Mendelova univerzita v Brně
Provozně ekonomická fakulta

Mobilní aplikace pro podporu učení

Bakalářská práce

Vedoucí práce:
Ing. Jan Turčínek, Ph.D.

Pavel Savrov

Brno, 2017

Vyjádřím vděčnost vedoucímu práce Ing. Janu Turčínkovi, Ph.D. za cenné rady a připomínky na konzultacích. Moje poděkování taky patří všem učitelům za trpělivost a znalosti, které jsem dostal během studia, rodině a Vadimu Lyapeikovovi za podporu.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Mobilní aplikace pro podporu učení** vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 3. ledna 2017 .....

Abstract

Savrov, P. Mobile application for learning support. Bachelor thesis. Brno, 2017.

Bachelor thesis is focused on designing and implementing mobile application for learning support. The target mobile platform is Android OS. With this application, the user is able to test his own knowledge on various topics. The application supports several types of questions and ready for internal use at university.

Abstrakt

Savrov, P. Mobilní aplikace pro podporu učení. Bakalářská práce. Brno, 2017.

Bakalářská práce je zaměřena na navrhování a implementaci mobilní aplikace pro podporu učení. Cílovou platformou vybrán OS Android. Pomocí této aplikace uživatel je schopen provádět testování svých znalostí z různých okruhů. Aplikace podporuje několik typů otázek a je připravena pro interní využití na vysoké škole.

Obsah

1 Úvod a cíl práce

1.1 Úvod

V současné době skoro každý student má chytrý telefon, který dokáže provádět spoustu různých operací - komunikace s lidmi přes internet, poslouchání hudby nebo prohlížení různých videa atd. Telefon už dávno není zařízení, které umí jenom volat a posílat SMS, dnes to je plně funkční počítač v kapse.

Student věnuje většinu svého času studiu ve škole, ale tímto se celá výuka nekončí. Důležitou částí je samostudium, který zabírá taky hodně času. S příchodem mobilní technologií se objevila možnost učení přímo na telefonech, tabletech atd. Všechno co potřebují uživatel je nainstalovat si příslušnou aplikaci a je to hotové. Kvůli tomu, že máme takové zařízení vždycky po ruce, studium může být pohodlnější a zabírat miň času.

Jednou z etap samostudia je testování svých znalosti z osvojených témat. Ideálně každý plášť naučeného materiálu má být otestován, aby student věděl co by měl zopakovat v případě odhalení nějakých znalostních mezer. A díky tomu, že existuje několik typy testu, uživatel si může otestovat z různých pohledů.

Aktualne na trhu jsou hodne dobrych hotovych reseni, ze kterych se lze inspirovat. Nektery z nich jsou uvedeny v jedne z dalsich kapitol. Pro vlastni ucely jsem zduraznil nejdulezitejsi vlastnosti vybrannych aplikaci, ke kterym jsem smeroval behem navrhu aplikace. Samotny navrh ma byt jednoduchym a snadno rozsiritelnym.

1.2 Cíl práce

Hlavnim cilem teto bakalarske prace je vytvorit softwarovy nastroj pro podporu uceni studentu, aby vsechny uzivatele aplikace meli jednoduchou, rychlou a bezbarierovou moznost overovani svych znalosti z urcitych okruhu. Vedlejsi cile jsou nastudovani problimatiky technologií spojených s tvorbou mobilních aplikací a navrh reseni ukolu zadani prace. Navrhovane reseni musi byt rozdelene do dvou casti. Prvni cast se sklada z webove aplikace, ktera ma za ucel spravu otazek. Druha cast je reprezentovana mobilni aplikací, ktera poskytuje samotne testovani znalosti uzivatelu.

2 Hotové řešení

Před začátkem vývoje je vždycky dobře se podívat na už hotové řešení, zjistit co mají dobře a co špatně, co se uživatelům líbí a jaká funkčnost může jim chybět. A na základě tohoto si naplánovat dobrou aplikaci která bude splňovat všechna přání koncových uživatelů. V případě platformy Android nejlepším řešením je se podívat do topu Google Play a najít aplikace které řeší podobný úkol a mají nejvyšší hodnocení.

2.1 AnkiDroid

AnkiDroid je Android klientem pro multiplatformní systém Anki, který pomoha uživatelům se naučit novou informaci. Základem je algoritmus SM-2, který dokáže seřadit všechny studijní otázky podle složitosti pro uživatele: nejtěžší se opakují o mnohem častěji než té, se kterými uživatel neměl problémy. Tato aplikace je dobrá v tom, že:

- Rich média obsah – texty, obrázky, videa, zvuky
- Synchronizace se vzdáleným serverem
- Lze vytvořit jednotlivé studijní otázky anebo celou sadu otázek přímo v aplikace
- Import a export dát
- Statistika využívání aplikace

2.2 Mondly

Je to aplikace pro stejnojmennou webovou stránku, která poskytuje hotové řešení pro studium nových jazyků. Aktuálně mají 33 různých kurzů a více než 600 lekcí. Důležité body pro nás jsou:

- Přívětivý design aplikace
- Během cvičení uživatel má 4 pokusy na opravu, když má více – musí udělat celé cvičení znovu
- Rich Média obsah
- Aplikace dokáže zpracovávat jednoduché věty (až 5 slov)
- V případě chybné odpovědi, uživatel uvidí správné řešení
- Několik typu otázek
- Po každém úkolů uživatel vidí statistiku a dosažené množství bodů
- Top všech uživatelů, což přináší elementy hry

2.3 English grammar test

"English grammar test" je aplikací která, jak lze porozumět z názvu, umožňuje uživatelům snadné otestování svých znalosti anglické gramatiky.

- Jednoduchý a nenáročný na paměť design aplikace
- Uživatel si může zvolit jeden z dvou režimu - smíšené testy nebo podle určitého tématu
- Po ukončení testu, v případě že uživatel má chyby, aplikace nabídne probrat problémové momenty znovu a vysvětlí jaké řešení je vhodné pro určitý příklad

2.4 Nauč se pravopis

Tato aplikace pomoha cizincům se naučit českou gramatiku pomoci krátkodobých testu. Zájmové jsou pro nás následující věci:

- Po spuštění aplikace uživatel se hned dostává do výběru testu
- Uživatel může zvolit délku testu - kolik příkladu bude muset řešit
- Když uživatel udělá chybu hned vyskočí nápověda, která popisuje správné řešení

2.5 Quizlet

Quizlet je vzdělávací multi platforma, která je určena pro studenty a učitele zároveň. Je mezi nejpopulárnějšími aplikacemi pro samostudium (Quantcast, 2016).

- Sety lze vytvářet přímo v aplikace
- Existuje 6 základních studijních aktivit
- Funkce Hvězdička která je určena pro studijní otázky, které užívatel chce prostudovat zvlášť
- Lze vytvořit třídy pro skupinu uživatelů a tímto způsobem šířit sety mezi všemi účastníky
- Přístup k veřejným sadám otázek, které udělali jiné uživatelé
- Class Progress - funkce, která ukazuje největší problémy ve třídě
- Jednoduchý design, které umožňuje se soustředit na procese učení

3 Typy elektronických testů

S rozvojem digitálních médií a alternativních výukových platforem se stalo o mnohem jednodušší provádět kontrolu znalostí studentů. Ale důležité je pochopit že tvorba e-learning materiálu je něco jiného než tvorba efektivních studijních materiálů. V praxi chceme se přidržovat druhé varianty. Jedním z jednoduchých způsobů jak udělat učení víc efektivnější je provádět kvizy během studia. Výhoda tohoto přístupu je v tom, že studenti, v případě že chtějí uspět, musejí vždycky zapracovat nad tím, co bylo minulé, abychom byly přepraveny k testům. Dolů jsou uvedené 6 typu quizů, které se používají nejčastěji (B. Clay, 2001).

3.1 Otázky s výběrem možností

Otázky s výběrem možností jsou mezi nejúčinnějšími způsoby jak otestovat znalosti studentů. Představují několik možných odpovědí na otázku, z nichž je pouze jedna správná a ostatní jsou chytáky, které mají odvést pozornost od skutečné odpovědi. Je důležité chápat, že nestačí jenom naházet takové otázky do testu a očekávat, že výsledkem bude hodnota reálné znalosti studentu. Při takovém přístupu lze spíš zkontrolovat dedukční schopnosti, než reálné znalosti o materiálu předmětu. I když je to skvělé pro studenty, že mají deduktivní logické myšlení, to není přímo souvisí s cílem testování. Na stránkách VS Vanderbilt (C. Brame, 2013) jsou uvedeny nejdůležitější rady pro vytváření efektivních otázek s možností výběrů z více odpovědí. Lze to shrnout do následujících bodů:

- Otázka má být smysluplná sama o sobě a měla by představovat určitý problém
- Otázka by neměla obsahovat irelevantní materiál, který nesouvisí s problémem, který má řešit student
- Otázka by neměla mít negativní tvar (používat negace), jenom v případě, když je to 100 % nutné
- Otázka má být ve tvaru dotazu
- Všechny odpovědi by měli být pravděpodobně
- Kazda varianta má být jasná a výstižná
- Odpovědi mají být vzájemně vylučující
- Odpovědi mají mít podobný obsah, aby student nedokázal najít řešení pomoci výjimky
- Jednotlivá odpověď by neměla obsahovat žádný záchytný bod, který naznačí správnou variantu
- Možností by neměli obsahovat varianty typu ”Všechny odpovědi jsou správné” nebo ”Žádná odpověď není správná”

- Počet variant může být libovolným, pokud všechny alternativy jsou smysluplné

3.2 Ano nebo Ne

Jedním z nejjednodušších způsobů testování jsou otázky, na které lze odpovědět Ano nebo Ne. Jediné pravidlo říká, že otázka nesmí být dvojaká - lze 100 % říct, že je to pravda nebo lež. Příklad:

- Dobře: Je OS Android vypracován ve společnosti Apple? - Ne.
- Špatně: Je OS Android vypracován ve společnosti Google? - Od začátku OS Android byl vynalezen ve společnosti Android Inc. A pak, v roce 2005, společnost Google koupila Android Inc.

3.3 Otevřená otázka

Otevřené otázky jsou velmi užitečné, pokud zkoušející chce dozvědět, co studenti myslí o určitém problému, nebo chce donutit studenty přemýšlet kreativně o problému, najít řešení, které není přesně definované. Vzhledem k tomu, otevřené otázky jsou ze své podstaty otevřené, nicméně, to je důležité, aby formulace byla velmi jasná a přesná, protože nejsou tam žádné odpovědi, ze kterých studenti si mohou vybrat.

3.4 Otazky s vyplňováním

Otázky, které potřebují vyplnit nějaké mezery je velmi užitečné, když se jedna o pochopení faktické informace, která může být reprezentovaná nějakým specifickým slovem, číslem nebo krátkou frázi. Při takovém postupu student nemůže prostě odhadnout odpověď, protože jsou žádným způsobem neomezené. To znamená, že je potřeba vědět přesnou odpověď a není jenom ji poznat z nějaké množiny. Základní pravidla při tvorbě otázek s vyplňováním jsou (iSpring Solutions, 2016):

- Jednoduchost a přehlednost
- Správná odpověď může být jenom jedna
- Používat mezery pro klíčové momenty
- Odpověď má být krátká
- Maximálně 2 mezery pro každou otázku

3.5 Přiřazovací otazky

Tyto otázky jsou podobně otázkám požadujících vyplňování s nápovědami - to dává studentům více šancí uspět při testování. Je to dobrý přístup když chceme dát studentovi zpevnit vše znalosti materiálu, ideálně v průběhu studia. Vypadá taková

otázka jako dlouhá věta nebo několik vět s mezerami a pak na konce jsou uvedené pojmy, které student musí dát do správního místa.

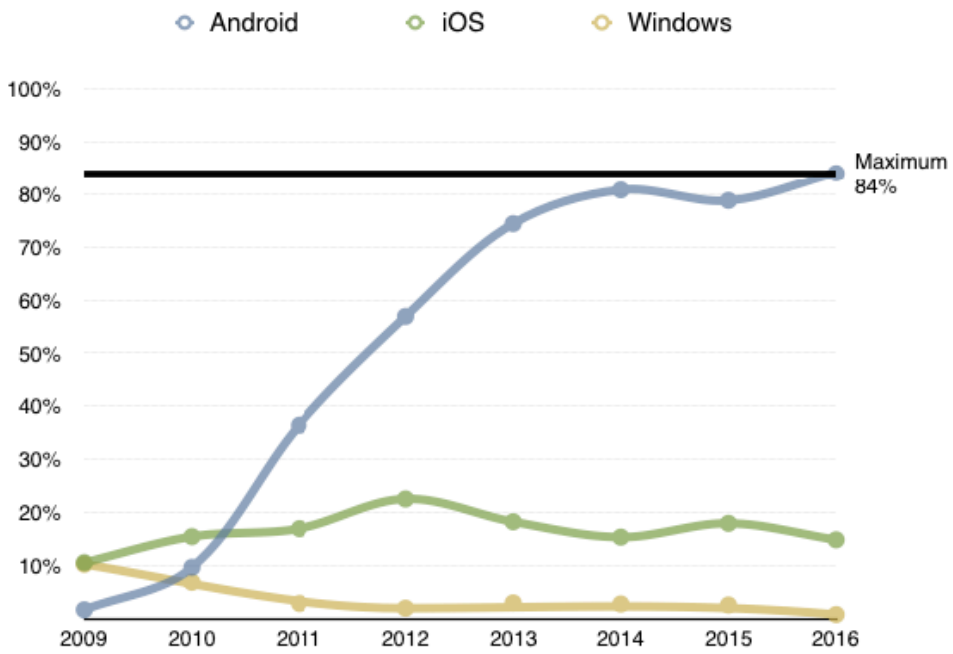
4 Platformy pro mobilní zařízeni

Mobilní platforma je operační systém pro smartfóny, tablety, PDA a jiné mobilní zařízení. Tato platforma spojuje funkčnosti operačních systémů pro počítače s rysy mobilních zařízení. Například, dotyková obrazovka, Wi-Fi, kamera, GPS atd. Historie mobilních platforem se začíná z Palm OS v roce 1996. A od této doby neustále probíhá vývoj operačních systémů, některé z nich se po čase přestali vyrábět, jiné se úspěšně rozvíjejí a mají milióny uživatelů.

