Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Vizuální podpora teorie grafů

Diplomová práce

Autor: Bc. Vladislav Braha
Studijní obor: Aplikovaná informatika

Vedoucí práce: RNDr. Andrea Ševčíková

Hradec Králové duben 2020

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 30.4.2020 Vladislav Braha

Poděkování:

Děkuji vedoucí diplomové práce RNDr. Andree Ševčíkové za metodické vedení práce a svým rodičům za podporu během svého studia.

Anotace

Proces gamifikace v posledních letech poukazuje na rozšířené možnosti zvýšení motivace studentů různých oborů k procvičování dané látky. Tato diplomová práce se zaměřuje na vytvoření nástroje s využitím gamifikace pro podporu výuky předmětu Teorie grafů. V její teoretické části seznamuje čtenáře s možnostmi použití gamifikace ve výukových nástrojích a dále představuje některé konkurenční aplikace. Práce se dále zabývá výběrem platformy pro realizaci takové aplikace, která umožňuje vizualizovat pojmy z Teorie grafů a zároveň nabízí procvičování právě probrané látky a přiblížení některých nezbytných pojmů z tvorby mobilní aplikace. V praktické části provede práce čtenáře od návrhu aplikace po její implementaci a popis algoritmů použitých k její realizaci. Výsledkem vlastní práce autora je aplikace dostupná pro mobilní zařízení Android s využitím vestavěné knihovny této platformy. Závěr je věnován shrnutím výsledků, problémům při vývoji a doporučením pro další rozvoj aplikace.

Annotation

Title: Visual Support for Graph Theory

The process of gamification in recent years points to widespread opportunities to increase the motivation of students of various disciplines to practice the subject. This thesis is focused on the application development using gamification which supports Graph Theory learning. The first part informs the reader about the usage of Gamification method as an educational tool and presents some of the competition´s gadgets created for this purpose. The other part of the thesis deals with the right platform selection that is capable to visualize Graph Theory terms and enables practical examples related to the topic. Necessary terms related to mobile application creation are also mentioned in this part. The practical part deals with a mobile application creation from the drafts to its algorithms used. The outcome of the author's work is the mobile application that works under the Android operation system using mainly its own libraries. The conclusion of this thesis is focused on the results summary, problems faced during the application creation and finally the recommendation for further development.

Obsah

Seznam obrázků

Seznam grafů

Seznam tabulek

Seznam ukázek kódu

Seznam zkratek

DB……Database
OS……Operation system
FOSS……Free and Open Source Software
APK……Android Application Package
AppStore……Application Store
BaaS……Backend as a Service
MBaaS……Mobile Backend as a Service
JSON…… JavaScript Object Notation
XML…… eXtensible Markup Language
PNG…… Portable Network Graphics
SVG……Scalable Vector Graphics
HTML…… Hypertext Markup Language
DIMA……Diskrétní matematika

1 Úvod

Jedním z předmětů vyučovaných na Fakultě informatiky a managementu Univerzity Hradec Králové (UHK) je předmět Diskrétní matematika (dále DIMA). V rámci předmětu se studenti zabývají teorií grafů, jejímž cílem je rozvoj logického a algoritmického myšlení a dále se věnují aplikací kombinatorických algoritmů v problémech z reálného života. Studenti často mívají problémy s pochopením matematických definic. Z tohoto důvodu jsou základní pojmy teorie grafů, jejich vlastnosti a vztahy mezi sebou znázorňované a vysvětlované pomocí vizuálních pomůcek jako jsou obrázky nebo vizualizační software, mnohokrát vytvořený k tomuto účelu.

Vzhledem k rozvoji výpočetních zařízení, které nosí studenti s sebou na přednášky či dokonce na cvičení matematických předmětů, se stále častěji a běžněji začínají využívat různé e-learningové nástroje, které jim umožňují snáze pochopit probíranou látku a případně si ji procvičovat.

Mobilní telefon v dnešní době k takovým zařízením rozhodně patří. Ten má student u sebe neustále a situace, kdy ho z kapsy vytáhne, aby si zkrátil čas během dne (např. při jízdě hromadnou dopravou nebo v čekárně u doktora), nastává několikrát. Nabízí se tedy možnost přinést výukové materiály blíže ke studentům právě pomocí této technologie. Probírané látce a procvičování se můžou věnovat i ve chvílích, kdy nemají po ruce skripta, notebook, nebo jenom dostupné internetové připojení.

Aby bylo používání aplikace pro studenty atraktivnější, je třeba ho rozšířit o některé herní prvky. Proces gamifikace v posledních letech poukazuje na rozšířené možnosti zvýšení motivace studentů různých oborů k procvičování dané látky, viz studie [1]. Její účinky jsou vidět i v herních titulech prodávaných celosvětově v milionových nákladech [2]. O tom, že se dá využít herních prvků nejen ve výukovém prostředí, ale i v běžném životě, nás přesvědčuje vývoj některých států, který používá stejné prvky využívané ve hrách pro hodnocení svých občanů [3].

Tato práce se zaobírá možností multioborově využívaných prvků gamifikace při podpoře výuky předmětu DIMA vyučovaného na UHK. Aplikací

používaných ve výuce teorie grafů je v dnešní době několik, na UHK jich také několik existuje. Tyto aplikace jsou především určeny pro PC a webové prohlížeče. Prozatím nebyla vyvinuta žádná mobilní aplikace, která by byla využívaná pro předmět DIMA.

Tato diplomová práce, skládající se z teoretické a praktické části, popisuje tvorbu právě takové aplikace, která by tuto pomyslnou mezeru zaplnila. Teoretická část se zabývá analýzou konkurenčních nástrojů a možnostmi vytvořeného programu s vysvětlením nezbytných pojmů. V praktické části následuje popis tvorby samotné aplikace od jejího návrhu po implementaci řešení včetně problémů, které během tvorby nastaly. Závěrem jsou popsány výsledky testování a používání včetně závěru a návrhů na rozšíření.

2 Cíl práce

Cílem této práce je vytvoření mobilní aplikace, která by sloužila pro podporu výuky Diskrétní matematiky za využití prvků gamifikace. Aplikace by umožňovala studentům lépe pochopit základní pojmy z teorie grafů, a navíc by nabídla procvičování právě probíraných pojmů. Diplomová práce zároveň ověřuje možnost, zda je v potenciálu mobilního zařízení vytvořit edukativní nástroj s podporou prvků gamifikace.

3 Teoretická východiska

V této části je vymezen pojem gamifikace a popsané některé studie jejího využití. Základní pojmy z teorie grafů použité v práci při popisu aplikace a v samotné aplikaci jsou primárními koncepty teorie grafů a pro čtenáře jsou nepochybně známé, možné je nalézt v [4], [5].

3.1 Vymezení pojmu gamifikace

Proces učení je úzce spojený s motivací. Lidé, kteří jsou motivováni do dané aktivity, jsou schopni obětovat dané aktivitě často i svůj volný čas a zároveň je práce více baví. Motivace je jednou z klíčových prvků při plnění nějaké činnosti. Rob van Roy popisuje ve své studii [6] několik druhů motivace. Jednou z nich je vnitřní motivace plynoucí z vlastních pocitů a zapálení do práce na dané činnosti. Dalším typem je motivace vnější, kdy jsou lidé přinuceni nějakou činnost dělat a často se snaží s tímto typem ztotožnit, aby ji dokončili. Rob za lepší motivaci považuje motivaci vnitřní, kdy se mohou lidé s danou činností ztotožnit. K umocnění této motivace autor studie přidává faktory, které by měl člověk k úkolu mít:

- autonomii (schopnost být pánem své činnosti),
- kompetenci (schopnost a dovednosti pro dokončení úkolu),
- ztotožnění se se skupinou, která na daném úkolu pracuje (cítit, že je člověk částí dané skupiny pracující na zadaném úkolu).

Ve své studii Rob také na vzorku studentů dokazuje růst vnitřní motivace na daných úkolech za pomoci *gamifikace* [6]. Využití gamifikace je stále předmětem nových objevů a aplikačních prostředí. S její praktickou implementací se během života setkal snad každý. O co se tedy jedná?

Gamifikace je podle Zamzamiho učení založené na herních prvcích, populární především v mobilních a jiných technologiích, sloužící k podpoře určitého typu chování za účelem zlepšení studijních výsledků. Tento druh učení spoléhá na získávání vědomostí skrze posbírané zkušenosti. [7]

Dicheva v [8] definuje gamifikaci jako „integraci herních prvků do neherních prostředí“. Jedná se o zatraktivnění činnosti pro její uživatele [8].

3.2 Využití gamifikace

Využití gamifikace popisuje Landers ve své práci [9], použil ji při tvorbě online vědomostních kvízů, kde se snažil namotivovat studenty k vyplnění kvízů tím, že sbírali odznáčky a po splnění úkolů/otázek se dostávali na další úrovně.

Gamifikace může přesahovat i hranice vzdělávání. Čínská vláda např. představila systém „Sezame Credit" vytvořený skupinou Alibaba (známá též pro internetový obchod), který je založený na hodnocení svých občanů na základě jejich chování. Ten občany země hodnotí na základě jejich finanční historie a dalších morálních hodnot, jako např. nakupování čínského zboží nebo příspěvků publikovaných na sociálních sítí. Na základě tohoto hodnocení uživatelům poskytuje buď výhody (dostupné půjčky na bydlení, odložené splátky za zboží, přednostní ošetření v nemocnicích atd.), nebo je naopak trestá (zpomalení internetu, omezený přístup k dobře placeným pozicím a školám, omezení ve výběru hotelů s vyšším standardem atd.). [3]

Mitchell ve své práci [10] poukazuje na vhodnost modelu použitého pro gamifikaci. Zatímco ve hrách je sbírání bodů a odznáčků součást vnitřní motivace pokračovat ve hře, v pracovním prostředí může být tento systém brán zaměstnanci jako způsob kontroly [10].

Sanchez v [1] popsal online kvízy, které vytvořil pro své studenty. Vyplňováním kvízů studenti sbírali skóre, které ovlivnilo jejich výslednou známku z daného předmětu. Díky tomu se jeho studenti zlepšili v probírané látce a dosahovali mnohem lepších studijních výsledků.

Proces gamifikace se v případě vzdělávání dělí podle Sancheze na čtyři stádia. Zaprvé je nutné vytvořit obsah, kde lektor popíše problematiku a vysvětlí ji studentům. Poté přidá do hry elementy, které z daného obsahu vytvoří nějakou interakci se studenty, probíhá zde tedy proces zvaný gamifikace. Např. vytvoří kvíz, kde si budou studenti moci procházet jednotlivé otázky a odpovídat na ně. Ve třetím kroku jsou přidány odměny za splnění aktivity (např. splnění daného kvízu). Ty mohou mít formu posbíraných bodů či již zmíněných odznáčků. Posledním stádiem jsou výstupní znalosti. Studenti by měli mít po splnění aktivit nové znalosti

reprezentované posbíraným skóre v průběhu plnění aktivit (např. procházení kvízu). Zároveň Sanchez tvrdí, že pokud nemá nějaká činnost studijní efekt, pak ji nemá cenu v rámci výukové gamifikace implementovat. [1]

V případě mobilních aplikací Nakashima popisuje, že by měly pro větší motivaci uživatelů obsahovat systém hodnocení, který umožní uživatelům sbírat skóre za jejich aktivitu v aplikacích, což zlepší jejich motivaci k dalšímu jejímu používání a zároveň jistou soutěživost v reálném čase vůči svým spolužákům nebo kolegům, kteří budou aplikaci používat také. Soutěživost podpoří nadšení do provádění zadaných úkolů. Nakashima zároveň ale varuje před dvěma úskalími, která mohou při tvorbě gamifikace pro aplikaci nastat. První z nich je malé množství účastníků, které snižuje soutěživost uživatelů. Druhým úskalím je soutěžení uživatelů s různou úrovní dovedností k dané aktivitě. Nerespektování těchto bodů může vést ke snížené motivaci k pokračování v aktivitách. [11]

3.2.1 DIMA na UHK

Ve výuce předmětu DIMA na Fakultě informatiky a managementu byly v minulých letech vyvinuty interaktivní nástroje pro podporu výuky GrAlg [12] , A-DIMA [13] a webové nástroje GraPro [14] a ProfVis [15], které jsou používány jako doplněk při vysvětlování pojmů z teorie grafů a grafových algoritmů a důkazů matematických tvrzení z oblastí teorie grafů. Mobilní aplikace pro podporu výuky předmětu DIMA zatím vytvořena nebyla. Zároveň žádná ze zmiňovaných aplikací v sobě nepoužívá prvky gamifikace. Pro studenty UHK bezpochyby není neznámá, setkávají se s ní v běžném životě, např. v sociálních sítích při sbírání palců, či srdíček za příspěvky, nebo v některých e-shopech, které nabízejí sbírání bodů za nákup a motivují uživatelé k opakovanému využití těchto služeb.

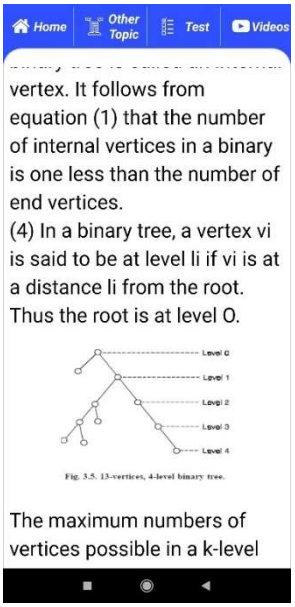
3.3 Konkurenční nástroje

Nástrojů zabývající se teorií grafů je několik, značná část z nich je napsaná v technologii Javascript a zaměřuje se hlavně na webové použití [16]. Optimalizace těchto nástrojů je pro ovládání na mobilních zařízeních omezená, v některých případech prakticky žádná. Existují také aplikace napsané přímo pro mobilní zařízení. Některé nabízejí studijní materiály bez ukázek, nebo pouze s obrázky (viz obrázek 1, 2), jiné nabízejí tvorbu grafů s možností spuštění některých základních algoritmů, případně nabízejí další za poplatek nebo po zakoupení plných verzí aplikace.

Obr. 1 Snímek z aplikace Teorie grafů

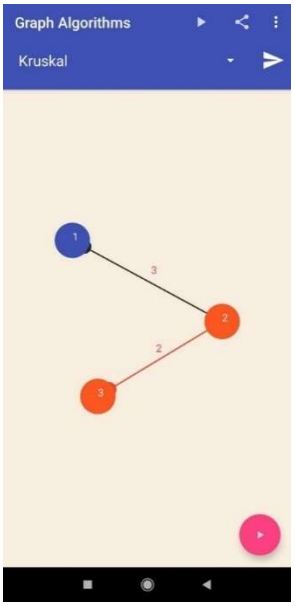
zdroj: snímek obrazovky dané aplikace (Google Play)

Aplikace Teorie grafů (obrázek 1) a Graph Theory (obrázek 2) obsahují popis algoritmů a některých pojmů z diskrétní matematiky ve formě textu a obrázků. Aplikace Graph Theory umožňovala v minulosti zřejmě testování dosažených znalostí i výuková videa. V době psaní této práce není však ani jedna z funkcionalit dostupná. Všechny grafy obsažené v aplikacích jsou pouze ve formě vložených obrázků, se kterými není možné dále pracovat.

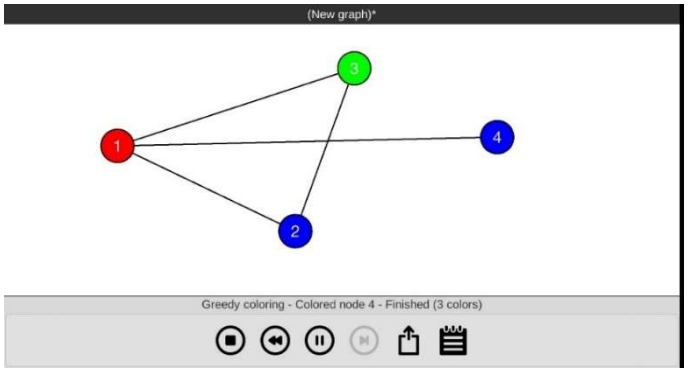


Obr. 2 Snímek z aplikace Graph Theory
zdroj: snímek obrazovky dané aplikace (Google Play)

Aplikace Graph Algorithms (obrázek 3) a Graphynx (obrázek 4) umožňují uživatelům vytvořit grafy a následně na nich vyzkoušet některé z dostupných algoritmů. Mají vždy jednu aktivitu, ve které je možné vybrat z rozbalovací nabídky příslušný algoritmus a ten na daném grafu spustit. V případě aplikace Graphynx je možné ještě graf exportovat jako obrázek ve formátu PNG nebo SVG, nicméně tyto aplikace nevyužívají výukový potenciál, který by propojil matematické definice s příslušnou ukázkou na zobrazených grafech.



