VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2017 Martin Filípek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Martin Filípek**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: HS Interactive s.r.o.
2. Struktura závěrečné zprávy:
a. Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta
b. Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti
c. Zvolený postup řešení zadaných úkolů
d. Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe
e. Znalosti či dovednosti scházející studentovi v průběhu odborné praxe
f. Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vedl odbornou praxi studenta

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Zdeňka Chmelíková, Ph.D.**

Konzultant bakalářské práce: Alexander Nikolskiy

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 26. dubna 2017 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 26. dubna 2017

HS Interactive s.r.o.
Výroba hardware a software
IČ: 27773710
28. října 2663 / 150, PSČ 702 00
Ostrava - Moravská Ostrava, Czech Republic

Rád bych poděkoval firmě HS Interactive s.r.o. za umožnění absolvování odborné praxe. Převážně jejímu řediteli a mému konzultantovi Ing. Alexandru Nikolskému, který dohlížel na její průběh, předával mi cenné zkušenosti a pomáhal s řešením problémů.

Abstrakt

Cílem této bakalářské práce je popsat absolvování odborné praxe ve firmě HS Interactive s.r.o. Praxe byla zaměřena na vývoj mobilní aplikace pro operační systém Android. Aplikace je mobilním klientem pro sociální síť MatchToMe. V úvodu popisuji důvody, které vedly k výběru odborné praxe. Dále se věnuji úkolům, které mi byly zadány s jejich implementací a postupem řešení problémů, které se objevily při vývoji. Závěr práce je věnován zhodnocení získaných zkušeností a dosažených výsledků.

Klíčová slova: odborná praxe, Java, operační systém Android, mobilní aplikace, MatchToMe, HS Interactive s.r.o.

Abstract

Purpose of this bachelor thesis is to describe a professional practice in company HS Interactive s.r.o. Practice was focused on the development of mobile application for the operating system Android. The application is a mobile client for social network MatchToMe. In the introduction I describe reasons that led to the selection of professional practice. Then I describe tasks that I have been awarded with their implementations and process of solution issues that have emerged during development. The conclusion of thesis is dedicated to the evaluation of the experience gained and the results achieved.

Key Words: professional practice, Java, operating system Android, mobile application, MatchToMe, HS Interactive s.r.o.

Obsah

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
CSS	–	Cascading Style Sheets
HTML	–	Hyper Text Markup Language
HTTPS	–	Hypertext Transfer Protocol Secure
JDK	–	Java Development Kit
JSON	–	JavaScript Object Notation
JVM	–	Java Virtual Machine
MB	–	megabyte
LIFO	–	Last In First Out
SDK	–	Software development Kit
XML	–	eXtensible Markup Language

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Při volbě vypracování závěrečné práce jsem měl na výběr z témat zadaných školou anebo absolvování odborné praxe.

Zvolil jsem si odbornou praxi, abych si ověřil své znalosti získané při studiu a zároveň nabral zkušenosti ve firmě zaměřené na vývoj v oboru informačních technologií. V budoucnu se chci věnovat vývoji mobilních aplikací pro operační systém Android a proto jsem oslovil firmy, které nabízejí odbornou praxi s tímto zaměřením. Po absolvování vstupních pohovorů jsme se rozhodl pro firmu HS Interactive s.r.o.

V úvodní části bakalářské práce Vás seznámím se zaměření této firmy. Hlavně se věnuji popisu nástrojů, které firma využívá. Uvádím také náplň práce na mé pozici.

Hlavní část obsahuje zadání mé práce. Obecně popíši historii operačního systému Android a nástroje potřebné k vývoji. Pak uvedu úkoly, které mi byly zadány. V části zaměřené na řešení zadaných úkolů popíši převážně komponenty, které jsem pro daný úkol použil a problémy, na které jsem narazil.

Konec bakalářské práce jsem věnoval popisu znalostí a dovedností získaných při studiu a jejich využití při odborné praxi. Taktéž zde uvádím to, co mi při absolvování odborné praxe scházelo. V závěru hodnotím dosažené výsledky a celkově celou svou činnost ve firmě.

2 O firmě HS Interactive s.r.o.

Firma HS Interactive s.r.o. byla založena v roce 2006 a od samého začátku se věnuje vývoji software a hardware pro převážně zahraniční klienty.

Hlavním zaměřením je vývoj moderních webových aplikací za pomocí frameworku Ruby on Rails. Její ředitel Ing. Alexander Nikolskiy má dlouholeté zkušenosti s vývojem v programovacím jazyce Java.

Interními softwarovými projekty společnosti jsou systém pro správu sportovních a relaxačních center ReseaManager, systém pro online rezervace ReseaOnline a v neposlední řadě i sportovní sociální síť a nástroj pro pořádání sportovních turnajů a jiných událostí MatchToMe.

Mezi hardwarové projekty patří jednotky pro řízení osvětlení ve sportovištích a pro ovládání solárií.

2.1 MatchToMe

Sportovní sociální síť umožňující vyzývání na zápasy, pořádání turnajů či sledování svých posunů v globálním žebříčku.

Hlavním cílem je inspirovat lidi k aktivnímu způsobu života. Snaha je o vytvoření nástroje, který by umožnil sportovním centrům získat nové návštěvníky a motivovat stávající, aby se vraceli. [22]

Mezi hlavní přednosti patří:

- Vytváření pozvánek a vyzývání uživatelů na zápas
- Vytváření turnajů
- Prohlížení veřejných profilů sport center
- Prohlížení klubových žebříčků
- Nalezení informací o daném sportu
- Prohlížení žebříčků v daném sportu
- Nalezení nových přátel sdílejících stejné nadšený pro sport

2.2 Pracovní zařazení

Do firmy jsem byl přijat na pozici Junior vývojář mobilních aplikací - Android.

Mým hlavním úkolem byla analýza, návrh a implementace nativní mobilní aplikace pro systém Android.

Úzce jsem spolupracoval s designérkou firmy, která rozhodovala o vzhledu aplikace a s Ing. Alexadrem Nikoslkiym, který se staral o tvorbu webových API pro komunikaci s aplikací.

2.3 Nástroje, které firma využívá

2.3.1 Bitbucket

Webová hostingová služba pro tvorbu zdrojových kódů a vývoj softwaru podporující využití verzovacího nástroje Mercurial nebo Git vlastněná firmou Atlassian. V rámci firmy využíváme verzovací systém Git.

Git napsal Linus Torvalds pro vývoj jádra Linuxu. Jeho využití bylo časem rozšiřováno až do dnešní podoby, kdy je převážně používán pro správu verzí softwaru.

Jedná se o silný nástroj pro práci v týmu. Hlavní funkcí je tvorba stromové struktury. Jednotliví členové týmu mohou zakládat nové větve vycházející z hlavní větve. Díky tomu lze velké projekty rozdělit na menší části a pracovat na nich paralelně. Po dokončení práce na své větvi jednoduše programátor požádá o začlenění do hlavní větve. Díky tomu může vedoucí projektu posoudit správnost kódu a rozhodnout o začlenění nebo přepracování. Pokud nastanou nějaké problémy, tzv. konflikty, většinou z důvodu, že dva programátoři pracovali na stejné větvi a upravili stejné části, lze jednoduše porovnat větve mezi sebou a rozhodnout o lepší implementaci.

Mezi hlavní přednosti patří možnost obnovení předchozí verze. To se hodí například při objevení chyby po sloučení větví, kdy může být projekt navrácen do funkční podoby a chybová větev být přepracována.

2.3.2 JIRA

Softwarový nástroj od společnosti Atlassian sloužící k řízení vývoje projektů zaměřený hlavně na oblast informačních technologií.

Systém umožňuje jednoduchou tvorbu i správu úkolů, týmové plánování, sledování chyb a problémů. Díky tomu zle jednoduše sledovat v jaké části vývoje se úkol nachází, do kdy má být dokončen, kolik zabral času, dotazovat se na upřesnění anebo vkládání objevených chyb.

Já jsem systém využíval pro sledování mých úkolů, zakládání úkolů na vývoj webových API a nahlašování chyb, které jsem objevil při práci se sociální sítí MatchToMe.