Dnes na trhu mobilních OS existuje 3 dominantní platformy: Android OS, Apple iOS a Windows Phone. V této kapitole porovnávám OS podle rozšířenosti ve světě, popisuju každý z výše uvedených systémů, jejich výhody a nevýhody a na konci kapitoly zvolím OS pro budoucí aplikaci.

4.1 Popularita operačních systémů

Obrázek 1 zobrazuje globální podíl OS systému na trhu od roku 2009 do roku 2016. Lze vidět, že v prvním čtvrtletí roku 2016 84 % všech prodaných smartphónů koncovým uživatelům byly telefony s operačním systémem Android. A furt to číslo roste. Na druhém místě je iOS s 14.8 %. Poslední místo patří Microsoftu a to je méně než 1 % (Statista Inc., 2016, Smartphone operating systems: global market share 2009-2015). Ostatní platformy nemá smysl probírat, protože jejich už skoro nikdo nevyužívá.



Obrázek 1: Podil OS systemu na trhu

4.2 Apple iOS

iOS je mobilní operační systém, který běží na mobilních zařízeních Apple, tj iPhone a iPad. To je hlavní software, který umožňuje komunikace s Apple zařízením. Tato platforma používá jádro XNU, které vyvinula společnost Apple. Zajímavým je to, že jádro iOS je velmi podobné jádru macOS. iOS byl představen 9. ledna 2007, spolu s mobilním telefonem iPhone první verze (R. Williams, 2015). Aktuální verze platformy je 10 od 13. září 2016. iOS umožňuje spouštět aplikace, které lze stáhnout z App Store. Pro přístup k Apple Store se vývojáře mají zaregistrovat v Apple Developer Program, který stojí 99 USD ročně.

Pro vytváření iOS aplikaci se používá jazyk Swift. Je silný a intuitivní programovací jazyk pro Mac OS, iOS, watchOS a tvOS. Swift převzal základní jazykové představy od Objective-C, Rust, Haskell, Ruby, Python, C#, CLU atd. (C. Lattner, 2014).

Výhody:

- Vyhrazená uživatelská základna
- Lepší výsledky e-commerce
- Statistické kód je kratší
- Omezený počet typu zařízení
- Lepší nástroje pro vývojáře

Nevýhody:

- Malý podíl na trhu
- Dlouhý proces publikace aplikace v Apple Store
- Cena ročního poplatku za možnost vydávat aplikace v AppStore
- Nutnost mít počítač od Apple pro práci s XCode

4.3 Google Android

Android – operační systém pro smartphony, tablety, e-knihy, digitální přehrávače, hodinky, herní konzole, netbooky, smartbooky, Google brýle, televize a jiné zařízení. V budoucnu je zaplánována podpora vozidel a robotů pro domácnost. Je založen na linuxovém jádře a vlastní implementace Java Virtual Machine od společnosti Google. Původně tento produkt byl vyvinut v Android, Inc., která pak byla koupena Google, Inc. Následně Google inicioval vznik aliance Open Handset Alliance (OHA), který je nyní podporuje další rozvoj platformy (TutorialsPoint, 2016). Hlavním zdrojem softwarů pro Android zařízení je Google Play, který už má kolem 2400000 aplikací (Statista Inc., 2016, Google Play: number of available apps 2009-2016). Pro publikace

vlastních aplikací každý vývojář musí zaplatit jednorázový registrační poplatek 20 USD.

Android umožňuje vytváření aplikací založených na jazyce Java. Pro vývojáře existuje Android Software Development Kit (SDK), které umožňuje přístup k nástrojům a API platformy. Aktuálním nejpopulárnějším prostředím je Android Studio od společnosti JetBrains.

Výhody:

- Velká komunita
- Největší podíl na trhu
- Širší demografie
- Vyšší tržby z reklam
- Menší požadavky na rozvoj
- Snadná publikace na Google Play

Nevýhody:

- Rozmanitost zařízení
- Více potenciálních chyb
- Delší vývoj

4.4 Windows Phone

Mobilní operační systém pro chytré telefony a mobilní zařízení, které slouží jako nástupce původního Windows Mobile. Platforma je známá svým originálním UI, které se jmenuje "Metro". Stejně UI se používá se i pro stolní zařízení od společnosti Microsoft. První prezentace této platformy se konala v říjnu 2010. Uživatele si mohou stahovat aplikace z Windows Store, který má o mnohem menší rozsah než konkurenti (Statista Inc., 2016, Number of apps available in leading app stores 2016). Aby vývojáře mohli publikovat zde vlastní projekty, mají vytvořit vývojářský účet. Ten účet může být vytvořen pro individuální účely anebo firemní, který stojí 19 USD a 99 USD resp.

Pro vývoj aplikace lze používat několik jazyků: C++, Visual C# a Visual Basic. Stejně, jako u platformy Android, vývojáře si mají stáhnout Windows Phone SDK (aktuali verze je 8.0). Jako prostředí pro vývoj se používá Visuál Studio od stejně společnosti (Microsoft, 2016).

Výhody:

- Rostoucí podíl na trhu v několika zemích
- Multiplatformní mezi všemi zařízeními společnosti Microsoft

Nevýhody:

- Globální podíl na trhu je velmy malý
- Malá komunita

4.5 Zvolení OS pro aplikace

Po seznámení s třemi nejpopulárnějšími mobilními platformami je nutné si zvolit platformu, pro kterou bude vytvořena aplikace v rámce této bakalářské práci. Kontrolní body podle kterých se provádí rozhodnutí jsou:

1. Dostupnost mobilních zařízení s tímto systémem pro studenty
2. Publikování aplikace není moc drahé
3. Světlá budoucnost platformy
4. Velká komunita vývojářů

Tabulka 1: Počet bodů pro určitou platformu

Platforma	**1**	**2**	**3**	**4**	**Soucet**
iOS	1	1	2	2	6
Android OS	3	2	3	3	**11**
Windows Phone	2	3	1	1	7

Z tabulky 1 lze spočítat, že nejvíc bodů dostal operační systém Android. Což odpovídá obrázku na začátku této kapitoly (obrazek 1) - Android OS v dnešní době je nejrozsáhlejší mobilní platformou.

5 Vytváření aplikace pro OS Android

Koncepce vývoje Android aplikace není nějak odlišná od ostatních platforem. Ale pro rychlý a pohodlný proces vytváření projektu potřebujeme speciální nástroje, které probereme v první části této kapitoly. Ted' stručně popíšeme celý proces vývoje Android aplikace (Google, 2016, Developer Workflow Basics).

Nejdřív každý vývojář si musí nastavit vývojové prostředí, ve kterém bude pracovat. Když to máme, můžeme si vytvořit první projekt a začít implementovat naši myšlenku. IDE Android Studio poskytuje několik nástrojů, které dělají tento proces jednodušší, např. vizualizace vytváření UI aplikace. Následujícím bodem je kompilace a spouštění projektu. Během této fáze lze vytvořit tzv. DEBUGGABLE APK PACKAGE, který lze nainstalovat do emulátoru nebo reálného Android zařízení. Nedílná součást vývoje je odladění kódu. Ideálně, každá třída, která implementuje určitou obchodní logiku má mít testy, které zjišťuje, že kód se chová podle očekávání vývojářů. Když je kód napsán a všechny testy jsou úspěšně složené, můžeme zveřejnit naší aplikace pro celý svět.

5.1 Instalace a nastavení prostředí

Pro vývoj používáme Android Studio. Je to oficiální IDE pro Android aplikace, které je založené na známém mezi Jáva vývojáři IntelliJ IDEA. Mezi hlavními rysy Android Studio patří:

- Pro automatické sestavení programu se používá Grádle
- Podpora funkce Instant Run, která spouští opravovaný kód bez sestavení celého projektu znovu
- Testovací nástroje, které umožňují jednoduše najít místa tzv. "memory leaks" atd.
- Podpora C++ a NDK

Kompletní popis všech rysů lze najít na oficiálních stránkách IDE (Google, 2016).

Systémové požadavky

Android Studio je multiplatformní vývojové prostředí. Je k dispozici ke stažení pro Windows, Mac OS X a Linux (Google, 2016).

- Microsoft Windows 7/8/10 (32- nebo 64-bit)
- Mac OS X 10.8.5+
- GNOME nebo KDE desktop

Splnění požadavků zajišťuje, že Android SDK může být úspěšně nainstalován na pracovní počítač. Nesmíme taky zapomenout na instalace JDK, které může být staženo z oficiálních stránek.

Instalace a nastavení Android Studio

Instalace Android Studio je velmi jednoduchá. Máme jenom si stáhnout balíček z oficiální stránky, nainstalovat a už od začátku máme Android SDK a Android Virtual Device (AVD). Po spouštění SDK Manager lze stáhnout několik důležitých komponent.

SDK Tools - obsahují kompletní sadu vývojových a ladicích nástrojů pro Android SDK.

Platform tools - podpora různých verzí platformy Android. Každá aktualizace těchto nástrojů je zpětně kompatibilní se staršími verzemi.

SDK Platform - komponenta, která se používá při kompilace pro určitou verze platformy. Skládá se z plně kompatibilní Android knihovny a obrazů systému, který můžeme spustit pomoci emulátoru. Navíc jsou tam ukázky kódu.

Připojení reálné a virtuální zařízeni

V průběhu vývoje je nutné stále testovat aplikace a je pro to vhodné mít nějaké zařízení na kterém ji lze spustit. Máme pro to dvě varianty.

1. Reálné zařízení
2. Android Virtual Device (AVD)

Pro reálné zařízení nejdřív máme zapnout funkci "Developers options" (Google, 2016, Run Apps on a Hardware Device). Jakmile to uděláme, můžeme provádět odladění přes USB, rychle dostávat hlášení o chybách, čerpat informace o využívání CPU, GPU, hardwarových vrstev atd.

Android emulátor (Google, 2016, Run Apps on the Android Emulator) simuluje zařízení a zobrazuje jej na vývojovém počítači. To umožňuje modelování, vývoj a testování aplikací pro Android bez použití hardwarového zařízení. Emulátor podporuje smartfony, tablety, Android Wear a Android TV zařízení. Pro spuštění AVD máme použít AVD Manager, kde lze nastavit libovolnou konfiguraci. Je to užitečné když vývojář chce otestovat aplikace na více různých zařízení.

Ale AVD není jediným Android emulátorem na trhu. Velmi populárním je Genymotion (Genymobile, 2016). Hlavní jeho výhodou je rychlost. Kazdy vývojář musí se počítat s tím, že neexistuje nic lepšího pro testování než reálné zařízení.

5.2 Komponenty aplikace

Android aplikace je založena na jednotlivých částech, které lze vyvolat nezávisle na sobě. Například, nějaká služba (Service) provádí svou činnost na pozadí nezávisle na ostatních komponentách. Existují 4 hlavní a 2 doplňkové komponenty:

1. **Activity** – umožňuje interakci uživatele s UI aplikací
2. **Service** – stará se o zpracování dát na pozadí aplikace
3. **Content Provider** – zpracuje data a ovládají databáze
4. **Broadcast Receivers** – řeší komunikaci mezi Android OS a aplikacemi
5. **Intent** – umožňuje komunikaci mezi aplikacemi
6. **Procesy a toky** – starají se o časově náročných operacích

Dále rozebereme každou komponentů více méně podrobnější a popíšeme příklady jich využití (Google, 2016, App Components).

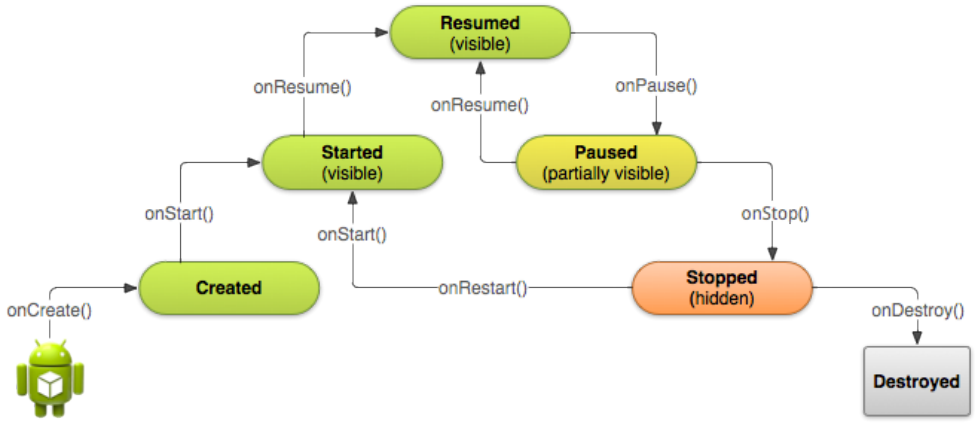
Activity

Aktivita - je součástí aplikace, která stará se o zobrazení UI a komunikace s ním, například, zadání telefonního čísla nebo posílání e-mail zprávy. Pro každou operaci je přiřazeno okno, které musí vykreslit vhodné pro tuto situace UI. Obvykle se aplikace skládá z několika operací, které jsou navzájem slabé propojené. Jedna z těchto operací má být hlavní a zobrazuje se uživatelu při spuštění aplikace. Zároveň, každá operace může nastartovat další aktivitu pro vykonání nějaké činnosti. Když se nová operace nastartuje, stará musí být pozastavena, ale se ukládá do tzv. stack – fronta, která je založena na principu LIFO. Tím pádem, když uživatel stiskne tlačítko *Zpet*, běžná aktivita bude zamítnuta a předchozí se obnoví (Android Developers, 2016, Activities).

Pro vytváření nové aktivity vývojář musí vytvořit novou třídu, která je potomkem třídy `Activity` a implementovat dvě metody které využívají systém pro zjištění, zda běžná aktivita je aktivní nebo ne (Android Developers, 2016, The Activity Lifecycle).

1. **onCreate()** - tato metoda je povinná protože ji zavolá systém při spuštění operaci. Tady mají být inicializovány všechny základny prvky aktivity, důležitým je zavolat metodu `setContentView()` z třídy `Activity` pro nastavení UI.
2. **onPause()** - tato metoda není povinna a pouze informuje systém, že uživatel odešel z aktivity, v tento okamžik vývojář má implementovat logiku, která uloží všechna data, abychom nebyli ztracené.

Životní cyklus je dobře vidět na obrázku dolů (obrazek 2). Začíná se metodou `onCreate()` a konči se v době zavolání `onDestroy()`, ale uživatel může ten cyklus jenom pozorovat od `onStart()` a do `onStop()`.



