Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

Tvorba responzivních mobilních aplikací v Android Studiu

Creating responsive mobile applications in Android Studio

Bakalárská práce

Vypracoval: René Urmann

Vedoucí práce: PaedDr. Petr Pexa, Ph.D.

České Budějovice 2015

Zadání bakalářské práce

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta pedagogická
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**René URMANN**
Osobní číslo:	**P12094**
Studijní program:	**B7507 Specializace v pedagogice**
Studijní obor:	**Informační technologie a e-learning**
Název tématu:	**Tvorba responzivních mobilních aplikací v Android Studio**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

Cílem bakalářské práce bude rozbor tvorby responzivních aplikací pro mobilní zařízení s Android OS ve vývojovém prostředí Android Studio, založeném na IntelliJ IDEA. Výstupem práce bude sada praktických příkladů a konkrétní aplikace pro menzy Jihočeské univerzity, která bude umět zobrazovat aktuální jídelníčky jednotlivých menz JČU, uživatel se bude moci přihlásit do iCanteen a objednat si jídlo přímo z mobilního telefonu resp. tabletu. V rámci práce bude také zpracována podrobná dokumentace a tutoriál.

Rozsah grafických prací: **CD ROM**

Rozsah pracovní zprávy: **40**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **UJBÁNYAI, Miroslav. Programujeme pro Android. Vyd. 1. Praha: Grada, 2012, 187 s. Průvodce (Grada). ISBN 978-80-247-3995-3.**
2. **MURPHY, Mark L. Android 2: průvodce programováním mobilních aplikací. Vyd. 1. Brno: Computer Press, 2011, 375 s. ISBN 978-80-251-3194-7.**
3. **Svetandroida. Svetandroida [online]. 2010 [cit. 2013-03-10]. Dostupné z: http://www.svetandroida.cz/**
4. **KYPTA, Tomáš. Vyvíjíme pro Android. Vyvíjíme pro Android [online]. 2011, č. 2011 [cit. 2013-03-10]. Dostupné z: http://www.svetandroida.cz/vyvijime-pro-android-1-uvod-201103**
5. **Getting Started with Android Studio. Android Developers [online]. 2012 [cit. 2014-03-28]. Dostupné z: http://developer.android.com/sdk/installing/studio.html**
6. **První dojmy z Android Studia. Dotekománie.cz [online]. 2014 [cit. 2014-03-28]. Dostupné z: http://dotekomanie.cz/2013/05/prvni-dojmy-android-studia-13/**
7. **Android Dev. Android Dev [online]. 2014 [cit. 2014-03-28]. Dostupné z: https://support.google.com/googleplay/android-developer/answer/113468?hl=cs**

Vedoucí bakalářské práce: **PaedDr. Petr Pexa, Ph.D.**
Katedra informatiky

Datum zadání bakalářské práce: **27. března 2014**

Termín odevzdání bakalářské práce: **30. dubna 2015**

Mgr. Michal Vančura, Ph.D.
děkan

L.S.

doc. PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 27. března 2014

Prohlášení

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 23. června 2015

René Urmann

Poděkování

Rád bych poděkoval panu PaedDr. Petru Pexovi, Ph.D. za ochotu, vstřícnost a spolupráci při vedení mé bakalářské práce, za odborné rady, připomínky a za čas, který mi věnoval. Děkuji také mé rodině, která mi umožnila toto studium a všem, kteří mi byli při studiích oporou.

Abstrakt

Cílem bakalářské práce bude rozbor tvorby aplikací pro mobilní zařízení s Android OS ve vývojovém prostředí Android Studio založené na IntelliJ IDEA. Výstupem práce bude sada praktických příkladů a konkrétní aplikace pro menzy JU, která bude umět zobrazovat aktuální jídelníčky jednotlivých menz JU a dále se uživatel bude moci přihlásit do iCanteen a objednat si jídlo přímo z mobilního telefonu. V rámci práce bude také proveden průzkum zaměřený na využití této aplikace mezi studenty a profesory JU.

Abstract

The aim of this thesis is to analyze the creation of applications for mobile devices with Android OS in development environment Android Studio based on IntelliJ IDEA. The outcome of the work will be a suite of practical examples and one specific application for canteen JU, which will be able to view current menus for various cafeterias JU and the user will be able to log into iCanteen and order food directly from their mobile phone. As part of the work will also be carried out a survey on the use of this application between students and professors JU.

Klíčová slova

Android, Aplikace, Menzy JU, iCanteen, Průzkum

Keywords

Android, Applications, Menzy JU, iCanteen, Survey

Obsah

1 Úvod

Android Studio je nové vývojové prostředí založené na IntelliJ IDEA. Android studio bylo firmou Google oficiálně představeno 16. května 2013 na konferenci Google I/O. Od června 2013 je zdarma k dispozici pro uživatele. IntelliJ IDEA je speciálně vytvořena pro rozvoj Androidu a je k dispozici ke stažení pro Windows, Mac OS X a Linux.

Cílem bakalářské práce je zpracovat problematiku tvorby aplikací pro mobilní zařízení s Android OS v Android Studiu. Dále zjištění všech možností tohoto vývojového prostředí, popsání postupu instalace a jeho důležitých komponent. Rozebrání struktury projektu, práci s vrstvami, pohledy, dialogy, aktivitami, a také rozebrat problematiku stylování a responsivity aplikací.

Praktickou část bakalářské práce bude tvořit aplikace Menzy JU, realizovaná s využitím vývojového prostředí Andoid Studio. Dále bude vytvořena sada ukázkových aplikací využívajících hardwarové senzory zařízení s operačním systemém Android OS, na nichž bude technologie detailně prakticky představena.

1.1 Cíle práce

Cílem bakalářské práce je dopodrobna rozebrat tvorbu aplikací pro mobilní zařízení s Android OS ve vývojovém prostředí Android Studio. Zjistit všechny možnosti Android Studia, popsat postup instalace a důležitých komponent. Rozebrat strukturu projektu, blíže se podívat na práci s vrstvami, pohledy, dialogy, aktivitami, a také rozebrat problematiku stylování a responsivity aplikací.

Dále vytvořím sadu praktických příkladů a konkrétní aplikaci pro menzy JU, která bude umět zobrazovat aktuální jídelníčky jednotlivých menz JU a dále se uživatel bude moci přihlásit do iCanteen a objednat si jídlo.

Součástí práce bude proveden dotazníkový průzkum o využití aplikace Menzy JU mezi strávníky školních jídelen Jihočeské univerzity. Dotazník bude zpracován pomocí několika otázek, ten poté odešlu studentům a profesorům Jihočeské univerzity. Z výsledku dotazníku zjistím, využití objednávkového portálu mezi strávníky, používání mobilních zařízení pro tuto činnost a budoucí využití aplikace Menzy JU. Cílem bude rozšířit tuto aplikaci a zjednodušit tím objednávaní jídel.

1.2 Metody práce

Vytvořím seznam několika otázek, z nichž posléze vytvořím dotazník, který následně rozešlu studentům a profesům Jihočeské univerzity, po určité době zpracuji odpovědi rozšířenosti používání objednávkového systému. Dotazník bude elektronický a bude vytvořen pomocí webové služby Google Form.

V úvodu práce proberu novinky v tvorbě Android aplikací a výhody vývoje v Android Studiu. Poté přejdu na samotnou práci s vývojovým prostředím. Popíši postup instalace Android Studia a jeho důležitých komponent. Rozeberu strukturu projektu, blíže se podívám na práci s vrstvami, pohledy, dialogy, aktivitami, a také rozeberu problematiku stylování a responsivity aplikací. Dále se budu věnovat tvorbě sady demonstrujících aplikací a konkrétní aplikaci Menzy JU.

1.3 Východiska práce

Donedávna bylo základním, doporučeným způsobem vývoje Android aplikací prostředí Eclipse s pluginem Android Developer Tools. Jeho vývoj byl však ukončen a přechod na Android Studio je tak důrazně doporučován.

Android Studio je společným dílem Googlu a JetBrains. Je postaveno nad Community verzí prostředí IntelliJ IDEA. Díky tomu získává velké množství možností práce s kódem (navigace v kódu, našeptávání, refaktoring, analýza kódu...), ve kterých je IDEA špičkou v oboru.

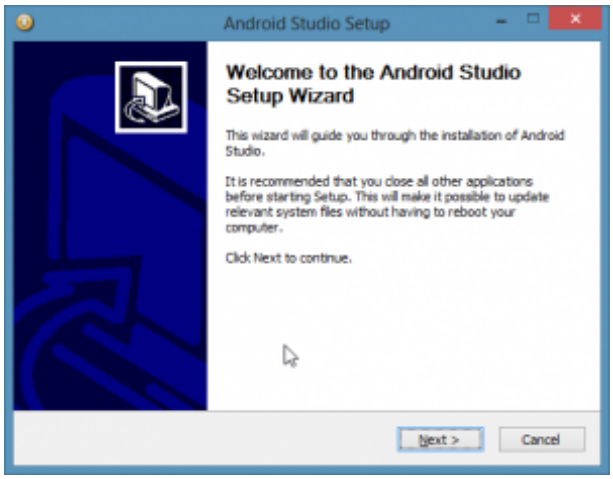
Toto vývojové prostředí je zdarma a má velmi dobrou podporu, tudíž je poslední dobou velmi vyhledávané.

2 Instalace Android Studia a potřebných komponent

Android Studio nabízí vše, co potřebujeme, abychom začali vytvářet aplikace pro Android, včetně Android Studio IDE a Android SDK nástrojů.[6]

2.1 Instalace Android Studia

Instalační balíček má přibližně 370 MB a obsahuje zároveň nejnovější Android SDK. Pryč je prvotní nastavování cest k SDK a instalace každého zvlášť.[6]

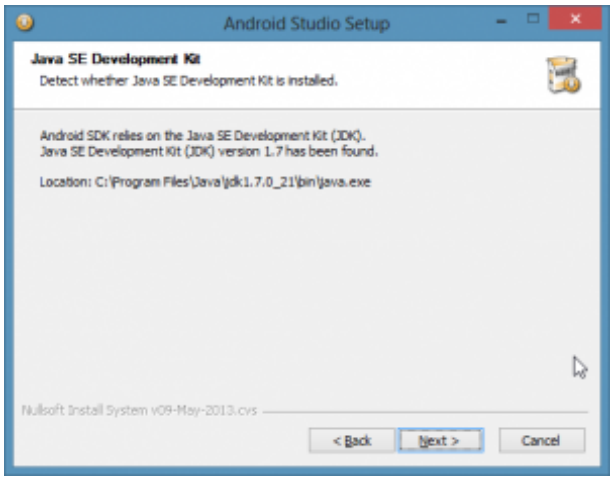


Obrázek 1: Instalace Android Studia

Samozřejmou nutností je mít nainstalovaný balíček JDK a nastavenou proměnnou prostředí

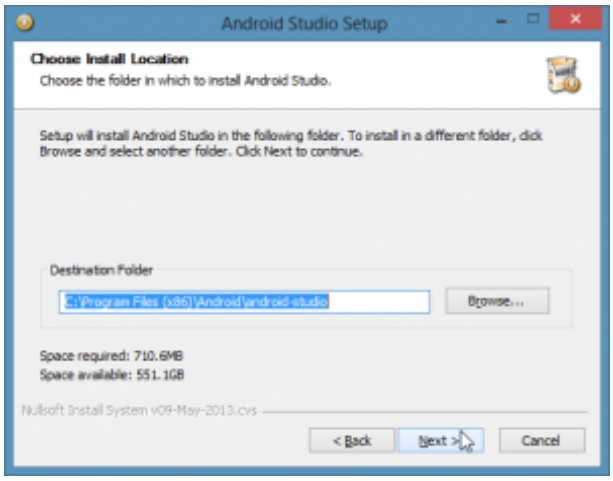
```
JAVA_HOME
```

tak, aby vedla do adresáře s JDK.



Obrázek 2: Instalace JDK

Jak lze vidět, po instalaci zabere Android Studio přes 700 MB na disku. Většina (kolem 460 MB) je však Android SDK. Přestože je větší než Elipse, jedná se o zanedbatelný rozdíl.[6]



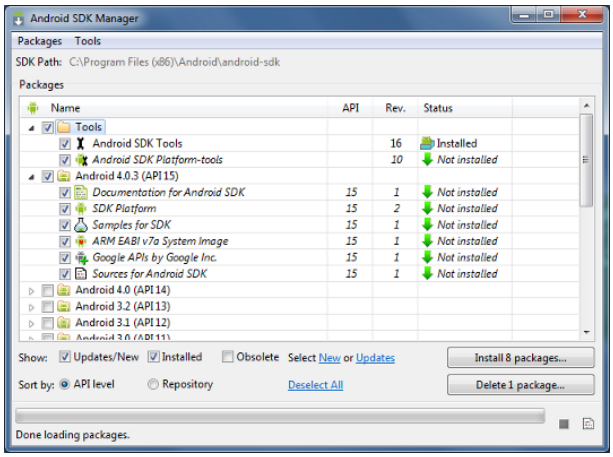
Obrázek 3: Velikost zabíráná Android Studiem na disku

2.2 Přidání SDK balíčků

Ve výchozím nastavení Android SDK nezahrnuje vše co potřebujeme, abychom začali vytvářet. SDK odděluje nástroje, platformy a další součásti do balíčků, které si můžeme stáhnout dle potřeby pomocí Správce Android SDK. Takže než začneme, existuje několik balíčků, které bychom měli přidat do svého Android SDK. V Android Studiu, klepneme na položku Správce SDK na panelu nástrojů. Když otevřeme SDK Manager poprvé, několik balíčků je vybráno výchozím nastavením. Necháme tyto položky vybrané.[6]

2.2.1 Získání nejnovějších SDK nástrojů

1. Otevřeme složku Nástroje a vybereme:
   - Android SDK Tools
   - Android SDK Platform-tools
   - Android SDK Build-tools (nejvyšší verze)
2. Otevřeme první složku Android XX (poslední verze) a vybereme:
   - SDK Platform
   - System image pro emulátor, jako ARM EABI v7a System Image

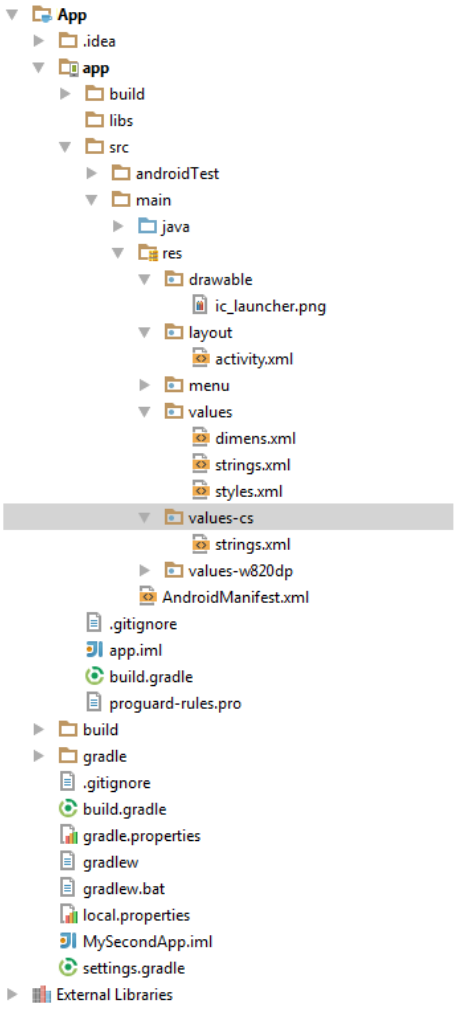


Obrázek 4: Náhled SDK Manageru

3 Projekt a struktura souborů

3.1 Struktura projektu

Nyní si řekneme něco o položkách ve struktuře projektu aplikace. Na obrázku níže je znázorněna struktura projektu.[7]



Obrázek 5: Struktura projektu

3.1.1 Složka app dále obsahuje podsložky:

- **build**, Android Studio si do tohoto adresáře ukládá předkompilované kusy kódu vždy, když spustíme aplikaci. Tímto adresářem se tedy nemusíme dále zabývat.
- **libs**, sem se ukládají knihovny naší aplikace, pokud bychom chtěli přidat nějakou funkčnost našemu projektu.
- **src**, zde nás zajímá main.