Dále je zde možnost propojení s webovou službou Bitbucket. Díky tomuto spojení, se lze podívat na konkrétní větev spadající pod daný úkol. [4]

3 Platforma Android

Mobilní operační systém Android je v dnešní době nejrozšířenějším systémem. Jeho největší využití je v mobilních telefonech, tabletech a chytrých zařízeních, jako jsou například chytré hodinky.

Jeho vývoj započala firma Android Inc. v říjnu roku 2003. V srpnu roku 2005 ji odkoupil Google Inc. a udělal z ní svoji dceřinou společnost. Do čela vývojového týmu se dostal spoluzakladatel firmy Android Inc. Andy Rubin. Tým pracoval na platformě založené na Linuxovém jádře, díky které získal Google několik patentů v oblasti mobilních technologií. Největším milníkem v historii platformy bylo datum 5. listopadu 2007, kdy došlo k vytvoření uskupení Open Handset Alliance. Ve stejný den došlo k představení otevřené mobilní platformy Android široké veřejnosti. Prvním mobilním telefonem s touto platformou se stal HTC Dream, který byl uveden v říjnu roku 2008 a došlo k uvolnění prvního SDK 1.0. Nejnovější verzí je 7.0 nazývaná Nougat vydaná 22. srpna 2016, které odpovídá SDK ve verzi 24.0. [24]

3.1 Nástroje potřebné pro vývoj

3.1.1 Android studio

Toto prostředí je základním a doporučeným pro vývoj Android aplikací. Vzniklo ze spolupráce firem Google a JetBrains. Tento nástroj je multiplatformní, zcela zdarma a je možné ho získat ze stránek společnosti Google [6]. Jeho instalace je velmi jednoduchá a je k ní vytvořen návod formou textu a videa.

Umožňuje jednoduchou editaci kódu, ladění, flexibilní sestavení aplikace, její následné otestování a nástroje ke sledování náročnosti aplikace na procesor, paměť a síť. Další výhodou je implementovaný verzovací systém Git a možnost instalace doplňků, které ulehčují tvorbu kódu, vložení obrázků s podporou pro různé rozlišení nebo podávají statistiky o množství kódu.

3.1.2 SDK tools

Soubor nástrojů pro vývoj mobilních aplikací na platformě Android. Obsahuje potřebné knihovny, ladící nástroj, obrazy pro emulátor virtuálních zařízení, vzorové zdrojové kódy a dokumentaci k jednotlivým API. [24]

Součástí Android studia je SDK Manager, který Vám umožní jednoduše stáhnout dostupné SDK s výběrem částí, které chcete nainstalovat. Jednotlivé balíčky jsou velikosti několika stovek MB a z toho důvodu není vhodné instalovat všechny verze. Pokud jste se rozhodli, že Vaše aplikace bude podporovat nejnižší verzi Android 4.1 je vhodné, z důvodu testování, stáhnout i verze nadcházející a vynechat verze předchozí.

3.1.3 Java JDK

Mobilní aplikace pro operační systém Android jsou vyvíjeny v programovacím jazyce Java. Za jeho vývojem stojí firma Sun Microsystems a byl představen 23. května 1995. Po akvizici 27. ledna 2010 se vlastníkem oficiální implementace stala společnost Oracle. Tento jazyk je objektově orientovaný, distribuovaný, interpretovaný, přenositelný, nezávislý na architektuře, víceúlohový a dynamický.

Pro vývoj mobilních aplikací je potřeba mít nainstalovaný balíček JDK. Ten obsahuje soubor základních nástrojů pro vývoj pro platformu Java. Je dostupný zdarma na stránkách společnosti Oracle. [24]

3.1.4 Možnosti testování

Testování mobilních aplikací je odlišné od deskopových aplikací. Při ladění deskopových aplikací se můžeme většinou spolehnout na správnou funkcionalitu, pokud nám fungují na vývojovém počítači. Vývoj mobilních aplikací je ztížen různými verzemi Androidu či rozdílnou vybaveností cílových zařízení.

Ideální je ověření správného fungování aplikace na reálném přístroji. K tomu nabízí operační systém Android v zařízení možnost povolení ladění přes USB. Po jejím povolení se zařízení propojí USB kabelem s vývojovým počítačem. Stane se tak dostupným pro vývojové prostředí, které jej při spuštění ladění nabídne pro cílovou instalaci.

Další možností je vytvoření virtuálního stoje za pomoci emulátoru. Ve většině případů je emulátor pomalý a nahrání aplikace do něj je zdlouhavé. Jeho výhodou je možnost vytvoření více virtuálních zařízení s různou verzí Androidu, velikostí či rozlišením displeje. Díky tomu lze do jisté míry uzpůsobit vzhled aplikace, odzkoušet nastavení různých jazyků či správné chování aplikace na různých verzích.

Ani jedna z možností není zcela vhodná, protože vlastnit všechny dostupné zařízení na trhu by bylo nákladné. Na chování aplikace v emulátoru se nelze zcela spolehnout, jelikož je ovlivněn konfigurací vývojového počítače i ovládáním za pomoci klávesnice a myši. Proto je snaha získávat zpětnou vazbu od uživatelů.

4 Seznam zadaných úkolů

4.1 Vytvoření základní aplikace s odpovídajícím vzhledem

Vytvořit základní aplikaci, která se stane hlavní větví verzovacího systému Git. Aplikaci bude možné používat na zařízeních s minimální verzí Androidu 4.0.3 (SDK 15) a ke svému běhu vyžaduje přístup k internetovému připojení.

Součástí úkolu je vytvoření hlavního vzhledu aplikace. Každá obrazovka bude obsahovat v horní části hlavní lištu. Navigace v aplikaci bude pomocí menu, které obsahuje seznam těchto položek: Domů, Pozvánky, Události, Sportovní centra, Sporty a odhlášení z aplikace.

4.2 Stažení a zpracování dat ze serveru

Navrhnout vhodnou třídu pro stažení dat ze serveru a obdržená data zpracovat do odpovídajícího objektového modelu.

4.3 Stažení a uložení obrázků z internetové adresy

Prostudovat si možnosti stahování obrázků z internetové adresy a jejich uložení do mezipaměti telefonu. Na základě získaných teoretických znalostí navrhnout vhodnou třídu, která se stará o obsluhu stažení, uložení a následné vložení do komponenty.

4.4 Uživatelské prostředí

Zaručit dostupnost menu z jakékoliv části aplikace. Implementovat uživatelské prostředí pro seznam příspěvků, sportovních center, sportů, událostí a pozvánek. Dále detail sportovního centra, sportu, události a uživatele. U pozvánek umožnit uživateli navrhnutí nového termínu. Po jeho vytvoření zobrazit výsledek.

4.4.1 Nekonečné rolování

Pokud se uživatel posune v seznamu na poslední položku vytvořit dotaz pro další data. Získaná data uložit do seznamu a zobrazit.

4.4.2 Kategorie

Seznamy pozvánek a událostí budou rozděleny do kategorií. Mezi nimi bude uživatel pohodlně přecházet posunem prstu do stran.

4.4.3 Detaily

Společným prvkem detailů je banner, pokud je dostupný, který po posunutí se prstem směrem nahoru zmizí. Zajistit jeho viditelnost pouze v detailu. Dále je důležité zahrnout související položky. Položky pro jednotlivé detaily jsou následující:

- Sportovní centrum – Podrobnosti, Zeď, Události, Fotky, Kdo nás sleduje
- Sport – Žebříček, Turnaje, Pravidla
- Událost – Zeď, Podrobnosti, Skupiny, Play–off, Fotky
- Uživatel – Podrobnosti, Zeď, Přátelé, Fotky, Události

4.5 Implementovat přihlášení uživatele

Vytvořit obrazovku pro přihlášení uživatele s odlišným vzhledem a uložit cookie sloužící k jeho identifikaci. Jestliže se uživatel rozhodne zůstat přihlášen nezobrazovat ji při novém startu aplikace.