Obrázek 2: Životní cyklus Android aplikace (Android Developers, 2016, The Activity Lifecycle)

Service

Služba je komponentou, která běží na pozadí, provádí dlouhotrvající operace a nemá žádné UI. Hezkým příkladem je stahování souboru z internetu přes prohlížeč. Dokonce může začít využívat nějakou další aplikace, ale stahování souboru se bude pokračovat. Kazda služba může mít 3 formy (Android Developers, 2016, Services):

1. **Nastartovanou (Started)** – služba je spuštěna příkazem `startService()` a může trvat nekonečně dlouho, když činnost je dokončena, má být pozastavena soběstačně.
2. **Zavázanou na nějaké další komponentě (Bound)** – používá příkaz `bindService()`, odlišenost od `started sluzby` je v tom, že komponenta, která službu spustila, dostává výsledky ji činnosti. Navíc, `bound sluzba` existuje jen do té doby, pokud má alespoň jednu komponentů která ji využívá.
3. **Naplánovaná (Scheduled)** – pro spuštění má být být zavolána nějakým API, například `JobScheduler`. Systém pak sám plánuje provedení služeb ve vhodných okamžicích.

Pro vytváření nové služby programátor má vytvořit třídu, která zdědí základní třídu `Service` a implementovat tzv. *callbacks*. Nejdůležitějšími jsou:

- **onStartCommand()** – volá se, když nějaká komponenta používá příkaz `startService()`.
- **onBind()** – komponent chce vytvořit vazbu na službu pomoci `bindService()`.
- **onCreate()** – volá se pro nastavení služby při prvním volání před `onStartCommand()` nebo `onBind()`.
- **onDestroy()** – služba se už nepoužívá a má být zničena.

Každá služba, stejně jako aktivita, musí být zmíněna v AndroidManifest.xml, ale tento soubor probereme v jedné z dalších kapitol.

Content provider

Content provider - je obal, který obsahuje data ve formě tabulky. Když aplikace používá interní databázový systém, například SQLite, jen tato aplikace má přístup k datům. Ale existují situace, když potřebujeme abychom data byli sdílené pro více aplikace. Jednoduchý příklad - kontakty v telefonním seznamu. Jsou také uložené do nějaké databázi, ale chceme zobrazovat tyto údaje v naší aplikaci taky. Přesně pro tento účel byl vytvořen mechanismus, který umožňuje sdílení dát pro všechny zájemce (M. Nakamura, 2014, Content Providers). Content providery jsou realizované jako podtřídy `ContentProvider` a musejí implementovat standardní sadu rozhraní API:

- **onCreate()** – inicializace poskytovatelů dát
- **query()** – načítání dát
- **insert()** – vložení dát
- **update()** – aktualizace už existujících dát
- **delete()** – smazání dát
- **getType()** – načítání MIME typu dát

Broadcast Receiver

Broadcast Receiver - je komponenta pro obdržení externích událostí a reakce na nich. Inicializovat takové události mohou jiné aplikace a služby. Třída BroadcastReceiver je základní pro třídu, ve které mají probíhat získávání a zpracování zpráv, které jsou posílání pomoci metody `sendBroadcast()`. Takový přijímač lze zaregistrovat dynamický, pomoci metody Context.registerReceiver, nebo statický, vytvořiv element v manifestu aplikací (M. Nakamura, 2014, Broadcast Receivers). Existují dva typy zpráv, které přijímač může obdržet:

1. Normal broadcasts
2. Ordered broadcasts

První typ se posílá pomoci metody `Context.sendBroadcast()` a je úplně asynchronní. Všechny příjemci zpráv provádějí svou činnost v libovolném pořadí. Je to rychlejší, ale zároveň znamená to, že nelze žádným způsobem využít výsledek operace. Co se týká pořadových zpráv, té se posílají metodou `Context.sendOrderedBroadcast()` jenom jednomu příjemci najednou. Z důvodů, že každá zpráva má přesně definované pořadí, může být kdykoli zamítnuta, abychom ostatní příjemci ji neobdrželi (Android Developers, 2016, Sending broadcasts). Existují ještě další komponenty, které

se používají při konstrukci výše uvedených subjektů, jejich logiky, a propojení mezi nimi:

- Fragmenty - představuji jenom část uživatelského rozhraní aktivity
- Pohledy - prvky UI, které jsou nakreslené na obrazovce, včetně tlačítek, obrazů atd.
- Vrstvy - soubory, které definují skupiny pohledu, jejich vztahy a atributy
- Intenty - asynchronní zprávy, které vážou ostatní komponenty spolu
- Zdroje - vnější prvky, jako jsou řetězce, konstanty atd.
- Manifest - konfigurační soubor pro celou aplikaci

5.3 Zdroje aplikace

Zdroje – jedna z hlavních komponent, se kterými pracuje velmi často každý vývojář. V Android všechny objekty, jako jsou obrazy, řetězci, barvy, animace, styly a dál je pravidlem uchovávat mimo zdrojového kódu. Platforma podporuje ukládání takových dát do externích souborů. Takový přístup ke zdrojům je jednodušší na údržbu, aktualizace, upravování.

Ukládaní zdrojů

Kazdy vývojář by měl oddělovat zdroje (obrázky, textové řetězce atd) od kódu pro samostatné zpracování. Navíc, vzhledem na obrovský počet android zařízení, aplikace musí podporovat různé typu konfigurace (velikost obrazovky, jazyk zařízení atd). Po exportu zdrojů lze k nim přistupovat pomoci identifikátorů automatické definovaných v souboru `R.java` (Android Developers, 2016, Providing Resources).

Pro každý typ zdrojů je určena speciální složka, je jich celkem 10 (Android Developers, 2016, Resource Types):

1. `animator/` – změna geometrických vlastností viditelného objektů (otočení, změna rozměru)
2. `anim/` – změna vizuálních vlastností viditelného objektu (průhlednost, barva)
3. `color/` – identifikátor barvy, který je reprezentován HEX kódem
4. `drawable/` – rastrové a vektorové zdroje různých formátů
5. `mipmap/` – spustitelné ikonky aplikace různých rozměrů pro různé konfigurace zařízení
6. `layout/` – nákres uživatelského rozhraní
7. `menu/` – soubory, které definují různé měnu uvnitř aplikace

8. `raw/` – libovolné soubory v výchozí formě (zvuky)
9. `values/` – jednoduché hodnoty (čísla, textové řetězce atd.)
10. `xml/` – ostatní XML soubory, které lze zpracovat pomoci `Resources.getXML()`

A je to kompletní dokončena struktura, ale co když vývojář chce podporovat změnu orientace nebo různou hustotu pixelů? Pro tyto účely se používají tzv. alternativní suroviny - zdroje, které ve svém názvu mimo identifikátory zdrojů obsahují identifikátor konfigurace. Například, textové zdroje pro podporu více jazykových verzí aplikace mají být uložené tímto způsobem:

```
../res/
   /values/
      strings.xml
   /values-cs/
      strings.xml
   /values-ru/
      strings.xml
```

Funguje to tak, že v případě že Android zařízení má nastaveny výchozí jazyk jako češtinu, pak aplikace zkusí najít surovinu s požadovaným identifikátorem v `../values-cs/` a když to nenajde tam, pokusí načíst hodnotu z výchozí složky `../values/`. Při neúspěchu vyhodí chybu, že surovina neexistuje.

Přístup ke zdrojům

Po přidání nějaké suroviny do Android projektu, lze ji začít využívat. Pro to je potřeba vytvořit odkaz na identifikátor zdrojů, který se vygeneruje do souboru `R.java`. Celkem existují jenom dva způsoby přístupu ke konkrétní surovině. Přístup přes **Java kód**. Abychom mohli používat nějakou surovinu v kódu, máme poslat identifikátor suroviny do určité metody jako parametr (Android Developers, 2016, Accessing Resources). Například, pomoci metody `setImageResource()` lze nastavit aby `ImageView` zobrazoval obrázek `res/drawable/myimage.png`:

```
ImageView imageView = (ImageView) findViewById(R.id.myimageview);
imageView.setImageResource(R.drawable.myimage);
```

Syntaxe odkazu je velmi jednoduchá a skládá se z 3 části - `<nazev_balicku>.R.<typ_zdroje>.<nazev_zdroje>` kde `<nazev_balicku>` je balíčkem, ve kterém se nachází surovina, `<typ_zdroje>` je typem (textový řetězec, barva atd.) a `<nazev_zdroje>` je názvem souboru zdroje bez přípony nebo hodnotou atributu `android:name` v XML elementu. Druhým způsobem je přístup přes **XML**. Lze definovat hodnoty některých XML elementů pomoci už existujících surovin. Například můžeme vytvořit objekt Button a přidat k němu nějaký text:

```
<Button
```

```
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/someText" />
```

Syntaxe tady je zcela stejná, ale na začátku odkazu máme přidat zavináč.

Typy zdrojů

Každé nové vydání platformy podporuje více a více typů zdrojů. Aktuálně je jich 14 (Google, 2016, Resource Types):

1. Animation Resources – animace, kterou můžeme používat pro různé komponenty typu `ViewGroup`
2. Color State List Resource – barva, která se mění na základě stavu určitého `View`
3. Drawable Resources – rastrová nebo vektorová grafika
4. Layout Resource – vrstvy, které se používají pro realizace UI aplikace
5. Menu Resource – zdroj, který představuje obsah určitého menu
6. String Resources – textové řetězce nebo pole textových řetězců, podpora formátování
7. Style Resource – zdroj, který definuje styl UI prvků
8. Bool – zdroj, který obsahuje logickou hodnotu
9. Color – zdroj, který reprezentuje barvu v HEX formátu
10. Dimension – zdroj, který se používá pro dimenze (s měrnou jednotkou)
11. ID – zdroj, který poskytuje unikátní identifikátor pro ostatní zdroje a komponenty
12. Integer – zdroj, který obsahuje celočíselnou hodnotu
13. Integer Array – pole celočíselných hodnot
14. Typed Array – pole ostatních typu zdrojů

Ted' se podíváme do těch nejdůležitějších, bez kterých se nedá vytvořit už skoro žádnou aplikaci. Prvním v seznamu je tzv. **Layout Resource** (Android Developers, 2016, Layout Resource). Tyto zdroje jsou zodpovědné za zevnějšek aplikace. Tyto prostředky jsou uvedeny ve formátu XML. Každý zdroj popisující strukturu vrstvy je uložen v samostatném souboru v adresáře `res/layout`. Název souboru bez přípony slouží jako identifikátor zdrojů. Pro nastavení implicitního pohledu máme použít statickou metodu `setContentView(R.layout.main)`, kde `R.layout.main` je unikátní identifikátor, který odpovídá názvu souboru obsahujícího strukturu náhledu. Obsah souboru `main.xml` může být:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

V tomto případě kořenem je element `LinearLayout` a obsahuje dceřiný element `TextView`. Pro každý náhled je potřeba vytvořit samostatný soubor. Když vývojář chce mít dva různých pohledu, např. `res/layout/screen_A.xml` a `res/layout/screen_B.xml`. Každý soubor v adresáře `res/layout` generuje unikátní identifikátor na základě jeho názvu. Tím pádem do souboru `R.java` mají být automatické zapsané dvě konstanty:

```
public static final int file1=0x7f030000;
public static final int file2=0x7f030001;
```

Na element typu `TextView` lze odkazovat taky pomoci ID, které se vygeneruje do souboru `R.java`. Například:

```
TextView tvText = (TextView) findViewByld(R.id.tvText);
tvText.setText("Savrov");
```

Voláním metody `findViewByld()` s parametrem `R.id.tvText`, což je identifikátor elementu typu `TextView`, svázali jsme objekt tvText s výše uvedeným elementem. Ted' místo řetězce "Pavel" se vypíše řetězec "Savrov". Dalším zdrojem, bez kterého se nedá vytvořit naprosto žádnou aplikace s UI je tzv. **Color** (Android Developers, 2016, Color State List Resource). Tento zdroj se taky používá XML pro nastavení určitě barvy v HEX formátu pro určitý identifikátor. Podporuje se několik barevných formátů:

- RGB
- RRGGBB
- ARGB
- AARRGGBB
- definované systémem barvy (red, blue, green atd.)

Barvy se mají být uložené do samostatného souboru s názvem `colors.xml` (adresář `res/values`). Tento soubor může vypadat takto:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="red">#f00</color>
    <color name="yellow">#FFFF00</color>
    <color name="green">#FF00FF00</color>
</resources>
```

Pro přístup k příslušné barvě máme použít identifikátor barvy ve tvaru `R.color.red`, například:

```
LinearLayout linearLayout = (LinearLayout)findViewById(R.id.linearLayout);
// provazovani objektu typu LinearLayout s prislusnym xml elementem
linearLayout.setBackgroundResource(R.color.yellow);
// nastaveni barvy pozadi
```

Ale toto není jediná možnost. Pro statické barevné nastavení lze použít xml soubor, který popisuje strukturu pohledu:

```
<LinearLayout
    android:id="+id/linearLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/red">
    // dalsi obsah tyto vrstvy
</LinearLayout>
```

Když chceme, aby naše aplikace obsahovala nějaké texty je vhodné použít **String Resources** (Android Developers, 2016, String Resources). Tyto zdroje přenášejí možnost snadné úpravy a překládání textových řetězců do různých jazyků. Pro vytváření těchto zdrojů obvykle se používá soubor `strings.xml` (adresář res/values). Text lze uložit pomoci třech formátu:

1. String - jeden textový řetězec
2. String Array - několik textových řetězců
3. Quantity Strings (Plurals) - jeden řetězec, který podporuje skloňování

Informace o používání těchto formátů lze najít na oficiálních stránkách (Google, 2016, String Resources), ale teď probereme nejvíc používány příklad - jednoduchý textový řetězec. Soubor textových řádku má následující strukturu:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="developer_name">Pavel A. Savrov</string>
```

```
    <string name="hello">Hello</string>
</resources>
```

Stejně, jako v `Layout Resources`, do souboru `R.java` se zapíšou dva řádky unikátních identifikátoru:

```
public static final int developer_name = 0x7f040000;
public static final int hello = 0x7f040001;
```

Používání textových řetězců je taky velmi jednoduché a dostupné dvěma způsoby. Buď to přes xml soubor:

```
<TextView
    android:layout_width="fill_ parent"
    android:text="@string/hello" />
```

Nebo přes Java kód:

```
TextView tvHello = findViewById(R.id.tvHello);
String helloString = getString (R.string.hello);
tvHello.setText(hello);
```

Tyto tři zdroje se používají nejčastěji a skládají zhruba 80 % obsahu pohledu jednotlivého aktivity nebo fragmentů. Informace o ostatních zdrojích a jich způsobech použití jsou dostupné online na oficiálních stránkách (Google, 2016, App Resources).

