TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics and Interdisciplinary Studies

Mobilní aplikace pro práci s univerzitním elearningovým portálem na platformě Android

Mobile application for e-learning portal of university for Android devices

STUDIJNÍ PROGRAM N2612 – ELEKTROTECHNIKA A INFORMATIKA
STUDY PROGRAMME N2612 – ELECTROTECHNOLOGY AND INFORMATICS

STUDIJNÍ OBOR 1802T007 – INFORMAČNÍ TECHNOLOGIE
STUDY BRANCH 1802T007 – INFORMATION TECHNOLOGY

DIPLOMOVÁ PRÁCE | DIPLOMA THESIS

Autor práce | Author — Bc. Ondřej Vacek
Vedoucí práce | Thesis supervisor — Ing. Igor Kopetschke

LIBEREC 2013

Tento list nahraďte
originálem zadání.

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Datum:

Podpis:

Abstrakt

Hlavním cílem této práce bylo navrhnout a vytvořit mobilní aplikaci pro operační systém Android, která umožňuje studentům Technické univerzity v Liberci (TUL) snadné procházení a práci na univerzitním e-learningovém portálu a slouží tak pro zjednodušení přístupu ke studijním materiálům přes mobilní zařízení. Aplikace je naprogramována v jazyce Java za použití standardních nástrojů pro vývoj. Aplikace komunikuje prostřednictvím internetu a zabezpečených webových služeb s e-learningovým portálem, který je postaven na open-source systému Moodle. Aplikace umožňuje studentům po přihlášení procházet a stahovat studijní materiály přiřazené k jednotlivým předmětům, editovat některé osobní údaje, nahrávat soubory do systému za účelem plnění úkolů zadaných vyučujícím a prohlížet nahrané přednášky z předmětů, do nichž jsou zapsaní. Při vývoji aplikace byl kladen velký důraz na zabezpečení komunikace se serverem proti případným útočníkům a tím zamezení ztráty citlivých údajů.

Pro komunikaci se serverem e-learningového portálu je spojení zabezpečeno SSL protokolem využívajícího bezpečnostní certifikát. Protokol pro webovou službu byl zvolen SOAP, který je v systému Moodle částečně implementován a v rámci této práce byly provedeny jeho úpravy za účelem doplnění jeho možností. Samotná aplikace byla vytvořena pro Android verze 4 a vyšší z důvodu přísnějších bezpečnostních podmínek. Dále se tato práce zabývá implementací SSL vrstvy pomocí elektronického certifikátu pro připojení e-learningového portálu (jak na straně serveru, tak na straně klienta) a zpracováním komunikace přes webové služby SOAP na platformě Android. Popisuje úpravy stávající implementace webových služeb na straně e-learningového portálu a zahrnuje podrobný popis struktury a funkčnosti aplikace pro platformu Android včetně propojení se serverem pro nahrávání přednášek. Tento server využívá technologii MediaSite a jedním z cílů aplikace je snadno zpřístupnit jeho obsah studentům.

Klíčová slova: java, Android, webové služby, Moodle, e-learning TUL

Abstract

The main goal of this thesis was to design and build an Android-based mobile application, which allows students of the Technical University of Liberec (TUL) to easily navigate and work on the university e-learning portal and which provides basic access to learning resources via a mobile device. The application is programmed in Java language using standard development tools. The application communicates via the Internet and secure web services with e-learning portal that is built on open-source system Moodle. Once the student is logged in, the application allows him to browse and download course materials assigned to individual courses, edit some personal information, upload files to the system in order to complete the teacher's tasks and view the recorded lectures of subjects in which he is enrolled. The development puts a big emphasis on securing the communication with server against possible attackers and leak of confidential informations.

The communication to the server is based on encrypted SSL protocol using security certificate. The SOAP protocol was chosen for the web service because it is partly implemented in the system Moodle. The SOAP implementation was slightly alternate and improve for application needs. The application itself was created for Android operating system version 4 and higher because of more strict security conditions than lower version. Next things described in this thesis are SSL implementation including security certificate (on both server and client side) and SOAP web service implementation on Android-based devices. The thesis also describes modifications on SOAP implementation of Moodle and detailed structure of application's functions including connection to the lecture-streaming server. This server is based on MediaSite technology and one of the main goals was to access it's contents to the students.

Key words: java, Android, web services, Moodle, elearning TUL

Poděkování

Děkuji vedoucímu diplomové práce Ing. Igoru Kopetschkemu za cenné rady, připomínky a metodické vedení práce.

Děkuji své rodině, partnerce a mým přátelům za psychickou podporu, čas a prostor, který mi během vytváření této diplomové práce poskytli.

Obsah

Seznam zkratek 8

1 Úvod **9**

2 E-learningový portál TUL **12**
2.1 Zabezpečení přístupu 13
2.2 Media Site 16

3 Webové služby **17**
3.1 Protokol SOAP 18
3.2 Definice webové služby (WSDL) 19
3.3 Server 21
3.3.1 Nastavení 21
3.3.2 Modifikace 22
3.3.3 Testování 24
3.4 Klient 25
3.4.1 Autentizace uživatele 26
3.4.2 Implementace funkcí 27
3.4.3 SOAP handler 30
3.4.4 Knihovna ksoap2 32

4 Tvorba klientské aplikace **34**
4.1 Asynchronní připojení na server 35
4.2 Zápis předmětů ze STAG 37
4.3 Responzivní layout 39
4.4 Aktivity 41

5 Závěr **48**
5.1 Dosažené výsledky 49
5.2 Rozšíření 50
5.3 Alternativy 51

Seznam zkratek

TUL	Technická univerizta v Liberci
LMS	Learning Management System
CSS	Cascading Style Sheets (kaskádový styl)
WS	Web Services (webové služby)
XML	Extensible Markup Language
SOA	Service-oriented Architecture
SOAP	Simple Object Access Protocol
WSDL	Web Service Description Language
W3C	World Wide Web Consortium
PHP	PHP Hypertext Preprocesor (programovací jazyk)
JSON	JavaScript Object Notation
SDK	Software Development Kit
DPI	Dots Per Inch

1. Úvod

Mobilní zařízení zažívají v posledních letech nevídaný rozmach. S těmito zařízeními přicházejí i nové požadavky uživatelů na konzumování obsahu. Přehledné zpracování a dostupnost nepřeberného množství informací usnadňuje uživatelům nacházet nové poznatky jednoduchou formou díky zařízením, která dnes téměř každý nosí u sebe. Tzv. „chytrých telefonů" a tabletů se v loňském roce prodalo vice, než osobních počítačů a notebooků, jak uvádí časopis *BusinessIT* v [7], a to jenom dokazuje rostoucí oblibu těchto zařízení mezi všemi uživateli v celosvětovém měřítku.

Tato mobilní zařízení s sebou přináší nový problém třídění a zobrazování informací. Jejich zobrazovací jednotky (displeje) jsou mnohem menší než u klasických osobních počítačů a notebooků a navíc mají jednotlivá zařízení velmi rozdílná jak rozlišení ve zobrazovacích bodech (pixelech), tak i samotný poměr stran. Z tohoto důvodu vznikl pojem *responzivní layout*, tedy způsob zobrazení obsahu, který se mění podle velikosti zobrazovacího zařízení za účelem přehledné prezentace zobrazovaných informací. Přehlednost je obvykle dosažena přeskupením informací pod sebe, jelikož je přirozené číst informace odshora dolů (tzv. *flow*). Ovládací prvky jako odkazy a tlačítka jsou pak přizpůsobena šířce prstu a ne kurzoru myši, jak tomu bývá na běžných počítačích.

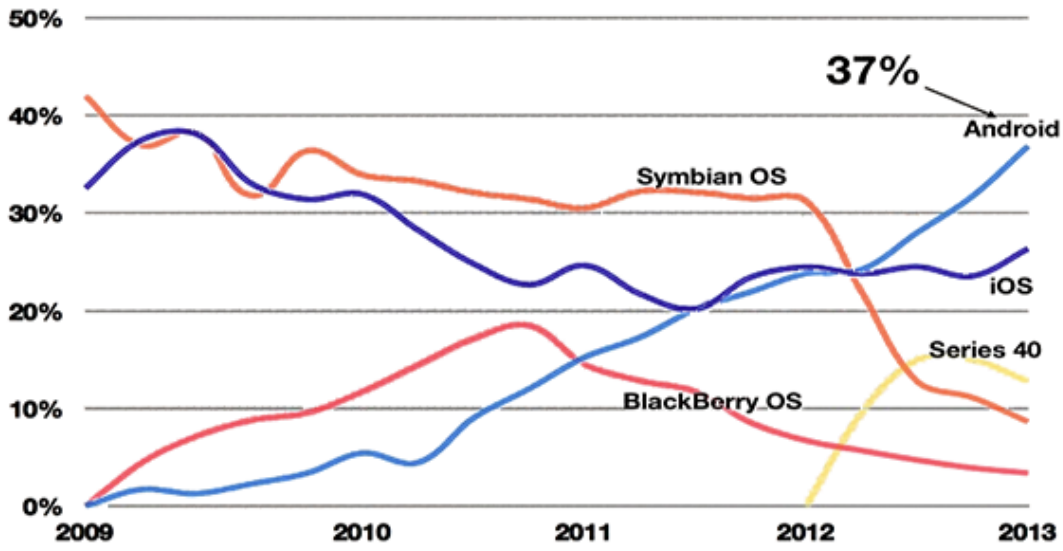
Hlavním přínosem této práce bylo přehledně zpřístupnit obsah informací z e-learningového portálu TUL jejím studentům na mobilních zařízeních. Portál je dostupný z adresy: *https://elearning.tul.cz*

Jelikož stávající elearningový portál nedisponuje *responzivním layoutem*, jeho zobrazení na mobilních zařízeních je těžko ovladatelné a nepřehledné. Studenti TUL, kteří tento portál využívají pro získání užitečných informací o svých předmětech a kteří by

rádi tyto informace získali na svém mobilním zařízení, se musí spokojit se vzhledem (*layoutem*) vytvořeným pro běžné počítače.

Pro přizpůsobení obsahu e-learningového portálu pro mobilní zařízení existují dva způsoby. První způsob spočívá ve vytvoření nového kaskádového stylu CSS (nebo editací starého stylu) webových stránek portálu. Tento způsob je jistě jednodušší a platformě nezávislý, avšak nepřináší dostatečnou flexibilitu systému a nezaručí stejné fungování portálu na všech zařízeních.

Druhým způsobem je naprogramování samostatné aplikace využívající stejný datový zdroj, jako elearningový portál. Právě tento způsob byl zvolen při řešení diplomové práce a tato zpráva popisuje její vlastnosti a funkčnost. Cílovou platformou mobilních zařízení byl zvolen operační systém *Android* od společnosti Google, jelikož je v současné době nejpoužívanějším systémem mezi svými konkurenty *iOS* od firmy Apple, *Windows Phone* od firmy Microsoft a dalšími. Vývoj zastoupení těchto systémů na trhu zachycuje graf na obrázku 1.1.



Obrázek 1.1: Zastoupení mobilní OS, zdroj: *StatCunter www.pingdom.cz*

Velký důraz byl kladen na bezpečnost aplikace, jelikož elearningový portál obsahuje nejen studijní materiály, ale i osobní údaje studentů a vyučujících. V první části této zprávy je proto popsán elearningový portál TUL a jeho nová bezpečnostní nastavení

pro minimalizaci nebezpečí úniku citlivých dat.

Dalším krokem při realizaci aplikace byl návrh a vytvoření služby zprostředkovávající přenos dat z databáze e-learningového portálu do aplikace. Za tímto účelem byl vybrán existující a v současné době nejrozšířenější protokol webových služeb (WS) s názvem SOAP. Více informací o fungování protokolu obsahuje kapitola 3.1. Stejně jako přístup na portál, i tato webová služba funguje přes šifrované spojení.

Elearningový portál TUL, který je v současné době provozován na systému Moodle, již disponuje základní implementací webové služby SOAP. Bylo zapotřebí jen drobných úprav v konfiguraci a doprogramování některých chybějících funkcí webové služby. V mobilní aplikaci bylo tuto službu nutné implementovat celou. Základní komunikační režii řeší open-source knihovna *ksoap2*. Této knihovně funkcí je v dále v textu věnována celá kapitola 3.4.4, kde je vysvětleno její použití a její funkcionalita.

Jakmile byla zvolená webová služba úspěšně implementována, jak na straně serverové (v systému Moodle), tak na straně klientské (mobilní aplikace pro pro platformu Android), a služba byla řádně otestována, přišel na řadu vývoj samotné aplikace. Při vývoji byl zohledněn především zmiňovaný *responzivní layout* proto, aby studenti TUL prohlížející data na svém mobilním zařízení dostali obsah studijních materiálů v přehledné a srozumitelné podobě.

Do aplikace se student přihlásí pomocí stejných přihlašovacích údajů, jaké používá při přístupu na elearningový portál. Při přihlášení má možnost uložit přihlašovací údaje, aby je při příštím spuštěním nemusel vyplňovat znovu. Po přihlášení vidí veškeré kurzy (nebo-li předměty), ve kterých je zapsán. Označením některého z kurzů se studentovi objeví obsah tohoto kurzu, kde má možnost prohlížet a stahovat přiložené studijní materiály, odesílat soubory pro splnění zadaných úkolů a otevírat odkazy s nahranými přednáškami a dalšími studijními materiály.

