OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA
KATEDRA INFORMATIKY A POČÍTAČŮ

Tvorba mobilní aplikace

SEMESTRÁLNÍ PRÁCE

Autor: Jan Sonnek Ostrava, 2021

Obsah

1 ANDROID

Android je operační systém, který vytvořila společnost Google. Na rozdíl od hlavního konkurenta Applu, je Android open-source, což znamená otevřený operační systém se zdrojovým kódem. Každý vývojář jej může používat a upravovat dle vlastních představ, ačkoliv musí dodržovat určitá pravidla. OS je založen na linuxovém jádře různých verzí, které zajišťuje zabezpečení systému jako celku, správu paměti, správu procesů, přístup k síti a ovladačům všech vnitřních senzorů a komponentů. Android je nejmladší a nejrychleji se rozvíjející multiplatformní operační systém primárně vyvíjen jako platforma převážně pro PDA, chytré mobily (smartphony) a tablety. Kromě zmíněných zařízení se s ním setkáme v chytrých hodinkách (smartwatch), televizích (Android TV) nebo v autě (Android Auto).(10)

Jak již bylo zmíněno, Android je multiplatformní OS, což znamená, že systém může běžet na zařízeních různých značek (Samsung, Lenovo, Huawei, Honor, atd). Podpora několika značek má i své nevýhody. Mezi největší nevýhodou je optimalizace systému na konkrétní určitou platformu (aktualizace nové verze Androidu). V optimalizaci systému vítězí Apple iOS. Proto, když se zeptáte lidí, proč preferují iOS než Android OS, bývá jejich odpověď většinou právě okamžitá aktualizace OS.(10)

1.1 Historie OS

I když OS patří společnosti Google, za jejím vznikem nestojí. Jejím zakladatelem je společnost Android Inc., která vznikla v říjnu roku 2003 ve městě Palo Alto, ve státě Kalifornia. O dva roky později Google odkoupil Android Inc. Odhadovaná částka, kterou Google za Android zaplatil činí 50 milionů dolarů. V listopadu 2007 byla založena Open Handset Aliance, která stojí za vývojem Androidu dodnes. Open Handset Aliance je sdružení více než 80 firem, mezi které patří Intel, HTC, LG, NVIDIA, Qualcomm, Google nebo Samsung, a to v oblasti mobilních operátorů, softwarových společností, společností vyrábějících polovodičové součástky nebo výrobci mobilních zařízení. V té době také vychází vývojářský Kit (SDK). V září 2008 se na trhu v USA objevil mobil HTC Dream (G1) s Androidem 1.0. V únoru 2009 přišel Android 1.1 jako aktualizace pro G1. V dubnu 2009 spatřil světlo světla první masový Android s verzí 1.5 (Cupcak). Odstartoval tak popularitu tohoto systému.(10)

Obrázek 1: Loga verzí OS

Přehled verzí Android			
Název	Verze	Rok vydání	API
Android Apple Pie	1.0	2008	1
Android Banana Bread	1.1	2009	2
Android Cupcake	1.5	2009	3
Android Donut	1.6	2009	4
Android Eclair	2.0	2009	5
Android Eclair	2.0.1	2009	6
Android Eclair	2.1	2010	7
Android Froyo	2.2, 2.2.1 – 2.2.3	2010	8
Android Gingerbread	2.3, 2.3.1	2010	9
Android GInger-bread	2.3.2 – 2.3.7	2011	10
Android Honeycomb	3.0, 3.1, 3.2	2011	11
Android Honeycomb	3.2.1, 3.2.2, 3.2.3	2011	12
Android Honeycomb	3.2.6	2012	13
Android Ice Cream Sandwich	4.0	2011	14
Android Ice Cream Sandwich	4.0.1, 4.0.2, 4.0.3, 4.0.4	2012	15
Android Jelly Bean	4.1	2012	16
Android Jelly Bean	4.1.1, 4.1.2	2012	17
Android Jelly Bean	4.2, 4.2.1 , 4.2.2	2013	18
Android KitKat	4.4	2013	19

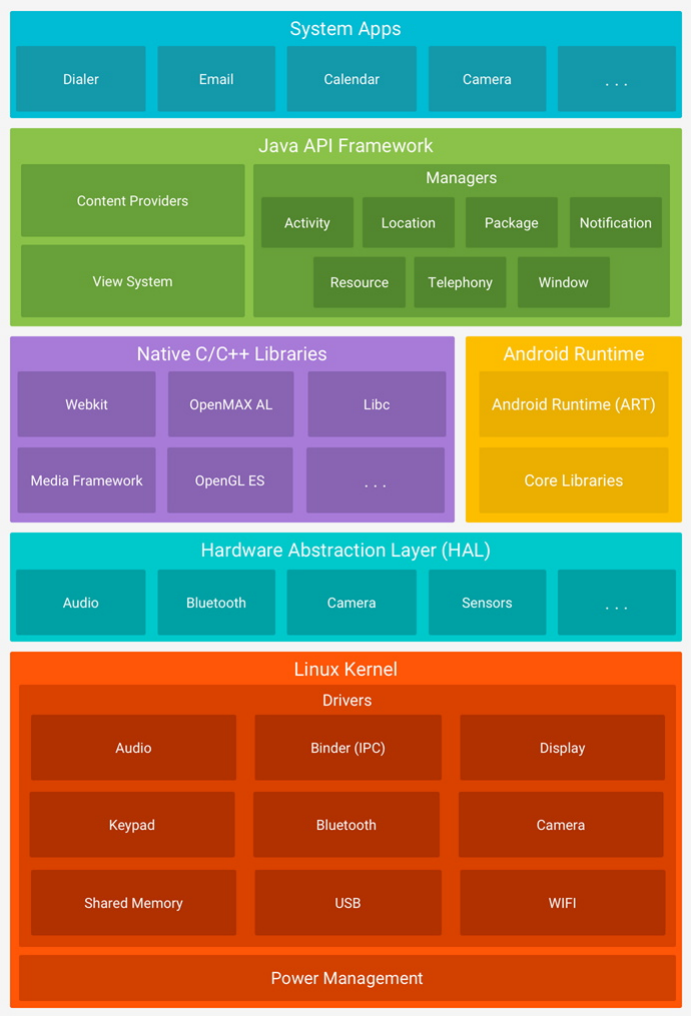
Android Kitkat	4.4W2	2014	20
Android Lollipop	5.0	2014	21
Android Lollipop	5.1.1	2015	22
Android Marshmallow	6.0	2015	23
Android Nougat	7.0	2016	24
Android Nougat	7.1.1	2016	25
Android Oreo	8.0	2017	26
Android Oreo	8.1	2017	27
Android Pie	9.0	2018	28
Android 10	10.0	2019	29

Tabulka 1: Vlastní tvorba verzí OS Android

1.2 Architektura Androidu

Celá architektura operačního systému se skládá z pěti vrstev, které jsou znázorněny na obrázku č. 2.

- Linux Kernel
- Hardware Abstraction Layer
- Nativní knihovny (NDK)
- Android Runtime – ART
- Java API Framework



Obrázek 2: Architektura OS Android

1.2.1 Linux Kernel

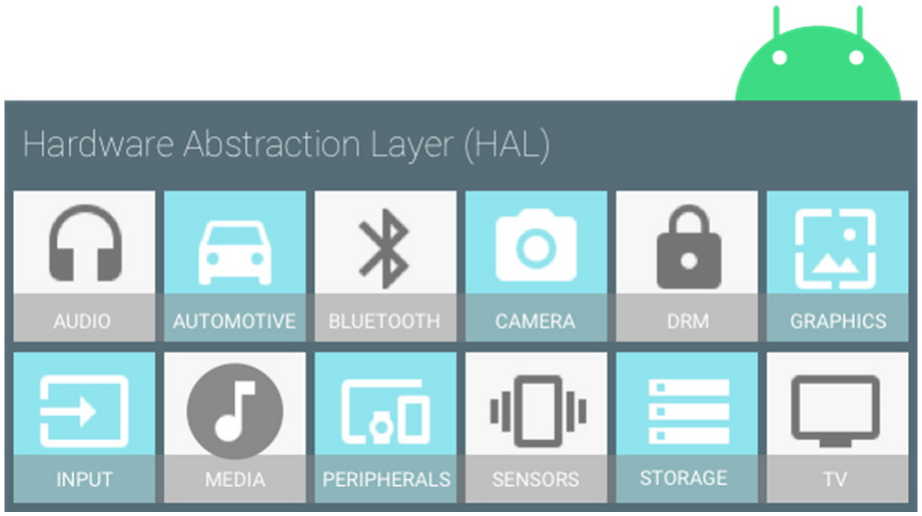
Na nejnižší vrstvě Androidu se nachází jádro neboli Linux Kernel a patří mezi nejzákladnější komponenty systému Android. „*Základní funkcí je implementace abstrakce mezi použitým hardwarem a softwarem ve vyšších vrstvách. Zabezpečuje správu paměti, správu procesů, základní síťovou vrstvou a ovladače. Řízení procesů umožňuje, aby více procesů běželo současně, aniž by se vzájemně ovlivňovaly.*“[1] *Mistrovství Android – str. 75*(10)