5 Zvolený postup řešení zadaných úkolů

5.1 Vytvoření základní aplikace s odpovídajícím vzhledem

5.1.1 Základní aplikace

Minimální verzi jsem si zvolil v Android Studio již při zakládání nového projektu. Volba ovlivní, pro která zařízení bude aplikace v obchodě Google Play dostupná. Protože jsem nevybral nejnovější verzi, musel jsem u některých komponent ošetřit jejich správnou funkčnost na nižších verzích. [21]

Během zakládání nového projektu jsem měl na výběr z několika šablon. Zvolil jsem vytvoření prázdné aktivity spolu s XML souborem rozvržení uživatelského rozhraní. Nazval jsem ji MainActivity a vybral možnost pro zpětnou kompatibilitu. Po potvrzení se vytvořila základní struktura projektu. Kvůli volbě zpětné kompatibility byla automaticky přidána knihovna appcompat–v7 spadající pod Support Library do závislostí sestavovacího systému Gradle a MainActivity vytvořena jako podtyp AppCompatActivity.

Z bezpečnostních důvodů používá operační systém Android pro běh aplikací Sandbox. Ten zaručí, že aplikace nebude mít přístup ke všem datům či prostředkům zařízení. Programátor si může zažádat o povolení. V závilosti na tom, jak moc jsou citlivá, systém rozhodne, jestli je schválí sám anebo o jejich udělení rozhodne uživatel. Jelikož moje aplikace potřebuje ke svému běhu přístup k internetu, deklaroval jsem v souboru AndroidManifest.xml potřebná povolení (Výpis 1). [20]

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

Výpis 1: Povolení přístupu k internetovému připojení

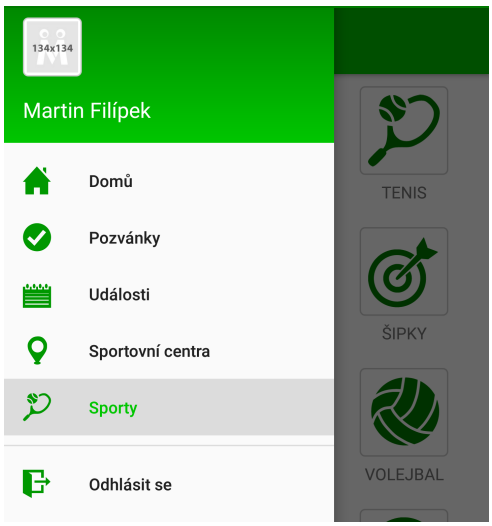
5.1.2 Hlavní vzhled aplikace

Vzhled aplikace má přímý dopad na dojem, který na uživatele udělá. Pro programátory a designéry existuje u každé platformy příručka, která jim předá potřebné informace pro vývoj. V případě platformy Android je posledním trendem návrhový styl Material Design. Hlavním cílem tohoto stylu je sjednocení vzhledu napříč rozdílnými platformami a zařízeními. [13]

Abych mohl použít komponenty a vzory které Material Design nabízí, musel jsem ve vývojovém prostředí upravit strukturu projektu. Konkrétně jsem do závislostí sestavovacího systému Gradle přidal knihovnu design. [14]

5.1.2.1 Zvolená komponenta hlavního menu

Díky zvolení tohoto stylu jsem mohl pro hlavní menu využít Navigation Drawer (Obrázek 1). Jde o postranní panel, který je většinu času pro uživatele neviditelný a nezabírá tak část obrazovky. Přesto k jeho zobrazení stačí jednoduché gesto či kliknutí na přidružené tlačítko.



Obrázek 1: Hlavní menu aplikace

K implementaci Navigation Drawer jsem přidal do závislostí knihovnu support–v4. Následně jsem upravil XML soubor definující rozvržení MainActivity. Hlavní kontejner jsem změnil na DrawerLayout, který umožní zobrazení menu za pomoci gesta. Dovnitř DrawerLayout jsem vložil FrameLayout pro vyčlenění místa na obrazovce k zobrazení obsahu a NavigationView.

NavigationView má dva atributy, které jsou klíčové pro splnění požadavků na vzhled menu. Prvním je headerLayout, sloužící k přiřazení souboru hlavičky. Vytvořil jsem tedy ve složce res/layout nový XML soubor jehož hlavním kontejnerem je RelativeLayout. Do RelativeLayout jsem vložil widget ImageView pro obrázek uživatele a TextView pro jeho jméno. Druhým atributem je menu sloužící k přiřazení souboru položek menu. Pro menu jsem přidal do základního adresáře res novou složku menu v níž jsem vytvořil menu XML soubor. V tomto souboru jsem definoval dvě skupiny. Do první jsem dal hlavní položky menu a do druhé možnost odhlášení z aplikace. [24, 5]

5.1.2.2 Zvolená komponenta hlavní lišty

Android aplikace standardně používají pro hlavní lištu ActionBar. Já se rozhodl ActionBar zaměnit za Toolbar. Vybral jsem si Toolbar hlavně z důvodu jeho větší flexibility a možnosti stejného chování. Umožňuje například definici jiného vzhledu než má aplikace, přidání tlačítka pro zobrazení menu, nadpisu a přidání menu akcí.

Abych zabránil aplikaci použít ActionBar jako hlavní lištu změnil jsem její vzhled v souboru res/values/styles na jeden z kategorie NoActionBar. Poté jsem znovu upravil soubor rozvržení

MainActivity. Do něj jsem přidal Toolbar nad FrameLayout a tyto dvě komponenty zabalil do kontejneru RelativeLayout. Díky tomu jsem mohl hlavní lištu ukotvit k hornímu okraji obrazovky a obsah umístit pod lištu. [19, 24]

5.1.3 Provázání s aktivitou

Toolbar jako hlavní lištu jsem nastavil pomocí metody setSupportActionBar() aktivity.

Vykreslení tlačítka menu v hlavní liště jsem docílil pomocí nativní třídy ActionBarDrawerToggle, které jsem při inicializaci předal instanci Toolbar a DrawerLayout. Aby došlo k otevření menu po kliknutí na tlačíko musel jsem instaci ActionBarDrawerToggle předat metodě addDrawerListener() widgetu DrawerLayout. [5, 19]

Jestliže je Navigation Drawer otevřen a uživatel klikne na tlačítko zpět na svém zařízení, musí dojít k jeho uzavření. K odchycení kliku dojde pomocí metody onBackPress() aktivity, tu jsem přepsal a přidal podmínku pro uzavření menu.

K detekci kliknutí na položku menu slouží rozhraní OnFragmentInteractionListener. OnFragmentInteractionListener jsem implementoval v záhlaví definice MainActivity a upravil jeho metodu onNavigationItemSelected(). V metodě jsem zjistil identifikační číslo zvolené položky na jehož základě jsem vložil odpovídající fragment do FrameLayout (Výpis 2) či odhlásil uživatele. Po provedení požadované akce jsem menu uzavřel.

```
FragmentManager fragmentManager = getSupportFragmentManager();
fragmentManager.beginTransaction()
               .add(R.id.mainContentFrame, new HomeFragment())
               .commit();
```

Výpis 2: Přidání nového fragmentu do FrameLayout

5.2 Stažení a zpracování dat ze serveru

Aplikace je navržena jako mobilní klient. Data jsou uložena v databázi na serveru a ke komunikaci používám webové API. Mým prvním krokem bylo vytvoření třídy DataLoader, pomocí které provedu stažení dat ze serveru, jejich uložení a navrácení výsledku.

5.2.1 Uložení do mezipaměti

V konstruktoru třídy DataLoader jsem vypočítal 1/32 z maximálního množství paměti, kterou JVM používá. Tuto hodnotu jsem využil při inicializaci nativní třídy LruCache díky čemuž došlo k vyčlenění mezipaměti pro data. LruCache jsem vybral, protože je vhodná pro ukládání silných odkazů s omezením jejich počtu. Odkazy jsou uloženy formou klíč – data a řadí se do fronty, kde nejnovější je zařazen na vrchol. Pokud ve frontě není místo pro novou hodnotu dojde k automatickému odstranění nejstarší.

Kvůli práci s mezipaměti i dostupnosti uložených dat jsem třídu implementoval jako Singleton. To zajistí, že v rámci aplikace existuje pouze jedna instance a je globálně dostupná. K uložení do mezipaměti jsem přistoupil z důvodu omezení datové náročnosti aplikace.