Přednášky jsou umístěny na samostatném serveru a jejich přehrávání v mobilním zařízení je možné pouze pomocí zabudovaného internetového prohlížeče v operačním systému. Server přednášek využívá aplikaci MediaSite od společnosti Sonic Foundry a je dostupný na adrese: *http://prednasky.tul.cz*

2. E-learningový portál TUL

Prvním bodem zadání této práce bylo seznámit se s e-learningovým portálem TUL. Portál běží na open-source systému Moodle, jehož charakteristické vlastnosti jsou na [2] popsány takto:

- LMS (Learning Management System) Moodle je dostupný certifikovaný Open Source výukový systém vhodný pro firmy, školy, úřady a další organizace, které chtějí využít forem e-learningu ve výuce zaměstnanců - studentů v uživatelsky přívětivém a jednoduchém prostředí.
- Celosvětový počet instalací systému již dnes přesahuje 30 000 v téměř 200 zemích. Systém byl lokalizován do více než 80 jazyků.
- Každý kurz v tomto systému je strukturovaným prostředím a sestává z jednotlivých instancí modulů, jako je fórum, studijní materiál, přednáška, test, slovník a další. Velké množství modulů základní instalace LMS spolu s nepřeberným množstvím volně dostupných modulů třetích vývojářských stran umožňují uživatelům jednoduše vytvářet, sestavovat a udržovat obsah výuky (ať již on-line kurzů, nebo i podkladů ke klasické prezenční výuce), včetně vytváření různých forem testů přímo přes jednoduchá webové rozhraní.
- Jednotlivé výukové kurzy jsou katalogizovány a tříděny do hierarchických kategorií, což případným zájemcům umožní snadnou orientaci a konzistentní přístup k nabídce kurzů, zapisování na kurzy a jejich absolvování.
- Kurzy mohou být vytvořeny přímo nástroji LMS nebo importovány jako výukové

objekty odpovídající dnes běžně dodržovaným standardům SCORM, AICC i méně oficiálním formátům.

- Systém obsahuje bohaté nástroje pro řízení, sledování a vyhodnocování aktivit vzdělávání.
- LMS Moodle lze propojit se stávajícími informačními systémy organizace a to jak formou dávkových aktualizačních skriptů, tak přímou autentizací uživatelů pomocí protokolů nad již existujícími databázemi.

Pro vývoj aplikace bylo nutné použít novou testovací instalaci systému Moodle, protože bylo nutné upravit určité zdrojové soubory systému a vývoj na aktuální verzi e-learningového portálu by mohl způsobit nestabilitu jiných částí systému. Více informací o těchto úpravách obsahuje následující kapitola 2.1. Za účelem instalace testovací verze byl použit software MAMP, který zajišťuje chod webového serveru na lokálním PC. Jeho instalace obsahuje samotný webový server Apache, překladač jazyka PHP a databázi MySQL. Všechny tyto 3 komponenty jsou nutné k instalaci a spuštění systému Moodle. Samotná instalace systému byla stažena z oficiálních internetových stránek systému:

http://download.moodle.org

Při vývoji byl použit Moodle verze 2.4.3+ (číslo sestavení: 20130405), která byla v době vývoje aktuálně nejnovější stabilní verzí.

2.1 Zabezpečení přístupu

Jak již bylo zmíněno v úvodu, velký důraz při vývoji aplikace byl kladen na bezpečnost přenosu informací. Z tohoto důvodu bylo nutné nastavit přístup na testovací portál přes protokol HTTPS. Nejprve byl vygenerován bezpečnostní certifikát. Pro testovací účely byl použit tzv. *self-signed* certifikát, který není běžně považován za důvěryhodný, ale na zabezpečení testovací verze systému postačil. Současný elearningový portál TUL

je zabezpečen certifikátem vydaným přímo pro TUL pod názvem „*maja.fm.tul.cz*“ a je podepsaný u certifikační autority „*The USERTRUST Network*“. Více o certifikátech a jejich bezpečnosti píše například autor Jiří Peterka v [6].

Bezpečnostní *self-signed* certifikát byl vygenerován pod operačním systémem MacOSX takto:

```
$openssl genrsa -des3 -out server.key 1024
```

příkaz vygeneruje RSA privátní klíč *server.key* s trojitým DES šifrováním o délce 1024 bitů, je potřeba zadat heslo, potvrdit klávesou ENTER, zopakovat heslo a znova ENTER

```
$openssl req -new -key server.key -out server.csr
```

na základě předchozího klíče vygeneruje self-signed certifikát, poté je potřeba vyplnit následující informace, obzvláště důležitá je položka *Common name*, která se musí shodovat s adresou serveru, v tomto případě *moodle*

```
$Country Name (2 letter code) [AU]:CZ
$State or Province Name (full name) [Some-State]:Czech republic
$Locality Name (eg, city) []:Liberec
$Organization Name (eg, company) [Internet Widgits Pty Ltd]:TUL
$Organizational Unit Name (eg, section) []:development
$Common Name (eg, YOUR name) []:moodle
$Email Address []:admin@moodle
$A challenge password []:
$An optional company name []:
```

poslední dva údaje jsou nepovinné, a proto zůstaly nevyplněny

```
$openssl x509 -req -days 365 -in server.csr -signkey server.key -out
server.crt
```

dále převede certifikát do formátu *crt*, podepíše jej klíčem a určí platnost na jeden rok, pro podpis je potřeba zadat heslo z prvního kroku a potvrdit klávesou ENTER

```
$cp server.key server.tmp
$openssl rsa -in server.tmp -out server.key
```

první příkaz zkopíruje klíč a druhý z něj odebere heslo pro použití na straně serveru.

Nyní se v domovské složce (nebo složce, kde se příkazy prováděly) nachází dva nové soubory. Soubor *server.crt* tvoří veřejný klíč certifikátu, *server.key* obsahuje privátní klíč potřebný k šifrování. Oba soubory je nutné nahrát do složky s nastavením Apache a editovat konfigurační soubor *ssl.conf*, který se nachází v téže složce. Na konec tohoto souboru se přidá definice *VirtualHost*, která nastaví zabezpečený přístup na testovací elearningový portál. Jeho definice vypadá takto:

```
<VirtualHost moodle:443>
        SSLEngine on
        SSLCertificateFile /cesta_k_souboru/server.crt
        SSLCertificateKeyFile /cesta_k_souboru/server.key
</VirtualHost>
```

Po editaci je nutné server Apache restartovat pro načtení změněné konfigurace. Detailnější informace o celkovém nastavení Apache jsou popsány v [4].

Nakonec je potřeba vytvořit nový záznam v lokálním DNS souboru *hosts*. Ten se nalézá v unixových systémech ve složce *„/etc/“*, v systému Windows ve složce *„%SystemRoot%\system32\drivers\etc\“*, kde *%SystemRoot%* je instalační složka systému. Do tohoto souboru je nutné přidat nový řádek `127.0.0.1 moodle`, který zajistí, že zadáním adresy *https://moodle* do webového prohlížeče se počítač odkáže na lokální server Apache.

Nyní je systém Moodle nastaven pro přístup přes zabezpečený protokol HTTPS. Pokud by bylo potřeba vynutit přístup na portál pouze přes tento protokol, je navíc nutné změnit záznam v konfiguračním souboru. Ten se nalézá v kořenovém adresáři systému pod názvem *config.php*. Zde je potřeba změnit hodnotu řádku `$CFG->wwwroot = 'http://moodle';` na hodnotu `$CFG->wwwroot = 'https://moodle';`, kde slovo *moodle* udává adresu systému Moodle.

2.2 Media Site

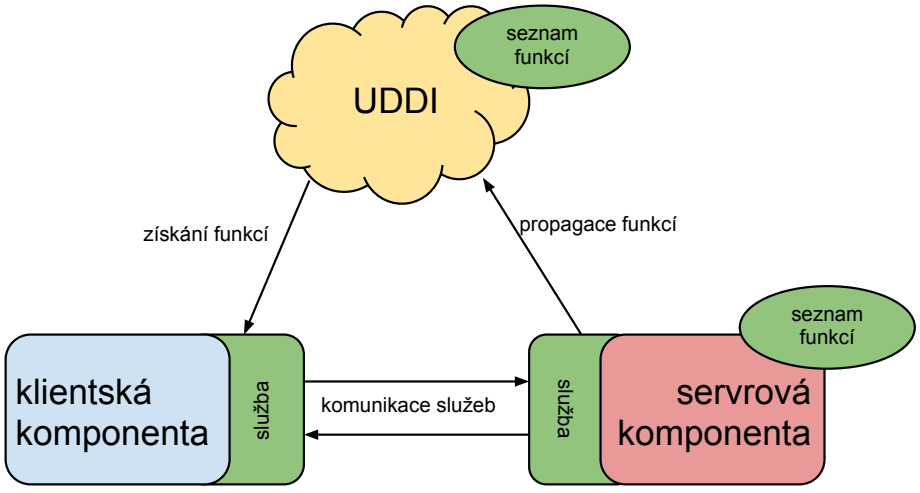
Jedná se o technologii pro záznam a přehrávání audiovizuálních záznamů. TUL ji využívá pro zachycení přednášek, které pak slouží studentům ke zopakování látky během studia. Technologie obsahuje jak hardwarové prvky, kterými jsou záznamová zařízení (videokamery, mikrofony, jednotka pro záznam přednáškových materiálů), tak softwarové vybavení, které nejen obsluhuje zmiňovaný hardware, ale také zprostředkovává zobrazení nahraných materiálů na webovém serveru.

Na TUL je nainstalován software MediaSite pro přehrávání záznamů ve verzi 5.5. Tato verze je již bohužel zastaralá a využívá pro zobrazení materiálů pouze technologii Silverlight od společnosti Microsoft. Tato technologie není ani částečně kompatibilní s platformou Android, a proto zobrazení přednášek v současné podobě na platformě Android není možné.

Nicméně v článku [8] společnost *Sonic Foundry, Inc.*, tvůrce technologie MediaSite, uvádí, že od verze 6.0.2 jejich softwaru je přehrávání na mobilních platformách, včetně platformy Android, možné díky podpoře HTML5 videí. Řešení nainstalovat vyšší verzi softwaru MediaSite se posléze ukázalo jako jediné možné, a proto byla klientská aplikace vyvíjena pro vyšší verzi přednáškového serveru na TUL a jeho správce byl požádán o instalaci nové verze.

3. Webové služby

Webové služby (anglicky *web services*) jsou jedním ze základních prvků obecně rozšířené architektury SOA (*Service-oriented architecture*, v překladu architektura orientovaná na služby). Jedná se o sadu principů a postupů, které popisuje vytváření aplikací bez ohledu na jejich velikost a složitost pomocí komponent, které na sobě nejsou závislé. Tyto komponenty obsahují unifikované spojovací služby. Služba je předem známá, má schéma předepsané určitým standardem a její definice a metody (respektive funkce) jsou strukturovaně popsány a uloženy pomocí UDDI (*Universal Description Discovery and Integration*) pro snadnou implementaci. Webové služby jsou označením skupiny spojovacích služeb, které využívají pro komunikaci protokol *TCP-IP*, ať už nezabezpečený (jako například *HTTP*) nebo zabezpečený (nejčastěji *HTTPS*). Celý princip webových služeb je popsán v [1].



Obrázek 3.1: Procesní schéma SOA

Architektura webových služeb je vždy server-klient. V praxi to znamená, že na jedné straně přenosu se nachází komponenta serverová (v případě této práce Moodle), která generuje seznam dostupných funkcí, přijímá požadavky na zpracování těchto funkcí a zprostředkovává odpovědi nebo chybová hlášení.

Klientská komponenta (zde aplikace pro Android) získá od severu seznam dostupných funkcí, vždy inicializuje spojení, sestavuje dotaz dle použitého protokolu, ve kterém volá konkrétní funkci, a předává jí příslušné parametry. Následně zpracuje a vyhodnotí odpověď. Celý proces je znázorněn v obrázku 3.1.