1.2.2 Hardware Abstraction Layer

Vzhledem k odchylnosti hardwarových zařízení mobilních telefonů či tabletů, jako je např. procesor nebo fotoaparát, na nichž může Android běžet, byla vytvořena vrstva Hardware Abstraction Layer (vrstva abstrahující hardware), zkráceně HAL. HAL vrstva se nachází nad úrovní Linux Kernel a vytváří rozhraní pro komunikaci s vyššími vrstvami systému s hardwarem. Z vývojářského hlediska je HAL dobrý

[1] Mistrovství - Android. Brno: Computer Press, 2017. ISBN 978-80-251-4875-4.

v tom, že programátor nemusí přesně znát hardwarové specifikace všech možných zařízení.(10),(17)



Obrázek 3: HAL

1.2.3 Nativní knihovny

Jednou z dalších vrstev Androidu jsou nativní knihovny (NDK). Tvoří mezivrstvu mezi různými komponenty vyšších vrstev a linuxovým jádrem. NDK jsou napsané v C/C++. Při programování může vývojář použít již existující knihovny nebo si knihovnu jednoduše naprogramovat dle potřeby sám. Webkit, který je součástí NDK, je určen k renderování a zobrazení webových stránek. Na této úrovni jsou implementovány také knihovny médií a grafické 2D. Pro vykreslování 3D grafiky slouží knihovna OpenGL založená na OpenGL ES (OpenGLforEmbedded Systems). Systémová knihovna LibC je knihovna optimalizovaná pro mobilní zařízení, která obsahuje jen části, které jsou zapotřebí pro Android.(10),(17)

1.2.4 Android Runtime - ART

Aplikace pro Android, které jsou vytvořeny v Android Studio jsou sestaveny do bajtového mezikódu, označovaného DEX. Soubory DEX vznikly kompilací z klasických souborů CLASS (soubory, ve kterých se píše zdrojový kód v jazyku Java nebo Kotlin) a JAR (vytvořená Java aplikace). Jsou kompaktnější než klasické soubory CLASS. Když se následně aplikace načítá do zařízení, Android Runtime (ART) používá proces označovaný jako Ahead-Of-Time (AOT) Tento proces se používá na přeložení bajtového kódu do nativních instrukcí konkrétního procesoru v příslušném zařízení. Jednoduše řečeno, ART slouží pro přeložení zdrojového kódu do strojového kódu (1 a 0).(10),(17)

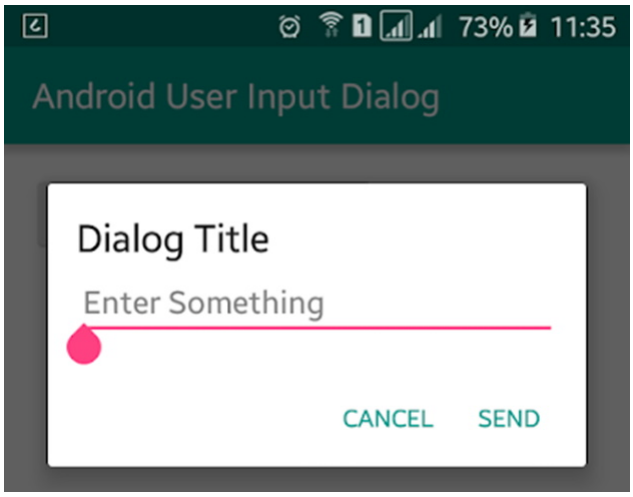
1.2.5 Java API Framework

Aplikační framework obsahuje v aplikacích opakovaně použitelný software, mezi které patří např. ovládací prvky, ikony atd. Tento framework je napsán v progra-

movacím jazyku Java a je to nejdůležitější vrstva pro vývojáře mobilních aplikací. Poskytuje aplikacím základní služby systému. Celý Java API Framework se skládá z:(17)

- Package Manager - Můžeme si package manager představit jako databázi. Tato databáze (package manager) má za úkol udržet všechny aktuální aplikace, které jsou na daném zařízení nainstalovány. Grafickým znázorněním package manageru není nic jiného než obrazovka zařízení, kde je možné vidět všechny nainstalované aplikace (Chrome, Obchod Play, YouTube, Gmail, Google, Disk, Fotky atd). Každá tato aplikace má svoji jedinečnou ikonu, která reprezentuje balíček aplikace.(17)

- Windows Manager - Windows Manager má na starosti správu oken, které tvoří mobilní aplikace. Aplikace většinou využívají dvě a více oken současně.



Obrázek 4: Windows Manager

Na obrázku 4 můžete vidět několik oken. Nejdůležitější je hlavní okno aplikace (Android User Input Dialog) a dialogové okno. Z obrázku můžeme tedy vidět, že aplikace má otevřené dvě okna současně. Za zmínku stojí i okna v horní části. V horní části je lišta, která zobrazuje různé ukazatele, jako např. Wi-Fi, stav baterie, mobilní signál, čas atd. O tuto lištu se nestará programátor, ale operační systém.(17)

- View System
  Spravuje všechny prvky grafického uživatelského rozhraní (UI), jako jsou tlačítka, editace textu, zobrazení textu, přepínače, ikony atd.(17),(18)

- Activity Manager – má na starosti správu životního cyklu aplikace (start, průběh, konec).

- Notification Manager, Resource Manager, Location Manager – poskytují pohodlný přístup k základním zdrojům. Tyto vrstvy jsou navrženy tak, aby jejich komponenty mohl uživatel snadno používat.(17)

1.3 Aplikace a její základní komponenty

Pojem aplikace je ve světě informatiky velmi široký pojem. Setkáváme se s tímto pojmem ve všech odvětví vývoje. Ať už se jedná o desktopové aplikace, webové aplikace nebo mobilní aplikace. Avšak nejvíce je pojem aplikace skloňován v souvislosti s mobilními zařízeními. U mobilních zařízení se pro označení aplikace používá zkratka „*app*app“ z anglického slova „*application*“ – aplikace.

1.3.1 Komponenty aplikace

Komponenty aplikace patří mezí stavební kameny každé Android aplikace. Každá část hraje svou významnou roli v systému, která slouží k odlišným účelům a má i odlišný životní cyklus. Mezi komponenty aplikace patří následující:

1. **Activity** – tvoří hlavní pilíř pro tvorbu grafického uživatelského rozhraní (GUI).
2. **Services** – komponenta, která umožňuje běh aplikace na pozadí tak, aby mohl uživatel vykonávat jinou činnost, např. si může pustit na mobilu hudbu, a přitom např. číst emaily. V takovém případě se přehrávač přepne do pozadí.
3. **Content Providers** – má za úkol poskytnout obsah s pracujícími daty jako je ukládání nebo načítání dat a zpřístupnit obsah dat pro všechny aplikace. Typickým příkladem, kde se Content Providers vyskytuje je správa dat z SQLite databáze.
4. **Broadcast receivers** – komponenta, která slouží k naslouchání ze zařízení nebo ze samotné aplikace. Stejně jako sevices nemá broadcast receivers žádné uživatelské rozhraní Pro komunikaci s uživatelem používají broadcast receivers pro upozornění stavový řádek. Příkladem této komponenty může být hlášení o nízkém stavu baterie.

2 VÝVOJOVÁ PROSTŘEDÍ, JAZYKY A NÁSTROJE PRO VÝVOJ

Vývojové prostředí, jinak známé pod zkratkou IDE (Integrated Development Enviroment – integrované vývojové prostředí) je programovací prostředí, které slouží programátorům pro tvorbu softwaru. Poskytují v jednom celku různé programátorské nástroje. Všechna IDE se skládají z grafického editoru, překladače, debuggeru, generátoru dokumentace, Version Control Systemu a uživatelského rozhraní.