5.4 Manifest aplikace

Soubor manifestů `AndroidManifest.xml` poskytuje základní informace Android zařízení o aplikace. Každá aplikace musí mít svůj manifest, který obsahuje další informace:

- Určuje název Jáva-balíčku aplikace, který slouží jako unikátní identifikátor
- Popisuje komponenty aplikace - aktivity, služby, poskytovatelé obsahů (content provider) atd., což umožňuje volání tříd, které řeší každý s těchto komponent
- Obsahuje seznam požadovaných oprávnění k přístupu k chráněné části API a interakci s jinými aplikacemi
- Deklaruje oprávnění, které jsou nutné, abychom aplikace třetích stran mohli navzájem působit s komponentami běžné aplikace
- Vyhlašuje minimální verze API, která je potřebná pro funkčnost aplikace
- Uvádí seznam použitých knihoven

Tento soubor shrnuje v sobě celou strukturu aplikace, jeho funkčnost a konfigurace. Během vývojů máme často tento soubor upravovat. Kořenovým elementem manifes-

tu je `manifest`. Kromě tohoto prvků jsou povinné prvky `application` a `uses-SDK`. Element `application` je klíčovým prvkem manifestu a obsahuje mnoho podřízených elementů, které definují strukturu a provoz aplikace. Pořadí prvků, které jsou na stejné úrovni, svévolné. Všechny hodnoty se nastavují pomocí atributů prvků. Kromě povinných výše uvedených prvků, další se používají podle potřeby. Probereme několik elementů, které budeme využívat během vytváření budoucí aplikace. Jedním z nejdůležitějších je prvek **activity**, který se používá pro registrace aktivity. Když aplikací je několik, každá musí být uvedena v manifestu, v případě že tento požadavek nebude splněn - systém ne bude vědět, že aktivita vůbec existuje a při spuštění aplikace vyhodí chybové hlášení. Pro tuto třídu je zaregistrován filtr volání, který zjišťuje jaká aktivita má být spuštěna při otevření aplikace. Podporuje se 5 atributů:

1. **android:name** – název třídy kde je aktivity implementovány
2. **android:label** – textová značka, která se zobrazuje u uživatelů
3. **android:launcMode** – řízení zásobníkem použitých aktivit
4. **android:parentActivityName** – definuje rodičovskou aktivitu
5. **android:exported** – povolení nebo zákaz používání aktivity externími aplikacemi

Service deklaruje službu jako jednu ze složek aplikace. Všechny služby musejí byt prezentované tágem `service`. Služby, které nebyli oznámený, nebudou detekóvány systémem a nikdy nebudou spuštěné. Tento prvek má mnoho atributů, které určují název, dostupnost, autorizace procesu a tak dále. Podporuje vnořené uzly `intent-filter`.

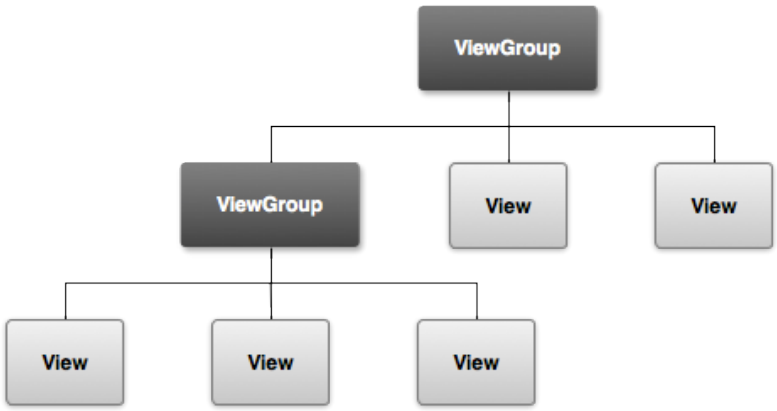
```
<service
   android:name=".LocalService"
   android:icon="@drawable/icon"
   android:label="@string/service_name" >
</service>
```

Element **Reciever** umožňuje registraci přijímače, který posluha na události v systému a, v případě zájmu, dokáže zareagovat i když aplikace není spuštěna.

```
<receiver android:name="ScheduleReceiver" >
   <intent-filter>
<action android:name="android.intent.action.BOOT_COMPLETED" />
   </intent-filter>
</receiver>
```

5.5 Uživatelské rozhraní

V této kapitole budou popsány základní elementy, ze kterých se buduje uživatelské rozhraní aplikace. Kazda taková komponenta je potomkem jedné ze dvou základních objektů: **View** a **ViewGroup**. Rozdíl mezi nimi je v tom, že View určen pro vykreslení nějakého určitého widgetu a objekty typu ViewGroup pomáhají seřadit použité View a ViewGroup do různých druhů návrhů architektury - lineární, relativní atd. Jinými slovy, uživatelské rozhraní každé komponenty aplikace (např. aktivity) je reprezentované stromem View a ViewGroup objektů (obr. 3). Každý návrh definuje



Obrázek 3: Hierarchie View a ViewGroup v uživatelském rozhraní (Android Developers, 2016, User Interface Layout)

strukturu uživatelského rozhraní a může být definován buď pomoci XML souboru nebo přímo ve zdrojovém kódu. Je to užitečné když vývojář chce nastavit nějaký implicitní pohled a pak měnit ho za běhu aplikace. Výhodou používání samostatného XML souboru pro každý návrh je zjevně - oddělení prezentační vrstvy programu od aplikační vrstvy, kde probíhají různé výpočty a jiné zpracování dát. Tím pádem, popis uživatelského rozhraní je externím zdrojem pro aplikační kód, což pro nás znamená, že můžeme tento návrh měnit bez úpravy zdrojového kódu, tudíž, nemusíme provádět sestavení programu znovu. Díky tomu, v praxi můžeme použít různé návrhy pro různá rozlišení a orientace obrazovky.

LinearLayout je ViewGroup, který zarovná všechny podřízené objekty v jednom směru - vertikálně nebo horizontálně. Směr zajišťuje atribut orientace `android:orientation`:

- android:orientation=”horizontal”
- android:orientation=”vertical”

Všechny podřízené prvky jsou naskládány jeden po druhém tak, aby každý prvek seznamu měl pouze jeden podřízený element v řadě, bez ohledu na to, jak široká je ta řada. Horizontální uspořádání prvků v seznamu znamená, že všechny elementy budou umístěny v jedné linii. LinearLayout má k dispozici zajímavý atribut

`android:layout_weight`, který přiřazuje individuální váhu určitému podřízenému elementu. Tento atribut určuje "důležitost" elementů a umožňuje ho rozšířit v nadřízeném pohledu. Taky můžeme určit atribut `android:weightSum`. Je-li atributu přiřazena hodnota 100, můžete určit hmotnost podřízeného elementů v procentech (Android Developers, 2016, Linear Layout).

Relative layout je `ViewGroup`, která zobrazují podřízené objekty v relativních pozicích, jinými slovy umožňuje určit jak podřízené objekty jsou umístěny navzájem sobě. Pozice každého `View` elementu může být určena vůči jinému elementu podle jeho `id` nebo vůči rodičovskému pohledu, což je `Relative layout` (A. Klimov, 2016, RelativeLayout). Existuje kolem 20 parametrů, pomoci kterých lze nastavit závislost mezi elementy pohledu (Android Developers, 2016, RelativeLayout.LayoutParams). Některé z nich jsou:

- `android:layout_alignParentLeft` – pokud je `true`, levý okraj tohoto pohledu je umístěn stejně jako levý okraj rodiče
- `android:layout_centerVertical` – pokud je `true`, element musí být vertikálně uprostřed svého rodiče
- `android:layout_above` – element je umístěn nad uvedeným elementem
- `android:layout_toRightOf` – levá stěna elementu je vpravo od uvedeného elementu

RecycleView je nový `ViewGroup`, který je následníkem `ListView` (Google, 2016, ListView), což znamená, že je taky určen pro prezentace nějakého seznamu dát. Hlavní výhodou je to, že tento element je založen na více rozšiřitelným frameworku, který podporuje horizontální a vertikální pohledy zároveň. `RecycleView` je velmi užitečný, když obsah se mění za běhu programy na základě akcí uživatele nebo stavu sítě (CodePath, 2016, Using the RecyclerView). Pro práci s tímto elementem potřebujeme:

- RecyclerView.Adapter – zpracování dát a je výkres
- LayoutManager – polohování jednotlivých položek
- ItemAnimator – animace při interakci uživatele se seznamem

`RecyclerView` podporuje novější typ adaptéru. Je skoro stejný jako u `ListView`, ale programátor potřebuji navíc implementovat dvě metody. Jedna je pro provázání dát a pohledu, druhá - pro naplnění pohledu a jeho správce. Když máme adaptér, potřebujeme nastavit `LayoutManager`. Existuje tři možností nastavení:

- `LinearLayoutManager` – vertikální nebo horizontální seznam
- `GridLayoutManager` – seznam ve tvaru mřížky
- `StaggeredGridLayoutManager` – stejně jako u předcházejícího, ale velikost buněk se dynamické mění podle obsahu

`ItemAnimator` se stará o animaci přidání, mazání nebo výběru prvků seznamu.

Vstupní prvky UI

Vstupní prvky UI jsou interaktivní komponenty v uživatelském rozhraní aplikace. Android poskytuje širokou škálu takových komponent (tabulka 2), které lze jednoduše použít. Například, tlačítka, textové pole, přepínače a mnoho dalších (Google, 2016, Input Controls). Každý vstupní prvek UI podporuje specifickou sadu vstupních událostí, tím pádem můžeme zpracovávat různé události. Například, zadáni textu nebo dotyk tlačítka.

Název	Popis	Třídy
Button	Tlačítko které může měnit svůj stav (pressed, clicked)	`Button`
Text field	Pole, do kterého lze zadat text, pomoci `AutoCompleteTextView` lze použit funkci automatického doplnění	`EditText`, `AutoCompleteTextView`
Checkbox	View, který realizuje ON/OFF funkčnost. Je užitečný, když máme nějakou skupinu možností, které jsou navzájem nevylučující	`CheckBox`
Radio button	Je podobný předchozímu, ale jenom jedna možnost ze skupiny může byt vybraná	`RadioGroup`, `RadioButton`
Toggle button	Tlačítko typu ON/OFF se světelným indikátorem	`ToggleButton`
Spinner	Drop-down seznam, který umožňuje uživatelům vybrat jednu možnost ze sady.	`Spinner`
Pickers	Dialogové okno, ve kterém lze vybrat jednu hodnotu pomocí tlačítek nahoru/dolů nebo prostřednictvím swipu.	`DatePicker`, `TimePicker`

Tabulka 2: Typy vstupních prvků UI

5.6 Testování a odladění aplikace

Všechny aplikace běží na zařízeních, které mají omezenou paměť, výkon procesoru, kapacitu akumulátoru atd. Z tohoto důvodů je velmi důležité otestovat chování programu v různých stavech systému. Dobře sestavené testy pomohou dodržovat aplikaci v lepším stavu. Kvůli velkému množství Android zařízení není možné otestovat kód na všech možných konfiguracích, proto v praxi se používá následující pravidlo: Testování musí být úspěšně dokončeno pro zařízení s nejnižší a nejvyšší možnými konfiguracemi. Testování v Android se dělí do 2 skupin (TutorialsPoint, 2016, Android testing):

1. Local unit tests - testování probíhá na JVM, to je nejrychlejší způsob, ale nelze otestovat kód, který používá Android API
2. Instrumented unit tests - testování probíhá na Android zařízení, je pomalejší, ale podporuje Android API

Během testování vývojář se musí soustředit na otestování obchodní logiky aplikace. Dobrým zvykem je mít následující rozdělení testů:

- 70-80 % - unit testy, které zajišťují stabilitu kódu
- 20-30 % - funkcionální testy, které zjišťují je-li celková aplikace je funkční

Unit testy

Formulace *unit test* v platformě Android se používá pro testy, které mohou být spouštěny na lokálním JVM nebo přímo v Android systému. Úkolem je otestovat určitou funkčnost určité komponenty (Vogella, 2016, Developing Android unit and instrumentation tests). Například, testování převedení z Celsia do Fahrenheitové stupnice. Unit test má za úkol zjistit, zda převod je správný nebo ne. Výsledky testu jsou uloženy do souboru `index.html`, který se nachází v adresáře `app/build/reports/tests/debug/`. Napsání unit testu je dost jednoduché a vypadá přibližně takto:

```
@Test
public void testConvertFahrenheitToCelsius() {
    float actual = ConverterUtil.convertCelsiusToFahrenheit(100);
    // čoekáváme hodnotu 212
    float expected = 212;
    // actual==expected?
    assertEquals("Conversion from celsius to fahrenheit failed: ",
        expected, actual, 0.001);
}
```

Přístrojove testy

Android API poskytuje speciální přístupové body pro testování různých Android komponent a životního cyklu aplikace. Tyto body se jmenují *instrumentation API* (Vogella, 2016, Writing Instrumentation tests to run on the Android device). Na rozdíl od unit testu, tyto testy dokážou odchytit události, za které je zodpovědný Android API. Například, stisk tlačítka pro přechod mezi aktivitami. Nebo můžeme napsat test, který je schopen imitovat stisknutí tlačítka, spouštění další aktivity, její uzavření a zase spouštění. Tímto způsobem můžeme zjistit je-li stav aktivity se obnoví správně. Tento druh testu lze spustit jenom na emulátoru nebo na reálném Android zařízení. Tím pádem máme přístup k reálným zdrojům zařízení, které nejde

jednoduše nafejkovat pomocí většiny mock rámců. Jedním z nejpopulárnějších rámců, který to dokáže je Mockito (Mockito, 2016). Příklad jeho využití je následující:

```
@Test
public void writeShouldWriteTwiceToFileSystem() {
    try {
        when(context.openFileOutput(anyString(), anyInt()))
            .thenReturn(fileOutputStream);
        Util.writeConfiguration(context);
        // check if the openFileOutput is called exactly once
        verify(context, times(1)).openFileOutput(anyString(), anyInt());
        // check if the write() method is called at least twice.
        verify(fileOutputStream, atLeast(2)).write(any(byte[].class));

    } catch (Exception e) {
        e.printStackTrace();
        fail();
    }
}
```

Výsledky testu jsou uložené do souboru `index.html` v adresáře `app/build/reports/androidTests/connected/`.