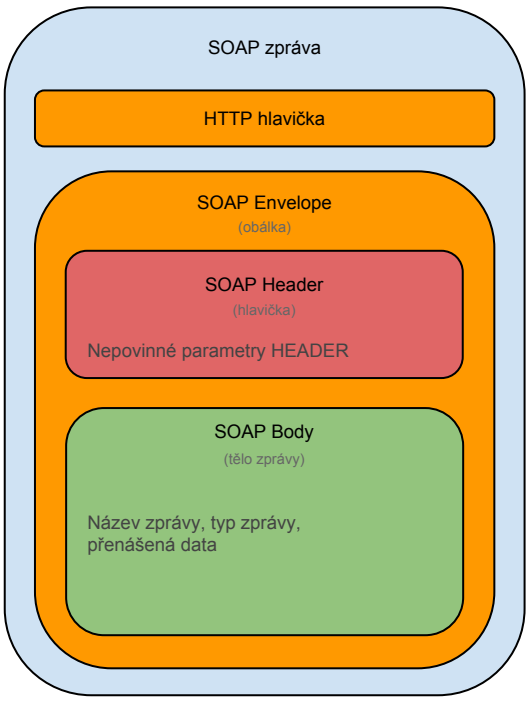
3.1 Protokol SOAP

Simple Object Access Protocol (SOAP) je považován za nejpoužívanější protokol z celé rodiny webových služeb. Byl definován v roce 2000 společností World Wide Web Consortium (W3C) [9]:

> *SOAP je jednoduchý protokol pro výměnu informací v decentralizovaném prostředí děleném na několik částí. Jedná se o protokol na základě XML specifikace, který se skládá ze tří částí: obálku, která určuje postupy a popisuje, co daná zpráva obsahuje a jak ji zpracovat, sadu kódovacích pravidel pro vyjádření předem definovaných datových typů a předpis pro reprezentaci vzdáleného volání procedur a jeho odpověď. SOAP lze potencionálně využít v kombinaci s mnoha dalšími protokoly, nicméně obecné použití je definováno v kombinaci s protokolem HTTP.*

Jeho specifikace obsahuje dvě nepatrně rozdílné verze, nicméně obě verze jsou používané jako standardní implementace služeb v SOA. Tyto dvě verze jsou SOAP v1.1 a SOAP v 1.2. Bez ohledu na obě specifikace SOAP verzí je zpráva protokolu SOAP založena na XML a vždy obsahuje *SOAP Envelope* (neboli obálku), volitelně *Header* (hlavičku) a musí obsahovat *Body* (tělo zprávy). Celá struktura zprávy v kombinaci s protokolem HTTP je znázorněna v obrázku 3.2.

Implementace webové služby SOAP tvoří novou cestu pro správu dat v systému Moodle. Díky zabezpečenému protokolu HTTPS, který byl nastaven (viz kapitola 2.1)



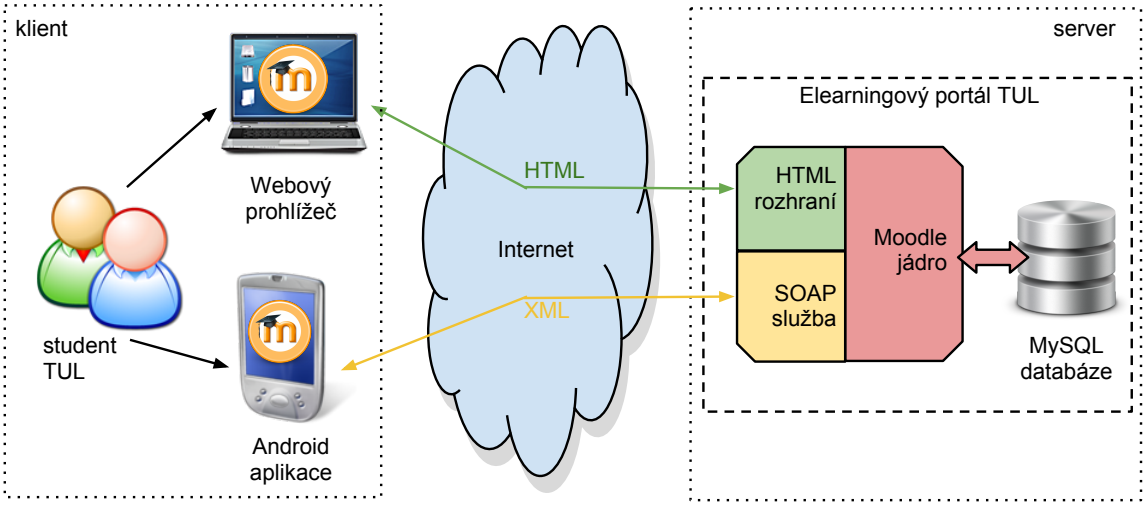
Obrázek 3.2: Schéma SOAP zprávy

pro celý systém je tento přenos dat bezpečnější než obvyklý přístup přes webový prohlížeč. Ten sice také využívá zabezpečeného spojení HTTPS, ale autentizace uživatele se provede jednou a poté už systém uživatele znova nekontroluje. U webové služby je nutno s každým požadavkem zaslat i *token*, který byl získán při autentizaci. Více je o této metodě přihlašování zmíněno v kapitole 3.4.2.

Klientská aplikace využívá stejných datových zdrojů a stejných procesních postupů v jádře systému jako při přístupu přes webový prohlížeč, pouze přenášená data jsou uložena ve struktuře XML místo HTML. Obrázek 3.3 znázorňuje tento rozdíl ve způsobu připojení i společné části.

3.2 Definice webové služby (WSDL)

Web Services Description Language (WSDL) je způsob strukturovaného popisu funkcí, jejich parametrů a datových typů webové služby SOAP. Má formát XML a může defi-



Obrázek 3.3: Schéma přístupu k e-learningovému portálu TUL

novat nejen dostupné metody, ale i datové struktury složitějších návratových objektů. Jedná se o specifické UDDI (viz kapitola 3) pro protokol webové služby SOAP. Je generován automaticky na základě implementace webové služby na serveru a usnadňuje implementaci webové služby v klientské aplikaci.

Bohužel systém Moodle ve své dosavadní verzi 2.4 negeneruje kompletní WSDL podle specifikace W3C. Tento problém je znám, ale zatím nebyl vývojáři systému vyřešen. Z toho důvodu nebylo možné implementovat webovou službu v klientské aplikaci automaticky, ale bylo zapotřebí sestavit její strukturu a volání jednotlivých funkcí podle dostupné dokumentace, kterou systém Moodle poskytuje. Tuto dokumentaci lze nalézt v systému Moodle v menu:

`Správa stránek >> Moduly >> Webové služby >> API Documentation`.

Dokumentace je pouze v anglickém jazyce a je dostupná pouze uživateli s administrátorskými právy.

3.3 Server

Systém Moodle, který zprostředkovává elearningový portál TUL (viz kapitola 2), obsahuje od své verze 2.0 základní implementaci čtyři různých protokolů webových služeb. Jsou jimi AMF (*Action message format*), REST (*Representational state transfer*), XML-RPC (*XML - remote procedure call*), který je předchůdcem SOAP a v neposlední řadě SOAP. Z těchto čtyř protokolů je právě SOAP nejpoužívanějším a komplexním řešením pro zpracování webových služeb a z toho důvodu byl vybrán jako komunikační protokol pro aplikaci.

3.3.1 Nastavení

Všechny webové služby jsou v základním nastavení systému Moodle vypnuté, a proto je zapotřebí změnit některá nastavení. K tomuto kroku je potřeba být přihlášen jako administrátor, nebo mít účet s administrátorskými právy.

- V prvním kroku je nutné povolit webové služby. Toto nastavení se nachází v menu: `Správa stránek >> Pokročilé funkce >> Povolit webové služby (enablewebservices)`.
- Dalším krokem je povolit samotný SOAP protokol. Toho se docílí kliknutím na ikonu přeškrtnutého oka  v menu: `Správa stránek >> Moduly >> Webové služby >> Správa protokolů >> SOAP`.
- Poté je zapotřebí vytvořit novou *externí službu*. Toto nastavení se nachází v menu: `Správa stránek >> Moduly >> Webové služby >> Externí služby`. Zde je potřeba vytvořit novou službu a přiřadit jí funkce. Službu lze vytvořit kliknutím na odkaz „*Přidat*“. Zde byla služba vytvořena s následujícími parametry: `Název: android_client, Povoleno: ano, Pouze oprávnění uživatelé: ano, Může stahovat soubory: ano, Požadovaná pravomoc: nezadáno`.
- K nově vytvořené službě je potřeba přiřadit dostupné funkce. Tato volba se nachází v řádku s nově vytvořenou funkcí pod odkazem „*Funkce*“. Pro klientskou

aplikaci bylo potřeba povolit tyto funkce (význam jednotlivých funkcí a jejich parametry jsou popsány v kapitole 3.4.3):

core_web_service_get_site_info
core_user_get_users_by_id
core_user_update_users
core_enrol_get_users_courses
core_course_get_contents

- Pro přístup do webové služby musí mít každý uživatel svůj unikátní *token*. Jedná se o 32bitový řetězec vygenerovaný systémem. Pokud by stačilo, aby k webové službě přistupovalo pár uživatelů, je možné jim ručně vytvořit *tokeny* v menu: `Správa stránek >> Moduly >> Webové služby >> Správa tokenů`. V rámci této práce však bylo zapotřebí, aby měl každý uživatel svůj vlastní *token*. Z toho důvodu bylo nutné povolit generování *tokenů* všem registrovaným uživatelům. V menu: `Správa stránek >> Uživatelé >> Oprávnění>> Definovat role` byla zvolena položka „*Registrovaný uživatel*“, jelikož každý student má právě tato oprávnění. Zde bylo potřeba vyhledat pravomoci „*Generovat token webové služby*“ a v editačním režimu zaškrtnout ve sloupci „*Oprávnění*“ volbu „*Povoleno*“. Poté ještě stejným způsobem povolit pravomoc a „*Vidět skryté detaily uživatelů*“ pro zobrazení adresy a telefonního čísla v profilu uživatele.

Těmito kroky je systém Moodle nastaven a připraven pro použití webové služby SOAP, která poskytuje základní funkce běžně dostupné studentům TUL.

3.3.2 Modifikace

Přestože systém Moodle obsahuje poměrně komplexní implementaci webových služeb, která je napojena přímo na funkce jádra celého systému, bylo potřeba provést úpravy přímo ve zdrojových souborech systému pro rozšíření možnosti funkcí pro potřeby klientské aplikace. Zdrojový kód systému Moodle není nikde podrobně zdokumentovaný, takže bylo zapotřebí projít velké množství souborů a řádků zdrojového kódu (převážně v jazyce PHP) pro lokalizaci částí vhodných pro úpravu. Všech-

ny provedené úpravy byly reportovány jako možná vylepšení na webový portál *https://tracker.moodle.org/*, který slouží vývojářům systému Moodle k reportování chyb a inovací.

Při implementaci funkce `core_user_update_users` bylo zjištěno, že tato funkce kontroluje, zda má uživatel oprávnění *„moodle/user:update"*. Toto oprávnění umožňuje uživateli měnit profily všech ostatních uživatelů. Tímto oprávněním disponuje pouze administrátor systému a přiřazení tohoto oprávnění ke všem studentským účtům by představovalo velké riziko zneužití této vlastnosti.

Z tohoto důvodu byl pozměněn zdrojový kód souboru *„/user/externallib.php"*. Na místo příkazu `require_capability ('moodle/user:update', $context);`, který provádí kontrolu zmiňovaného oprávnění, byla umístěna podmínka:

```
if ($user['id'] == $USER->id)
     require_capability ('moodle/user:editownprofile', $context);
else
     require_capability ('moodle/user:update', $context);
```

Proměnná `$user['id']` obsahuje identifikátor uživatele, kterému patří editovaný profil, zaslaný přes službu SOAP. Proměnná `$USER->id` obsahuje identifikátor přihlášeného uživatele. Podmínka potom určí, že pokud se tyto identifikátory rovnají, jedná se o editaci vlastního profilu a zkontroluje se pouze oprávnění *„moodle/user:editownprofile"*, které uživatele opravňuje k editaci vlastního profilu a je dostupné všem studentům TUL.

Během samotného ukládání profilových informací bylo v základní implementaci webové služby SOAP povoleno (z neznámých důvodů) měnit pouze pole *město* a *popis*. Jelikož systém Moodle ve svém webovém rozhraní umožňuje editovat i podrobnější informace, jako *adresa*, *telefon*, *mobil* a další, bylo nutné implementaci rozšířit.

Ve stejném zdrojovém souboru *„/user/externallib.php"*, ve kterém byla provedena první popsaná změna, byla upravena funkce `update_users_parameters()`, která vrací pole akceptovatelných informací pro editaci profilu. Do zmiňovaného pole byly přidány nové hodnoty korespondující s názvy požadovaných informací. Tato úprava stačila k rozšíření na požadovanou funkcionalitu.

Kromě těchto úprav, které se týkají pouze webových služeb, byl systém Moodle dále doplněn o možnost načítání obsahu modulu odesláním bezpečnostního *tokenu* metodou GET. Tento způsob je použit při získání dostupných kurzů ze systému STAG (více v kapitole 4.2). Dále byl vytvořen PHP skript pro upload souboru na server, který umožňuje klientské aplikaci zaslat soubor a uložit jej jako řešení tzv. offline úkolu. Skript se nachází ve složce *„/webservice/uploadassing.php"*.

3.3.3 Testování

Díky nekompletnímu WSDL (viz kapitola 3.2) bylo zapotřebí vyzkoumat jednotlivé funkce webové služby SOAP manuálně. Částečně bylo možné opřít se o dokumentaci, která je zmíněna také v kapitole 3.2, ale tato dokumentace je obecná pro všechny čtyři implementované služby v systému Moodle a popis jednotlivých funkcí zahrnuje jen obecnou strukturu a nikoliv konkrétní implementaci ve webové službě SOAP.