- **Editor** – je nástrojem pro pořizování zdrojového kódu. Editor má schopnost zvýrazňovat syntaxi kódu a syntaktické chyby, provádět úpravy existujícího kódu nebo pomocí našeptávače generovat nové části kódu.
- **Překladač** – je program, který provádí zdrojový kód (kód v určitém programovacím jazyce – Java, C#, C/C++, Kotlin) do strojového kódu – kombinace 1 a 0.
- **Debugger** – slouží při programování k nalezení chyb.
- **Generátor dokumentace** – umožňuje generovat dokumentaci ze zdrojového kódu včetně komentářů.
- **Version Control Systém (VCS)** – je systém, který umožňuje jednotné uložiště zdrojového kódu na server. Umožňuje synchronizací zdrojového kódu všech spolupracujících programátorů.

V současnosti lze vyvíjet mobilní aplikace v několika vývojových prostředí. Mezi nejpoužívanější patří:

- **Android Studio** – je moderním vývojovým prostředím, které v roce 2013 sesadilo Eclipse IDE z piedestalu jako primární vývojové prostředí pro vývoj Android aplikací. V Android Studiu se programuje primárně v jazycích Java a Kotlin ve spolupráci s XML. Toto IDE je možné si zdarma stáhnout na adrese:`https://developer.android.com/studio/`.
- **Eclipse** – je jedno z nejpoužívanějších vývojových prostředí určeno primárně pro vývoj Java aplikací. Do roku 2013 bylo primární IDE pro vývoj Android aplikací, poté mu tuto výsadu sebralo Android Studio. Díky mnoha pluginů je možné v Eclipse programovat v C, C++, Python, Perl, PHP, HTML5, CSS, JavaScript, TypeScript aj. Toto IDE je možné si zdarma stáhnout na adrese:`https://www.eclipse.org/downloads/`.(3)
- **Netbeans** – e vývojové prostředí vytvořeno v jazyku Java primárně pro vývoj Java aplikací. Mezi velkou výhodu patří multijazyčnost tohoto prostředí. Kromě již zmíněné Javy, se zde může programovat v jazycích C, C++, PHP, HTML5 a Javascript. Dají se v něm i vyvíjet mobilní aplikace pro Android OS, avšak tato možnost se téměř vůbec nepoužívá. Toto IDE je možné si zdarma stáhnout na adrese:`https://netbeans.apache.org/download/index.html`.(12)

- **Xamarin Studio (Visual Studio)** – patří k vývojovému prostředí, ve kterém je možné programovat aplikace jak pro operační systém Android, tak pro iOS od Applu. V roce 2016 se stalo součástí Visual Studia od Microsoftu. V Xamarinu se při programování nepoužívá Java nebo Kotlin ale C#. Velkou výhodou pro tento typ vývoje je, že vývojář může napsat svoji aplikaci v jednom jazyce (C#) a přitom bude fungovat na Androidu a iOS. Toto IDE si můžete stáhnout na adrese:`https://visualstudio.microsoft.com/cs//`.

- **IntelliJ IDEA** – je dalším moderním vývojovým prostředím od společnosti JetBrains s.r.o., ve kterém lze vytvářet Android aplikace. Primárně je určeno pro vývoj v jazycích Java, Kotlin, Groovy, Scala aj. Toto IDE je možné zdarma stáhnout na adrese: `https://www.jetbrains.com/idea/download/`.

2.1 Android Studio

Jak již bylo zmíněno, Android Studio je oficiální, multiplatformní vývojové prostředí pro platformu Android OS. Je zcela zdarma a k dispozici pro Windows, Linux nebo MacOS. Kromě Android Studia je potřebný pro vývoj balíček Java JDK od společnosti Oracle, který je zcela zdarma dostupný na oficiálních stránka společnosti(`https://www.oracle.com/cz/java/technologies/javase-downloads.html`).

2.1.1 Adresářová struktura

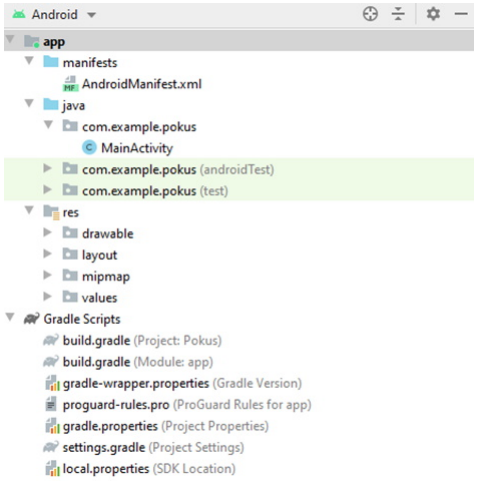
Každý projekt v Android Studiu obsahuje jeden nebo více modulů se soubory zdrojových kódů. Aplikace Android Studio zobrazuje ve výchozím nastavení soubory projektů tak, jak je zobrazeno na obrázku. Tento pohled je uspořádán pomocí modulů, které umožňují rychlý přístup ke klíčovým zdrojovým souborům projektu. Všechny soubory sestavení jsou viditelné na nejvyšší úrovni pod skripty Gradle a každý modul aplikace obsahuje následující složky:(1),(17)

- **Manifest** – AndroidManifest.xml obsahuje důležité informace o projektu, jako je název aplikace nebo ikona aplikace.(1),(17)

- **Java** –ve složce se nacházejí soubory s kódem tříd v programovacím jazyce Java nebo Kotlin.(1),(17)

- **Res** – ve složce se nacházejí XML soubory s definicí uživatelského rozhraní.(1),(17)

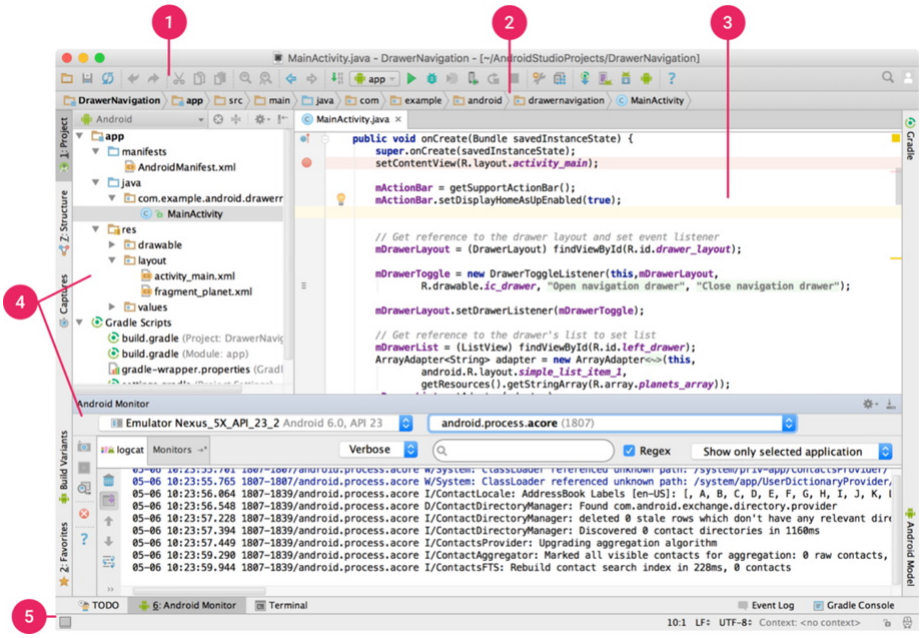
- **Gradle Scripts** – skripty pro kompilátor Gradle. Každý modul má svůj soubor build.gradle.(1),(17)



Obrázek 5: Adresářová struktura Android Studio

2.1.2 Uživatelské rozhraní

Na obrázku níže je znázorněna struktura hlavního okna Android Studia. Je důležité znát vývojové prostředí, ve kterém zrovna programuje. I když se to nezdá, tak se jednotlivá vývojová prostředí od sebe liší, i když se jedná převážně o grafický desing IDE.