Obr. 3 Snímek z aplikace Graph Algorithms
zdroj: snímek obrazovky dané aplikace (Google Play)



Obr. 4 Snímek z aplikace Graphynx
zdroj: snímek obrazovky dané aplikace (Google Play)

3.4 Výběr platformy

Do roku 2017 se na trhu mobilních OS vyskytovalo několik systémů, které měli podíl větší jak 1 %, všechny však v průběhu let ztrácely svůj tržní podíl vůči 2 hlavním rivalům od Googlu a Applu [17]. Po konci operačního systému Windows Phone [18] se na trhu vyskytují v podstatě již dva operační systémy pro mobilní zařízení, a to Android od společnosti Google a iOS od společnosti Apple. Ke konci roku 2019 činilo rozšíření operačního systému Android zhruba 87 % oproti 13% podílu operačního systému iOS, přičemž společnost International Data Corporation předpokládá, že se v následujících letech bude podíl operačního systému Android ještě zvyšovat, viz tabulka 1 [19]. Pro tvorbu aplikace byla vybrána platforma operačního systému Android pro co nejvyšší rozšiřitelnost mezi uživatele.

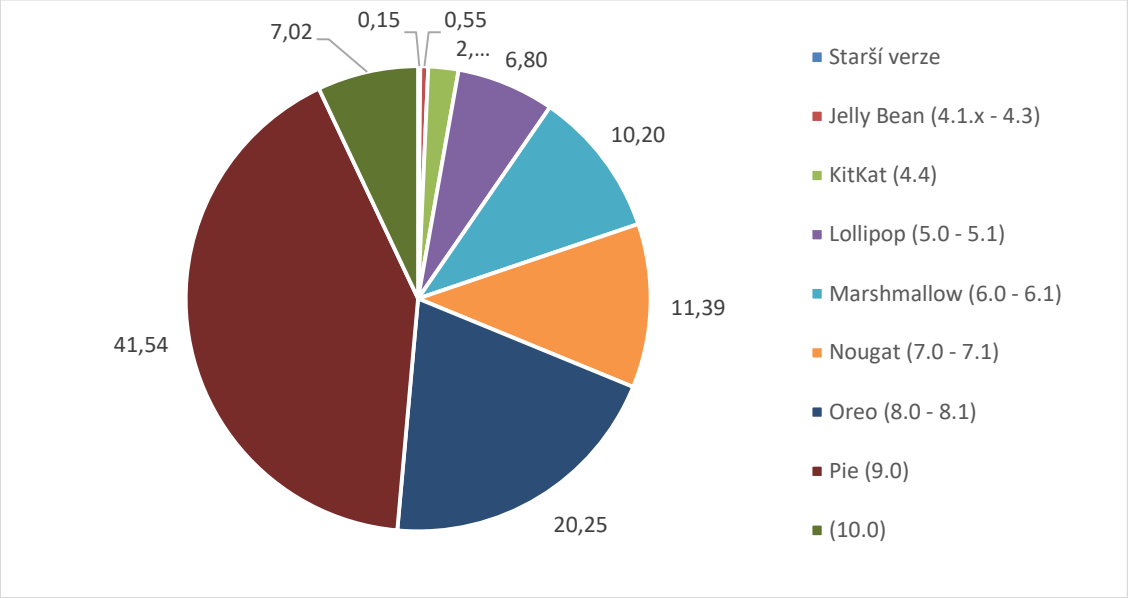
Tabulka 1 Porovnání podílu operačních systému na trhu

Year	2017	2018	2019	2020	2021	2022
Android	85.1 %	85.1 %	87.0 %	87.0%	87.2 %	87.3 %
iOS	14.7 %	14.9 %	13.0 %	13.0 %	12.8 %	12.7 %
Others	0.2 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %
TOTAL	**100.0 %**	**100.0 %**	**100.0 %**	**100.0 %**	**100.0 %**	**100.0 %**

zdroj: International Data Corporation [19]

4 Operační systém Android

Android je open-source operační systém určený pro mobilní zařízení vyvíjený primárně společností Google. Byl vydán v roce 2007 a od té doby vyšlo již několik verzí [20]. Podle Statcounteru měl ke konci roku 2019 největší zastoupení Android 9 Pie se svými téměř 40 % [21] (viz graf 1), oficiální podíl na trhu (dle zařízení přihlášených do obchodu Google Play) byl v době psaní práce znám pouze z května 2019 [22] a vzhledem k poměrně rapidním změnám byla použita tato nejnovější dostupná data i přes mírnou nepřesnost, viz graf 1. V nich je vidět, že nejnovější verze získávají rychle podíl na trhu oproti verzím starším, a to i přesto, že výrobci zařízení často podporují svá zařízení novými verzemi operačního systému jenom několik měsíců po uvedení zařízení na trh [23].



Graf 1 Podíl verzí operačního systému Android na trhu na začátku roku 2020
zdroj: Statcounter [21]

4.1 Distribuce aplikací

Aplikace běžící na Androidu jsou instalovány skrze instalační balíčky apk (Android application package). Ty může uživatel stáhnout do telefonu ručně, nebo skrze různé online tržiště. Výhodou stahování aplikací přes tzv. Marketplace (tržiště) je, že zpravidla umožňují aplikace aktualizovat v telefonu, jakmile vývojář vydá nějakou novou verzi aplikace. [24]

Zařízení většinou od výrobce obsahují předinstalované Google aplikace včetně tržiště Google Play. To umožňuje uživatelům stahovat aplikace (některé jsou zdarma, jiné placené). Vývojáři, kteří chtějí začít nabízet aplikace na Google Play jsou povinní před nahráním instalačního apk souboru zaplatit jednorázový poplatek. [25]

Existují však i alternativní tržiště aplikací, která nabízejí publikování apk souborů zdarma. Jako příklad je možné uvést tržiště F-Droid, které však umožňuje publikaci pouze FOSS aplikací. FOSS je zkratka pro Free and Open Source Software, tedy software, který je k dispozici ve formě open-source (dostupného zdrojového kódu) [26]. To není problém u projektů nevyužívající knihovny třetích stran, jakmile ale aplikace využívá knihovny od proprietárních zdrojů (např. služby Google, více v kapitole 6.1), není možné takovou aplikaci v tomto tržišti publikovat.

Mezi další zdroj online tržiště aplikací patří Amazon App Store. Jedná se o jednu ze služeb společnosti Amazon, kterou můžou znát lidé jako tržiště zboží různého charakteru. Amazon nabízí také softwarová řešení, jako např. online uložiště, virtuální servery, tržiště aplikací atd. K publikování aplikace už není vyžadováno splnění licenčních podmínek FOSS, stačí splnit jisté parametry aplikace týkající se např. podpory velikosti zařízení, ovládání aplikace, nároků na zařízení atd. Z důvodů licenčních podmínek a ceny za publikaci aplikace byla pro šíření instalačního apk vytvářené aplikace použito právě toto tržiště aplikací (více o tom v kapitole 6.1) [27].

4.2 Životní cyklus aplikace

Aplikace prochází několika fázemi, na které vývojář může a nemusí reagovat. Operační systém k nim přistupuje jako k haldě, tedy poslední otevřená aktivita je přidána na vrch haldy, zatímco ostatní čekají, než se opět stanou aplikací běžící na popředí. Toho může být dosaženo např. přepnutím mezi naposledy otevřenými aplikacemi, což je možnost dostupná skrze tlačítko na zařízeních Android [28]. Během životního cyklu může také dojít k vynucenému vypnutí aplikace, ať už z důvodu, že je aplikace ukončena uživatelem (např. aplikace zamrzne a neodpovídá), nebo během docházejících systémových prostředků.

V případě, že se systém rozhodne ukončit běžící proces, rozhodne se podle několika kritérii, kterými označí důležitost běžících aplikací. Následující seznam kritérii je seřazen podle důležitosti pro operační systém:

1. Služba běžící na popředí – jedná se o služby, které jsou vyžadovány právě běžící aplikací.
2. Viditelný proces – proces, který obsluhuje činnost uživatele.
3. Servisní proces – jedná se o službu, není přímo viditelná, ale obsluhuje nějakou část, které si je uživatel vědom (např. služba pro nahrávání a stahování dat).
4. Cache proces – tento proces není v době běhu výše zmíněných procesů potřeba, v případě, že systém potřebuje uvolnit některé prostředky, jsou ukončeny právě tyto procesy.

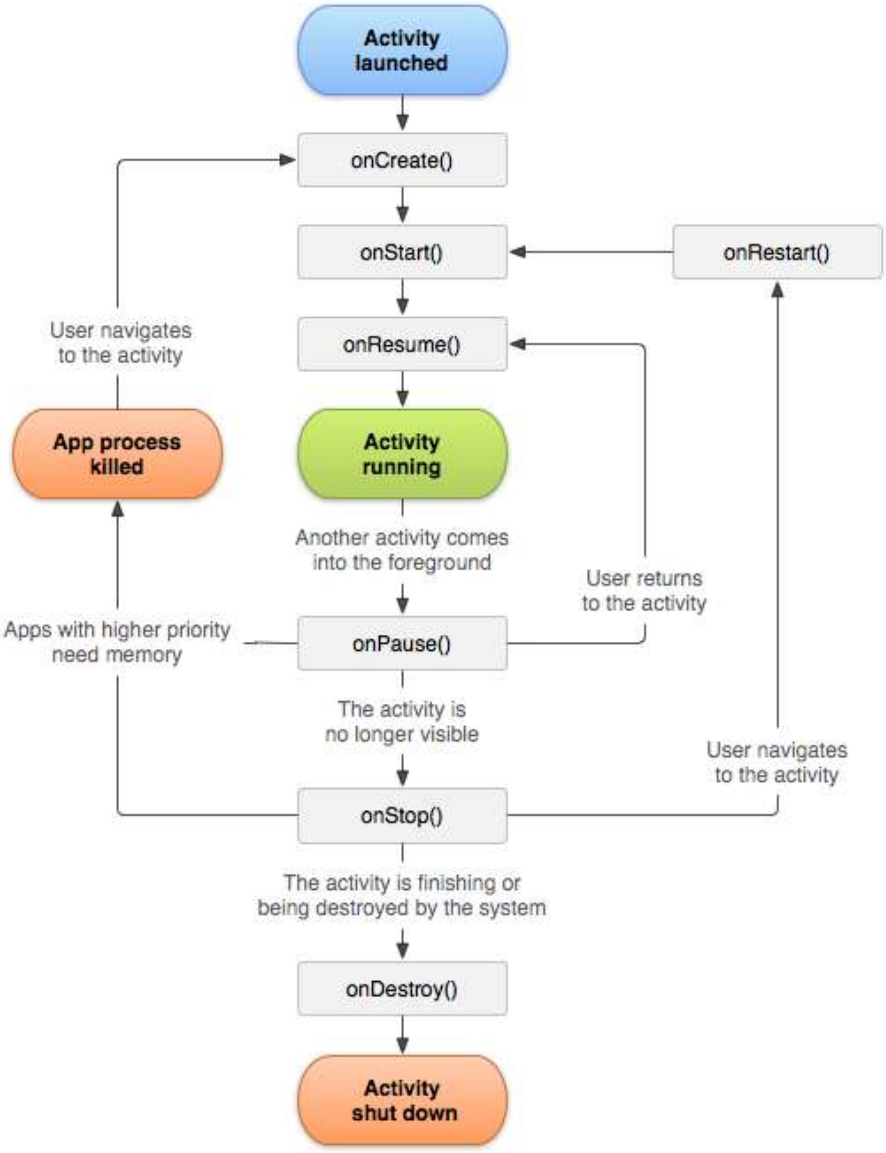
Podle Googlu je tedy důležité rozlišovat, zdali se jedná o aktivitu, službu, nebo něco jiného, aby systém mohl správně fungovat [29]. Na obrázku 5 je znázorněn životní cyklus aplikace, který začíná spuštěním aplikace a končí buď jejím vypnutím, nebo (násilným) ukončením systémem.

Jak je vidět, aplikace prochází několika stavy:

- onCreate – jedná se o stav, kdy se aplikace vytváří, během ní by měl vývojář vytvořit potřebné objekty, např. vytvořit View atd. Zároveň zde může přistupovat k uloženým datům, která byla uložena během předchozího stavu aplikace (pokud nějaký byl).
- onStart – tento stav následuje vždy po předchozím onCreate. V tomto stavu se aplikace stává viditelnou uživateli, přechází potom do onResume, nebo onStop (v případě, že aplikace přestane být viditelná).
- onResume – zde začíná aplikace reagovat na uživatelovy podměty a zároveň se nachází na vrchu haldy spuštěných aplikací.
- onPause – stav nastávající vždy po stavu onResume. Jedná se o stav, kdy aplikace není ukončena, dochází k němu, když přechází do pozadí (ať už otevřením jiné aplikace, nebo např. stisknutím domovského tlačítka na zařízení). Z tohoto stavu

může přejít obnovením zpátky do onResume. Vývojářům je doporučeno v tomto stavu nedělat náročné operace.

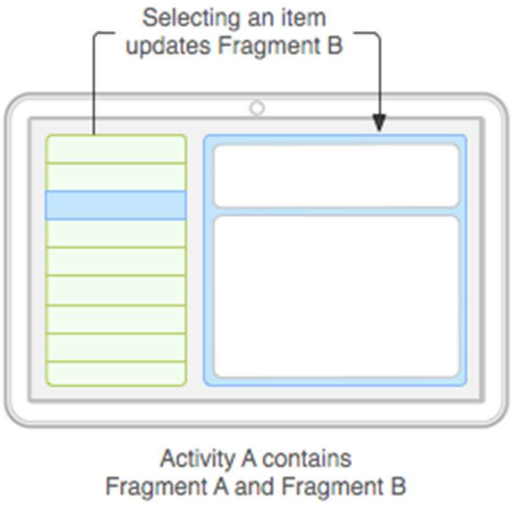
- onStop – do tohoto stavu aplikace dojde, jakmile přestane být viditelná uživateli (avšak není ještě ukončena), přechází do onRestart, nebo onDestroy (podle další aktivity uživatele).
- onDestroy – poslední stav, ve kterém se aktivita nachází před ukončením (ať už voláním ukončovací funkce vývojářem, nebo systémem, který uvolňuje prostředky další aplikace). [30]



Obr. 5 Životní cyklus aktivity v OS Android
zdroj: Medium [28]

4.3 Životní cyklus Fragmentu

Fragment je stavitelná část aktivity s vlastním životním cyklem, na který mohou vývojáři reagovat. Fragment musí být vždycky umístěn v aktivitě, nemůže existovat sám o sobě. Jakmile je např. aktivita pozastavena, musí se pozastavit i fragment, podobně u ukončení aktivity musí být ukončena činnost fragmentu. Fragment slouží jako znovupoužitelná komponenta systému Android, která aktivitám umožňuje dynamicky měnit svůj obsah (nebo např. celé fragmenty). Tato funkce je zvlášť užitečná pro situace, kdy má aplikace nějaké navigační menu a na základě jeho výběru se zobrazuje jiný obsah, viz obrázek 6. [31]

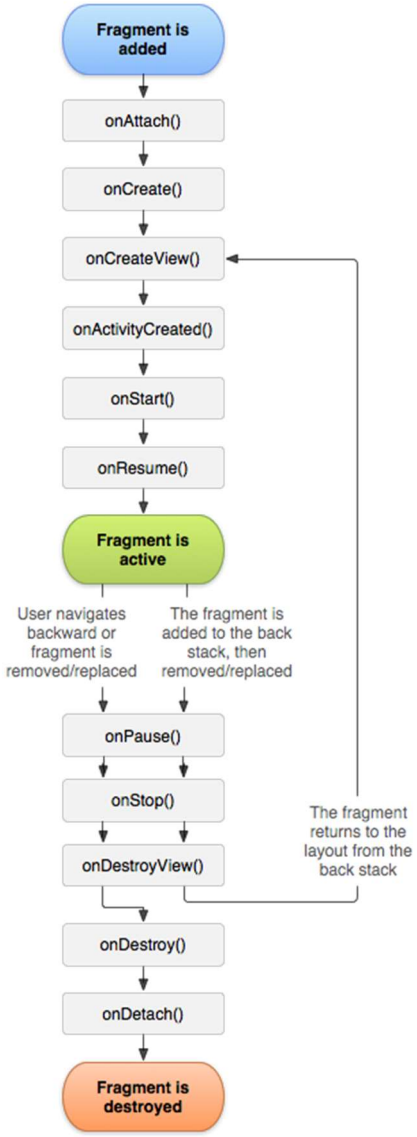


Obr. 6 Příklad využití Fragmentu na navigačním menu
zdroj: Android Developers [31]

Fragment má životní cyklus podobný životnímu cyklu aktivity, viz obrázek 7. Z tohoto důvodu budou zmíněny stavy, ve kterých se výrazně liší:

- onCreate – do tohoto stavu fragment přichází, když začne být vytvářen. Vývojáři by zde měli nainicializovat prvky, které by měli ve fragmentu zůstat poté, co fragment přejde do stavu onPause.
- onCreateView – do tohoto stavu přijde fragment, jakmile se zobrazí uživatelské prvky. Metoda obstarávající tento stav vrací View, bez kterého by se prvky nezobrazily.