Druhým způsobem, jak zjistit co nejvíce informací o implementaci funkcí webové služby SOAP bylo použití testovacího modulu, který je součástí vývojářské části systému Moodle. Tato část je také dostupná pouze pod administrátorským účtem a nachází se v menu: `Správa stránek >> Vývoj >> Web service test client`. Popis modulu není lokalizován do českého jazyka a je pouze v angličtině.

Po vstupu do modulu má uživatel (administrátor) jako první možnost výběru *Autentizační metoda* ze dvou hodnot:

- *simple* - Je požadováno uživatelské jméno a heslo pro testovací volání funkce.
- *token* - K autentizaci do testované webové služby ze potřeba znát vygenerovaný *token* (viz kapitola 3.4.2);

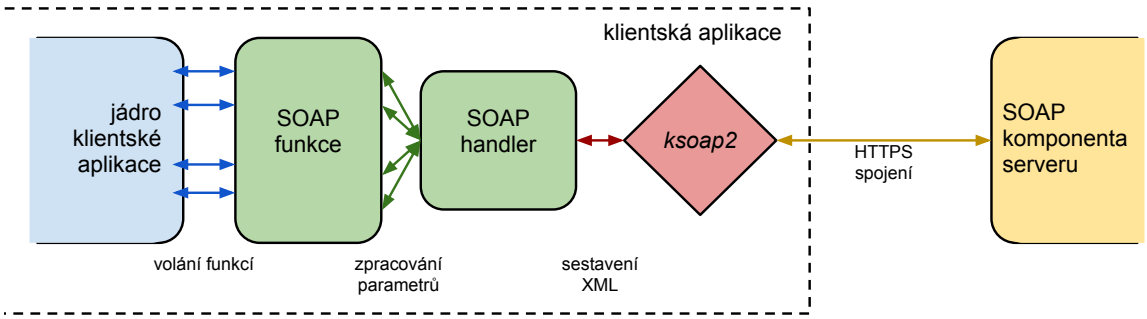
Dále je potřeba zadat protokol testované webové služby. Ve výběru jsou jen ty, které jsou v nastavení systému Moodle povoleny. A nakonec je nutné vybrat testovanou funkci. Testovací modul zdaleka neobsahuje všechny funkce webové služby, ale jen pár vybraných, které k testování musí postačit.

Po odsouhlasení výběru se objeví vstup pro zadání uživatelského jména a hesla, nebo *tokenu* podle předchozí volby, a další vstupy podle parametrů zvolené funkce k testování. Po jejich správném vyplnění a odeslání na server testovací modul zobrazí odpověď. Odpověď již není ve formátu XML, ale byla serverem předzpracována. Je zapsána v podobě pole jazyka PHP [`array($key => $value)`], ale je z ní patrné složení odpovědi a přijaté informace. Pokud bylo některé z polí špatně vyplněno, objeví se místo odpovědi tzv. *SoapFault*, neboli SOAP chyba s jejím popisem.

3.4 Klient

Součástí vytvořené mobilní aplikace pro platformu Android je klientská komponenta webové služby SOAP. Komponenta zajišťuje autentizaci uživatele oproti serverové části v systému Moodle, předzpracování dat, které slouží jako parametry požadovaných funkcí, sestavení celého volání dle specifikace SOAP do podoby XML, odeslání požadavku serverové komponentě v systému Moodle a zpracování informací z přijatého XML. Veškerá komunikace se systémem Moodle probíhá po internetu přes zabezpečený protokol HTTPS.

Celá komunikace komponenty SOAP se serverem byla rozdělena do tří základních kroků zpracování pro efektivnější implementaci nových funkcí. Schéma této komponenty je znázorněno na obrázku 3.4 a jeho dílčí části jsou podrobně popsány v následujících kapitolách.



Obrázek 3.4: Schéma třístupňové SOAP komponenty

3.4.1 Autentizace uživatele

Uživatel (neboli student TUL) je v klientské aplikace ověřován na základě uživatelského jména a hesla. Tyto údaje jsou identické s přihlašovacími údaji do e-learningového portálu TUL, jelikož i samotné ověřování probíhá na straně serveru systémem Moodle. Jedná se o přihlašovací údaje do sítě *LIANE*. Více informací o *LIANE* na *http://liane.tul.cz/*.

Samotné ověřování na serveru neprobíhá přes webovou službu SOAP, ale jedná se požadavek typu *GET*. Tento požadavek je jedním ze způsobu předávání proměnných protokolem HTTP/HTTPS z klientské aplikace na server. V systému Moodle je umístěn PHP skript, který je standardní součástí systému a slouží ke generování tzv. *tokenů* (viz kapitola 3.3). Skript nese název *token.php* a je umístěn ve složce *login* kořenového adresáře systému Moodle.

Skript přijímá prostřednictvím *GET*u tři parametry. Uživatelské jméno pod názvem `username`, heslo pod názvem `password` a službu pod názvem `service`. Parametr `service` slouží ke specifikaci *externí služby*, která obsahuje povolené funkce. Pro účely klientské aplikace byla vytvořena *externí služba* s názvem `android_client`. Jak nastavit *externí službu* je popsáno v kapitole 3.3. Kompletní HTTPS dotaz na ověření uživatele poté vypadá takto:

```
https://elearning.tul.cz/login/token.php?username=jméno&password=heslo
&service=android_client
```

Pokud uživatel vyplnil přihlašovací jméno a heslo korektně, skript *token.php* vrátí token ve formě 32bitového řetězce zakódovaný ve formátu JSON. Jedná se o standardní serializaci dat. Pro jeho deserializaci byl v klientské aplikaci použita knihovna *JSONParser*, která je obsažena mezi základními vývojovými prostředky platformy Android (tzv. SDK). Odpověď na správně zadané údaje tedy vypadá například takto:

```
{"token":"58b51312f2258521f83336624cceea98"}
```

Token si aplikace uloží a zachová jej v paměti po celou dobu přihlášení uživatele, jelikož je tento token vyžadován u každého volání jednotlivých funkcí SOAP. Tím je navýšena bezpečnost přenosu informací mezi systémem Moodle a klientskou aplikací.

Pokud uživatel zadá špatné uživatelské jméno nebo špatné heslo, skript *token.php* vrátí následující chybu a aplikace tak pozná, že uživatel zadal nesprávné přihlašovací údaje a uživateli to oznámí.

```
{"error":"Uživatelské jméno nebylo nalezeno v databázi.",
"stacktrace":null,"debuginfo":null,"reproductionlink":null}
```

Po pečlivém zvážení byla uživateli ponechána možnost uložit v aplikaci jak uživatelské jméno, tak heslo. Platforma Android bohužel umožňuje zašifrovat uložená data pouze dalším heslem. V případě, že by bylo toto heslo uloženo ve zdrojovém kódu aplikace, může jej případný útočník vyčíst a data dešifrovat, jelikož existují nástroje na dekompilaci zdrojového kódu aplikace. Heslo je proto před uložením převedeno do kódování *Base64* a drobnou úpravou pozměněno tak, aby co nejvíce znesnadnilo jeho dekódování. Nicméně útočník může ze zdrojového kódu aplikace zjistit postup pro jeho dekódování a toto zabezpečení jen ztíží jeho snažení. Ponechat uživateli volbu uložení hesla bylo rozhodnuto na základě faktu, že k jeho získání je potřeba fyzický přístup k mobilnímu zařízení a značný zásah do samotného systému Android.

3.4.2 Implementace funkcí

Jednotlivé funkce a jejich implementace by se při kompletním WDSL sestavily za pomocí automatického generátoru a nebylo by nutné znát jejich parametry, datové typy, ani strukturu odpovědi. Bohužel s nekompletním WSDL toho nebylo možné dosáhnout, a proto bylo nutné implementovat vlastní zpracování.

Všechny metody byly naprogramovány jako veřejné metody třídy `MoodleSoapFunctions.java`, která při vytváření instance požaduje jako jediný parametr konstruktoru bezpečnostní *token*. Ten je v objektu uložen a při volání jednotlivých metod pak není nutné jej znovu zadávat. Jednotlivé metody se od sebe liší

především strukturou vstupních parametrů a strukturou odpovědi ze serveru. Jejich implementace je popsána v následujícím přehledu jednotlivých metod.

- `getUser()` - tato metoda je volána ihned po autentizaci uživatele a získání bezpečnostního *tokenu*. Využívá SOAP funkce `core_web_service_get_site_info`, která nemá žádné povinné parametry a v odpovědi jsou uloženy základní informace o celém systému Moodle a základní informace o uživateli, kterému patří přijatý *token*. Pro uložení informací byla vytvořena třída `User.java`, která obsahuje atributy: *username* - uživatelské jméno, *firstname* - křestní jméno uživatele, *lastname* - příjmení uživatele, *pictureURL* - cesta k profilovému obrázku a *userID* - identifikátor uživatele v rámci systému Moodle. Poslední atribut je potřeba k získání zapsaných kurzů a podrobnějších informací o uživateli. Ostatní atributy složí pouze k personalizaci aplikace (zobrazení uživatelova profilového fota a oslovení uživatele na úvodní obrazovce jménem; bohužel jméno je uvedeno pouze v prvním pádu).

- `getProfile(int userID)` - metoda vrací kompletní informace o profilu uživatele, jediným parametrem je identifikátor uživatele. Využívá SOAP funkci `core_user_get_users_by_id`, která vrací informace o jenom nebo více uživatelích a záleží na pravomocech uživatele, který ji volá, zda informace dostane. Volání metody vyžaduje jeden vstupní parametr a tím je pole *userids*, které obsahuje unikátní identifikátory uživatelů v systému Moodle. Parametr musí být vždy typu pole. Proto je parametr *userID* převeden na jednoprvkové pole. Klientská aplikace využívá tuto metodu pro zobrazení detailního profilu studenta, kde má student rovnou možnost některé své údaje měnit. Data jsou načtena do instance třídy `Profile.java`, která dědí atributy od třídy `User.java` a přidává nové.

- `setProfile(Profile data)` - tato metoda umožňuje editaci uživatelských dat. Protože systém Moodle nepoužívá vlastní databázi uživatelů, ale je napojen databázi TUL pod správou *LIANE* (univerzitní síť), není v systému Moodle dovoleno uživatelům měnit jejich základní údaje, jako *křestní jméno*, *příjmení*, *email*, *uživatelské jméno* nebo *heslo*. Tyto údaje jsou společné pro mnoho aplikací na TUL a

jejich editací by mohlo dojít k narušení integrity celé sítě. Z toho důvodu je tato metoda použita pouze pro editaci přidružených, nepovinných informací, jako *adresa*, *město*, *mobilní telefon* a další. Jediný parametr této metody *data* obsahuje profilové údaje, které jsou převedeny do podoby XML dat pro zpracování. Metoda využívá SOAP funkci `core_user_update_users`, která tato data přijme v argumentu a uloží je. Součástí objektu *data* musí být atribut *userid*, podle kterého jsou informace přiřazeny k uživateli.

- `getCourses(int userID)` - metoda načte seznam všech kurzů / předmětů, do kterých je student zapsán. Vstupní parametr *userID* je identifikátorem uživatele, jehož kurzy jsou požadovány. Metoda vrátí pole objektů třídy `Course.java`, která obsahuje *krátký a dlouhý název* (oba slouží k popisu předmětu) a *unikátní identifikátor*, který slouží k jednoznačnému určení kurzu v systému Moodle a na jeho základě lze získat obsah tohoto kurzu. Metoda je napojena na SOAP funkci `core_enrol_get_users_courses`, která rovněž vyžaduje jeden vstupní parametr *userid* pro identifikaci uživatele. Metoda je využita pro získání dostupných kurzů na domovské obrazovce, ve které se uživatel ocitne ihned po přihlášení.

- `getCourseContent(int courseID)` - tato metoda navrací obsah požadovaného kurzu. Vstupním parametrem *courseID* je identifikátor kurzu a metoda vrací pole objektů třídy `CourseItem.java`, které uchovávají jednotlivé části, do kterých je kurz standardně rozdělen. Každý objekt z návratové hodnoty pak obsahuje pole objektů třídy `CourseModul.java`, které obsahuje jednotlivé *moduly* v každé části. *Modulem* je zde nazvána jakákoliv interakce mezi systémem Moodle a uživatelem. Může se jednat o statický text, soubor ke stažení, odkaz na nahranou přednášku nebo jiný web, informační stránku nebo úkol zadaný vyučujícím. Podle typu obsahu klientská aplikace rozhodne, co s danou informací udělá a jak ji prezentuje uživateli. Při získávání informací se systému Moodle byla využita SOAP funkce `core_course_get_contents`, která má také pouze jeden vstupní parametr *courseid* určující identifikátor požadovaného kurzu. Tato metoda je klíčová pro zobrazení informaci ze systému Moodle a použití zmiňovaných

tříd je rozepsáno dále v textu zprávy.

3.4.3 SOAP handler

Druhým důležitým článkem schématu, který je znázorněn na obrázku 3.4, je *SOAP handler*. Jedná se o třídu `SOAPHandler.java`, která vznikla pro zjednodušení implementace konkrétních metod, které byly popsány v předchozí kapitole 3.4.2. Tato třída obsahuje jednou statickou metodou, která slouží k inicializaci a správnému zpracování metod knihovny *ksoap2*.

