python a dalších). JSON zápis dat vychází ve srovnání s ostatními formáty zdaleka nejlépe, navíc je dobře implementovaná podpora jak v jazyce PHP, tak v jazyce Java. Více o formátu JSON je uvedeno na [11].

Příklad formátu JSON:

```
{
"1": {"id": "409897", "short": "IAS", "name": "Asemblery"},
"2": {"id": "409903", "short": "IPR", "name": "Prvky počítačů"}
}
```

Data, která jsou od parseru uložená v datovém typu pole, PHP jednoduše převede do formátu JSON pomocí funkce `json_encode()`. Vypsáním na standardní výstup a ukončením skriptu dojde k odeslání dat odpovědí protokolu HTTP do klientské části aplikace.

5 Implementace klientské aplikace

Tato kapitola popisuje důležité aspekty implementace klientské části pro operační systém Android v jazyce Java. Zabývá se návrhem a tvorbou grafického uživatelského rozhraní, principem přechodu mezi jednotlivými aktivitami aplikace, uložením uživatelských dat, připojením aplikace k internetu, zpracováním dat od serverové části a jejich zobrazením v uživatelském rozhraní.

5.1 Tvorba uživatelského rozhraní

Jak již bylo popsáno v kapitole **Grafické uživatelské rozhraní,** na návrh těchto rozhraní se používá formát XML, díky kterému je jejich tvorba velice jednoduchá a komfortní. Do jisté míry by se dala přirovnat ke kódování webových stránek v jazyce HTML.

Pro návrh uživatelského rozhraní se formát XML používá stále častěji. Příkladem mohou být jazyky Extensible Application Markup Language (XAML) společnosti Microsoft, jazyk Flex společnosti Adobe a jazyk XML User Interface Language (XUL). Technika oddělení návrhu rozhraní pomocí XML a programové logiky do souborů se zdrojovým kódem je v mnoha ohledech velmi praktická, více informací o tomto přístupu naleznete v [4].

XML dokumenty definující podobu rozhraní obsahují stromy elementů, které určují rozložení kontejnerů a widgetů v aktivitě. Atributy elementů určují, jak má prvek vypadat a kde se má vykreslit. Takto například vypadá kód pro tlačítko:

```
<Button
    android:id="@+id/backButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/back"
    android:onClick="backButton1Clicked"
    android:layout_marginRight="10dp"

/>
```

Atribut id určuje identifikátor elementu. Je tak možné na něj odkazovat z jiných míst XML souboru, například při relativním pozicování jiných prvků v závislosti na tomto elementu. Nebo ze zdrojového kódu jazyka Java, potřebujeme-li například některé atributy změnit, a to pomocí volání `findViewById(R.id.backButton1)`, které vrátí instanci daného elementu a díky tomu potom můžeme provádět požadované změny.

Atributy `layout_width` a `layout_height` nastavené na `wrap_content` určují, že šířka a výška widgetu se přizpůsobí obsahu. Nápis na tlačítku nastavíme atributem `text` tak, že uvedeme odkaz do `resources` na datový typ `string` pojmenovaný jedinečným identifikátorem.

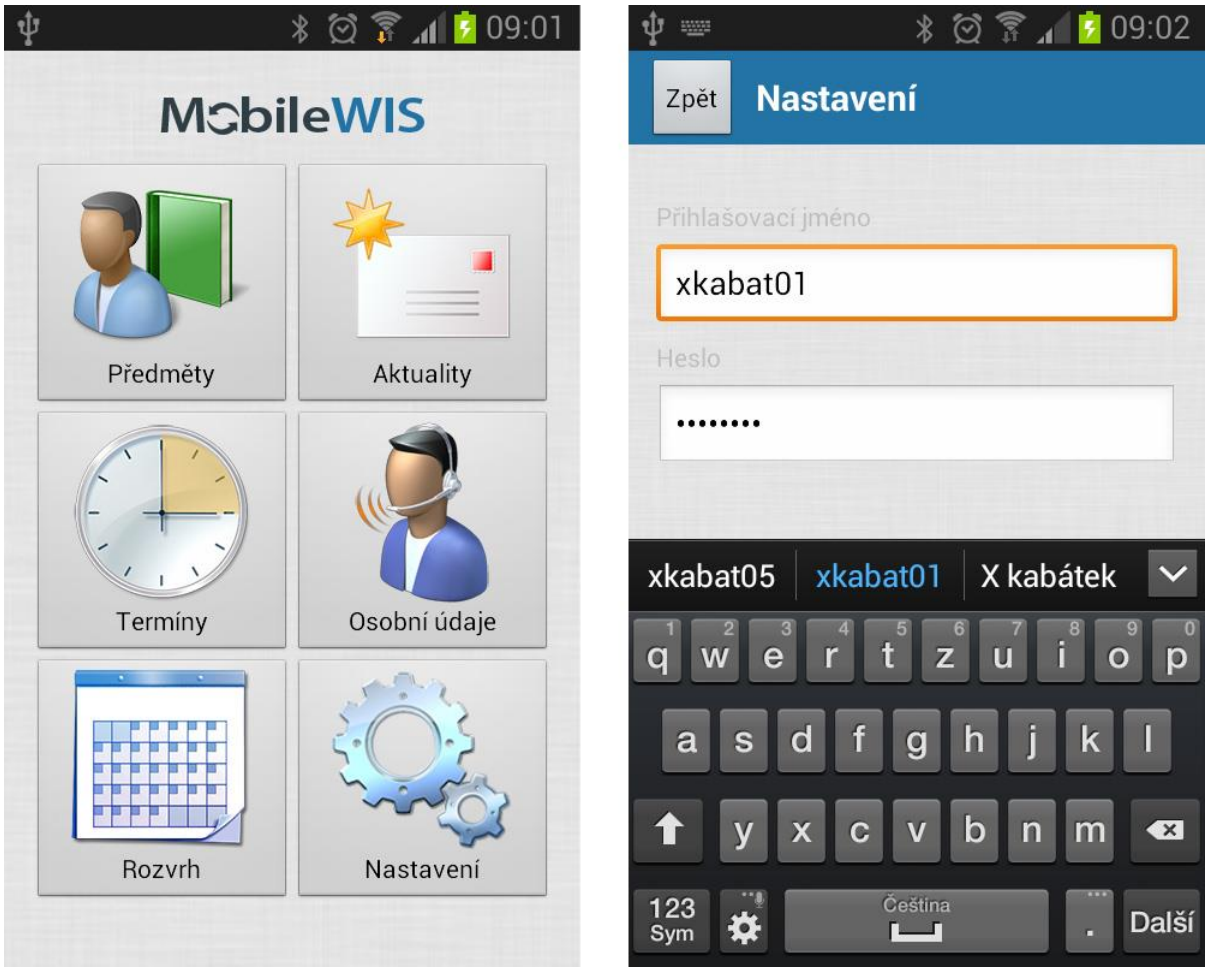
Funkci, zavolanou událostí kliknutí na tlačítko, specifikujeme atributem onClick, jako hodnota poslouží název volané funkce. Tato funkce musí mít jeden parametr, kterým je `View` (pohled), který událost vyvolal.