5.2.2 Komunikace se serverem

Následně jsem ve třídě vytvořil metodu getData() pro získání dat z adresy, kterou si předám jako argument. V metodě jsem ověřil, zda-li se data již nenachází v mezipaměti. Pokud ano navrátil jsem je. Při neúspěchu jsem vytvořil HTTPS GET požadavek. Jestliže proběhlo navázání spojení se serverem v pořádku, zpracoval jsem doručená data do řetězce a uložil jej do mezipaměti. Při ukládání jsem použil předanou adresu jako klíč. Nakonec jsem řetězec navrátil.

5.2.3 Převedení dat do JSON objektu a jejich zpracování

Pro tento krok jsem vytvořil generickou třídu Loader. Kdybych nepoužil generičnost, musela by pro každý model existovat specifická třída, což by vedlo k nadbytečnému kódu.

Součástí třídy jsou dvě metody. Jednou navrátím jediný objektový model, druhou jejich pole. V každé metodě jsem nejdříve zavolal metodu getData() třídy DataLoader. Navrácený řetězec z této metody je strukturovaný v JSON formátu. Abych si ulehčil jeho další zpracování vytvořil jsem z něj jeden či pole JSON objektů.

Ke zpracování JSON objektu jsem implementoval abstraktní generickou třídu Parser, která obsahuje abstraktní metodu parse(). V modelech jsem provedl inicializaci této třídy a implementoval metodu parse(). Pomocí metody jsem vytvořil novou instanci modelu a naplnil jeho proměnné daty z JSON objektu. Instanci třídy jsem předal jako argument metod Loader a po vytvoření JSON objektu jsem zavolal metodu parse().

Toto řešení se ukázalo jako nevyhovující, jelikož při změně webového API jsem musel upravit implementaci metody parse(). Později je naplánováno nahrazení jinou třídou ve které využiji reflexe jazyka Java.

5.2.4 Práce s vlákny

Síťové úlohy se v Android aplikacích musí provést v novém vlákně. V rámci aplikace k tomuto účelu používám nativní třídu AsyncTask. Třída AsyncTask umožňuje běh krátkých operací na pozadí bez blokace hlavního vlákna. O alokaci i odstranění vlákna se nemusí starat programátor, protože tento úkol připadá systému Android.

Výhodou AsyncTask je jednoduchá implementace. Vytvořil jsem podtyp a nahradil metodu doInBackground(). V tomto případě jsem v ní zavolal metodu objektového modelu. K získání výsledku jsem přepsal metodu onPostExecute(), ve které jsem vložil výsledek do adaptéru. [7]

5.3 Stažení a uložení obrázků z internetové adresy

Řešení tohoto úkolu je podobné kapitole 5.2.

Vytvořil jsem novou Singleton třídu ImageDownloader. V jejím konstruktoru jsem maximální dostupnou paměť převedl do kilobajtů a vydělil 8. Tuto hodnotu jsem použil při inicializaci LruCache. [8]

Pro stažení, uložení a navrácení výsledku z předané adresy jsem implementoval metodu getImage(). Získaná data jsem dekódoval pomocí metody decodeStream() třídy BitmapFactory do Bitmap objektu, jenž je vhodný pro uchování obrázků v bitové podobě.

5.3.1 Práce s vlákny

Vytvořil jsem podtyp ImageDownaloderAsyncTask třídy AsyncTask. V metodě doInBackground() jsem zavolal metodu getImage() třídy ImageDownloader a v onPostExecute() nastavil výsledek do instance widgetu ImageView, kterou jsem si předal v konstruktoru. 5.2.4

Při řešení tohoto úkolu jsem narazil na omezení třídy AsyncTask. Jednotlivé instance využívají stejné vlákno. Z toho vyplývá, že pokud jich je vytvořeno velké množství najednou, musí počkat na jeho uvolnění. Docházelo tak k prodlevě při stahování důležitějších dat z databáze serveru.[7]

Problém jsem vyřešil implementací nové Singleton třídy ImageSetter. V konstruktoru jsem inicializoval pole, do kterého jsem ukládal instance ImageDownloaderAsyncTask. Při potřebě stažení důležitějších dat jsem zavolal metodu cancelAll() třídy ImageSetter, ve které jsem zrušil nedokončené stahování a pole vyprázdnil.

5.4 Uživatelské prostředí

Kvůli volbě implementace hlavního menu i zachování jeho dostupnosti v jakékoliv části aplikace jsem pro tvorbu uživatelského prostředí využil fragmenty. Fragmenty slouží k tvorbě moderního vzhledu. Mezi jejich přednosti patří znovu použitelnost a možnost jejich kombinace k tvorbě více okenních aplikací. Mají svůj vlastní životní cyklus, přijímají vlastní vstupní události a lze je přidávat či odebírat za běhu aplikace.

Podmínkou jejich použití je přiřazení k aktivitě, která ovlivňuje jejich životní cyklus. Příkladem může být uzavření aplikace kdy dojde k pozastavení či zničení přidružené aktivity a stejně tak fragmentů. Jejich přidání do aktivity je vysvětleno v kapitole 5.1.2.

V aplikaci jsem potřeboval zajisti komunikaci fragmentů s hlavní aktivitou. K tomu jsem v jednotlivých fragmentech definoval rozhraní a implementoval jej v záhlaví definice MainActivity. Implementaci rozhraní jsem zachytil pomocí metody onAttach() fragmentu, ve které jsem uložil instanci aktivity do třídní proměnné. Díky tomu jsem mohl volat metody rozhraní. Při odpojení fragmentu od aktivity by došlo k úniku paměti, proto jsem v metodě onDeatch() provedl uvolnění aktivity (Výpis 3). [12]

```java
@Override
public void onAttach(Context context) {
    super.onAttach(context);
    if (context instanceof OnHomeFragmentInteractionListener) {
        mListener = (OnHomeFragmentInteractionListener) context;
    } else {
        throw new RuntimeException(context.toString()
                + " must implement OnHomeFragmentInteractionListener");
    }
}

@Override
public void onDetach() {
    super.onDetach();
    mListener = null;
}
```

Výpis 3: Test implementace rozhraní aktivitou

5.4.1 Seznamy

Nejvíce používanou komponentou uživatelského prostředí v Android aplikacích jsou seznamy. Seznamy dovolují uživateli vybrání určité položky z mnoha dostupných. [24]

Operační systém Android nabízí k implementaci seznamů více možností. Vybral jsem pokročilou a flexibilní komponentu RecyclerView. Hlavní předností RecyclerView je pokročilá správa vykreslování velkého množství dat, které mohou být efektivně posouvána, s udržováním omezeného počtu náhledů. Dále se hodí pro seznamy jejichž položky se mění v reálném čase na základě akce od uživatele nebo při reakci na síťové události.

Pro použití RecyclerView jsem přidal odpovídající závislost do systému Gradle. RecyclerView používám ve více fragmentech. Pro ně jsem vytvořil univerzální XML soubor rozvržení, ve kterém jsem definoval widget ProgressBar sloužící k vykreslení upozornění na probíhající načítání a RecyclerView.

RecyclerView si vynucuje použití správce rozložení a adaptéru. Úlohou správce rozložení je výpočet velikosti a určení pozice položky v RecyclerView. Stejně tak rozhodování o jejím znovu použití, jestliže není pro uživatele viditelná. RecyclerView standardně nabízí tyto správce rozložení:

- LinearLayoutManager – vertikální či horizontální
- GridLayoutManager – mřížka

- StaggeredGridLayoutManager – stupňovitá mřížka

V rámci aplikace jsem nejvíce využíval LinearLayoutManager. K provázání LinearLayoutManager s RecyclerView jsem použil metodu setLayoutManager(), které jsem předal instanci LinearLayoutManager.

5.4.1.1 Adaptér

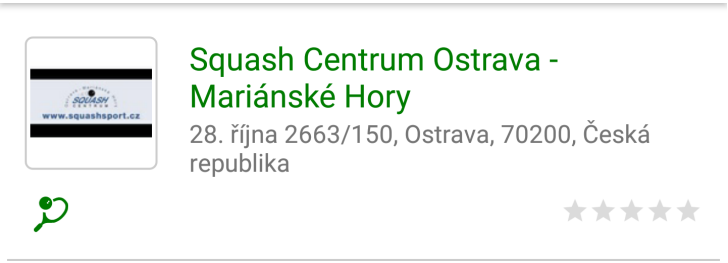
Stará se o naplnění RecyclerView daty. Toho docílí vytvořením nové instance položky seznamu a jejím naplněním daty z datového objektu. Pokud není položka pro uživatele viditelná, zajistí její nahrazení novými daty.