Odeslání požadavků na systém Moodle za pomocí knihovny *ksoap2* probíhá vždy stejně, a proto tento postup mohl být unifikován v rámci jedné statické metody `callService(...)`. Jednotlivé parametry metody jsou popsány v následujícím přehledu:

- `token` - datový typ *String* (řetězec znaků), bezpečnostní *token* získaný při autentizaci uživatele o délce 32 znaků
- `METHOD_NAME` - datový typ *String*, název požadovené SOAP funkce
- `param` - datovy typ *Object* (univerzální datový typ, všechny třídy jsou potomkem této), parametr volané funkce, jeho datový typ je určen podle parametru volané SOAP funkce
- `paramname` - datový typ *String[]* (pole řetězců), určuje název / názvy parametrů volané SOAP funkce
- `asarray` - datový typ *boolean*, nepovinný parametr, výchozí hodnota je *false*, pokud je jeho hodnota nastavena na *true*, pak je proměnná *param* obalena jednorozměrným polem. Některé z SOAP funkcí vyžadují předávat parametry jako pole.

Z těchto parametrů je poskládán celý dotaz a následně odeslán na server. Nejdříve je ale nutné poskládat dotaz s HTTPS hlavičkou a data do XML dle definice SOAP (viz obrázek 3.2). Je nutné sestavit tzv. *namespace*. Jde o URL adresu specifickou pro danou

webovou službu SOAP. Je sestavena z adresy systému Moodle, ze skriptu zpracovávajícího SOAP volání, který je na adrese „*/webservice/soap/server.php*“ , a z bezpečnostního *tokenu* předávaného pomocí GET. Příklad konkrétní *namespace* implementace v klientské aplikaci vypadá následovně:

```
https://elearning.tul.cz/webservice/soap/server.php?
wstoken=58b51312f2258521f83336624cceea98
```

Pomocí takto sestaveného *namespace* je vytvořen objekt z knihovny *ksoap2* zajišťující HTTPS komunikaci. Dále je vytvořena *SOAP Envelope* (XML obálka) definující verzi 11, jelikož byla při vývoji ověřena kompatibilita této verze se SOAP implementací v systému Moodle. Poté je vytvořeno tělo zprávy obsahující vygenerovaný *namespace*, název volané SOAP funkce a její předávané parametry. Na závěr bylo nutné nastavit parametr obálky *dotNet* na hodnotu *true*, což zajišťuje kompatibilitu kódování vygenerovaného XML se serverem. Bez nestavení této vlastnosti docházelo k chybám při deserializaci odpovědí. Příklad vygenerované SOAP zprávy ve formátu XML s voláním funkce `core_user_get_users_by_id`:

```
<v:Envelope xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:d="http://www.w3.org/2001/XMLSchema"
            xmlns:c="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:v="http://schemas.xmlsoap.org/soap/envelope/">
    <v:Header />
    <v:Body>
        <core_user_get_users_by_id
            xmlns="http://moodle/webservice/soap/server.php?wstoken=…"
            id="o0" c:root="1">
            <n0:userid i:type="n0:userid"
                 xmlns:n0="http://moodle/webservice/soap/server.php?…">
                 <item i:type="d:string">3</item>
            </n0:userid>
        </core_user_get_users_by_id>
    </v:Body>
</v:Envelope>
```

Celá takto vytvořená zpráva je odeslána přes HTTPS na server na adresu, která se shoduje s *namespace* vytvořeným dříve, pouze se na její konec přidá znak „#“ a název volané funkce. Odpověď ze serveru je zpracována opět knihovnou *ksoap2* a ta ji vrátí

