Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Tvorba mobilních aplikací v jazyce Kotlin

Monika Kopřivová

Bakalářská práce
2021

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Monika Kopřivová**
Osobní číslo:	**I18144**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Tvorba mobilních aplikací v jazyce Kotlin**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem bakalářské práce je vytvoření manuálu pro tvorbu mobilních aplikací v jazyce Kotlin pro operační systém Android. V teoretické části práce bude popsána problematika tvorby aplikací pro operační systém Android se zaměřením na daný programovací jazyk. Také zde budou popsány i další mobilní operační systémy. Dále bude popsáno vývojového prostředí, základní konstrukce Jazyka Kotlin a tvorba GUI aplikací s praktickými ukázkami jednotlivých ovládacích komponent. V praktické části práce student vytvoří dvě funkční mobilní aplikace. První z aplikací bude zaměřena především na design a na ukázku základního rozložení layoutu. Druhá aplikace se bude zabývat ukázkou práce s některým ze senzoru mobilního telefonu (např. bluetooth).

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

SPÄTH, Peter. *Pro Android with Kotlin: Developing Modern Mobile Apps*. Berlin: Springer, 2018. ISBN 978-1484238196.
LACKO, Ľuboslav. *Mistrovství – Android*. Brno: Computer Press, 2017. Mistrovství. ISBN 978-80-251-4875-4.
SILVA, Vladimir. *Vývoj her pro Android: profesionálně*. Brno: Zoner Press, c2013. Encyklopedie Zoner Press. ISBN 978-80-7413-255-1

Vedoucí bakalářské práce: **Ing. Miroslav Dvořák, Dipl.tech.**
Katedra informačních technologií

Datum zadání bakalářské práce: **31. října 2020**
Termín odevzdání bakalářské práce: **14. května 2021**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 26. února 2021

Prohlašuji:

Tuto práci jsem vypracovala samostatně. Veškeré literární prameny a informace, které jsem v práci využila, jsou uvedeny v seznamu použité literatury.

Byla jsem seznámena s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 7. 5. 2021

Monika Kopřivová

PODĚKOVÁNÍ

Tímto bych chtěla poděkovat svému vedoucímu práce Ing. Miroslavu Dvořákovi, Dipl.tech. za ochotu, vstřícnost a pomoc při vypracování této práce. Děkuji také své rodině a svým přátelům za podporu a trpělivost během celého studia.

ANOTACE

V bakalářské práci je popsána tvorba mobilních aplikací. Práce by měla sloužit jako pomocný návod pro vývoj aplikací v programovacím jazyce Kotlin. Jsou zde uvedeny výhody tohoto programovacího jazyka a na příkladech ukázáno programování v něm. Součástí bakalářské práce jsou také dvě aplikace, které by měly sloužit jako vzor při programování mobilních aplikací v jazyce Kotlin.

KLÍČOVÁ SLOVA

Mobilní technologie, aplikace, Kotlin, Java, piškvorky, kalkulátor.

TITLE

Creation of mobile applications in the Kotlin language

ANNOTATION

The bachelor thesis describes the programming of mobile applications. The work should be a guide for creating applications in the Kotlin programming language. The advantages of the Kotlin programming language are written in the bachelor's thesis and the examples show programming in the Kotlin language. The bachelor's thesis also includes two applications that should be a model for programming mobile applications in the Kotlin language.

KEYWORDS

Mobile technologies, application, Kotlin, Java, tic tac toe, calculator.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

GPS	Global Positioning System
HTML	HyperText Markup Language
IDE	Integrated Development Environment
OS	Operating System
RFCOMM	Radio Frequency Communication
SDK	Software Development Kit
SW	Software
UI	User Interface
URL	Uniform Resource Locator
UX	User Experience
XML	Extensible Markup Language

ÚVOD

Hlavním cílem bakalářské práce je seznámení s vývojem mobilních aplikací v programovacím jazyce Kotlin pro operační systém Android a vytvoření dvou funkčních vzorových aplikací. Práce by měla čtenářům poskytnout základní informace o vývoji mobilních aplikací a na dvou vzorových aplikacích představit vývoj aplikací v programovacím jazyce Kotlin.

Teoretická část práce se nejdříve věnuje vysvětlení pojmu mobilní aplikace a vývoji mobilních aplikací. V práci jsou představena vývojová prostředí pro tvorbu mobilních aplikací. Tato vývojová prostředí jsou zde popsána a porovnána. Dále je v práci představen programovací jazyk Kotlin. Ten je následně porovnán s programovacím jazykem Java. Jsou zde popsány výhody jazyka Kotlin a vysvětleny důvody vzniku a obliby tohoto programovacího jazyka.

Praktická část práce se věnuje tvorbě mobilních aplikací a návrhu dvou vzorových aplikací. Tato část práce se nejdříve zabývá tvorbou vzhledu aplikací. To zahrnuje různé možnosti rozložení, přidávání ikon a obrázků a celkový design aplikací. Dále se praktická část práce věnuje příkladům programování některých částí aplikací. Poslední úsek práce je věnován návrhu a vývoji dvou ukázkových aplikací.

První aplikace slouží především jako ukázka práce s rozložením a designem. Aplikací je jednoduchý kalkulátor, na němž lze pozorovat práci s rozvržením aplikace. Aplikace je navržena tak, aby byla co nejpřehlednější a její ovládání bylo pro uživatele snadno pochopitelné a bezproblémové. Aplikace funguje jako běžný kalkulátor, dovoluje tedy práci se základními matematickými operacemi a zobrazuje výsledky těchto operací.

Druhá aplikace je o něco složitější, jejím cílem je seznámení se složitějšími operacemi při vývoji mobilních aplikací v programovacím jazyce Kotlin, konkrétně s technologií Bluetooth. Aplikací je hra piškvorky. Piškvorky jsou strategickou hrou, při níž spolu soupeří dva hráči střídající se v kreslení dvou rozdílných symbolů. Aplikace umožňuje hru proti jinému hráči pomocí využití technologie Bluetooth. Na této aplikaci jsou ukázány složitější postupy v programování, aplikace uchovává současný stav hry, udržuje komunikaci mezi zařízeními, rozhoduje o ukončení hry a o vítězi hry.

Závěr práce se zabývá jednoduchým popisem výsledných aplikací, jejich možnými budoucími úpravami a závěrečným shrnutím celé práce.

1 MOBILNÍ APLIKACE

Mobilní aplikace jsou programy běžící na mobilních telefonech. Jedná se o softwarové aplikace, které jsou přímo vyvíjené pro mobilní zařízení. Aplikace jsou běžně k dispozici na distribučních platformách pro daný operační systém. Na trhu v dnešní době existuje nespočet mobilních aplikací. Aplikace lze rozdělit dle operačního systému, pro který byly vyvíjeny. Dalším možným rozdělením je rozdělení na nativní aplikace, webové aplikace a hybridní aplikace.

Nativní aplikace jsou přímo v zařízení a jsou přístupné prostřednictvím ikon na domovské obrazovce zařízení. Instalace nativních aplikací probíhá prostřednictvím obchodu s aplikacemi. Jsou vyvinuty speciálně pro jednu platformu a mohou plně využít všech funkcí zařízení - mohou používat fotoaparát, GPS, akcelerometr, kompas, seznam kontaktů atd. Mohou také zahrnovat gesta. Nativní aplikace mohou také využívat systém upozornění zařízení a mohou pracovat i v offline režimu. [1]

Webové aplikace nejsou skutečné aplikace, nejsou tedy přímo v zařízení. Jsou to webové stránky, které v mnoha ohledech vypadají a působí jako nativní aplikace, ale nejsou implementovány. Jsou spouštěny prohlížečem a obvykle jsou psány v HTML. Uživatelé k nim nejprve přistupují tak, jako by přistupovali k jakékoli webové stránce: přejdou na speciální adresu URL a poté mají možnost je přidat na svou domovskou obrazovku vytvořením záložky na tuto stránku. [1]

Hybridní aplikace jsou částečně nativní aplikace a částečně webové aplikace. Mnoho lidí je nesprávně nazývá webovými aplikacemi. Stejně jako nativní aplikace jsou v obchodě s aplikacemi a mohou využívat mnoho dostupných funkcí zařízení. Stejně jako webové aplikace se spoléhají na vykreslení HTML v prohlížeči, ale prohlížeč je v tomto případě přímo v aplikaci. [1]

1.1 Vývoj mobilních aplikací

Vývoj mobilních aplikací je rozsáhlý proces, zahrnující spoustu faktorů, nejedná se pouze o naprogramování dané aplikace, jak by se spousta lidí mohla mylně domnívat. Při vývoji mobilních aplikací je důležité rozhodnout se, pro jaký operační systém bude aplikace naprogramována a zda se bude jednat o aplikaci nativní, webovou či hybridní. Dle těchto faktorů se dále rozhoduje o technologii, která bude při vývoji použita. Také je důležité zvolit metodiku vývoje.

Ve chvíli, kdy jsou vyřešeny všechny předchozí body a je schválen základní návrh aplikace, přichází na řadu samotný návrh a programování aplikace. Během této části nastává opět několik faktorů, které nelze opomenout. Jedním z nejdůležitějších bodů je samotný design aplikace.

Design v případě vývoje mobilních aplikací neznamená pouze vzhled uživatelského rozhraní (UI), ale především design celého výsledného produktu z hlediska funkčnosti a navigace, tzv. user experience (UX). [2]

Dalším velice důležitým bodem v této části vývoje je způsob, jak bude zacházeno s daty. Pokud bude aplikace ukládat data, je nutné vyřešit způsob ukládání dat, rychlost ukládání dat a především bezpečnost. Ochrana uživatelských dat je v dnešní době velice důležitá a v případě, že aplikace zpracovává uživatelská data, musí tuto ochranu poskytovat.

Po samotném naprogramování aplikace je ve vývoji ještě několik důležitých bodů, které by neměly zůstat opomenuty. Jedním z nich je otestování výsledného produktu. Produkt by měl být otestován ještě před uvedením na trh, aby byl prostor pro vyřešení případných chyb.

Na testování mobilních aplikací by se rozhodně nemělo šetřit. Pokud bude aplikace průběžně testována, může to vývojářům ušetřit výdaje a celý projekt se díky testování může výrazně urychlit. Z těchto důvodů by se testování nemělo zanedbávat a měl by mu být během vývoje projektu věnován dostatečný prostor. [2]

Další důležitou částí vývoje mobilní aplikace je dokumentace. Ta by měla být psána v průběhu celého vývoje mobilní aplikace. Správná dokumentace usnadňuje práci nejen vývojářům, kteří aplikaci vytvoří a zpětně chtějí něco změnit nebo upravit, ale i vývojářům, kteří nebyli součástí projektu po celou dobu, ale připojili se k vývoji až později. V takovém případě dokumentace usnadňuje pochopení aplikace a jejího kódu. V některých případech, zvlášť jedná-li se o složitější aplikaci, je vhodné vytvořit i jednoduchou dokumentaci nebo uživatelskou příručku pro její budoucí uživatele, která jim pomůže pochopit základní práci s aplikací.

Důležité je, aby byl projekt dobře dokumentovaný, jinak se může například přechod k jinému dodavateli stát velice nákladným a téměř nemožným. Častou chybou je, že se dokumentace píše až na konci projektu. Je však důležité dokumentovat v projektu průběžně každou funkci. [2]

Jedním z posledních bodů ve vývoji je podpora mobilní aplikace a její další vývoj.

V neposlední řadě je důležité myslet už na začátku vývoje na dlouhodobé výdaje na úpravy a údržbu výsledné aplikace, správu dat a cloudu, vytváření postupných updatů a poskytování uživatelské podpory. [2]

2 MOBILNÍ OPERAČNÍ SYSTÉMY

Tato část práce se zaměřuje na mobilní operační systémy. Operační systém mobilního telefonu je základní softwarové vybavení, které slouží k tomu, aby uživatelé mohli telefon bez problémů ovládat, instalovat do něho aplikace a spravovat hardware. V dnešní době patří mezi nejpoužívanější operační systémy pro mobilní zařízení Android a iOS.

2.1 Android

Android je operační systém založený na jádře operačního systému Linux. Jedná se o nejrozšířenější mobilní operační systém, se kterým se lze setkat u mnoha značek a typů mobilních telefonů. Současnou nejnovější verzí tohoto operačního systému je Android 11.

Android je rozsáhlý operační systém, který byl vytvořený společností Google a je založený na open source platformě, jde tedy o počítačový software, který má otevřený zdrojový kód. Přičemž „otevřený kód“ zde reprezentuje snadnou dostupnost, čímž se myslí dostupnost technická i licenční. [3]

V současné době je Android nejoblíbenějším operačním systémem pro mobilní zařízení. Zastoupení na trhu se pohybuje okolo 80 %. Z tohoto důvodu je tato platforma výzvou pro vývojáře aplikací. [4]

2.2 iOS

Druhým nejpoužívanějším mobilním operačním systémem je iOS. Jedná se o operační systém, který je spjat s mobilními telefony iPhone a značkou Apple. Nejnovější verzí tohoto operačního systému je v současnosti iOS 14.