V případě atributu určujícího odsazení prvku zprava `layout_marginRight` musíme vzít v potaz, že mobilní zařízení mají různou velikost displeje. Velikost úhlopříčky obrazovky se udává

v palcích. Velikost obrazovky může být u některých levných telefonů i pod hranicí tří palců, zato u mnohých tabletů dosahuje až deseti palců. Každá obrazovka navíc může mít různé rozlišení. Z těchto veličin potom vychází tzv. pixel density, což se udává většinou pod zkratkou PPI či DPI. Tato jednotka určuje jemnost zobrazení displeje. Hustota 240 dpi například znamená, že na palec čtvereční fyzické velikosti displeje se vejde 240 obrazových bodů. Bližší informace je možné čerpat z [12].

To je vyřešeno jednotkou **dp (density-independent pixel)**, což je abstraktní jednotka (definována vztahem 1dp = 160px/dpi) založená na fyzické hustotě obrazovky. Lépe tak reflektuje fyzické rozměry obrazovky než velikost rozlišení.

Podoba mnou navrženého uživatelského rozhraní aplikace je vidět na následujícím obrázku (Obrázek 5.1).



Obrázek 5.1

5.2 Přechod mezi aktivitami

Naši aplikaci rozdělíme na jednotlivé aktivity. Aktivita pro úvodní stranu (`HomepageActivity`), zobrazení předmětů (`StudyActivity`), zobrazení detailu předmětu (`CourseActivity`), atd. Pro každou aktivitu navrhneme uživatelské rozhraní.

K přepnutí aktivity slouží Intenty (záměry), což jsou zprávy předávané mezi aktivitami. `Intent` je třída, obsahující popis a argumenty záměru. Jakmile máme záměr, můžeme jej předat systému a vyžádat si tak spuštění aktivity pomocí `startActivity(intent)`. Jako záměr můžeme uvést jak aktivitu nebo také například URL webu, na který chceme přejít. Takto by vypadal záměr, kdybychom se chtěli přepnout na aktivitu `CourseActivity` a předat jí jako argument nějaký identifikátor:

```
Intent intent = new Intent(this, CourseActivity.class);
intent.putExtra("ID", id);
startActivity(intent);
```

Pokud bychom chtěli v přepnuté aktivitě předaný argument přečíst, použili bychom následující kód:

```
Bundle extras = getIntent().getExtras();
if(extras != null) id = extras.getInt("ID");
```

5.3 Uložení dat

Při vývoji většiny aplikací se řeší problematika uložení různých typů dat, která potřebujeme při každém spuštění aplikace. V našem případě se jedná pouze o přihlašovací údaje do systému IS FIT, aby je při každém použití aplikace uživatel nemusel znovu zadávat.

Nabízí se nám hned tři možnosti řešení. První je uložení do souboru. To probíhá podobně jako v Java aplikaci pro desktop. Pro získání objektu typu `InputStream` nebo `OutputStream` použijeme metody pro otevření souboru `openFileInput()`, případně `openFileOutput()`. Pomocí `InputStreamReader` či `OutputStreamWriter` provedeme požadovanou I/O operaci. Nakonec použitím metody `close()` uvolníme objekty.

Druhou možností je použití některé databáze. Jedna z možností je databáze **SQLite**, což je poměrně populární zabudovaná databáze, obsahující SQL rozhraní. Je velmi rychlá a není náročná na operační paměť. Existuje mnoho komerčních i open-source produktů, které obsahují SQLite (například Adobe, Apple, Google, PHP, Python, atd.). V operačním systému Android je databáze SQLite zabudována do jeho běhového prostředí, takže ji může používat každá aplikace. Z důvodu použití klasického SQL rozhraní, je pro vývojáře, kteří mají zkušenosti s jinou databází založenou na SQL, velice snadná na ovládání. Více o SQLite se můžete dozvědět na [4].