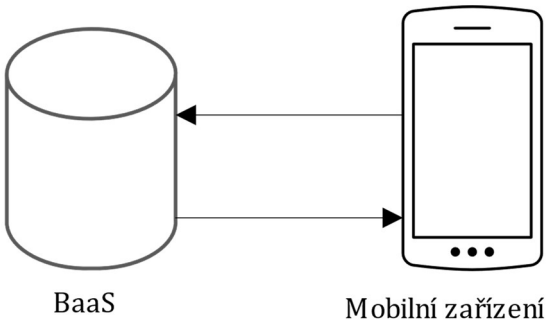
- onPause – tento stav indikuje, že uživatel opouští současný fragment (např. při výměně zobrazení současného fragmentu za jiný). Vývojáři by zde měli uložit současný stav fragmentu, pokud jeho stav chtějí obnovit. [31]



Obr. 7 Životní cyklus Fragmentu
Zdroj: Android developers [31]

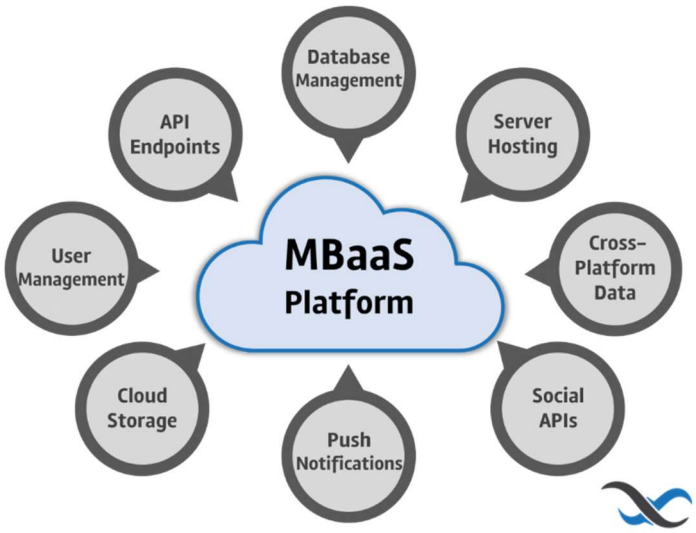
4.4 Backend as a Service

Backend jako služba (též používaná anglická zkratka BaaS) je označení modelu, který poskytuje vývojářům webových a mobilních aplikací různé backendové služby a funkcionality, které v mnoha případech potřebují vývojáři napříč různými aplikacemi. Bez těchto služeb by vývojář musel nastavit a udržovat servery, připravit infrastrukturu na adaptibilní nárůst dat/výkonu, vytvořit API, nastavit a udržovat databázi, starat se o zabezpečení, nasazování nových verzí, starat se o push notifikace (upozornění) atd. Podle scénáře a velikosti aplikace může využití těchto služeb ušetřit až týdny práce. Výhodou tohoto modelu je to, že se vývojáři mohou soustředit na vývoj frontendové části (části, kterou vidí uživatelé) a nemusí se starat o technickou část některých služeb běžících na pozadí (např. virtuální servery, kontejnery pro běh aplikací atp.). Vztah je možné vidět na obrázku 8. Díky tomu zvládnou vývojáři vyvinout aplikace v mnohem rychlejším čase. [32]



Obr. 8 Komunikace BaaS
zdroj: vlastní zpracování

Jistou podtřídou BaaS jsou služby nazývané jako mobilní backend, jako např. MBaas (viz obrázek 9). Ty jsou přímo designované pro použití na mobilních operačních systémech, stačí si vybrat poskytovatele těchto služeb a integrovat jeho SDK (Software Development Kit) do svojí aplikace. [33]



Obr. 9 Služby MBaaS
zdroj: Backendless [34]

Tyto služby nabízí několik poskytovatelů. V tabulce 2 je zobrazen přehled u vybraných MBaaS.

Tabulka 2 Přehled poskytovaných vybraných služeb u vybraných MBaaS

Služby	Firebase	Kinvey	Amazon	Kumulos
Cloudová databáze	x	x	x	
Služby pro ověření uživatelů	x		x	
Cloud messaging	x			x
Uložiště	x	x	x	x
Testovací nástroje	x		x	x
Systém pro hlášení chyb	x			x

zdroj: DevTeam.Space [35]

Firebase od společnosti Googlu nabízí téměř všechny dostupné služby, zároveň umožňuje používání těchto služeb zdarma (na rozdíl od některých jiných společností) při nízké frekvenci používání, který je u této aplikace předpokládán.

Jako příklad toho, co Firebase nabízí zdarma, je uvedeno v tabulce 3 (několik služeb a jejich limit pro bezplatné použití denně).

Tabulka 3 Služby nabízené zdarma u MBaaS Firebase

Zápis do databáze	20tis
Čtení z databáze	20tis
Smazání záznamů z databáze	20tis
Uložiště dat	5 GB
Počet simultánních zápisů do DB	100
Hlášení chyb	Neomezeně

zdroj: Firebase [36]

Vzhledem k dostupnosti nabízených služeb a předpokládaným potřebám aplikace je jako MBaaS služba vybrána právě Firebase od společnosti Google.

5 Aplikace

5.1 Návrh aplikace

Před vytvořením samotné aplikace byly nakresleny návrhy, jak by mohlo vypadat uživatelské prostředí. Důraz byl kladen na přehlednost a využívání systémových knihoven operačního systému Android. Pro vytvoření návrhů byl využit online nástroj k tvorbě wireframů pro různé informační projekty Balsamiq [37].

Obr. 10 Přihlašovací obrazovka
zdroj: vlastní zpracování skrze Balsamiq [37]

Přihlašovací obrazovka umožňuje přihlášení i registraci z jednoho okna pro ušetření zbytečných přesunů mezi výběrem těchto dvou možností, viz obrázek 10. Kromě e-mailu je ještě zadávána přezdívka uživatele, která se v rámci celé aplikace používá.

V samotné aplikaci je vyjížděcí menu (v Androidu realizováno skrze komponentu Navigation Drawer [38]), které obsahuje přehled všech dostupných aktivit

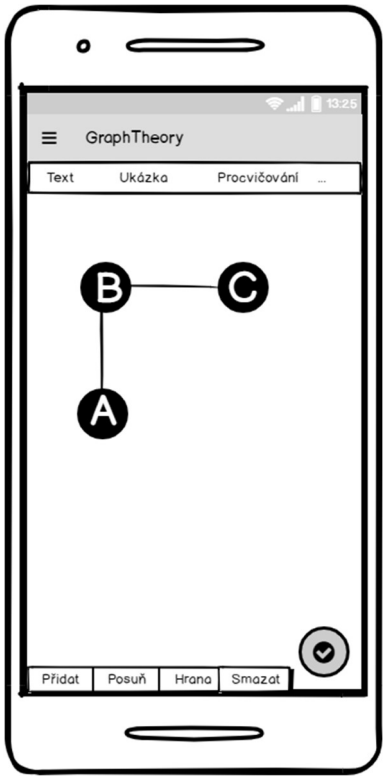
v aplikaci, mezi kterými se je možné přepínat, viz obrázek 11. Aktivity nejsou po přihlášení všechny ihned dostupné. Uživatelům se postupně odemykají v závislosti na tom, jak plní předcházející úkoly v záložce procvičování.

Obr. 11 Přehled dostupných aktivit
zdroj: vlastní zpracování skrze Balsamiq [37]

Hlavní aktivity se skládají ze tří částí oddělených od sebe záložkami (Google označuje ve svém Material Designu tyto záložky Tab Layoutem), viz obrázek 12. V první části nazvané Text, se nachází učební text, který koresponduje obsahem s prezentacemi promítanými během přednášek. V další záložce s názvem Ukázka jsou příklady aktuálně vysvětlované látky. Poslední kartou je Procvičování, které umožní vyzkoušet si probranou látku na předchozích obrazovkách pomocí konkrétních příkladů.

V dolní části aplikace (viz obrázek 12) se nacházejí tlačítka, která umožňují uživateli ovládat objekty, se kterými právě na aktuální obrazovce pracuje. Vzhledem k omezenému prostoru na displejích mobilních zařízení bylo nutné zkrátit text jednotlivých možností s ohledem na pochopitelnost dané volby. Nad tlačítky v dolní

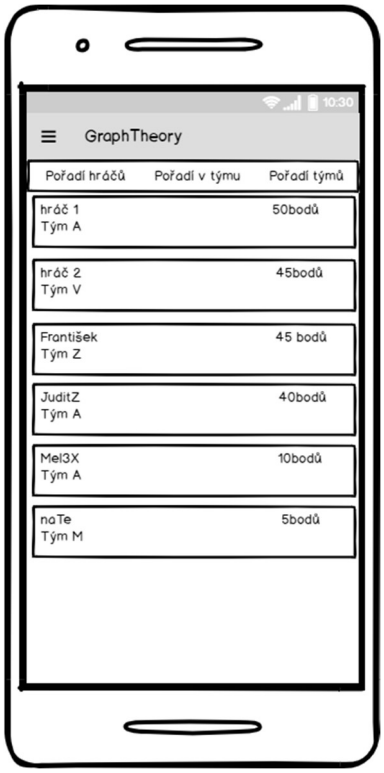
části obrazovky se vyskytuje, v pravé části, další tlačítko, které spustí ověření výsledku vůči vygenerovanému zadání aplikace. V případě, že uživatel splní danou aktivitu, ve které se ještě nenacházel, odemkne se mu další téma ve vyjíždějícím menu.



Obr. 12 Aktivita s procvičováním
zdroj: vlastní zpracování skrze Balsamiq [37]

Posledním oknem návrhu je statistika uživatelů v aplikaci, viz obrázek 13. Ta umožňuje srovnávat výsledky na základě jejich nasbíraného skóre. Skóre se zvyšuje splněním úlohy vygenerované aplikací, přičemž jeho hodnota se postupně s procvičováním stejného cvičení snižuje, aby byli uživatelé motivováni v pokračování procvičování dalších cvičení. Celkový výsledek je srovnáván s dalšími uživateli. Ve statistice je možné porovnávat skóre jak se všemi uživateli, tak i pouze s těmi nacházejícími se ve stejném týmu. K dispozici je i porovnání všech bodů získaných jednotlivými týmy. To by mělo podpořit vnitřní motivaci uživatelů v procvičování (více v kapitole 3.1). Výběr týmu je na samotném uživateli. Volba probíhá během registrace, kdy je po úspěšném vyplnění registračních údajů zobrazeno pole pro zadání názvu týmu. Název

je dále, pro účely aplikace, převeden na unifikovaný název (jednotný formát pro velikost písmen a mezer). Díky tomu je pak možné shlukovat uživatele, kteří zadají různý název týmu např. Cyklisté a CYKLISTÉ.



Obr. 13 Statistika uživatelů v aplikaci
zdroj: vlastní zpracování skrze Balsamiq [37]

5.2 Tvorba aplikace

Vzhledem k využívání současných komponent Androidu je aplikace mířena na verze Androidu 7 a vyšší; vzhledem k tomu by měla být aplikace použitelná na více jak 80 % procentech zařízení s tímto operačním systémem, viz graf 1 v kapitole 4.

Při tvorbě aplikace je potřeba použít některé ikony, které usnadní uživatelům orientaci. Android k tomu umožňuje používat SVG ikony. Aby je vývojář nemusel vytvářet všechny, existují tzv. knihovny SVG obrázků, které umožňují jejich použití zdarma, nebo za nějaký drobný poplatek. Pro ikony, které nebyly dostupné v rámci knihovny Android Studia, byly použity SVG obrázky z SVG Repo, které nabízí, dle svých statistik, více jak 300 tis. ikon k bezplatnému užití [39].

Vzhledem k návrhu aplikace bylo potřeba na začátku vyřešit, jakým způsobem budou některé funkcionality, jako např. synchronizace dat, přihlašování atd. řešeny. Vzhledem k nutnosti udržovat backendovou strukturu pro běh potřebných funkcionalit a předpokládanému počtu uživatelů v řádech nižších desítek aktivních uživatelů byla zvolena struktura Firebase MBaaS (viz Backend as a Service).

Pro integraci služeb Firebase je nutné vytvořit si účet u Googlu, který je svázaný s vytvářenou aplikací. Po jeho vytvoření dostane vývojář přístup do prostředí zvané konzole, kde je možné zaregistrovat vícero aplikací. Po registraci aplikace může vývojář zapnout služby, které ji budou poskytovány, jako je např. autentizace, databáze atd. [40]

Jakmile má vývojář nakonfigurovaný Firebase, může ho začít integrovat do aplikace. Pro vývoj aplikací z platformy Android je možné využít přímo vývojové prostředí IntelliJ od společnosti JetBrains, které má pro Androidí projekty odnož v podobě Android Studia. Je k dispozici zdarma (na rozdíl od jiných JetBrains produktů) a jako výchozí sestavovací nástroj má Gradle, jenž umožňuje přidávat závislosti na dalších objektech, které jsou nutné k běhu programu [41]. Pro integraci Firebase je tedy třeba přidat jeho implementaci do závislostí Gradlu, viz ukázka kódu 1, a následně přidat do aplikace konfigurační JSON, který nabídne přímo konzole.

```
dependencies {
…
        implementation 'com.google.firebase:firebase-database:19.2.0'
        implementation 'com.google.firebase:firebase-auth:19.2.0'
…
}
```

Ukázka kódu 1 Přidání závislostí na Firebase
zdroj: vlastní zpracování

5.2.1 Přihlašování a registrace

Ve vstupní aktivitě jsou dostupné dva scénáře – přihlášení a registrace. V obou případech vypadá proces podobně, viz obrázek 14. Uživatel vyplní vstupní pole, ale u registrace, na rozdíl od přihlášení, je ještě vyzván k zápisu do týmu dle vlastního výběru.

Obr. 14 Přihlašovací obrazovka
zdroj: vlastní zpracování

Název týmu je unifikován (pro malá a velká písmena v názvu) pro potřeby agregace uživatelů do jednotlivých týmů. V případě registrace je třeba, po vyplnění vstupních polí, zavolat příslušnou metodu, která se postará o registraci na straně Firebase (skrze API), ta umožňuje na sebe navěsit listener, který se provolá, jakmile je metoda hotová, viz ukázka kódu 2.

```
FirebaseAuth mAuth = FirebaseAuth.getInstance();
mAuth.createUserWithEmailAndPassword(email, password)
    .addOnCompleteListener(this, task -> {
        if (task.isSuccessful()) {
            // Sign in success, update UI with the signed-in user's information
            createUserAndStartActivity(selectedTeam);
        } else {
            // If sign in fails, display a message to the user.
            Toast.makeText(LoginActivity.this, "Authentication failed.", Toast.LENGTH_SHORT).show();
        }
    });
```

Ukázka kódu 2 Možná registrace uživatele
zdroj: vlastní zpracování

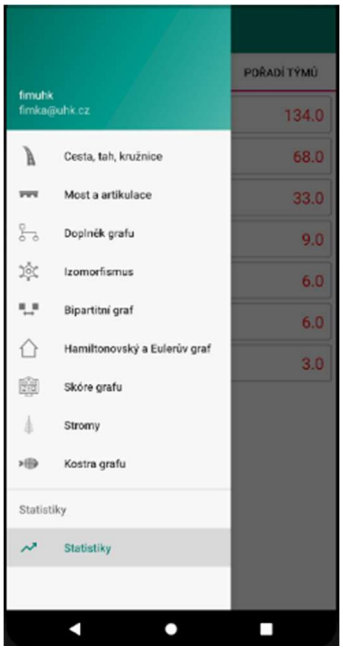
Je potřeba zmínit, že metody využívané pro komunikaci s Firebase jsou asynchronní, tedy jejich listenery jsou provolány, až když je daná událost dokončena a nelze spoléhat na jejich dokončení (např. registrace) v dalších krocích.