Tato značka mobilních telefonů je v dnešní době již legendární. Svým vlastníkům přináší perfektní zpracování mobilních zařízení s důrazem na intuitivní a co možná nejjednodušší ovládání. Jedná se o plynulý operační systém, který je díky své možnosti pravidelných upgradů stále aktuální. [5]

Operační systém iOS je využíván především na mobilních platformách Applu, což zahrnuje smartphony iPhone, tablety iPad a také multimediální přehrávače iPod Touch. [6]

2.3 Windows Phone

Tento mobilní operační systém je méně známý a používaný než předchozí dva, stále však existuje celkem dost zařízení právě s tímto mobilním operačním systémem. Vychází z desktopových Windows a i jeho vzhled je odvozen od Microsoft Windows. Nejnovější verzí

tohoto operačního systému jsou Windows 10. Windows 10 jsou již univerzálním systémem pro všechna zařízení, proto název již neobsahuje slovo Phone.

Ovládání OS Windows se oproti Androidu i iOS zcela odlišuje. Úvodní obrazovka je vytvořena formou dlaždic, které lze snadno měnit, lze upravovat jejich pořadí i velikost. Právě díky této možnosti je OS Windows vhodný i pro starší uživatele, kteří již nemají nejlepší zrak. Mezi další výhody patří to, že tento OS dobře funguje i na levnějších telefonech a je také velmi úzce propojen s počítačovou platformou Windows. [5]

3 VÝVOJOVÁ PROSTŘEDÍ

Tato část se věnuje vývojovým prostředím pro tvorbu mobilních aplikací. Pojem vývojové prostředí nebo také IDE označuje software, který usnadňuje programátorům vývoj systémů.

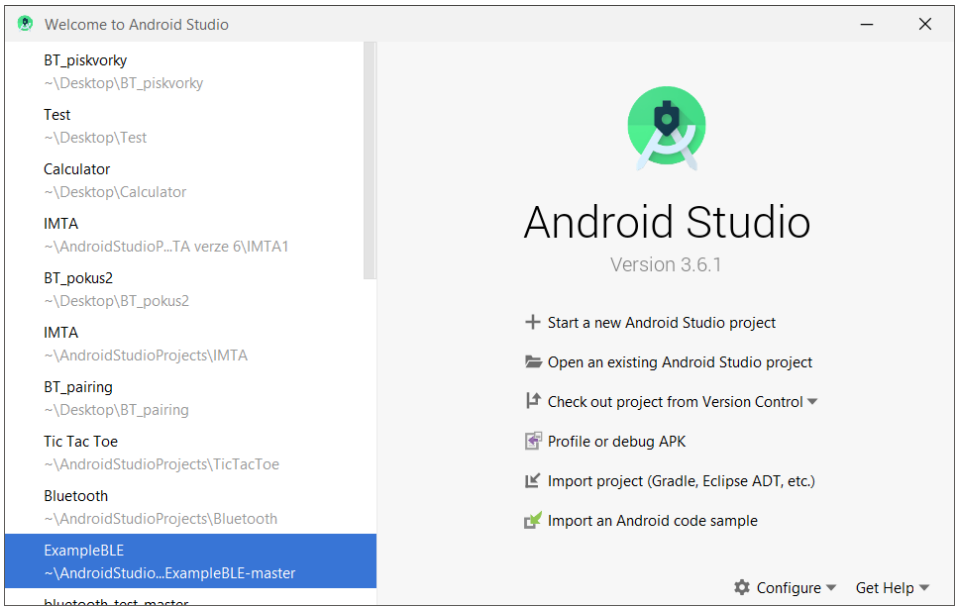
Často podporuje různé programovací jazyky. Vývojová prostředí pak mají řadu různých funkcí a nástrojů, které pomáhají vývojářům při vytváření softwaru, většinou obsahují editor zdrojového kódu, debugger a kompilátor, některá pak obsahují i interpret. [7]

3.1 Android Studio

Prvním vývojovým prostředím, na které se práce zaměřuje, je Android Studio. Toto vývojové prostředí je v současné době asi nejpoužívanějším prostředím pro vývoj mobilních aplikací pro Android. Android Studio je vyvíjeno společnostmi Google a JetBrains. Poslední dostupnou verzí je Android Studio 4.1.2

3.1.1 Instalace a první spuštění

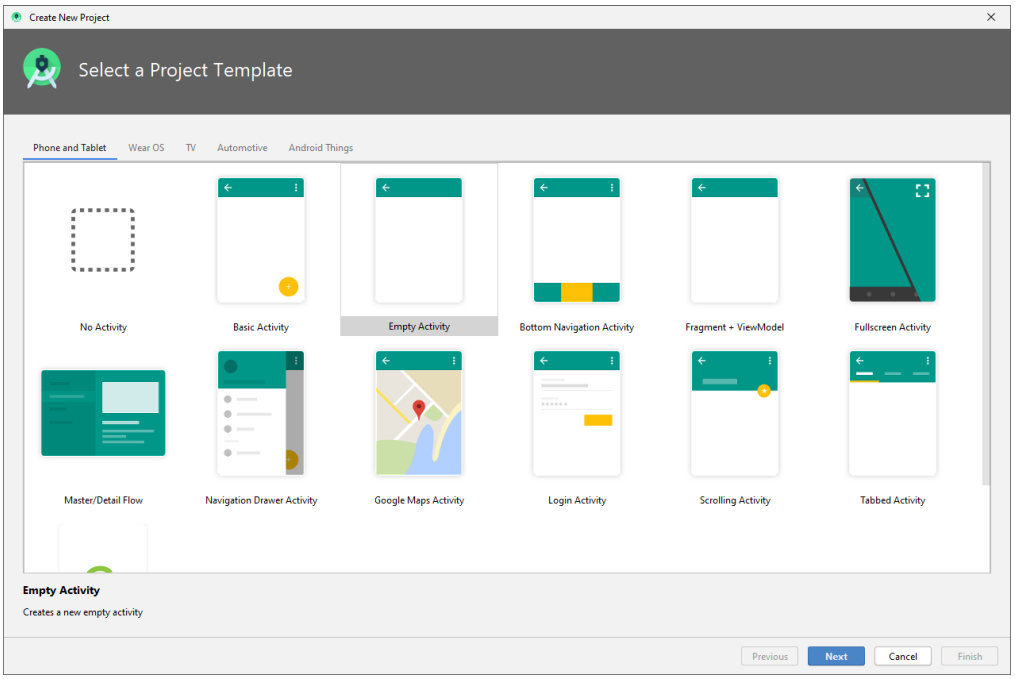
Instalace Android Studia je v celku jednoduchá, celá instalace je dobře vysvětlena, popsána a dá se zvládnout bez jakýchkoli problémů. Android Studio si společně s vývojovým prostředím samo nainstaluje i SDK. Android Studio má k dispozici instalační balíčky pro všechny tři nejpoužívanější operační systémy (Windows, Mac OS, Linux). Při prvním spuštění po instalaci se ihned objevuje obrazovka pro vytvoření nového projektu.



Obrázek 1 Android Studio - vytvoření projektu

3.1.2 Vytvoření projektu

Stejně jako samotná instalace je i vytvoření projektu jednoduché a v několika krocích dobře popsané. Nejdříve se objevuje úvodní obrazovka s nabídkou. Zde je možné zvolit buď nový projekt, nebo otevřít nějaký již existující. Po zvolení možnosti nového projektu se otevírá okno s možnými šablonami projektu. Zde je na výběr z několika základních šablon, volba šablony záleží na konkrétních požadavcích na výslednou aplikaci. Nejčastější volbou bývá vytvoření „prázdné aktivity“. Po výběru některé z šablon následuje okno pro vyplnění základních informací o projektu, jako například jeho jméno a umístění. Poté je již projekt vytvořen.



Obrázek 2 Android Studio - výběr šablony

3.1.3 Kladné stránky

Jak je již zmíněno výše v práci, instalace i samotné vytvoření projektu je dobře popsané a měl by to zvládnout i úplný začátečník. Vzhledem k tomu, že je Android Studio nejrozšířenějším IDE pro vývoj mobilních aplikací pro Android, je možné nalézt na internetu spoustu českých i anglických návodů na samotnou instalaci i práci v něm. Z toho důvodu je toto vývojové prostředí vhodné i pro úplné začátečníky, kteří nemají s vývojem mobilních aplikací žádné zkušenosti.

3.1.4 Záporné stránky

Jednou z nevýhod je velký výběr možností, které Android Studio nabízí. Z jednoho úhlu pohledu se to dá považovat za výhodu, z jiného úhlu je to však i nevýhodou. Už jen při vytváření samotného projektu nabízí Android Studio celkem velké množství šablon, stejně obsáhlé je pak

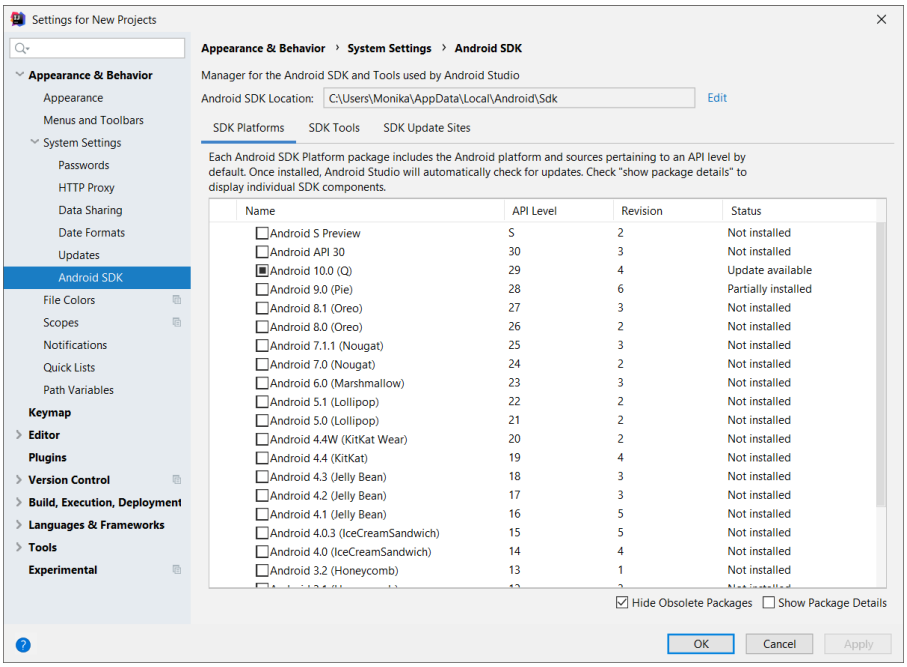
i množství layoutů a komponent a také samotné navigační menu. Pro začátečníky je nabídka možná až příliš široká a využijí jen malou škálu z možných nastavení.

3.2 IntelliJ IDEA

Dalším z populárních vývojových prostředí je IntelliJ IDEA. Toto vývojové prostředí není primárně určeno pro vývoj mobilních aplikací, i přesto se v něm dají mobilní aplikace pro Android vcelku jednoduše vytvářet. Dříve zmíněné Android Studio je založeno právě na IntelliJ IDEA, ale IntelliJ IDEA je obecnější IDE zatímco Android Studio se specializuje především na vývoj mobilních aplikací pro Android. IntelliJ IDEA je vyvíjena společností JetBrains a poslední dostupnou verzí je IntelliJ IDEA 2020.3.1.

3.2.1 Instalace a první spuštění

Na rozdíl od Android Studia je IntelliJ IDEA placeným nástrojem. Je však možné stáhnout si bezplatnou odlehčenou community verzi nebo 30 denní ultimate zkušební verzi. Pro vývoj mobilních aplikací je nutné si samostatně stáhnout a nainstalovat Android SDK. Samotná instalace IntelliJ IDEA je snadná a v několika krocích dobře popsána. Samotné doinstalování SDK je však již poměrně složitější a určitě není pro úplné začátečníky.



Obrázek 3 IntelliJ IDEA - přidání SDK

3.2.2 Vytvoření projektu

Po úspěšném nainstalování IntelliJ IDEA a SDK je možné jednoduše vytvořit nový projekt. V úvodním okně je nutné zvolit, že se jedná o Android projekt. Následuje zadání základních informací o projektu a výběr šablony. Vše je již velice jednoduché a bezproblémové.

3.2.3 Kladné stránky

Instalace samotné IntelliJ IDEA a vytváření projektu je snadné a bezproblémové. Vzhledem k tomu, že toto vývojové prostředí není určeno primárně pro vývoj mobilních aplikací, je zde omezenější nabídka různých funkcí a možností. To může být výhodou pro začátečníky, kteří by stejně nevyužili širokou škálu možných funkcí. Další výhodou je, že IntelliJ IDEA je univerzální IDE a lze ji tedy použít i pro jiné věci než vývoj mobilních aplikací. Toto IDE je tedy vhodné pro všechny vývojáře nejen pro ty, kteří vytvářejí pouze mobilní aplikace.

3.2.4 Záporné stránky

Hlavní nevýhodou tohoto vývojového prostředí je především jeho cena. Na rozdíl od Android Studia není bezplatné a je tedy potřeba se zamyslet, zda se vyplatí si za toto vývojové prostředí zaplatit. Další velkou nevýhodou pak je instalace SDK. Pro vývojáře, kteří již mají s tímto vývojovým prostředím zkušenosti, by to neměl být velký problém. Avšak pro úplné začátečníky je to docela obtížné. Pro začátečníky, kteří by chtěli začít vyvíjet mobilní aplikace pro Android, není tedy IntelliJ IDEA úplně nejvhodnější, lepší variantou by v tomto případě bylo již dříve zmíněné Android Studio.