3.1.2 Main se dále větví na java a res.

- **java**, Java kód pro naše activity.
- **res**, resources, v této složce se nachází obrázky ve složkách:
  - ▷ drawable-hdpi
  - ▷ drawable-mdpi
  - ▷ drawable-xhdpi
  - ▷ drawable-xxhdpi

Obrázky pro výchozí složky jsou řazeny vzestupně podle kvality. V případě, že má zařizení fullHD displej, bude použitý nejkvalitnější adresář xxhdpi. Nejméně kvalitní složku hdpi použijeme, když bude mít displej zařízení například rozměry 480x320. V případě jednodušších aplikací doporučuji přidat složku drawable, tato složka nahradí všechny drawable-xxxx složky.[7]

- **Manifest** - AndroidManifest.xml Musí se nacházet v kořenovém adresáři každé aplikace. Obsahuje informace o naší aplikaci, které předá Androidu předtím, než se spustí kód aplikace. Nachází se zde jednotlivé activity a jejich nastavení, definují se zde práva aplikace a další nastavení jako je nastavení ikony atd.[7]
- **Layout**, zde se nachází xml soubory, ve kterých definujeme, jak se má co vykreslit.[7]

- **Values**, zde se nachází proměnné naší aplikace, týkající se textů popř. barev atd.[7]

V nově vytvořené aplikaci, tzv. Hello world aplikaci, se nachází v souboru activity.xml tento řádek:

```
android:text="@string/hello_world"
```

- @string - nás odkazuje na strings.xml
- `hello_world`

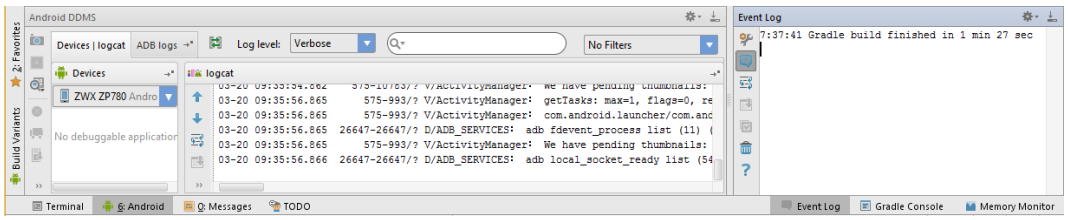
  - jméno stringu, jehož hodnotu si aplikace vytáhne z proměnné uložené v souboru strings.xml. Pokud se podíváme do tohoto souboru, najdeme zde řádek:

  ```
  <string name="hello_world">Hello world!</string>
  ```

  hodnota je tedy "Hello world!"

Tento systém je při programování mobilních aplikací velmi důležitý. Toto pravidlo známe i z programování pro klasické počítače. Jde o to, abychom měli jednu proměnnou a tu volali např.: 50x. Pokud totiž dojde ke změně hodnoty, stačí ji přepsat na jednom místě. Kdyby to bylo obráceně, museli bychom 50x přepsat text. Další výhoda, je snadné vytvoření vícejazyčné aplikace. Stačí přidat složku values-cs a do ní překopírovat strings.xml a přepsat jednotlivé názvy. Aplikace při spuštění na zařízení, které je v CZ jazyce, potom bude tahat hodnoty pro jednotlivé stringy z právě nově vytvořeného souboru.[7]

3.2 Android DDMS



Obrázek 6: Dalvik Debug Monitor Service

Dalvik Debug Monitor Service je GUI aplikace, kterou najdeme ve spodní části Android studia (je součástí SDK). Vede podrobné výpisy o všem, co vývojář potřebuje.[7]

3.2.1 Debugging a chyby

Pokud se něco pokazí, najdeme podrobný výpis v záložce Android. Můžeme zde nastavovat spousty filtrů, například jaké activity chceme vypsat, z jaké aplikace, jaký typ hlášek vypsat atd. Pomocí tohoto nástroje je snadné najít, co přesně se pokazilo.

3.2.2 TODO

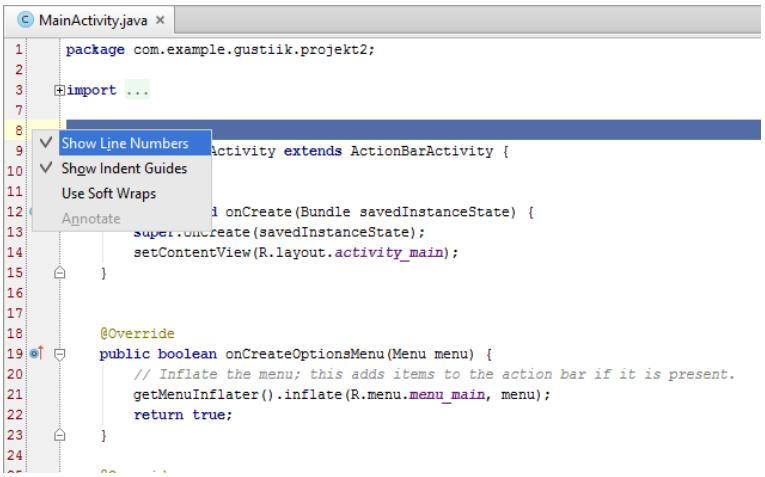
Pokud do naší aplikace někam přidáme komentář

```
"//TODO Nějaký text"
```

tak můžeme velmi jednoduše vypsat seznam všech TODO a to tak, že klikneme na: View -> Tool windows -> TODO Používat tuto funkci doporučuji, zejména u větších projektů.[7]

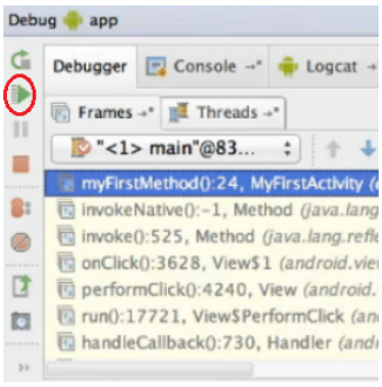
3.2.3 Debugging

V Android Studiu je poměrně jednoduchý postup debuggingu. Pokud se stane, že dojde k pádu aplikace, nejdříve se podíváme do Android DDMS záložka Android a projdeme si log. Zjistíme na jakém řádku je chyba. AS (alespoň v současné verzi) nemá automaticky zobrazená čísla řádků. Zobrazíme je pravým kliknutím na prostor na levé straně od okna, kam píšeme kód.[7]



Obrázek 7: Nastavení zobrazování řádků kódu

Ne vždy ale chybu hned vidíme. Můžeme si nastavit debug point na řádcích podle potřeby. (Kliknutím vedle čísla řádku) Potom můžeme spustit aplikaci v debug módu (Shift+F9 nebo klik na ”zeleného brouka”v horním panelu). Aplikace se zastaví v bodě, kam jsme umístili debug point a automaticky nás přesune v DDMS do Debugger módu. Zde vidíme přehled o všech proměnných a můžeme přeskočit k dalšímu debug pointu pomocí zelené šipky na levé straně.[7]



Obrázek 8: DDMS Debugger

Samozřejmě že pro správný debugging musíme mít zapnuté emulované zařízení s Androidem přímo na PC, popř. musíme mít připojen mobil s Androidem a nastaveným ladění přes USB.[7]

4 Životní cyklus a nový projekt

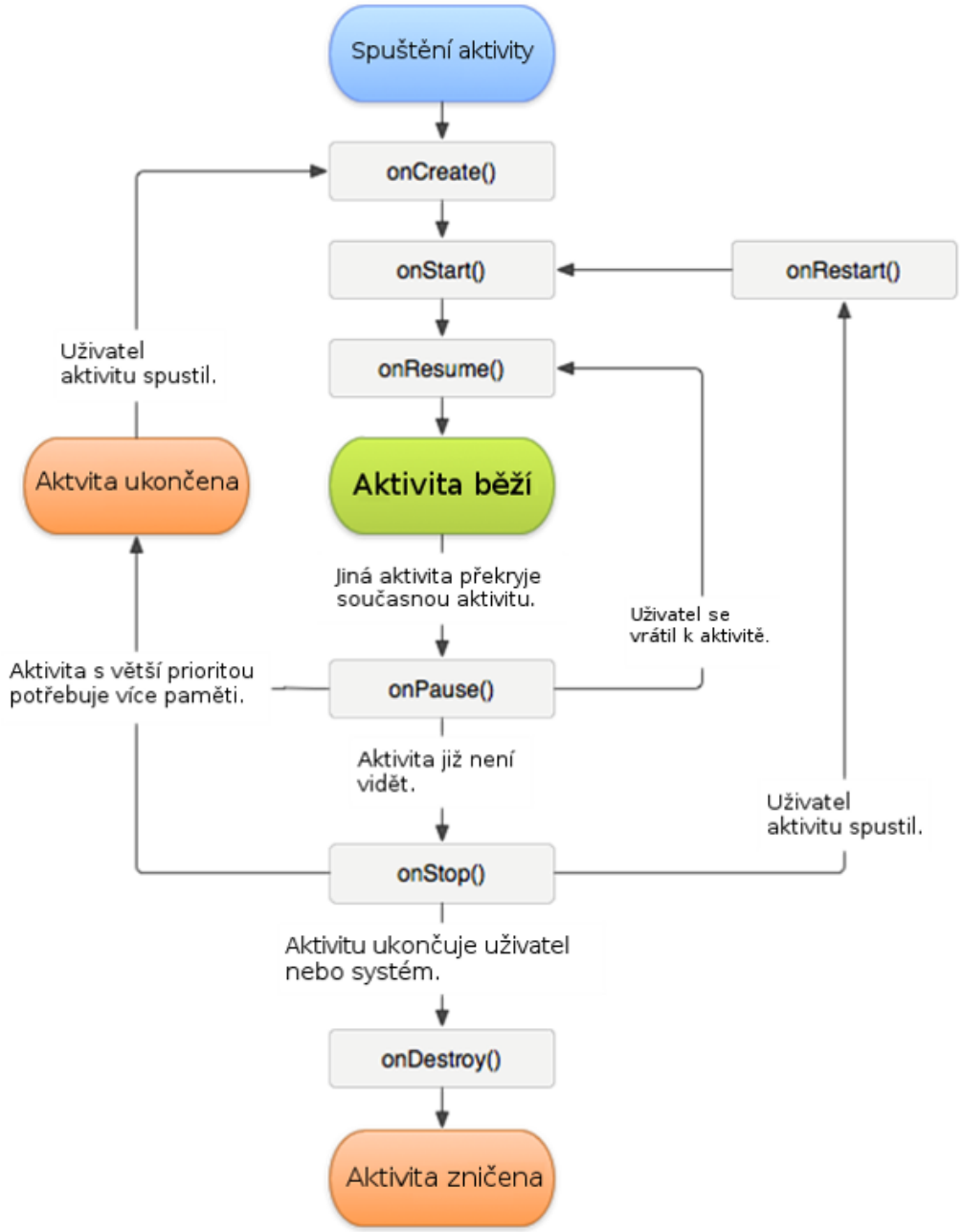
Nyní je třeba pochopit jak funguje aktivita a její životní cyklus a vědět jaké je pro nás nejvhodnější API. Při vytváření nového projektu si musíme dávat pozor na vhodné zvolení API, v pozdější fázi jej sice změnit můžeme, ale je lepší předejít zbytečným složitostem.[8]

4.1 Aktivity

Aktivitu si můžeme představit jako základní komponentu pro vykreslení grafického návrhu naší aplikace. Uvedu příklad. Zapneme aplikaci, zobrazí se úvodní animace. To je první aktivita. Poté se zobrazí menu aplikace s možnostmi ”Informace”, ”Poznámky”a ”Editor”, tohle bude druhá aktivita. Po kliknutí na jednotlivé nabídky v menu nás aplikace přesměruje do třetí, čtvrté nebo páté aktivity. Z naší aplikace můžeme přistupovat k aktivitám z jiných aplikací. Například nemusíme vytvářet vlastní aktivitu pro volání jako takové.[8]

4.2 Životní cyklus aktivity

Android, jakožto OS primárně pro chytré telefony, sám zodpovídá za activity a vše co s nimi souvisí. Je jasné, že majitel telefonu raději odpoví na příchozí hovor nebo SMS, než aby si hrál s aplikací. Může nastat několik situací, jako například příchozí hovor, pozastavení aplikace uživatelem (když se vrátí do hlavní nabídky) atd. Z toho plyne, že není vždy jasné, jak dlouho aktivita poběží, a proto existují různé metody, které se volají v různých situacích. Podívejte se na následující schéma.[8]



Obrázek 9: Schéma životního cyklu

4.2.1 Stav aktivity

Každá aktivita se nachází současně jen v jednom z následujících stavů.

1. Běží - Aktivita se úspěšně spustila a běží na popředí, je tedy pro uživatele viditelná.

2. Pauza - Aktivita jde vidět, je ale například překrytá jinou aktivitou. (Upozornění na příchozí SMS, hovor, nebo například dialog o plném nabití baterie při nabíjení). Uživatel se k takové aktivitě nijak nedostane, nemůže s ní pracovat.

3. Zastavená - Aktivita není vidět, uživatel k ní nemá přístup, ale její objekt ještě nebyl úplně zničen. Uživatel se k ní bude moci vrátit, pokud nebude zničena, například nedostatkem paměti.

4. Ukončená - Aktivita je úplně vypnutá.

4.2.2 Metody Lifecycle

`onCreate()`

Při spuštění activity se Android postará o základní věci jako vytvoření objektu a spuštění procesu. Dále zavolá metodu onCreate(). V ní nadefinujeme vše potřebné, aby se aktivita mohla rozjet, například jaké grafické rozhraní se nám má zobrazit, jak se dané grafické rozhraní bude zobrazovat (zda má být zobrazen status bar atd.), jestli se mají vytvořit globální proměnné nebo zda budeme globálně přistupovat k objektu TextView atd. Android studiem vygenerovaný kód:[8]

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_JmenoActivity);
}
```

Aktivita v metodě onCreate() nastaví layout pro svůj vzhled (GUI) řádkem:

```
setContentView(R.layout.activity_JmenoActivity);
```

Odkazuje tedy na layout *activity_menoactivity*, který je ve složce res/layout. R. poukazuje na automaticky vygenerovaný kód R.java, kde jsou uloženy zdroje.

`onStart()`

Volá se, pokud aktivita byla poprvé spuštěna (po onCreate()) nebo byla aktivována po svém skrytí (příchozí SMS, systémový dialog například o nabití baterie nebo jiný dialog jiné activity), nemůže dostat uživatelský vstup.

`onResume()`

Volá se těsně před tím, než je aktivita posunuta do popředí (restart, první spuštění nebo odpauzování), může dostat uživatelský vstup.

`onPause()`

Volá se před přechodem activity na pozadí. Systém dostává pravomoc násilného ukončení activity.

`OnStop()`

Volá se, když se má aktivita zastavit, není viditelná pro uživatele.

`OnDestroy()`

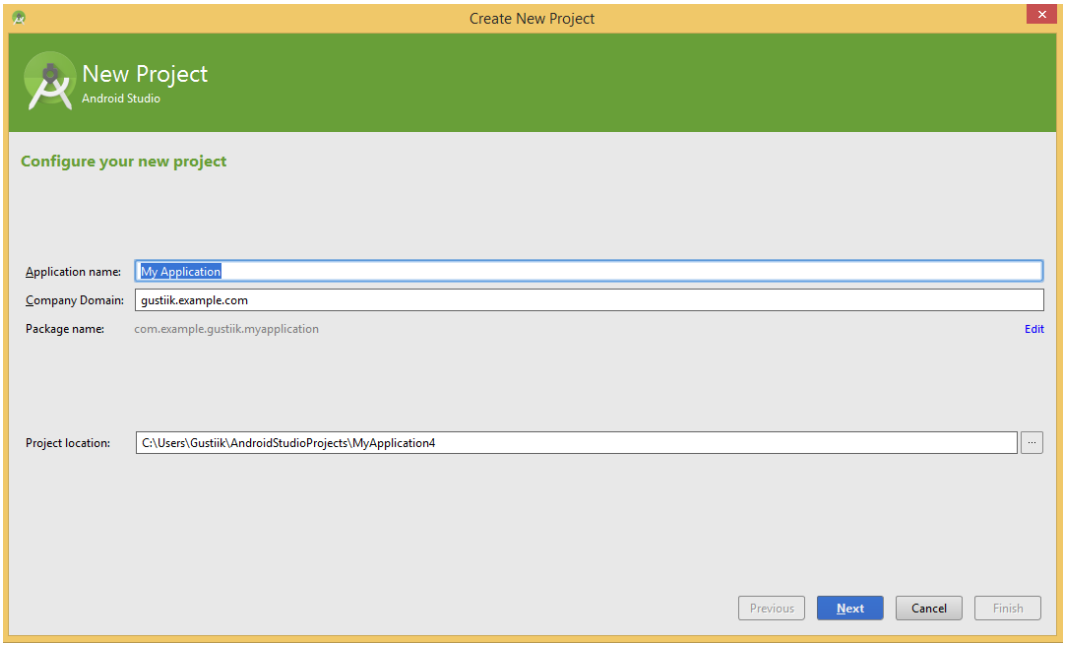
Volá se, před zrušením activity.

`onRestart()`

Jak vyplývá z předchozího diagramu, pokud byla zavolána onStop() a aktivita se restartuje, volá se onRestart(), jenž se provede před onStart(). Při programování není vždy nutné brát zřetel na všechny zmíněné metody. Někdy je zkrátka není potřeba použít. Jedinou podmínkou je použití metody OnCreate().[8]

4.3 Vytvoření projektu

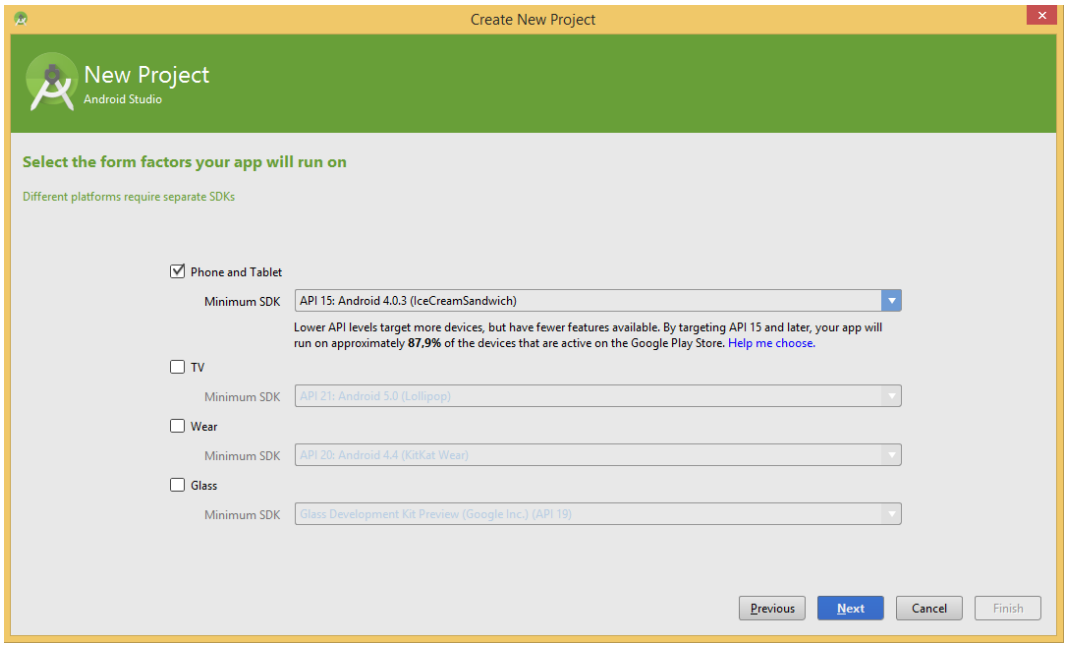
Vybereme File -> New project, Vyskočí okno ”Create New Project”.



Obrázek 10: Nový projekt

- Application name- jméno naší aplikace
- Company Domain- toto je originální ID každé aplikace. Nikdy se nesmí se žádnou jinou shodovat.

V dalším okně je třeba vybrat typ zařízení a API.



Obrázek 11: Nový projekt 2

Zatrhneme Phone and Tablet a vybereme API. Po výběru API si můžeme všimnout, že nám Android Studio vypsalo, kolik procent současných zařízení bude schopno naši aplikaci spustit.

4.3.1 API

API určuje jaké třídy a funkcionalitu můžeme při tvorbě naší aplikace použít. Pokud zvolíme API například 15, aplikace poběží na všech telefonech s API 15 až 21 (v době kdy píši tyto tutoriály je 21 nejnovější verze), ovšem uživatel s API 14 aplikaci nezapne.

Obecně platí, že čím vyšší API použijeme, tím více máme při programování možností. Nejen, že budeme moci využít u vyššího API například nějaké pokročilé animace, ale také si programování usnadníme. Například u malého API musíme řešit zákaz změny orientace nebo možnost uspávání aktivity přímo v Java kódu, což je značně složitější oproti řešení stejného problému ve vyšším API, kde stačí třeba jen jeden řádek v XML kódu.

V případě naší první aplikace nám stačí malé API. Pokud zvolíme API 10 (Android 2.3.3) tak naše aplikace pojede takřka všem. Z toho vyplývá, že čím

menší API zvolíme, tím více uživatelů bude moci aplikaci používat. V současnosti je podle statistik 99,2 procent zařízení kompatibilních s API 10. Pokud bychom ovšem vyvíjeli například nějakou složitější hru, nebo aplikaci, která potřebuje různé složité animace atd., je lepší zvolit vyšší API kvůli více funkcím.[8]

5 Vrstvy a pohledy (Layouts and views)

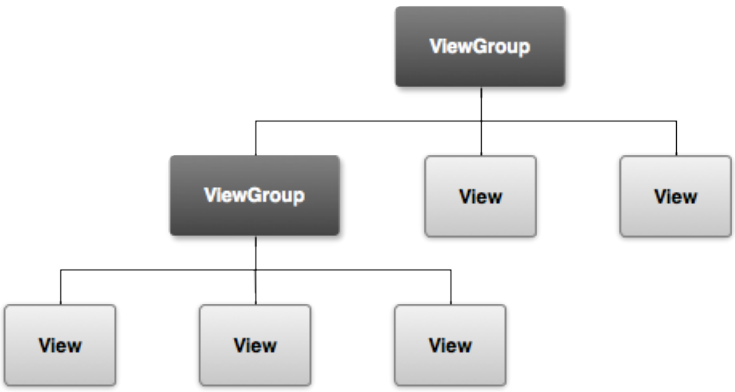
5.1 Vrstvy

Rozložení vrstvev definuje vizuální strukturu uživatelského rozhraní, jako je rozhraní pro aktivitu nebo widget. Vrstvu můžete deklarovat dvěma způsoby:

- Deklarovat prvky uživatelského rozhraní v XML.
  - ▷ Android poskytuje jednoduchý XML slovní zásobu, která odpovídá názoru, tříd a podtříd, jako jsou ty, pro widgety a rozvržení.
- Instanci rozvržení elementů za běhu.
  - ▷ Vaše aplikace může vytvářet View a ViewGroup objekty (a manipulovat s jejich vlastnosti) programově.

5.2 Pohledy(Views)

K použití views bychom měli vědět něco o nich, jak se odlišují od svého potomka ViewGroup, jaké metody mají a jak je používat. View je základní komponenta uživatelského rozhraní. Textový popisek, tlačítko, LinearLayout, rozbalovací menu atd. to všechno jsou view. LinearLayout určuje uspořádanost jednotlivých view. Třída View má velmi důležitého potomka ViewGroup, který má možnost obalovat jednu nebo více View i ViewGroup a stará se o jejich rozmístění a vykreslení. Na následujícím obrázku je ukázáno, jak může takový jednoduchý layoutový strom vypadat.[4]



Obrázek 12: Ilustrace hierarchie pohledu, která definuje rozložení UI.

Stromy mohou být mnohem složitější, měli bychom se snažit o co největší jednoduchost. Měli bychom také vytvářet kompaktní bloky uživatelského rozhraní jako fragmenty. V následujících řádcích budeme probírat jednotlivé potomky třídy View, ukážeme si, jak vypadají, popíšeme si důležité atributy a důležité funkce.[4]

5.2.1 View

Všechny ostatní třídy jsou podtřídou třídy View. Implementuje veškeré základní atributy a metody, jak ”technické”, což znamená jak se View vykreslí, tak i ”vzhledové”, s daným View můžeme dělat různé manipulace, měnění jeho atributů a tak dál. My se věnujeme jenom ”vzhledovým”metodám. View si můžeme představit jako obdélník, protože tak opravdu vypadá, ale díky šikovným tvůrcům, neboli kreslířům, můžou ostatní třídy vypadat mnohem lépe.[4]

Důležité atributy View

`ANDROID:ALPHA`

- Hodnoty: Floatové číslo v rozmezí $< 0; 1 >$.
- Odpovídající funkce: setAlpha, getAlpha
- Popis: 0 nastavuje průhlednost, 1 nastavuje neprůhlednost. Pokud chceme zadat desetinné číslo, musíme za něj napsat f(float) např.: 0.6f[4]

`ANDROID:BACKGROUND`

- Hodnoty: Odkaz na obrázek ze složky drawable, barvu nebo hexadecimální zápis barvy ve formátu #aarrggbb, #argb, #rgb, #rrggbb.
- Odpovídající funkce: setBackgroundResource, setBackgroundColor a odpovídající gettery
- Popis: Při použití barvy se doporučuje uložení hodnoty do složky color.xml. Zjednodušuje to pozdější úpravu barvy použitou na více místech v projektu.[4]

`ANDROID:ID`

- Hodnoty: Buď existující id (@id/foo),
  anebo nově vytvořené (@+id/bar).
- Odpovídající funkce: setId a odpovídající getter.
- Popis: Id unikatní být nemusí, avšak doporučuje se, aby v každé XML vrstvě bylo použito maximálně jednou, jinak by mohlo dojít k chybě.[4]

`ANDROID:PADDING`

- Hodnoty: Jakákoli délková jednotka
- Odpovídající funkce: setPadding, getPaddingLeft, getPaddingTop, getPaddingRight a getPaddingBottom

- Popis: Specifikuje vnitřní okraje konkrétního view. Při užití atributu android:padding se nastaví padding u všech čtyř stran stejný. Pokud tedy chceme nastavit padding u jednotlivých stran použijeme atributy android:paddingLeft, android:paddingRight, android:paddingTop a nebo android:paddingBottom.[4]

`ANDROID:VISIBILITY`

- Hodnoty: visible, invisible, gone
- Odpovídající funkce: setVisibility, getVisibility
- Popis: visibility:visible nastaví aby view byl vidět, visibility:hidden nastaví aby view vidět nebyl, ale zabírá stejné místo jako kdyby vidět byl, display:none také nastaví, aby view vidět nebyl, ale nezabírá žádné místo.[4]

Šikovné metody

`VIEW FINDVIEWBYID(INT ID)`

- Popis: Vyhledává podle id.[4]

`VIEW FINDVIEWWITHTAG(OBJECT TAG)`

- Popis: Funguje stejně jako findViewByld, akorát místo id hledá pomocí tagu. Vyhledává jenom tagy bez klíče.[4]

`CONTEXT GETCONTEXT()`

- Popis: Vrací objekt Context v rámci view, ve kterém existuje.[4]

`VIEW GETROOTVIEW()`

- Popis: Vrací kořenové view. Když konkrétní view není ničím potomkem, je tedy buď samo kořenovým view, anebo bylo zrovna vytvořeno, vrací samo sebe.[4]

`OBJECT SETTAG(INT KEY, OBJECT TAG)`

- Popis: Metoda vhodná kdybychom potřebovali k view přidat další data. Klíčem je id nastavené v resources. K získání uloženého objektu použijeme getTag(int key). Metoda Object setTag(int key, Object tag) má i variantu bez klíče, Object setTag(Object tag).[4]

5.2.2 TextView

TextView je přímý potomek View a zobrazuje text. Mezi jeho potomky se řadí například Button nebo EditText.

Důležité a zajímavé atributy TextView

`ANDROID:AUTOLINK`

- Hodnoty: none, web, email, phone, map, all, či kombinace oddělené svislítkem (|)
- Odpovídající funkce: setAutoLinkMask(int mask)
- Popis: Jednotlivé konstanty jsou překládány na čísla. Svislá čára je bitová operace nebo.[4]

`ANDROID:GRAVITY`

- Hodnoty: left, right, `center_horizontal` a mnoho dalších. Možno kombinovat svislítkem.
- Odpovídající funkce: setGravity(int mask)
- Popis: Nastavuje zarovnání textu na střed, doprava, doprava atd.[4]

`ANDROID:LINES`

- Hodnoty: Číslo či odkaz na číslo v resources
- Odpovídající funkce: setLines
- Popis: Abychom nepočítali přesnou výšku TextView, aby korespondovala s n řádky, použijeme atribut android:lines. Jeho hodnota je výška v řádcích. Existují také atributy android:maxLines a android:minLines.[4]

`ANDROID:TEXT`

- Hodnoty: Řetězec nebo identifikátor řetězce.
- Odpovídající funkce: setText
- Popis: android:text je atribut nastavující obsah TextView[4]

Užitečné metody třídy TextView

`INT GETLINECOUNT()`

- Popis: Vrací počet řádků textu zabírající v konkretním TextView. [4]

`CHARSEQUENCE GETTEXT()`

- Popis: Vrátí text. Na String ho převedete zavoláním metody[4] toString().

`INT LENGTH()`

- Popis: Vrátí délku textu ve znacích.[4]

5.2.3 ImageView

ImageView zobrazí obrázek. Obrázky se na telefonech moc nepoužívají. Na displejích mdpi odpovídá přibližně jeden pixel jednomu dp. Měli bychom přidat i velikosti pro ostatní displeje. Jestliže máme v /res/drawable-ldpi, /res/drawable-mdpi, /res/drawable-hdpi a /res/drawable-xhdpi stejně pojmenované obrázky odlišující se jenom velikostí, pak už je snadné ImageView použít:[4]

```
<ImageView android:src="@drawable/image"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
/>
```

5.3 ViewGroup

ViewGroup umí na rozdíl od View jedno nebo více view obalit. Má na starosti jejich vykreslení a rozmístění. Nyní něco málo k vlastnostem samotné ViewGroup. Mimo metody a atributy poděděných od View má ViewGroup i své vlastní. To jsou: [4]

- addView přidá nové view.
- View getChildAt(int index) vrací View na nějakou pozici (čísluje se od nuly). Pozice se stanovují dle času přidání - view co bylo přidáno později, má pozici s vyšším číslem.
- int getChildCount() vrací počet potomků view
- int indexOfChild(View child), která zjistí, na jaké pozici se nachází předané view
- void removeAllViews() smaže všechny potomky
- void removeViewAt(int index) smaže view na zadané pozici
- void remove Views (int start, int count) smaže count view začínající na pozici index [4]

5.3.1 LayoutParams

LayoutParams jsou vnitřní třídy ViewGroup nebo jejího potomka, které definují takzvané layout atributy, což jsou atributy s předponou layout_, které se nastaví každému obsaženému view, ale pracuje s nimi právě obalující ViewGroup. Takže android:layout_width a android:layout_height jsou jediné atributy definované ve ViewGroup.LayoutParams. Jsou definovány tři speciální konstanty MATCH_PARENT, FILL_PARENT a WRAP_CONTENT. U většiny Layoutů stačí nastavit právě tyto základní hodnoty. Jak je možné vytvořit view a vložit do nějaké skupiny?[4]

```
View v = new View(this); v.setBackgroundColor(Color.BLUE);
v.setLayoutParams(new ViewGroup.LayoutParams(
ViewGroup.LayoutParams.MATCH_PARENT, 30));
((ViewGroup) findViewById(R.id.view_group)).addView(v);
```

Nyní narazíme na problém. LayoutParams totiž v jednom ze svých konstruktorů přijímá dva integery, sířku a výšku. Ale v pixelech. Jak to převést?