Obdobně funguje přihlašování jenom s tím rozdílem, že se volá metoda signInWithEmailAndPassword (uživatel se může přihlašovat i jinými údaji, podporované je přihlášení skrze sociální sítě nebo Gmail). Na objektu FirebaseAuth se může volat i metoda na odhlášení uživatele nebo kontrolu, zdali je uživatel přihlášen. Je to vhodné při situacích, kdy je třeba rozhodovat, jestli se má uživateli ukazovat přihlašovací obrazovka.

Po přihlášení do konzole je hned vidět nový záznam o přihlášeném uživateli; jeho účet je možné dočasně zablokovat, případně kompletně smazat. Během registrace je také vytvořen objekt User, který je dále používán pro zpracování statistik (viz kapitola 5.7).

5.2.2 Navigační menu

Jakmile uživatel projde registrací/přihlašováním, objeví se mu první aktivita. Každá reprezentuje jednu z probíraných oblastí z návrhu a je na ně možné se dostat z menu. Android má v sobě přímo implementované vyjíždějící menu, tzv. hamburger menu (☰). To koresponduje s tzv. material designem (grafickým návrhem ovládacího prostředí pro interakci s uživateli, který má Android ve svých komponentách použit [42]). Na obrázku 15 je vidět rozbalené menu aplikace, ze kterého se může uživatel navigovat na jednotlivé aktivity



Obr. 15 Navigační menu aplikace
Zdroj: vlastní zpracování

Pro jeho použití je potřeba nadefinovat menu v XML layoutu, viz ukázka kódu 3.

```
<androidx.drawerlayout.widget.DrawerLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:id="@+id/drawer_layout"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:fitsSystemWindows="true"
  tools:openDrawer="start">

<!-- Content -->

<com.google.android.material.navigation.NavigationView
  android:id="@+id/nav_view"
  android:layout_width="wrap_content"
  android:layout_height="match_parent"
  android:layout_gravity="start"
  android:fitsSystemWindows="true"
  app:headerLayout="@layout/navigation_drawer_header"
  app:theme="@style/navigationDrawer"
  app:menu="@menu/navigation_drawer_content" />


</androidx.drawerlayout.widget.DrawerLayout>
```

Ukázka kódu 3 Přidání Navigation Draweru do layoutu
zdroj: vlastní zpracování

Jakmile je menu přidané v layoutu, musí se přidat reakce na jeho akce v aplikační logice. Toho je docíleno implementací rozhraní OnNavigationItemSelectedListeneru z NavigationView, které umožňuje přepsat si výchozí metodu pro ošetření událostí. V ukázce kódu 4 je demonstrováno, jakým způsobem může být ověřeno, jestli se jedná o konkrétní položku v menu a dále následná reakce na výběr. Poté je menu zavřeno a metoda vrací booleanovskou proměnnou, která říká, zda má být položka v menu po kliku označena, jako poslední vybraná, při dalším otevření navigačního menu.

```
@Override
public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
  int id = menuItem.getItemId();

  if (id == R.id.nav_first) {
    Intent newActivityIntent = new Intent(this, NewActivity.class);
    finish();
    startActivity(newActivityIntent);
  }

  DrawerLayout drawer = findViewById(R.id.drawer_layout);
  drawer.closeDrawer(GravityCompat.START);
  return true;
}
```

Ukázka kódu 4 Reagování na událost výběru položky v menu
zdroj: vlastní zpracování

5.2.3 Přecházení mezi obrazovkami

Pro horní navigaci mezi okny slouží menu, které umožnuje překlikávat mezi třemi zobrazenými částmi. Android má pro tento use case připravenou komponentu zvanou Tab Layout, která se zobrazí hned pod toolbar (horní lištu). Tu je třeba opět přidat do layoutu, viz ukázka kódu 5.

```
<com.google.android.material.tabs.TabLayout
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:layout_gravity="fill"
    app:tabTextAppearance="@style/TabStyle"
    android:id="@+id/tabLayout">
 </com.google.android.material.tabs.TabLayout>
```

Ukázka kódu 5 Vložení TabLayoutu do aktivity
zdroj: vlastní zpracování

V aplikační logice je přidán listener na událost vyvolanou na liště uživatelem. Můžou nastat tři události, viz ukázka kódu 6, které je možné ošetřit. První označuje situaci, kdy uživatel vybere jednu z nabídek, druhá, když uživatel klikne na jinou možnost, než je současná zvolená, a třetí, když vybere opět stejnou možnost z nabídky.

```
TabLayout tabLayout =  view.findViewById(R.id.tabLayout);
tabLayout.addTab(tabLayout.newTab().setText("Text první záložky"));
tabLayout.addTab(tabLayout.newTab().setText("Text druhé záložky"));
tabLayout.addTab(tabLayout.newTab().setText("Text třetí záložky"));

tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {
    @Override
    public void onTabSelected(TabLayout.Tab tab) {
      //handle action
     }

     @Override
    public void onTabUnselected(TabLayout.Tab tab) {
     //handle action
    }

    @Override
     public void onTabReselected(TabLayout.Tab tab) {
     //handle action
    }
});
```

Ukázka kódu 6 Reagování na události v TabLayoutu
zdroj: vlastní zpracování

Aby bylo možné přecházet ve výukových aktivitách mezi obrazovkami, jak je znázorněno v návrzích aplikace, je potřeba „zabalit“ jejich obsah do Fragmentu. Ten umožňuje právě tuto funkcionalitu (viz sekce 4.3). Pro výměnu fragmentu je využito tzv. Fragment managera, který pro tyto situace OS poskytuje, viz ukázka kódu 7.

```
FragmentManager fragmentManager = getFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
fragmentTransaction.replace(R.id.generator_activity_group, fragmentName);
fragmentTransaction.remove(fragmentToRemove);
fragmentTransaction.add(R.id.generator_activity_group, fragmentName);
fragmentTransaction.commit();
```

Ukázka kódu 7 Vybrané možnosti práce s FragmentManagerem
zdroj: vlastní zpracování

První volání metody getFragmentManager vrátí FragmentManagera, který je pro aktuálně spuštěnou aktivitu. Dále je spuštěna řada operací související s přidáváním, nebo odebíráním fragmentu zakončena potvrzením o naplánování změny fragmentu. Tato změna však není provedena hned, ale je naplánována, jakmile hlavní vlákno dokončí svoji běžící operaci [43].

Vzhledem k tomu, že při změně fragmentu dochází k jeho zničení (životní cyklus fragmentu je popsán v kapitole 4.3), je třeba ošetřit ukládání jeho stavu, aby mohl

být při případném návratu na obrazovku znovu obnoven. Pro tyto situace vznikl ViewModel (součást Android Jetpacku – knihovny od Googlu snažící se zjednodušit vývoj aplikací), který umožňuje zachovávat data spojená s uživatelským rozhraním během různých stavů aktivit i fragmentů [44]. ViewModel je v praxi třída, ze které dědí třída, která se ukládá při změně stavu. V ní může být cokoliv, co je třeba zachovávat. V ukázce kódu 8 je naznačeno, jakým způsobem je možné získat ViewModel pro současnou aktivitu.

```
MapViewModel generatedMapViewModel =
ViewModelProviders.of(getActivity()).get(MapViewModel.class);
```

Ukázka kódu 8 Inicializace ViewModelu pro konkrétní aktivitu
zdroj: vlastní zpracování

Pro fragmenty, při vytváření View, je nainicializován příslušný ViewModel, a zkontrolován, že neobsahuje údaje z předchozího zobrazení fragmentu a pokud ano, tak je nastaven daný stav. V životním cyklu onDetach je uložen stav do ViewModelu pro případný návrat zpět.

5.2.4 Ovládání objektů

Pro ovládání vytvářených grafů, dle návrhu aplikace, byla využita navigace na spodní části obrazovky (tzv. Bottom Navigation View). Spodní menu umožňuje zobrazit text, ikony, případně jejich libovolnou kombinaci [45]. Pro využití v této aplikaci postačí zobrazení textu pro zanechání co největší plochy canvasu (viz dále). Přidání menu je třeba zadefinovat v xml zdrojích, viz ukázka kódu 9.

```
<com.google.android.material.bottomnavigation.BottomNavigationView
    android:id="@+id/navigation"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom"
    app:labelVisibilityMode="labeled"
    app:menu="@menu/settings_options" />
```

Ukázka kódu 9 Přidání Bottom Navigation View
zdroj: vlastní zpracování

Jednotlivé možnosti v menu jsou v samostatném xml souboru, kde je možné definovat jak text, tak i případnou ikonu. Jakmile bylo menu přidáno, bylo ošetřeno reagování na jednotlivé možnosti. Pro tyto účely

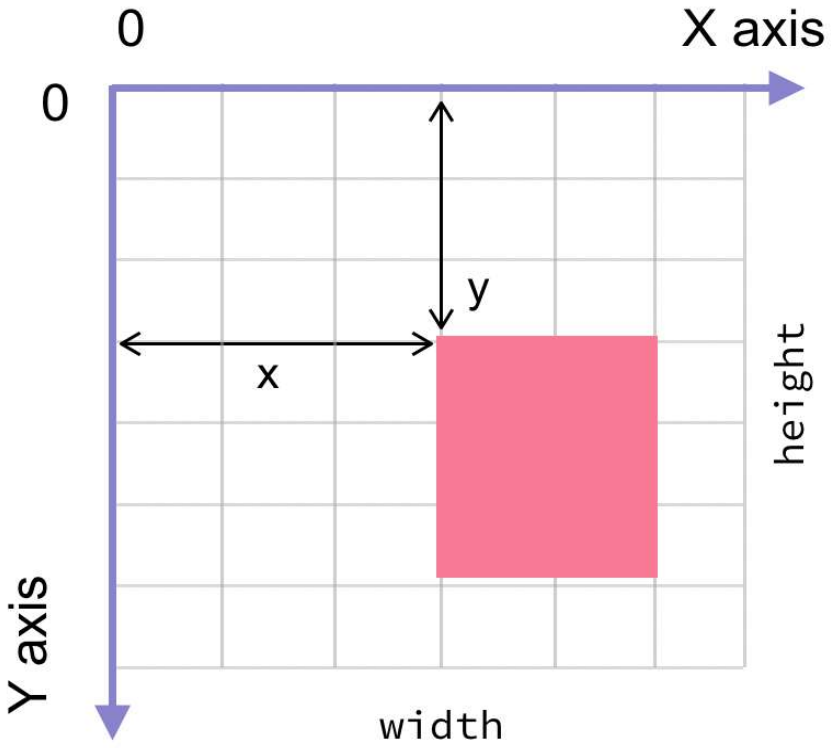
byl přidán do aplikační logiky listener, který ošetřuje jednotlivé změny ve výběru, viz ukázka kódu 10.

```
BottomNavigationView bottomNavigationView = findViewById(R.id.graph_generator_navigation);
  bottomNavigationView.setSelectedItemId(R.id.node);
  bottomNavigationView.setOnNavigationItemSelectedListener(menuItem -> {
    switch (menuItem.getItemId()) {
      // handle each selected item
    }
  });
```

Ukázka kódu 10 Listener událostí na Bottom Navigation View
zdroj: vlastní zpracování

5.2.5 Základní práce s objekty

Základem hlavních aktivit (těch, které obsahují ukázku nebo procvičování) je canvas obalený ve fragmentech. Canvas je třída v Androidu, která umožňuje vykreslit 2D objekty na obrazovku telefonu včetně bitmapy. K vykreslení takového objektu potřebuje souřadnice a objekt Paint. Souřadnicový systém canvasu začíná v levém horním rohu, kde jsou souřadnice [0,0] a do levého pravého rohu se dále souřadnice zvětšují, viz obrázek 16. [46]



Obr. 16 Souřadnicový systém canvasu
zdroj: Medium [46]

Třída Paint typicky obsahuje informace o objektu, který se má vykreslit. Obsahuje např. informace o barvě či šířce objektů (např. v případě vykreslení úseček) nebo informaci o tom, zda se má na vykreslovaný prvek použít antialiasing. K nastavování těchto hodnot má třída Paint připravené settery, které může programátor použít.

```
Paint mPaint = new Paint();
mPaint.setAntiAlias(true);
mPaint.setColor(DEFAULT_COLOR);
mPaint.setStyle(Paint.Style.STROKE);
mPaint.setStrokeJoin(Paint.Join.ROUND);
mPaint.setStrokeCap(Paint.Cap.ROUND);
```

Ukázka kódu 11 Nastavení třídy Paint
zdroj: Vlastní zpracování

Jakmile je nastavena třída Paint, je možné ji předat v rámci parametru metody objektu, který se bude kreslit. Ukázka kódu 12 přibližuje, jak probíhá volání metody pro vykreslení kružnice, která v aplikaci reprezentuje uzel.

```
mCanvas.drawCircle(coordinate.x, coordinate.y, BRUSH_SIZE, mPaint);
```

Ukázka kódu 12 Zavolání metody canvasu
zdroj: Vlastní zpracování

Pro úsečku se volá metoda drawLine, která si za parametr bere souřadnice počátečního a koncového bodu, mezi kterými se vykreslí (v případě aplikace souřadnice uzlu, který je s dalším uzlem propojen), viz ukázka kódu 13.

```
mCanvas.drawLine(x1, y1, x2, y2, mPaint);
```

Ukázka kódu 13 Volání metody pro vykreslení přímky v canvasu
zdroj: Vlastní zpracování

Obdobně, jako bylo ukázáno v předchozích příkladech, lze vykreslovat i další elementy, např. text, který je použit pro označení, o jaký uzel se jedná. Veškerá práce probíhá v metodě onDraw(), která v parametru předává právě zmíněný Canvas.

Ještě je potřeba zmínit, že třída, která se stará o vykreslení objektů, by měla dědit z třídy View, aby vykreslení probíhalo pomocí hardwaru telefonu (grafické karty), v opačném případě by probíhalo softwarově (skrze procesor) a mohlo by docházet k výrazným dopadům na rychlost vykreslování scény [46].

Pro předávání stavu na View byly vytvořeny některé objekty, které pomáhají popisovat stav, který se nachází na obrazovce. Pro View byl vytvořen objekt Graph (využívaný např. ViewModelem), který obsahuje seznam uzlů, hran a hran vyznačených v grafu, viz ukázka kódu 14.

```
private ArrayList<Edge> edges;
private ArrayList<Coordinate> nodes;
private ArrayList< Edge> redEdges = new ArrayList<>();
private ArrayList<Coordinate> redNodes = new ArrayList<>();
```

Ukázka kódu 14 Objekt Graph
zdroj: vlastní zpracování

Objekt Edge obsahuje začátek a konec úsečky, která spojuje dva uzly. Zároveň má metody kontrolující např. jestli se jedná o dvě shodné hrany, nebo zda nějaký bod leží na jednom z jejich konců. Po vytvoření základní kostry aplikace byly vytvořené samostatné výukové aktivity.

5.2.6 Struktura aplikace

Vzhledem k tomu, že se některé prvky na různých aktivitách opakují (vyjíždějící menu, přepínání mezi fragmenty, informační dialog atd.), bylo třeba zamezit tvoření tzv. boilerplatu (stále se opakujícímu kódu [47]). K těmto účelům byly vytvořeny abstraktní třídy jak pro fragmenty, tak pro běžné aktivity, které z nich dědí (viz příloha 1 schéma aplikace). První takovou třídou je AbstractAppCompactActivity, která obstarává události spojené s vyjíždějícím menu a rotací displaye. Dále byla vytvořena AbstractActivity, která se stará o přepínání mezi fragmenty ve výukových aktivitách, napojením na databázi a zobrazování zpráv uživateli (např. při splnění aktivity). Poslední vytvořenou abstraktní třídou je AbstractFragment, jenž obstarává ukládání a případné obnovování prvků ve View při opouštění / navracení se z fragmentu.

V jednotlivých balíčcích jsou třídy obstarávající logiku jednotlivých aktivit. V balíčku se nachází zpravidla aplikační logika jednotlivé aktivity (která se z důvodu různých úkolů pro uživatele mění obsahem) a fragment pro ukázkovou aktivitu, která je různá z důvodu generování ukázek pro každou aktivitu zvlášť. Třída pro obstarávání cvičení je pro aktivity zpravidla stejná, neboť

její fungování je pro všechny typy příkladů shodné, liší se však podmínky splnění zadaného úkolu, což si obstarává aplikační logika jednotlivých aktivit.