3.3 Xcode

Další vývojové prostředí, které tato práce popisuje, se zaměřuje na vývoj aplikací pro iOS. Tímto vývojovým prostředím je Xcode. Jedná se o jedno z nejlepších vývojový prostředí pro tvorbu aplikací pro iOS. Toto IDE je vyvíjeno společností Apple. Je možné ho zdarma stáhnout z App Store, ale je dostupné pouze pro operační systémy Mac OS. Aktuálně nejnovější verzí je 10.3 Xcode.

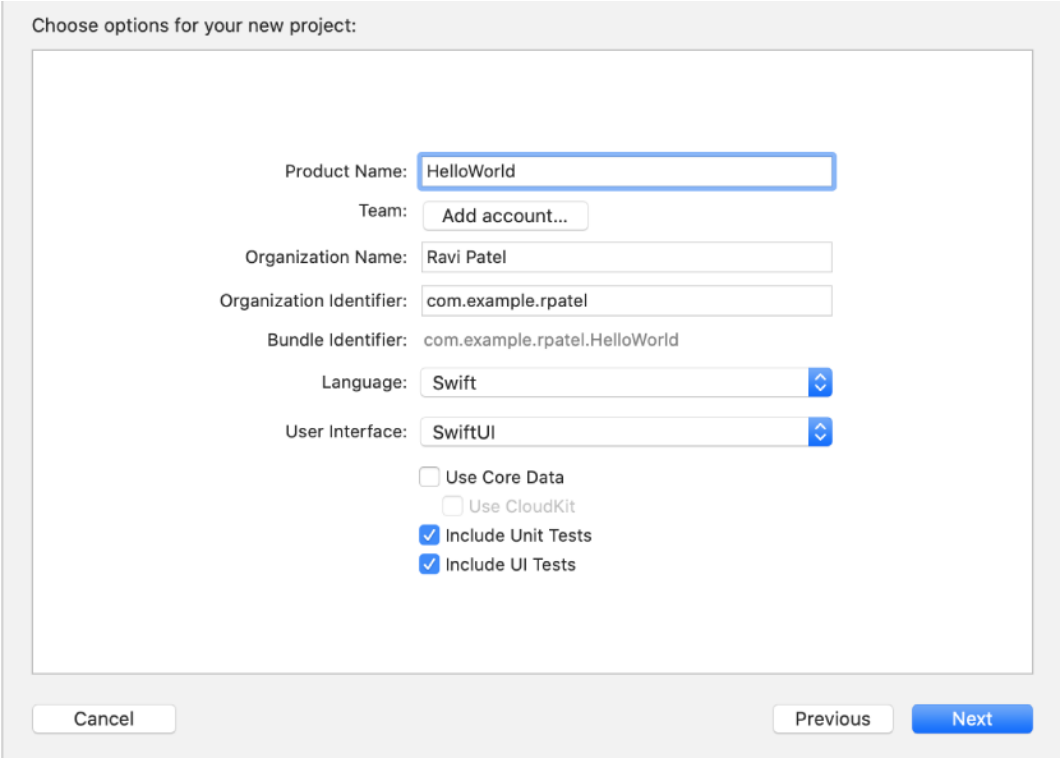
3.3.1 Instalace a první spuštění

Instalace Xcode je opět celkem snadnou záležitostí. Během instalace je vše dobře vysvětleno a vše by mělo probíhat bez problémů. Jak již bylo zmíněno, toto vývojové prostředí je dostupné zdarma, ale pouze pro operační systémy Mac OS. V případě, že by uživatel vlastnil počítač s jiným operačním systémem a chtěl by vyvíjet aplikace pomocí Xcode, musel by si například vytvořit virtuální počítač s Mac Os.

3.3.2 Vytvoření projektu

Vytvoření projektu v Xcode je snadno zvládnutelné a neměl by to být problém ani pro úplné začátečníky. Vše se odehrává v několika krocích, kde si uživatel zvolí cílový operační systém nebo platformu a případně také šablonu, následuje vyplnění názvu projektu, identifikátoru

organizace a názvu organizace, také je možné zvolit programovací jazyk a rozhraní. Po vyplnění těchto základních informací se již vytvoří projekt a je možné začít vyvíjet aplikaci.



Obrázek 4 Xcode - vytvoření projektu [11]

3.3.3 Kladné stránky

Mezi kladné stránky patří jeho bezproblémová instalace a snadné vytvoření projektu. Další výhodou je, že Xcode je dostupné zcela zdarma. Jedná se o nejpoužívanější vývojové prostředí pro tvorbu aplikací pro iOS, vše je tedy snadné a přizpůsobené i pro úplné začátečníky. Právě pro ně je Xcode, vzhledem k pořizovacím nákladům a jednoduchosti instalace a založení projektu, asi nejvhodnějším IDE. Samozřejmě je vhodné i pro pokročilé uživatele, protože nabízí spoustu možností. Je například možné vyvíjet aplikace pro iPhone, iPad, Mac, Apple Watch i Apple TV.

3.3.4 Záporné stránky

Xcode je možné nainstalovat pouze na zařízení s Mac OS. To je asi největší nevýhodou tohoto vývojového prostředí. Pro úplné začátečníky, kteří Mac OS nemají je téměř nemožné si Xcode stáhnout a vyvíjet pomocí něho aplikace. Pro pokročilejší uživatele by neměl být problém vytvořit si například virtuální zařízení s Mac OS, ale i toto řešení má své nevýhody. Samozřejmě se problém dá řešit i jinak než pomocí virtualizace, ale všechna řešení už jsou trošku komplikovanější a odradí pak především uživatele, kteří si vývoj pro iOS chtějí pouze vyzkoušet.

4 PROGRAMOVACÍ JAZYKY

Tato část bakalářské práce se zabývá programovacími jazyky vhodnými pro vývoj mobilních aplikací. V této práci budou představeny dva programovací jazyky. Prvním ze zvolených programovacích jazyků je Java a druhým Kotlin. Oba dva zvolené jazyky jsou vhodné pro vývoj aplikací pro Android. Tyto dva jazyky pak budou vzájemně porovnány.

4.1 Java

Java je objektově orientovaný programovací jazyk. Jde o silně typovaný programovací jazyk, což znamená, že do proměnných lze přiřadit pouze hodnoty předem zvoleného datového typu.

Společnost Sun Microsystems odstartovala projekt, který vedl ke vzniku programovacího jazyka Java, v roce 1991 a výsledný název Java je údajně převzat podle kávy, kterou jeho tvůrci pili (Java coffee). Největší motivací pro vytvoření tohoto jazyka však bylo masové šíření internetu. Statické stránky, které byly vytvářeny pouze pomocí HTML kódu, brzy přestaly webdesignérům stačit a pro stránky s dynamickým obsahem byl programovací jazyk Java ideální volbou. [8]

Z Javy se stal jeden z nejpoužívanějších programovacích jazyků na světě. Tento programovací jazyk je oblíbený z mnoha důvodů, jedním z nich je určitě jeho jednoduchost. Na rozdíl od jiných jazyků se v Javě například nemusí řešit práce s alokací a dealokací paměti. Dalším z mnoha důvodů, proč se tento jazyk těší takové oblíbenosti je například i jeho nezávislost na platformě. To znamená, že aplikace vytvořená v Javě by měla běžet na jakémkoliv operačním systému a na jakékoliv libovolné architektuře.

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello world");
    }
}
```

Obrázek 5 Ukázka kódu v jazyce Java

4.2 Kotlin

Kotlin je moderní staticky napsaný programovací jazyk. Jedná se o relativně nový programovací jazyk, prvně byl představen v roce 2011. Roku 2017 se pak Kotlin stal oficiálním programovacím jazykem pro mobilní platformu Android. Aplikace napsané v jazyce Kotlin jdou spustit téměř na všech operačních systémech a výhodou je i to, že Kotlin je zpětně kompatibilní. Oblíbeným se pak tento jazyk stal především kvůli své jednoduchosti. Kotlin

například využívá typové inference. To znamená, že pokud lze nějakým způsobem zjistit typ proměnné, například z výstupního typu volané funkce, tak se pak u deklarace proměnné typ psát nemusí a může se místo něho použít klíčové slovo var nebo val. Var se využívá pro deklaraci proměnných, jejichž hodnota se může měnit a val pro proměnné, jejichž hodnota je neměnná.

Jedním z cílů programovacího jazyka Kotlin je snadné porozumění kódu. Programy jsou krátké a pochopení kódu je snadné. Profesionální vývojáři chtějí psát stručné aplikace, protože méně kódu znamená nižší náklady na údržbu. [9]

Programovací jazyk Kotlin byl vyvíjen pragmaticky s ohledem na použitelnost. Právě z toho důvodu jsou v Kotlinu změny již v samotné syntaxi kódu, která odstraňuje nejčastěji opakované texty, které nemají žádnou přidanou hodnotu. Například psaní středníků je v Kotlinu nutné pouze pro oddělení více příkazů na jedné řádce, defaultní viditelnost je vždy public, třídy jsou v základu final, také má podporu top level funkcí a odstraňuje klíčová slova static, new, final, void, implements a extends. [10]

```
fun main(args: Array<String>){
    println("Hello world")
}
```

Obrázek 6 Ukázka kódu v jazyce Kotlin

4.3 Porovnání

Tato část práce porovnává dva výše zmíněné programovací jazyky. Porovnávání těchto jazyků je bráno z hlediska programování mobilních aplikací. Nejdříve jsou zde rozebrány výhody a nevýhody obou porovnávaných programovacích jazyků, z čehož je následně odvozen závěr, kde je uvedeno, který z těchto jazyků je pro vývoj mobilních aplikací lepší. V přílohách této práce je pak pro porovnání k dispozici ukázka stejného kódu v obou programovacích jazycích.

Mezi výhody jazyka Java patří především to, že je Java multiplatformní jazyk a funguje téměř na každém zařízení, serveru nebo operačním systému. Java je objektově orientovaný jazyk, což usnadňuje vytváření modulárních aplikací. Mezi další výhody patří také to, že je Java jednoduchá na použití, kompilování, debuggování a nasazení.

Java má ale i několik nevýhod, jednou z nich je to, že oproti některým jiným jazykům může být trochu komplikovanější. Zároveň také může být mírně pomalejší a může vyžadovat o něco více systémové paměti než jiné programovací jazyky.

I když je Java relativně jednoduchý jazyk, Kotlin je v porovnání s ní jednodušší, což je jedna z jeho hlavních výhod. Téměř jakýkoli kus kódu napsaný v jazyce Kotlin je mnohem kratší ve

srovnání s kódem v jazyce Java. Další podstatnou výhodou je bezpečnost vůči NullPointerException.

Java nemá přímou jazykovou podporu pro not-null proměnné a přidat jí do jazyka bez ztráty zpětné kompatibility nejspíše ani nepůjde. Kotlin tuto přímou podporu má a to v podobě oddělení nullable typu pomocí operátoru ?, což dovoluje například eliminovat chyby, které nastávají při změně not-null proměnné na nullable. [10]

I když je Kotlin oproti Javě jednodušší, jeho kód může být zpočátku trochu složitější na čtení a pochopení. Nevýhodou je také to, že je Kotlin relativně nový programovací jazyk a existuje k němu tedy mnohem méně výukových materiálů než třeba k Javě. Řešení problémů v tomto jazyce je tedy složitější, protože vývojářská komunita je mnohem menší a zjišťování informací o řešení konkrétního problému je tedy mnohem komplikovanější.

	Java	Kotlin
Null safety	Ne	Ano
Použití data class	Celkem složité na napsání	Pouze přidání klíčového slova v definici třídy
Smart část	Ne	Ano
Neveřejná pole	Ano	Ne
Statické členy	Ano	Ne
Wildcard types	Ano	Ne
Inline funkce	Ne	Ano
Ternární operátory	Ano	Ne
Extension funkce	Ne	Ano

Tabulka 1 - Porovnání jazyků Java a Kotlin

Jak plyne z porovnání těchto dvou programovacích jazyků, Kotlin je jednodušší a modernější, ve vývoji mobilních aplikací je tedy výhodnější použít právě Kotlin. Nevýhodou je sice menší množství výukových materiálů, ale vzhledem k tomu, že je Kotlin oficiálním programovacím jazykem pro mobilní platformu Android, bude do budoucna množství těchto materiálů stále přibývat. Již v dnešní době existuje mnoho aplikací, které Kotlin využívají, jednou z nich je například i sociální síť Pinterest.

5 LAYOUT

Tato část práce se zaobírá layoutem. Layout je základním stavebním kamenem každé aplikace. Při tvorbě mobilních aplikací je jedním z prvních kroků základní návrh toho, jak bude aplikace v budoucnu vypadat, na základě toho je pak zvolen vhodný layout. V této části práce jsou popsány základní layouty, je zde vysvětleno, jak který z nich vypadá a kdy je vhodné ho použít.

Nejprve je však důležité ujasnit si, co přesně to layout je. Layout mobilních aplikací definuje strukturu uživatelského rozhraní, které obsahuje ovládací prvky nebo widgety, které se zobrazují na obrazovce aplikace. Všechny prvky v layoutu jsou vytvořeny pomocí hierarchie View a ViewGroup objektů. View objekty jsou obvykle něco, co uživatel může vidět a komunikovat s tím. Zatímco ViewGroup je v podstatě neviditelný kontejner, který určuje strukturu rozvržení pro View a další ViewGroup objekty.

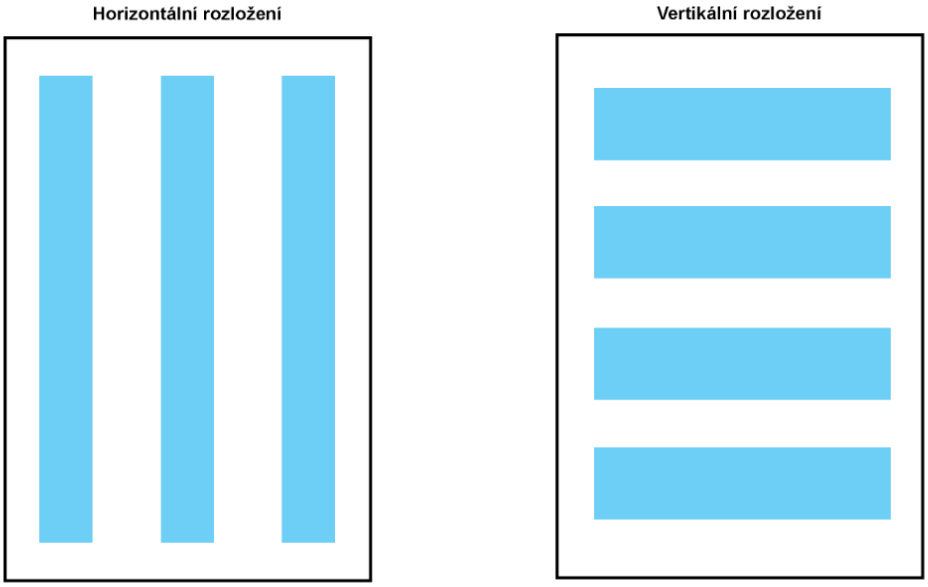
Každý layout má sadu atributů, které definují jeho vizuální vlastnosti. Všechny layouty mají několik společných atributů a pak má každý z layoutů ještě vlastní specifické atributy. Layout je pak psán pomocí XML.

Atributy	Základní popis
android:id	Slouží k jednoznačné identifikaci view
android:layout_width	Šířka layoutu
android:layout_height	Výška layoutu
android:layout_margin	Vnější odsazení
android:layout_padding	Vnitřní odsazení
android:layout_gravity	Umístění podřízených view
android:layout_x	Souřadnice x layoutu
android:layout_y	Souřadnice y layoutu

Tabulka 2 - Základní atributy layoutu

5.1 Lineární layout

Lineární layout organizuje všechny prvky buď vertikálně nebo horizontálně. Který z těchto dvou způsobů zarovnání se použije lze nastavit pomocí atributu android:orientation. Ve výchozím nastavení je orientace vodorovná. Vertikální layout má vždy pouze jeden prvek na řádku, jedná se tedy o sloupec jednotlivých řádků. Zatímco horizontální layout bude zobrazovat pouze jeden řádek prvků. Tento typ layoutu je vhodný pro aplikace, které na obrazovce mají jen několik málo prvků. Ty lze pomocí tohoto layoutu úhledně seřadit. Je sice možné zanořovat do layoutu další lineární layout, aby například v jednom řádku byly dva sloupce, ale při výskytu vícero takovýchto zanoření už je vhodné přemýšlet nad zvolením jiného typu layoutu.



Obrázek 7 Lineární layout - orientace

Jednotlivým prvkům pak lze nastavit pomocí atributu android:margin odsazení, které mezi sebou budou mít a pomocí atributu android:gravity je možné nastavit, zarovnání prvků vpravo, vlevo nebo na střed. Tento typ layoutu také umožňuje přiřazení váhy jednotlivým prvkům pomocí atributu android:layout_weight. Tento atribut přiřazuje důležitost, podle které se následně rozhoduje, kolik místa na obrazovce bude který prvek zabírat.

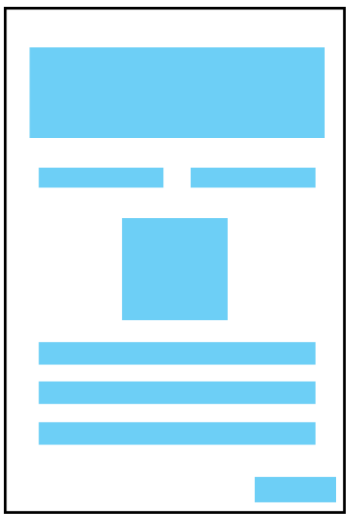
```
<LinearLayout
    android:layout_width="409dp"
    android:layout_height="729dp"
    android:orientation="vertical"
    tools:layout_editor_absoluteX="1dp"
    tools:layout_editor_absoluteY="1dp"
    tools:ignore="MissingConstraints">