Adaptér jsem vytvořil jako podtyp Adapter<ViewHolder>. Z jeho deklarace je patrné použití návrhového vzoru ViewHolder. ViewHolder slouží k uchování odkazů na jednotlivé objekty View položky seznamu.

Pro každý seznam v aplikaci jsem vytvořil XML soubor s definicí vzhledu položky. V adaptéru jsem implementoval třídu ViewHolder. V konstruktoru třídy ViewHolder jsem přiřadil instance View prostřednictvím findViewById() do odpovídajících proměnných.

Data jsou většinou vložena do adaptéru až po jejich stažení a zpracování. Proto jsem implementoval metodu pro jejich uložení do třídního pole, přidání dalších, odstranění, navrácení pole a počtu položek v poli. V metodách pro vložení dat do pole jsem zavolal metodu notifyItemRangeInserted() adaptéru, díky které dojde k notifikaci o jejich přidání a vykreslení.

Naplnění instancí třídy ViewHolder daty z objektového modelu jsem docílil prostřednictvím metody onBindViewHolder() adaptéru. Metoda je automaticky zavolána pro každou položku seznamu a dojde tak k jejímu vykreslení. Příklad položky seznamu sportovních center je na obrázku 2.



Obrázek 2: View sportovního centra

V aplikaci pracuji s několika seznamy a pro každý jsem musel vytvořit vlastní adaptér. Ukázalo se, že implementace většiny jejich metod je totožná. Proto jsem přistoupil k vytvoření abstraktní generické podtřídy AbstratAdapter<AbstractViewHolder> třídy Adapter<ViewHolder>, abstraktní metody prepareViewHolder() a objektového modelu AbstractModel. To vedlo k oddělení podtříd třídy AbstractViewHolder do vlastní složky a zpřehlednění struktury projektu. Nové adaptéry jsem vytvořil jako podtyp třídy AbstratAdapter a implementoval v nich metodu

prepareViewHolder(). Implementace metody prepareViewHolder() je totožná s původní implementací onBindViewHolder(). [9]

5.4.1.2 Nekonečné rolování

Zobrazení velkého množství dat není pro RecyclerView problém. Ten nastává při jejich stažení ze serveru a zpracování. Tato operace může trvat několik sekund. Uživatel by po tuto dobu viděl obrazovku naznačující načítání což by mohlo vést ke špatnému dojmu z aplikace. Proto jsou data rozděleny na menší celky a pro jejich dodatečné stažení jsem implementoval nekonečné rolování.

Počátečním krokem bylo vytvoření abstraktního podtypu EndlessRecyclerViewScrollListener třídy OnScrollListener, který slouží k detekci posunu v seznamu. K docílení požadované funkcionality jsem nahradil metodu onScrolled(). V metodě jsem provedl test, zda-li je součet pozice poslední zobrazené položky se zvoleným prahem viditelnosti větší než celkový počet položek. Pokud tomu tak bylo zavolal jsem metodu onLoadMore() (Výpis 4).

V odpovídajících fragmentech jsem provedl inicializaci EndlessRecyclerViewScrollListener a implementaci metody onLoadMore(). Pomocí metody onLoadMore() jsem vytvořil instanci podtřídy AsyncTask, které jsem předal instanci adaptéru a počet položek v seznamu. V podtřídě AsyncTask jsem provedl stažení nových dat a jejich vložení do adaptéru. K provázání EndlessRecyclerViewScrollListener s RecyclerView jsem použil metodu addOnScrollListener().[1]

```
public void onScrolled(RecyclerView view, int dx, int dy) {
    super.onScrolled(view, dx, dy);
    int totalItemCount = mLayoutManager.getItemCount();
    int lastVisibleItemPosition = ((LinearLayoutManager) mLayoutManager).
        findLastVisibleItemPosition();

    if (loading && (totalItemCount > previousTotalItemCount)) {
        loading = false;
        previousTotalItemCount = totalItemCount;
    }

    if (!loading && (lastVisibleItemPosition + visibleThreshold) >
        totalItemCount) {
        onLoadMore(totalItemCount);
        loading = true;
    }
}
```

Výpis 4: Implementace metody onScrolled

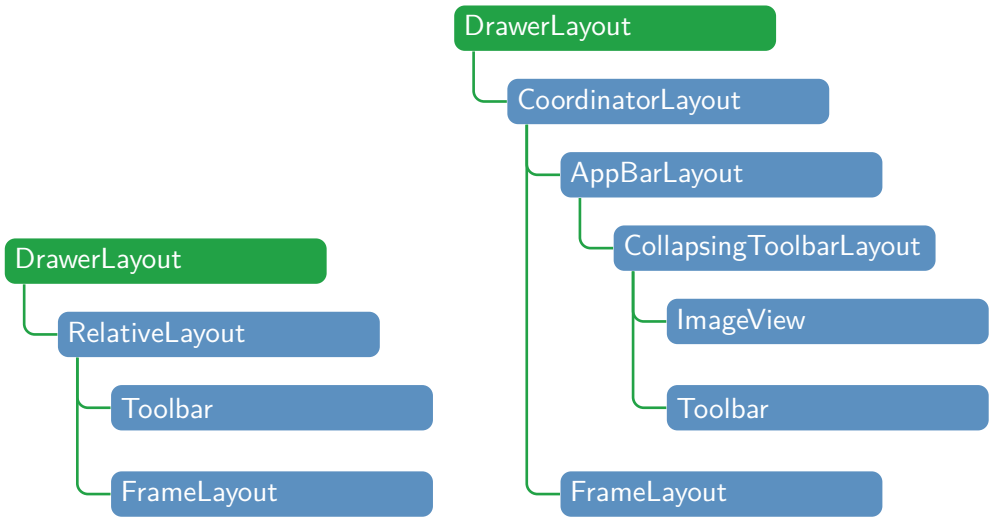
5.4.2 Kategorie

Pro splnění tohoto úkolu jsem využil TabLayout jež slouží k zobrazení horizontálních záložek. Spojil jsem jej spolu s komponentou ViewPager umožňující přecházení mezi jednotlivými záložkami, které jsou tvořeny fragmenty, za použití posunu prstu vlevo či vpravo.

Pro naplnění ViewPager fragmenty jsem vytvořil adaptér, který je podtypem FragmentStatePagerAdapter. V adaptéru jsem implementoval metodu addFragment(). Prostřednictvím metody jsem uložil předanou instanci fragmentu a nadpisu do třídních polí. Poté jsem upravil metodu navracející instanci fragmentu či nadpisu podle pozice v poli a počet fragmentů v poli. Posledním krokem byla inicializace adaptéru, který jsem naplnil požadovanými fragmenty a nadpisy.[2]

5.4.3 Detaily

K zobrazení banneru jsem se rozhodl použít Collapsing Toolbar (Obrázek 4). Ten nahradí hlavní lištu za obrázek a po posunu směrem nahoru obrázek pomalu zmizí a zobrazí se znovu hlavní lišta. K jeho implementaci jsem musel provést několik kroků. Prvním byla změna rozvržení hlavní aktivity (Obrázek 3). Kontejner RelativeLayout jsem zaměnil za CoordinatorLayout, pomocí kterého lze specifikovat chování pro jednotlivé objekty View jež obaluje. Nad stávající Toolbar jsem přidal ImageView pro obrázek banneru a tyto komponenty zabalil do CollapsingToolbarLayout, u nějž jsem specifikoval jak bude reagovat na posun. CollapsingToolbarLayout jsem následně zabalil do AppBarLayout řídící animace při posunu. Pomocí atributu layout_behavior jsem FrameLayout určil, že se hlavní obsah vykreslí pod CollapsingToolbarLayout. [23]