5.3 Výukové texty

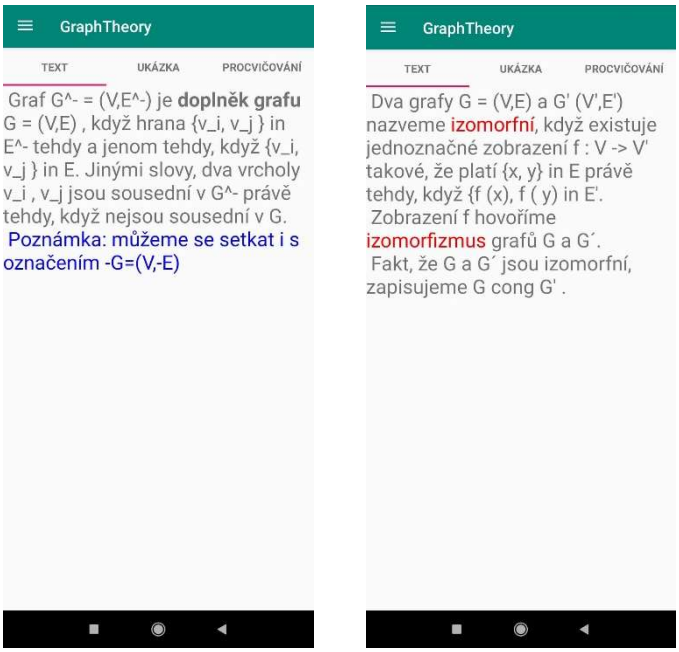
Každá výuková aktivita po otevření zobrazí výukový text. Ten koresponduje s texty promítanými na přednáškách předmětu DIMA a jejich obsah nebude dále rozebírán. Android nabízí pro zobrazování textu tzv. TextView, což je interface pro zobrazování textu uživatelům [48]. Pro úpravu zobrazení textu jsou HTML tagy podporovány OS Androidem. Jak název napovídá, jedná se o formátování pomocí značkovacího jazyka, který se používá pro tvorbu webových stránek. Podporuje základní tagy, jako např. tučný text, kurzíva nebo barvu a velikost textu. Jeho použití je vcelku jednoduché, stačí text obalit do tagů podle formátování, které se bude na text aplikovat, a předat ho příslušnému TextView, viz ukázka kódu 15.

```
textView.setText(Html.fromHtml("<font fgcolor='blue'><b>Kostra grafu</b></font> G=(V,E)
        je strom T=(V,E´)...."));
```

Ukázka kódu 15 Stylování textu pomocí HTML tagů pro TextView
zdroj: vlastní zpracování

V ukázce kódu 15 je předvedeno, jak může vypadat označení tučného modrého textu. Toto formátování však nepodporuje zobrazování matematických výrazů v matematické podobě a jeho použití vyprodukovávalo výsledky, které jsou vidět na obrázcích 17.

Výsledky byly nedostatečné, a proto byla vybrána externí knihovna, která by umožňovala lépe stylovat text, a hlavně ho dokázala reprezentovat v matematické podobě. KaTex je jednou z knihoven založených na známé knihovně Tex používané pro matematické zápisy [49]. Formátování textu probíhá pomocí klíčových symbolů, viz ukázka kódu 16, jehož výsledek je ukázán na obrázku 18.



Obr. 17 Ukázka textu formátovaného pomocí HTML tagů
zdroj: vlastní zpracování

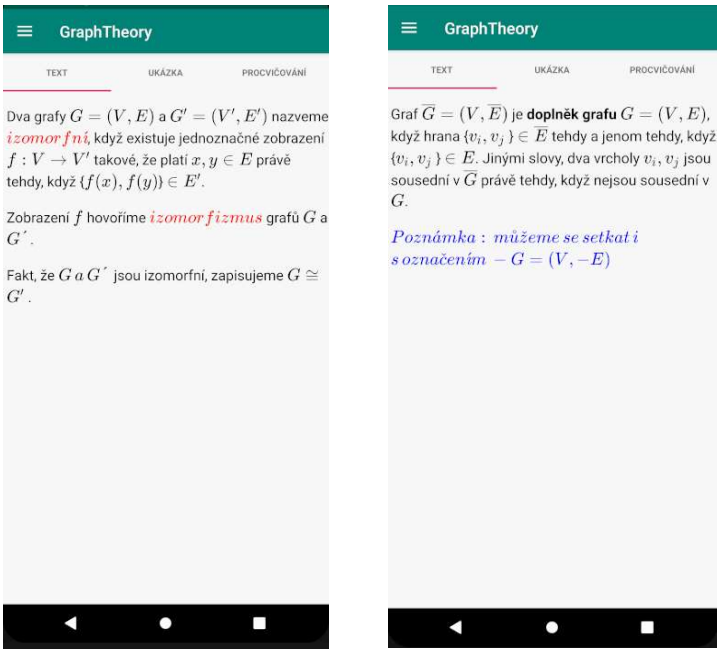
```
<string name="fourth_activity_text">
  <![CDATA[
    Dva grafy $G = (V,E)$ a $G\' = (V\',E\')$ nazveme $\\color{red}{izomorfní}$, když...
  ]]>
</string>
```

Ukázka kódu 16 Text s formátováním pro knihovnu KaTex
zdroj: vlastní zpracování

Každý text, který má být formátován pomocí této knihovny, je třeba ohraničit $ z každé strany, případně $$ pro označení textu, který se má zobrazit na řádku samostatně. Uvnitř dolarových znaků slouží pro konkrétní formátování dvě zpětná lomítka a za nimi následuje formátovací řetězec. Formátovacích řetězců umožňující měnit styl, kterým bude text vykreslen, je velká řada. Pokud má být formátován více než jeden znak, musí být takový řetěz umístěn mezi množinovými závorkami ({ }).

Při používání některých speciálních znaků ve strings.xml jsou v rámci procesování escapovány (odstraněny, nebo převedeny na speciální znak, či sekvenci znaků) a nahrazeny speciálními znaky. Ty však KaTex neumí správně reprezentovat. Proto je potřeba označit text pro knihovnu <![CDATA[, která označuje v XML část, která se má předávat bez escapování [50]. Toto opatření zajistí, že zobrazený text bude naformátován přesně tak, jak bylo zamýšleno. Jak je možné vidět na obrázku

18, texty takto naformátované jsou mnohem čitelnější a zároveň respektují matematický zápis výrazů.



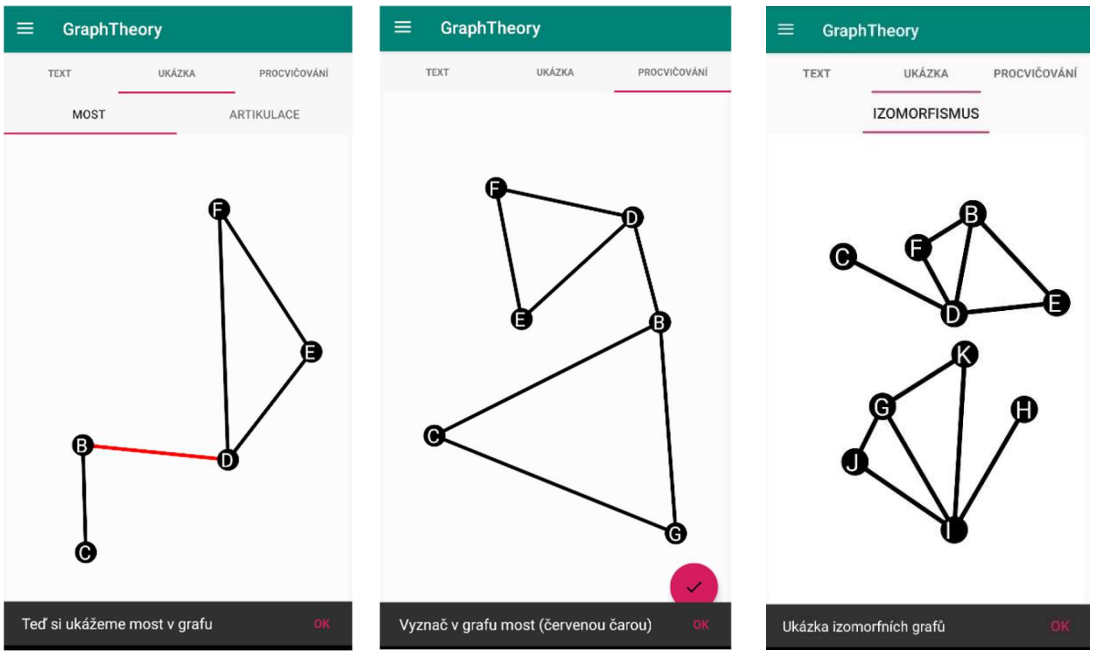
Obr. 18 Ukázka textu formátovaného pomocí KaTex
zdroj: vlastní zpracování

5.4 Paint

Pro vykreslování grafů byly vytvořeny dvě třídy, které umožňují zobrazování grafů. První slouží pro zobrazování ukázek a druhá třída pro tvorbu grafů uživateli aplikace. Vzhledem k podobné funkčnosti je popsána třída pro tvorbu grafů.

Třída mění chování v reakci na změnu vyvolanou v BottomNavigationView, které mění chování objektů na plátně. Listener poslouchá změny v podobě stisknutí a tažení prstem po obrazovce a v závislosti na volbě v menu vytváří, nebo posouvá objekty. Zároveň v případě změny aktualizuje všechny objekty na plátně (např. při tažení uzlem zároveň aktualizuje souřadnice vykreslovaných hran), nebo odstraňuje objekty související s právě odebraným objektem (např. při odebrání uzlu odebere také všechny navázané hrany). Třída se také stará o označení jednotlivých uzlů písmeny, kdy při přidání nového uzlu je zároveň přidáno písmeno označující daný vrchol, které se vykreslí uvnitř daného vrcholu. V případě tapnutí prstu byla přidána malá tolerance, kdy prst klikne na souřadnice blízké souřadnici vykreslovaného uzlu a ty jsou nadále považovány za klepnutí přímo na daný objekt.

V případě, že se nacházejí dva uzly blízko sebe, je vybrán uzel bližší souřadnicím klepnutím do obrazovky. Zároveň byla přidána metoda umožňující serializaci vykreslených grafů do objektu zmíněného v sekci Základní práce s objekty, která je využívána pro třídu kontrolující grafy vytvořené uživatelem. Třída umí i z předaného serializovaného objektu celý graf vykreslit, to je naopak použito při předávání grafových objektů od vytvořeného generátoru grafů. Ukázky 5vykresleného grafu jsou možné vidět na obrázku 19.



Obr. 19 Ukázky vykreslených grafů
zdroj: vlastní zpracování

5.5 Generování uzlů a hran

Většina výukových aktivit potřebuje ke svému fungování generátor grafů. Grafy se skládají hlavně z uzlů a hran. Byla tedy vytvořena třída GraphGenerator, která umí generovat na základě předaných atributů, jako jsou rozměry canvasu a počet vrcholů, jak vrcholy, tak hrany, k již vytvořeným vrcholům.

Během generování uzlů bylo třeba ohlídat několik situací, ke kterým by při náhodném umístění uzlů mohlo docházet. Jednou z nich je jejich překrývání, např. při situaci, kdy je vrchol (v případě této aplikace kruh v canvasu) vygenerován na souřadnicích velmi blízko umístěných jinému vrcholu. V takovém případě by došlo k zakrytí vrcholu, který by byl vykreslen dříve a tím

pádem by byl pro uživatele, do doby manipulace s uzlem vykresleným později, neviditelný. Musí být tedy ošetřena tato možnost během generování a další uzel vykreslen v dostatečné vzdálenosti od předchozího uzlu, viz ukázka kódu 17.

```
for (Coordinate coordinate : coordinateArrayList) {
  double D = Math.pow(xCoordinate - coordinate.x, 2) + Math.pow(yCoordinate - coordinate.y, 2);
      if (Math.sqrt(D) <= NODE_RADIUS){
         isInOtherCircle = true;
      }
}
```

Ukázka kódu 17 Kontrola vzdálenosti generovaného uzlu
zdroj: vlastní zpracování

Specifickým případem jsou okraje canvasu. V dolní části obrazovky se nachází navigace pro přepínání mezi jednotlivými stavy, v levé části naopak reaguje na pohyby navigationDrawer. Během testování (více v kapitole 6.3) bylo zjištěno, že někteří uživatelé nosí na svých zařízeních kryt, který jim ztěžuje přístup k okrajům telefonu. Při generování uzlů tedy byly ještě přidány okraje, které zvyšují vzdálenost, ve které se jednotlivé uzly generují dál od fyzických okrajů zařízení.

U generování hran bylo potřeba ošetřit více situací, které mohou nastat. Prvním takovým případem je osamocený uzel, tedy uzel, který není spojen s žádným jiným. V aplikaci je spojování uzlů řešeno pomocí SparseArray. Ten mapuje integery k objektům a měl by být úspornější než HashMapa. Umožňuje také vkládat objekty bez ohledu na pořadí jejich indexu, je tedy možné vložit do prázdného SparseArray např. objekt na index 0 a poté 33 [51]. To je, vzhledem k tomu, že v aplikaci jsou uzly uloženy v ArrayListu, využito právě na mapování, který uzel má být se kterým dalším spojen. Ve SparseArray se tedy nachází index objektu, který reprezentuje uzel, který bude propojován s dalšími, a objekt, v tomto případě seznam dalších indexů odkazujících se opět na seznam uzlů. V aplikaci je spojení vrcholu na nultém indexu s vrcholem na 3 a 4 indexu znázorněno v ukázce kódu 18.

```
SparseArray<ArrayList<Integer>> connectedNodes = new SparseArray<>();
ArrayList<Integer> indexesOfConnectedNodes = new ArrayList<>();
indexesOfConnectedNodes.add(3);
indexesOfConnectedNodes.add(4);
connectedNodes.put(0, indexesOfConnectedNodes);
```

Ukázka kódu 18 Spojování uzlů
zdroj: vlastní zpracování

Přestože se může zdát, že stačí prohledat všechny indexy uzlů, zda jsou obsaženy buď v ArrayListu, nebo mají hodnotu na svém indexu, mohlo by nicméně takto dojít k situacím, kdy budou dva uzly propojeny pouze spolu, ale se zbytkem spojeny nebudou. K tomu, aby se předešlo těmto situacím, bylo ještě potřeba nasadit algoritmus podobný hledání mostu v grafu. Tento algoritmus vezme náhodnou hranu a projde všechny uzly, na které dosáhne od obou uzlů na koncích hran. Tedy vezme další uzly, které jsou spojeny s jedním z konců vybrané hrany a které ještě algoritmus neprošel. Uzly si dá do zásobníku a dále prochází hrany spojené s uzlem v zásobníku a opět hledá ještě neobjevené vrcholy na koncích těchto hran. Jakmile nemá co dalšího k objevení, odstraní vrchol ze zásobníku a pokračuje dalším v řadě. Na konci spočítá, jestli tímto způsobem našel všechny uzly, a pokud ne, znamená to, že se v grafu vyskytuje uzel, který je osamocen. To následně vyvolá vygenerování nového grafu, který je opět tímto způsobem otestován, dokud není vygenerován graf, který je spojitý. Tímto způsobem je zajištěno, že není vygenerován graf s osamoceným vrcholem.

5.6 Generování ukázek a kontrola grafů

V každé aktivitě je od uživatele očekáváno splnění zadaného úkolu, který mu odemkne další studijní téma (další aktivitu), to vždy navazuje na předchozí ukázku, která je k dispozici na vedlejší záložce s názvem Ukázka. K ověření splnění úkolu byla vytvořena třída GraphChecker, ve které se nacházejí statické metody pro ověřování výsledků přijatých z paintView (plátna, na kterém uživatel pracoval s grafem). V následujících kapitolách jsou popsány vybrané principy pro generování ukázek z dané oblasti a ověření výsledku úkolu od uživatele ve zmíněné třídě GraphChecker. Název kapitoly vždy koresponduje s probíraným tématem a zároveň pořadí kapitol respektuje pořadí aktivit v menu aplikace. Algoritmy k jednotlivým kapitolám jsou k nahlédnutí ve zdrojových kódech, struktura aplikace je přiložena v příloze 1.

Vzhledem ke struktuře objektu Graph, který je použit pro předávání grafu v aplikaci, je pro všechny ukázky vygenerován graf a k němu příslušné spojení

označené červenými hranami. U generování hran v ukázce, na rozdíl od cvičení, není pro uživatele důležité navazování zvýrazněných hran na ty již vygenerované, neboť z pohledu uživatele není původní hrana vidět. Díky tomu si je možné dovolit jisté zjednodušení v některých případech.