Poslední možností jsou SharedPreferences (sdílené preference). Objekt `SharedPreferences` můžeme získat metodou `getSharedPreferences(String name, int mode)`, která díky `name` umožnuje spravovat několik různých objektů s preferencemi. `Mode` nastavíme na `MODE_PRIVATE`. K preferencím díky tomu bude moci přistupovat pouze naše aplikace. Pomocí editoru můžeme provádět v preferencích změny takto:

```
options = getSharedPreferences("options", 0);
SharedPreferences.Editor editor = editor.edit();
editor.putString("username", userName);
```

Pokud budeme chtít toto nastavení načíst, provedeme to následovně:

```
options = getSharedPreferences("options", 0);
userName = myFolder.getString("username", "");
```

`SharedPreferences` jsou pro naše účely nejvhodnějším řešením, díky jednoduchosti jejich použití. Využijeme je tedy pro uložení přihlašovacích údajů.

5.4 Připojení k internetu

V podkapitole **Bezpečnost a oprávnění** bylo vysvětleno, jak se aplikaci přidělí práva pro připojení k internetu. Nyní si ukážeme, jak takové spojení realizovat. Klientská aplikace pro Android bude se serverem komunikovat pomocí HTTP spojení. K tomu použijeme třídu `DefaultHttpClient`. Kdybychom chtěli realizovat spojení ve stejném vlákně, jako nám běží uživatelské rozhraní, při čekání na odpověď by naše rozhraní úplně přestalo reagovat. Třída `DefaultHttpClient` nám běh ve vlákně UI nedovolí a vyhozenou výjimkou nás na tuto chybu upozorní. Ještě předtím než si ukážeme, jak sestavit samotné spojení, vysvětlíme si postup pro spuštění paralelního vlákna, seznámíme se s funkcí pro posílání zpráv mezi vlákny a jejich odchyt pomocí handlerů.

Bez používání vláken a paralelních procesů by se dnes programování neobešlo. Vlákna, jak je známe z jazyka Java, jsou instancí třídy `java.lang.Thread` nebo nějakého jejího potomka. Při vytváření nového vlákna stačí pouze odvodit potomka od třídy `Thread` a překrýt klíčovou metodu `run()`, popisující, co vlákno při svém běhu provádí. Metoda `run()` neslouží k přímému spuštění vlákna, k tomuto účelu slouží volání metody `start()` z třídy `Thread`. O použití vláken v jazyce Java nalezneme více v [13].

V naší aplikaci bude z třídy `Thread` odvozena třída `Connections`, která zajišťuje všechny aspekty komunikace se serverem. Nejdříve pomocí příslušných metod nastavíme všechny parametry spojení a potom pomocí volání metody `start()` vlákno spustíme. Spojení pomocí třídy `DefaultHttpClient` provedeme následovně. Nejdříve si vytvoříme instanci třídy starající se o HTTP spojení `DefaultHttpClient`, potom použijeme HTTP metodu GET pomocí třídy `HttpGet`. Jejímu konstruktoru předáme URL adresu našeho cíle a data v podobě parametrů našeho požadavku.

Provedeme spojení pomocí metody `execute()`, jenž nám vrátí odpověď v podobě `HttpResponse` objektu. Pomocí metody `getStatusLine()` získáme status odpovědi. Pokud se kód statusu rovná hodnotě 200, což značí HTTP OK, vše proběhlo v pořádku.

Z odpovědi můžeme získat `HttpEntity` pomocí metody `getEntity()`. Z entity získáme objekt typu InputStream metodou `getContent()`. Poté můžeme pomocí `BufferedReader` přečíst odpověď.

Pokud se nám podařila získat kompletní odpověď, měli bychom o tom informovat aktivitu. Ta by si měla data zpracovat a zobrazit je v uživatelském rozhraní. Metodou třídy `Handler` `sendEmptyMessage()` odešleme prázdnou zprávu. Ta je v aktivitě zachycená vytvořenou instancí třídy `Handler` překrytou metodou `handleMessage()`. V těle metody `handleMessage()` potom provedeme volání metod, které se starají o zpracování a následné vykreslení odpovědi.

5.5 Zpracování JSON formátu

V této podkapitole si ukážeme, jak pomocí zabudovaných knihoven v jazyce Java zpracovat odpověď serveru ve formátu JSON. Podle typu odpovědi vytvoříme z textového řetězce obsahujícího JSON data buď instanci třídy `JSONObject` nebo `JSONArray`. Přímo z objektů pak můžeme získávat položky pomocí metod `getString()`, `getInt()`, `getBoolean()`, případně další JSON objekty metodou `getJSONObject()` nebo JSON pole metodou `getJSONArray()`. Objektem se dá navíc procházet iterací a získávat tak hodnoty, pokud neznáme jejich klíč.

JSON pole je možné procházet běžným cyklem. Takto získáme z odpovědi serveru přesně ty informace, které nás zajímají. V následující podkapitole si ukážeme, jak je vykreslit do uživatelského rozhraní.

5.6 Zobrazení dat

Máme připravené uživatelské rozhraní, včetně widgetů (např. `TableLayout`, `TextView`, apod.) ve kterých chceme zobrazit požadované informace. Odpověď serveru jsme již zpracovali do vhodné podoby, můžeme tedy tyto data vykreslit.

Instance widgetů, které už máme předpřipravené v souboru s šablonou ve formátu XML, získáme pomocí metody `findViewById()`, které jako parametr zadáme unikátní identifikátor hledaného widgetu. Takto si například můžeme vyhledat tabulku, do které budeme vkládat nové řádky:

```
TableLayout tableLayout =
    (TableLayout)findViewById(R.id.informationsTable);
```

Nový řádek do tabulky vložíme tak, že si vytvoříme novou instanci třídy `TableRow`, nastavíme jí parametry pro vykreslení a vložíme `View` do tabulky.