```
 public float dpToPixels(int dp, Context ctx) {
Resources r = ctx.getResources();
return TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP,
dp, r.getDisplayMetrics()); }
```

Třída ViewGroup má na rozdíl od svých potomků dvě třídy definující layoutové atributy. Výše jmenovanou ViewGroup.LayoutParams a potom ViewGroup.MarginLayoutParams, která od ViewGroup.LayoutParams dědí a od níž dědí všechny třídy definující layoutové parametry. ViewGroup.MarginLayoutParams definuje čtyři layoutové parametry: android:layout_marginTop, android:layout_marginLeft, android:layout_marginTopBottom a android:layout_marginRight. Každý z nich přijímá jako hodnotu nějaký rozměr. Tyto atributy nastavují vnější okraje. Jsou víceméně totožné s CSS margin. Nejlépe to vysvětlí kód a obrázek.[4]

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView android:layout_width="match_parent"
  android:layout_height="40dp"
  android:background="@color/blue"
           android:text="Bez okrajů" />
<TextView android:layout_width="match_parent"
  android:layout_height="40dp"
  android:background="@color/red"
  android:padding="12dp"
  android:text="S paddingem"
/>
```

Obrázek 13: Text u view s paddingem je dole trochu useknutý, neboť u něj pro samé okraje nezbylo místo.

6 Stylování a design

V této kapitole si povíme, jak styly zapisovat, jaká je dědičnost stylů a ukážeme si některé vestavěné motivy.

6.1 Zápis stylu

Ve složce values mohou být styly. Je konvencí, že se pro styly používá soubor styles.xml, ale pokud chcete, můžete je dát klidně i do strings.xml, fungovat to bude. Kořenovým elementem XML souboru ve složve values je vždy <resources>. Samotný styl (může jich samozřejmě být více) je reprezentován elementem <style>, který má atribut name, jež specifikuje jméno stylu. Je konvencí, že jméno stylu je psáno pomocí PascalCase. A máme-li styl se jménem MyText, můžeme k němu přistupovat pomocí identifikátoru R.style.MyText resp. style/MyText. Potomky elementu <style> jsou elementy <item>, jejichž jediným atributem je opět name, to však tentokrát obsahuje název nějakého atributu, jehož hodnota je pak obsahem <item>-u.[4]

6.2 Motivy

Motivy (themes) jsou vlastně jen jiným využitím stylů. Zatímco styl je aplikován na jedno View (i kdyby to View bylo GroupView a mělo potomky, jich se už ten styl absolutně netýká), motiv je aplikován na všechna View v rámci nějaké Activity nebo celé aplikace.[4]

6.3 Dědičnost stylů

Zatímco v CSS může několik různých bloků cílit na stejný element, který potom všechny jejich styly zkombinuje, Android žádné kaskády neumí, a každé View tak může mít jen jeden styl. Přesto se v něm ale dá vyřešit problém, kdy potřebuji, aby měly všechny EditTexty v aplikaci stejný vzhled, až na to, že polovina bude mít tučné písmo a polovina ne. S tím, co známe doposud, bychom skončili buď u dvou duplikujících se stylů, až na to, že jeden má navíc android:textStyle=”bold”, anebo

u jednoho stylopisu a ručně přidaného atributu android:textStyle=”bold”každému EditTextu, který má být tučný.

Tvrzení, že každé View může mít jen jeden styl je sice pravdivé, ale ne úplně přesné. Může mít jen jeden atribut style. Ale atributy, které nejsou nastavené v předaném stylu, se vezmou z motivu dané Activity (či aplikace, pokud Activity vlastní motiv nastavený nemá), a pokud potřebné atributy nejsou ani v motivu, použijí se výchozí. Tím, že od sebe styly mohou dědit, můžeme vytvořit styl MyEditText, kde definujeme všechno pro netučné EditTexty, tedy všechno, co mají společné, a potom styl MyEditText.Bold, který bude dědit od MyEditText a přidá jen tučný text.

Samotný fakt, že nějaký styl dědí od jiného, můžete deklarovat dvěma způsoby. Dědit od nějakého stylu můžete pomocí atributu parent elementu <style>, jehož hodnotou je odkaz na style surovinu - rodiče.

Druhou možností jak dědit, ale to lze pouze od vlastních stylů, nikoli od androidích, je pomocí tečky. Pokud máte vytvořený styl MyEditText, od nějž chcete dědit a přidat tučné písmo, stačí nový styl pojmenovat MyEditText.Bold (za tečkou místo Bold může být cokoli). Android už pozná, že chcete dědit.[4]

Následující tři elementy <style> dělají to samé (přičemž poslední dodržuje pojmenovávací konvence a celkově ho považuji za nejlepší):

```
<style name="MyBoldStyle" parent="MyStyle">
    <item name="android:textStyle">bold</item> </style>
<style name="MyBoldStyle" parent="@style/MyStyle">
    <item name="android:textStyle">bold</item> </style>
<style name="MyStyle.Bold">
    <item name="android:textStyle">bold</item> </style>
```

6.4 Vestavěné motivy

Do Gingerbreadu (Android 2.3) bylo všechno jednoduché. Ale pak přišel Honeycomb (3.0) s novým holografickým stylem a věci zkomplikoval. No a aby toho nebylo málo, Ice Cream Sandwich (4.0) přidal ještě další motivy (ačkoli tam není změna tak markantní). Všechny možné motivy naleznete v souboru

{složka adk}/platforms/android-verze/data/res/values/themes.xml a v případě ICS a výše ještě
{složka adk}/platforms/android-verze/data/res/values/themes_device_defaults.xml, my si ukážeme ty základní důležité: Theme

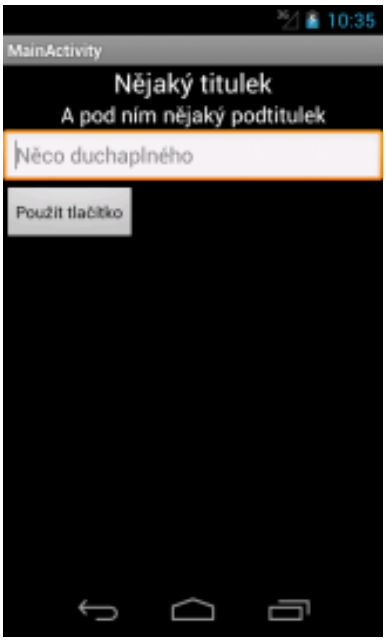
@STYLE:

- @android:style/Theme

ANDROID.R:

- android.R.style.Theme

Výchozí motiv na Androidu < 3.0. Můžete předpokládat, že budete pracovat s tmavým pozadím a světlým textem navrchu.



Obrázek 14: Theme

Theme.Light

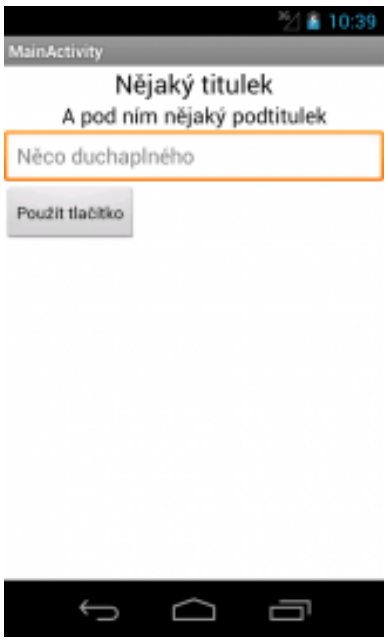
@STYLE:

- @android:style/Theme.Light

ANDROID.R:

- android.R.style.Theme_Light

Víceméně inverzní k Theme.



Obrázek 15: Theme.Light

Holo

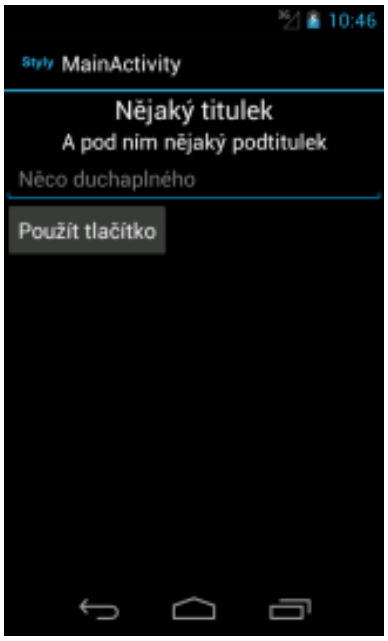
@STYLE:

- @android:style/Theme.Holo

ANDROID.R:

- android.R.style.Theme_Holo

Výchozí, tmavý holografický motiv na Androidu 3.x.



Obrázek 16: Holo

Holo.Light

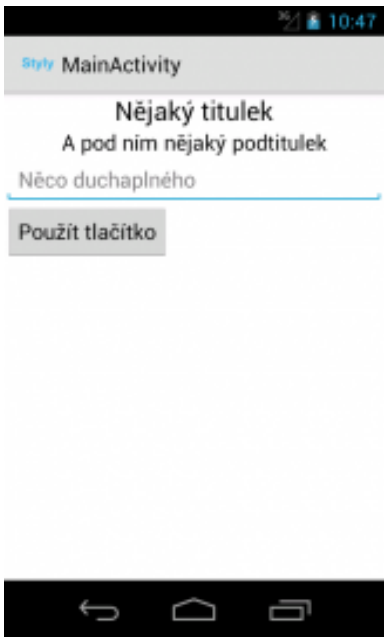
@STYLE:

- @android:style/Theme.Holo.Light

ANDROID.R:

- android.R.style.Theme_Holo_Light

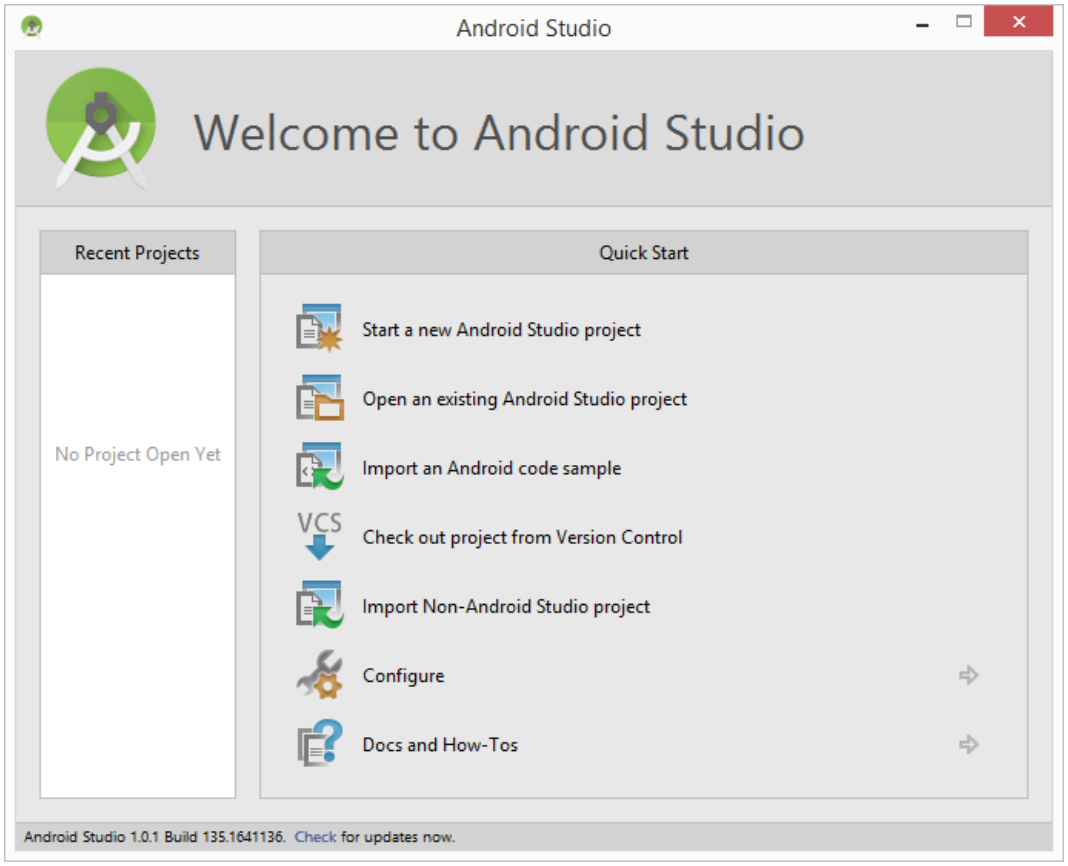
Světlá verze Holo



Obrázek 17: Holo.Light

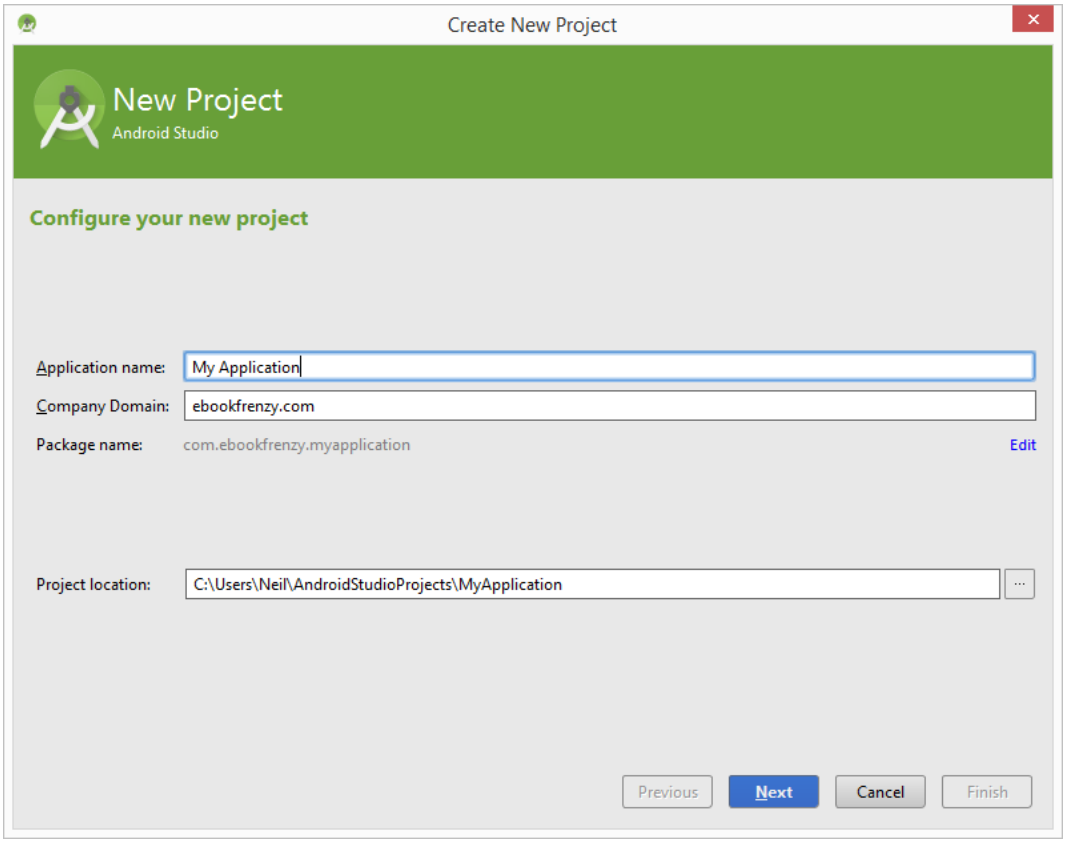
7 Tvorba základní aplikace

Prvním krokem v procesu vývoje aplikací je vytvořit nový projekt v prostředí Android Studia. Začněme tedy tím, že spustíme Android Studio tak, aby se objevila obrazovka "Vítejte v Android Studio", jak je znázorněno na obrázku:[9]