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button"/>

    <Button
        android:id="@+id/button2"
```

Obrázek 8 Lineární layout - ukázka kódu

5.2 Relativní layout

Relativní layout umožňuje určit, jak jsou podřízené prvky vzájemně umístěny. Pozici každého prvku lze určit jako relativní k jeho sourozeneckým prvkům nebo vzhledem k nadřazenému

prvku. Je možné zarovnat dva prvky podle pravého ohraničení nebo vytvořit jeden pod druhým, vycentrovaný na obrazovce, vycentrovaný vlevo atd. Ve výchozím nastavení jsou všechny prvky kresleny v levém horním rohu, takže je třeba definovat polohu každého z nich pomocí různých vlastností rozvržení. Jde o jedno z nejsložitějších rozvržení a je zapotřebí několik vlastností, aby výsledné rozložení skutečně vypadalo podle představ. Opět je zde možné zanořovat do layoutu další layouty, což dělá už tak celkem složitý layout ještě složitějším. Avšak při správném používání tohoto layoutu je možné s jeho pomocí vytvořit velice pěkné aplikace. Tento typ layoutu je pak vhodný především v případě složitějších aplikací, které obsahují vícero prvků, kde každý prvek má mít své vlastní specifické umístění.



Obrázek 9 Relativní layout

Vzhledem k sourozencům je prvky možné umístit pomocí atributů android:layout_toLeftOf nebo android:layout_toRightOf nalevo nebo napravo od zadaného prvku a pomocí atributů android:layout_above a android:layout_below pod nebo nad jeho sourozence.

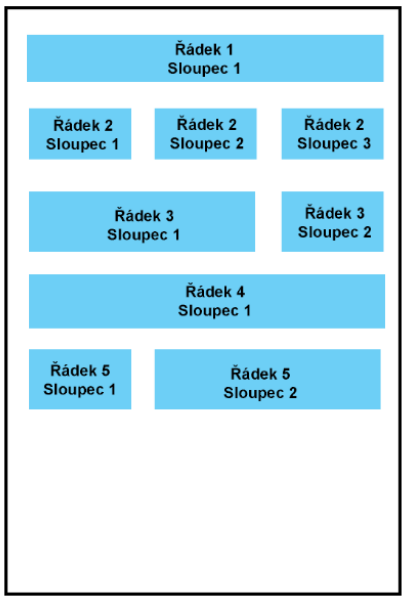
Prvek lze také umístit s ohledem na jeho nadřazený prvek, například na střed vodorovně, svisle nebo ho vycentrovat v obou směrech, také je možné prvek zarovnat s některým z okrajů nadřazeného prvku. K vycentrování prvku lze využít atributy android: layout_centerHorizontal, android: layout_centerVertical nebo android: layout_centerInParent. Pro zarovnání s některým z okrajů je pak možné využívat atribut android: layout_alignParentBottom, kde místo Bottom lze použít Left, Riht a Top.

```
<RelativeLayout 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:paddingLeft="16dp"
    android:paddingRight="16dp">

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button"
        android:layout_below="@+id/name"
```

Obrázek 10 Relativní layout - ukázka kódu

5.3 Tabulkový layout

Tabulkový layout je uspořádán stejně jako tabulka do řádků a sloupců. V layoutu však nejsou vidět čáry ohraničující jednotlivé prvky tabulky. K vytváření řádků v tabulce slouží element TableRow. Každý řádek obsahuje nula nebo více buněk, přičemž každá buňka může obsahovat pouze jeden objekt. Tabulka bude mít tolik sloupců jako řádek s největším počtem buněk. Na rozdíl od relativního layoutu není práce s tabulkovým layoutem nijak složitá, jedná se pouze o umisťování objektů do buněk tabulky. Jak už název tohoto layoutu napovídá, je vhodné ho použít ve chvíli, kdy má požadovaná aplikace mít rozložení podobné tabulce.



Obrázek 11 Tabulkový layout

V tabulkovém layoutu pak lze pracovat i se sloupci. Sloupce mohou být roztažitelné, aby se zmenšily na velikost, kdy se tabulka vejde na obrazovku, nebo smrštitelné, aby se vyplnilo dostupné místo na obrazovce. Pro práci se sloupci jsou pak k dispozici atributy android:collapceSolumns, android:shrinkColumns a android:stretchColumns.

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1">

    <TableRow>
        <TextView
            android:layout_column="1"
            android:text="Open..."
            android:padding="3dp"/>
```

Obrázek 12 Tabulkový layout - ukázka kódu

5.4 Frame layout

Tento typ layoutu je jedním z nejjednodušších pro uspořádání ovládacích prvků. Je navržen tak, aby byla blokována určitá oblast na obrazovce a zobrazována jedna položka. Využívá se především z toho důvodu, že může být obtížné zobrazit všechny prvky v určité části obrazovky, aniž by se vzájemně překrývaly. Do tohoto layoutu může být přidáno i více prvků, přičemž jejich pozice bude nastavena pomocí atributu android:layout_gravity. Prvky layoutu jsou uloženy v zásobníku a nahoře je vždy naposledy přidaný prvek. Velikost layoutu je pak určena velikostí jeho největšího prvku, ať už je viditelný či nikoliv.

Viditelnost prvků lze nastavovat pomocí atributu android:visibility, kde jsou k dispozici možnosti visible pro viditelný prvek, invisible pro neviditelný, ale stále přítomný prvek a gone pro prvek, který není viditelný ani přítomný. Důležitý je pak také atribut android:measureAllChildren. Ten určuje, zda se má při měření počítat i s prvky, které mají stav gone. Pokud je hodnota atributu měřených prvků nastavena na true, bude zobrazena skutečná šířka a výška rozvržení rámečku, i když je viditelnost pohledů ve stavu gone.

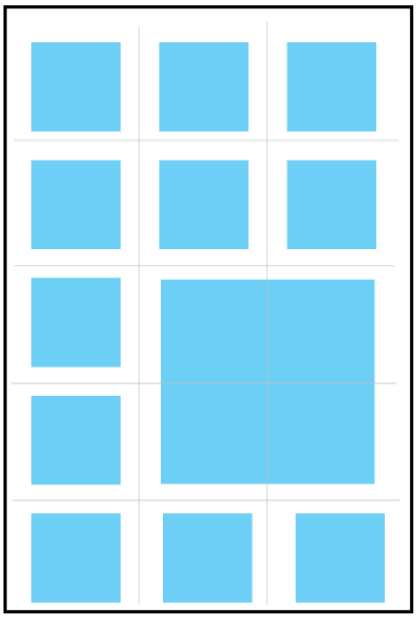
```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:measureAllChildren="true">

    <ImageView
        android:id="@+id/imageview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

Obrázek 13 Frame layout - ukázka kódu

5.5 Grid layout

Jedná se o typ layoutu, ve kterém jsou prvky umístěny do obdélníkové mřížky. Je možné určit počet sloupců a řádků dané mřížky. Také lze u layoutu přizpůsobit velikost, barvy nebo okraje.

Mřížka se skládá ze sady tenkých čar, které od sebe oddělují oblasti jednotlivých buněk. Při předpokladu, že má mřížka N sloupců, bude mít N + 1 mřížkových indexů, které začínají od 0. Index 0 je fixován k přední hraně a index N k zadní hraně bez ohledu na to, jak je layout nakonfigurován.



Obrázek 14 Grid layout

Prvky zabírají jednu nebo více souvislých buněk, to je definováno pomocí parametrů rowSpec a columnSpec. Každá specifikace definuje sadu řádků nebo sloupců, které mají být obsazeny a také to, jakým způsobem by měly být prvky zarovnány do výsledné skupiny buněk. Prostor mezi buňkami může být vytvořen buď pomocí elementu Space nebo nastavením leftMargin, topMargin, rightMargina bottomMargin. Počet řádků a sloupců dané mřížky pak lze nastavit pomocí atributů android:rowCount a android:columnCount.

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:rowCount="4"
    android:columnCount="2"
    android:orientation="vertical">

    <TextView
        android:text="Cell 0"
        android:textSize="14dip"
        android:layout_row="0"
        android:layout_column="0"/>
    <Space
        android:layout_row="1"
```

Obrázek 15 Grid layout - ukázka kódu

6 ZÁKLADNÍ KOMPONENTY

V této části práce budou popsány základní komponenty, které lze v Android Studiu používat, a základní práce s těmito komponentami. Komponenty slouží k základnímu zobrazování informací. Pro různé komponenty je pak možné nastavit jejich chování třeba po kliknutí. Mezi základní komponenty, které bude tato práce popisovat, patří například Button, Toast, Spinner nebo TextView.

6.1 Button

Jedním z nejpoužívanějších prvků v mobilních aplikacích jsou tlačítka. Na tlačítka mohou uživatelé kliknout a po kliknutí očekávají, že se provede určitá akce. Práce s tlačítky je velmi snadná, stačí si přidat tlačítko do mobilní aplikace a v kódu se pak na tlačítko odkazovat. Poté existuje jednoduchá metoda, která obstarává, co se má stát po kliknutí na dané tlačítko. Pak pouze stačí naprogramovat kód, který do této metody vložíme, nebo se na něj v metodě odkážeme. Pokud je vše správně nastaveno, tak se po stisknutí tlačítka provede daný kód.