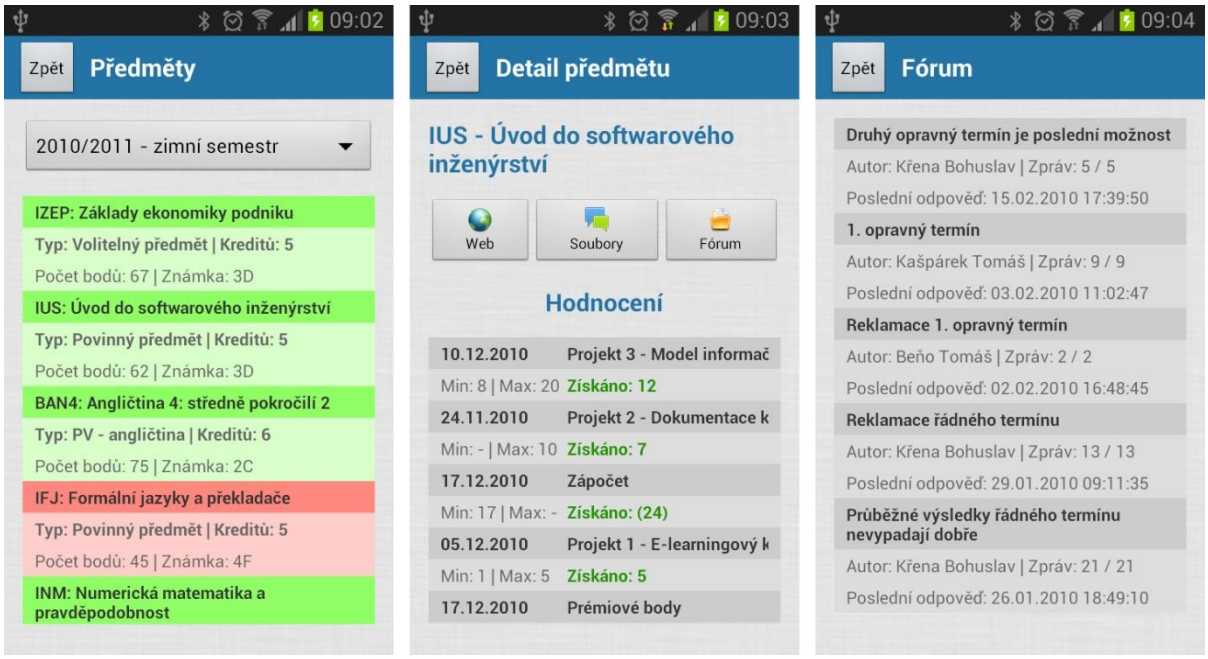
Vkládanému řádku nastavíme barvu pozadí metodou `setBackgroundColor()`, odsazení obsahu widgetu od jeho okrajů, metodou `setPadding()`, a nakonec šířku a výšku řádku, metodou `setLayoutParams()`, která očekává parametr typu `LayoutParams`.

```
TableRow tableRow = new TableRow(this);

tableRow.setBackgroundColor(0xFFCCCCCC);
tableRow.setPadding(15, 5, 15, 5);
tableRow.setLayoutParams(new LayoutParams(
        LayoutParams.MATCH_PARENT,
        LayoutParams.WRAP_CONTENT));
```

Textový popisek je do řádku možné vložit podobně. Navíc ale musíme metodou `setText()` nastavit text popisku a dále pak použít `setTextColor()` pro nastavení barvy textu. Metoda `addView()` třídy `tableRow` přidá popisek do řádku tabulky.

Výsledek můžeme zkontrolovat spuštěním aplikace. Pokud by se nějaký prvek vykresloval špatně, zkusíme hledat chybu ve dříve zmíněné utilitě **Android Debug Monitor**. Data vykreslená do uživatelského rozhraní vidíme na následujícím obrázku (Obrázek 5.2):



Obrázek 5.2

6 Multiplatformní webová aplikace

Protože Android není jediný operační systém pro mobilní zařízení a aplikace vyvinutá pro něj na iOS ani Windows Mobile fungovat nebude, rozhodl jsem se nad rámec zadání bakalářské práce vyvinout i multiplatformní webovou aplikaci, která bude obsahovat stejnou funkcionalitu jako Android verze.

Webové stránky optimalizované pro používání na mobilním zařízení lze napsat v čistém HTML nebo XHTML za použití kaskádových stylů CSS. Práci je možné si do jisté míry zjednodušit použitím nějakého frameworku. Možností je několik, například **Twitter Bootstrap**, který usnadňuje vytváření webů s responzivním designem, tedy takovým, který se podle typu zařízení přizpůsobí velikosti obrazovky. Ještě dál v pomoci při vývoji mobilní aplikace jde JavaScriptový framework založený na jQuery a jQuery UI, pojmenovaný příhodně **jQuery Mobile**. Ten navíc nabízí i spoustu užitečných vlastností, jako je například asynchronní načítání obsahu pomocí **Ajaxu**.

Ajax zprostředkovává asynchronní komunikaci mezi serverem a klientem bez nutnosti znovu načítat stránku. To umožňuje programům vytvářet další interakce, aniž by se uživatelům zpomalil běh aplikace. Zkratka AJAX znamená Asynchronní JavaScript a XML a používá se pro pojmenování technik využívaných k vytváření dynamických webových aplikací. Více o Ajaxu je možné nastudovat v [14].