Obrázek 18: Vítejte v Android Studio

Jakmile se objeví toto okno, Android Studio je připraveno pro vytvoření nového projektu. Chcete-li vytvořit nový projekt, jednoduše klikněte na vytvořit nový projekt. Zobrazí se první obrazovka, průvodce novým projektem, jak je znázorněno na obrázku:[9]



Obrázek 19: Vytvoření nového projektu

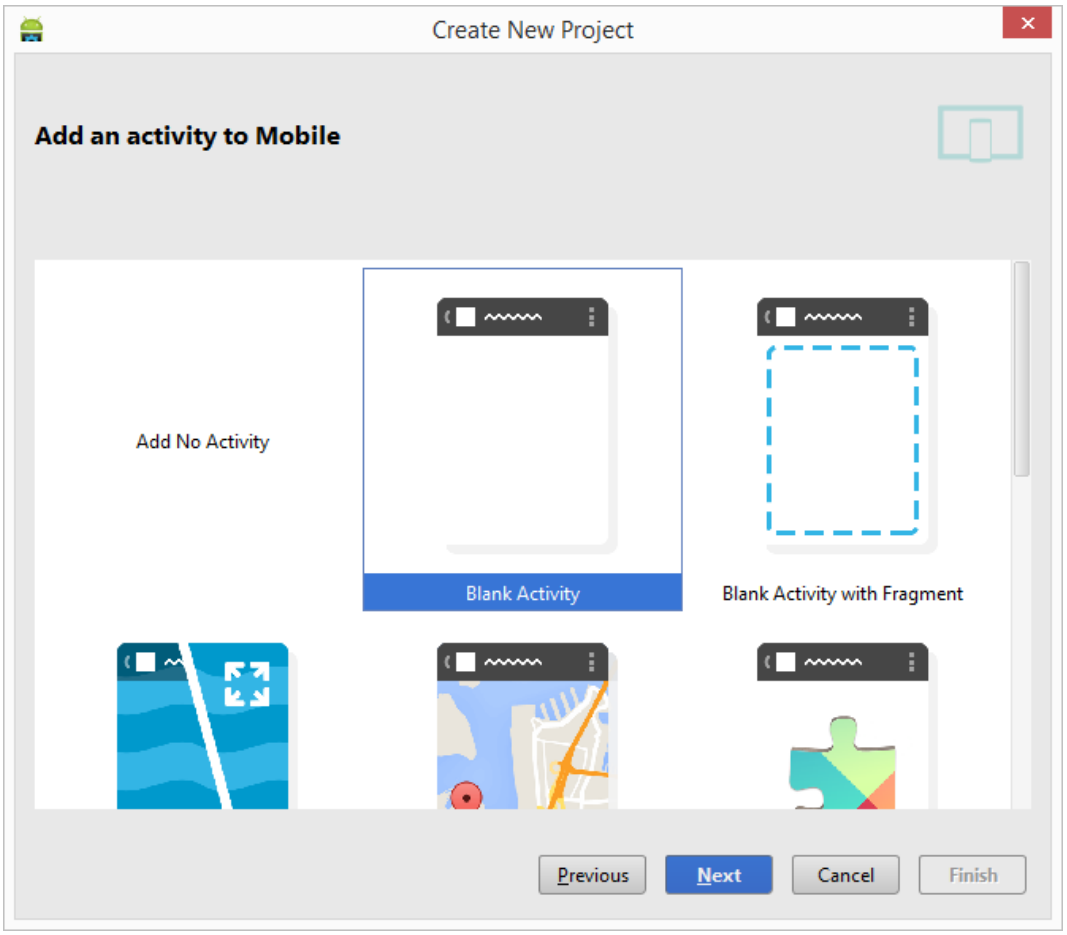
7.1 Definice projektu a SDK nastavení

V okně Nový projekt, nastavte název aplikace AndroidSample. Název aplikace je název, na který se bude aplikace odkazovat a bude jí identifikovat v Android Studiu. Název bude také použit při nahrání aplikace do obchodu Google Play. Jméno balíčku se používá k jednoznačné identifikaci aplikace v rámci systému Android. To by mělo být založeno na URL svého doménového jména následovaným názvem aplikace. Například, pokud je vaše doména www.mycompany.com a aplikace je pojmenována AndroidSample, pak název balíčku bude vypadat takto:[9]

```
com.mycompany.androidsample
```

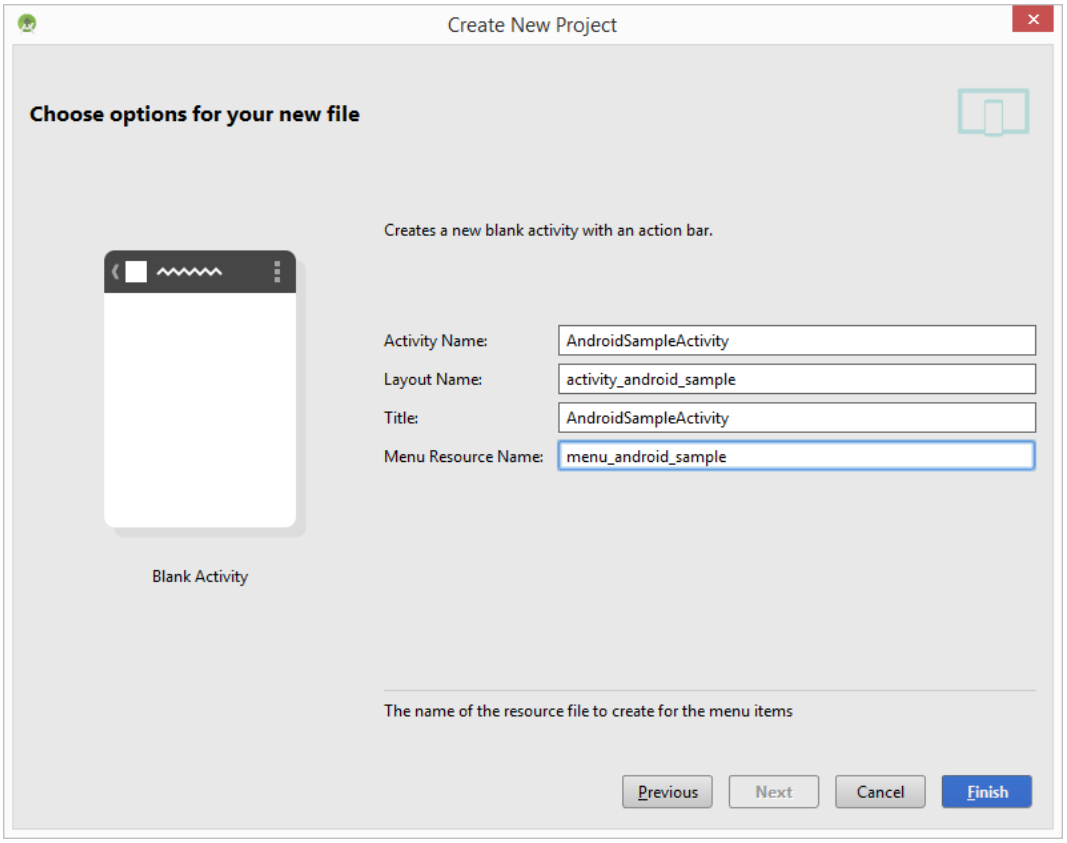
7.2 Vytvoření aktivity

Dalším krokem je definování typu počáteční aktivity, která má být vytvořena pro aplikaci. K dispozici při vývoji aplikací pro Android je škála různých typů aktivit. Pro účely tohoto příkladu vybereme možnost vytvořit Blank Activity.[9]



Obrázek 20: Výběr aktivity

S vybranou možností Blank Activity, klepněte na tlačítko Další. Na poslední obrazovce nastavujeme název aktivity a titulek aplikace AndroidSampleActivity. Aktivita se bude skládat z jedné vrstvy uživatelského rozhraní, která je pro účely tohoto příkladu pojmenována názvem `activity_android_sample`:[9]

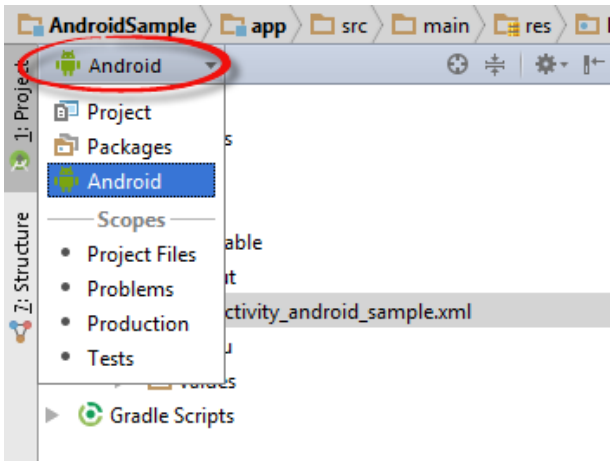


Obrázek 21: Nastavení projektu

Nakonec klikneme na tlačítko Dokončit k zahájení procesu tvorby projektu.

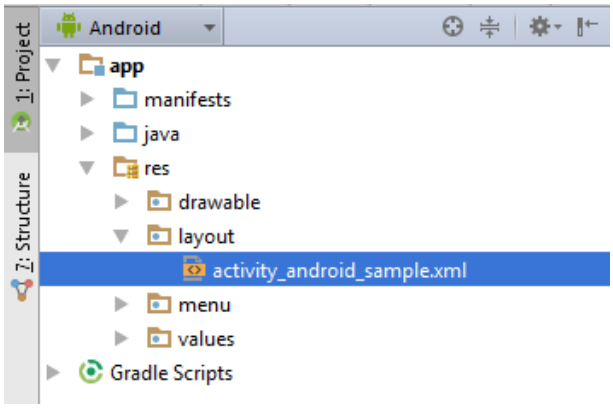
7.3 Úprava aplikace

V tomto bodě Android Studio vytvořilo a otevřelo základní projekt. Nově vytvořený projekt a odkazy na související soubory jsou uvedeny v okně tools > Project nacházející se na levé straně hlavního okna projektu. Okno projekt má řadu režimů, ve kterém mohou být zobrazeny informace. Ve výchozím nastavení bude tento panel v režimu Android. Toto nastavení je řízeno rozbalovacím menu v horní části panelu, jak je zvýrazněno na obrázku. Pokud panel není v současné době v režimu Android, klikneme na něj v tomto menu:[9]



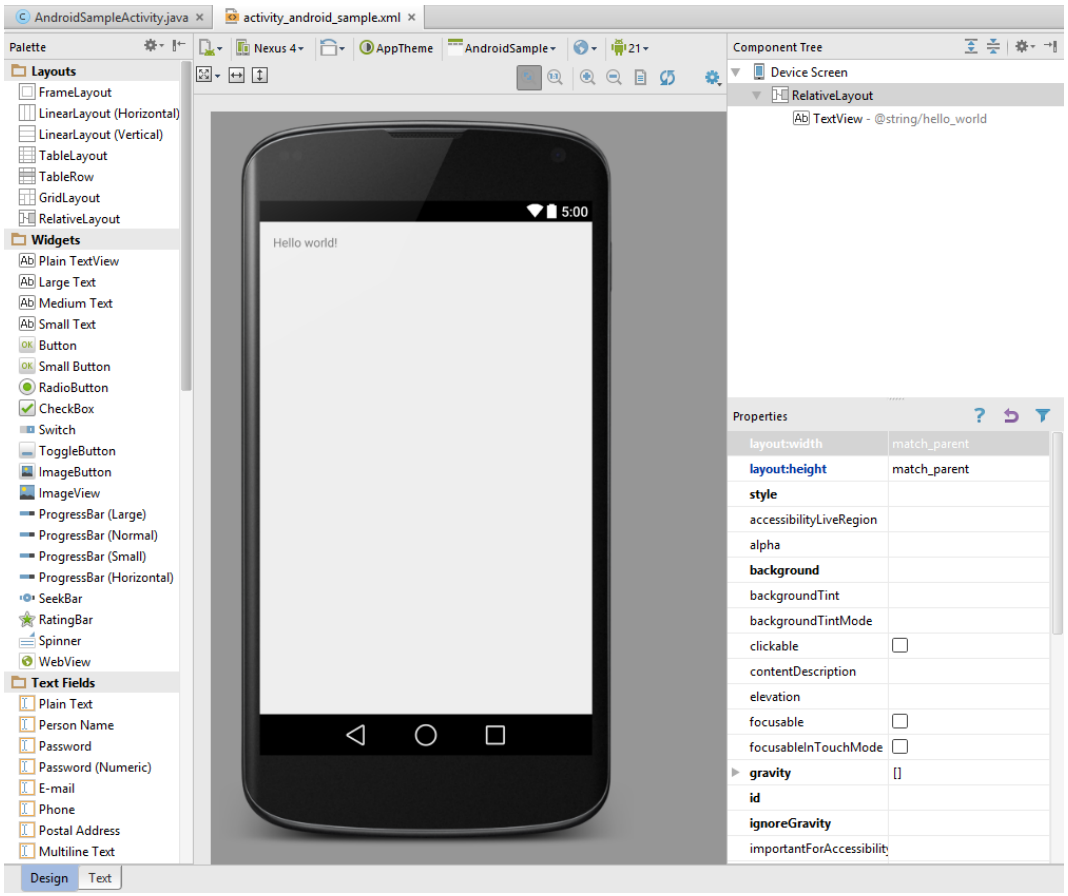
Obrázek 22: Projekt menu

Dalším krokem v tomto výukovém programu je upravit uživatelské rozhraní naší aplikace tak, aby zobrazoval větší textové pole. Uživatelské rozhraní naší aktivity je uloženo v souboru s názvem `activity_android_sample.xml`, který se nacházejí v app > res > layout v hierarchii souborů projektu. Pomocí panelu Project, vyhledejte tento soubor.[9]



Obrázek 23: Uživatelské rozhraní

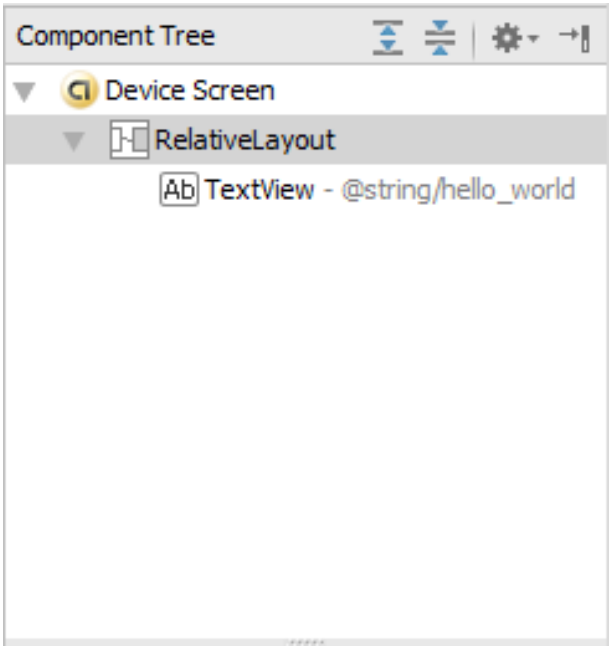
Jakmile se zde nacházíme, dvakrát klikneme na soubor, abychom jej načetli do uživatelského rozhraní, které se objeví ve středu panelu hlavního okna Android Studia:



Obrázek 24: Design mód

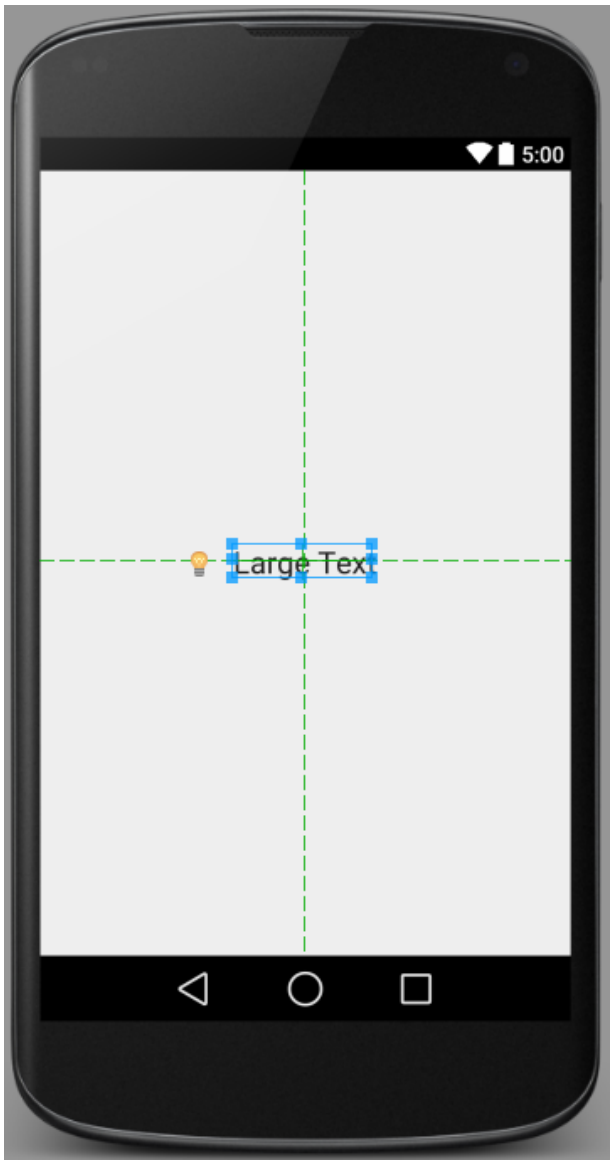
V panelu nástrojů v horní části editoru vrstev je menu, které je defaultně nastaveno na Nexus 4, který je vizualní reprezentací zařízení v rámci panelu Designer. Široká škála dalších zařízení je k dispozici v této nabídce. Chcete-li změnit orientaci zařízení mezi landscape a portrait stačí použít rozbalovací nabídky znázorněné ikonou  . Jak lze vidět na obrazovce přístroje, vrstva již obsahuje štítek, který zobrazuje Hello World! zprávu. Panel na levé straně je paleta obsahující různé kategorie prvků uživatelského rozhraní, které mohou být použity ke konstrukci uživatelského rozhraní, jako jsou tlačítka, popisky a textová pole. Je třeba poznamenat, že ne všechny komponenty uživatelského rozhraní jsou pro uživatele viditelné. Jedna taková kategorie se skládá z vrstev. Android podporuje řadu různých vrstev, které poskytují různé úrovně kontroly nad tím, jak jsou umístěny a spravovány komponenty vizuálního uživatelského rozhraní na obrazovce. Současný design

byl vytvořen pomocí RelativeLayout. To můžeme ověřit přezkoumáním informací v panelu Component Tree, který ve výchozím nastavení je umístěn v pravém horním rohu panelu Designer.[9]



Obrázek 25: Component Tree

Jak můžeme vidět, hierarchie komponent uživatelského rozhraní se skládá z RelativeLayout rodiče s jediným potomkem v podobě TextView objektu. Prvním krokem při modifikaci žádosti je odstranit komponentu TextView z designu. Začněme kliknutím na objekt TextView v rámci pohledu uživatelského rozhraní. Zobrazí se s modrým ohraničením kolem. Po výběru stiskneme klávesu Delete na klávesnici a tím odstraníme objekt z vrstvy. V panelu Palette vyhledáme kategorii Widgety. Klepneme a přetáhneme Large Text objekt a umístíme jej na střed v designu módu uživatelského rozhraní, ve chvíli kdy se objeví zelené čáry označující střed displeje:



Obrázek 26: Zarovnání textu

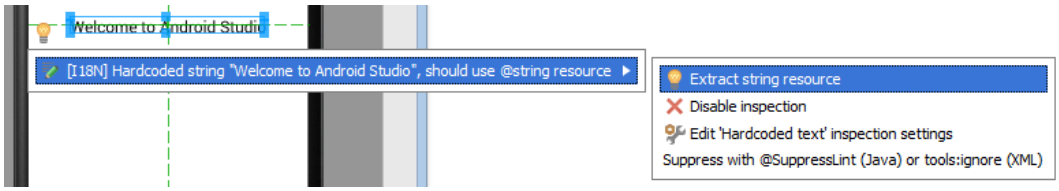
Dalším krokem je změnit text, který je v současné době zobrazený komponentou TextView. Poklepáním na objekt v návrhu vrstvy se zobrazí editační panel textu a id objektu. V tomto panelu změňte vlastnost text z ”Large Text”na ”Vítejte v Android Studio”.[9]

Obrázek 27: Editační panel

V tomto bodě je důležité vysvětlit žárovku vedle objektu TextView. To naznačuje možný problém a poskytuje některé doporučené řešení. Kliknutím na ikonu nás informuje v tomto případě, že tento problém je následující:[9]

```
[I18N] Hardcoded string "Welcome to Android Studio",
should use @string resource
```

Tato chyba nám připomína, že při vývoji aplikací pro Android by měly být uloženy atributy a hodnoty jako textové řetězce v samostatných XML souborech ve složce resources. Je to výhodné pro snadnější překlad aplikace. V tomto případě se chystáme vytvořit nový zdroj s názvem welcomestring a přiřadit k němu řetězec "Vítejte v Android Studiu". Klikneme na šipku vpravo od varovného hlášení pro zobrazení menu možných řešení.[9]



Obrázek 28: Chybové hlášení

Z nabídky vybereme volbu Extract string resources pro zobrazení dialogového okna Extract Resource. V tomto dialogu zadáme welcomestring jako název zdroje a klikneme na tlačítko OK. Řetězec je nyní uložen jako zdroj v app > res > values > strings.xml.[9]

8 Parsovaní dat pomocí knihovny Jsoup

V této kapitole chci popsat, jak používat JSoup v Android aplikacích. JSoup je Java knihovna, která nám pomáhá získat a manipulovat s HTML soubory. Pomocí této knihovny můžeme parsovat html stránku pro Android zařízení. Jsou situace, kdy chceme analyzovat a extrahovat některé informace z HTML stránky a zobrazovat je v aplikaci. V tomto případě můžeme použít knihovnu JSoup, která má sadu výkonných API, která jsou velmi snadno použitelná a integrovatelná v našich Android projektech. V této kapitole se budeme věnovat tomu, jak nastavit projekt Android, který používá JSoup knihovnu a získává nějaké informace.[10]

8.1 Úvod Jsoup

Jak již bylo zmíněno JSoup je Java knihovna poskytující sadu API, která umožňuje parsovat data a manipulovat s HTML soubory. Existuje několik metod, jak číst a parsovat HTML stránku. V našem případě chceme, aby se stránka načítala ze vzdáleného serveru, tudíž musíme poskytnout adresu URL. Pokud chceme parsovat stránku jako DOM, píšeme:[10]

```
Document doc  = Jsoup.connect(URL).get();
```

kde doc je instancí třídy dokumentu, kterou má dokument načíst. Nyní máme dokument a můžeme získávat informace. Můžeme získat title nebo jiné informace, pomocí HTML tagů. Například pokud chceme dostat všechny tagy pojmenovaný meta, píšeme:[10]

```
Elements metaElems = doc.select("meta");
```

kde select je metoda používaná, když chceme získat tagy pomocí CSS-dotazu. Například, pokud chceme získat hodnotu atributu z tagu, píšeme:

```
String name = metaElem.attr("name");
```

kde ”name”je název atributu. Navíc můžeme vybrat všechny prvky z HTML stránky, které mají specifickou hodnotu CSS třídy. Například na této webové stránce jsou některé prvky, kde jejich CSS třída se rovná hodnotě ”topic”, takže píšeme:[10]

```
Elements topicList = doc.select("h2.topic");
```

kde vybereme právě h2 tag, který má třídu s názvem ”topic”.

8.2 Nastavení projektu a integrace JSoup

První věc, kterou musíme udělat je vytvoření standardního projektu v Android Studiu, který obsahuje jednoduchou aktivitu. Jakmile je projekt připraven, musíme přidat JSoup dependency. V sekci dependency ve složce build.gradle přidáme:[10]

```
compile 'org.jsoup:jsoup:1.7.3'
```

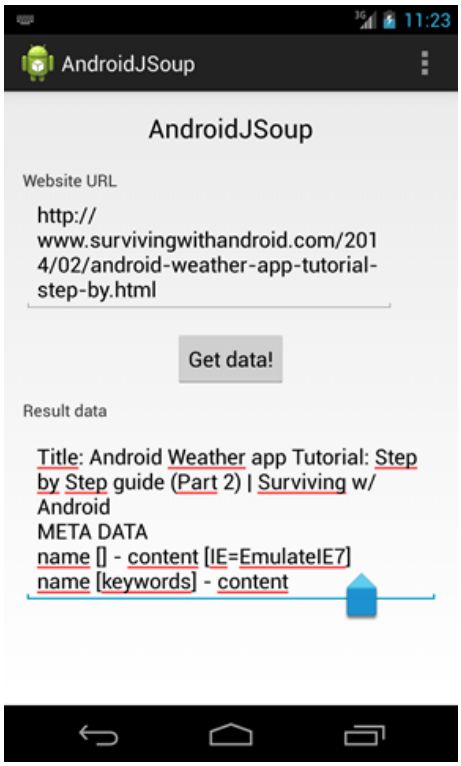
Potom máme tedy:

```
dependencies {compile fileTree(dir: 'libs', include: ['*.jar'])
      compile 'com.android.support:appcompat-v7:19.+'
      compile 'org.jsoup:jsoup:1.7.3' }
```

Nyní máme vše připraveno k použití JSoup API.

8.3 Vytvoření Aplikace: Parsování HTML stránky

Jakmile budeme znát některé základní informace o JSoup API, můžeme začít kódování naši aplikaci. Na konci získáme:



Obrázek 29: Náhled aplikace

Jako první věc je mít na paměti, že voláme vzdálené webové stránky, takže nemůžeme použít naše JSoup API v hlavním vlákně jinak by mohla vyskočit ANR chyba(Aplikace nereaguje), takže v tomto příkladu budeme používat AsyncTask. Jak můžete vidět rozložení je velmi jednoduché: do pole EditText vložíme URL, po zmáčknutí tlačítka "Get data"se HTML začne parsovat a v dalším poli EditText ukáže výsledky. V hlavní aktivitě máme:[10]

```
@Override
protected void onCreate(Bundle savedInstanceState) {
     ...
final EditText edtUrl = (EditText) findViewById(R.id.edtURL);
   Button btnGo = (Button) findViewById(R.id.btnGo);
   respText = (EditText) findViewById(R.id.edtResp);
  btnGo.setOnClickListener(new View.OnClickListener() {
```

```
@Override
public void onClick(View view) {
       String siteUrl = edtUrl.getText().toString();
      (new ParseURL()).execute(new String[]{siteUrl});
}});}
```

kde ParseURL je třída, která má na starosti parsování souboru HTML pomocí JSoup. V této třídě, máme:

```
1. private class ParseURL extends AsyncTask<String, Void, String> {
2. @Override
3. protected String doInBackground(String... strings){
4.         StringBuffer buffer = new StringBuffer();
5.         try {
6.            Log.d("JSwa", "Connecting to ["+strings[0]+"]");
7.            Document doc  = Jsoup.connect(strings[0]).get();
8.            Log.d("JSwa", "Connected to ["+strings[0]+"]");
9. //Vrátí titulek HTML stránky
10. String title = doc.title();
11. Log.d("JSwA", "Title ["+title+"]");
12. buffer.append("Title: " + title + "\r\n");
```

```
//Vrátí info meta tagu
13. Elements metaElems = doc.select("meta");
14. buffer.append("META DATA\r\n");
15. for (Element metaElem : metaElems){
16.     String name = metaElem.attr("name");
17.     String content = metaElem.attr("content");
18.     buffer.append("name ["+name+"] - content
19.                   ["+content+"] \r\n");}
20. Elements topicList = doc.select("h2.topic");
21.    buffer.append("Topic list\r\n");
22. for (Element topic : topicList) {
23.    String data = topic.text();
24.   buffer.append("Data ["+data+"] \r\n");
25. }}
26. catch(Throwable t)
27. {t.printStackTrace();}
28.   return buffer.toString();}
29.   ...
30. @Override
31. protected void onPostExecute(String s) {super.onPostExecute(s);
32. respText.setText(s);}
```

Analýza této třídy: Na řádku 8 jsme se připojili ke vzdálenému url a získali DOM reprezentaci stránky HTML. Poté na řádku 11 načítáme titulek stránky. Pomocí informací o JSoup uvedených výše začneme výběrem meta tagů pomocí metody select (řádek 13). Víme, že existuje několik atributů meta tagů, takže načítáme atributy název a obsah (řádek 16 až 17). V poslední části vybíráme textový obsah všech h2 tagů, které mají na třídu ”topic”.[10]

8.4 JSoup a Volley

Ve výše uvedeném příkladu jsme použili AsyncTask běžící na pozadí, ale pokud dáváte přednost volley, můžete ho použít také. V tomto případě budeme parsovat dokument, který nepoužívá URL, ale String obsahující načtený doc:

```
StringRequest req = new StringRequest(Request.Method.GET, url,
new Response.Listener<String>(){
@Override
public void onResponse(String data){
Document doc = Jsoup.parse(data);
....
}},
new Response.ErrorListener(){
@Override
public void onErrorResponse(VolleyError volleyError
{// Handle error }
});
```

9 HW čidla a práce s nimi

Zařízení s operačním systémem Android májí spoustu hardwarových čidel. V této kapitole si řekneme jak naprogramovat ukázkové aplikace s využitím některých těchto čidel.

9.1 Accelerometer

Jako první si vytvoříme aplikaci využívající G-senzor daného zařízení. Založíme nový projekt s názvem AccelerometerDemo a hlavní aktivitu pojmenujeme MainActvity.