v podobě svého objektu třídy `SoapObject.java`. Jednotlivé třídy knihovny *ksoap2* jsou popsány v následující kapitole 3.4.4. Příklad příchozí SOAP odpovědi na volání funkce `core_enrol_get_users_courses` ze serveru před zpracováním:

```
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ns1="http://moodle/webservice/soap/server.php?wstoken=…"
    xmlns:ns2="http://xml.apache.org/xml-soap"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <ns1:core_enrol_get_users_coursesResponse>
        <return SOAP-ENC:arrayType="ns2:Map[2]" xsi:type="SOAP-ENC:Array">
            <item xsi:type="ns2:Map">
                <item>
                    <key xsi:type="xsd:string">id</key>
                    <value xsi:type="xsd:int">3</value>
                </item>
                <item>
                    <key xsi:type="xsd:string">shortname</key>
                    <value xsi:type="xsd:string">test 2</value>
                </item>
                <item>
                    <key xsi:type="xsd:string">fullname</key>
                    <value xsi:type="xsd:string">Testovací kurz 2</value>
                </item>
                    <key xsi:type="xsd:string">visible</key>
                    <value xsi:type="xsd:int">1</value>
                </item>
                </item>
            </return>
        </ns1:core_enrol_get_users_coursesResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.4.4 Knihovna ksoap2

Jak již bylo zmíněno v kapitole 3.1, výměna informací mezi klientskou částí a serverem probíhá v podobě XML souboru sestaveného dle předepsaného schématu W3C. O sestavení dat do XML, jeho odeslání a následné zpracování odpovědi zajišťuje knihovna

ksoap2, která byla implementována jako součást SOAP komponenty v klientské aplikaci a její umístění znázorňuje schéma na obrázku 3.4.

Jedná se o open-source knihovnu vydanou pod *MIT licencí*, která opravňuje kohokoliv využívat tuto knihovnu zdarma pro jakékoliv účely, jak popisuje znění licence v [5]. Obsahuje velké množství dostupných tříd pro práci se SOAP webovou službou a jen některé jsou použity v implementaci v klientské aplikaci. Veškeré instance těchto tříd a volání jejich parametrů obstarává třída *SOAPHandler*, která je popsána v předchozí kapitole 3.4.3. Jedná se o třídy:

- `SoapObject.java` - obecná třída obsahující nějaká data, obsahuje název, *namespace* a může obsahovat parametry a další objekty svého typu.
- `SoapSerializationEnvelope.java` - tato třída zajišťuje vytvoření *SOAP envelope* (obálky) v podobě XML a serializaci všech dat uložených v objektech třídy `SoapObject`. V této třídě se nastavuje SOAP verze 11 (viz kapitola 3.1) a atribut *dotNet* na hodnotu *true* (viz kapitola 3.4.3).
- `HttpsTransportSE.java` - tato třída zajišťuje přenos XML dat na server a zpět po zabezpečeném protokolu HTTPS

4. Tvorba klientské aplikace

Hlavním cílem této práce bylo vyvinout mobilní aplikaci pro platformu Android, která bude sloužit studentům TUL pro připojení k univerzitnímu e-learningovému portálu a umožní jim snadný přístup k informacím, které jsou na něm uloženy. Nejnižší kompatibilní verze platformy Android s klientskou aplikací je verze 4.0.

Aplikace umožňuje studentovi TUL základní činnosti dostupné přes webové rozhraní e-learningového portálu TUL, které jsou běžně na TUL využívány. Portál obsahuje další škálu činností, nicméně implementace všech by byla časově velmi náročná a vyžadovala by velký zásah do zdrojových kódů systému Moodle. Proto byl výběr implementovaných činností zúžen na ty, které se na e-learningovém portálu TUL používají nejčastěji a které student ocení a využije ve svém mobilním zařízení. Jsou jimi:

- *zobrazení zapsaných kurzů* (systém Moodle označuje školní předměty jako *kurzy*)
- *zobrazení a editace profilových informací*
- *zápis nových kurzů pomocí systému STAG*
- *zobrazení obsahu kurzu včetně statických textů*
- *stahování nahraných materiálů*
- *nahrávání offline úkolů*
- *zobrazení přednášek ze serveru MediSite*

Důraz byl kladen na bezpečnost přenosu dat mezi aplikací a serverem a na přehlednost prezentovaných informací na všech mobilních zařízeních bez ohledu na rozměry jejich zobrazovacích jednotek. Aplikace využívá webové služby SOAP, díky které dochází k výměně informací s e-learningovým portálem TUL. Implementace webové služby SOAP je popsána v kapitole 3.4.

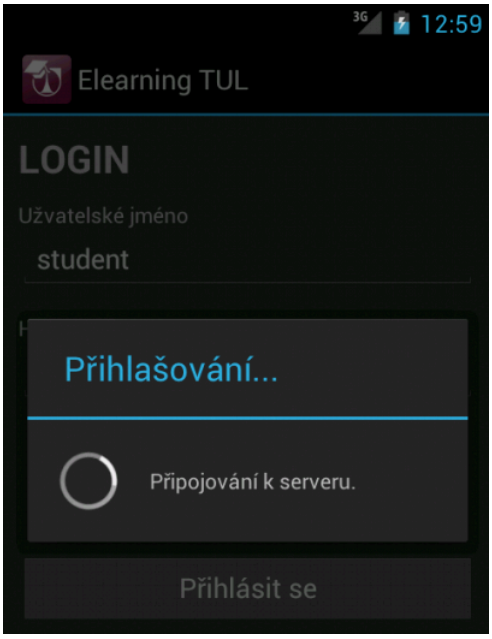
Pří vývoji klientské aplikace bylo využito vývojového prostředí (IDE) *IntelliJ IDEA 12 CE* od společnosti JetBrains, sada vývojových nástrojů Android SDK v14. Aplikace byla testována na několika virtuálních zařízeních s různými rozměry displejů, které jsou součásti SDK, a na mobilním telefonu HTC Desire Z.

4.1 Asynchronní připojení na server

Jelikož komunikace za použití webových služeb probíhá prostřednictvím protokolu HTTPS, v závislosti na rychlosti internetového připojení v mobilním zařízení dochází k prodlevám v odpovědi. Při vývoji a testování klientské aplikace byl internetový přenos informací nasimulován pouze po místní síti LAN, ale i přesto trvaly některé požadavky a jejich zpracování v řádu sekund. S přihlédnutím k obvyklému internetovému připojení většiny mobilních zařízeních, které je stále v některých odlehlejších místech realizováno pomocí technologie Edge, by se požadavky na server mohly vyšplhat až na desítky sekund.

Po dobu komunikace se serverem musí být uživatel informován, že komunikace stále probíhá, že aplikace čeká na odpověď ze serveru. Navíc nesmí dojít k zastavení aplikace po dobu spojení, aby bylo možné aplikaci zastavit během přenosu. Z tohoto důvodu jsou veškerá spojení se serverem uskutečňována v samostatném vlákně, což zajistí plynulý běh aplikace. Během komunikace se serverem se uživateli na obrazovce objeví dialogové okno. Obsahuje krátkou informaci o tom, co aplikace provádí a systémovou komponentu znázorňující činnost aplikace (dva rotující body v kružnici, viz obrázek 4.1).

Vytváření nových vláken na platformě android lze vyřešit dvěma způsoby. Prvním způsobem je instance třídy `Thread.java`. Tento způsob je standardní cestou pro vytvoření nového vlákna v jazyce Java. Druhý způsob je součástí Android SDK a je specifický pro tuto platformu. Jedná se o rozšíření (zdědění) třídy `AsyncTask.java`, která nabízí širší možnosti implementace. Tato třída obsahuje tři abstraktní metody, které je nutné implementovat. Pro spuštění stačí vytvořit instanci nové třídy a zavolat metodu



Obrázek 4.1: Dialogové okno klientské aplikace

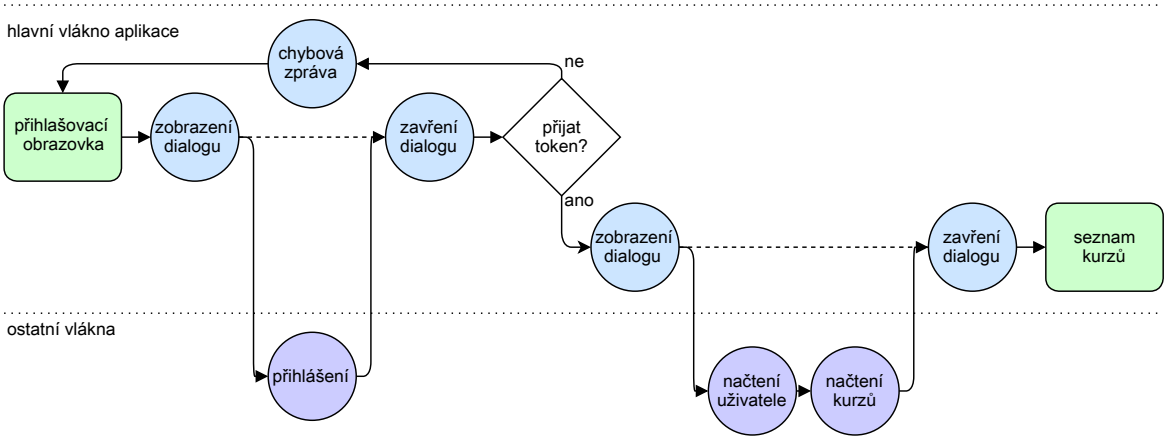
`MyAsyncTask.execute(Object... params)`. Jako parametr této metody může být libovolné množství proměnných typu Object. Abstraktní metody pro implementaci jsou tyto:

- `onPreExecute()` - tato metoda je zavolána ihned po volání metody `execute`, ale ještě před vytvořením nového vlákna. Běží tedy v hlavním vlákně aplikace a byla použita pro inicializaci dat potřebných ke spojení a vyvolání dialogového okna informujícího uživatele o probíhající aktivitě.
- `doInBackground(Object... params)` - tato metoda je spuštěna po skončení metody `onPreExecute` a jako jediná běží v samostatném vlákně. Obsahuje volání metod komunikujících přes SOAP nebo zprostředkovávajících jinou komunikaci po internetu. Vstupním parametrem metody je pole objektů typu *Object*. Jedná se o objekty vložené jako parametry při volání funkce `execute` a délka pole záleží na počtu předaných parametrů. Typ návratové hodnoty této metody je *Object*.
- `onPostExecute(Object o)` - tato metoda se provede po skončení metody `doInBackground` a opět se již provádí v hlavním vlákně aplikace. To jí umožňu-

je měnit komponenty na obrazovce, a proto zde probíhá nastavení komponent aplikace podle odpovědi serveru. Jediným vstupním parametrem je objekt typu *Object*, který vrátila metoda `doInBackground`. Obvykle obsahuje informace přijaté ze serveru.

- `onProgressUpdate(Object... params)` - tato metoda slouží pro synchronizaci nového vlákna s hlavním vláknem aplikace. Je volána z hlavního vlákna vždy, když v novém vlákně dojde k volání metody `publishProgress()`. Počet parametrů této funkce je neomezený. Tato funkce byla využita pro zobrazení stavu odesílání souboru na server, kdy uživatel vidí velikost již odeslané části v procentech a má možnost odesílání zrušit.

Třída `AsyncTask.java` obsahuje i další metody, které ale neměly pro účely aplikace žádný význam, a proto nebyly implementovány. Schéma v obrázku 4.2 znázorňuje část přihlášení uživatele po zadání uživatelského jména a hesla.



Obrázek 4.2: Přihlášení uživatele s použitím vláken

4.2 Zápis předmětů ze STAG

Elearningový portál TUL využívá spojení s dalšími portály TUL pro výměnu informací. Jedním takovým spojením je napojení na systém STAG. Jedná se o informační systém

obsahující informace o studiu všech studentů na TUL a elearningový portál jej využívá pro automatické zapisování do jednotlivých kurzů (předmětů).

Rozšíření systému Moodle pro spojení se systémem STAG není standardní součástí instalace. Jedná se o komponentu, kterou na TUL vytvořil Ing. Igor Kopetschke. Komponenta nabízí přihlášenému uživateli zapsat kurzy, ve kterých zatím není zapsán, ale dle systému STAG je studuje, nebo je studoval. Alternativním způsobem je zapsání studenta vyučujícím daného předmětu.

Jelikož se nejedná o základní součást systému Moodle, neexistuje jeho rozhraní pro webovou službu. Z tohoto důvodu není tato funkce (jako jediná) realizována přes webovou službu SOAP. Komponenta pro napojení systému STAG byla upravena tak, aby akceptovala bezpečnostní *token* předaný za pomocí *GET*. Pokud uživatel není přihlášen ve webovém rozhraní, komponenta zkontroluje, zda nedostala *token* a pokud ano, načte informace o uživateli z databáze pomocí tohoto *tokenu*. URL adresa komponenty s *tokenem* vypadá takto:

https://elearning.tul.cz/mod/stag/view.php?token=58b51312f2258521f83336624cceea98

Komponenta vypíše dostupné kurzy k zapsání stejně jako ve webovém rozhraní. Klientská aplikace poté přijme HTML kód, který by se zobrazil uživateli ve webovém prohlížeči, vyhledá v něm dostupné kurzy za pomocí parsování HTML a zobrazí je uživateli k zapsání.

Poté co uživatel vybere alespoň jeden kurz a stiskne tlačítko „*Zapsat*“, aplikace znova vyšle požadavek na komponentu propojení STAGu s *tokenem* předaným metodou GET. Požadované kurzy k zapsání se odešlou v požadavku metodou POST. Komponenta studenta opět přihlásí díky zaslanému *tokenu* a vybrané kurzy jsou zaslány metodou POST. Tím bylo nasimulováno odeslání HTML formuláře pro zapsání kurzů tak, jak by to proběhlo přes webové rozhraní.

Touto jednoduchou úpravou současné komponenty bylo umožněno studentům zapisovat kurzy v klientské aplikaci bez nutnosti velkých zásahů do fungování systému

Moodle. Celý proces načítání dostupných kurzů a jejich zapisování probíhá v samostatném vlákně aplikace, stejně jako tomu je u komunikace přes webovou službu SOAP.

4.3 Responzivní layout

Jak již bylo zmíněno v úvodu, jedním z hlavních důvodů, proč je platforma Android tolik rozšířená, je množství zařízení, na kterých lze tento operační systém spustit. Lze jej najít na mobilních telefonech s 2,5" displejem i tabletech s displejem větším, než 10". Jejich rozlišení se pak v závislosti na DPI (dots per inch - hustota obrazových bodů na palec) pohybuje od 240x320 pixelů do 2560x1600 pixelů. Tato rozlišení mají navíc různé poměry stran a každé zařízení se může pohybovat v poloze *portrait*, kdy je delší strana zařízení ve vertikální ose (častěji mobilní telefony), a po otočení o 90° se zařízení ocitne v poloze *landscape*, kdy je delší strana v horizontální ose (častěji u tabletů). Měnění těchto poloh je ve většině případů určují gyroskopické senzory v zařízení a reagují na jeho natočení uživatelem.

Jelikož má klientská aplikace zlepšit, zjednodušit a zpřehlednit studentům práci a informace na e-learningovém portálu TUL, musí všechny zmiňované vlastnosti mobilních zařízení zohledňovat a zajistit, aby zobrazované informace zůstaly v ucelené podobě bez ohledu na rozlišení, DPI nebo poměr stran zobrazovacích jednotek mobilních zařízení.

Layout (z angl. rozložení) jednotlivých prvků na obrazovce je při vývoji aplikací na platformu Android zapisován ve formátu XML. Jednotlivé komponenty (jako například tlačítka, popisky atd.) jsou zapisovány jako XML tagy a veškeré jejich vlastnosti (jako rozměry, barva apod.) jsou zapsány pomocí atributů těchto tagů. Kořenový tag je vždy pouze jeden, obsahuje *namespace* pro validaci všech dostupných tagů a jejich atributů a vždy se jedná o prvek typu *layout*. Výběr konkrétního rozložení je důležitý pro následné zobrazení všech komponent, které obsahuje. V klientské aplikaci byly použity dva typy rozložení:

- `LineraLayout` - rozložení obsahuje základní vlastnost *orientation*, která nabývá dvou hodnot *horizontal* (jako výchozí) a *vetrikal*. Obsažené prvky uvnitř rozložení se řadí vedle sebe, nebo pod sebou v závislosti na hodnotě *orientation*.

- `RelativeLayout` - rozložení, které určuje pozice prvků uvnitř s ohledem na ostatní prvky. Každému z obsažených prvků lze přiřadit pozici vůči okrajům rozložení atributem *layout:alignParent*, nebo vůči pozici jiného prvku atributem *layout:alignComponent*.

Tato rozložení tudíž nejsou pevně závislá na velikosti ani poměru stran displeje. Rozměry jednotlivých komponent (výška a šířka) byly také udány relativně hodnotami:

- `match_content` - rozměr prvku v daném směru určuje jeho obsah (například text), pokud je jeho obsah prázdný, nemá v tomto směru prvek velikost.

- `fill_parent` - rozměr prvku v daném směru je určen velikostí jeho rodičovského prvku. Jedná-li se o kořenový prvek, přejímá rozměry displeje.

Díky tomu, že pozice a velikosti prvků byly nastaveny relativně a ne za pomocí konkrétních hodnot, bylo zajištěno vykreslení komponent bez závislosti na vlastnostech displeje. Navíc byl zvolen nejmenší možný displej na testovacím virtuálním zařízení proto, aby byly všechny komponenty s jistotou zobrazeny na obrazovce.

Jedinou výjimku tvoří obrázky. Jejich velikost je definována v hodnotách *dp* (Density-independet Pixel). Jedná se o sjednocenou velikost prvků na displeji bez závislosti na DPI displeje. Přepočet mezi *dp* a obrazovými body (*pixely)* je

$$S_{dp} = \lceil \frac{S_{px} * 160}{displayDensity} \rceil$$

kde S_{dp} je výsledný rozměr v *dp*, S_{px} je skutečný rozměr v *pixelech* a $displayDensity$ je hustota obrazových bodů displeje určující jeho DPI. Jako referenční hustota displejů na platformě Android byla zvolena tvůrci platformy $160DPI$, proto při této hustotě platí, že $S_{dp} = S_{px}$.

Při testování změny polohy aplikace bylo zjištěno, že nedochází pouze k překreslení jednotlivých komponent na displeji, ale k celému resetu právě zobrazené aktivity (více o aktivitách v následující kapitole 4.4). Jelikož u většiny aktivit probíhá při spuštění asynchronní načítání informací ze serveru, docházelo při otočení zařízení k opětovnému spojení se serverem. Toto byl nežádoucí jev a musel být odstraněn.

Pro jeho vyřešení byla použita metoda `onRetainNonConfigurationInstance()`, která má návratovou hodnotu typu *Object*. Metoda byla implementována v každé aktivitě, která získává informace ze serveru, aby vracela právě ony získané informace. Pokaždé, když je změněna konfigurace aktivity a je potřeba ji resetovat (například při změně orientace displeje), je před jejím ukončením zavolána právě tato funkce, která uloží všechna získaná data do dočasné instance aktivity. Po opětovném spuštění aktivity je získána její dočasná instance zavoláním metody `getLastNonConfigurationInstance()`. Poté aktivita zkontroluje, zda je dočasná instance prázdná a pokud zjistí, že obsahuje data, uloží je a nemusí zahajovat spojení se serverem. Toto řešení v kódu vypadá takto (část konstruktoru aktivity):