Tlačítko lze vytvořit třemi způsoby, záleží na jeho výsledné podobě. Může se jednat buď pouze o text, pouze o ikonu, anebo může jít o kombinaci obojího. V závislosti na tom je pak napsán XML kód v layoutu. Pokud má být výsledkem obyčejné textové tlačítko je použit běžný Button, který obsahuje atribut android:text, ve kterém je napsáno to, co se bude zobrazovat na daném tlačítku. Pro tlačítko s ikonou je použit ImageButton s atributem android:src, který obsahuje odkaz na ikonu. A pro tlačítko s ikonou i textem je použit běžný Button s atributem android:drawableLeft, do kterého je přidána daná ikona.

Obrázek 16 Button - možnosti zobrazení

V kódu se pak na všechny objekty z layoutu používá odkazování pomocí id. Jak je již zmíněno v předešlé kapitole, každé View v layoutu má id atribut, který je určen pro jeho přesnou identifikaci. V kódu se pak používá metoda findViewById, které se předá dané id a propojí se tak prvek z layoutu s prvkem v kódu. To, co bude vykonáno po kliknutí na tlačítko, lze nastavit buď rovnou v layoutu pomocí atributu android:onClick, kterému je přiřazena metoda, která se má po kliknutí na tlačítko vykonat. Druhým a zřejmě používanějším způsobem

je metoda setOnClickListener, která obsahuje kód, případně volání jiných metod, který se má vykonat po kliknutí na tlačítko.

```
var button:Button = findViewById<Button>(R.id.button)
button.setOnClickListener { it: View!
    //Kód co se má stát po kliknutí
}
```

Obrázek 17 Button - ukázka kódu

6.2 TextView

Dalším velice často používaným prvkem je TextView.Tento prvek slouží pro zobrazování textu. V aplikacích se vyskytuje téměř všude, kde je potřeba zobrazovat nějaký text, je tedy opravdu velmi používaný. Možností pro zobrazení textů je celá řada, ale TextView je tím nejzákladnějším z nich. Text lze vložit do prvku přímo v XML nebo ho lze vložit až později pomocí kódu. Zobrazovaný text pak lze pomocí kódu měnit za běhu aplikace, například při zavolání nějaké události. Práce s TextView je velice jednoduchá, nejsložitější bývá většinou nastavení vzhledu.

Samotný text je možné nastavit pomocí atributu android:text. Je také možné upravovat velikost textu a jeho barvu pomocí atributů android:textSize a android:textColor. Stejně jako u všech ostatních prvků i u TextView je důležitým atributem android:id. Pro propojení prvku z layoutu s proměnnou v kódu se opět využívá findViewById. Text, který je v TextView zobrazen, lze nastavit pomocí tečkové notace a slova text, za kterým následuje znaménko rovná se a v uvozovkách pak nový text. Pokud je v kódu tento způsob nastavení textu vícekrát, bude se zobrazovat vždy ten text, který byl v kódu vykonán naposledy, protože se text vždy přepíše. Pokud by měl být za předchozí text pouze přidán text nový, lze k tomu využít metodu append.

```
var textView : TextView!  = findViewById<TextView>(R.id.textView)
textView.text = "Text 1"
textView.append("Text 2")
```

Obrázek 18 TextView - ukázka kódu

Dříve se u TextView používaly atributy, které specifikovaly, o jaký typ textu půjde. Jednalo se například o atributy android:editable, android:password nebo android:phoneNumber. Tyto a podobné atributy jsou však již zastaralé a nahrazují se samostatným prvkem. Jedná a o prvek EditText, který obsahuje atribut android:inputType. Tento atribut pak určuje, o jaký typ textu se jedná, může jít pouze o běžný text, v takovém případě je atributu přiřazeno slovo text, v případě hesla je pak použito slovo password a třeba v případě telefonního čísla slovo phone. Možností je celá řada. Tento prvek slouží pro vkládání textu uživatelem. Stejně jako funguje

nastavení textu buď přes tečkovou notaci, nebo přímo v XML, je možné stejným způsobem nastavit nápovědu. Nápověda je text, který se zobrazuje uživateli před kliknutím do editovatelného pole. Pro nastavení nápovědy pak slouží klíčové slovo hint. Získání textu, který uživatel zadal, je pak velmi jednoduché, stačí pouze v kódu použít klíčové slovo text a tečkovou notaci.

6.3 Toast

V případě Toast se jedná o prvek, který nelze nastavit pomocí XML. Jedná se o malé vyskakovací okénko s textem, na které nelze klikat. Většinou se používá pro zpětnou vazbu po nějaké události. Zabírá pouze množství místa potřebného pro zprávu a aktuální aktivita zůstává interaktivní a viditelná. Toast automaticky zmizí po uplynutí časového limitu.

Vytvoření této vyskakovací zprávy je velice jednoduché. Stačí vytvořit instanci Toast a zavolat metodu makeText. Tato metoda má celkem tři parametry, prvním z nich je kontext, což jsou v podstatě informace o prostředí aplikace. Druhým parametrem je textová zpráva a poslední z parametrů je délka trvání. Toast je pak možné zobrazit zavoláním metody show.

```
val text = "Toto je toast"
val dobaTrvani : Int  = Toast.LENGTH_SHORT
val toast : Toast!  = Toast.makeText(this.applicationContext, text, dobaTrvani)
toast.show()
```

Obrázek 19 Toast - ukázka kódu

Toast se automaticky zobrazuje uprostřed v dolní části obrazovky. To je možné změnit pomocí metody setGravity. Tato metoda má tři parametry, všechny jsou celočíselné hodnoty. Prvním parametrem je konstanta pro umístění v rámci obrazovky, následují čísla určující posunutí polohy x a y.

6.4 ListView

Jedním z často používaných prvků je i ListView. Jedná se o rolovatelný seznam, který obsahuje určitou skupinu položek. Prvky v seznamu jsou zobrazeny pod sebou a je možné jednotlivé položky vybírat. Práce s tímto prvkem je však trochu komplikovanější, než práce s již zmíněnými prvky.

K vyplnění dat v ListView se využívají adaptéry. Položky seznamu se automaticky vloží do seznamu pomocí adaptéru, který pak načte obsah ze zdroje. Adaptér funguje jako most mezi AdapterView a daty k zobrazení, poskytuje tedy přístup k datovým položkám. K dispozici je buď ArrayAdapter nebo BaseAdapter. Každý z nich se hodí pro jiný typ položek. BaseAdapter umožňuje dělat téměř cokoli, vyžaduje to však trochu více kódu. Je vhodný pro vytváření

vlastního adaptéru pro zobrazení vlastních položek seznamu. ArrayAdapter je vhodný pro práci s daty, která jsou uchovávána v polích. U adaptéru je důležité správné přiřazení layoutu, na obrázku níže je možné vidět použití již předdefinovaného layoutu přímo pro ListView. Pokud je vytvořen adaptér, stačí ho pomocí tečkové notace a klíčového slova adapter přiřadit k ListView.

```
var list : ListView! = findViewById<ListView>(R.id.list)
val poleCisel1 : Array<String> = arrayOf("jedna", "dva", "tri")
var adapter1 = ArrayAdapter<String>( context: this, android.R.layout.simple_list_item_1, poleCisel1)
list.adapter = adapter1
```

Obrázek 20 ListView - ukázka kódu

Pokud se má vykonat nějaký kód po kliknutí na položku v seznamu, je potřeba k listView přidat metodu setOnItemClickListener. V ní je pak uveden kód, který se má vykonat po kliknutí na vybranou položku.

6.5 ProgressBar

Tento prvek není v mobilních aplikacích využíván tak často jako předchozí prvky, ale i přesto se v mobilních aplikacích vyskytuje celkem často. Většinou složí pro zobrazení aktuálního stavu nějaké události, jako příklad lze uvést například stahování souborů. Tento prvek má dva základní styly, je buď zobrazen jako horizontální obdélníček nebo jako kolečko. O tom, která z těchto dvou možností bude využita, rozhoduje atribut style.

Dále lze rozlišovat indeterminate progres, který se využívá, pokud není předem známá délka operace, na obrazovce se tak například zobrazuje stále se točící kolečko. Druhou možností je determinate progres, ten je využíván, pokud má být zobrazen přesný stav. Tyto dvě možnosti se nastavují pomocí atributu android:indeterminate, který má hodnotu typu boolean.

```
var progressBar : ProgressBar! = findViewById<ProgressBar>(R.id.progressBar)
progressBar.max = 100
progressBar.progress = 25
```

Obrázek 21 ProgressBar - ukázka kódu

Pokud je nastaveno, že se má zobrazovat přesný stav, používá se buď atribut android:progress, nebo se stav nastavuje přímo v kódu pomocí slova progress a tečkové notace. Úplně stejným způsobem se nastavuje také maximální hodnota, u té je použito klíčové slovo max nebo atribut android:max.

6.6 Spinner

Méně častým prvkem je Spinner. I přesto je však celkem praktický a určitě se pro něj v mnoha mobilních aplikacích najde vhodné využití. Jedná se o rozbalovací nabídku, která poskytuje

uživateli rychlý způsob, jak vybrat jednu hodnotu z určité sady. Ve výchozím stavu je zobrazena aktuálně vybraná hodnota, pokud není zatím žádná hodnota vybrána, zobrazuje se první možnost ze sady. Klepnutím na tuto položku se zobrazí rozevírací nabídka se všemi ostatními dostupnými hodnotami, ze kterých si uživatel může vybrat novou. Uživatel může vždy vybrat pouze jednu položku.

```
var spinner : Spinner! = findViewById<Spinner>(R.id.spinner)
var poleCisel : Array<String> = arrayOf("jedna", "dva", "tri")
var adapter = ArrayAdapter<String>( context: this, android.R.layout.simple_spinner_dropdown_item, poleCisel)
spinner.adapter = adapter
```

Obrázek 22 Spinner - ukázka kódu

Práce s tímto prvkem je opět trochu komplikovanější. Spinner stejně jako ListView využívá adaptér. Je znovu možné využít již dříve zmíněný ArrayAdapter. Důležité je pak u adaptéru neopomenout správné přiřazení layoutu. Na obrázku níže lze vidět přiřazení již předdefinovaného layoutu pro Spinner. V momentě, kdy je adaptér vytvořen, se pak velice jednoduše pomocí klíčového slova adapter a tečkové notace přiřadí k prvku Spinner.

Pro získání vybrané položky je pak možné použít tečkovou notaci a klíčové slovo selectedItem. Případně pak ještě zavolat metodu toString, pokud by měl výstupem být text.

Obrázek 23 ProgressBar, Spinner, Toast - ukázka

6.7 ImageView

ImageView slouží v aplikacích běžně k zobrazování obrázků. Používá se k zobrazení jakýchkoli obrazových prostředků, ať už jde o android.graphics.Bitmap nebo o android.graphics.drawable.Drawable. Také se dá využít k aplikaci odstínů na obrázek a ke zpracování škálování obrázků.

U tohoto prvku je důležitá především práce s atributy. Pro přiřazení konkrétního obrázku do ImageView se využívá atribut android:src. Pro správnou velikost a rozložení obrázku v ImageView jsou důležité především atributy android:layout_width, android:layout_height, android:scaleType, android:adjustViewBounds a android:cropToPadding. První dva z těchto atributů již byly v práci zmíněny, používají se pro nastavení výšky a šířky prvku, v tomto případě nastavují šířku a výšku ImageView. Dalším atributem je android:scaleType, tento

atribut řídí, jak by měla být upravena nebo přesunuta velikost obrázku, tak aby odpovídala velikosti ImageView. Atribut android:adjustViewBounds pak určuje, zda má ImageView upravit své hranice tak, aby zachoval poměr stran obrázku. To se stane, pokud je hodnota nastavena na true. Atribut android:cropToPadding má také hodnoty true a false, v případě že je nastaven na true, se obrázek ořízne, aby se vešel do ImageView.

Při nastavování ImageView je důležité si vyzkoušet různé varianty a vybrat tu, která bude odpovídat požadavkům. Například atribut android:scaleType má osm možných variant, přičemž každá přizpůsobí obrázek jiným způsobem. Nelze napsat, která z možných kombinací atributů je ta nejvhodnější, protože každý uživatel očekává trochu jiné chování obrázku v ImageView. Je tedy potřeba si vyzkoušet různé možnosti společně s kombinací dalších atributů a na základě toho zvolit vhodnou variantu.