Obrázek 6: Uživatelské rozhraní Android Studio IDE

1. **Toolbar**umožňuje provádět širokou škálu akcí, jako spouštění a pozastavení projektu vytvoření AVD, debugging apod.(1),(17)

2. **Navigation Bar** umožňuje pohodlně procházet projekt, otevírat, zavírat a upravovat soubory. Umožnuje tak vývojářům pohodlnější zobrazení adresářové struktury z Tool Window.(1),(17)

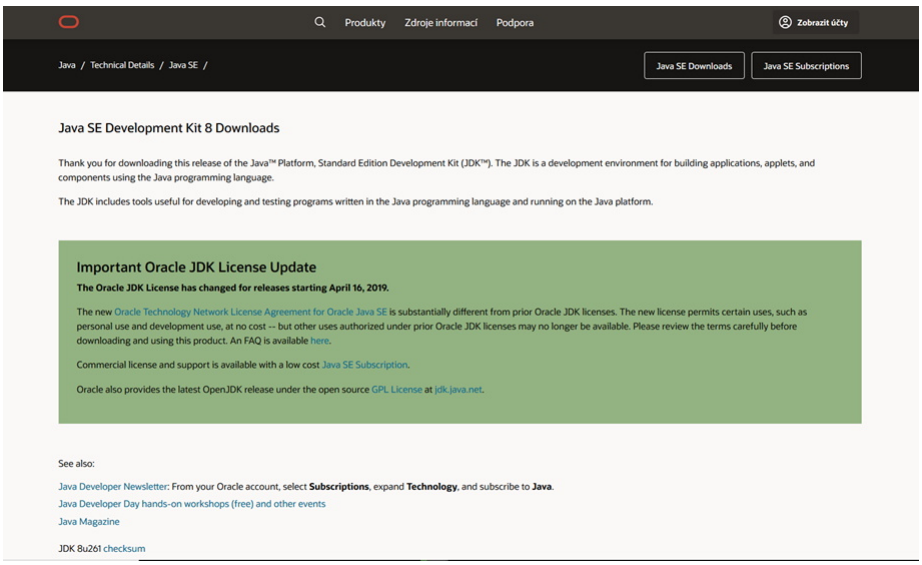
3. **Editor Window**slouží k zobrazení aktuálního souboru, ve kterém je psán zdrojový kód. (1),(17)

4. **Tool Window** umožnuje přístup k potřebným úkolům mezi které patří správa projektu, vyhledávání nebo správa verzí.(1),(17)

5. **Status Bar** slouží k zobrazování aktuálního stavu IDE, zpráv nebo upozornění o aktuálním stavu aplikace. Pokud vyjde nová verze IDE nebo se nepodaří projekt správně zkompilovat, objeví se varování právě zde.(1),(17)

2.2 Java Development Kit (JDK)

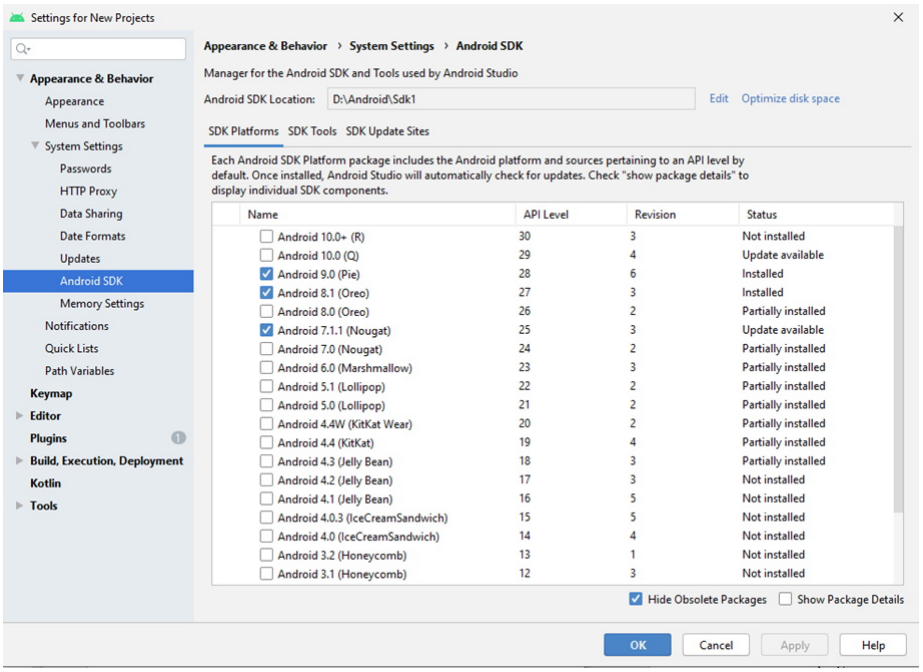
Java Development Kit (JDK) je soubor základních nástrojů a knihoven pro vývoj aplikací a apletů pro platformu Java a Kotlin. Základní součástí JDK je Java Runtime Environment (JRE), jež slouží pro spouštění aplikací i vývojových nástrojů dále překladač, debugger atd. Jelikož se aplikace pro Android programují primárně v jazycích Java a Kotlin, je potřeba JDK, resp. JRE nainstalovat.(1),(11),(17),(18)



Obrázek 7: Webová stránka Java SE Development Kit 8 (JDK)

2.3 Android SDK

Jedná se o sadu nástrojů, která zahrnuje všechny potřebné nástroje pro vývoj aplikací a její debugování. Dříve, když programátor vyvíjel aplikace v Eclipse IDE, musel si balíček SDK stáhnou zvlášť. To se ovšem změnilo příchodem Android Studia, které se už naistaluje automaticky při instalaci IDE. Pro správu SDK se používá SDK Manager, který je součástí vývojového prostředí.



Obrázek 8: SDK Manager v Android Studio IDE

2.4 Java

Java je objektově orientovaný programovací jazyk (OOP), který vyvinula firma Sun Microsystems v 90. letech minulého století. Jedná se v současné době o jeden z nejpopulárnějších a nejvyužívanějších programovacích jazyků. Díky své unikátní konstrukci mohou programátoři své programy spouštět na stolním počítači, webových serverech nebo na mobilních telefonech (Android). Java kód je možné částečně kompilovat a částečně interpretovat. Javovský zdrojový kód se nepřekládá do strojového kódu, jako je tomu např. u jazyku C++, ale překládá se do byte codu. „Bajtkod“ se pak překládá pomocí speciálního programu, který se nazývá Java Virtual Machine, zkráceně JVM. Díky tomuto virtuálnímu stroji lze program spustit na různých operačních systémech – MS Windows, Linux OS nebo macOS.(5),(7),(13),(14),(15),(16)

Obrázek 9: Main.java v jazyku Java

```java
package com.company;
public class Person {
    private String name;
    private String lastname;
    private int age;
    private String email;
    private int phone;

    public Person(String name, String lastname, int age, String email, int phone) {
        this.name = name;
        this.lastname = lastname;
        this.age = age;
        this.email = email;
        this.phone = phone;
    }
    public String getName() { return name; }

    public void setName(String name) { this.name = name;}

    public String getLastname() { return lastname; }

    public void setLastname(String lastname) { this.lastname = lastname; }

    public int getAge() { return age; }

    public void setAge(int age) { this.age = age; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }

    public int getPhone() { return phone; }

    public void setPhone(int phone) { this.phone = phone; }
    @Override
    public String toString() {
        return "Person{" +
                "name='" + name + '\'' + ", lastname='" + lastname + '\'' + ", " +
                "age=" + age + ", email='" + email + '\'' + ", phone=" + phone +
                '}';
    }
}

```

Obrázek 10: Ukázka Person.java v jazyku Java

2.5 Kotlin

Kotlin je nový programovací jazyk vytvořený formou JetBrains, která stojí za vytvořením IntelliJ IDEA a Android Studio. Jedná se o staticky typovaný programovací jazyk, který je primárně zaměřený na Java Virtual Machine a JavaScript. V mnoha literaturách se lze dočíst, že se jedná hlavně o funkcionální jazyk, v podstatě je ale Kotlin objektově orientovaný jazyk (OOP), který podporuje funkcionální programování. Mezi hlavní výhody Kotlinu patří bezpečnost, jednoduchost a velká podobnost s jazyky Java a C#. Ovšem mezi hlavní výhody patří vynechání středníku na konci příkazu, což hodně programátorů uvítalo. Při překladu zdrojového kódu totiž patří mezi nejčastější důvod chybného zkompilování právě středník, který jsme zapomněli vložit.(6)

Obrázek 11: Main.kt v jazyku Kotlin

Obrázek 12: Ukázka Person.kt v jazyku Kotlin

2.6 XML

Jazyk XML (eXtensible Markup Language) je rozšiřitelný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C. Jedná se o zjednodušenou podobu staršího jazyka SGML. Umožňuje snadné vytváření konkrétních značkovacích jazyků (tzv. aplikací) pro různé účely různých typů dat. K hlavní výhodě jazyka patří multiplatformovost a univerzálnost.(8)