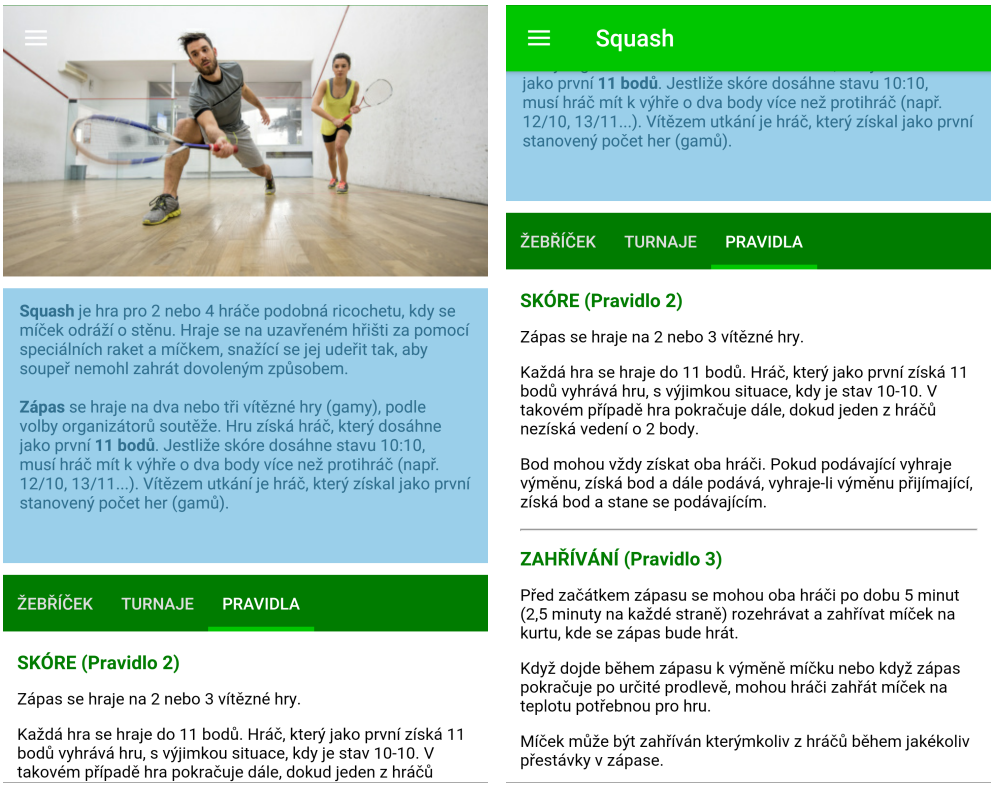


(a) Původní (b) Nová

Obrázek 3: Struktura rozvržení MainActivity

V dalším kroku jsem provedl inicializaci CollapsingToolbarLayout a ImageView v hlavní aktivitě. Dále jsem v ní implementoval metody prostřednictvím nichž jsem nastavil obrázek do ImageView, nadpis do CollapsingToolbarLayout a viditelnost či neviditelnost obrázku i nadpisu. Metody jsem přidal do definice rozhraní fragmentů, abych je mohl volat.

Posledním krokem byla úprava rozvržení detailů. Nebylo možné posouvat s obsahem, aby došlo ke skrytí banneru. Posunu jsem docílil záměnou hlavního kontejneru za NestedScrollView. U NestedScrollView jsem nastavil atribut fillViewport=true, kterým jsem specifikoval, že chci obsahem zaplnit celou obrazovku.



(a) Před posunem (b) Po posunu

Obrázek 4: Detailu sportu

5.4.3.1 Související položky

K zobrazení souvisejících položek jsem znovu využil kombinaci TabLayout, ViewPager a adaptéru (Kapitola 5.4.2, Obrázek 4). Nedošlo ale k celkovému vykreslení obsahu fragmentů. Problémem bylo použití NestedScrollView. V tomto kontejneru se špatně vypočítá výška obsahu fragmentu přiřazeného do třídy ViewPager. Použije se pouze zbývající místo na obrazovce po skrytí banneru.

Řešením bylo rozšíření třídy ViewPager. V konstruktoru jsem vypočítal poloviční výšku obrazovky zařízení a uložil ji do třídní proměnné. Následovalo přepsání metody onMeasure(). V ní jsem jako první zavolal metodu onMeasure() předka, aby došlo k inicializaci View fragmentu. Poté jsem provedl přesný výpočet výšky View fragmentu. U fragmentů jejichž výška je menší než hodnota třídní proměnné jsem použil právě tuto hodnotu. Díky tomu je u záložek použita vždy stejná minimální výška a lze mezi nimi pohodlně přecházet. Nakonec jsem zavolal metodu onMeasure() předka, aby došlo k přepočtu velikosti View fragmentu s novými hodnotami (Výpis 5). [3]

```
public MyViewPager(Context context) {
    super(context);
    WindowManager windowManager = (WindowManager) context.getSystemService(
        Context.WINDOW_SERVICE);
    Display display = windowManager.getDefaultDisplay();
    Point point = new Point();
    display.getSize(point);
    mScreenSize = point.y / 2;
}

@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    super.onMeasure(widthMeasureSpec, heightMeasureSpec);

    if (mCurrentView == null) {
        return;
    }

    mCurrentView.measure(widthMeasureSpec, MeasureSpec.makeMeasureSpec(0,
        MeasureSpec.UNSPECIFIED));
    int h = mCurrentView.getMeasuredHeight();

    heightMeasureSpec = MeasureSpec.makeMeasureSpec(h > mScreenSize ? h :
        mScreenSize, MeasureSpec.EXACTLY);

    super.onMeasure(widthMeasureSpec, heightMeasureSpec);
}
```

Výpis 5: Přepočet výšky obsahu

5.4.3.2 Pravidla sportu

Pravidla sportu server doručí jako JSON pole objektů. Každý obsahuje nadpis a textové pole kde index představuje odstavec. Zachování stejné struktury textu, jako na internetových stránkách sociální sítě, je dosaženo použitím HTML značek.

Ze začátku jsem k zobrazení pravidel používal widget TextView. Zjistil jsem, že v TextView nelze použít všechny značky a proto jsem jej nahradil za WebView, prostřednictvím kterého lze zobrazit internetové stránky uvnitř aplikace.

Ze získaného pole jsem vytvořil řetězce jehož struktura odpovídá HTML stránce. K tomu jsem použil StringBuilder. Do něj jsem prostřednictvím metody append() vložil HTML hlavičku v níž jsem pomocí CSS definoval velikost textu i barvu nadpisů. Do těla stránky jsem následně přidal hodnoty z jednotlivých objektů. Takto získaný řetězec jsem nahrál do instance WebView metodou loadData().

5.4.3.3 Rozpis zápasů události

Jednou z dodatkových informací o události je rozpis zápasů tzv. Play–off pavouk. Ke každému je přiřazen nadpis s umístění hráčů v turnaji. Množství rozpisů je ovlivněno počtem zúčastněních hráčů a jejich velikost se liší podle počtu zápasů. Kvůli tomu jsem je tvořil dynamicky a vkládal do LinearLayout kontejneru, který jsem definoval v rozvržení fragmentu.

K vykreslení rozpisu jsem použil tabulku TableLayout a pro každou velikost jsem implementoval odpovídající metodu. V každé metodě jsem vytvořil novou instanci TableLayout, TextView pro nadpis a postupně tvořil TableRow, jež odpovídá jednomu řádku v tabulce, které jsem přidal metodou addView() do instance TableLayout. (Výpis 6). Pro sloupce řádku jsem vytvořil nové soubory v res/drawable definující jejich pozadí. Podle toho, jestli jde o sloupec se jménem hráče nebo pouze o spojující čáru řádků, jsem zvolil odpovídající pozadí. V tomto kroku jsem narazil na problém s poníženou metodou pro nastavení pozadí hlavního kontejneru sloupce. Vyřešil jsem jej implementací podmínky, ve které jsem si ověřil verzi systému Android na jejímž základě jsem použil poníženou či novou metodu.