6.1 Uživatelské rozhraní

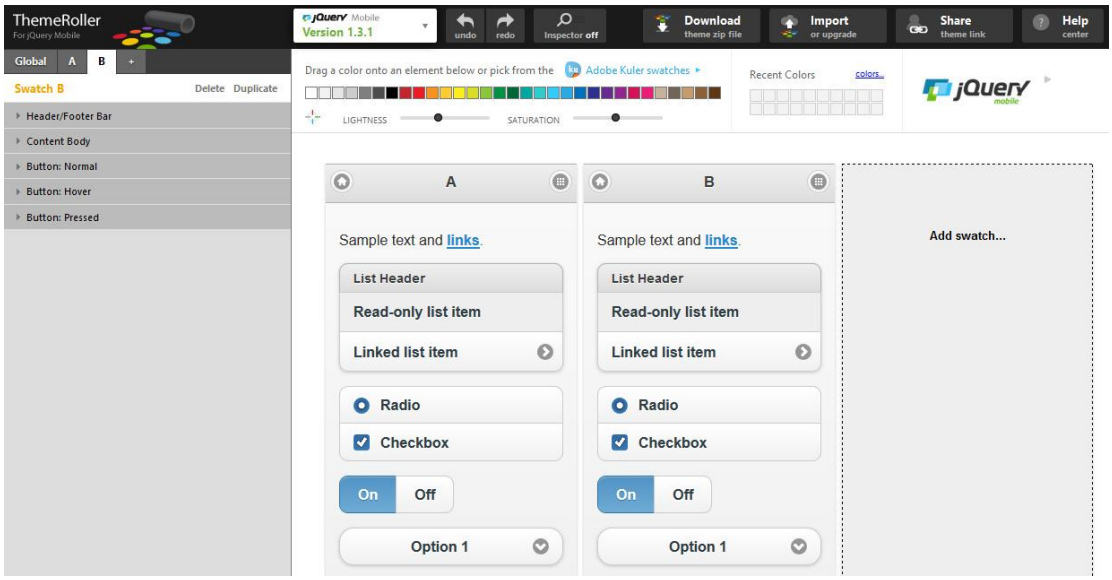
Při tvorbě uživatelského rozhraní webové verze klientské části aplikace budeme vycházet z rozložení prvků Android verze. Webová verze by měla mít určitá specifika. Jedním z nich je například nutnost přihlásit se před každým vstupem do systému uživatelským jménem a heslem. Nebude tak potřeba stránka s nastavením přihlašovacích údajů, jejíž místo v menu může nahradit tlačítko pro odhlášení. Další rozložení může zůstat stejné.

Pro základní nastavení vzhledu aplikace můžeme použít utilitu **ThemeRoller**, která nám vytvoří barevné schéma aplikace ve formátu CSS a přidá k němu soubor s použitými ikonami. Jak vypadá ThemeRoller si můžeme prohlédnout níže (Obrázek 6.1).

Nejdříve je nutné vybrat verzi jQuery Mobile, pro kterou bude šablona určena. Poté je již možné začít vytvářet novou šablonu nebo importovat stávající a tu editovat.

Každá šablona může obsahovat více barevných schémat, která můžeme při tvorbě aplikace různě kombinovat. V globálním nastavení šablony si zvolíme font písma, zaoblení hran jednotlivých prvků, barvu ikon, stínování a základní barevnost. U každého schématu pak nastavujeme barevnost záhlaví a zápatí, dále těla aplikace a tlačítek v jednotlivých stavech.

Šablonu pak můžeme exportovat tlačítkem Download, které nám nabídne ke stažení Zip archív se všemi soubory šablony, které nahrajeme k aplikaci.