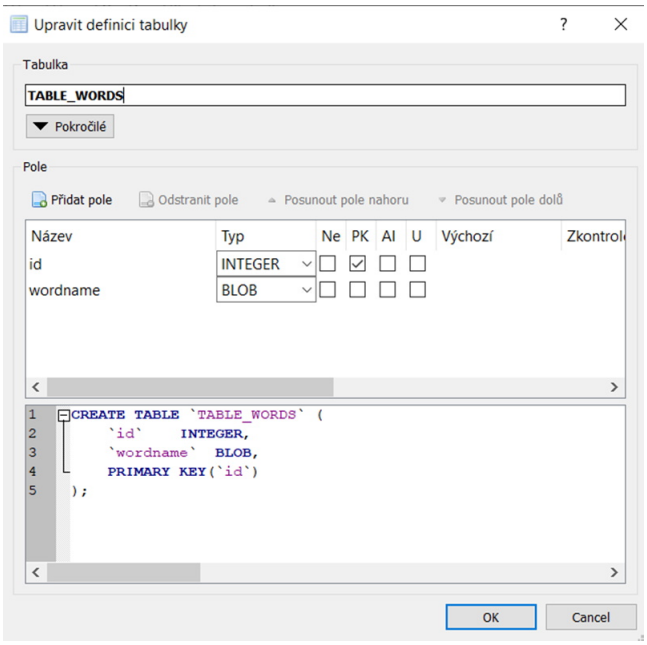
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#fff"
    android:orientation="vertical"
    tools:context=".MainActivity"
    android:weightSum="1">