Během vkládání do instance kontejneru jsem narazil na problém. Pořadí nadpisů i tabulek se neshodovalo s internetovými stránkami. Vyřešil jsem jej ukládáním do zásobníku. Při vkládání do kontejneru ze zásobníku jsem postupoval podle principu LIFO (Obrázek 5). Protože by nebylo možné, na mobilních zařízeních, zobrazení celého rozpisu změnil jsem hlavní kontejner rozvržení fragmentu na HorizontalScrollView, které umožní posun do stran.

```
private void createSpider2(Spider spider, int from, int to) {
    if (spider != null) {
        TableLayout tableLayout = createTableLayout();
        TextView title = createTitle(from, to);
        Match match = spider.getMatches().get(0);
        Person person1 = match.getPerson1();
        Person person2 = match.getPerson2();
        long score1 = match.getScore1();
        long score2 = match.getScore2();
        TableRow firstRow = createTableRow();
        TableRow secondRow = createTableRow();
        TableRow thirdRow = createTableRow();

        firstRow.addView(createPlayerColumn(person1), 0);

        secondRow.addView(createScoreView(person1, person2, score1, score2),
             0);
        secondRow.addView(createPlayerColumn(whoWon(person1, person2, score1
            , score2)), 1);

         thirdRow.addView(createPlayerColumn(person2), 0);

        tableLayout.addView(firstRow);
        tableLayout.addView(secondRow);
        tableLayout.addView(thirdRow);

        mStack.push(tableLayout);
        mStack.push(title);
    }
}
```

Výpis 6: Vytvoření nejmenšího rozpisu

Obrázek 5: Rozpis zápasů

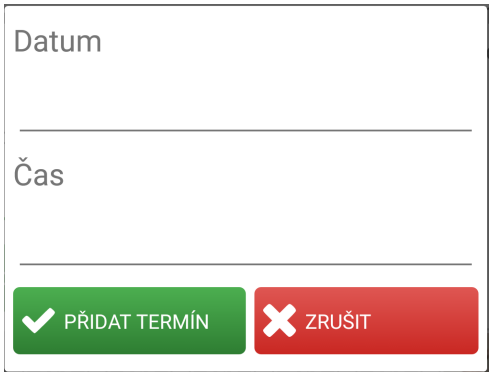
5.4.4 Vytvoření nového termínu pozvánky

K vytvoření nového termínu pozvánky musí uživatel zadat datum a čas. Pro tuto operaci jsem využil DialogFragment. DialogFragment je fragment, který se vykreslí jako plovoucí okno přes stávající obrazovku a uživatel se tak může rychle vrátit zpět. [11]

Jako prvním jsem vytvořil podtřídu z DialogFragment. V jejím rozvržení jsem definoval prvky pro zadání požadovaných údajů spolu s vlastními tlačíky, které jsou tvořeny obrázkem

a textem (Obrázek 6). Z důvodu ulehčení vložení požadovaných údajů jsem uvnitř třídy inicalizoval TimePickerDialog sloužící k zobrazení dialogového okna s výběrem času a pro datum DatePickerDialog. Zobrazení těchto oken je vyvoláno kliknutí na přidružený prvek pro vkládání textu. Hodnoty z těchto dialogů jsem převedel do podoby odpovídající jazykové lokalizaci zařízení a nastavil do odpovídajícího prvku. [18, 10]

Po kliku na potvrzovací tlačítko jsem provedl test zda-li byly požadované údaje zadány. Pokud ano vykreslí se přes dialog ProgressDialog sloužící k upozornění uživatele na probíhající operaci. Následně jsem datum i čas převedl do SQL formátu a vytvořil HTTPS POST požadavek, kterému jsem předal identifikační číslo pozvánky a získané hodnoty jež odešle jako parametr. Server vrátil odpověď. Jestliže byla kladná, uzavřel jsem dialog, přidal nový termín do pozvánky a zavolal metodu notifyItemChanged() adaptéru pro její překreslení. Pří záporné či nezadání údajů se zobrazí upozornění na vložení neplatných údajů.



Obrázek 6: Dialog pro nový termín

5.5 Implementovat přihlášení uživatele

Webové API jsou zatím navrženy k zasílání statických dat. Aby mohly být svázány s konkrétním uživatelem je zapotřebí jeho přihlášení. Vytvořil jsem tedy novou aktivitu navrženou pro použití fragmentů. Rozhodl jsem se tak, protože v budoucnu rozšířím aktivitu o možnost zaslání zapomenutého hesla. Oproti hlavní aktivitě má odlišný vzhled. Musel jsem tedy přidat nové téma do souboru res/values/styles a přiřadit jej v AndroidManifest.xml. Poté jsem vytvořil ve složce res/drawables nový XML soubor s definicí přechodu barev pozadí, nový fragment a XML soubor jeho rozvržení (Obrázek 7).

Obrázek 7: Přihlašovací obrazovka

5.5.1 Ověření přihlašovacích údajů

Přihlašovací údaje se řídí určitými požadavky. Ve fragmentu jsem inplementoval metodu v níž jsem provedl jejich ověření. Pokud byly splněny vytvořil jsem HTTPS GET požadavek. Při úspěšném přihlášení navrátil server cookie a ze zaslaných dat jsem vytvořil objektový model obsahující základní informace o uživateli. Neúspěch je naznačen zobrazením chybové hlášky jejíž znění je navráceno serverem. Pro její zobrazení jsem použil Snackbar, který se vykreslí přes spodní část obrazovky. [16]

5.5.2 Uložení cookie

Cookie se bude používat při komunikaci se serverem. Musel jsem ji tedy vhodně uložit. K tomu jsem použil SharedPreference, která je sdílená v rámci celé aplikace a slouží k ukládání primitivních datových typů formou klíč – hodnota. Takto uložené hodnoty jsou soukromé a přístup k nim má pouze aplikace. Využil jsem ji i pro uložení volby neodhlášení z aplikace. [15]

K uložení cookie dojde v metodě obstarávající HTTP GET požadavek. Pro přístup k SharedPreference je vyžadována instance aktivity. Abych zachoval znovupoužitelnost metody vytvořil jsem novou třídu ActivityTracker. Pomocí její statické metody setCurrentActivity() jsem uložil instanci aktivity do třídní proměnné. Během uspání nebo zničení aktivity by došlo k úniku paměti. Proto jsem v metodě onPause() aktivity provedl její uvolnění z proměnné.

5.5.3 Přechod do hlavní aktivity

Z fragmentu jsem zavolal metodu rozhraní, které jsem předal získaní model. Prostřednictvím metody jsem vytvořil novou instanci explicitního záměru Intent a objektu Bundle, do kterého jsem uložil předaný model. Nakonec jsem spustil MainActivity prostřednictvím metody startActivity(), které jsem předal instanci záměru. [17]

5.5.4 Vynechání zobrazení přihlašovací obrazovky

Spolu s touto aktivitou jsem vytvořil další, sloužící k zobrazení úvodní obrazovky aplikace, Singleton podtřídu NetworkReceiver třídy BroadcastReciver a v souboru AndroidManifest.xml specifikoval, že tato aktivita se spustí jako první.

V podtřídě NetworkReceiver jsem nahradil metodu onReceive(), ve které jsem ověřil dostupnost internetového připojení a výsledek uložil do třídní proměnné. Pro získání hodnoty jsem implementoval metodu isConnected(). Pomocí metody onResume() aktivity jsem zaregistroval naslouchač NetworkReceiver spolu s odpovídajícím filtrem. Díky tomu bude spuštěn v podprocesu aktivity a bude volána metoda onReceive() při změně stavu internetového připojení.

V průběhu běhu aktivity jsem ověřil dostupnost internetového připojení. Při jeho dostupnosti jsem získal hodnotu zůstat přihlášen z SharedPreference. Podle ní jsem rozhodl o spuštění hlavní či přihlašovací aktivity. Při nedostupnosti jsem rovnou spustil přihlašovací aktivitu.

6 Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné v průběhu praxe

Při vývoji nativních aplikací pro platformu Android jsou nezbytné znalosti programovacího jazyka Java. Základy a principy tohoto jazyka jsem získal v učebním předmětu Programovací jazyky I. Díky získaným znalostem jsem neměl problém při práci se základními datovými typy, příkazy, objekty, kolekcemi, vlákny a internetovou komunikací.

Za nejvíce přínosný považuji předmět Tvorba aplikací pro mobilní zařízení 2, jelikož je přímo zaměřen na vývoj mobilních aplikací pro platformu Android. Nevýhodou bylo, že jsem předmět vykonával souběžně s odbornou praxí. Umožnilo mi to sice využít získané poznatky přímo v praxi, ale většinou jsem byl při vývoji napřed. Nejvíce mi pomohla praktická znalost práce s aktivitami, adaptéry, záměry, vlákny, tvorba rozvržení a práce s modely.