5.6.1 Cesta, tah, kružnice, sled

Pro generování grafových ukázek algoritmus u tahu kontroluje, zda cesty navazují na sebe a jestli se v seznamu nevyskytují dvě hrany mezi stejnými uzly. Pro otevřený tah se kontroluje, zdali seznam obsahuje maximálně každý vrchol dvakrát (vzhledem k tomu, že ArrayList obsahuje seznam hran, které mají počáteční a koncový bod). Pro kružnici pouze kontroluje, jestli na sebe navazují hrany a jestli počáteční a koncová hrana končí ve stejném bodě. U cvičení algoritmus provádí stejné kroky a navíc kontroluje, zda všechny vytvořené hrany procházejí přes existující hrany, tato kontrola slouží u všech případů z procvičování, kdy má uživatel za úkol něco vyznačit a v dalších kapitolách tento fakt nebude již zmiňován.

5.6.2 Most a artikulace

V této aktivitě bylo potřeba vyřešit algoritmus, který by rozpoznal, jestli označený vrchol, případně hrana, je artikulací, nebo mostem. Tento algoritmus je velmi podobný tomu, který byl použit pro generátor grafů – v případě hledání mostu se vezme vrchol na konci označené hrany a projdou se všechny uzly, které s ním souvisí. Aby se neprocházely hrany dvakrát, jsou zduplikovány do druhého seznamu a v případě použití hrany (tedy nalezení právě hledaného souseda konkrétního uzlu, kde na jednom jeho konci je konkrétní uzel a na druhém nově objevený) je hrana ze zduplikovaného seznamu odebrána a dále už není zahrnuta při procházení hran. Nově objevené uzly jsou uloženy do zásobníku (seznamu) a dále se procházejí sousedé jednotlivých vrcholů. Zároveň je k tomu evidován seznam již navštívených vrcholů pro ošetření situací, kdy vede více hran k jednomu vrcholu. Pokud je objeven vrchol, který již byl navštíven, je hrana ze seznamu zduplikovaných hran odstraněna. Takto jsou projiti všichni sousedé jednoho koncového uzlu mostu a porovnává se, zda je počet

projitých uzlů roven n–1, kde *n* je počet všech uzlů v daném grafu a je třeba odečíst 1 vzhledem k tomu, že jsou hledány uzly nacházející se v sousedství nalezeného uzlu. Pro ověření toho, jestli se jedná o artikulaci grafu, je využito podobného principu.

5.6.3 Doplněk do grafu

K vytváření doplňku do grafu se prochází graf a kontrolují se hrany mezi jednotlivými uzly. Jakmile algoritmus narazí na neexistující hranu mezi dvěma uzly, je tato hrana přidána do seznamu a na konci je z těchto chybějících hran vytvořen další graf, který je doplňkem do grafu. V ukázce bylo potřeba nějakým způsobem znázornit dva grafy, aby na jednom byl vidět doplněk a na druhém původní graf. K těmto účelům byl vytvořen algoritmus zobrazující dva grafy nad sebou. Ten funguje na principu posunu souřadnic pod či nad polovinu zobrazovaného prostoru na daném zařízení. Pro kontrolu vytvořeného doplňku byl využit algoritmus vytvářející doplněk k vygenerovanému grafu – ten je následně porovnán s grafem nakresleným uživatelem a rozhodnuto, zda se jedná o doplněk.

5.6.4 Izomorfismus

Pro izomorfismus bylo využito algoritmu pro zobrazování dvou grafů nad sebou, stejně jako u doplňku do grafu. Generátorem je vytvořen graf, který je následně zduplikován a přeskládán na náhodné souřadnice, pak ho algoritmus přesune nad sebe. Ve cvičení je za pomocí náhodného čísla rozhodnuto, zda se vygeneruje izomorfní graf, stejně jako v ukázce, nebo se vygeneruje graf podobný. Jeho změna je v uzlu, který je přidán navíc, nebo náhodně odebrán (s patřičnými hranami). Uživatel si prohlédne graf a musí rozhodnout, jestli se jedná o izomorfní graf v tzv. Alertdialogu.

5.6.5 Bipartitní graf

V páté aktivitě jsou pro ukázky bipartitních grafů generovány vždy dvě skupiny uzlů za pomocí generátoru a tyto uzly jsou dále spojeny navzájem mezi sebou, čímž je dosaženo vytvoření bipartitního grafu. Pro cvičení jsou vytvořeny dvě úlohy; první, která podobně jako v předchozím úkolu uživatele vyzývá k rozpoznání,

zda se jedná o bipartitní graf, a druhá, která vyžaduje nakreslení bipartitního grafu. Pro první úlohu je využito podobného principu využívající náhodné číslo a následné změny v jednom uzlu v grafu. Pro kontrolu vytvořeného grafu uživatelem algoritmus vybere uzel a najde všechny hrany, které mají jeden konec v právě vybraném uzlu. Uzly, které jsou s ním propojeny, vloží do seznamu. Následně zkontroluje, že všechny tyto uzly jsou propojeny s těmi, které nebyly přidány do tohoto seznamu.

5.6.6 Hamiltonovský a Eulerův graf

U generování grafů pro hamiltonovskou kružnici a eulerův tah byla použita optimalizace, která zjednodušuje vygenerování takového grafu. Pomocí generátoru jsou vyprodukovány uzly a následně jsou mezi nimi vytvořeny hrany tak, aby byl graf připravený na vytvoření kružnice, či tahu. Pro ukázku je následně příslušnou kružnicí či tahem iterováno a jsou postupně přidávány do grafu v podobě červených hran, aby uživatel viděl, jak bylo uzly skrze hrany postupně procházeno. Při ověřování vytvořeného grafu algoritmus kontroluje, zda prochází kružnice skrze všechny vrcholy přes vygenerované hrany. Pro Eulerův tah se kontroluje, zdali na sebe jednotlivě označené hrany na sebe navazují (uživatel nepřerušil cestu během jejich označení) a jestli tah vede přes všechny hrany.

5.6.7 Skóre grafu

V aktivitě obsahující skóre grafu je generátorem vytvořen graf a následně se kontrolují všechny jednotlivé vrcholy a počítá se, kolik hran jimi prochází. Toto číslo je ukládáno do seznamu, který je následně setříděn (pro lepší zobrazení uživateli). U cvičení je dále vygenerováno skóre pro uživatele a následně je stejným způsobem (jako u ukázky) spočítáno skóre nakresleného grafu. Pro generování skóre je využito algoritmu Havel Hakimi, jehož princip spočívá v seřazováním skóre a následném odčítání jedničky od daného počtu skóre, které je opakováno, dokud nedorazí na konec, případně dřív, pokud algoritmus zjistí, že vygenerované skóre nemůže být skóre grafu. V případě, že algoritmus vyhodnotí, že se nejedná o skóre grafu, je generováno nové skóre do splnění podmínek algoritmu.

5.6.8 Strom a les

Pro generování stromu jsou za pomocí generátoru vytvořeny uzly, které jsou postupně propojovány od nejnižší *y* souřadnice, přičemž k některým uzlům jsou přidány další. Pro generování ukázky lesu je využito rozpůlení obrazovky vertikálně v polovině, kdy generátoru uzlů je předána poloviční *x* hodnota. Následný vygenerovaný druhý graf má pro všechny hodnoty *x* vynásobenou souřadnici 2, čímž je zobrazený druhý graf posunut na druhou polovinu obrazovky. Pro cvičení byly vytvořeny tři úlohy – nakreslení lesu, stromu a ověření, zda je možné nakreslit zadaný strom. Metoda, která kontroluje vytvořený graf, je pro obě úlohy stejná – přijímá v parametru počet komponent, které měl uživatel vytvořit. Během běhu metody je zkontrolováno, jestli je graf acyklický, tedy neobsahující kružnici. Algoritmus si pro ověření acykličnosti zkopíruje pro každý uzel seznam hran a vezme první uzel na druhé straně hrany, který je s právě vybraným uzlem spojený. Tuto hranu odebere ze zkopírovaného seznamu a dále pokračuje v hledání dalších uzlů, přičemž si vždy odebere použitou hranu ze seznamu. Pokud tímto způsobem nedojde zpět k prvnímu vybranému uzlu, vezme algoritmus případnou další hranu, která spojuje první vybraný uzel s dalším a opět opakuje postup. Takto projde všechny uzly v grafu, pokud do té doby nenarazí na kružnici. Jakmile je graf zkontrolován, následuje počítání komponent, při kterém algoritmus vezme náhodný vrchol a postupný procházením sousedních vrcholů si vede seznam navštívených vrcholů, dokud neprojde všechny sousední vrcholy. Poté zkontroluje, zda je počet navštívených vrcholů roven počtu uzlů v grafu, a pokud ne, pokračuje v prohledávání zbylé části grafu od náhodně vybraného uzlu, který se nevyskytuje v seznamu navštívených vrcholů. Takto pokračuje, dokud nenavštíví všechny vrcholy grafu.

5.6.9 Kostra grafu

Pro vytvoření ukázky kostry grafu byla úloha optimalizována vygenerováním uzlů za pomoci generátoru a následného propojení pomocí hran tak, že se vezme první uzel a ten je propojen s druhým uzlem v grafu. Takto jsou postupně spojeny jednotlivé uzly, jejichž hrany jsou zároveň označeny jako kostra. Dále jsou mezi

náhodné uzly, které ještě nejsou spojeny, přidány hrany, které je propojí. Tím, že kostra grafu je strom, který je acyklický, je využito pro ověření označené kostry algoritmu pro ověřování stromu. Před předáním vytvořeného grafu metodě jsou ještě vyznačené hrany překonvertovány do běžných hran.

5.7 Statistiky a ukládání dat do databáze

V této kapitole jsou popsány principy použité při tvorbě statistik uživatelů, která je dostupná ve vyjíždějícím menu. Během registrace je vytvořen objekt uživatel, jehož podoba je vidět v ukázce kódu 19. Tento objekt je při registraci uložen do databáze a některé jeho položky jsou dále aktualizovány v průběhu používání aplikace.

```
public class User implements Comparable{

    private String email;
    private String team;
    private double score;
    private int unlockTopics;
    private String uuID;
    private HashMap<String, Double> remainingPointsFromActivity;
    private String nickName;

}
```

Ukázka kódu 19 Třída User
zdroj: vlastní zpracování

Jakmile uživatel splní nějaké zadání z procvičovaných témat, je zavolána metoda recordUserPoints z třídy DatabaseConnector, která aktualizuje skóre daného uživatele a zároveň zkontroluje, zda nedošlo k odemčení dalšího tématu.

```
DatabaseConnector databaseConnector = new DatabaseConnector();
databaseConnector.recordUserPoints(userName, "fourth");
```

Ukázka kódu 20 Zaznamenání splněného procvičování
zdroj: vlastní zpracování

Metoda přijímá jako parametr jméno uživatele a identifikátor splněné aktivity. V metodě je prohledána HashMapa, zda obsahuje záznam s daným klíčem. Pokud ano, přičte se daná hodnota ke skóre a hodnota se sníží o jedna (pokud již není nulová). Pokud pro daný klíč hodnota není, je příslušný záznam vytvořen, viz ukázka kódu 20. Třída DatabaseConnector následně zkontroluje, jestli se počet

záznamů nezvýšil, a pokud ano, aktualizuje hodnotu unlockedTopics, která je využívána navigationDrawerem pro ověření, zdali je nějaké téma již odemčeno a může být zobrazeno. Tento přístup by měl uživatele motivovat k odemykání témat postupným procházením jednotlivých vytvořených cvičení. Při zaznamenání bodů či aktualizaci hodnot dochází rovnou k volání metod zapisující hodnoty do databáze.

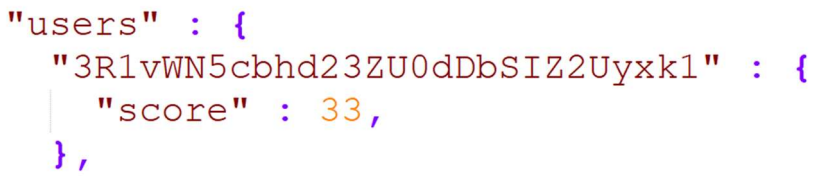
Firebase ukládá data ve formátu JSON. Na rozdíl od SQL databází zde nejsou žádné tabulky. Jakmile jsou přidána data, ukládají se jako strom ve formátu JSON pod příslušným klíčem. Jednotlivé klíče mohou být vybrány uživatelem, důležité je, aby byly unikátní. Zároveň je možné data do sebe libovolně zanořovat až do 32. úrovně. Nicméně je potřeba to provádět s rozvahou, neboť špatnou strukturou může snadno docházet ke ztrátě výkonu aplikace a jako „best practice“ se označuje metoda zploštění dat, která data co nejméně zanořuje [52] .

V ukázce kódu 21 je vidět, jakým způsobem probíhá volání zapsání hodnot do databáze. Je potřeba určit první uzel, do kterého se budou zapisovat data, a pak určit vnořené uzly, do kterých se bude zapisovat.

```
val database = FirebaseDatabase.getInstance()
database.getReference("users").child(user.uuID).child("score").setValue(score)
```

Ukázka kódu 21 Zápis hodnot do databáze v jazyce Kotlin
zdroj: vlastní zpracování

Výsledkem tedy bude JSON s dvěma vnořenými uzly a hodnotou pro dané uživatelovo skóre, viz obrázek 20.



Obr. 20 Výsledný JSON po provedení zápisu do databáze
zdroj: Vlastní zpracování

Při načtení dat z databáze se postupuje obdobným způsobem jako pro zápis. Je třeba definovat kořen, ze kterého se budou vyčítat všechny listy, a dále je vytvořen listener, který je volán (asynchronně) při každé změně vyvolané na databázi. Listener vrací objekt DataSnapshot, ze kterého je možné

dostat zpětně objekt, který odpovídá uložené reprezentaci dat. Metoda onCancelled je provolána v případě, pokud uživatel nemá oprávnění číst data z databáze, viz ukázka kódu 22. [53]

```
val ref = database.getReference("users")
  val postListener = object : ValueEventListener {
    override fun onDataChange(dataSnapshot: DataSnapshot) {
      for (postSnapshot in dataSnapshot.children) {
          val user = postSnapshot.getValue(User::class.java)
        if (user != null) {
          users.add(user)
        }
      }
    }
    override fun onCancelled(databaseError: DatabaseError) {
      // Getting Post failed, log a message
    }
  }
ref.addValueEventListener(postListener)
```

Ukázka kódu 22 Získání dat z databáze v jazyce Kotlin
zdroj: vlastní zpracování

Stejně jako u registrace probíhá práce s databází asynchronně. To znamená, že zapsání hodnot neproběhne okamžitě, ale ani načtení hodnot z databáze neproběhne ihned.

V aktivitě statistik jsou, dle návrhu, zvoleny tři možnosti zobrazení – všichni uživatelé, uživatelé ve stejném týmů a jednotlivé týmy, viz obrázek 21. Řazení je zvoleno sestupně, hodnoty skóre jsou řazeny za pomocí Comparatoru a obaleny jsou v CardView (grafický kontejner pro zobrazování stejných dat [54]).

K zobrazení je využito listu zvaného RecyclerView umožňujícího zobrazení velkého množství dat, nebo dat, která se často mění. Výhodou tohoto listu je, že si nedrží v paměti všechny položky, které se můžou zobrazit. Drží právě tolik, kolik je potřeba k jejich zobrazení na obrazovce daného zařízení plus několik předcházejících a následujících, aby uživatel nečekal při posouvání seznamem. [55]

Pro vytvoření RecyclerView je třeba přidat RecyclerView do zdrojového XML a vytvořit k němu tzv. Adaptér. Adaptér je třída, která předává data, která se budou zobrazovat, a zároveň říká, jakým způsobem budou data zobrazena. Třída obsahuje tedy seznam dat k zobrazení a vnitřní třídu, kde je specifikováno, jak budeme jednotlivé záznamy reprezentovat (v jakém formátu). V ukázce kódu 23

jsou vidět tři metody, které je potřeba overridovat. V první je specifikováno zdrojové XML, které se bude v jednotlivých řádcích listu plnit, přičemž třída, kterou vracíme, je vnitřní třída adapteru. V druhé metodě jsou plněny parametry vnitřní třídy daty a poslední metoda předává velikost zobrazovaných dat View.