6 Metodika

6.1 Funkční požadavky

Web aplikace

Webová část aplikace má být navrhována jako komfortní systém správy otázek, neměl jsem za úkol řešit samo testování na web aplikaci. Vzhledem k tomu jsem definoval následující funkční požadavky:

- Registrace a přihlášení autorů do systému
- Vytváření, editace, mazání určitých sad otázek autorem
- Přehled vytvořených autorem sad otázek
- Vytváření, editace, mazání jednotlivých otázek autorem
- Přehled statistiky používání balíčku otázek pro autory
- Přehled obecné statistiky pro administrativu aplikace

Android aplikace

Dřív, v kapitole 2 na straně 8, jsem rozepsal zásadní body, které by měli být realizované v mé aplikaci. Pro první verze jsem rozhodl přidat jenom tyto nejdůležitější funkce:

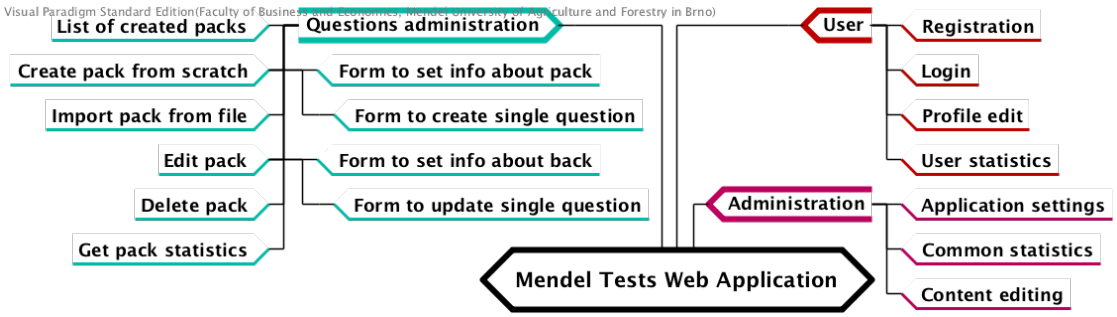
- Načítání dostupných dát z REST serveru v json formátu
- Ukládání dát do lokální databáze, aby uživatel mohl používat aplikace i bez aktivního připojení k sítí
- Provádění testování s podporou několika typů otázek, časovým limitem a zobrazením nápovědy v případě špatné odpovědi
- Přidání libovolné sady otázek do položky "Oblíbené" pro zjednodušený přístup
- Zobrazení statistiky pro jednotlivou sadu otázek
- Synchronizace lokální databáze a externího zdroje dát

6.2 Návrh webové aplikace

Struktura

Před začátkem napsání kódu, je důležité si všimnout co přesně očekáváme od web aplikace. To znamená, že máme si vytvořit nějakou strukturu webové stránky. Pro tyto účely se používá myšlenková mapa (mindmap diagram). Princip je jednoduchý - do centru mapy nakreslíme klíčový objekt, což v našem případě je webová aplikace,

pak od centru provádíme čáry do hlavních témat, které naše aplikace má podporovat. Každá téma se pak rozděluje do drobnějších úkolů.



Obrázek 4: Myšlenková mapa

Podle návrhu hlavním úkolem bude správa otázek - vytváření nových balíčků, editace, mazání atd. Samotné testování má probíhat v mobilních aplikacích. Navíc určitě má být implementovaná jakási správa uživatelů a administrace aplikace.

Databáze

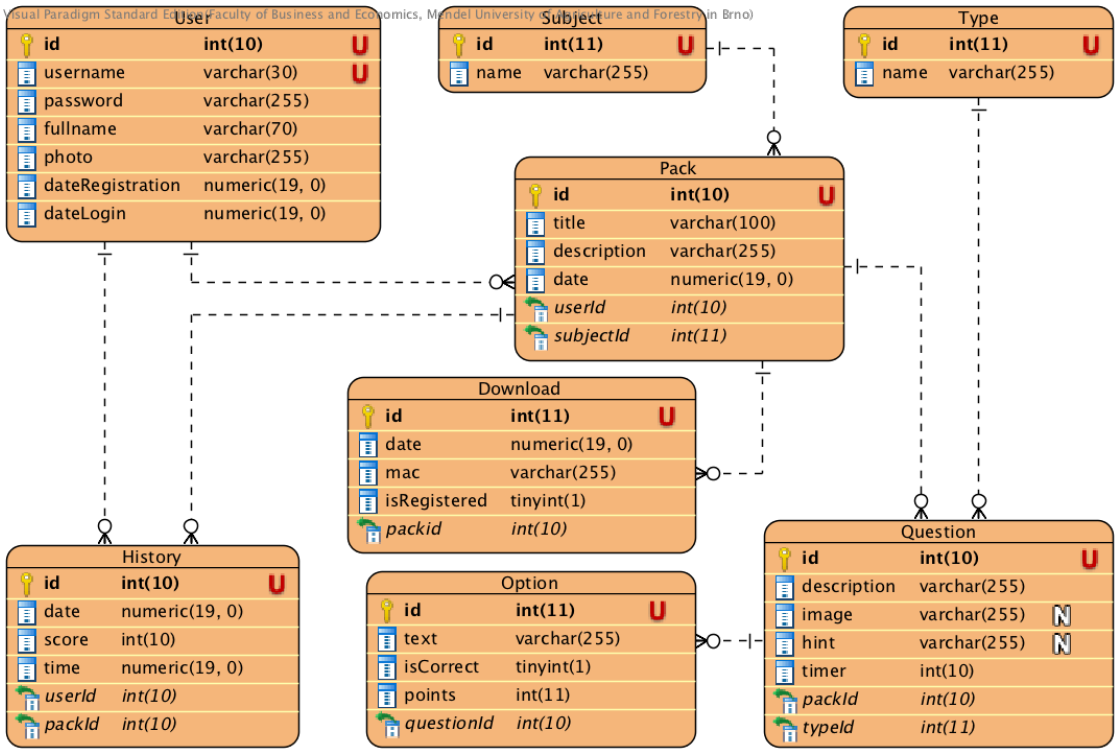
Při návrhu databáze máme si všimnout dvě věci. Zaprvé, databáze nesmí obsahovat duplicitní informace, protože taková data zbytečně zabírají místo a zvyšují pravděpodobnost vzniku chyb a různých kolizí. Zadruhé, databáze musí zajistit zachování správnosti a celistvosti informaci, protože jinak rozhodnutí založené na takových datech bude chybné (Microsoft, 2010).

Proces návrhu databáze se skládá z několika kroků:

1. určení účelu databáze
2. definování informace, kterou chceme uložit
3. rozdělení informace do tabulek
4. vytváření relace mezi tabulkami
5. provádění kontroly návrhu a upravení popřípadě chyb

Po procházení všech 5 etap dostal jsem návrh databáze, který zhruba by měl odpovídat databáze finální verzi webové aplikace (obrazek 5).

Entita `User` je určená k ukládání informace o libovolném uživateli, ať to bude učitel nebo student. Zatím položky `username`, `password`, `fullname` a `photo` uživatel má vyplnit sám, ostatní dvě - datum registrace a datum posledního přihlášeni řeší systém. Každý uživatel má bezbariérový přístup ke všem vytvořeným sadám otázek. Kazdy takový balíček obsahuje informací o názvu, popisu, datu vytváření, kdo ho vytvořil a pro jaký předmět je určen. Kazda otázka může odkazovat jenom na jeden balík, stejné pravidlo platí i pro typ otázky. Navíc do tabulky `Question` se ukládá informace, která popisuje samo zadání, v případě potřeby obrázek, nápovědu a čas



Obrázek 5: Návrh databáze

určeny na řešení zadání. Samozřejmě každá otázka má obsahovat odpověď, abychom zjistili, zda uživatel odpověděl správné či ne. Pro některé typy otázek, například testy, potřebujeme ještě mít nějaké dostupné varianty. Pro tyto účely existuje entita `Option`, která mimo položky popisu varianty ještě popisuje, zda tato varianta je správná a kolik bodů dostane uživatel v případě zvolení této možnosti. Poslední tabulka se jmenují `History` a zahrnuje informace o tom, jak proběhlo určité testování: kdo byl otestován, jaká sada otázek se používala, kdy, kolik času bylo ztraceno a kolik bodů bylo dosázeno. Počítám s tím, že tato informace bude užitečná nejenom pro studenty ale pro autory testů taky. Navíc můžeme sledovat statistiku stahování určitých sad otázek - do tabulky `Download` se ukládá informace o datu stahování, MAC adrese zařízení a stavu uživatele - je přihlášeny nebo host.

Technologie

Existuje hodně server-side jazyků, které umožňují vytvořit backend aplikace. Volba jazyku více méně je závislá na požadavků aplikace, použité databáze a dokonce i preferenci vývojáře. Vědění, co každý jazyk může nabídnout a čím se odlišuje od ostatních je užitečné při rozhodování o tom, jak vybudovat backend aplikace (C. Wodehouse, 2015). V dnešní době seznam dostupných řešení je velký a můžeme si vybrat něco z těchto variant: PHP, Python, Ruby, C#, C++ a Java.

Jazyky z 3 po 6 pozici se používají ve velkých aplikacích se složitou logikou. Což není naším případem. PHP nebo Python je to zcela subjektivní otázka, ale podle statistiky GitHub (C. Zapponi, 2014) více využívaným jazykem z těchto dvou je Python. Navíc Python lépe navržen, má lepší web rámce než PHP a syntaxe je jednodušší (H. Boparai, 2014). Nevýhodou je to, že Python je o něco složitější k osvojení. Svazek Python a, například, Flask rámce je hezkým řešením pro náš úkol.

Uživatelské rozhraní

Dalším bodem navrhování je tvorba GUI aplikace. Grafické rozhraní je velmi důležité protože vytváří první dojem o aplikaci. Způsob navrhování aplikace z pohledu uživatele se jmenuje User Experience Design (UXD) a právě ho budeme využívat pro modelování budoucího rozhraní (S. Horný, 2014). Existuje několik fází, které máme překonat, abychom dostali vhodný výsledek:

1. zjistit cílovou skupinu uživatelů a představit si jak s aplikací budou komunikovat
2. definovat funkce, které GUI má pokrývat
3. definovat interakci mezi jednotlivými funkcemi
4. vytvořit wireframe - skic aplikace
5. grafická úprava vzhledu jednotlivých komponent
6. testování

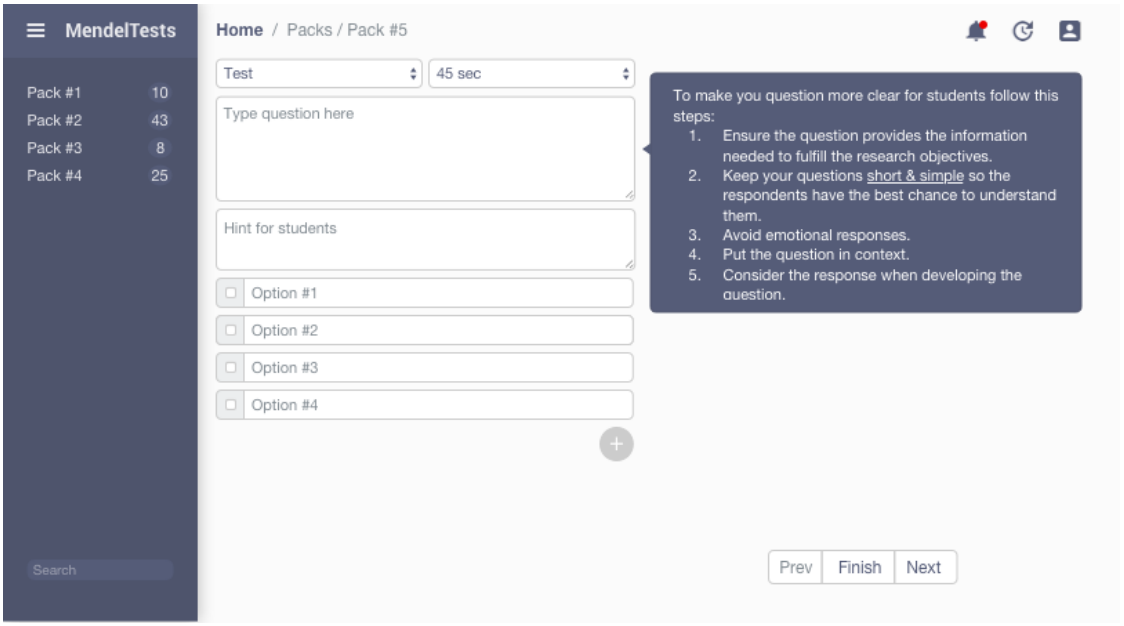
Pro tuto práci jsem rozhodl udělat uživatelské rozhraní pro jednu z nejdůležitějších části aplikace - přidání otázek do balíčků. Tuto část aplikace budou používat jenom autory, to znamená, že UI má byt maximálně informativní a jednoduché na vytváření jednotlivých otázek. Z tohoto důvodu všechna data pro určitou otázku lze zadat v jednom okně, žádné zbytečné přechody.

6.3 Návrh mobilní aplikace

Metodika vývoje

V dnešní době existuje několik metodik vývojů ref.metodiky, které pomáhají vývojářům úspěšně dokončit projekt. Mnou byla vybrána Agilní metodika, protože, v porovnání s ostatními, umožňuje kontrolovat proces vývojů v každém okamžiku. Já, jako vývojář, jsem vždy jistý, že pracuji ve správném směru. Zároveň, zákazník aplikace vidí co už máme řešené a, v případě nějakých nejasnosti, může říct co mám upravit.