```
Content data = (Content) getLastNonConfigurationInstance();
if (data == null)
    new LoadTask().execute();
else
    this.init(data)
```

Tím byl efektivně vyřešen problém, při kterém docházelo k opětovnému načítání již načtených dat při otočení zařízení uživatelem o 90°.

4.4 Aktivity

Každá aplikace na platformě Android se skládá z tzv. *aktivit*. Jedná se o ucelenou část aplikace, která je definována třídou, která je potomkem třídy `Activity`, a k ní přidruženým rozložením obsažených komponent v podobě XML souboru (více o tomto XML souboru v předchozí kapitole 4.3). Zjednodušeně lze aktivitu nazvat *obrazovkou*, neboť vždy vyplňuje celý displej zařízení.

Aplikace má vždy jedu výchozí aktivitu definovanou ve speciálním XML souboru s názvem *AndroidManifest.xml*. Tento soubor obsahuje základní informace o aplikaci, jakými jsou její jméno, požadovaná oprávnění od systému, minimální kompatibilní verze systému, zmiňovanou výchozí aktivitu, seznam zbylých aktivit a mnoho dalších. Každá aktivita obsahuje řadu metod pro zpracování událostí systému, které mohou a nemusejí být implementovány. Klientská aplikace využívá tyto:

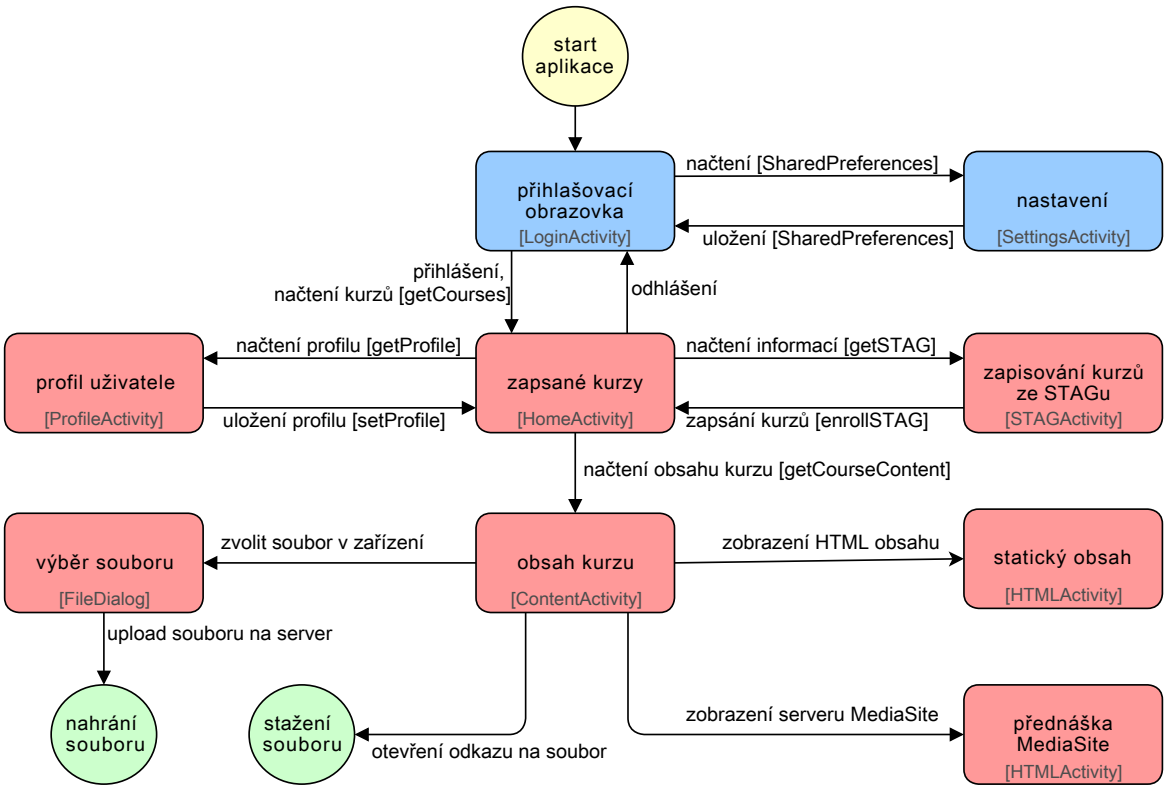
- `onCreate()` - metoda volaná při startu/restartu aktivity, obsahuje veškerou svoji inicializaci a volání asynchronní komunikace se serverem
- `onCreateOptionsMenu()` - metoda volaná při stisku tlačítka *Menu* na zařízení, definuje položky a vzhled menu dané aktivity. Každá aktivita může mít jiné menu, nebo nemusí mít žádné.
- `onOptionsItemSelected()` - metoda volaná při výběru položky v menu, jejím vstupním argumentem je ID položky
- `onActivityResult()` - metoda volaná při ukončení aktivity spuštěné metodou `startActivityForResult()` (více o této funkci dále v této kapitole)
- `onRetainNonConfigurationInstance()` - funkce volaná před restartem aktivity, její výstup se uloží do dočasné instance aktivity

Každá aktivita na platformě Android může inicializovat další aktivity. To se provádí funkcí `startActivity(Intent intent)`. Argumentem této metody je objekt typu `Intent`, který zajišťuje přepínání aktivit. Jeho inicializace vypadá takto: `new Intent(Context context, Class newActivity)`. První argument určuje *context*, což je ve většině případů objekt právě spuštěné aktivity. Pokud je tedy nová aktivita spouštěna z aktuální aktivity, stačí zadat pouze hodnotu `this`. Druhým argumentem je odkaz na třídu nově spouštěné aktivity.

Pokud je potřeba získat výsledek nějaké činnosti po skončení nové aktivity, je nutné použít metodu `startActivityForResult(Intent intent, int requestCode)`. Druhý parametr metody slouží k identifikaci výsledku při jeho vyhodnocení. Aktivita

je ukončena metodou `finish()`. Poté se stává aktivní její rodočovská aktivit, která ji v první řadě vytvořila. Pokud Aktivita vrací výsledek, nastaví se před jejím ukončením metodou `setResult(int resultCode)`.

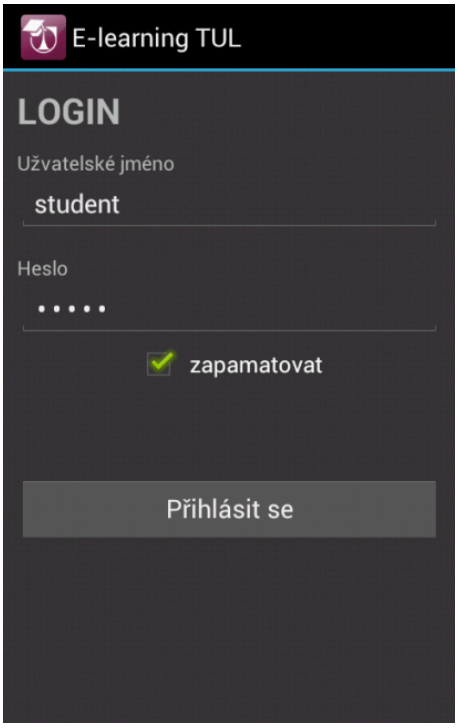
Po ukončení nově vyvolané aktivity s očekávaným výsledkem je zavolána metoda `onActivityResult(int requestCode, int resultCode`. Jejími parametry jsou požadovaná hodnota při vytváření aktivity a výsledek nastavený před ukončením aktivity. Tímto způsobem lze rozlišit jednotlivé aktivity a jejich výsledky. Dále je možné mezi aktivitami předávat primitivní datové typy pomocí objektu `Intent` a jeho metody `putExtra()`, která hodnotu uloží v jedné aktivitě, a metody `getExtra()`, která ji načte v aktivitě druhé. Aktivitu může ukončit uživatel stisknutím hw. tlačítka *Zpět*.



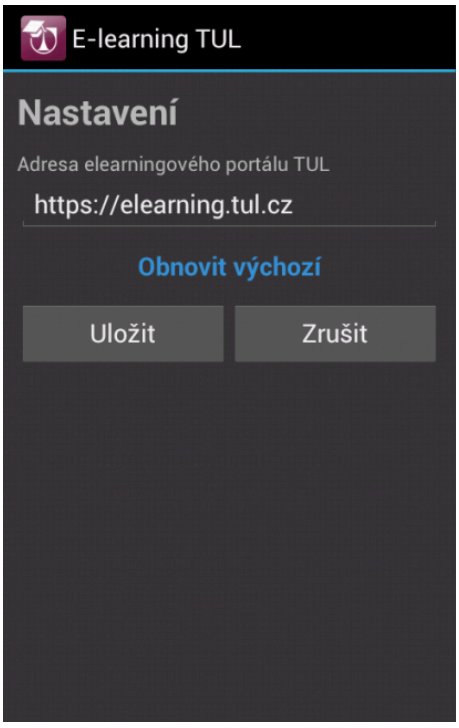
Obrázek 4.3: Schéma aktivit klientské aplikace

V klientské aplikaci bylo vytvořeno celkem osm aktivit. Jejich návaznost je znázorněna ve schématu aplikace v obrázku 4.3 a jejich funkce jsou popsány v následujícím seznamu:

- *LoginActivity* - tato aktivita je výchozí aktivitou klientské aplikace. To znamená, že při spuštění aplikace se načte jako první a ostatní aktivity vycházejí právě z této. Jedná se o přihlašovací obrazovku se vstupem pro uživatelské jméno a heslo do systému Moodle, respektive do sítě LIANE. Obsahuje také zaškrtávací pole pro uložení přihlašovacích údajů a tlačítko pro přihlášení. Pokud uživatel zaškrtne pole *zapamatovat*, při příštím spuštění aplikace bude automaticky spuštěn proces přihlašování se zapamatovanými údaji. Při stisknutí hardwarového tlačítka *Menu* se objeví možnost upravit nastavení aplikace (*SettingsActivity*), zobrazit krátké informace o aplikaci samotné a ukončit ji. Vzhled přihlašovací obrazovky je na obrázku 4.4



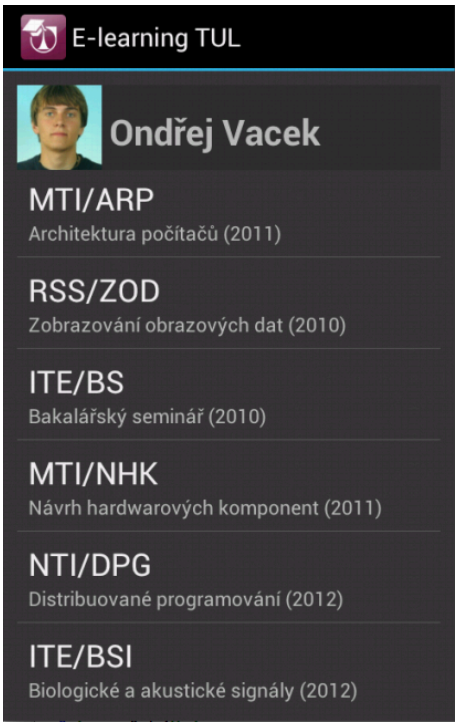
Obrázek 4.4: Přihlašovací obr.



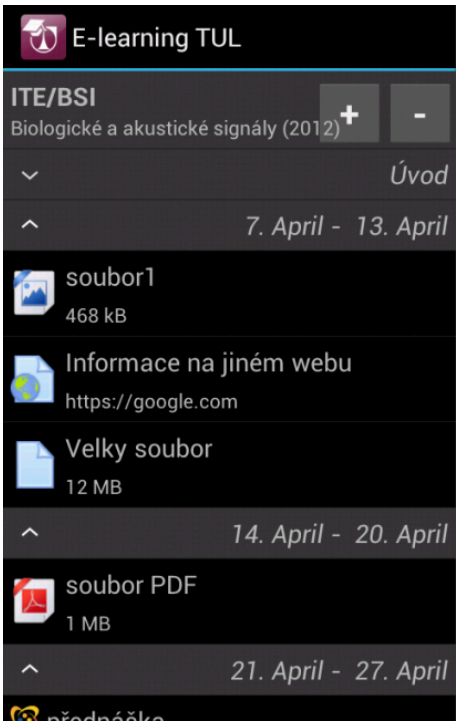
Obrázek 4.5: Nastavení

- *SettingsActivity* - tato aktivita obsahuje nastavení webové adresy e-learningového portálu TUL. Tato možnost byla ponechána z důvodu případného přesunu portálu na jinou doménu, nebo lze pouze změnou protokolu vypnout zabezpečenou komunikaci a použít nezabezpečeného protokolu HTTP. Na obrazovce je jedno vstupní pole, kde má uživatel možnost zadat adresu. Poté je zde

nápis *Obnovit výchozí*, při jehož stisknutí se zadaná adresa změní na přednastavené „*https://elearning.tul.cz*“. Tato možnost je ponechána uživateli v případě, že by omylem zadal nesprávnou adresu portálu. Dále jsou dvě tlačítka pro uložení adresy a zrušení úprav. Při uložení aplikace zkontroluje, zda adresa serveru začíná „*http://*“ nebo „*https://*“. Pokud tomu tak není, uložení neproběhne a uživatel je informován o nesprávném tvaru adresy.



Obrázek 4.6: Seznam kurzů

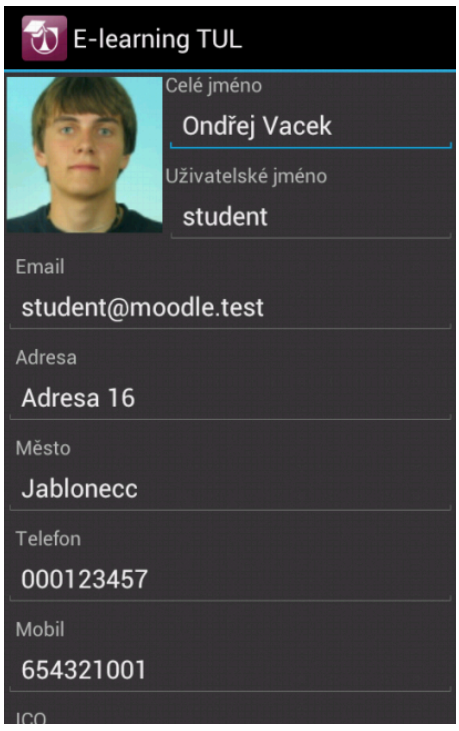


Obrázek 4.7: Obsah kurzu

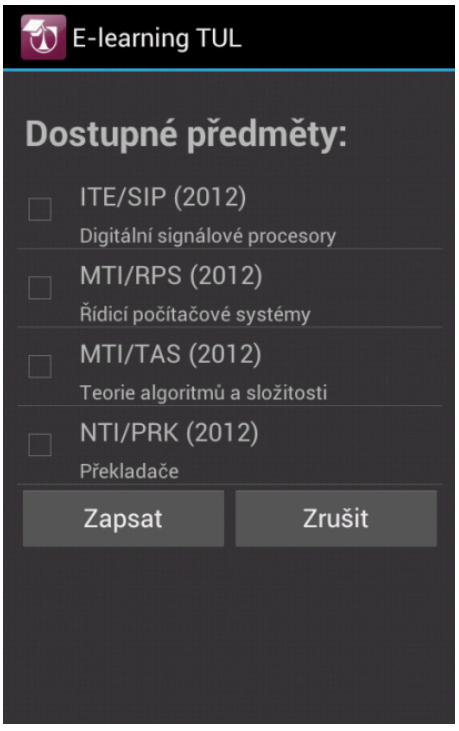
- *HomeActivity* - tato aktivita se uživateli objeví jako první po jeho přihlášení a obsahuje jeho profilovou fotografii a jeho celé jméno. Pokud nemá v systému fotografii nastavenou, je na jejím místě zástupný obrázek systému Moodle. Pod touto hlavičkou je rolovací seznam zapsaných kurzů (neboli předmětů, v rámci jednotného pojmenování se systémem Moodle se zde hovoří o kurzech). Na prvním řádku je velkými písmeny kód kurzu, na druhém se nachází celý název a rok zahájení. Kliknutím na příslušný kurz dojde ke zobrazení jeho obsahu. Při stisknutí hw. tlačítka *Menu* má uživatel možnost zobrazit aktivitu se svými profilovými údaji (*ProfileActivity*), aktivitu pro zapsání do nových kurzů (*STAGActivity*) a možnost

odhlásit se. Příklad přihlášeného uživatele je znázorněn na obrázku 4.6.

- *ContentActivity* - tato aktivita zobrazuje uživateli obsah kurzu po jeho zvolení na *HomeActivity*. V horním řádku obrazovky je opět kód a název kurzu, vpravo pak dvě tlačítka: symbol „+“pro rozbalení obsahu ve skupinách a symbol „-“, který zabalí skupiny zpět. Tato vlastnost se dá využít v případě, že v kurzu je více obsahu, uživatel může všechny skupiny zabalit a pak rozbalit pouze jednu požadovanou. Tím se zvýší přehlednost dat v obsahu. Dále je na obrazovce samotný obsah. Stejně jako v systému Moodle je tříděn do skupin, které vytvářejí vyučující a každá skupina má jméno a vlastní obsah. Kliknutím na jméno se skupina zabalí nebo rozbalí. Pod názvem skupiny se nacházejí prvky obsahu. Dle typu prvku je přiřazena ikona, vedle které se nachází název prvku a případně doplňující informace (například velikost souboru, adresa odkazu apod.). Typy prvků jsou popsány v kapitole 3.4.2. Po kliknutí na požadovaný prvek se vykoná akce jemu přidružená (například soubor se začne stahovat, otevře se odkaz v okně prohlížeče apod.). Ukázka obsahu kurzu je na obrázku 4.7.



Obrázek 4.8: Profil uživtele



Obrázek 4.9: Propojení STAG

- *ProfileActivity* - na tuto aktivitu se uživatel dostane přes menu v *HomeActivity*. Jedná se o výpis všech informací, které jsou o uživateli v systému Moodle uloženy. Je zde zobrazeno velké profilové foto uživatele, dále jeho jméno, přihlašovací jméno a e-mail. Tyto údaje jsou načteny systémem Moodle z univerzitní sítě LIANE a stejně jako v systému Moodle se nedají měnit. Dále je výčet doplňujících informací o uživateli, které má možnost vyplnit. Pod informacemi se nacházejí dvě tlačítka pro uložení informací na server a zrušení editace. Obě uživatele zavedou zpět na *HomeActivity*. Vzhled této aktivity je znázorněn na obrázku 4.8.

- *STAGActivity* - tato aktivita zobrazuje uživateli kurzy, ve kterých doposud není zapsán, ale podle spárování systému Moodle se systémem STAG je studuje, nebo studoval. Uživatel se na ni dostane z *HomeActivity*. Na obrazovce je seznam kurzů s kódem kurzu, jménem kurzu a zaškrtávacím políčkem. Při kliknutí na jednu z těchto tří věcí se předmět označí pro zapsání. Pod seznamem jsou dvě tlačítka pro zapsání vybraných kurzů a zrušení zapisování. Obě tlačítka vrátí uživatele zpět na *HomeActivity*.

- *HTMLActivity* - tato aktivita obsahuje vestavěný prohlížeč systému Android a při jejím vytváření se předá pouze URL adresa k zobrazení. Je spouštěna kliknutím na prvek obsahu kurzu typu *soubor* nebo *odkaz*. Zatímco odkaz otevře požadovanou adresu, URL adresa na stažitelný soubor zajistí jeho stažení systémem Android bez samotného otevření této aktivity.