MainActivity.java:[11]

Musíme nejprve naší třídě implementovat SensorEventListener:

```
public class MainActivity extends Activity implements
  SensorEventListener{}
```

Nyní vytvoříme objekt pro SensorManager a Sensor v rámci naší MainActivity třídy. Deklarujeme TextViews a RelativeLayout, které použijeme pro zobrazování hodnot akcelerometru.[11]

```
private SensorManager mSensorManager;
private Sensor mAccelerometer;
TextView title,tv,tv1,tv2;
RelativeLayout layout;
```

Napíšeme metodu OnCreate, který definuje výchozí bod naší aktivity:[11]

```
@Override
public final void onCreate(Bundle savedInstanceState){
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   //získání služby senzoru
   mSensorManager =
  (SensorManager)getSystemService(Context.SENSOR_SERVICE);
```

```
   //získání čidla akcelerometru
   mAccelerometer =
   mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
  //získání vrstvy
  layout=(RelativeLayout)findViewById(R.id.relative);
  //získaní textviews
  title=(TextView)findViewById(R.id.name);
  tv=(TextView)findViewById(R.id.xval);
  tv1=(TextView)findViewById(R.id.yval);
  tv2=(TextView)findViewById(R.id.zval);
```

Nyní musíme napsat dvě metody, které jsou spojeny s SensorEventListener jmenovitě onAccuracyChanged a onSensorChanged[11]

```
@Override
public final void onAccuracyChanged(Sensor sensor, int accuracy){
// Dělej něco, když se změní hodnoty snímače}
@Override
public final void onSensorChanged(SensorEvent event){
// 3 hodnoty, jeden pro každou osu
   float x = event.values[0];
   float y = event.values[1];
   float z = event.values[2];
//zobrazení hodnoty pomocí TextView
   title.setText(R.string.app_name);
   tv.setText("X axis" +"\t\t"+x);
   tv1.setText("Y axis" + "\t\t" +y);
   tv2.setText("Z axis" +"\t\t" +z);}
```

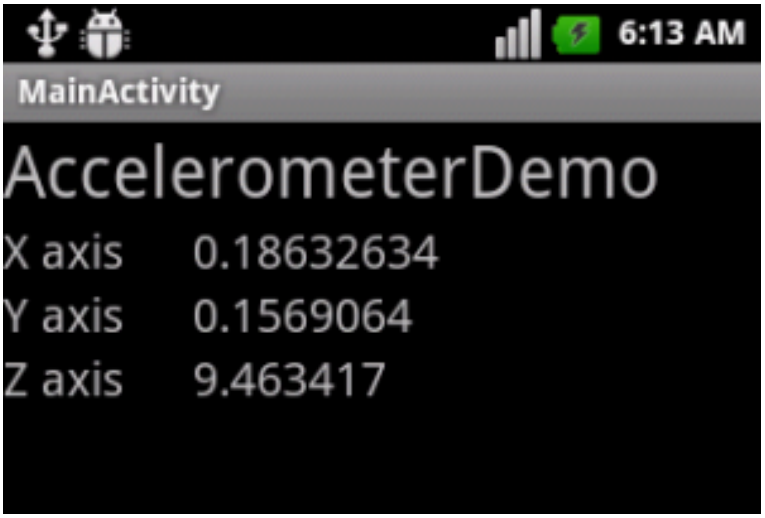
Poté je důležité registrovat a zrušit registraci SensorListener v metodách životního cyklu aktivity onResume a onPause:[11]

```
@Override
protected void onResume(){
super.onResume();
mSensorManager.registerListener(this, mAccelerometer,
SensorManager.SENSOR_DELAY_NORMAL);}
@Override
protected void onPause(){
super.onPause();
mSensorManager.unregisterListener(this);}
```

activity_main.xml:

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:id="@+id/relative">
<TextView
android:textSize="30dp"
android:id="@+id/name"
android:layout_width="fill_parent"
android:layout_height="wrap_content" />
<TextView
android:textSize="20dp"
android:layout_below="@+id/name"
android:id="@+id/xval"
android:layout_width="fill_parent"
android:layout_height="wrap_content" />
<TextView
android:textSize="20dp"
android:id="@+id/yval"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_below="@+id/xval" />
<TextView
android:textSize="20dp"
android:id="@+id/zval"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_below="@+id/yval" />
</RelativeLayout>
```

Spustíme aplikaci a měli bychom vidět tuto obrazovku:



Obrázek 30: Náhled hotové aplikace

9.2 Svítilna

V druhém příkladě si vytvoříme aplikaci využívající blesk fotoaparátu daného zařízení. Aplikace bude mít pouze jedno tlačítko ve středu vrstvy, které bude použito pro zapnutí a vypnutí blesku fotoaparátu. Když stisknete tlačítko "Torch-on", náš program zapne blesk a text tlačítka se změní na "Torch-OFF". Podobně při opětovném stisknutí tlačítka, vypne blesk a změní tlačítko text "Torch-ON". Založíme nový projekt s názvem FlashlightExample a hlavní aktivitu pojmenujeme MainActvity.[12]

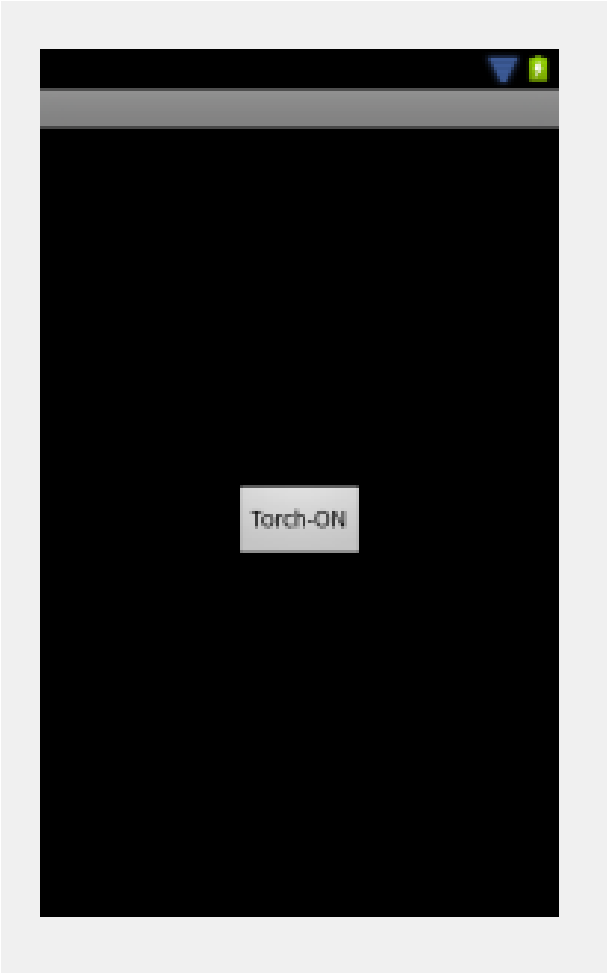
Vytvoření vrstvy:

layout.xml pro tuto aplikaci by měl vypadat následovně:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/relativeLayout1"
android:layout_width="fill_parent"
```

```
android:layout_height="fill_parent">
<Button
android:id="@+id/buttonFlashlight"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerVertical="true"
android:layout_centerHorizontal="true"
android:text="Torch-ON"/>
</RelativeLayout>
```

Vrstva aplikace by měla tedy vypadat:



Obrázek 31: layout.xml

Vytvoříme následující objekty a proměnné přímo ve třídě MainActivity.java:[12]

```
//Nastavení, zda je blesk zapnu/vypnut
private boolean isFlashOn = false;
//Vytvoření objektu kamera
private Camera camera;
//Tlačítko blesku private Button button;
```

Do metody onCreate() napíšeme:[12]

```
//Nalezení tlačitka
button = (Button) findViewById(R.id.buttonFlashlight);
Context context = this;
// Načtenít balíčků, které jsou aktuálně nainstalované
// na zařízení, jehož součástí je kamera, GPS atd.
PackageManager pm = context.getPackageManager();
```

Měli bychom zkontrolovat, zda zařízení má fotoaparát, než budeme pokračovat s kódováním. Následující fragment kódu zkontroluje, zda je kamera instalována ve vašem přístroji nebo ne:[12]

```
if(!pm.hasSystemFeature(PackageManager.FEATURE_CAMERA)){
Logger message Log.e("err", "Zařízení nemá kameru!");
//Chybová hláška, že zařízení nemá kameru
Toast.makeText(getApplicationContext(),
"Zařízení nemá kameru!",Toast.LENGTH_SHORT).show() return; }
```

Vytvoříme nový objekt Camera pro přístup k fotoaparátu na zařízení:[12]

```
 camera = Camera.open();
```

Získání aktuálního nastavení pro služby Camera:

```
 final Parameters p = camera.getParameters();
```

Vytvoříme posluchač pro tlačítko, které jsme vytvořili:[12]

```
button.setOnClickListener(new OnClickListener(){
public void onClick(View arg0) { } });
```

Do metody onClick() napíšeme:[12]

```
//Když blesk je zapnut if (isFlashOn){
Log.i("info", blesk je zapnut!");
//Vypnout blesk
p.setFlashMode(Parameters.FLASH_MODE_OFF);
//Předání parametru objektu camera camera.setParameters(p);
//Vypnout blesk
isFlashOn = false;
//nastavení textu tlačítka na Torch-ON
button.setText("Torch-ON"); }
//Když je blesk vypnut else
{ Log.i("info", "blesk vypnut!");
//Zapnout blesk
p.setFlashMode(Parameters.FLASH_MODE_TORCH);
//Předání parametru objektu camera
camera.setParameters(p);
//Zapnout blesk
isFlashOn = true;
//nastavení textu tlačítka na Torcn-OFF
button.setText("Torch-OFF"); }
```

Je vždy dobré uvolnit prostředky, které drží aplikace, pokud to není potřeba:[12]

```
@Override
protected void onStop(){
super.onStop();
if (camera != null){
camera.release(); } }
```

Musíme ještě přidat oprávnění v AndroidManifest.xml:[12]

```
<uses-permission
android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera" />
```

10 Aplikace Menzy JU

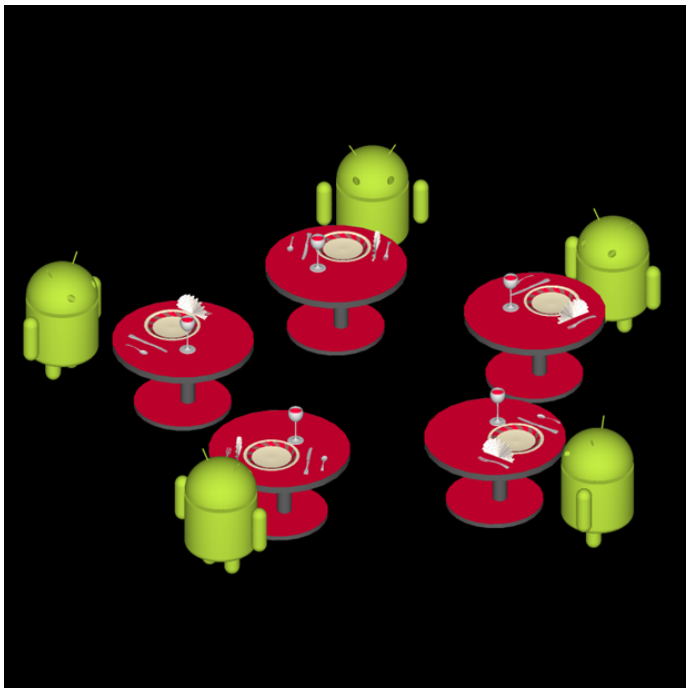
Hlavním cílem mé bakalářské práce bylo vytvořit mobilní aplikaci pro jídelny Jihočeské univerzity v Českých Budějovicích. Aplikace by měla sloužit k zobrazování aktuálního jídelníčku a dále by měla umožnit uživatelům objednání si jídla ze systému iCanteen skrz uživatelské rozhraní aplikace.

10.1 Možnosti přístupu k informacím o jídelnách

V tuto chvíli mají strávníci se zařízením s operačním systémem Android OS přístup k informacím o aktuálním přehledu jídel pouze přes webovou stránku, která není momentálně optimalizovaná pro mobilní zařízení. Prohlížení a objednávání jídel na stránkách přes webový prohlížeč je tak zbytečně složité a uživatel musí stále přibližovat stránku.

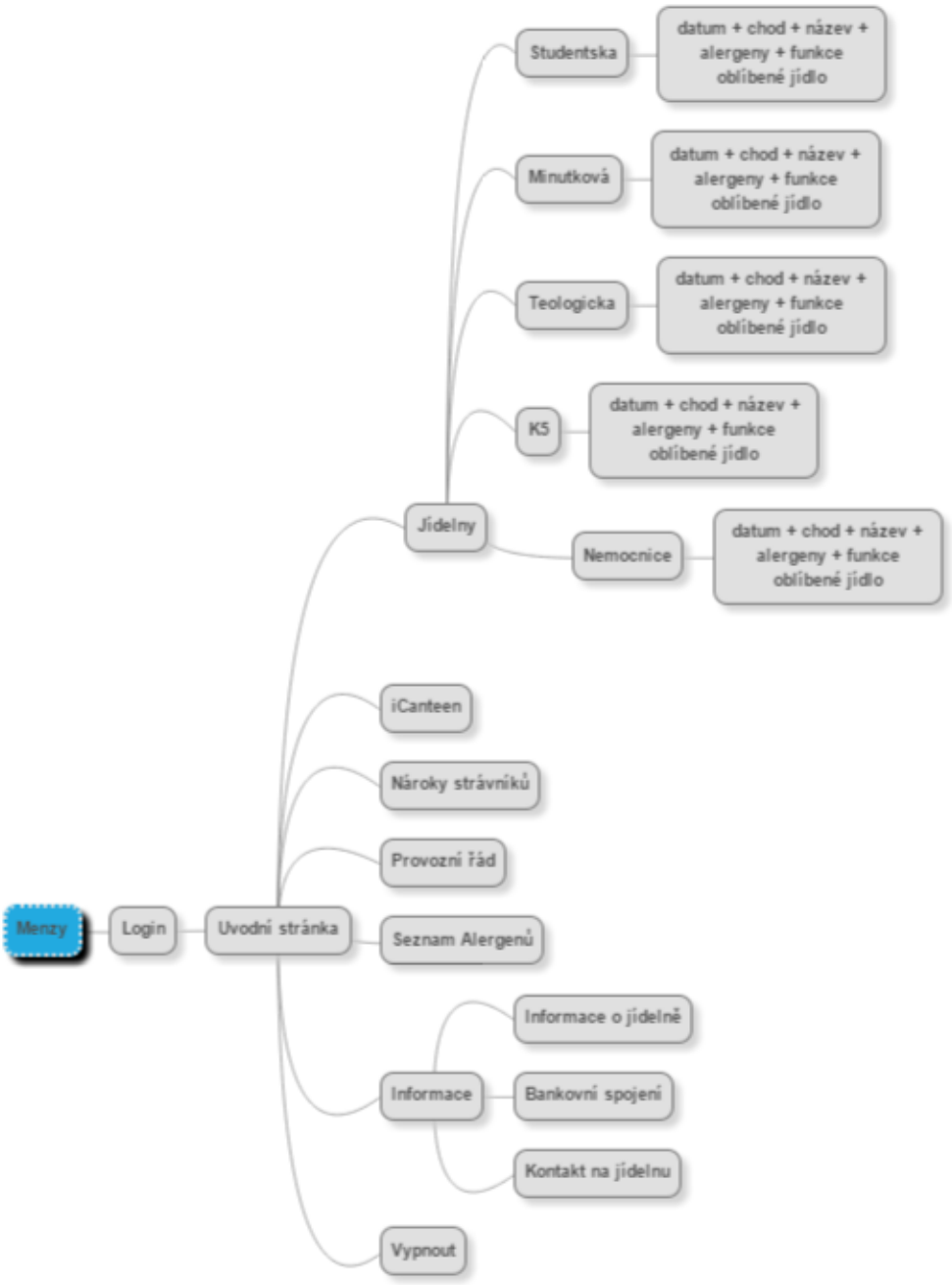
10.2 Návrh

Cílem aplikace je poskytnout uživatelům co nejpohodlnější a nejrychlejší přístup k požadovaným informacím. Z tohoto důvodu bylo rozhraní aplikace navrženo tak, aby uživatel nemusel stále přibližovat a aplikace se přizpůsobila displeji daného zařízení. Dalším cílem bylo vytvořit aplikaci tak, aby se v ní uživatel dokázal rychle zorientovat a co nejmenším počtem kliknutí se dostal k požadované informaci. Jako první věc jsem vytvořil logo aplikace. Toto logo je vytvořeno z loga Jihočeské univerzity, což je pět červených kruhů. Z těchto kruhů jsem vytvořil jídelní stoly, na které jsem vložil talíř s příbory a ke stolům postavil Android panáčky.



Obrázek 32: Logo

Dále jsem si rozvrhnul sekce aplikace pomocí myšlenkové mapy, kterou si můžete prohlédnout níže.



Obrázek 33: Myšlenková mapa

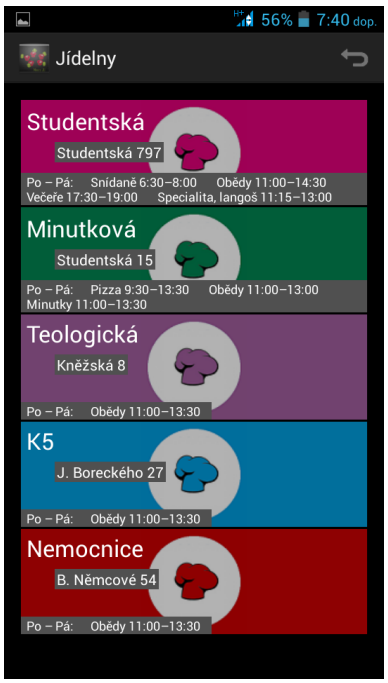
Úvodní obrazovka aplikace s menu vypadá následovně:

Obrázek 34: Úvodní obrazovka

10.3 Sekce

10.3.1 Jídelny

Sekce jídelny je jedna z nejdůležitějších v této aplikaci. Obsahuje seznam jídelen, jejich adresy a otevírací doby. Jídelny jsou od sebe barevně odlišeny pro dobrou a rychlou orientaci.