Metodika, kterou jsem si zvolil, míní, že vývojář má rozdělit úkol do několika části a postupně je řešit. Pro aplikaci, kterou jsem zmínil dřív, byli naplánované další kroky:



Obrázek 6: Uživatelské rozhraní webové aplikace

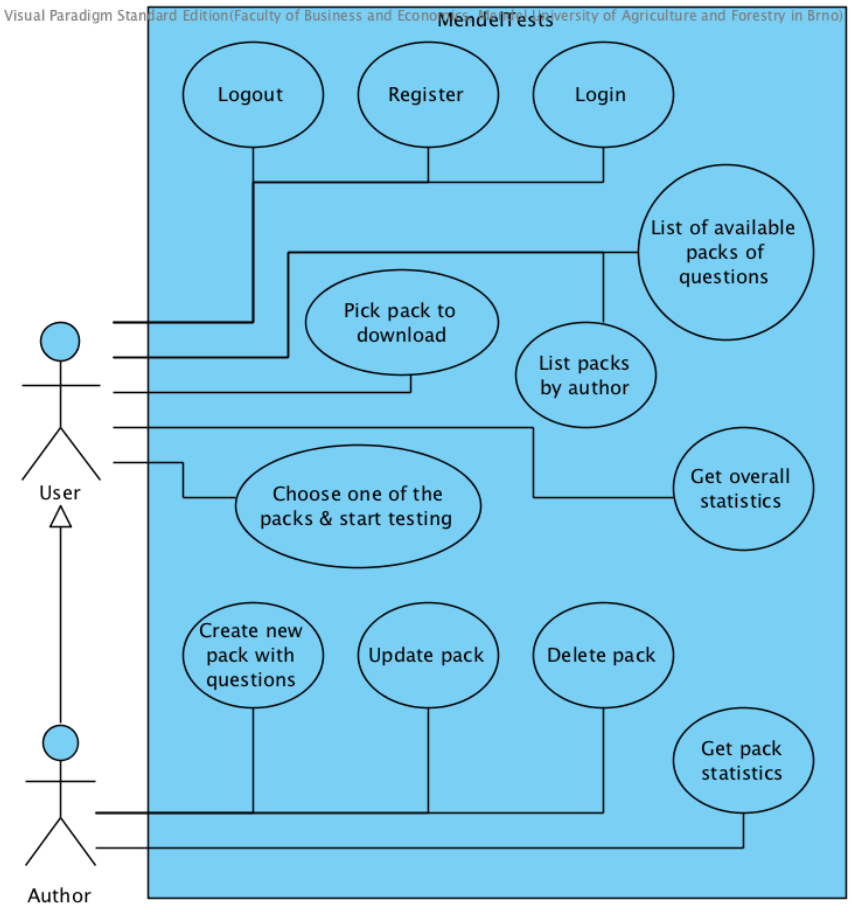
- Připravit server, který se bude starat o uložení dát
- Server má podporovat REST služby, aby aplikace mohla jednoduše přistupovat k datům, které jsou tam uloženy
- Vývoj aplikace má se začít z vytváření komunikačního kanálu klient-server
- Vymyslit a implementovat všechny možné případy užití (UseCase) aplikace
- Vytvořit uživatelské rozhraní, které bude využívat už připravené případy užití
- Otestovat konečnou verzi aplikace a zveřejnit pro cílovou skupinu

Diagramy

Use Case. Diagram případů užití se používá při fázi analýzy projektu pro identifikaci funkčnosti systému, popisuje interakci uživatele nebo externího zařízení se systémem. Tento diagram neukazuje mnoho detailů, jeho úkolem je zdůraznit vztahy mezi aktory, případy užití a systémem (obrázek 7) (P. Rejnková, 2009, Diagram případů užití).

State Machine. Stavový diagram se používá k modelování dynamické povahy systému. Popisuje všechny možné stavy objektů v rámci určitého systému. Nejdůležitějším smyslem stavového diagramu je modelování životního cyklu objektů od jeho vytváření až po ukončení pracovního poměru (P. Rejnková, 2009, Stavový diagram).

Obrazek 8 zobrazuje, jak se mění stav systému v průběhu aktivního testování, jak systém reaguje na různé události, které provádí uživatel.

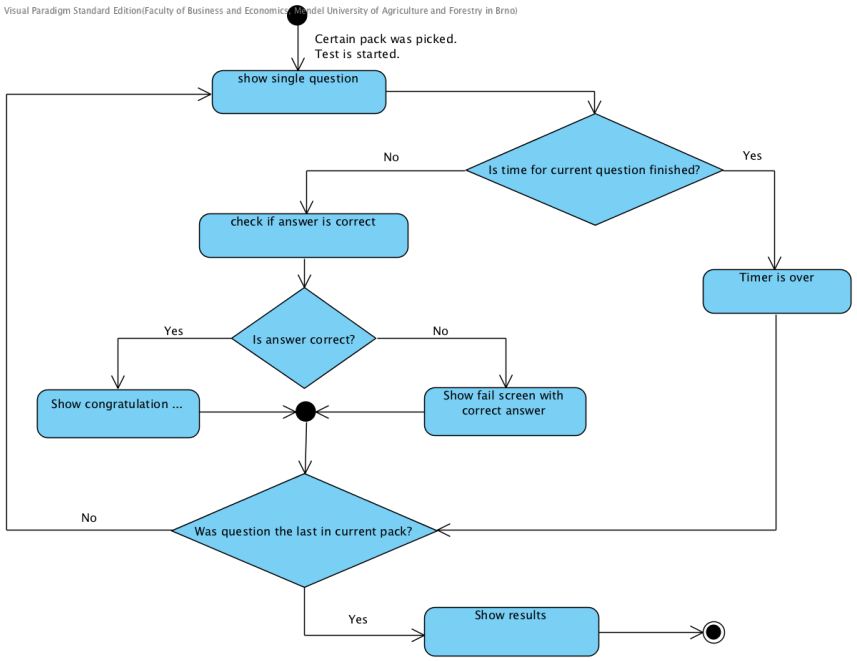


Obrázek 7: Diagram případů užití

Sequence diagram Sekvenční diagram zobrazuje chování a spolupráci jednotlivých objektů pomoci tzv. zpráv. Hlavními prvky sekvenčních diagramů jsou objekty, vertikální čáry, které mají název ”lifeline”. Čára životu je přímkou, která zobrazuje tok času. Pak jsou tam ještě obdélníky, které označují činnost objektů nebo výkon specifické funkce a šipky, které jsou určené pro zobrazení směru přenosu zpráv mezi objekty (P. Rejnková, 2009, Sekvenční diagram). Obrázek 9 vysvětluje, jak spolupracují jednotlivé prvky systému od momentu přihlášení uživatele do systému do ukončení testování. Zprávy jsou reprezentované pomocí čáry a popisují zprávy, které dostávají jedny objekty od jiných.

Návrh databáze

Databáze slouží k uložení dát, se kterými pak může libovolná aplikace provádět CRUD operace. Databáze má být navrhnuta tak, aby dokázala zachytit všechny objekty a operace nad nimi. Například, když chceme smazat nějaký balíček otázek (Pack), systém automatické by měl smazat a všechny otázky (Question), které mají na tento balíček reference. Toto lze dosáhnout pomocí nastavení vlastnosti `"delete`



Obrázek 8: Stavový diagram

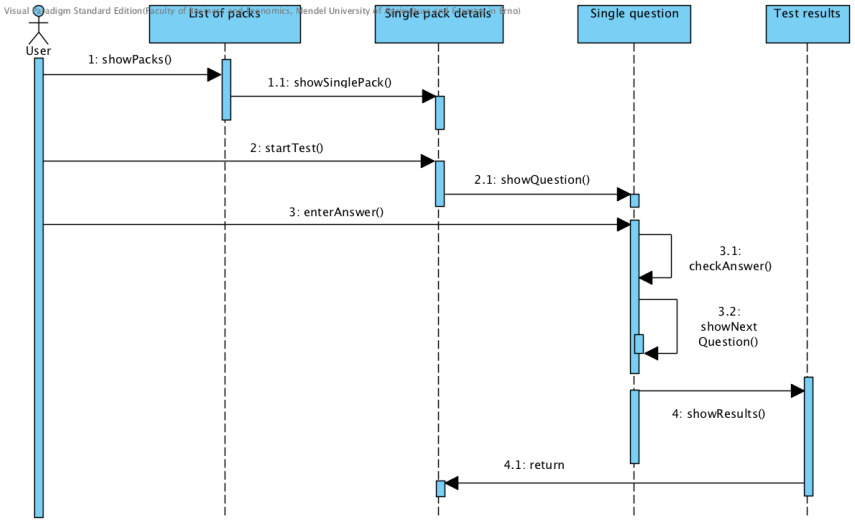
`on cascade"` u cizího klíče. Všechny potřebné entity a vazby mezi nimi jsou zobrazený na obrázku 5.

Technologie

Platforma Android je založena na jazyce Java, ale to není jediné možné řešení na trhu. Lze použít, například, C++ a Android NDK, Python a Kivy rámec nebo Kotlin. Že všech třech alternativ Kotlin - je nejlepší volbou. Je určen speciálně pro vývoj Android aplikací (JetBrains, 2016). Ale žádný jazyk není ideální. V případě Kotlin je to nízká čitelnost kódu pro začátečníky a relativně malá komunita. Konečné rozhodl jsem použít v tomto projektu jazyk Jáva.

V případě nutnosti je dobrým zvykem používání externích knihoven, které řeší určité úkoly místo nás. Kvůli takovému přístupu vývoj probíhá rychleji a vývojář se nemá zabývat problémy, které už byly řešeny více zkušenými vývojáři. Pro tento projekt jsem rozhodl použít několik knihoven mimo standartních od Android, které se používají v každém projektu:

- Glide - rychlý a efektivní open source rámec, který řeší zpracování obrázku: načítání obrázku z internetu, dekódování a cachování do paměti nebo disku.
- Retrofit - Type-safe HTTP klient pro Android, který řeší komunikace typu klient-server
- Butter Knife - rámec, který umožňuje používání speciálních anotací, aby vyloučit boilerplate (standartní) kód



Obrázek 9: Sekvenční diagram

- Android View Animations - knihovna, která poskytuje rozsáhlý seznam animací pro `View` elementy UI

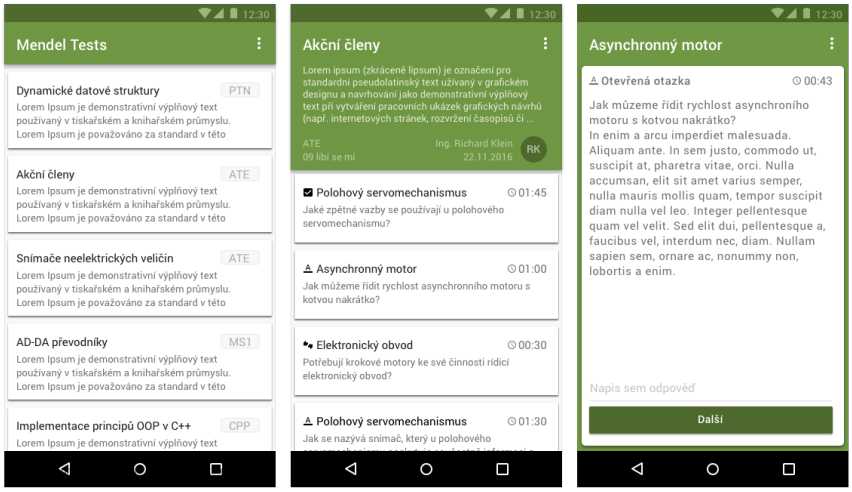
Návrh uživatelského rozhraní

Tuto aplikaci většinou budou využívat studenti, kteří jsou zvyklé na moderní UI. Aby aplikace pro ně byla nejenom užitečná, ale i atraktivní, rozhodl jsem se řídit vodítkem od společnosti Android (Google, 2016), který říká nám se držet tzv. Material Design. Uživatelské rozhraní má být čisté a robustní, což znamená jasné viditelné prvky, dostatečný kontrast a velikost každého elementů na obrazovce a poskytování klíčové informací hned, jinými slovy, klíčová informace aplikace má být dostupná v jakýkoliv okamžik. Rozhodl jsem se, že aplikace bude mít 3 hlavní pohledy:

1. Seznam všech dostupných balíčků otázek
2. Přehled jednotlivého balíčku
3. Pohled na aktivní proces testování

Dole jsou uvedeny návrhy a popisy pro uvedené výše pohledy:

První obrázek zleva zobrazuje, jak přibližně bude vypadat obrazovka zařízení hned po přihlášení do systému. To je obyčejný seznam všech dostupných balíčků pro uživatele. Pomocí protáhnutí doprava lze přidat určitý balíček do seznamu oblíbených pro snadnější přístup. Protáhnutí opačným směrem systém pochopí jako žádost na smazání ze seznamu dostupných. Kazda položka obsahuje informace o předmětu, pro který je tento balíček vytvořen a jeho tématu. Taky se zobrazuje kus popisu od autora, aby uživatel hned mohl pochopit o čem to testování je. Obrázek uprostřed zobrazuje pohled na vybraný uživatelem baliček otázek. Tedy lze najít kompletní



Obrázek 10: Návrh uživatelského rozhraní Android aplikace

informace: o čem tento balík je, kdo a kdy jeho vytvořil a kolika studentům se líbil. Navíc je tam krátký přehled otázek - témata, typ, dostupný čas pro řešení atd. Krajní obrázek zobrazuje, jak by vzorně měla vypadat obrazovka během testování. Rozhodl jsem nepřidávat tam nic zbytečného a soustředit uživatele na úkolu. Pro to je dominantní barva bílá. V závislosti na tom, kterého typu je otázka, může se měnit spodní část obrazovky: textové pole, tlačítka nebo přepínače. Po stisknutí na tlačítko ”Další” se zahájí animace přechodu mezi otázkami.

7 Aplikace MendelTests

V této kapitole popíšu průběh vytváření reálné Android aplikace pro testování znalosti studentů. Ale nejdřív, před začátkem implementace zadaní, mel jsem si zvolit minimální verze SDK. Tento bod je velmi důležitý, protože řeší kolik potenciálních uživatelů si mohou stáhnout aplikace a jaké omezeni na API má vývojář - některá funkčnost platformy může byt nedostupná na starších zařízení. Základními funkcemi aplikace jsou:

- Načítaní dostupných dat z REST serveru v json formátu
- Ukládaní dat do lokální databáze, aby uživatel mohl používat aplikace i bez aktivního připojení k sítí
- Provádění testování s podporou několika typů otázek, časovým limitem a zobrazením nápovědy v případě spatné odpovědi
- Přidání libovolné sady otázek do položky "Oblíbené" pro zjednodušeny přístup
- Ukázání statistiky pro jednotlivou sadu otázek
- Synchronizace lokální databázi a externího zdroje dat

7.1 Struktura aplikace

Navrh struktury aplikce se dodrzuje znamou třívrstvou architekturu (Microsoft, 2016).