- *FileDialog* - tato aktivita umožňuje uživateli vybrat soubor na paměťovém médiu mobilního zařízení. Uživateli se zobrazí po kliknutí na prvek *offile úkol* v obsahu kurzu. Uživatel díky této aktivitě vybere soubor a po stisknutí tlačítka „*Odeslat*“zahájí odesílání vybraného souboru na server, který jej přiřadí ke správnému úkolu. Systém Android překvapivě neobsahuje žádnou systémovou komponentu pro vybrání souboru a z toho důvodu bylo nutné přidat tuto aktivitu.

5. Závěr

Hlavním cílem této práce bylo navrhnout a vytvořit aplikaci pro mobilní zařízení na platformě Android. Díky vzrůstající oblibě této platformy napříč celou šálou nejrůznějších zařízení obzvláště u studentů narůstá její počet uživatelů každým rokem. Mobilní zařízení se stávají nedílnou součástí každodenního života, a proto je snaha přizpůsobit činnosti těmto zařízením logickým krokem do budoucna.

Práce se zaměřila na e-learningový portál TUL, který studenti TUL hojně využívají pro nalezení studijních materiálů, informací o předmětech a pro zobrazení nahraných přednášek. E-learningový portál je založen na systému Moodle, který nabízí jedinou možnost přístupu přes webové stránky a ty nejsou nikterak optimalizované pro mobilní zařízení. Aplikace tedy poslouží studentům jako alternativní přístup na tento portál pro přehledné a jednoduché zobrazení studijních materiálů.

Stejným problémem se již v minulosti zabýval na TUL projekt ESF (Evropský sociální fond) v rámci OpVK (Operační program Vzdělávání pro konkurenceschopnost) s názvem *„Inteligentní multimediální e-learningový portál, CZ.1.07/2.2.00/07.0008 – Advanced Learning Space"*. Dle dostupných informací však projekt nebyl úspěšně ukončen.

Pro úspěšné nasazení klientské aplikace je zapotřebí implementovat nové úpravy systému Moodle a nainstalovat novou verzi softwaru MediaSite. Správci těchto systémů již byli informováni a jakmile budou potřebné úpravy dokončeny, aplikaci můžou začít používat všichni studenti TUL se zařízením na platformě Android.

5.1 Dosažené výsledky

V rámci této diplomové práce vznikla aplikace s názvem *E-learning TUL* pro platformu Android ve verzi 4 a vyšší. Aplikace je určena pro studenty TUL, kteří se s její pomocí připojí na E-learningový portál TUL na svém mobilním zařízení a mají možnost procházet a stahovat studijní materiály, zobrazovat informace o předmětech, zobrazovat záznamy z přednášek prostřednictvím serveru MediaSite, nahrávat vypracované úlohy na server, editovat svůj vlastní profil a zapisovat předměty, které dle systému STAG studují.

Při vývoji byl kladen důraz na dvě věci. První bylo dostatečně zabezpečit veškerou komunikaci mezi aplikací a serverem. Z toho důvodu byl zvolen protokol HTTPS a veškerá komunikace byla přizpůsobena primárně pro tento protokol. Nicméně uživateli byla ponechána možnost změnit protokol na nezabezpečený HTTP pro případ komplikací, ke kterým příležitostně dochází při použití určitých osobních certifikátů.

Druhá věc, na kterou se se zaměřil vývoj aplikace byla *responzivní layout*. Jedná se o takové rozvržení prvků na obrazovce, které se přizpůsobí její velikosti a poměru stran a bez ohledu na vlastnosti zařízení jsou všechny prvky aplikace viditelné a snadno dostupné. Toto je hlavním důvodem, proč by studenti měli využívat tuto aplikaci. Webové rozhraní e-learningového portálu není nijak optimalizováno pro mobilní zařízení a obzvláště na malých displejích je obtížné se na portále pohybovat.

Vzhled aplikace byl zvolen v souladu se základními prvky systému Android. Jedinými grafickými prvky jsou ikony souborů a dalších prvků obsahu předmětů. Při vývoji šlo především o to, jak zobrazit informace v co nejucelenější a nejpřehlednější podobě a tím ulehčit uživateli přístup k potřebným informacím.

Pro komunikaci mezi aplikací a systémem Moodle, který poskytuje e-learningový portál, byl zvolen protokol webové služby SOAP. Systém Moodle obsahuje základní implementaci tohoto protokolu. Pro potřeby aplikace bylo provedeno několik úprav ve zdrojovém kódu systému Moodle (jak v implementaci SOAP, tak mimo ni) a tyto úpravy byly reportovány na vývojářský server jako vylepšení systému. Proto je možné, že nová verze systému Moodle bude obsahovat tato vylepšení již v základu.

V klientské aplikaci bylo implementováno kompletní rozhraní webové služby SOAP pro použití se systémem Moodle. Díky neúplné automatické dokumentaci (WSDL) na straně serveru bylo nutné SOAP implementovat ručně. Tato překážka však byla úspěšně vyřešena a aplikace obsahuje kompletní komponentu pro komunikaci se systémem Moodle.

Je doporučeno používat aplikaci pouze s univerzitním e-learningovým portálem, jelikož jiná instance systému Moodle neobsahuje zmiňované úpravy ani nastavení a s největší pravděpodobností by se aplikace k jinému systému ani nepřipojila.

Jednou ze zmiňovaných činností aplikace v zadání této diplomové práce bylo *práce s testy*, protože systém Moodle obsahuje komponentu pro vytváření a skládání testů. V průběhu vývoje aplikace byla tato součást několikrát konzultována s vedoucím této diplomové práce a po pečlivém zvážení bylo rozhodnuto tuto možnost neimplementovat hned ze dvou důvodů. Prvním je, že komponenta systému Moodle pro práci s testy je nesmírně rozsáhlá a vedle desítky možných testových úloh obsahuje i vlastní pseudo-programovací jazyk pro definici chování testu. Implementace této komponenty v plném rozsahu by zabrala nepoměrně dlouhou dobu nemluvě o faktu, že u systému Moodle se na jejím vývoji podílí celý tým vývojářů. Druhým důvodem je fakt, že na TUL vyučující nevyužívají možnost tvorby testů příliš často a pokud k tomu dojde, přejí si, aby je studenti vyplnili ve škole pod jejich dozorem. Z těchto důvodů by implementace testů neměla pádný důvod.

5.2 Rozšíření

V prvé řadě je nutné zmínit, že systém Moodle disponuje desítky modulů pro nejrůznější výukové metody a klientská aplikace nemá sloužit jako plnohodnotná náhrada jeho webového rozhraní. Jedná se pouze o doplněk systému Moodle, který umožňuje studentovi TUL snadný přístup k základním a nejpoužívanějším činnostem na systému. Pokud některá z požadovaných činností chybí, student má vždy možnost zobrazit webové rozhraní a provést ji tam.

Aplikace zatím nebyla otestována v praxi v reálných situacích, a proto je možné, že studenti budou některé důležité činnosti postrádat. Ty mohou být zajisté implementovány později v rámci rozšiřování aplikace. Zdrojový kód aplikace je přehledně okomentován a implementace webové služby SOAP je navržena tak, že umožňuje jednoduché napojení nových funkcí.

V neposlední řadě by bylo vhodné sjednotit e-learningové portály všech kateder TUL do jednoho, jelikož tomu stále tak není, ale to již nebylo součástí této práce.

5.3 Alternativy

Během vývoje aplikace bylo nalezeno několik alternativních způsobů pro práci se systémem Moodle na platformě Android. První alternativa se objevila v průběhu března 2013, kdy klientská aplikace byla již v pokročilém stádiu vývoje. Jedná se o univerzální aplikaci pro systém Moodle přímo od tvůrců systému. Aplikace byla doposud vydána pouze ve verzi *beta* a při jejím testování došlo k častým pádům aplikace a funkčnost aplikace byla omezena jen na strohý výpis objektů v kurzech bez dalších možností.

Posléze bylo nalezeno několik podobných aplikací, které byly vyvinuty na prestižních univerzitách v zahraničí a slouží ke stejným účelům. Každá z těchto aplikací je napojena pouze na specifickou verzi jejich univerzitního portálu a stejně jako v klientské aplikaci pro TUL i tyto aplikace vyžadovaly specifické úpravy v systému Moodle dle potřeb univerzity.

Literatura

[1] Charitha Kankanamge. *Web Services Testing with soapUI*. Birmingham, Packt Publishing Ltd., 2012. ISBN 978-1-84951-566-5

[2] Kocan, Marek. *Charakteristiky Moodle* [online]. cit. 04/2013. URL: <http://www.moodlecon.cz/jine/moodle>.

[3] Murphy Mark L. *Android Programming Tutorials*. Spojené státy americké, CommonsWare, LLC., 2011. ISBN 978-0-9816780-4-7

[4] Netcraft Tem. *Web Server Surveys* [online]. 12/2012. URL: <http://news.netcraft.com/archives/web_server_survey.html>.

[5] Open Source Initiative *The MIT License (MIT)* [online]. 2000. URL: <http://opensource.org/licenses/mit-license.php>.

[6] Peterka, Jiří. *Kvalifikovaný certifikát na dvě věci* [online]. 04/2002. URL: <http://www.lupa.cz/clanky/kvalifikovany-certifikat-na-dve-veci/>.

[7] Redakce časopisu BusinessIT *Chytrých telefonů se loni poprvé prodalo více než počítačů* [online]. 02/2012 URL: <http://www.businessit.cz/cz/chytre-telefony-tablety-smartphone-pc-prodeje-trh-android-ios.php>.

[8] Sonic Foundry, Inc. *Support for Mediasite playback on mobile devices* [online]. 28.1.2013. URL: <https://support.sonicfoundry.com/Knowledge/Article/000003025>.

[9] World Wide Web Consortium. *Simple Object Access Protocol (SOAP) 1.1* [online]. 8.5.2000. URL: <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>.