    <ListView
        android:id="@+id/list"
        android:layout_height="wrap_content"
        android:layout_width="match_parent">
    </ListView>
</LinearLayout>
```

Obrázek 13: Ukázka XML

2.7 SQLite

SQLite reprezentuje relační databázový systém, jenž je obsažen v jedné malé knihovně. Jedná se o systém, který je velmi šetrný k paměti zařízení a poměrně rychlý, což je velká výhoda při použití v zařízeních jako jsou např. mobilní telefony. Další výhodou této relační databáze je její snadná použitelnost v rámci aplikace, jelikož SQLite nepracuje na principu klient-server, jako jiné databázové systémy.(9)



Obrázek 14: Ukázka SQLite v DB Browseru

3 NÁVRH APLIKACE

Tato část práce je věnovaná samotnému návrhu aplikace.

3.1 Základní informace

Všichni vývojáři, než se pustí do vývoje, si musí nejprve určit priority, jako programovací jazyk a vývojové prostředí. Toto rozhodnutí je nesmírně důležité, protože může ovlivnit samotný vývoj. V mém případě jsem zvolil pro vývoj Android Studio. Za programovací jazyky jsem si zvolil Javu, Kotlin a XML.

3.2 Existující aplikace

V Google Play existuje celá řada aplikací, které slouží jako manuál pro tvorbu aplikací. Nechci tím říct, že jsou špatné, ale všechny mají jeden společný problém. A tím problémem je, že jsou v angličtině. Toto beru jako velký nedostatek, a proto jsem se rozhodl to napravit. Ať už se jedná o vývoj mobilních aplikací nebo webových aplikací, je dobré se vždy podívat na již exitující aplikace. Získáte tak informace o tom, co dělá aplikaci dobrou (tyto prvky pak použijete ve své aplikaci), a nebo špatnou (tyto prvky do aplikace nepoužijete nebo se je pokusíte opravit).

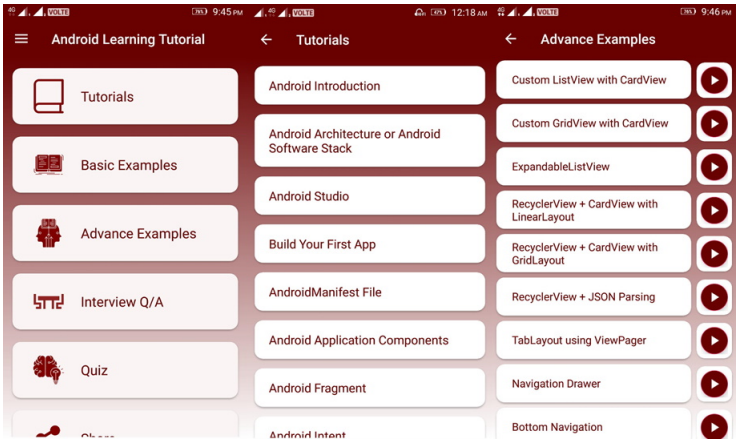
3.2.1 Learn Android Tutorial – Android App Development

Learn Android Tutorial považuji asi za jednu z nejlepších „tutorial“ aplikací. Velkou výhodou této aplikace je, že je zcela zdarma, a navíc oproti jiným aplikacím může běžet zcela v offline režimu, takže nepotřebuje připojení k internetu. Při tvorbě své aplikace mi Learn Android Tutorial posloužil jako vzor, co se týká zobrazování menu nabídky. V rámci vývoje jsem se snažil aplikovat nedostatky jak této, tak stávajících aplikace. Mezi hlavní nevýhody této aplikace považuji:

1. Vysvětluje vývoj aplikací pouze v Javě.
2. Celá aplikace popisuje vývoj v anglickém jazyce.

Celá aplikace Learn Android Tutorial – Andrpif App Development je rozdělena na:

- Tutorial – potřebná teorie o Androidu,
- Basic Examples – jednoduché ukázky příkladů jako je: Menu; UI Widgets, Intents, atd,
- Advance Example – praktické příklady,
- Interview Q/A – nejčastější otázky na Android, příkazy a jejich stručné odpovědi,
- Quiz – prověření uživatelových znalostí,
- Tips and Tricks – tipy a triky pro vývoj,
- Share – možnost sdílení aplikace.



Obrázek 15: Learn Android Tutorial – Android App Development

3.3 Návrh aplikace

Zadání aplikace patří mezi nejdůležitější část celého návrhu každé aplikace. Pokud se přesně nenadefinují požadavky aplikace, může toto chybné zadání vést k fiasku aplikace. Pokud vytvoříme něco, s čím klient nesouhlasí, nemusí nám za náš výtvor zaplatit. Proto je důležité si přesně nadefinovat požadavky, které má aplikace splňovat. Celý návrh se dělí do dvou skupin: funkční a nefunkční požadavky. Mezi funkční požadavky patří dané body, které musí aplikace mít (to, co si klient přesně přeje). Mezi nefunkční požadavky patří např. OS, na kterém se bude aplikace vyvíjet, programovací jazyk atd.

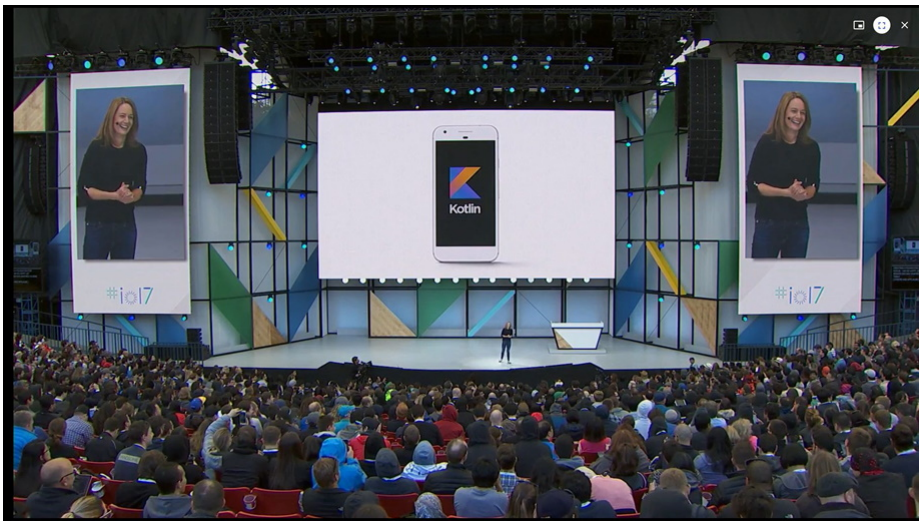
Funkční požadavky	**Nefunkční požadavky**
Vzdělávací aplikace	Windows 10
Android OS	Java, Kotlin
Manuál	Notebook
Programování	
Android - Java a Kotlin	
čeština	

Tabulka 2: Funkční a nefunkční požadavky

Z tabulky vyplývá, že zadání mé aplikace zní: Vytvořit vzdělávací mobilní aplikaci pro Android OS, která bude sloužit jako manuál pro vývoj mobilních aplikací pro Android OS v jazycích Java a Kotlin a celý manuál bude v češtině.

3.4 Volba jazyka

Než jsem se pustil do tvorby aplikace, měl jsem před sebou velký problém. Jaký programovací jazyk zvolit. Jak jsem se již zmínil, pro tvorbu nativní aplikace se používají dva programovací jazyky, Java a Kotlin. Java patří mezi nejpoužívanější programovací jazyky a díky své konstrukci se řadí mezi multiplatformní jazyk. Díky této výhodě si ho Google vybral jako primární a zároveň oficiální programovací jazyk pro tvorbu mobilních aplikací. To se však změnilo v roce 2017 na změnilo, kdy po 10 letech Javu nahradil Kotlin.



Obrázek 16: Onzámení Kotlinu jako primární jazyk na Google I/O 17

Tím došlo k rozdělení Android vývojářů na dva tábory:

1. Java vývojáře
2. Kotlin vývojáře

Tak který si vybrat? Osobně doporučuji používat oba dva. Více než 80 % Android aplikací jsou naprogramovány v Javě, a proto se firmám vyplácí udržovat a spravovat aplikace v Javě, než je přeprogramovat do Kotlinu. Ovšem když se chcete pustit do nové aplikace, doporučuje se ji psát v Kotlinu. Ovšem je tu i další možnost, a to je psát aplikace v obou jazycích. Kotlin totiž vychází z Javy, a proto je možné během vývoje používat hotové javovské knihovny. Navíc, pokud narazíte na určitý problém, který nedokážete vyřešit v Kotlinu nebo ho dokonce Kotlin neumí řešit, můžete část aplikace napsat v Javě.

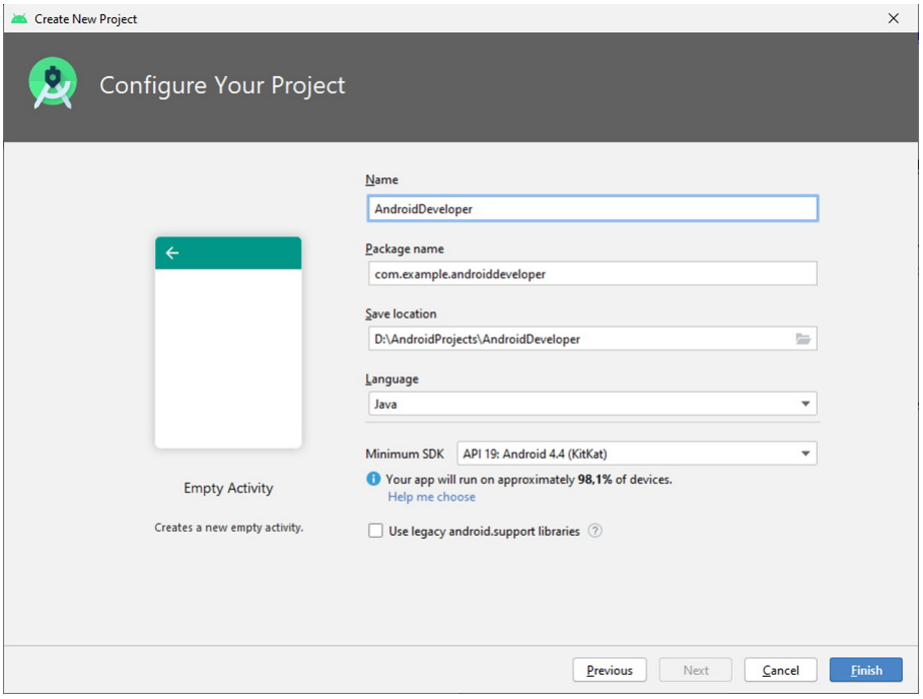
Proto jsem se rozhodl vytvořit svou aplikace v Javě a zároveň v Kotlinu, abych tak ukázal využití těchto jazyků při tvorbě aplikace.

3.5 Příprava a vytvoření nového projektu

Než se pustíme do samotné tvorby aplikace, je nutné si stáhnout potřebné IDE. Pro tvorbu aplikací jsem stejně jako každý současný android programátor zvolil Android Studio. Stačí je pouze stáhnout ze stránky `https://developer.android.com/studio/index.html` a nainstalovat ho. Oproti vývoji mobilních aplikací pro iOS, kde je zapotřebí vlastnit Apple zařízení s Mac OS, má vývoj pro Android značně velkou výhodu. Umožňuje vyvíjet aplikací na všech dostupných platformách (Microsoft Windows, Linux OS a Mac OS). Po stáhnutí a otevření instalačního balíčku se program jednoduše naistaluje Pokud si nevíte rady s instalací nebo během ní došlo k problému, máte možnost vyhledat pomoc na internetové stránce `https://developer.android.com/studio/install.html`, kde máte i podrobný manuál, jak postupovat při instalaci na různých platformách.

Po úspěšném nainstalování se Android Studio otevře. Popis, čím vším je nutné se proklikat zde neuvádím. Je to poměrně pracná činnost a věřím, že po prostudování manuálu, na který jsem již vložil odkaz to každý zvládne. Během instalace se nabízí možnost vytvořit i AVD (Android Virtual Devices), které simuluje reálné mobilní zařízení. Tuto možnost vytvoření mohu jenom doporučit. Při testování aplikace není nutné svůj projekt exportovat a nahrát do fyzického zařízení, ale je možné si projekt jednoduše spustit AVD.

Po úspěšném dokončení potřebného nastavení se zobrazí nabídka s možností vytvoření nového projektu. Jakmile tak učiníme, zobrazí se nám další důležité okno, Configure Your Project, viz obrázek č. 17.



Obrázek 17: Vytvoření nového projektu v Android Studiu

V tomto kroku patří mezi nejdůležitější body:

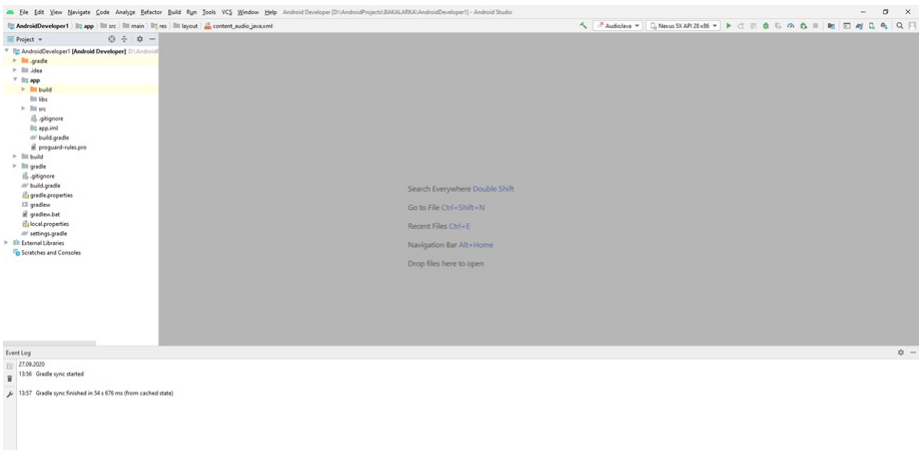
- Name,
- Save location,
- Language,
- Minimum SDK.

U názvu je velmi důležité, aby byl název unikátní (jedinečný). Kdybychom tak neučinili, výsledek by měl katastrofální následky. Při vytvoření několika projektů se stejným názvem by došlo ke sloučení projektů a jen těžko bychom se vyznali. Já jsem si aplikaci pojmenoval Android Developer.

Dalším krokem je pak umístění projektu. Projekt si je možné uložit kamkoliv, ovšem Android Studio stejně jako IntelliJ IDEA nebo Eclipse nabízí možnost ukládat soubory do složky, kterou vývojové prostředí samo vytvoří pro tyto účely. V mém případě se jedná o složku AndroidProject.

V dalším kroku je třeba zvolit programovací jazyk. Pro Android si můžeme vybrat ze dvou primárních jazyků, a to Javy a Kotlinu. Pokud vyvíjíte aplikace sami, je volba jazyka na Vás, ale pokud pracujete v týmu, musíte pracovat v jazyku, na které se předem domluvíte. Já pro svou aplikaci používám oba jazyky, nicméně pro vytvoření projektu jsem vybral Javu.

V posledním kroku je definováno minimální SDK (minimum SDK). Pokud je zvolena nižší verze SDK, je větší šance, že aplikace poběží na většině zařízení. Pro svou aplikaci jsem nastavil minimální SDK na API 19 (Android 4.4 – KitKat).



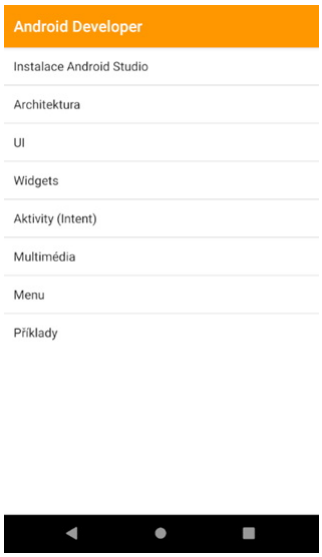
Obrázek 18: Ukázka připraveného IDE pro vývoj

Po vytvořen nového projektu se zobrazí toto okno. Jedná se hlavní okno celého Android Studio IDE. Pokud se toto okno zobrazí, úspěšně se tak podařilo dokončit instalaci a můžeme se pustit do vývoje.

3.6 Tvorba aplikace

Jak jsem se již dříve zmínil v části 3.3 Návrh demonstrativní aplikace, mám za úkol naprogramovat aplikaci v českém jazyce, která poslouží jako manuál pro vývoj mobilních aplikací pro Android.

Dalším krokem je úkol, jak zobrazit všechny příklady např. kalkulačku, video přehrávač, galerii v jedné aplikaci? V odborné literatuře nebo ve video tutoriálech tvoří každý takový příklad svou vlastní aplikaci. I na to má Android svůj způsob řešení. Na rozdíl od desktopové aplikace, kde by se to muselo složitě zabudovávat do kódu, Android umožňuje propojit několik aplikací do jedné **VELKÉ** aplikace. Jelikož se můj návrh aplikace skládá z několika menších aplikací, rozhodl jsem se popsat prvky, které jsme použil při tvorbě hlavní části aplikace, tj. zobrazovací menu celé aplikace (viz. obrázek níže), protože jednotlivé příklady včetně zdrojových kódů jsou popsány v samotné aplikaci.



Obrázek 19: Ukázka ListView v ukázkové aplikaci

Kromě programovacích jazyků Java a Kotlin bylo nutné použít několik rozšiřujících doplňků potřebných pro funkčnost, ovladatelnost a správný chod aplikace. Než jsem se pustil do samotného vývoje, určil jsem potřebné priority, které chci, aby tak aplikace fungovala podle mých představ. Jak by aplikace měla fungovat? Po kliknutí aplikace se uživateli zobrazí seznam jednotlivých tutoriálů. Po výběru daného tutoriálu se uživateli zobrazí demonstrující příklad a po kliknutí ikony pro zobrazení zdrojového kódu se zobrazí zdrojový kód daného příkladu v jazycích Java, Kotlin a XML. Takto by aplikace měla fungovat. Pro realizaci tohoto chodu jsem použil Activity, ListView a Intent. Důvod, proč jsem použil tyto komponenty jsou popsány níže.

Android prvky

- **ListView**
  Android ListView je kontejner, který slouží k zobrazení několika položek v rolovacím seznamu. Typickým příkladem, kde se můžeme uživatel setkat se zobrazením ListView je seznam SMS zpráv nebo seznam kontaktů v mobilu. Pro zajištění zobrazení položek, používá Android tzv. Adapter. Tento Adapter zajistí propojení jednotlivých položek seznamu ve správném pořadí tak, jak jsem si ho sám nadefinoval. Pro seznam je dobré použít patřičnou datovou strukturu, v mém případě pole datového typu string. Důvodem, proč jsem použil datovou strukturu pole je v tom, že se mi zobrazí položky v seznamu přesně tak jak chci, tj. na první pozici se mi zobrazí Instalace Android Studio, na druhé pozici Architektura, atd. Jedná se tak o pevné umístění položek, které uživatel nemůže v aplikaci změnit (např. změnit jejich pořadí). To může pouze programátor ve zdrojovém kódu.