1. `cz.savrov.mendeltests.data`
   Balik reprezentuje veskerou logiku pro manipulace s daty aplikace: nacitani, mazani a aktualizace dat. Tato cast je nezavisla na Android platforme a muze byt lence využíván v jine libovolne aplikaci - jenom staci zmenit POJO tridy zakladnich objektu a pristupove body k API serveru.
   Existuje jenom dva omezeni, ktere vyuzivaji context aplikace. Prvnim je trida, ktera resi praci s cookie soubory, coz umoznuje nam vytvorit komunikacni kanal klient-server a zajistit identifikace uzivatelu. Druhim je pouzivani interni databaze, ktera potrebuji mit pristup ke kontextu aplikace.
2. `cz.savrov.mendeltests.domain`
   Tady je uložen kód, který řeší všechny potřebné případy užití pro tuto aplikaci. Například, mimo obecných use case jako dostat seznam otázek, provést přihlášeni atd, jsou tam trošku specifické případy – dostat údaje o autorovi sady otázek. Tento balíček je nezávislý na Android SDK a taky se může byt využíván v jiné aplikaci. Kazdy use case probíhá mimo tzv. Main Thread. Na to se používá třida `ThreadPoolExecutor`, která se stara o rychlé poskytování už spouštěných vláken pro provedení jednotlivých případu užití.
3. `cz.savrov.mendeltests.mvp`

Jak je zřejmě z názvu balíčku, tady je uložena část, která se stara o vizualizaci dat. Všechny aktivity, fragmenty, služby jsou uloženy zde.

Navíc mám ještě několik doplňkových balíčku jako `cz.savrov.mendeltests.utils` a jiné kde se nachází pomocné třídy.

7.2 AndroidManifest.xml

Už byly probrány vlastnosti tohoto velmi důležitého souboru dřív na straně 27, ale to bylo obecně. Teď se podíváme na manifest daného projektu.

Soubor se začíná z nastavení parametru `package`, který slouží k identifikaci aplikace. V našem případě je to `cz.savrov.mendeltests`. Dále se popisuji všechny povolení, které potřebuje aplikace od systému. Například pro nás je velmi důležité mít dovolení k používaní sítě vzhledem k tomu, že data se ukládají na externí server. Pak je popsaná samotná aplikace - jaké má aktivity a služby. Zatím aplikace potřebují jenom jednu službu, která má flag `exported` nastaveny na hodnotu `false`, což znamená, že tuto službu může využívat jenom tato aplikace.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cz.savrov.mendeltests">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
        android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"
        />

    <application
        android:name=".App"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity"/>
        <activity android:name=".mvp.packlist.PackListActivity"
            android:label="@string/title_activity_pack_list"
            android:theme="@style/AppTheme">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".mvp.packsingle.PackSingleActivity"/>
        <activity android:name=".mvp.packactive.PackActiveActivity"
```

```
            android:windowSoftInputMode="stateAlwaysHidden|adjustResize"/>
        <activity android:name=".mvp.profile.ProfileActivity"/>

        <service android:name=".mvp.service.PushHistory"
            android:exported="false"/>
    </application>

</manifest>
```

7.3 Knihovny

Pro zjednodušení vývoje vývojáři se často používají hotové řešení, od svých kolegů – Android vývojářů. Je to rychlý a jednoduchý způsob přidat požadovanou funkcionalitu do vlastního projektu.

Pro import hotových řešení (dale *knihovny*) lze použít soubor `build.gradle`. Tento soubor obsahuje informaci, která je potřebná při sestaveni celého projektu. V tento okamžik nás zajímá sekce `dependencies`, která obsahuje odkazy na repositáře jednotlivých knihoven.

```
dependencies {
    ...

    compile 'com.android.support:appcompat-v7:25.1.0'
    compile 'com.android.support.test.espresso:espresso-core:2.2.2'
    compile 'com.android.support:design:25.1.0'
    compile 'com.android.support:cardview-v7:25.1.0'
    compile 'com.android.support:recyclerview-v7:25.1.0'

    compile 'com.github.bumptech.glide:glide:3.7.0'
    compile 'jp.wasabeef:glide-transformations:2.0.1'

    compile 'com.squareup.retrofit2:retrofit:2.1.0'
    compile 'com.squareup.retrofit2:converter-gson:2.1.0'
    compile 'com.squareup.okhttp:okhttp:2.5.0'

    compile 'com.jakewharton:butterknife:8.4.0'
    annotationProcessor 'com.jakewharton:butterknife-compiler:8.4.0'

    compile 'com.nineoldandroids:library:2.4.0'
    compile 'com.daimajia.easing:library:1.0.1@aar'
    compile 'com.daimajia.androidanimations:library:1.1.3@aar'

    compile 'com.github.lecho:hellocharts-library:1.5.8@aar'

    compile 'com.wang.avi:library:2.1.3'
```

```
    testCompile 'junit:junit:4.12'
    testCompile 'org.mockito:mockito-core:1.10.19'
    testCompile "org.robolectric:robolectric:3.1.4"
}
```

7.4 Uživatelské rozhraní

Vzhled aplikace je řešen v balíčku `cz.savrov.mendeltests.mvp`. Obsah jednotlivého pohledu se skládá hlavně z fragmentu, který řeší prezentaci dát, a presenteru, který se stará o poskytování dát fragmentu. Tento přístup vytváření prezentační vrstvy pro Android aplikace se jmenuje MVP (Model-View-Controller).

Každá schéma obrazovky je popsána pomoci XML kódu v `res/layout`, ale některé části, které se mění za běhu programu, jsou taky popsané uvnitř Java tříd, prezentujících fragmenty. Například, pro umístění tlačítek a dalších statických UI elementů stačilo definovat jich v XML souborech, zatím co pro vytváření animace nějaké součástí obrazovky jsem měl použít zdrojový kód fragmentu.

Velká pozornost byla věnována jednoduchosti uživatelského pohledu a podpoře různých typů zařízení.

Seznam balíčků otázek

Tento pohled se skládá ze seznamů sad otázek a jednoduché navigace, pomocí které lze přijít do profilu uživatele a mít stručný statistický přehled využívání aplikace a tlačítka pro seřazení seznamu podle data nebo stavu synchronizace. Většina prostoru obrazovky je k dispozici pro samotný seznam, který je reprezentován elementem RecycleView (viz. v sekci 5.5 na straně 29).

Kód, který se stará o tento pohled, je uložen do fragmentu `PackListFragment` a je součástí `PackListActivity`. Jako každá třída, která se zabývá prezentací dát, tento fragment má vlastní `PackListPresenter`, který poskytuje data.

Sám pohled je vytvořen pomocí dvou schémat - jedno je pro zobrazení dostupných dát, druhé je určeno pro chybové hlášení. Tato dvě schémata jsou zabalená do elementu `SwipeRefreshLayout`, který umožňuje jednoduchou implementace swipu pro aktualizace dát, ze serveru nebo lokálního úložiště.

```
<?xml version="1.0" encoding="utf-8"?>
<cz.savrov.mendeltests.utils.ScrollSwipeRefreshLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/refresh_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior">
```

```
    <FrameLayout
        android:id="@+id/container"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:id="@+id/list_layout"
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <android.support.v7.widget.RecyclerView
                android:id="@+id/recycler_view"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:scrollbars="vertical"
                />

        </LinearLayout>

        <!-- Error case layout -->
        <LinearLayout...>

    </FrameLayout>
</cz.savrov.mendeltests.utils.ScrollSwipeRefreshLayout>
```

Jednotlivá položka seznamu je vytvořena pomoci widgetu CardView a dvou textových polí - název a stručný popis. Samotný pohled lze videt na obrázku 10-a.

Přehled balíčku otázek

Schéma tohoto pohledu je o něco složitější, než předcházející. Odlišenost je v tom, že tady, mimo tlačítek na hoře, se používají dva fragmenty. První zobrazuje seznam otázek, aby uživatel věděl, co přibližně jeho čeká, druhý fragment obsahuje statistické data a jejich přehled ve tvaru kruhového diagramu.

Implementace fragmentu se seznamem je velmi podobná předchozímu fragmentu se seznamem sad otázek. Je tam taký fragment `PackSingleOverviewFragment` pro zobrazení dát z `PackSingleOverviewPresenter`. Pro aktualizace dát se používá již nám známý SwipeRefreshLayout. Jednotlivá položka seznamu obsahuje data o typu otázky, záhlaví a stručný popis otázky. Navíc je tam uveden dostupný čas pro řešení každé úlohy. Všechny tyto údaje jsou prezentované pomocí TextView widget. Tomuto fragmentu odpovídá obrázek 10-b.

```
<cz.savrov.mendeltests.utils.ScrollSwipeRefreshLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/refresh_layout"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.v4.widget.NestedScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent" >

        <RelativeLayout android:id="@+id/container"
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <LinearLayout android:id="@+id/list_layout"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:orientation="vertical">

                <android.support.v7.widget.RecyclerView
                    android:id="@+id/recycler_view"
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:scrollbars="vertical" />

            </LinearLayout>

            <!-- Error case layout -->
            <LinearLayout...>

        </RelativeLayout>

    </android.support.v4.widget.NestedScrollView>

</cz.savrov.mendeltests.utils.ScrollSwipeRefreshLayout>
```

Druhý fragment je reprezentován diagramem, který umožňuje znázornit statistiku využívání balíčku za posledních 12 měsíců. Diagram je vytvořen pomocí externí knihovny a v xml souboru je prezentován jenom jedním elementem `BarChart` - sloupcovým diagramem.

```
<cz.savrov.mendeltests.utils.ScrollSwipeRefreshLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/refresh_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="cz.savrov.mendeltests.mvp.packsingle.history.PackSingleHistoryFragmen
```

```
    <RelativeLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:paddingBottom="8dp"
        android:paddingEnd="2dp"
        android:paddingStart="2dp"
        android:paddingTop="8dp">

        <LinearLayout
            android:id="@+id/data_layout"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">

            <com.github.mikephil.charting.charts.BarChart
                android:id="@+id/chart"
                android:layout_width="match_parent"
                android:layout_height="match_parent" />

        </LinearLayout>

        <!-- Error state layout -->
        <LinearLayout...>

    </RelativeLayout>

</cz.savrov.mendeltests.utils.ScrollSwipeRefreshLayout>
```

Přechod mezi těmito fragmenty je zajištěn pomocí třídy `ViewPagerAdapter`, která ukládá informace o tom jaký fragment má být zobrazen a v jakém pořadí. Ale zájmovější je využívání `CoordinatorLayout` jako kořeny element pohledu rodičovské aktivity. Obsahem tohoto elementu je `AppBarLayout` a vlastní implementace elementu `ViewPager`, který je spojen s výše uvedeným `ViewPagerAdapter`. `AppBarLayout` lze chápat jako více vyvinutý `LinearLayout` (viz. v sekci 5.5 na straně 29), který může obsahovat další `View` elementy s tzv. flagem, určující chování při swipu. V této aplikaci informace o balíčku se nachází uvnitř AppBarLayout a má flagy `scroll|exitUntilCollapsed|snap`. Tím pádem při swipu dolů tato část zmizí z obrazovky, což umožní využít více prostoru pro zobrazení seznamu otázek v balíčku.

```
<android.support.design.widget.CoordinatorLayout
    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/coordinator_layout"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:fitsSystemWindows="true"
tools:context=".mvp.packsingle.PackSingleActivity">

<android.support.design.widget.AppBarLayout
    android:id="@+id/app_bar_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:fitsSystemWindows="true"
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">

    <android.support.design.widget.CollapsingToolbarLayout
        android:id="@+id/collapsing_toolbar"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fitsSystemWindows="true"
        android:minHeight="?attr/actionBarSize"
        app:contentScrim="?attr/colorPrimary"
        app:expandedTitleMarginEnd="64dp"
        app:expandedTitleMarginStart="48dp"
        app:layout_scrollFlags="scroll|exitUntilCollapsed|snap">

        <!-- Header section content. Like pack description, info about
            author and so on -->

    </android.support.design.widget.CollapsingToolbarLayout>
</android.support.design.widget.AppBarLayout>

<cz.savrov.mendeltests.mvp.utils.CustomViewPager
    android:id="@+id/content"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior" />

<android.support.v7.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="@color/colorPrimary"
    app:layout_anchor="@id/flHeader"
    app:layout_collapseMode="pin"
    app:theme="@style/ThemeOverlay.AppCompat.Dark">
```

```
        <!-- Toolbar content. Like navigation and title -->

    </android.support.v7.widget.Toolbar>

</android.support.design.widget.CoordinatorLayout>
```

Flagy `scroll|exitUntilCollapsed` tvoří jeden celek, který říká systému, že při swipu nahoru View, který tento flag má, bude se valit nahoru, pokud nezmizí. Poslední flag `snap` znamená, že View vždy má být stočený do nejbližšího kraje.

Vědomostní test

Samotné testování, je taky prezentované fragmentem `PackActiveTestFragment`, který, jako všechny ostatní fragmenty v projektu, má vlastního poskytovatele dát `PackActiveTestPresenter`. Struktura pohledu je nejsložitější v projektu a má tři základní sekce - otázka, správná odpověď a špatná odpověď. Přechod mezi těmito sekcemi je realizován pomocí animace a je jenom vizuální, nejde tady o změnu fragmentu.

Sekce s obsahem otázky je reprezentovaná pomocí elementu `CardView`. Tento představitel třídy `View` může obsahovat uvnitř další všemožné `View` elementy. Tělem `CardView` v tomto případě je dva elementy typu `LinearLayout`. První obsahuje informace o otázce, například popis, typ, zbývající čas atd. Druhý `LinearLayout` se zabývá tím, že zobrazuje vhodnou pro tuto otázku odpověď, buď to `EditText` pro otevřenou otázku, nebo `Button` pro různé typy testu. Tomuto fragmentu odpovídá obrázek 10-c.

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPrimary"
    android:visibility="visible">

    <android.support.v7.widget.CardView
        android:id="@+id/cvQuestion"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:visibility="visible"
        app:cardElevation="4dp">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical"
```

```
            android:padding="14dp">

            <!-- Layout for question description -->
            <LinearLayout...>

            <LinearLayout
                android:id="@+id/llAnswer"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="vertical">

                <!-- Layout for TEST type answer -->
                <LinearLayout...>

                <!-- Layout for TRUE/FALSE type answer -->
                <LinearLayout...>

                <!-- Layout for OPEN type answer -->
                <LinearLayout...>

                <!-- Layout for MULTIPLE CHOICE type answer -->
                <LinearLayout...>

            </LinearLayout>

        </RelativeLayout>

    </android.support.v7.widget.CardView>

    <!-- Layout for CORRECT answer -->
    <RelativeLayout...>

    <!-- Layout for WRONG answer -->
    <RelativeLayout...>

</RelativeLayout>
```

Sekce s potvrzení správnosti otázky se skládá z několika `TextView` a jedné `ImageView`. Tento pohled je velmi jednoduchý, a zajímavě má jenom to, že zobrazuje náhodné vybrané gratulace pro uživatele.