Obr. 21 Statistika uživatelů v aplikaci
Zdroj: vlastní zpracování

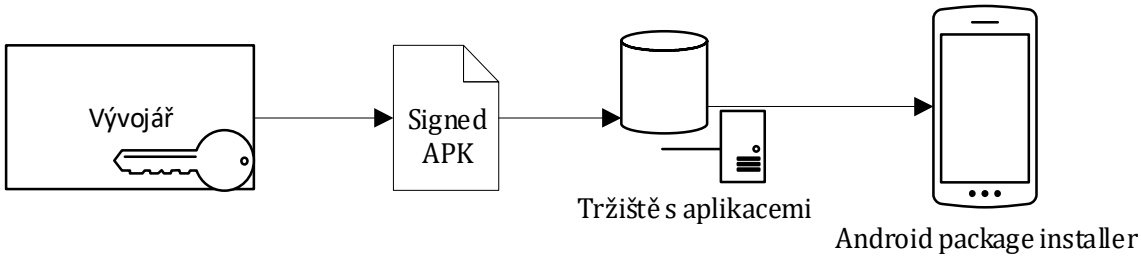
```
@Override
public UserViewHolder onCreateViewHolder(ViewGroup parent,
                    int viewType) {
  View v = LayoutInflater.from(parent.getContext())
      .inflate(R.layout.recycler_statistic_row, parent, false);

  return new UserViewHolder(v);
}

@Override
public void onBindViewHolder(UserViewHolder holder, int position) {
  // - get element from your dataset at position
  User user = users.get(position);
  holder.userName.setText(user.getNickName());
  holder.userScore.setText(String.valueOf(user.getScore()));
  holder.userTeam.setText(user.getTeam());
}

@Override
public int getItemCount() {
  return users.size();
}
```

Ukázka kódu 23 Nastavování hodnot RecyclerView
zdroj: vlastní zpracování

6 Publikace a testování

6.1 Publikace aplikace

Po dokončení aplikace se mohlo přejít k fázi testování na uživatelích. K tomu byla vygenerována tzv. signed apk. Jedná se o instalační soubor aplikace, který je vývojářem digitálně podepsaný. Android kontrolou certifikátu zjišťuje, zda není instalační soubor podvrhnutý. Je tedy důležité si tento klíč pro další verze aplikace dobře uschovat, bez něho není možné instalovat aktualizace v rámci aktualizačního procesu obchodu, kde je aplikace umístěna. Během instalace instalátor balíčků kontroluje certifikát (resp. veřejný klíč v certifikátu) a pokud se neshoduje s předchozí verzí, je instalace zamítnuta. Certifikát pro aplikaci je vytvořen výrobcem, operátorem, alternativním uložištěm nebo vývojářem samotným, není tedy potřeba žádná centrální autorita poskytující tento typ certifikátů. Certifikáty jsou v OS Android již od počátku, současně ve verzi 3, který je ale podporován až od 9. verze OS. Při kontrole se nejdříve zjišťuje, jestli je k dispozici certifikát v nejnovější verzi a teprve poté se hledá ten ze starší generace. Vzhledem k minimální verzi Androidu v projektu byla použita verze 2, která už hashuje obsah apk [56]. Na obrázku 22 je daný proces znázorněn pro obecné tržiště aplikací.



Obr. 22 Proces podpisu aplikace
zdroj: vlastní zpracování

Po podepsání aplikace vývojář požádá dané tržiště s aplikacemi o její schválení k publikaci. V případě Google Play není dodnes jisté, zdali proces schvalování provádějí lidští testeři, nebo se jedná o automatizovaný proces. Android Police uvádí, že Google pravděpodobně testuje ručně ty aplikace, které označí automatický schvalovací proces [57]. V případě publikování aplikace do Amazon App Storu

je třeba si vytvořit účet u Amazonu, vytvořit popis aplikace, přidat snímky obrazovky k popisu, nastavit tzv. Content Rating (hodnocení škodlivosti obsahu na mladistvých a dětech), určit si cenu za stažení aplikace a nahrát signed apk. U Amazonu však do procesu vstupují lidští testeři, ale není jisté, zda u každé nahrané verze, nicméně během procesu schvalování přicházejí e-maily ohledně dotazů k aplikaci, na které je třeba skrze formulář na stránkách Amazonu odpovědět. Během ověřování může docházet k nedorozuměním mezi testovacím týmem a autorem aplikace. V případě schvalovacího procesu přišla zpráva ohledně poskytnutí přihlašovacích údajů do aplikace, a i přes odpověď skrze nabízený formulář přišel stejný dotaz ještě dvakrát a až na třetí vyplnění e-mailu již zareagovala druhá strana kladně. Bylo zarážející, že při nahrávání nové verze se problém s přihlašováním testerů opakoval, a to i přesto, že přihlašovací údaje pro jejich účet zůstaly stejné a ve formuláři se ani nezměnil obsah testového pole. Je možné jen odhadovat, jestli se testování konkrétního požadavku věnuje pouze jeden, nebo více lidských testerů. Jakmile je aplikace schválena k uvedení, přijde informační e-mail, že je aplikace k dispozici ke stažení.

6.2 Stažení aplikace

Pro instalaci aplikace je potřeba si stáhnout instalační soubor APK. Ten je poskytován zpravidla různými internetovými tržišti. V případě stažení aplikace skrze Google Play stačí navštívit zpravidla vestavěnou aplikaci v telefonu a příslušnou aplikaci si vybrat. V případě Amazon App Storu (stav k počátku roku 2020) je potřeba jít na jejich stránky, kde rovnou vyskočí nabídka se stažením příslušného souboru. Tento proces je třeba dělat pro všechna alternativní tržiště, neboť Google Play neumožňuje stahovat aplikace, které dále instalují do zařízení další aplikace. Po instalaci obchodu se v něm uživatel musí zaregistrovat a následně v ní nalézt aplikaci GraphTheory (název aplikace, která byla tvořena v popisu výše) a tu si stáhnout. Na začátku roku 2020 ještě Amazon neumožňoval vyhledávání s mezerou, pokud tedy uživatel vyhledal název „Graph Theory“, ve výsledcích aplikaci nenašel. Tento postup byl pro uživatele zpracován a distribuován ve formě PDF (viz příloha Návod na instalaci aplikace).

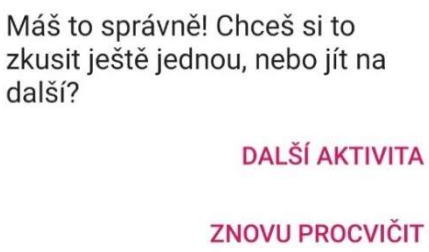
6.3 Testování aplikace

Stejně jako u jakéhokoliv jiného informačního systému proběhla před publikací této aplikace testovací fáze. Základní testování probíhalo skrze Unit testy v rámci vývoje aplikace. Aplikace byla vyzkoušena na úzké skupině zařízení a uživatelů. V rámci testování se podařilo odladit několik chyb a zlepšit uživatelské prostředí. V další části jsou uvedené změny, které byly zapracované na základě výsledků testování.

Jednou z úprav prošel systém výukových aktivit. V prvních verzích aplikace nebyla dostupná vícevrstvá navigace, jednotlivá témata se měnila po splnění úkolu ze sekce Procvičování. To vedlo k nedorozuměním v ovládání aplikace a plnění samotných aktivit. Nově byla přidána druhá vrstva navigačního menu v záložce ukázky pro aktivity s více ukázkami, viz obrázek 23.

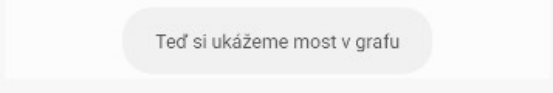
Obr. 23 Vícevrstvá navigace
Zdroj: vlastní zpracování

Další změnou prošlo oznamování uživateli o splněném cvičení. Původně bylo předpokládáno, že uživatel po splnění cvičení přejde automaticky na další cvičení. Nově přibyla možnost zobrazení skrze AlertDialog, která mu umožňuje zvolit si, zda chce pokračovat v další výukové aktivitě, nebo si procvičí stejnou aktivitu ještě jednou. V případě znovu procvičení je mu vygenerováno nové zadání na stejné téma, viz obrázek 24.

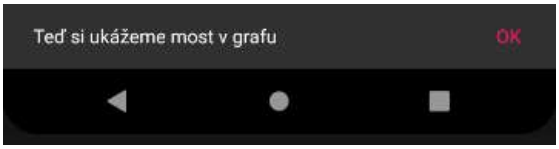


Obr. 24 AlertDialog po splnění aktivity
Zdroj: vlastní zpracování

Způsob předávání zadání uživatelům bylo změněno. Nejdříve byly zprávy zobrazovány pomocí Toastových zpráv. Ty byly pro zobrazování zadání dostatečné z hlediska textu, který byly schopny zobrazit. Nedostatečné však byly z pohledu doby zobrazení textu. Android umožňuje délku zobrazit pomocí konstanty, která je však reprezentována konstantami v podobě *Toast.LENGTH_LONG* nebo *Toast.LENGTH_SHORT*, případně číselnou reprezentací od 0 do 1. Zobrazení zprávy delší dobu však systém neumožňuje. Byla to zároveň jedna z častějších výtek od uživatelů, kteří aplikaci testovali v raných fázích. Byla tedy vybrána jiná knihovna pro zobrazování zpráv – Snackbar. Ten umožňuje zobrazit text na libovolně dlouhou dobu. Ve výchozím nastavení se zobrazují pouze dva řádky textu, což může být, vzhledem k dostupnému prostoru na displeji, omezující, ovšem počet řádků lze upravit voláním metody *setMaxLines()*. Na obrázku 25 a 26 je vidět výslednou změnu.



Obr. 26 Původní zobrazení zprávy skrze Toast Message
Zdroj: vlastní zpracování



Obr. 25 Zobrazení zprávy pomocí Snackbaru
Zdroj: vlastní zpracování

Z uživatelského pohledu byla v dolním menu aplikací odstraněna možnost pro smazání zobrazeného grafu u cvičení, neboť uživatelé hlásili, že tuto možnost občas nechtěně zvolí a zmizí jim celý graf. Uvolněním této možnosti z menu se zvětšila plocha pro další možnosti, které byly snáze dostupné.

Během vývoje bylo třeba opravit a vyřešit několik technických problémů. Jednou z nich bylo vyřešení volání metod ve třídě Paint po dokončené inicializaci View. Vzhledem k tomu, že generování grafů je tvořeno už při startu aktivit, bylo třeba vytvořit mechanismus, který zavolá metodu pro předání grafu ve správný čas. Pro tyto účely byl vytvořen CommunicationInterface, který je zavolán

z Fragmentu ve chvíli, kdy je View úspěšně vytvořeno a v rámci jeho implementace v různých třídách se teprve volá generování grafů a jejich následné předání třídě Paint.

Dalším vybraným technickým problémem bylo uchovávání označení uzlů při jejich překreslení. To bylo vyřešeno implementací metody, která ukládá přiřazený název uzlu do hashmapy, která je v rámci dalších vykreslení aktualizována o změny a označení jsou opět vygenerována stejná pro dané uzly.

Po změnách zapracovaných po testování na úzké skupině lidí se bylo třeba připravit na situace, které mohou nastat, jakmile bude aplikace publikována širší skupině uživatelů. U užší skupiny uživatelů nebylo problém napojit jejich zařízení na vývojové prostředí a odladit případné pády nebo chyby přímo na místě, to však nelze dělat u širší skupiny uživatelů. Z tohoto důvodu byl vybrán nástroj na reportování chyb, který umožňuje zaznamenávat jednotlivé pády a společně s logy je nahrát na server. Takovýmto nástrojem je Crashlytics od společnosti Google. Google ho poskytuje společně s databází Firebase a na rozdíl od ní jeho používání nepodmiňuje žádným omezeným provozem a nabízí ho v současné době zcela zdarma. Jeho použití je z hlediska vývojáře využívajícího služby Firebase ve své aplikaci velmi snadné, stačí Crashlytics přidat do závislostí. Jakmile dojde k pádu aplikace, je obsah zaznamenávaných logů poslán na server Googlu, kde je možné si jej přečíst. Vývojář tedy vidí to samé, co by viděl v konzoli (logcatu) vývojového prostředí a může se pokusit opravit problém, aniž by musel daného uživatele kontaktovat. Během testování v širším okruhu uživatelů nedošlo k pádům aplikace.

7 Shrnutí výsledků

Cílem práce bylo vytvoření nástroje pro podporu výuky teorie grafů za pomocí prvků gamifikace. Ten byl vytvořen pro mobilní telefony běžící na operačním systému Android s využitím převážně vestavěné knihovny. Aplikace umožňuje procházet studijní texty a prohlížet si k nim vygenerované ukázky. Zároveň nabízí ke každému tématu možnost jeho procvičení.

Během fáze testování byly odladěny některé chyby způsobené úmyslnou, či neúmyslnou prací s aplikací, jako např. ošetření vstupních polí, případně pády aplikace během prvotního testování. Po publikování aplikace v externím tržišti již závažnější chyby, které by vedly k pádu aplikace, hlášeny nebyly.

Před představením aplikace studentům byl vytvořen dotazník, který zjišťoval, jaké jsou reakce na vytvořenou aplikaci. Dotazník také umožňoval přidávat zpětnou vazbu s nápady na rozšíření aplikace a další témata, která by uživatelé rádi v aplikaci viděli. Vzhledem k nízké návratnosti dotazníku (5 %) není možné činit větší závěry. Přesto tito uživatelé hodnotili aplikaci jako užitečný doplněk při studiu. Zároveň uvedli, že aplikace na jejich zařízení běží bez zásek a instalaci z alternativního zdroje hodnotili jako zvládnutelnou vzhledem k vytvořenému návodu k instalaci aplikace z externího tržiště.

Na základě dosažených výsledků lze potenciál mobilních zařízení pro podporu výuky hodnotit jako dostatečný a je v rámci takto vytvořené aplikace možné hovořit o použitelném doplňku při studiu daných témat z teorie grafů.

8 Závěry a doporučení

Gamifikace se ukazuje jako nástroj, kterým lze motivovat studenty různých oborů k lepším studijním výsledkům. Sanchez zkoušel použít gamifikaci při tvorbě online kurzů a došel k pozitivním výsledkům její implementace [1], Nakashima využil prvků gamifikace při tvorbě aplikace pro měření počtu kroků a z výsledků svého výzkumu zjistil, že někteří účastníci změnili své chování ve snaze získat více bodů [11]. To, jestli bude mít gamifikace pozitivní vliv na používání této aplikace, ukáže až čas po zavedení do výuky. V současném stavu se zdá, že aplikace má všechny potřebné vlastnosti pro její úspěšné zavedení. Je potřeba dohlédnout, jak zmiňuje Nakashima [11], aby počet uživatelů nebyl příliš malý a tím neohrozil možnou soutěživost, kterou gamifikace přináší.

Byť se aplikace ukazuje jako použitelný nástroj pro výuku, jsou zde ještě prostory ke zlepšení. Jednou z věcí, kterou by bylo dobré do budoucna odladit, je zobrazení grafů na různých zařízeních. Vzhledem k různým velikostem displejů, které aplikaci zobrazují, by bylo dobré vytvořit algoritmus, který by na základě dostupných informací o rozlišení a nastavené velikosti textu přepočítával velikost zobrazených elementů v grafu. V současné době je tento algoritmus obtížné odladit z důvodu malé dostupnosti testovacích zařízení.

Do budoucna by se aplikace mohla rozšířit i o další témata z teorie grafů a přidat podporu např. pro orientované grafy. Mohla by se také rozrůst o podporu zobrazení grafů ve formě tabulky, nebo o exportování a importování grafů pro jiná zařízení. Zajímavá by byla možnost testování všech dostupných témat, které uživateli dělají největší problém, ty by se v rámci procvičování mohly zobrazovat častěji. Těmto rozšířením je možné se v budoucnu věnovat.

9 Seznam použité literatury

[1] SANCHEZ, Diana R., Markus LANGER a Rupinder KAUR. Gamification in the classroom: Examining the impact of gamified quizzes on student learning. *Computers & Education* [online]. 2020, **144**, 103666 [cit. 2020-01-29]. ISSN 0360-1315. Dostupné z: doi:10.1016/j.compedu.2019.103666

[2] 9 Examples of Apps Using Gamification for User Engagement. *GetSocial* [online]. 17. říjen 2019 [cit. 2020-04-06]. Dostupné z: https://blog.getsocial.im/is-gamification-the-only-way-for-apps-to-survive/

[3] PRESS, Livia a Alessandro VIEIRA DOS REIS. Sesame Credit and the Social Compliance Gamification in China. In: . 2019.