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="350dp"
    android:layout_height="300dp"
    app:srcCompat="@drawable/picture1"
    android:scaleType="fitXY"
    android:cropToPadding="true"
    android:adjustViewBounds="true"
    tools:ignore="MissingConstraints" />
```

Obrázek 24 ImageView - ukázka kódu

7 POKROČILEJŠÍ PRÁCE S APLIKACÍ

Tato část práce se věnuje pokročilejším technikám ve vývoji mobilních aplikací. Bude zde popsáno vytváření nových oken, vytvoření vyskakovacího okna a následně i práce s technologiemi jako například s Bluetooth.

7.1 Vytvoření nového okna

Každá aplikace se skládá z jednoho nebo více oken, přičemž každé okno pak může obsahovat komponenty. Mezi těmito okny je běžně možné v aplikaci přepínat například pomocí tlačítek. Přepnutí okna je většinou vyvoláno jakožto reakce na nějakou událost, jak již bylo zmíněno, může jít o kliknutí na tlačítko, nebo se nové okno zobrazí třeba po dokončení nějaké funkce. Při přepnutí do nového okna je zároveň možné poslat do nově vytvořeného okna nějaké údaje z okna starého.

Samotné vytvoření nového okna je vcelku jednoduché. Stačí vytvořit nový intent (objekt s informacemi pro systém, který obsahuje popis operace, která má být provedena) a v parametru zadat, kam se má okno přepnout. Intent má dva parametry, přičemž první z nich je kontext a druhý je třída. Kontext jsou informace o prostředí aplikace a třída je právě ten parametr, který určuje, do jaké třídy se aplikace přepne. Pro zasílání zpráv mezi okny pak existuje metoda putExtra, ta se zavolá pomocí tečkové notace na vytvořený intent. Tato metoda má dva parametry a to jméno a hodnotu. Pomocí jména je pak možné se v novém okně dotazovat na hodnotu. A hodnota pak obsahuje data, která budou předána do nového okna. Poté již pouze stačí zavolat metodu startActivity a do parametru jí předat vytvořený intent. Jakmile se tato metoda vykoná, dojde k přepnutí do nového okna.

```
var intent: Intent = Intent( packageContext: this, ResultActivity::class.java)
var pocetBodu = 10
intent.putExtra( name: "body",pocetBodu)
startActivity(intent)
```

OKNO 1

```
val bundle:Bundle? = intent.extras
val pocetBodu : Int  = bundle!!.getInt( key: "body")
```

OKNO 2

Obrázek 25 Vytvoření nového okna

Aby se v novém okně zobrazila zpráva, která byla poslána z předchozího okna, je zapotřebí použít extras a následně zavolat metodu get, která má jako parametr právě výše zmíněné jméno. Extras jsou prvky typu klíč a hodnota, přičemž klíč je v tomto případě to samé co jméno. Také je možné sloučit použití extras a následné zavolání get metody do jednoho kroku a použít metodu getExtra, která má opět jako parametr výše zmíněné jméno. Právě díky jménu, nebo

také klíči, lze získat hodnotu, která byla předána z jednoho okna do druhého. Tuto hodnotu je možné uložit do proměnné a pak s ní dále pracovat.

7.2 Vyskakovací okno

Vyskakovací okna se v mobilních aplikacích používají většinou, pokud má uživatel před pokračováním funkce zadat nebo potvrdit nějakou informaci. Většinou zabírají pouze část obrazovky. Jsou vytvářena pomocí třídy Dialog, nejvíce používaný je pak AlertDialog. Vyskakovací okna mají celou řadu možných nastavení od vcelku jednoduchého rozhodování, přes zadávání textu až po výběr ze seznamů. Tato práce se zaměří především na jednoduché vyskakovací okno s dvěma tlačítky - ano a ne.

Vyskakovací okna jsou vytvářena pomocí builderu, kterému je v parametru předán kontext. Je důležité zmínit i některé metody, kterými lze nastavovat obsah vyskakovacího okna. Důležité jsou především metody setTitle a setMessage, přičemž první z nich nastavuje titulek okna a druhá, ta důležitější z nich, nastavuje zobrazovanou zprávu. Dále je možné nastavit až tři tlačítka pomocí metod setPositiveButton, setNegativeButoon a setNeutralButton. Tyto metody mají v parametru text, který se bude zobrazovat na tlačítku a lze jim jako další parametr přiřadit i listener, který obsahuje kód s tím, co se má stát po kliknutí na jednotlivá tlačítka. Pokud to není zadáno parametrem, lze to zapsat ihned za vytvoření tlačítka. Pomocí parametru je to však přehlednější, zvláště pokud jde o delší kód. Velice důležitá je pak metoda show, která zařídí zobrazení vyskakovacího okna.

Pokud by mělo jít o vyskakovací okno se seznamem, z kterého si bude uživatel vybírat, tak by na builder byla zavolána metoda setItems. Této metodě by bylo přiřazeno pomocí parametru pole obsahující položky a případně opět listener. V tomto případě by se jednalo o seznam, který má pouze jednu možnost výběru. V případě výběru více položek ze seznamu by byla použita metoda setMultiChoiceItems. Tato metoda opět obsahuje v parametru pole s položkami a může mít i druhý parametr, kterým by byl opět listener.

Trochu složitější je pak možnost vyskakovacího okna, do kterého se zadává text. K tomu je nutné mít vytvořenou XML aktivitu, ve které je definovaná komponenta EditText. V kódu se pak pomocí builderu zavolá metoda setView. Tato metoda má jeden parametr a ten obsahuje inflater. To je v podstatě služba zodpovědná za definování layoutu. Pomocí inflateru je pak zavolána metoda inflate, která má dva parametry, prvním je layout ve kterém je definovaný EditText a druhým parametrem je nadřazené zobrazení. Vše lze zapsat do metody setView, ale pro přehlednost je lepší si to rozdělit na několik kroků.

Výše pak byl zmíněný listener, který lze vložit jako parametr například u tlačítek. V tom případě by se jednalo o DialogInterface.OnClickListener. Při vyskakovacím okně s tlačítky by pak uvnitř mohl být kód pro každé tlačítko.

```
val dialogClickListener = DialogInterface.OnClickListener { dialog, which ->
        when (which) {
            DialogInterface.BUTTON_POSITIVE -> { //kód pro tlačítko ANO
            }
            DialogInterface.BUTTON_NEGATIVE -> { //kód pro tlačítko NE
            } } }

val builder: AlertDialog.Builder = AlertDialog.Builder( context: this)
builder.setMessage("Chcete pokračovat?")
builder.setPositiveButton( text: "Ano", dialogClickListener)
builder.setNegativeButton( text: "Ne", dialogClickListener)
builder.show()
```

Obrázek 26 Ukázka AlertDialogu

7.3 Práce s Bluetooth

Bluetooth je rozšířená technologie, kterou má každé mobilní zařízení. V dnešní době není tato technologie v mobilních aplikacích využívána úplně často, spíše se dnes využívá k propojení mobilního zařízení například s chytrými hodinkami, přičemž se pak v aplikaci zobrazují získaná data. Je tedy dobré znát základy práce s Bluetooth, čemuž je věnován právě tento úsek práce a zároveň i jedna z aplikací. Bude zde popsáno především spárování dvou zařízení a komunikace mezi nimi.

Android zahrnuje podporu Bluetooth, což umožňuje zařízení bezdrátově vyměňovat data s jinými zařízeními právě pomocí Bluetooth. Přístup k funkcím Bluetooth je poskytován prostřednictvím rozhraní Android Bluetooth API. Tato rozhraní API umožňují aplikacím bezdrátově se připojit k jiným zařízením Bluetooth a umožňují bezdrátové funkce point-to-point a vícebodové bezdrátové připojení. Pomocí rozhraní Bluetooth API může aplikace pro Android vyhledávat další zařízení Bluetooth, spárovat zařízení, přenášet data do a z jiných zařízení a spravovat více připojení. [12]

Aby bylo možné v aplikaci využívat funkce Bluetooth je nutné deklarovat určitá oprávnění. Tato oprávnění jsou deklarována v souboru AndroidManifest. Jedná se především o povolení BLUETOOTH a ACCESS_FINE_LOCATION. V závislosti na požadavcích na aplikace a na verzi mobilního operačního systému Android se pak liší ještě další dodatečná oprávnění.

Než bude aplikace moci komunikovat přes Bluetooth je třeba ověřit, zda je Bluetooth v zařízení podporováno a pokud je podporováno, tak je třeba se ujistit, že je povolené. Pokud je Bluetooth podporováno ale deaktivováno, je možné požádat uživatele o jeho povolení přímo v aplikaci. [12]

Aby to bylo možné, je zapotřebí nastavit BluetoothAdapter. Pro jeho získání je třeba zavolat metodu getDefaultAdapter. Pomocí metody isEnabled je pak třeba zjistit, zda je Bluetooth aktuálně povoleno. Pokud metoda vrátí hodnotu false, znamená to, že je Bluetooth deaktivováno. Pro jeho aktivaci je možné využít metodu startActivityForResult. Tato metoda má dva parametry, prvním z nich je Bluetooth Intent, který je také potřeba vytvořit a druhým parametrem je REQUEST_ENABLE_BT. Uživateli se následně v aplikaci zobrazí okno, které si žádá zapnutí Bluetooth, pokud to uživatel odsouhlasí, mělo by následně být Bluetooth povoleno.

```
var btAdapter : BluetoothAdapter! = BluetoothAdapter.getDefaultAdapter()
if(!btAdapter!!.isEnabled){
    var enableBluetoothIntent: Intent = Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE)
    startActivityForResult(enableBluetoothIntent, REQUEST_ENB_BT)
}
```

Obrázek 27 Povolení Bluetooth

Další důležitou částí je spárování zařízení pomocí Bluetooth. Pro zobrazení spárovaných zařízení je třeba uložit všechna spárovaná zařízení do pole pomocí Blutooth adaptéru a metody bondedDevices. Pak stačí pole vypsat například pomocí ListView. Pro zobrazení všech dostupných zařízení a následné spárování s některým z nich je opět potřeba využít adaptér. Pomocí něho se následně zavolá metoda startDiscovery, která, jak plyne z jejího názvu, prozkoumává okolí a hledá další zařízení se zapnutým Bluetooth. Dále je zapotřebí BroadcastReceiver, aby bylo možné přijímat informace o objevených zařízeních. Pokud v tomto receiveru nastane akce ACTION_FOUND, je možné si uložit informace o nalezeném zařízení. Pro následné spárování je třeba vybrat zařízení, se kterým se bude navazovat propojení a následně zařízení spárovat pomocí metody createBond.

Nejsložitější je pak nastavení vzájemné komunikace mezi dvěma zařízeními. Je zapotřebí nějakým způsobem určit, které zařízení bude komunikovat na straně serveru a které na straně klienta. Aby komunikace dobře fungovala, musí být implementovány jak mechanismy na straně serveru, tak na straně klienta. Serverové zařízení a klientské zařízení získávají požadované BluetoothSocket různými způsoby. Server a klient se považují za vzájemně propojené, pokud mají každý připojený BluetoothSocket ke stejnému kanálu RFCOMM. V tomto okamžiku může každé zařízení získat vstupní a výstupní toky a může začít přenos dat.

Obecný postup pro přenos dat je následující. Je třeba získat InputStream a OutputStream, které zpracovávají přenosy přes soket pomocí getInputStream a getOutputStream. Čtení a zápis dat do streamů pak probíhá pomocí read(byte[]) a write(byte[]). Data jsou pak odesílána a zpracovávána pomocí handleru. [12]

Spárování dvou mobilních zařízení a jejich následná komunikace je na vysvětlení a podrobné popsání mírně složitější. Pokud by tedy popis v této kapitole nebyl dostačující, je možné si zobrazit a důkladněji prostudovat kódy v aplikaci Bluetooth piškvorky, která je součástí této bakalářské práce a je k ní tedy přiložena. Aplikace je zároveň popsána níže v této práci a zabývá se celkovou problematikou práce s Bluetooth technologií, konkrétně povolením Bluetooth, zobrazením a spárováním zařízení a především vzájemnou komunikací dvou zařízení.

7.4 Práce s kamerou

V dnešní době je v mobilních aplikacích velice často využívána práce s mobilním fotoaparátem. Právě z toho důvodu je základní práci s kamerou věnována i část této práce. Bude zde popsáno pouze jednoduché pořízení a zobrazení fotografií. Je však možné využít fotoaparát i k nahrávání videí, lze videa a fotografie následně ukládat anebo případně načítat fotografie ze zařízení. Pro začínající vývojáře je však nejpodstatnější povolení kamery a následné pořízení a zobrazení fotografií, proto je tato část věnována právě tomu.

Android zahrnuje podporu pro různé fotoaparáty a funkce fotoaparátů dostupné na zařízeních, což umožňuje pořizovat obrázky a videa v aplikacích. Před zahájením vývoje aplikace pomocí rozhraní Camera API je třeba se ujistit, že manifest obsahuje příslušná povolení, která umožňují použití hardwaru fotoaparátu a dalších souvisejících funkcí. Především povolení kamery pomocí CAMERA a povolení funkcí fotoaparátu pomocí feature android.hardware.camera. [13]

Android má způsob delegování akcí na jiné aplikace, v tomto případě to znamená vyvolat Intent, který popisuje, co se má udělat. Tento proces se skládá ze tří částí. První z nich je samotný Intent, poté následuje volání ke spuštění externí aktivity a pak kód pro zpracování obrazových dat. [13]

Nejdříve je tedy potřeba samotné vytvoření Intentu, přičemž v parametru je ACTION_IMAGE_CAPTURE, což je standardní akce Intent, kterou lze odeslat, aby aplikace fotoaparátu zachytila obrázek a vrátila jej. Pak následuje metoda startActivityForResult, která má v parametru výše vytvořený Intent a konstantu.

Pro následné zobrazení pořízené fotografie je možné využít například ImageView. Mobilní fotoaparát zakóduje fotografii a ve zpáteční cestě je tak Intent doručen metodě onActivityResult jako malá Bitmapa pod klíčem data. Je tedy nutné pracovat s touto Bitmapou. Je zapotřebí ji rozkódovat pomocí extras a následně je možné uložit jí do ImageView a zobrazit tak pořízenou fotografii. Právě to lze vidět i na obrázku 27.