```
String[] values = new String[]{"Instalace Android Studio", "Architektura","UI", "Widgets",

        "Aktivity (Intent)","Multimédia","Menu","Příklady",
};
```

Obrázek 20: Datová struktura s položkami, které se mají zobrazit v ListView

Když je nadefinovaná datová struktura, přichází na řadu její propojení s Adaptérem. Kromě nadefinování datového typu, jména a vytvoření nového objektu musí Adaptér obsahovat čtyři parametry. Jedná se o Context (kontext), resource (zdroj) a object (objekt).

Parametr:	**Význam:**	**Příkaz ve zdrojovém kódu**
Context	Slouží jako odkaz na aktuální třídu, ze které se budou data získávat.	this
resource	ID prostředku pro potřebné rozložení.	android.R.layout.simple_list_item1
object	Jedná se o objekt, jehož data se mají zobrazit.	listOfItems

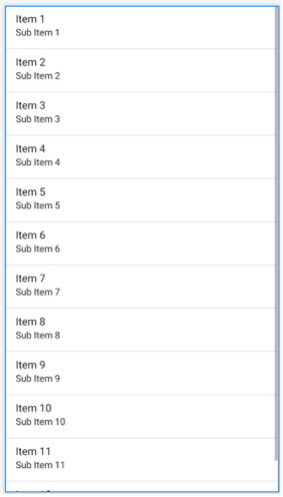
Tabulka 3: Tabulka požadavků pro aplikaci

Pomocí příkazu listView.setAdapter(adapter) pak už stačí propojit Apadpter s XML souborem, který je naprogramovaný jako rolovací seznam (viz obrázek).

```
ArrayAdapter<String> adapter = new ArrayAdapter<>( context: this, android.R.layout.simple_list_item_1,listOfItems);

listView.setAdapter(adapter);
```

Obrázek 21: Propojení datové struktury s adapterem



Obrázek 22: Ukázka ListView pomocí XML v Android Studiu

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.listview_item_click_event);

    listView = (ListView) findViewById(R.id.list);
    String[] values = new String[]{"Instalace Android Studio", "Architektura","UI", "Widgets",

            "Aktivity (Intent)","Multimédia","Menu","Příklady",
    };

    ArrayAdapter<String> adapter = new ArrayAdapter<~>( context: this,
            android.R.layout.simple_list_item_1, android.R.id.text1, values);

    listView.setAdapter(adapter);
```

Obrázek 23: Ukázka třídy ListItem.java v jazyku Java z aplikace

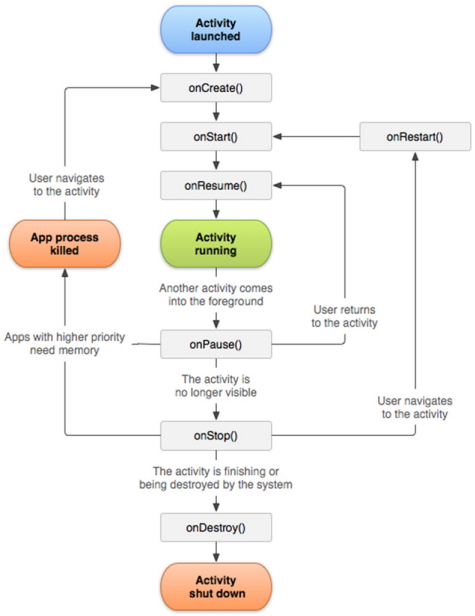
```
protected fun onCreate(savedInstanceState:Bundle) {
  super.onCreate(savedInstanceState)
  setContentView(R.layout.listview_item_click_event)
  listView = findViewById(R.id.list) as ListView
  val values = arrayOf<String>("Instalace Android Studio", "Architektura", "UI", "Widgets",
                               "Aktivity (Intent)", "Multimédia", "Menu", "Příklady")
  val adapter = ArrayAdapter<String>(this,
                                     android.R.layout.simple_list_item_1, android.R.id.text1, values)
  listView.setAdapter(adapter)
```

Obrázek 24: Ukázka třídy ListItem.ktv jazyku Kotlin z aplikace

- **Aktivity**
  V Android světě slouží Aktivity jako hlavní třída, která se po spuštění aplikace spustí hned jako první. Můžeme říct, že se jedná o jeden ze stavebních kamenů všech Android aplikací. Aktivity umožňují pohodlnou komunikaci mezi uživatelem a samotnou aplikací. Všechny aplikace se zpravidla skládají z více aktivit, které jsou mezi sebou navzájem propojeny

  Ať už programujeme v jazyku C, C++, C#, Java nebo Kotlin, pro spuštění programu používáme metodu main(). Stejným způsobem spouštíme aplikace i v Android, s tím rozdílem, že se metoda jmenuje onCreate(). Není to ovšem jediná metoda Aktivity, je jich celkem 7.
  Metody Aktiviry:

  - **onCreate():** – metoda, která se volá hned po vytvoření aktivity;
  - **onStart():** – metoda, která se volá, když se uživatel vrátí do aktivity;
  - **onResume():** – metoda, která se volá, když aktivita komunikuje s uživatelem;
  - **onPause():** – metoda, která se volá, pokud dojde k přesunutí do jiné aktivity;
  - **onStop():** – metoda, která se volá v doby, kdy není aktivita viditelná pro uživatele;
  - **onRestart():** – metoda, která se volá, při ukončení aktivity;
  - **onDestroy():** – metoda, která se volá při restartu aktivity po jejím zastavení;



Obrázek 25: Životní cyklus Activity

- **Intent**
  Intent je abstraktní operace, která nám slouží k vykonání určité operace. Jedná se o objekt, který v systému Android slouží pro komunikaci mezi komponentami aplikace (předávání informací mezi sebou) a přecházení z jedné aplikace do jiné aplikace. V podstatě celé aplikace se skládají z aktivity a ze zpráv mezi intenty. Intenty se převážně skládají z:

  – činnost, která se má vykonat,
  – parametr, nad kterým se má určitá činnost provést,
  – aplikace, která má danou akci provést.