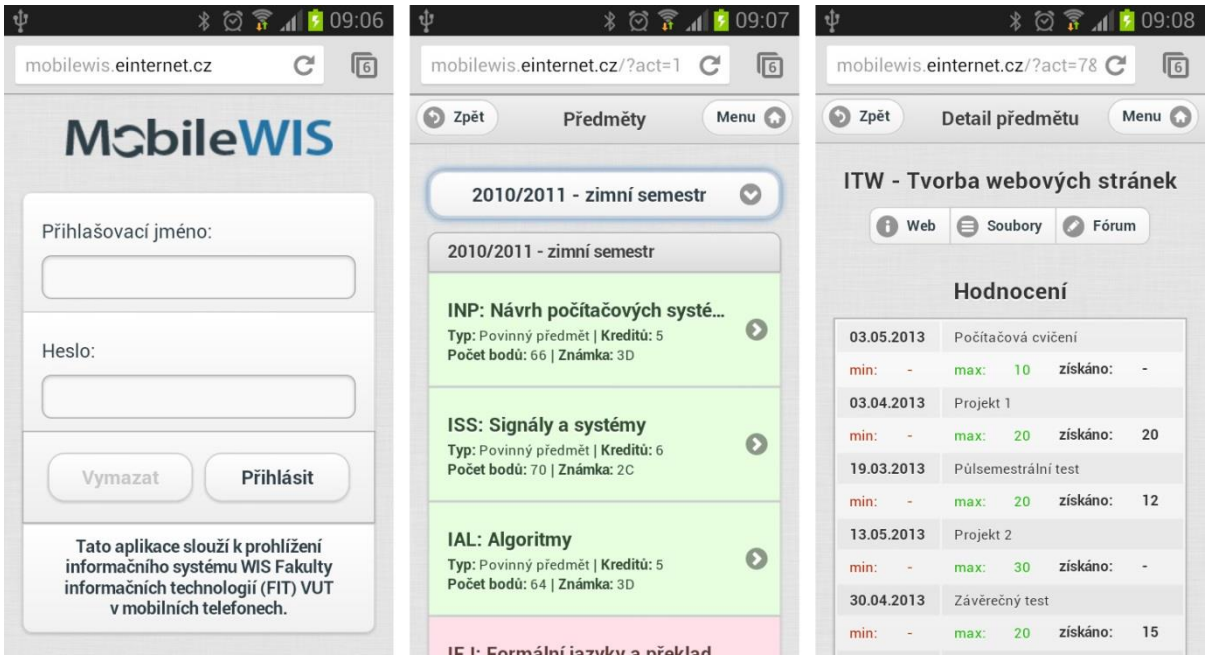


Obrázek 6.1

6.2 Získání a vykreslení dat

Vzhledem ke skutečnosti, že dynamické generování webové aplikace, stejně jako fungování serverové části, zajišťuje jazyk PHP, můžeme k API získávající data z IS FIT přistupovat napřímo, bez nutnosti komunikovat po některém síťovém protokolu. Díky tomu je výsledkem volání API rovnou datový typ pole (mnohdy vícerozměrné), obsahující všechny potřebná data.

Ta poté můžeme vykreslit přímo do uživatelského rozhraní. Výsledná aplikace vypadá takto (Obrázek 6.2):



Obrázek 6.2

7 Testování a nasazení do provozu

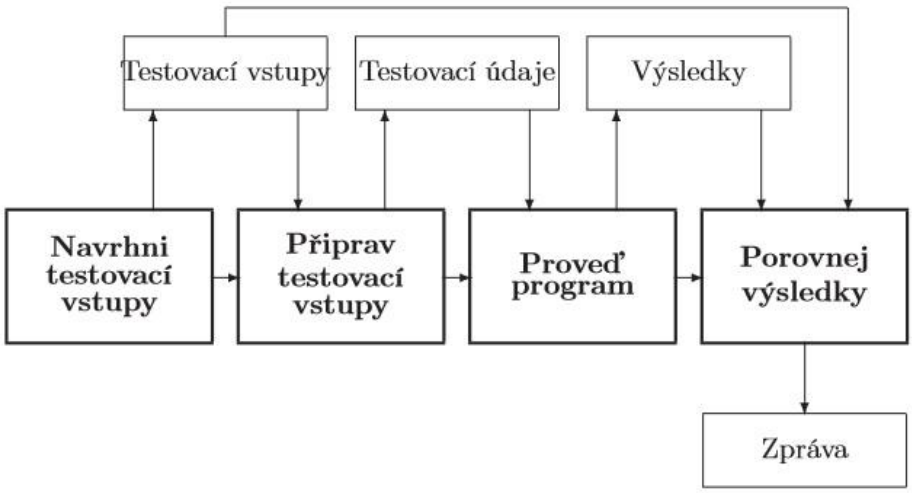
Tato kapitola bude věnována testování aplikace, prováděného za účelem ověření, že vše funguje správně, přesně tak, jak bylo zamýšleno. Testování serverové části je zaměřené na simulování různých variant vstupů od IS FIT. U klientských aplikací se testuje, jestli fungují korektně na různých mobilních telefonech s různými verzemi operačních systémů a v případě webové aplikace navíc i s různými prohlížeči. Dále bude popsáno testování na reálných uživatelích.

7.1 Testování serveru

Předmětem testování serverové aplikace by mělo být ověření, jak se server bude chovat v nestandardních situacích (výpadek internetového připojení, apod.), dále také jestli serverová aplikace správně zpracovává data získaná od IS FIT.

To je možné ověřit tak, že vytvoříme testovací vstupy (založené na reálných odpovědích IS FIT), které budeme předkládat serveru ke zpracování. Tímto se budeme snažit napodobit veškeré možné varianty odpovědí od IS FIT. Problémem při testování je výběr testovacích vstupů, jejich množina je příliš rozsáhlá, a proto není možné je otestovat všechny. Testování proto může pouze upozornit na některé chyby, ale neprokáže bezchybnost aplikace. Bližší informace k testování aplikací nalezneme v [5].

Testování probíhalo podle přiloženého schématu (Obrázek 7.1 převzatý z [5]). Díky tomuto testování se opravdu podařilo odstranit několik problémů. Tyto problémy se týkaly převážně kódování znaků na HTML entity, které by se v klientských aplikacích mohly zobrazovat nekorektně. Dále se podařilo odstranit některé nedostatky při zpracování tabulky s dosaženými výsledky jednotlivých předmětů, která se vykresluje několika možnými způsoby.



Obrázek 7.1 (převzato z [5])

7.2 Testování aplikace pro Android

Testování nativní aplikace pro Android je rozděleno na tři části. V první části jsem aplikaci předával různé testovací vstupy a sledoval jsem, jak se bude aplikace chovat. Díky tomuto testování se mi podařilo najít a opravit některé chyby v zobrazení, které mohly být způsobeny delším načteným textem v některých polích.

V druhé části jsem se rozhodl otestovat, jak bude aplikace fungovat v emulátoru, při různých konfiguracích **Android Virtual Device**. Emulátoru je možné nastavit typ zařízení a jeho rozlišení, cílový operační systém (měl jsem nainstalován Android 4.2.2 - API Level 17), typ procesoru, velikost operační paměti, velikost vnitřní paměti, atd. Postupně jsem zkoušel, jak se aplikace chová při různých konfiguracích. Tímto testováním aplikace prošla bez problému.