Obrázek 35: Sekce Jídelny

Po kliknutí na jednu z jídelen se nám zobrazí tabulka s aktuálním jídelníčkem dané jídelny. V tabulce jsou čtyři sloupce: Datum, Chod, Název a Obsahuje alergeny. Nad posledním sloupcem je tlačítko s názvem Alergeny, které odkazuje na stránku s legendou k jednotlivým alergenům. Dále zde máme tři tlačítka na horním panelu. Tlačítko zpět, které nás vrátí o úroveň výš v aplikaci. Tlačítko aktualizovat, které nám znovu načte aktuální data. A tlačítko oblíbené, které nám zvýrazní v tabulce naše uložené oblíbené jídlo. Jídlo uložíme jako oblíbené jednoduše kliknutím na něj, posléze se objeví dialogové okno, jestli chcete opravdu jídlo uložit jako oblíbené.

Datum	Chod	Název	Obsahuje alergeny
23.6.2015 út	Polévka 1	Polévka hrstková	1, 6, 9, 12
-	Minutka 1	Kuřecí nudličky po čínsku, hranolky smažené, obloha	1
-	Minutka 2	Kuřecí nugety, hranolky smažené, obloha	1, 7, 9
-			
-	Polévka 1	Polévka rajčatová s rýží	1, 6, 7, 9, 12
-	Minutka 1	Kuřecí plátek na bazalce, hranolky smažené, obloha	1
-	Minutka 2	Kuřecí stripsy se sezamem, hranolky smažené, obloha	1, 6, 7, 11
-	-	-	-
-	-	-	-
Středa	Minutka 1	Kuřecí plátek na žampionech, hranolky smažené, obloha	1

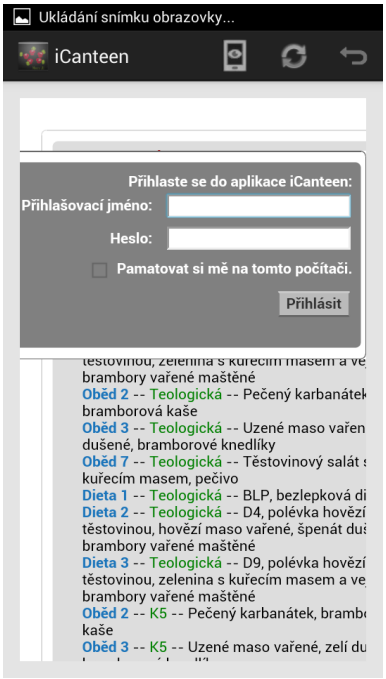
Obrázek 36: Sekce Minutková

Aktuální jídelníček v této sekci se načítá ze stránky http://menza.jcu.cz/Minutkova.html pomocí parsovací knihovny Jsoup.

10.3.2 iCanteen

Další sekcí je sekce iCanteen. V této sekci se můžeme přihlásit do objednávkového systému iCanteen a plně využívat jeho funkce. Tato sekce funguje jako internetový prohlížeč, avšak s tou výhodou, že můžeme využít tlačítko přizpůsobit na horním panelu aplikace, které nám přizpůsobí stránku pro daný displej zařízení bez nutnosti přibližování.

Obrázek 37: Přihlášení s nepřizpůsobenou velikostí



Obrázek 38: Přihlášení s přizpůsobenou velikostí

Tohoto efektu jsem docílil pomocí nastavení WebSettings daného webView.

```
if (item.getItemId() == R.id.action_screen){
this.webView = (WebView) findViewById(R.id.webview);
WebSettings webSettings = webView.getSettings();
webSettings.setJavaScriptEnabled(true);
webSettings.setDisplayZoomControls(false);
webSettings.setBuiltInZoomControls(false);
webSettings.setLoadWithOverviewMode(false);
webSettings.setUseWideViewPort(false);
WebViewClientImpl webViewClient = new WebViewClientImpl(this);
webView.setWebViewClient(webViewClient);
return true;
}
```

Další věc, kterou jsem musel nastavit bylo to, aby po přihlášení se stránka dále zobrazovala v rozhraní aplikace a nevyskakovala do defaultního prohlížeče zařízení. Toho jsem docílil pomocí vnitřní třídy:

```
private class A extends WebViewClient{
 public boolean shouldOverrideUrlLoading(WebView view, String url){
 view.loadUrl(url);
 return true;
 }
}
```

Další výhodou oproti webovému prohlížeči je to, že zůstaneme přihlášeni v systému iCanteen až do systémem nastavené doby expirace. Toho jsem dosáhl tak, že při spuštění této aktivity odesílám požadavek rovnou jako tlačítko přihlášení. Tedy:

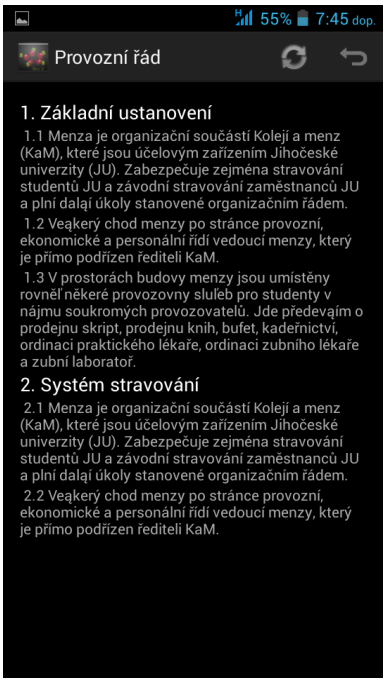
```
protected Void doInBackground(Void... voids){
webView.loadUrl("https://menza.jcu.cz/faces/secured/main.jsp?
terminal=false&amp;status=true&amp;printer=false&amp;keyboard=false";
return null;
}
```

Obrázek 39: Sekce iCanteen

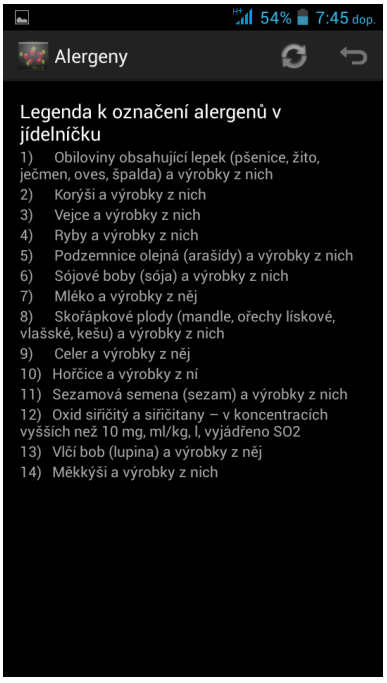
10.3.3 Nároky strávníku, Provozní řád, Alergeny, Informace

Všechny tyto sekce jsou parsovány také pomocí knihovny Jsoup.

Obrázek 40: Nároky Strávníků



Obrázek 41: Provozní řád



Obrázek 42: Alergeny

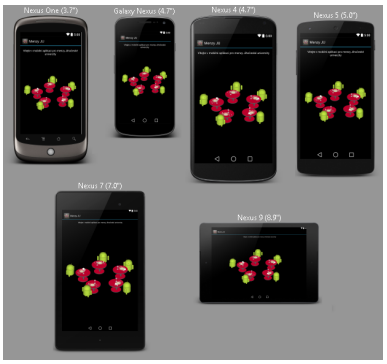
Obrázek 43: Informace o jídelně

10.4 Testování aplikace

V momentu, kdy je aplikace hotová, je potřeba ji před publikováním na obchod Google Play důkladně otestovat za všech podmínek a na všech zařízeních s různým rozlišením displeje.

10.4.1 Emulátor

Díky Android Studiu a jeho integrovanému systému emulátorů je práce velmi snadná. V tomto vývojovém prostředí jsou přednastavená zařízení, na kterých můžeme aplikaci testovat. Dají se však vytvořit i vlastní emulátory, kde můžeme nastavit vše od rozlišení displeje, velikost displeje přes verzi operačního systému až po hardwarové vybavení zařízení.



Obrázek 44: Emulátory

10.4.2 Fyzické zařízení

Testovat aplikaci bychom měli určitě také na fyzickém zařízení. Tato možnost je velice jednoduchá. Stačí pouze připojit zařízení přes USB kabel do svého počítače a v zařízení povolit instalaci aplikací z neznámých zdrojů. Poté již můžeme směle testovat.

10.4.3 Testování aplikace Menzy JU

Aplikace Menzy JU byla testována na mobilním zařízení ZOPO ZP780, Samsung S3 mini a na tabletu s uhlopříčkou 8 palců. Aplikace se na všech zařízeních zobrazovala a chovala správně.

10.5 Publikování aplikace

Nyní si řekneme jak publikovat aplikace na Google Play:

- Musíme si založit účet vývojáře [13]
- Nastavit obchodní účet pro Google platby, pokud budeme prodávat aplikace [13]
- Prozkoumání vývojářské konzole [13]

10.5.1 Založení vývojářského účtu

1. Navštívíme stránku https://play.google.com/apps/publish/signup/ [13]
2. Zadáme základní informace o své identitě vývojáře - jméno, e-mailovou adresu, a tak dále. Tyto informace můžete později upravit [13]
3. Přečteme si a přijmeme distribuční smlouvu pro vývojáře pro vaši zemi či region. Všimněte si, že aplikace, které chcete publikovat na Google Play, musí být v souladu se zákonem amerického exportu [13]
4. Musíme zaplatit 25 USD jako registrační poplatek za použití plateb Google [13]
5. Pokud je vaše registrace ověřena, budete upozorněni na e-mailovou adresu, kterou jste zadali při registraci [13]

10.5.2 Nastavení obchodního účtu pro Google platby

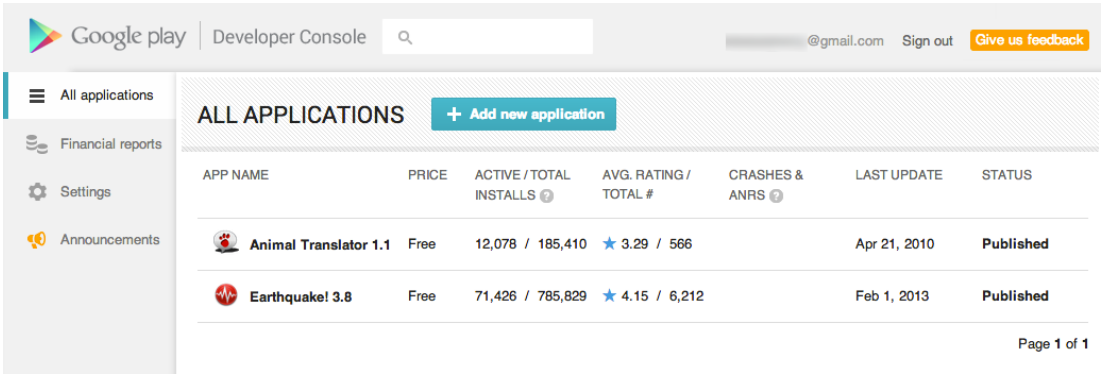
Pokud chcete prodávat placené aplikace, budete potřebovat obchodní účet pro Google platby. [13]

1. Přihlaste se do své vývojářské konzole Google Play na https://play.google.com/apps/publish/ [13]
2. Otevřete Financial reports vlevo v navigaci [13]
3. Klikněte na Setup a Merchant Account now [13]

Toto Vás zavede na platební stránky Google [13]

10.5.3 Prozkoumání vývojářské konzoly

Pokud je vaše registrace ověřena, můžete se přihlásit ke své vývojářské konzoli, která je domovem pro vaše publikované aplikace a nástrojů na Google Play. [13]



Obrázek 45: Zdroj: http://developer.android.com/distribute/googleplay/start.html

11 Závěr

Během psaní bakalářské práce jsem prostudoval veškeré dostupné informace o vývoji aplikací pro operační systém Android OS. Donedávna bylo základním, doporučeným způsobem vývoje Android aplikací prostředí Eclipse s pluginem Android Developer Tools. Jeho vývoj byl však ukončen a přechod na Android Studio je tak důrazně doporučován. Celé Android Studio je zdarma a má výbornou podporu vývojářů tohoto prostředí, tudíž je velmi vyhledávané mezi programátory aplikací pro Android OS.

Jako praktickou část práce jsem vytvořil aplikaci Menzy JU, která slouží k prohlížení aktuálního jídelního lístku z jídelen Jihočeské univerzity. Aplikace také umožňuje objednávání jídel, práce s burzou a všechny ostatní funkce systému iCanteen skrz uživatelské rozhraní aplikace.

Dále jsem vytvořil sadu ukázkových aplikací využívajících hardwarové senzory zařízení s operačním systemém Android OS. Jedna z těchto aplikací je aplikace jménem Svítilna, která využívá blesk fotoaparátu jako běžnou svítilnu. Druhá tato aplikace se jmenuje G-Senzor. Tato aplikace využívá gravitační čip daného zařízení a umožnuje tak pracovat s hodnotamy tří os našeho prostoru. Základy obou těchto aplikací jsou detailně popsány v teorické části práce. Všechny vytvořené aplikace jsou k dispozici ke stažení v obchodu Google Play pro všechny zařízení splňující požadavky dané aplikace.

Seznam použité literatury a zdrojů

[1] UJBÁNYAI, Miroslav. *Programujeme pro Android*. Vyd. 1. Praha: Grada, 2012. ISBN 978-80-247-3995-3.

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

[3] Svetandroida [online]. 2010 [cit. 2013-03-10]. Dostupné z: http://www.svetandroida.cz/

[4] *Vyvíjíme pro Android - Zdroják* [online]. 2015 [cit. 2015-06-16]. Dostupné z: http://www.zdrojak.cz/serialy/vyvijime-pro-android/

[5] KYPTA,Tomáš. Vyvíjíme pro Android. Vyvíjíme pro Android [online]. 2011, č. 2011 [cit. 2013-03-10]. Dostupné z: http://www.svetandroida.cz/vyvijime-pro-android-1-uvod-201103

[6] Android Developers [online]. 2014 [cit. 2015-06-22]. Dostupné z: https://developer.android.com/

[7] Managing Projects Overview | Android Developers. Android Developers [online]. 2014 [cit. 2015-06-22]. Dostupné z: https://developer.android.com/tools/projects/index.html

[8] Starting an Activity | Android Developers. Android Developers [online]. 2014 [cit. 2015-06-22]. Dostupné z: https://stuff.mit.edu/afs/sipb/project/android/docs/training/basics/activity-lifecycle/starting.html

[9] EBookFrenzy.com - Quality, affordable technology eBooks [online]. 2015 [cit. 2015-06-22]. Dostupné z: http://ebookfrenzy.com/

[10] Surviving with android [online]. 2015 [cit. 2015-06-22]. Dostupné z: http://www.survivingwithandroid.com/

[11] Tutorial 1: Android Accelerometer Demo. THREADS OF LIFE [online]. 2014 [cit. 2015-06-22]. Dostupné z: http://karanbalkar.com/2012/09/tutorial-1-android-accelerometer-demo/

[12] How to turn on/off only camera flashlight - simple torch example. Programmerguru [online]. 2014 [cit. 2015-06-22]. Dostupné z: http://programmerguru.com/android-tutorial/android-flashlight-example/

[13] Get Started with Publishing. *Android Developers* [online]. 2015 [cit. 2015-06-23]. Dostupné z: http://developer.android.com/distribute/googleplay/start.html

Seznam obrázků

Přílohy

1. CD – na přiloženém CD se nachází plné znění bakalářské práce pod názvem souboru `urmann_bakalarska-prace.pdf`, dále jsou zde instalační soubory vytvořených aplikací a jejich zdrojové kódy.