  Mezi typické činnosti, které Intent provádí jsou:

  – Vytočení a příjem hovoru – Intent.***ACTION_CALL***,
  – Odesílání SMS – Intent.***ACTION_VIEW***,
  – Odesílání Emailu – Intent.***ACTION_SEND***,
  – Zobrazené webové stránky – Intent.***ACTION_VIEW***,
  – Zahájení jiné činnost, jako např. otevření nové aktivity.

```java
if (i == 0) {
    Intent intent = new Intent( packageContext: ListViewItem.this,  Install.class);
    startActivity(intent);
}

if (i == 1){
    Intent intent = new Intent( packageContext: ListViewItem.this, Architecture1.class);
    startActivity(intent);
}

if (i == 2) {
    Intent intent = new Intent( packageContext: ListViewItem.this, ListViewUI.class);
    startActivity(intent);
}

if (i == 3) {
    Intent intent = new Intent( packageContext: ListViewItem.this, ListViewWidgets.class);
    startActivity(intent);
}
if (i == 4) {
    Intent intent = new Intent( packageContext: ListViewItem.this, ListViewIntent.class);
    startActivity(intent);
}
if (i == 5) {

    Intent intent = new Intent( packageContext: ListViewItem.this, ListViewMultimedia.class)
    startActivity(intent);

}

if (i == 6) {
    Intent intent = new Intent( packageContext: ListViewItem.this, ListViewMenu.class);
    startActivity(intent);
}

if (i == 7) {
    Intent intent = new Intent( packageContext: ListViewItem.this, ListViewExample.class);
    startActivity(intent);
}
```

Obrázek 26: Intent z aplikace v jazyku Java

```
if (position == 0)
{
  val myIntent = Intent(view.getContext(), Install::class.java)
  startActivityForResult(myIntent, 0)
}
if (position == 1)
{
  val myIntent = Intent(view.getContext(), Architecture1::class.java)
  startActivityForResult(myIntent, 0)
}
if (position == 2)
{
  val myIntent = Intent(view.getContext(), ListViewUI::class.java)
  startActivityForResult(myIntent, 0)
}
if (position == 3)
{
  val myIntent = Intent(view.getContext(), ListViewWidgets::class.java)
  startActivityForResult(myIntent, 0)
}
if (position == 4)
{
  val myIntent = Intent(view.getContext(), ListViewIntent::class.java)
  startActivityForResult(myIntent, 0)
}
if (position == 5)
{
  val myIntent = Intent(view.getContext(), ListViewMultimedia::class.java)
  startActivityForResult(myIntent, 0)
}
if (position == 6)
{
  val myIntent = Intent(view.getContext(), ListViewMenu::class.java)
  startActivityForResult(myIntent, 0)
}
if (position == 7)
{
  val myIntent = Intent(view.getContext(), ListViewExample::class.java)
  startActivityForResult(myIntent, 0)
}
```

Obrázek 27: Intent z aplikace v jazyku Kotlin

Ve své aplikaci jsem použil explicitní Intent. Jedná se o Intent, který slouží k zahájení jiné činnosti, tj spuštění nové aktivity. Příkaz je velice jednoduchý. Stačí vytvořit nový objekt Intent a jako parametr (hodnoty v kulatých závorkách) je nutné uvést Context (výchozí třídu, ze které voláme) a třídu (.class), která se má spustit. Pak už stačí vyvolat tento Intent pomocí metody startActivity a do parametru vložit Intent, který se má spustit (Intent).

Resumé

Tato semestrální práce je zaměřena na vývoj mobilních aplikací pomocí jazyků Java a Kotlin. Za tímto účelem byla vytvořena mobilní aplikace, která slouží jako manuál pro tvorbu aplikací pro Android OS. Nejprve bylo zapotřebí charakterizovat celý operační systém a jeho jednotlivé vrstvy. Poté programovací jazyky a nástroje, které slouží pro vývoj aplikací. V závěru je vysvětlen způsob testování aplikací a jejich následné publikování.

Summary

This semester's thesis is focused on the development of mobile applications using Java and Kotlin. To this end, a mobile application was created, which serves as a manual for creating applications for Android OS. First it was necessary to characterize the entire operating system and its individual layers. Then the programming languages and tools that serve as application development. In the end, the method of testing applications and then publishing them is explained.

Závěr

Při tvorbě této závěrečné práce mi dělaly největší potíže tabulky a správné zobrazení obrázků tam, kde chci.

Reference

[1] Allen, Grant *Android 4: průvodce programováním mobilních aplikací.* Brno: Computer Press, 2013. ISBN 978-80-251-3782-6.

[2] Čada, Ondřej *Objektové programování: naučte se pravidla objektového myšlení.* Praha: Grada, 2009. ISBN 978-80-247-2745-5.

[3] Eclipse desktop & web IDE . *Eclipse desktop & web IDE* [online].[citováno 18. 08. 2020]. `https://eclipse.org/ide/`.

[4] GoalKicker.com. *Android Notes for Professionals.* 2. 12. 2018 [citováno 15. 11. 2020]. Dostupné z `https://books.goalkicker.com/AndroidBook/AndroidNotesForProfessionals.pdf`.

[5] HEROUT, Pavel, *Učebnice jazyka Java. 5., rozš.vyd.,* České Budějovice: Kopp, 2010. 978-80-7232-398-2.

[6] JEMEROV, Dmitry a ISAKOVA, Svetlana. *Kotlin in action.* Shelter Island, NY: Manning Publications Co., 2017. 9781617293290

[7] KISZKA, Bogdan. *1001 tipů a triků pro jazyk Java.* Brno: Computer Press, 2009. 978-80-251-2467-3.

[8] KOČÍ, Michal. Interval.cz—Svět internetu, Technologií a Bezpečnosti [online].[citováno 21. 12. 2020]. Dostupné z: `https://www.interval.cz/clanky/co-je-xml/`

[9] KREIBICH, Jay A. *Using SQLite.* Sebastopol: O'Reilly Media, Inc, USA, 2010. 978-0-596-52118-9

[10] LACKO, Ľuboslav.*Mistrovství – Android.* Brno: Computer Press., 2017. 978-80-251-4875-4.

[11] LACKO, Ľuboslav, *Vývoj aplikací pro Android.* Brno: Computer Press, 2015. 978-80-251-4347-6

[12] Overview – NetBeans IDE [online].[citováno 18. 08. 2020]. Dostupné z: `https://netbeans.org/features/index.html`

[13] PECINOVSKÝ, Rudolf. *Java 8: úvod do objektové architektury pro mírně pokročilé.* Praha: Grada Publishing, 2014. Knihovna programátora (Grada). 978-80-247-4638-8.

[14] PECINOVSKÝ, Rudolf.*OOP a Java 8. Návrh a vývoj složitějšího projektu vyhovujcího zadanému rámci.* Nakladatelství Tomáš Bruckner. Řepík-Živonín 2015. 978-80-87924-03-17.

[15] SCHILDT, Hubert, *Java 8: výukový kurz.* Brno: Computer Press,2016. 978-80-251-4665-1

[16] SPELL, Brett, *Java: programujeme profesionálně.* Praha: Computer Press, 2002. 80-7226-667-5

[17] UJBÁNYAL, Miroslav, *Programujeme pro Android.* Praha: Grada, 2012. 978-80-247-3995-3

[18] VÁVRŮ, Jiří a UJBÁNYAL, Miroslav, *Programujem pro Android. 2., rozš. vyd.* Praha: Grada, 2013. 978-80-247-4863-4

Seznam obrázků

Seznam tabulek