Nakonec jsem se rozhodl program testovat na různých fyzických telefonech a na různých verzích Androidů. Podařilo se mi aplikaci vyzkoušet na HTC HD2 (Android 2.3.5), Samsung S7562 Galaxy S Duos (Android 4.0.4), Samsung i8190 Galaxy S III mini (Android 4.1.2), Samsung Galaxy Note II N7100 (Android 4.1.1).

Více mobilních telefonů se mi pro účely ověření kompatibility nepodařilo získat, nicméně dle dotazníku z uživatelského testování, se nikdo z účastníků s žádnými problémy tohoto typu nesetkal.

7.3 Testování webové aplikace

Pro snadnější testování některých vlastností webové aplikace i bez mobilního telefonu jsem vytvořil jednoduchý „emulátor“ mobilního telefonu. Při otevření aplikace ze stolního počítače se stránka přepne do rámu s omezenou šířkou a výškou, který tak vytváří aplikaci přibližně stejný prostor pro zobrazení uživatelského rozhraní jako obrazovka mobilního telefonu.

Přesměrování do emulátoru je realizováno detekcí zařízení na úrovni PHP pomocí HTTP hlaviček, ve většině případů pomocí USER AGENT.

Testování bylo prováděno za účelem zjistit, jestli se správně zobrazují různé kombinace testovacích vstupů, které byly navrženy s cílem odhalit nestandardní chování aplikace při neočekávané odpovědi serveru.

Dále jsem se zaměřil na chování aplikace v různých prohlížečích na různých telefonech. Aplikaci jsem otestoval v nejpoužívanějších prohlížečích: Chrome, Firefox, Opera a integrovaných prohlížečích v systémech Android a iOS. Zde se ukázalo, jak dobrá volba byla použití jQuery Mobile, který byl dobře optimalizován pro všechny tyto prohlížeče.

7.4 Uživatelské testování

Poslední test jsme s vedoucím mé práce Ing. Janem Kouřilem provedli na studentech FIT. Ty jsme oslovili s prosbou o otestování aplikace a vyplnění jednoduchého dotazníku tak, abychom získali zpětnou vazbu k projektu a odhalili chyby, které i po mém důkladném testování v aplikaci mohly přetrvávat.

Pomocí aplikace Google dokumenty jsem vytvořil dotazník, jehož cílem bylo získat od respondentů hodnocení použitelnosti, funkčnosti, vzhledu a užitečnosti vyvinuté aplikace na stupnici 1 až 10 bodů. Zajímaly nás překážky, které bránily uživatelům v používání aplikace. Chtěli jsme zjistit, jaké funkce v systému studentům chybějí, jestli zaznamenali nějaké nestandardní

chování aplikace a v neposlední řadě jsme chtěli získat jejich podněty pro zlepšení aplikace. Testování se zúčastnilo 49 respondentů. Webovou verzi vyzkoušelo 39 studentů a verzi pro Android si nainstalovalo 16 studentů. Průměrné hodnocení naleznete v následující tabulce (Tabulka 7.1):

	Počet respondentů	Hodnocení použitelnosti	Hodnocení funkčnosti	Hodnocení vzhledu	Hodnocení užitečnosti
Webová verze	33	7,82	7,48	8,45	6,58
Nativní aplikace	10	8,55	7,55	7,91	8,00
Obě verze	6	6,83	6,67	7,83	6,83

Tabulka 7.1

Slovní odpovědi na položené dotazy nakonec přispěly k vyřešení některých skrytých chyb. Přinesly neocenitelnou zpětnou vazbu na projekt. A v neposlední řadě ukázaly některé směry, kterými by se další vývoj aplikace mohl ubírat.

7.5 Nasazení aplikace

Pokud bychom chtěli nabídnout aplikaci širšímu okruhu uživatelů, nejjednodušším způsobem by bylo ji vystavit v **Google Play**, což je oficiální obchod s aplikacemi pro platformu Android. Vývojáři tam mohou po zaplacení jednorázového poplatku publikovat své aplikace podepsané odpovídajícím certifikátem. V případě aplikace zpřístupňující rozhraní IS FIT, která má pouze úzký okruh potenciálních uživatelů, postačí výsledný balíček (soubor s příponou .apk) dát ke stažení na web.

Při instalaci aplikací, které nebyly staženy z Google Play, musí uživatel nastavit systému Android, aby důvěřoval aplikacím z neznámých zdrojů (Nastavení > Zabezpečení > Neznámé zdroje).

Odkazy na hotové aplikace naleznete v následující tabulce (Tabulka 7.2):

MobileWIS – verze pro Android	http://mobilewis.einternet.cz/MobileWIS.apk
MobileWIS - webová verze	http://mobilewis.einternet.cz/

Tabulka 7.2

8 Závěr

V rámci zadání praktické části bakalářské práce bylo vyvinuto funkční rozhraní pro IS FIT na mobilní zařízení, které je i na malé obrazovce mobilního telefonu dobře přehledné a použitelné. Byla implementována klientská aplikace nejen pro uživatele zařízení s operačním systémem Android, ale díky rozšíření práce i pro uživatele ostatních platforem. Studenti tak mají možnost pohodlně přistupovat k pečlivě vybraným funkcím IS FIT ze svých mobilních zařízení.

Cílem teoretické části práce bylo podrobně popsat všechny důležité aspekty vývoje, od popisu technologií, analýzy problému, návrhu řešení, přes samotnou implementaci, až k testování a uvedení do provozu. Účelem práce bylo, aby čtenář po přečtení této práce získal ucelený pohled na tvorbu podobné aplikace. V rámci udržení stanoveného rozsahu práce nebylo vždy možné popsat všechny implementační detaily, ale vždy byl kladen důraz na nastínění principu řešení tak, aby čtenář věděl jakým směrem se při vlastní tvorbě ubírat.