[4] MILKOVÁ, Eva. *Teorie grafů a grafové algoritmy*. Hradec Králové: Gaudeamus, 2013. ISBN 978-80-7435-267-6.

[5] SEDLÁČEK, Jiří. *Úvod do teorie grafů*. 3. vyd. Praha: Academia, 1981.

[6] VAN ROY, Rob a Bieke ZAMAN. Need-supporting gamification in education: An assessment of motivational effects over time. *Computers & Education* [online]. 2018, **127**, 283–297 [cit. 2020-02-01]. ISSN 0360-1315. Dostupné z: doi:10.1016/j.compedu.2018.08.018

[7] ZAINUDDIN, Zamzami, Samuel Kai Wah CHU, Muhammad SHUJAHAT a Corinne Jacqueline PERERA. The impact of gamification on learning and instruction: A systematic review of empirical evidence. *Educational Research Review* [online]. 2020, **30**, 100326 [cit. 2020-04-06]. ISSN 1747-938X. Dostupné z: doi:10.1016/j.edurev.2020.100326

[8] DICHEVA, D., C. DICHEV, G. AGRE a G. ANGELOVA. Gamification in education: A systematic mapping study. *Educational Technology and Society*. 2015, **18**(3), 75–88.

[9] LANDERS, Richard N. a Rachel C. CALLAN. Casual Social Games as Serious Games: The Psychology of Gamification in Undergraduate Education and Employee Training. In: Minhua MA, Andreas OIKONOMOU a Lakhmi C. JAIN, ed. *Serious Games and Edutainment Applications* [online]. London: Springer, 2011 [cit. 2020-01-29], s. 399–423. ISBN 978-1-4471-2161-9. Dostupné z: doi:10.1007/978-1-4471-2161-9_20

[10] MITCHELL, Robert, Lisa SCHUSTER a Hyun Seung JIN. Gamification and the impact of extrinsic motivation on needs satisfaction: Making work fun? *Journal of Business Research* [online]. 2020, **106**, 323–330 [cit. 2020-02-01]. ISSN 0148-2963. Dostupné z: doi:10.1016/j.jbusres.2018.11.022

[11] NAKASHIMA, Ryota, Takahiro SATO a Takuya MARUYAMA. Gamification Approach to Smartphone-app-based Mobility Management. *Transportation*

Research Procedia [online]. 2017, **25**, World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016, 2344–2355 [cit. 2020-02-02]. ISSN 2352-1465. Dostupné z: doi:10.1016/j.trpro.2017.05.234

[12] ŠITINA, J. *Grafové algoritmy a jejich využití*. B.m., 2010. Univerzita Hradec Králové, Fakulta informatiky a managementu.

[13] HÜBNER, A. *Aplikace pro podporu výuky předmětu Disktrétní matematika*. Hradec Králové, 2014. Bakalářská práce. Univerzita Hradec Králové.

[14] RŮTOVÁ ŠŤASTNÁ, Markéta. *Vizuální podpora výuky předmětů zabývajících se teorií grafů a grafovými algoritmy* [online]. B.m., 2017 [cit. 2020-04-19]. Univerzita Hradec Králové, Fakulta informatiky a managementu. Dostupné z: https://theses.cz/id/ab65wp/

[15] SKOŘEPA, Tomáš. *Vizuální podpora výuky předmětů zabývajících se teorií grafů a grafovými algoritmy* [online]. B.m., 2018 [cit. 2020-04-19]. Univerzita Hradec Králové, Fakulta informatiky a managementu. Dostupné z: https://theses.cz/id/4rwhxs/

[16] 60+ useful graph visualization libraries. *KDnuggets* [online]. [cit. 2020-01-18]. Dostupné z: https://www.kdnuggets.com/60-useful-graph-visualization-libraries.html/

[17] Mobile OS market share 2019. *Statista* [online]. [cit. 2020-04-06]. Dostupné z: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/

[18] *Windows 10 Mobile End of Support: FAQ - Windows Help* [online]. [cit. 2019-12-29]. Dostupné z: https://support.microsoft.com/en-us/help/4485197/windows-10-mobile-end-of-support-faq

[19] *IDC - Smartphone Market Share - OS* [online]. [cit. 2019-12-29]. Dostupné z: https://www.idc.com/promo/smartphone-market-share/os

[20] The history of Android OS: its name, origin and more. *Android Authority* [online]. 18. srpen 2019 [cit. 2019-12-31]. Dostupné z: https://www.androidauthority.com/history-android-os-name-789433/

[21] StatCounter Global Stats - Browser, OS, Search Engine including Mobile Usage Share. *StatCounter Global Stats* [online]. [cit. 2019-12-30]. Dostupné z: https://gs.statcounter.com/

[22] *Distribution dashboard | Android Developers* [online]. [cit. 2019-12-30]. Dostupné z: https://developer.android.com/about/dashboards?hl=cs

[23] Why Your Android Phone Isn't Getting Operating System Updates and What You Can Do About It. *How-To Geek* [online]. [cit. 2020-01-02]. Dostupné

z: https://www.howtogeek.com/129273/why-your-android-phone-isnt-getting-operating-system-updates-and-what-you-can-do-about-it/

[24] What Is an APK File and What Does It Do? *MakeUseOf* [online]. [cit. 2019-12-31]. Dostupné z: https://www.makeuseof.com/tag/what-is-apk-file/

[25] Publishing your first app in the Play Store: what you need to know. *Android Authority* [online]. 20. květen 2014 [cit. 2019-12-31]. Dostupné z: https://www.androidauthority.com/publishing-first-app-play-store-need-know-383572/

[26] *gnu.org* [online]. [cit. 2020-01-02]. Dostupné z: https://www.gnu.org/philosophy/free-sw.html

[27] *Test Criteria for Amazon Appstore Apps | App Testing* [online]. [cit. 2020-01-02]. Dostupné z: https://developer.amazon.com/docs/app-testing/test-criteria.html

[28] ME, Droid By. Activity Life cycle of Android. *Medium* [online]. 29. prosinec 2017 [cit. 2020-01-02]. Dostupné z: https://medium.com/@droidbyme/activity-life-cycle-of-android-2e298809df6a

[29] Processes and Application Lifecycle. *Android Developers* [online]. [cit. 2020-01-02]. Dostupné z: https://developer.android.com/guide/components/activities/process-lifecycle?hl=cs

[30] *Understand the Activity Lifecycle | Android Developers* [online]. [cit. 2020-01-03]. Dostupné z: https://developer.android.com/guide/components/activities/activity-lifecycle#onstart

[31] *Fragments | Android Developers* [online]. [cit. 2020-01-03]. Dostupné z: https://developer.android.com/guide/components/fragments

[32] What is BaaS? | Backend-as-a-Service vs. Serverless. *Cloudflare* [online]. [cit. 2020-01-05]. Dostupné z: https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/

[33] JUMBAM, Amosa. MBaaS: What Is It And Why You Should Use It When Building Your Apps. *TheTool* [online]. 12. červen 2017 [cit. 2020-01-05]. Dostupné z: https://thetool.io/2017/mbaas-for-app-developers

[34] What Is Mobile Backend As A Service (MBaaS)? | Backendless MBaaS. *Backendless* [online]. 27. listopad 2019 [cit. 2020-01-05]. Dostupné z: https://backendless.com/what-is-mobile-backend-as-a-service-mbaas/

[35] How To Choose The Best Mobile Backend As A Service (MBaaS)? *DevTeam.Space* [online]. 26. červenec 2018 [cit. 2020-01-06]. Dostupné z: https://www.devteam.space/blog/how-to-choose-the-best-mobile-backend-as-a-service-mbaas/

[36] Firebase Pricing. *Firebase* [online]. [cit. 2020-01-06]. Dostupné z: https://firebase.google.com/pricing?hl=cs

[37] *Balsamiq. Rapid, effective and fun wireframing software.* | *Balsamiq* [online]. [cit. 2019-12-27]. Dostupné z: https://balsamiq.com/

[38] Update UI components with NavigationUI. *Android Developers* [online]. [cit. 2019-12-28]. Dostupné z: https://developer.android.com/guide/navigation/navigation-ui?hl=cs

[39] *SVG Repo - Free SVG Vectors and Icons* [online]. [cit. 2020-01-27]. Dostupné z: https://www.svgrepo.com/

[40] Add Firebase to your Android project. *Firebase* [online]. [cit. 2020-01-06]. Dostupné z: https://firebase.google.com/docs/android/setup?hl=cs

[41] Android Studio: An IDE built for Android. *Android Developers Blog* [online]. [cit. 2020-01-06]. Dostupné z: https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html

[42] Material Design for Android. *Android Developers* [online]. [cit. 2020-01-07]. Dostupné z: https://developer.android.com/guide/topics/ui/look-and-feel?hl=cs

[43] FragmentTransaction. *Android Developers* [online]. [cit. 2020-01-08]. Dostupné z: https://developer.android.com/reference/androidx/fragment/app/FragmentTransaction?hl=cs

[44] ViewModel Overview. *Android Developers* [online]. [cit. 2020-01-08]. Dostupné z: https://developer.android.com/topic/libraries/architecture/viewmodel?hl=cs

[45] BottomNavigationView. *Android Developers* [online]. [cit. 2020-01-14]. Dostupné z: https://developer.android.com/reference/com/google/android/material/bottomnavigation/BottomNavigationView?hl=cs

[46] FRANKS, Rebecca. Getting Started with Drawing on the Android Canvas . *Medium* [online]. 6. červenec 2019 [cit. 2019-12-30]. Dostupné z: https://medium.com/over-engineering/getting-started-with-drawing-on-the-android-canvas-621cf512f4c7

[47] What is a boilerplate code? *Educative: Interactive Courses for Software Developers* [online]. [cit. 2020-01-11]. Dostupné z: https://www.educative.io/edpresso/what-is-a-boilerplate-code

[48] TextView. *Android Developers* [online]. [cit. 2020-01-19]. Dostupné z: https://developer.android.com/reference/android/widget/TextView?hl=cs

[49] *KaTeX – The fastest math typesetting library for the web* [online]. [cit. 2020-01-19]. Dostupné z: https://katex.org/

[50] CDATASection. *Android Developers* [online]. [cit. 2020-01-19]. Dostupné z: https://developer.android.com/reference/org/w3c/dom/CDATASection?hl=cs

[51] SparseArray. *Android Developers* [online]. [cit. 2020-01-15]. Dostupné z: https://developer.android.com/reference/android/util/SparseArray?hl=cs

[52] Structure Your Database | Firebase Realtime Database. *Firebase* [online]. [cit. 2020-01-14]. Dostupné z: https://firebase.google.com/docs/database/ios/structure-data?hl=cs

[53] Read and Write Data on Android | Firebase Realtime Database. *Firebase* [online]. [cit. 2020-01-14]. Dostupné z: https://firebase.google.com/docs/database/android/read-and-write?hl=cs

[54] Create a Card-Based Layout. *Android Developers* [online]. [cit. 2020-01-15]. Dostupné z: https://developer.android.com/guide/topics/ui/layout/cardview?hl=cs

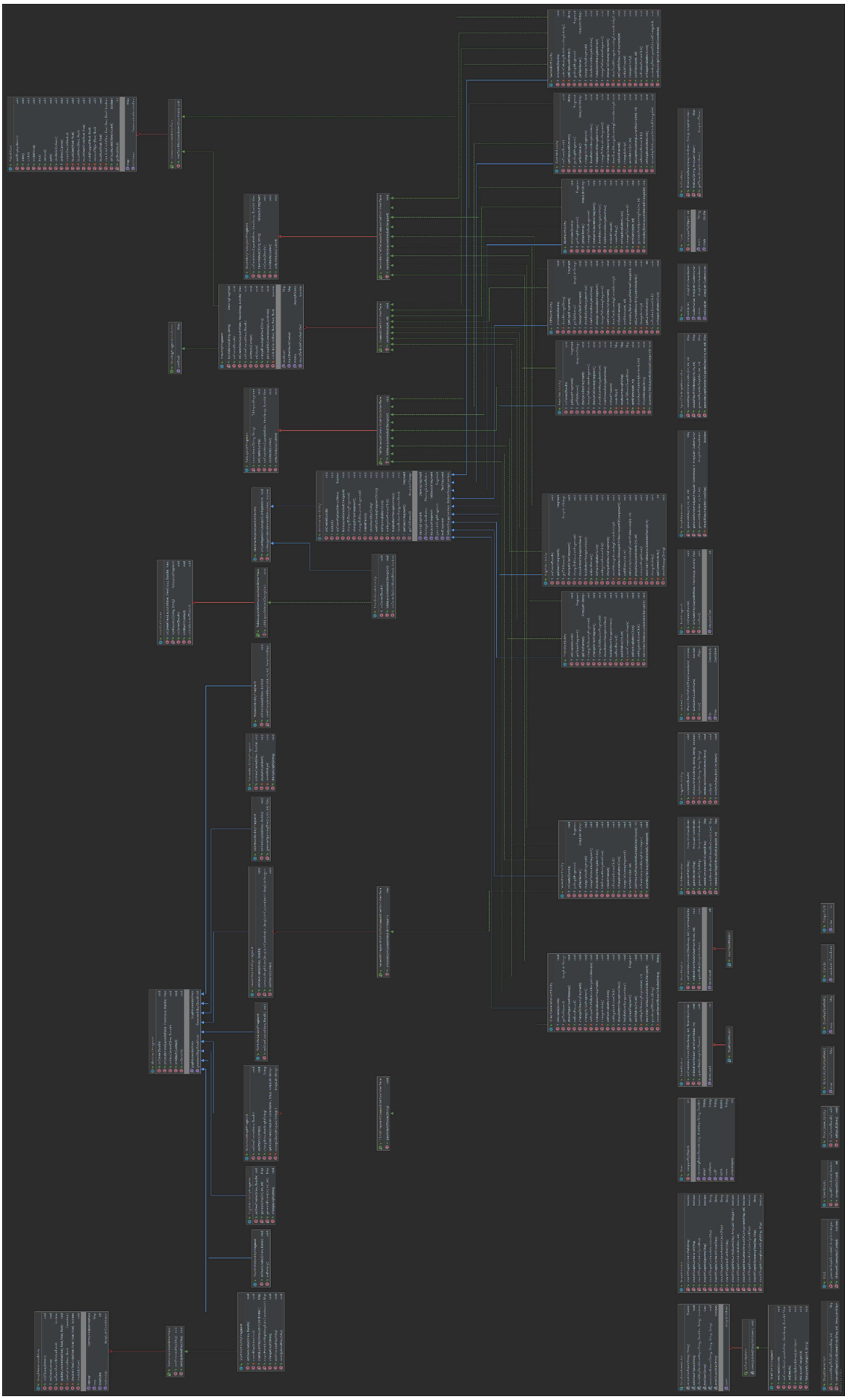
[55] Create a List with RecyclerView. *Android Developers* [online]. [cit. 2020-01-18]. Dostupné z: https://developer.android.com/guide/topics/ui/layout/recyclerview?hl=cs

[56] Application Signing. *Android Open Source Project* [online]. [cit. 2020-02-04]. Dostupné z: https://source.android.com/security/apksigning?hl=cs

[57] [Update: Delays may exceed 7 days] Google Play Store silently extends app review to all submissions, fails to inform developers. *Android Police* [online]. 3. září 2019 [cit. 2020-01-27]. Dostupné z: https://www.androidpolice.com/2019/09/03/google-play-expands-store-app-review/

10 Přílohy

Struktura aplikace



Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání diplomové práce

Autor:	**Bc. Vladislav Braha**
Studium:	I1700005
Studijní program:	N1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název diplomové práce:	**Vizuální podpora teorie grafů**
Název diplomové práce AJ:	Visual Support for Graph Theory

Cíl, metody, literatura, předpoklady:

- Rešerše v oblasti využití vizualizace/gamifikace ve vzdělávání - Teoretické zpracování možností a významu vizualizace/gamifikace ve výuce - Nastudování potřebné části z teorie grafů a grafových algoritmů - Výběr technologie pro tvorbu ICT nástroje (vizualizace/herního prostředku) - Vytvoření ICT nástroje - Popis ICT nástroje a jeho ovládání - Testování herního prostředku

MILKOVÁ, Eva: Teorie grafů a grafové algoritmy, 2013. DEMEL, Jiří: Grafy a jejich aplikace, 2002. TAMASSIA, Roberto: Handbook of Graph Drawing and Visualization, 2014.

Garantující pracoviště:	Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu
Vedoucí práce:	RNDr. Andrea Ševčíková
Oponent:	prof. RNDr. PhDr. Antonín Slabý, CSc.
Datum zadání závěrečné práce:	14.1.2015