```
    photoButton.setOnClickListener{ it: View!
        val takePictureIntent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
            startActivityForResult(takePictureIntent, CAMERA_REQUEST)
    }
}

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == CAMERA_REQUEST && resultCode == RESULT_OK) {
        val imageBitmap : Bitmap  = data?.extras?.get("data") as Bitmap
        imageView?.setImageBitmap(imageBitmap)
    }
}
```

Obrázek 28 Ukázka práce s kamerou

7.5 Práce se senzory

Poslední část této kapitoly bude zaměřena na práci se senzory. Většina mobilních zařízení se systémem Android má zabudované senzory, které měří pohyb, orientaci a různé podmínky prostředí. Tyto senzory jsou schopny poskytovat nezpracovaná data s vysokou přesností, jsou tedy užitečné pro sledování pohybu, umístění zařízení nebo sledování změn v okolním prostředí.

I když to možná není na první pohled patrné, senzory jsou v mobilních aplikacích využívány opravdu často. Například hra může sledovat odečty z gravitačního senzoru zařízení a odvodit tak složitá uživatelská gesta a pohyby, jako je třeba naklonění, otřes, otočení nebo švih. Podobně aplikace pro počasí může k výpočtu a hlášení použít teplotní senzor a senzor vlhkosti, nebo aplikace pro kompas může použít senzor geomagnetického pole a akcelerometr právě pro hlášení kompasu. [14]

Platforma Android podporuje tři kategorie senzorů:

- Pohybové senzory – Tyto senzory měří sílu zrychlení a sílu rotace podél tří os. Tato kategorie zahrnuje akcelerometry, gravitační senzory, gyroskopy a rotační vektorové senzory.
- Senzory prostředí – Tyto senzory měří různé parametry prostředí, jako je teplota a tlak okolního vzduchu, osvětlení a vlhkost. Tato kategorie zahrnuje barometry, fotometry a teploměry.
- Senzory polohy – Tyto senzory měří fyzickou polohu zařízení. Tato kategorie zahrnuje orientační senzory a magnetometry.

Některé z těchto senzorů jsou hardwarové a některé softwarové. Hardwarové senzory jsou fyzické součásti zařízení. Odvozují svá data přímým měřením specifických vlastností prostředí, jako je zrychlení nebo síla geomagnetického pole. Softwarové senzory odvozují svá data z jednoho nebo více hardwarových senzorů a někdy se jim říká virtuální senzory nebo

syntetické senzory. Jen málo zařízení se systémem Android má všechny typy senzorů. Například většina telefonů a tabletů má akcelerometr a magnetometr, ale méně zařízení má barometry nebo teploměry. [14]

Vzhledem k velkému množství různých senzorů bude v této práci popsána pouze základní práce se senzory a jeden konkrétní příklad týkající se práce s pohybovým senzorem.

Pro práci se senzory je důležitý SensorManager pro seznam senzorů a přístup k nim. Instanci SensorManager lze získat voláním metody getSystemService a předáním konstanty SENSOR_SERVICE v parametru této metody. Seznam všech senzorů v zařízení je možné získat voláním metody getSensorList, která má v parametru konstantu TYPE_ALL. Místo této konstanty lze použít i jiné v závislosti na tom, jaký je požadovaný typ senzorů, který má být vypsán. Je také možné zjistit, zda na zařízení existuje konkrétní typ senzoru, pomocí metody getDefaultSensor, které je předána konstanta typu pro konkrétní senzor. Pokud má zařízení více než jeden senzor daného typu, musí být jeden ze senzorů označen jako výchozí senzor. Pokud pro daný typ senzoru neexistuje výchozí senzor, volání metody vrátí hodnotu null, což znamená, že zařízení tento typ senzoru nemá. [14]

Třída Sensor pak poskytuje metody pro získání informací o senzoru, jako je například název senzoru, typ senzoru nebo rozlišení senzoru. Jako příklad lze uvést metody getResolution a getMaximumRange k získání rozlišení senzoru a maximálního rozsahu měření. Pro sledování dat ze senzorů je třeba využít rozhraní SensorEventListener, které má metody onAccuracyChanged a onSensorChanged. První ze zmíněných metod se volá při změně přesnosti senzoru a druhá se volá při změně hodnot.

Jako konkrétní příklad zde byla zvolena práce s akcelerometrem, konkrétně vypisování hodnot os mobilního zařízení, tedy osy x, y a z. Nejdříve byl získán SensorManager pomocí getSystemService, jak je již popsáno výše. Poté byl získán seznam senzorů, ale v parametru metody getSensorList byla použita konstanta TYPE_ACCELEROMETER, aby byly získány senzory pouze typu akcelerometr. Dále byl pak pro SensorManager zaregistrován listener, který byl vytvořen pomocí již zmíněného rozhraní SensorEventListener. Uvnitř listeneru byla přepsána metoda onSensorChanged, aby se vždy při změně vypisovaly aktuální hodnoty senzoru. Hodnoty jsou pak na obrazovku vypisovány pouze jako text, to by se však dalo upravit a hodnoty vypisovat jinak, nebo by se například mohla pomocí souřadnic vytvořit jednoduchá hra. Podobným způsobem by pak probíhala práce i s dalšími senzory. Jak je již napsáno výše, je důležité nastavit typ senzorů, se kterými se má v aplikaci pracovat, v metodě getSensorList.

```
sm = getSystemService(Context.SENSOR_SERVICE) as SensorManager
textView1 = findViewById(R.id.textView1)
list = sm!!.getSensorList(Sensor.TYPE_ACCELEROMETER)
if(list.isNotEmpty()){
    sm!!.registerListener(sel, list.get(0), SensorManager.SENSOR_DELAY_NORMAL)
}

var sel: SensorEventListener = object : SensorEventListener {
    override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {}
    override fun onSensorChanged(event: SensorEvent) {
        val values : FloatArray! = event.values
        textView1!!.text = """x: ${values[0]} y: ${values[1]}z: ${values[2]}""".trimIndent()
    }
}
```

Obrázek 29 Práce se senzory

8 APLIKACE KALKULÁTOR

První aplikací je jednoduchý mobilní kalkulátor, který se zabývá především otázkou vhodného rozvržení a designu. Hlavním cílem této aplikace je pomocí vzhledu poskytnout jednoduché a přehledné ovládání pro všechny typy uživatelů. Nejsložitějším úkolem tedy bylo zvolení vhodného layoutu.

Rozvržení aplikace bylo navrženo podobně jako je tomu u běžných mobilních kalkulátorů, v horní části obrazovky se zobrazují výpočty a jejich výsledky a pod tímto oknem jsou v několika řádcích vedle sebe umístěna tlačítka pro ovládání kalkulátoru. Toto rozložení je možné implementovat pomocí vícero různých layoutů. Vzhledem k tomu, že jsou tlačítka zobrazena v řádcích a sloupcích je možné využití tabulkového layoutu. Kvůli hornímu panelu s výpočty a výsledky byl však nakonec zvolen lineární layout. Respektive několik lineárních layoutu vnořených do jednoho hlavního.

Celá ovládací část s tlačítky je jeden lineární layout a každý řádek tlačítek je pak reprezentován dalším lineárním layoutem. Výhodou je v tomto případě to, že kdyby byl v některém z řádků jiný počet tlačítek, nemusí se nic přepisovat a tlačítka se vždy automaticky zarovnají a vyplní celou šířku řádku. Právě z důvodu budoucích úprav aplikace byl zvolen tento typ layoutu. Do budoucna je tedy velice snadné přidat další matematické operace a s tím i další tlačítka, aniž by se musel upravovat celý layout. Což by v případě tabulkového layoutu bylo trochu komplikovanější.

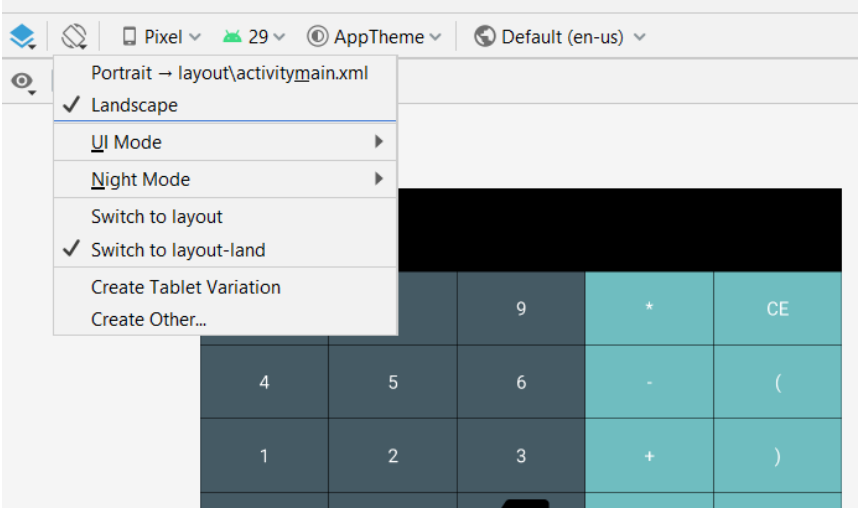
Samotná tlačítka jsou pak řešena trochu nezvyklým způsobem pomocí komponenty TextView. Tato komponenta byla zvolena kvůli přehlednosti kódu, funkčnost je v tomto případě stejná, jako by tomu bylo při využití Button. Každé komponentě je v kódu přidána událost na kliknutí, která se stará o výpočty a jejich zobrazování. Tlačítko s ikonou smazání je pak vytvořeno pomocí ImageView. Kdy je pomocí android:src přiřazena ikona a pomocí dalších atributů pak probíhá její nastavení. To je podrobněji popsáno v kapitole komponenty.

Horní část okna zobrazuje výpočty a výsledky. Tato část je reprezentována pomocí dvou pod sebou umístěných TextView. Přičemž první část s výpočty se aktualizuje pokaždé, když se klikne na nějaké z tlačítek, zatímco druhá část s výsledkem se aktualizuje pouze při kliknutí na tlačítko rovná se. Práce s oběma těmito TextView je tedy velice snadná, používá se pouze tečková notace, klíčové slovo text a přiřazení hodnoty. Tím se TextView vždy aktualizuje a zobrazují se tak vždy aktuální výsledky a výpočty.

Samotné výpočty jsou pak realizovány pomocí ExpressionBuilder, který se pomocí metod build a evaluate stará o přesné výpočty. ExpressionBuilderu je pomocí parametru předán

textový řetězec, obsahující výpočet a pomocí dvou již zmíněných metod je pak z řetězce vytvořen přesný výsledek, jsou brány ohledy i na pořadí matematických operací a používání závorek.

Jedním z dalších problémů u této aplikace bylo řešení otáčení obrazovky. Tedy aby se vzhled kalkulátoru přizpůsobil aktuálnímu otočení obrazovky. To je v aplikaci řešeno pomocí dvou různých XML souborů s layoutem. V obou je použit již zmíněný lineární layout, liší se pouze počtem řádků a počtem tlačítek na řádku. V nastavení projektu je pak layout přiřazen k orientaci. Nejjednodušší způsob, jak toto nastavit, je přímo v designéru, kde se nachází přímo ikona pro orientaci. Lze to však nastavit i dalšími způsoby, například přímo v nastavení projektu.



Obrázek 30 Aplikace kalkulátor - nastavení orientace

Aplikace je tedy ve výsledku plně funkční a zároveň i uživatelsky přívětivá. Bylo zvoleno jednoduché rozložení, aby byla celá aplikace co možná nejpřehlednější a zároveň i snadno použitelná pro všechny uživatele. Jak již bylo zmíněno, byl zvolen vhodný layout, jednoduchý design a uživatelsky přívětivé rozložení komponent.

9 APLIKACE BLUETOOTH PIŠKVORKY

Druhou aplikací je hra piškvorky. Hlavním cílem této aplikace je přiblížení práce se senzory, konkrétně byla zvolena technologie Bluetooth. Aplikace se tedy zaměřuje na komunikaci dvou zařízení za využití Bluetooth. Hra piškvorky je tedy realizována pomocí této technologie. Dalším cílem byla jednoduchá ovladatelnost aplikace a její uživatelská přívětivost. Jedním ze způsobů, jak toho dosáhnout je rozdělení aplikace do několika částí.

Jednotlivé části jsou realizovány pomocí nových oken. První okno zobrazuje dvě tlačítka a seznam dostupných zařízení, která se vyskytují poblíž a mají aktuálně zapnuté Bluetooth. S těmito zařízeními je možné provést pomocí aplikace spárování. V druhém okně je pak opět několik tlačítek a také je zde zobrazen seznam spárovaných zařízení. Pomocí těchto tlačítek a seznamu je možné navázat komunikaci s jiným zařízením. Při navázání komunikace se pak aplikace přepne na třetí okno, ve kterém je zobrazena samotná hra, kterou mohou uživatele pomocí Bluetooth hrát.

Po zapnutí aplikace se nejprve ověří, zda je na mobilním zařízení zapnuto Bluetooth, pokud tomu tak není, je uživatel aplikací vyzván, aby Bluetooth povolil. Jak je již popsáno výše v práci využívá se k tomu metoda startActivityForResult. Poté, co uživatel povolí Bluetooth, může pokračovat v používání aplikace. V prvním okně lze za využití metody startDiscovery a BroadcastReceiveru zobrazit všechna dostupná zařízení. Ta se v okně vypisují pomocí ListView. Někdy může zařízení chvíli trvat, než nalezne ve svém okolí další dostupná zařízení, seznam se tedy plní postupně. V seznamu je pak možné na jednotlivá zařízení klikat. Po kliknutí na některé z nich se provede spárování, které probíhá pomocí metody createBond. V závislosti na verzi Androidu je možné, že bude nutné povolit v zařízení kromě Bluetooth i polohu. Poloha by měla být vyžadována až od verze Android 10. V úvodním okně se kromě seznamu zařízení vyskytují dvě tlačítka, první z nich slouží právě pro zobrazení seznamu a druhé z nich slouží k přepnutí do dalšího z oken.

V druhém okně se opět nacházejí tlačítka a seznam. V tomto případě se však jedná o seznam, ve kterém se zobrazují pouze spárovaná zařízení. O zobrazení spárovaných zařízení se stará jedno z tlačítek a následně metoda bondedDevices. Další z tlačítek umožňuje návrat do předchozího okna. Nejsložitější částí aplikace je však zahájení a následné udržování komunikace. Při zahajování komunikace je nutné rozhodnout, které ze zařízení je v pozici serveru a které je klientem. To je zajištěno tím, že jeden z uživatelů klikne na tlačítko „poslouchat“ a druhý klikne v seznamu spárovaných zařízení na to zařízení, se kterým chce komunikovat. Ten, který klikne v seznamu na spárované zařízení, se stává klientem a ten, který

klikne na tlačítko „poslouchat“, se stává serverem. To, které ze zařízení je klient a které je server, však nemá na průběh hry žádný vliv, pořadí hráčů je určeno náhodně. Zařízení se pak navzájem vyhledávají pomocí BluetoothSocket a jeho metod. Ve chvíli kdy dojde k propojení, se aplikace přepne do okna s hrou. Pokud k propojení nedojde, vypíše se na obrazovku chybová hláška, která uživatele upozorňuje, že nedošlo k propojení.

Po navázání spojení si zařízení vyšlou náhodně vygenerovaná čísla, zařízení s vyšším číslem je určeno jako začínající hráč. Hráčům se na obrazovce zobrazuje, který z nich je právě na tahu. Hrací pole je vždy přístupné pouze hráči, který právě hraje. Pokud hráč není na tahu, hrací pole sice vidí, ale není mu povoleno na cokoli klikat. Samotná komunikace je v aplikaci realizována za využití třídy SendRecieve, která obsahuje InputStream a OutputStream. Čtení a zápis dat do streamů je realizován pomocí metod read a write, které obě pracují s polem bytů. Posílané informace je tedy nutné vždy převést právě na pole bytů. Poté co ten z hráčů, který je na tahu, klikne na některé z volných políček, je z jeho zařízení odeslána zpráva, která má v sobě zakódované číslo reprezentující zakliknuté políčko. Druhé zařízení pomocí třídy SendRecieve zaznamená, že mu dorazila zpráva, zašle ji handleru a rozkóduje její obsah. Handler je v této aplikaci využit pro všechny zprávy mezi zařízeními, i pro zprávy o tom, zda proběhlo či neproběhlo připojení. Na základě toho jaký typ zprávy handleru dorazí, vykoná nějakou z částí kódu, především je však využíván právě k rozkódování obsahu posílaných zpráv. Poté, co rozkóduje zprávu, se hráči zobrazí, které tlačítko jeho protivník vybral a je mu umožněna hra. Tento proces se opakuje do chvíle, než některý z hráčů vyhraje nebo než dojde k zaplnění hracího pole.