V případech, kdy se pro řešení dané problematiky nabízelo více možností, jsem se snažil částečně naznačit i různé alternativy tak, aby čtenář, který se při vývoji své aplikace ocitne v podobné situaci, měl možnost se rozhodnout na základě svých aktuálních potřeb.

Samotné testování potom odhalilo některé nedostatky aplikace, které se díky tomu podařilo odstranit. V případě rozeslaného dotazníku mě velice mile překvapila ochota studentů pomoci s testováním. Dále pak samozřejmě jejich velice pozitivní zpětná vazba, která nejen ukázala, že aplikace jako taková má opravdu praktické využití, ale také, že nezanedbání důsledného návrhu uživatelského rozhraní se velice pozitivně projevilo na hodnocení týkající se použitelnosti a přehlednosti aplikace.

Další vývoj projektu jsem plánoval směřovat k ostatním mobilním platformám a nabídnout tak nativní aplikaci i uživatelům jiných mobilních operačních systémů. Poznatky a návrhy z rozeslaného dotazníku mě ale donutily se zamyslet nad dalším rozvojem současné aplikace pro Android, ať už po stránce implementovaných funkcí - dotazovaní studenti nejčastěji zmiňovali nemožnost se přihlašovat na termíny cvičení, projektů a zkoušek. Dále by bylo jistě zajímavé aplikaci obohatit o uživatelské rozhraní uzpůsobené tabletům, které nabízejí větší obrazovku i rozlišení, a proto by bylo možné jednotlivé prvky rozhraní lépe uspořádat.

Věřím, že další rozvoj projektu určitě přispěje i k rozšíření uživatelské základny, a pomůže tak ještě více vylepšit aplikaci **MobileWIS**.

Literatura

[1] KONEČNÝ, Matěj. ZDROJÁK.CZ. *Vyvíjíme pro Android: První krůčky* [online]. 2012 [cit. 2013-05-09]. Dostupné z: http://www.zdrojak.cz/clanky/vyvijime-pro-android-prvni-krucky/

[2] *Android Developers: Activity* [online]. 2013 [cit. 2013-05-10]. Dostupné z: http://developer.android.com/reference/android/app/Activity.html

[3] KYPTA, Tomáš. *ABC Linuxu: Vyvíjíme pro Android – tvoříme aktivity* [online]. 2011 [cit. 2013-05-10]. Dostupné z: http://www.abclinuxu.cz/clanky/vyvijime-pro-android-tvorime-aktivity

[4] MURPHY, Mark L. *Android 2: průvodce programováním mobilních aplikací*. Vyd. 1. Brno: Computer Press, 2011, 375 s. ISBN 978-80-251-3194-7.

[5] KŘENA, Bohuslav a Radek KOČÍ. FAKULTA INFORMAČNÍCH TECHNOLOGIÍ VUT V BRNĚ. *Úvod do softwarového inženýrství*. Brno, 2010. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IUS-IT/texts/IUS_opora.pdf?cid=6837

[6] HRUŠKA, Tomáš a Michal MÁČEL. FAKULTA INFORMAČNÍCH TECHNOLOGIÍ VUT V BRNĚ. *Pokročilé informační systémy: Analýza, návrh, implementace informačního systému*. Brno, 2012. Dostupné z: https://www.fit.vutbr.cz/study/courses/WAP/private/opory/OporaPISAnalyzaNavrhImplementace.pdf

[7] LAVIN, Peter. *PHP objektově orientované: koncepty, techniky a kód*. Praha: Grada, 2009. ISBN 978-80-247-2137-8.

[8] ZAPLETAL, Lukáš. *Root.cz: Protokol HTTP 1.1 pod lupou* [online]. 2001 [cit. 2013-05-08]. Dostupné z: http://www.root.cz/clanky/protokol-http-1-1-pod-lupou/

[9] KOLEKTIV AUTORŮ. *Manuál PHP* [online]. 2005 [cit. 2013-05-08]. Dostupné z: http://jonatan.spse.pilsedu.cz/doc/php-man/index.html

[10] MARCHAL, Benoit. *XML v příkladech*. Vyd. 1. Praha: Computer Press, 2000, 447 s. ISBN 80-722-6332-3.

[11] JSON. *Úvod do JSON* [online]. 2006 [cit. 2013-05-08]. Dostupné z: http://www.json.org/json-cz.html

[12] KONEČNÝ, Matěj. ZDROJÁK.CZ. *Vyvíjíme pro Android: Suroviny, Intenty a jednotky* [online]. 2012 [cit. 2013-05-09]. Dostupné z: http://www.zdrojak.cz/clanky/vyvijime-pro-android-suroviny-intenty-a-jednotky/

[13] HEROUT, Pavel. *Učebnice jazyka Java*. 3., rozš. vyd. [i.e. 4. vyd.]. České Budějovice: Kopp, 2008, 381 s. ISBN 978-80-7232-355-5.

[14] RESIG, John. *JavaScript a Ajax: moderní programování webových aplikací*. Vyd. 1. Překlad Ondřej Baše, Ondřej Žižka. Brno: Computer Press, 2007, 360 s. ISBN 978-80-251-1824-5.

[15] DEACON, John. *Model-View-Controller (MVC) Architecture* [online]. 2009 [cit. 2013-05-15]. Dostupné z: https://techsimplified2.com/Uploads/Agendas/October28,2011.pdf

Seznam příloh

Příloha 1. CD se zdrojovými kódem