Posledním předmětem, ze kterého jsem využil získané znalosti, je Vývoj internetových aplikací. Uplatnil jsem hlavně znalosti o struktuře HTML stránek, webovém API a formátu pro výměnu dat JSON. Neměl jsem tak problém s porozuměním struktury JSON formátu a jeho zpracováním.

7 Znalosti či dovednosti scházející v průběhu odborné praxe

Do firmy jsem nastoupil pouze se znalostmi získanými v průběhu studia. Hned na začátku se ukázalo, že hodně věcí nevím. Za největší nevýhodu považuji nezkušenost v oblasti vývoje nativních aplikací pro operační systém Android. I když jsem absolvoval předmět Tvorba aplikací pro mobilní zařízení 2, tak jsem neznal většinu komponent, které byly součástí aplikace.

Za další nedostatek považuji neznalost verzovacího nástroje Git. S tímto systémem jsem nikdy nepracoval, jelikož jsem v rámci školních projektů neměl potřebu uchovávat změny souborů v průběhu času a ani jsem se nesetkal s prací v týmu. Po seznámení se s tímto nástrojem bych řekl, že je pro týmovou práci nepostradatelným. Hlavně z důvodu možnosti tvorby nových částí projektu bez ovlivnění již fungujících. Taktéž umožňuje jednoduché obnovení předchozí verze. Toho jsem využil převážně při přepracování vzhledu aplikace.

V rámci školy jsem se nikdy nesetkal s tak rozsáhlým projektem. Z toho důvodu mi dělalo problém si zvyknout na práci rozdělenou na menší celky. Byl jsem zvyklý, že pokud si s daným problémem nevím rady, mohu začít na další části a k rozdělané se vrátit. Tento přístup ale v rámci firmy není zcela možný, protože se špatně odhaduje, které části jsou již hotovy a kolik zabraly času.

8 Závěr

Hlavním tématem této bakalářské práce je mobilní aplikace, pro operační systém Android, MatchToMe. V jednotlivých částech jsem se snažil popsat problematiku vývoje mobilních aplikací pro tento systém. Věnoval jsem se zadaným úkolům, jejich obecné problematice, problémům na které jsem při vývoji narazil a jejich řešení.

Mobilní aplikace je mobilním klientem pro sociální síť MatchToMe. Jedná se o rozsáhlý projekt a z toho důvodu nebyla možnost zahrnout všechny funkce, které síť nabízí. V průběhu vývoje přibyly k částem, na kterých jsem pracoval, další rozšíření, které nebyly z časových důvodů zahrnuty. Navázal jsem proto s firmou HS Interactive s.r.o. spolupráci a aplikace bude dále vyvíjena, aby mohlo dojít k jejímu uveřejnění na Google Play.

Jsem velmi rád za umožnění vypracování bakalářské práce formou odborné praxe. Mohl jsem si tak ověřit již osvojené dovednosti a prohloubit je. Za přínosný považuji pobyt ve firemním prostředí, hlavně práci v týmu. Taktéž poznání verzovacího systému a řízení projektů bude pro mě velkým přínosem do budoucna. Nové zkušenosti s vývojem Android aplikací a tvorby v programovacím jazyce Java určitě zúročím v další profesní kariéře. Během praxe mi můj konzultant Ing. Alexander Nikolskiy ukázal pokročilejší možnosti tohoto jazyka, které zrychlují vývoj a budou přínosné při dalším rozvoji mobilní aplikace.

Literatura

[1] *Endless Scrolling with AdapterViews and RecyclerView* [online]. Poslední změna 2. ledna 2017 [cit. 5. dubna 2017]. Dostupné na: <https://github.com/codepath/android_guides/wiki/Endless-Scrolling-with-AdapterViews-and-RecyclerView>.

[2] *Google Play Style Tabs using TabLayout* [online]. Poslední změna 4. dubna 2017 [cit. 6. dubna 2017]. Dostupné na: <https://guides.codepath.com/android/google-play-style-tabs-using-tablayout>.

[3] *Google Play Style Tabs using TabLayout* [online]. [cit. 15. dubna 2017]. Dostupné na: <https://guides.codepath.com/android/google-play-style-tabs-using-tablayout>.

[4] Atlassian. *JIRA* [online]. [cit. 3. dubna 2017]. Dostupné na: <https://www.atlassian.com/software/jira>.

[5] Bangad, A. *Navigation View - Material Design Support Library Tutorial* [online]. [cit. 4. dubna 2017]. Dostupné na: <http://www.android4devs.com/2015/06/navigation-view-material-design-support.html>.

[6] Google. *Android Studio* [online]. [cit. 3. dubna 2017]. Dostupné na: <https://developer.android.com/studio/index.html>.

[7] Google. *AsyncTask* [online]. [cit. 5. dubna 2017]. Dostupné na: <https://developer.android.com/reference/android/os/AsyncTask.html>.

[8] Google. *Caching Bitmaps* [online]. [cit. 5. dubna 2017]. Dostupné na: <https://developer.android.com/topic/performance/graphics/cache-bitmap.html>.

[9] Google. *Creating Lists and Cards* [online]. [cit. 5. dubna 2017]. Dostupné na: <https://developer.android.com/training/material/lists-cards.html>.

[10] Google. *DatePickerDialog* [online]. [cit. 6. dubna 2017]. Dostupné na: <https://developer.android.com/reference/android/app/DatePickerDialog.html>.

[11] Google. *DialogFragment* [online]. [cit. 6. dubna 2017]. Dostupné na: <https://developer.android.com/reference/android/app/DialogFragment.html>.

[12] Google. *Fragments* [online]. [cit. 5. dubna 2017]. Dostupné na: <https://developer.android.com/guide/components/fragments.html>.

[13] Google. *Material Design* [online]. [cit. 4. dubna 2017]. Dostupné na: <https://material.io/guidelines/>.

[14] Google. *Material Design for Developers* [online]. [cit. 4. dubna 2017]. Dostupné na: <https://developer.android.com/training/material/index.html>.

[15] Google. *SharedPreferences* [online]. [cit. 9. dubna 2017]. Dostupné na: <https://developer.android.com/reference/android/content/SharedPreferences.html>.

[16] Google. *Snackbar* [online]. [cit. 7. dubna 2017]. Dostupné na: <https://developer.android.com/reference/android/support/design/widget/Snackbar.html>.

[17] Google. *Start Another Activity* [online]. [cit. 15. dubna 2017]. Dostupné na: <https://developer.android.com/training/basics/firstapp/starting-activity.html>.

[18] Google. *TimePickerDialog* [online]. [cit. 6. dubna 2017]. Dostupné na: <https://developer.android.com/reference/android/app/TimePickerDialog.html>.

[19] Google. *Toolbar* [online]. [cit. 4. dubna 2017]. Dostupné na: <https://developer.android.com/reference/android/support/v7/widget/Toolbar.html>.

[20] Google. *<uses-permission>* [online]. [cit. 3. dubna 2017]. Dostupné na: <https://developer.android.com/guide/topics/manifest/uses-permission-element.html>.

[21] Google. *<uses-sdk>* [online]. [cit. 3. dubna 2017]. Dostupné na: <https://developer.android.com/guide/topics/manifest/uses-sdk-element.html>.

[22] HS Interactive s.r.o. *MatchToMe* [online]. [cit. 3. dubna 2017]. Dostupné na: <http://www.hs-interactive.eu/cs/products>.

[23] Leiva, A. *Design Support Library (IV): Collapsing Toolbar Layout* [online]. Poslední změna 9. července 2015 [cit. 15. dubna 2017]. Dostupné na: <https://antonioleiva.com/collapsing-toolbar-layout/>.

[24] Vávrů, J. a Ujbányai, M. *Programujeme pro Android: 2., rozšířené vydání.* 1. vyd. Praha: Grada Publishing, a.s, 2013. ISBN 978–80–247–8854–8.

A Obsah CD

Přiložené CD obsahuje:

1. Instalační APK aplikace
    - eu.hs_interactive.matchtome.apk