```
class SendRecieve(myBluetooth:BluetoothSocket, private val handler: Handler) : Thread() {
    var inputStream: InputStream ?= myBluetooth.inputStream
    var outputStream: OutputStream ?= myBluetooth.outputStream
    override fun run(){
        var buffer = ByteArray( size: 1024)
        var bytes: Int? = null
        while (true){
            try {
                if(interrupted()){return}
                bytes = inputStream?.read(buffer)
                if(bytes != null){
                    handler.obtainMessage(STATE_MESSAGE_RECIEVE, bytes,  arg2: -1, buffer).sendToTarget()
                }
            }catch (e:IOException){
```

Obrázek 31 Aplikace Bluetooth - přijetí zprávy

Po dohrání hry se oběma hráčům zobrazí pomocí AlertDialog vyskakovací okno s výsledkem hry a s otázkou, zda chtějí pokračovat ve hře. V případě že oba hráči vyberou, že chtějí pokračovat, je spuštěna další hra. Pokud hráči pokračují ve hře, je na tahu ten z nich,

který v minulém kole prohrál. Pokud však jeden z hráčů pokračovat nechce, nebo do uplynutí určité doby nevybere odpověď, je hra ukončena a hráči jsou přesunuti na úvodní obrazovku.

I tato aplikace je ve výsledku plně funkční, akorát mohou nastat problémy s verzí Androidu. Díky rozdělení do několika různých oken je aplikace přehledná a snadno ovladatelná. Komunikace mezi dvěma zařízeními probíhá bez problémů a po odeslání zprávy z jednoho zařízení je zpráva téměř okamžitě přijata a rozkódována druhým zařízením. Což znamená, že hned poté co hráč, který je právě na tahu, zaklikne vybrané políčko, se druhému hráči zobrazí, které políčko jeho protivník vybral a je mu umožněna hra. Hra je tedy plynulá a uživatelsky přívětivá.

ZÁVĚR

Cílem této bakalářské práce bylo seznámení s vývojem mobilních aplikací. Konkrétně pak vývoj aplikací pro operační systém Android v programovacím jazyce Kotlin. Dalším cílem práce bylo vytvoření dvou ukázkových aplikací. První z aplikací se zaměřuje především na práci s rozložením a designem, zatímco druhá aplikace se zaměřuje na složitější operace a to konkrétně na práci s technologií Bluetooth.

V bakalářské práci je v několika kapitolách popsána základní práce s mobilními aplikacemi. Byly představeny základní typy layoutů a jejich možné atributy. Dále byla přiblížena práce s několika základními komponentami a v neposlední řadě pak bylo ukázáno programování složitějších operací. Do budoucna by pak tato část práce mohla být rozšířena o popis dalších komponent a o ukázky práce s dalšími složitějšími technologiemi.

Jako první ukázková aplikace byl zvolen jednoduchý kalkulátor, ten měl sloužit jako ukázka práce s rozložením a designem. Aplikace měla umět ovládat základní matematické operace jako všechny běžné kalkulátory. Aplikace tyto své vytyčené cíle splňuje. Pro tuto aplikaci bylo důležité zvolit jednoduché rozložení, aby byla co možná nejpřehlednější a aby byla především snadno ovladatelná pro všechny uživatele. Po zvážení vhodného layoutu, byl vybrán jednoduchý design a uživatelsky přívětivé rozložení komponent. Aplikace je tedy plně funkční a díky svému výslednému vzhledu i snadno použitelná.

Hlavním cílem druhé aplikace byla práce s technologií Bluetooth. Cílem bylo vytvořit jednoduchou hru piškvorky, při které spolu budou soupeřit dva hráči. Hráči proti sobě měli hrát právě pomocí Bluetooth. U této aplikace bylo nejdůležitější propojit dvě mobilní zařízení a zajistit mezi nimi správné posílání zpráv. Při vývoji aplikace nastalo několik problémů, z nichž se většina povedla vyřešit. Prvním problémem bylo spárování dvou zařízení a řešení tohoto problému nebylo ve výsledku nijak náročné. Složitější pak byl problém se správnou synchronizací hrací plochy a určením, který z hráčů bude začínat hru. Díky správnému zasílání zpráv se však i tento problém podařilo vyřešit. Zůstal však problém s různými verzemi operačního systému Android. Různé verze tohoto operačního systému mají odlišné požadavky pro Bluetooth a v některých verzích si tak musí uživatel vše správně nastavit v nastavení jeho mobilního zařízení. Aspoň z části je však i tento problém řešen tím, že aplikace uživatele na začátku přiměje zapnout Bluetooth.

Jak již bylo zmíněno, práce by mohla být do budoucna rozšířena o popis dalších komponent a případně i o popis práce s dalšími složitějšími technologiemi. Aplikace by rovněž mohly být do budoucna mírně upraveny. U první z aplikací by mohly být přidány některé další

matematické operace a tím zároveň i upraveno její rozložení. U aplikace s piškvorkami by pak mohla být k dispozici i složitější hra a případně by se mohlo ještě více zapracovat na problému s různými verzemi operačního systému Android. V případě většího rozšíření této aplikace by mohla být teoreticky přidána i hra proti počítači.

Závěrem tedy lze říci, že práce samotná splnila všechny vytyčené cíle a stejně tak je tomu i u obou naprogramovaných aplikací. Do budoucna pak mohou být aplikace i práce ještě dále rozšířeny.

POUŽITÁ LITERATURA

[1] BUDIU, Raluca. Mobile Apps. *Nielsen Norman Group* [online]. 2013, 14. 9. 2013 [cit. 2020-07-15]. Dostupné z: https://www.nngroup.com/articles/mobile-native-apps/

[2] Vývoj mobilních aplikací - kompletní průvodce. *Pixelfield* [online]. 2019 [cit. 2021-07-17]. Dostupné z: https://pixelfield.cz/vyvoj-aplikaci/

[3] UJBÁNYAI, Miroslav. *Programujeme pro Android*. Grada Publishing, a.s, 2012. ISBN 978-80-247-3995-3.

[4] LACKO, Ľuboslav. *Mistrovství - Android: Kompletní průvodce vývojáře*. Brno: Computer Press, 2017. ISBN 978-80-251-4875-4.

[5] Operační systém na telefonech: Jaký operační systém je nejlepší volbou pro váš telefon. *NINIT* [online]. [cit. 2020-09-11]. Dostupné z: https://www.newsinit.cz/operacni-system-na-telefonech/

[6] LACKO, Ľuboslav. *Vývoj aplikací pro iOS*. Brno: Computer press, 2018. ISBN 978-80-251-4942-3.

[7] Programming software and the IDE. *BBC* [online]. [cit. 2020-07-29]. Dostupné z: https://www.bbc.co.uk/bitesize/guides/zgmpr82/revision/5

[8] ROUBALOVÁ, Eliška. *Java bez předchozích znalostí*. Brno: Computer Press, 2015. ISBN 978-80-251-4572-2.

[9] SPÄTH, Peter. *Pro Android with Kotlin: Developing Modern Mobile Apps*. Berlin: Springer, 2018. ISBN 978-1484238196.

[10] KŘÍŽEK, Otakar. Proč použít Kotlin místo Javy. *SBLOG* [online]. 30. 10. 2018 [cit. 2021-02-12]. Dostupné z: https://blog.seznam.cz/2018/10/proc-pouzit-kotlin-misto-javy/

[11] Creating an Xcode Project for an App. *Apple Developer* [online]. [cit. 2021-01-27]. Dostupné z: https://developer.apple.com/documentation/xcode/creating_an_xcode_project_for_an_app

[12] Bluetooth overview. *Android Developers* [online]. 2021, 10. 1. 2021 [cit. 2021-04-08]. Dostupné z: https://developer.android.com/guide/topics/connectivity/bluetooth

[13] Camera API. *Android Developers* [online]. 2021, 24. 2. 2021 [cit. 2021-04-08]. Dostupné z: https://developer.android.com/guide/topics/media/camera

[14] Sensors Overview. *Android Developers* [online]. 2019, 27. 12. 2019 [cit. 2021-04-09]. Dostupné z: https://developer.android.com/guide/topics/sensors/sensors_overview

PŘÍLOHY

PŘÍLOHA A – KÓD V JAZYCE JAVA

```
public class Osoba {

    private String jmeno;
    private String prijmeni;
    private int vek;

    public Osoba(String jmeno, String prijmeni, int vek) {
        this.jmeno = jmeno;
        this.prijmeni = prijmeni;
        this.vek = vek;
    }

    public String getJmeno() {
        return jmeno;
    }

    public void setJmeno(String jmeno) {
        this.jmeno = jmeno;
    }

    public String getPrijmeni() {
        return prijmeni;
    }

    public void setPrijmeni(String prijmeni) {
        this.prijmeni = prijmeni;
    }

    public int getVek() {
        return vek;
    }

    public void setVek(int vek) {
        this.vek = vek;
    }

    @Override
    public String toString() {
        return "Osoba se jmenuje " + jmeno + " " + prijmeni + " a je stará " + vek + " let";
    }
}


public class Main {
    public static void main(String[] args) {
        Osoba osobaJava = new Osoba("Monika", "Kopřivová", 21);
        osobaJava.setJmeno("Anna");
        System.out.println(osobaJava);
    }
}
```

PŘÍLOHA B – KÓD V JAZYCE KOTLIN

```
class Osoba (private var jmeno: String, private var prijmeni: String, private var vek: Int) {

    fun setJmeno(jmeno: String){
        this.jmeno = jmeno
    }

    fun setPrijmeni(prijmeni: String){
        this.prijmeni = prijmeni
    }

    fun setVek(vek: Int){
        this.vek = vek
    }

    fun getJmeno(): String {
        return this.jmeno
    }

    fun setPrijmeni(): String {
        return this.prijmeni
    }

    fun setVek(): Int {
        return this.vek
    }

    override fun toString() : String{
        return "Osoba se jmenuje " + jmeno + " " + prijmeni + " a je stará " + vek + " let"
    }
}

fun main(args: Array<String>) {
    var osobaKotlin: Osoba = Osoba("Monika", "Kopřivová", 21)
    osobaKotlin.setJmeno("Anna")
    println(osobaKotlin)
}
```

PŘÍLOHA C – ZDROJOVÉ KÓDY A DOKUMENTY

K práci jsou přiloženy veškeré zdrojové kódy obou naprogramovaných aplikací.