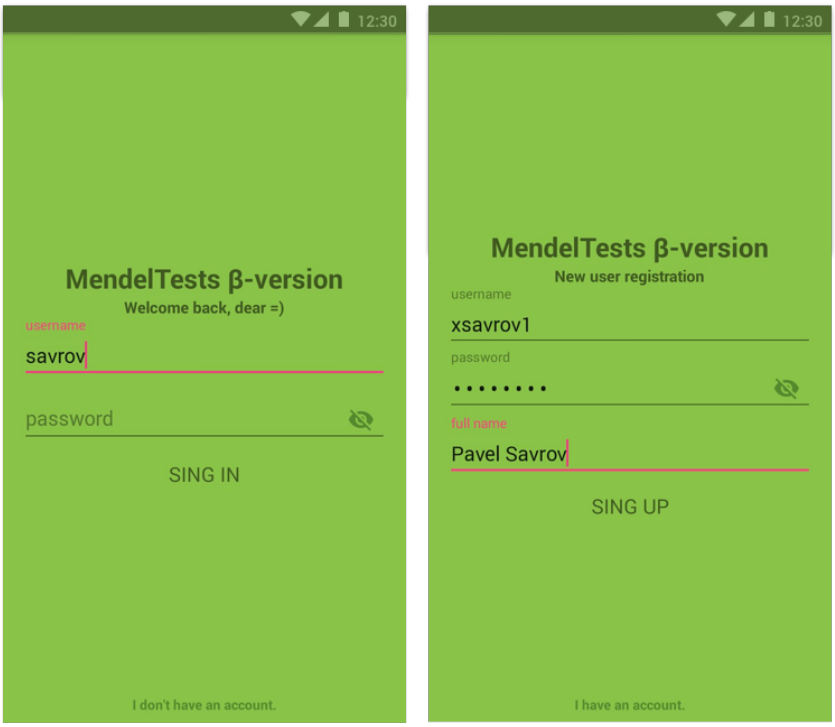
Poslední sekce, oznámení o špatně odpovědi, je velmi podobná předchozí, ale se rozlišuje jenom v tom, že místo gratulace je tam uvedena nápověda a správné odpovědi, aby uživatel věděl, co měl špatně a jak to má být ve skutečnosti.

Po testování uživatel se přesměruje na fragment, který mu zobrazí výsledek. Fragment `PackActiveResultFragment` má jenom dva důležité `View`. Prvním je kru-

hový diagram `PieChartView` dostupný z externí knihovny, který říká kolik procent dostal student za test. A druhým je `TextView`, který informuje o tom, jaké místo obsadil student mezi uživateli.

Přihlášeni a registrace

Tyto dva pohledy jsou velmi podobné, liší se jenom v počtu elementů `EditText`. Srdce schématu tvoří `TextInputLayout` a `EditText`, který k němu je přiřazen. Spojením těchto dvou `View` lze dosáhnout pomocí moderní animace, když, po stisknutí na `EditText`, návěští se vláčně posune nahoru. A samozřejmě jsou tam tlačítka typu `Button` pro posílání vstupních dát na server.



Obrázek 11: Přihlášení a registrace nového uživatele

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
 android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="@color/colorPrimary"
android:visibility="visible">

   <RelativeLayout
```

```
    android:id="@+id/rlSingIn"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_alignParentTop="true">

    <-! Layout for 'TextViews like app name and so on ->
    <…LinearLayout>

    <LinearLayout
      android:id="@+id/llInput"
      android:layout_width="match_parent"
               android:layout_height="wrap_content"
      android:layout_centerInParent="true"
      android:layout_marginEnd="@dimen/fab_margin"
      android:layout_marginStart="@dimen/fab_margin"
      android:orientation="vertical">

        <android.support.design.widget.TextInputLayout
          android:id="@+id/tilUsername"
          android:layout_width="match_parent"
          android:layout_height="wrap_content">

         <EditText
          android:id="@+id/etUsername"
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:hint="@string/username"
          android:inputType="text"/>

      </android.support.design.widget.TextInputLayout>

      <-! Very similar layout for password field ->
      <android.support.design.widget.…TextInputLayout>

      <-! registration button ->
      <…Button/>
    </LinearLayout>
    </RelativeLayout>
</RelativeLayout>
```

7.5 Zpracováni dat

Aplikace je zaměřena na prezentaci externích dat, tím pádem zpracování těchto dat je klíčovou funkcionalitou aplikace. Při práci s daty naše aplikace se má postarat o

získání aktuálních dat z externího serveru, aktualizování interní databáze a aktualizace modifikovaných dat.

Získání dat

Dostupné služby

Všechna potřebná data získáváme z ječného serveru, který poskytuje REST (Representation State Transfer) rozhrání. Libovolná REST adresa se skládá z několika částí:

- základní URL adresa serveru
- adresa služby
- parametry požadavků

Například, pro získání seznamu otázek pro baliček s id 1 máme použít následující URL adresu: `http://localhost/questions?packId=1`. Pro účely aplikace jsou dostupné následující služby:

Adresa	HTTP metoda	Popis
/users	POST	vytvářeni nového uživatele
/users/ID	DELETE	smazat uživatele podle id (jenom pro aktuálního uživatele)
/users/ID	PUT	aktualizovat uživatele podle id
/users?q=QUERY	GET	získání seznamu uživatelů podle požadavku
/users/ID	GET	získání uživatele podle id
/packs	POST	vytvářeni nového balíčku
packs/ID/ ?_expand=user	GET	získat autora podle id balíčku
/access?userId=ID& _expand=pack	GET	získat balíčky, ke kterým uživatel má přístup
/packs/ID	GET	získat balíček podle id
/questions	POST	vytvářeni nové otázky
/questions/ID	DELETE	mazání otázky podle id (jenom pro vlastníka balička)
/questions/ID	PUT	aktualizovat otázku podle id (jenom pro vlastníka balička)
/questions?packId=ID	GET	získat otázky podle id balíčku
/history	POST	vytvářeni nového výsledku o úspěšnosti
/history?userId=ID	GET	získat výsledky podle id uživatele

Formát dat

Data dostáváme ve formátu JSON, což pro nás je velkou výhodou kvůli jednoduchosti zpracováni a malé velikosti souboru. Alternativou mohl by být formát CSV, ale není tak dobře strukturovaný (je složitě načíst nějaké pole dat) a průhledný. Je to pravda, že CSV soubor váži méně než JSON soubor, ale ten rozdíl není tak velký. Příklad JSON souboru, když uživatel požádá server o balíček s id = 1:

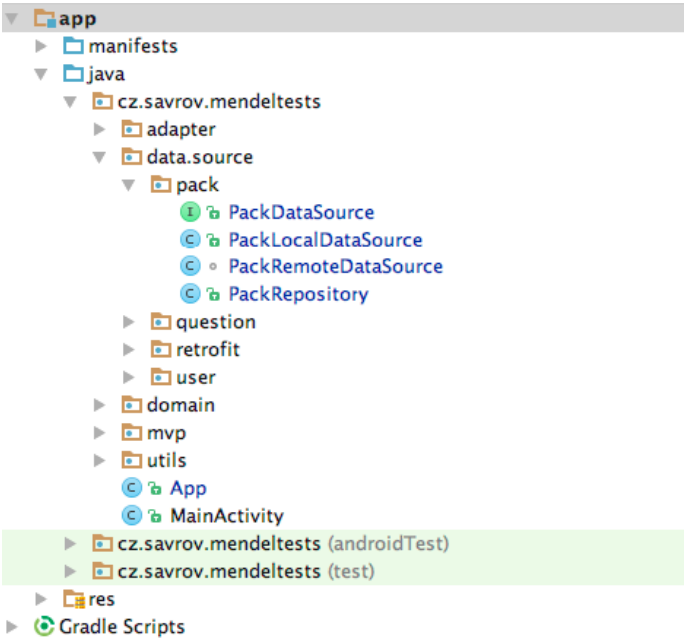
```
{
  "id": 1,
  "title": "Set title",
  "category": "Set category",
  "description": "Set description"
  "userId": 3,
```

```
  "date": 1479812324000
}
```

Prenos dat

Když máme připravené služby, můžeme začít je používat pomocí aplikace. Jak už jsem zmínil dřív, přenos dat ze serveru a zpátky je řešeno v balíčku `cz.savrov.mendeltests.data.source`. Aby aplikace dokázala pracovat se serverem musíme nejdřív se postarat o tom, aby se naučila přijímat a odesílat požadavky. Na to jsem rozhodl použít rámec Retrofit (Retrofit, 2016). Je HTTP klientem pro Android, který používá OkHttp knihovnu (OkHttp, 2016). Balíček `retrofit` kromě třídy Retrofit obsahuje rozhraní, které popisuje přístupové body. Navíc je tam balíček `deserializer`, třídy kterého se starají o správném přečtení JSON souboru - některé požadavky, které posíláme na server, provádějí výběr několika tabulek externí databázi a tím pádem dostáváme v JSON souboru položky, které teď nepotřebujeme.

Ještě v balíčku `cz.savrov.mendeltests.data.source` lze najit podadresáře s implementacemi jednotlivých služeb (obr. 12).



Obrázek 12: Prehled struktury balicku pro prenos dat

Z obrázku výše je vidět že každý podadresář má rozhraní, kterého se dodržuji třídy, implementující lokální a externí datové zdroje. Navíc je tam další třida (tady na obrázku se jmenuje `PackRepository.java`), která obsahuje obecné implementace pro zpracování dat určitého druhu.

Zpracování dat

Pomoci knihovny Retrofit, o které jsem zmínil par vet v sekci 7.3 na straně 45 dostavám ze serveru už připravené POJO objekty. Tím pádem se vůbec nemusím starat o parsování dat na rozdíl od případu, když data jsou v CSV formátu. Hned po přijmu aktuálních dat, provádím aktualizaci lokální databázi. Algoritmus, který řeší tuto problematiku vypadá takto:

```
@Override
public void getPackList(long userId, @NonNull GetPackListCallback
    callback) {
    checkNotNull(callback, "callback cant be null");

    // rozhodnuti zda cache je validni nebo ne
    // v pripade ze validni - aktualizace dat cache z lokalniho zdroju
    // v opacnem pripade - posilame pozadavek na server

    if (isCacheValid) {
        updateCacheFromLocalDataSource(userId, callback);
    } else {
        updateCacheFromRemoteDataSource(userId, callback);
    }
}

private void updateCacheFromLocalDataSource(long userId, final
    GetPackListCallback callback) {
    localDataSource.getPackList(userId, new GetPackListCallback() {
        @Override
        public void onDataLoaded(List<Pack> packs) {
            // jednoducha aktualizace cache: smazat vsechno - nacist znovu
            refreshCache(packs);
            callback.onDataLoaded(packs);
        }

        @Override
        public void onDataNotAvailable(String reason) {
            callback.onDataNotAvailable(reason);
        }
    });
}

private void updateCacheFromRemoteDataSource(final long userId, final
    GetPackListCallback callback) {
        remoteDataSource.getPackList(userId, new GetPackListCallback() {
            @Override
            public void onDataLoaded(List<Pack> packs) {
```

```
            // jednoducha aktualizace cache: smazat vsechno - nacist
               znovu
            refreshCache(packs);
            // aktualizace lokalni DB. Aktualizuji se jenom te prvky,
            // ktere byli nacteny ze serveru
            refreshLocalDataSource(packs);
            callback.onDataLoaded(packs);
        }

        @Override
        public void onDataNotAvailable(String reason) {
            callback.onDataNotAvailable(reason);
            updateCacheFromLocalDataSource(userId, callback);
        }
    });
}
```

Lokální databáze

Z ohledu na to, že chceme aby aplikace dokázala pracovat bez jakéhokoliv omezení i v případě, ze mobilní zařízeni není připojeno na síť, potřebujeme použit lokální uložiště. Určitě lze použit klasický databázový systém pro Android - SQLite, ale teď se na trhu objevil jeho konkurent, který se jmenuje **Realm**. Během vývoje této aplikace přečetl jsem hodně článků a většina lidi tvrdí, že to je fakt dobré řešení. A já se s tím souhlasím.

Aby vývojář mohl začít využívat tento databázový systém, má provést dva jednoduché nastavení:

- provést inicializaci v hlavní třidě aplikace (následník třídy `Application`)
- každá POJO třida má být následníkem abstraktní třídy `RealmObject`

Určitě existují i další nastavení, například: anotace, schéma, verze a další, díky kterým lze nastavit databázový systém podle svých požadavků.

Co se tyká sestavení dotazů, tady se nevyužívá klasický SQL, místo toho vývojářovi je předloženo používat metody. Například:

```
@Override
public void getPack(long packId, @NonNull GetPackCallback callback) {
    Pack pack = realm.where(Pack.class)
            .equalTo("id", packId)
            .findFirst();
    if (pack != null) {
        callback.onDataLoaded(pack);
    } else {
```

```
        callback.onDataNotAvailable("There is no pack with that ID in
            local data source");
    }
}
```

Synchronizace dat

V případě, že nastane situace, že uživatel využíval aplikace, ale data se nemohli hned aktualizovat, rozhodl jsem uložit jich do lokální databáze se speciálním tágem. Pak zavést službu, která při vraceni moznosti připojení na internet, prohlídá lokální databázi, najde všechny prvky s určitým tágem a posle je na server. Je možné, že na serveru tyto prvky dostanou nové id. V tomto případě je potřeba nahradit zastaralé prvky a nezapomenout na aktualizace propojených s nimi dat.

```
public class PushHistory extends IntentService {
    private static final String TAG = PushHistory.class.getSimpleName();

    public PushHistory() {
        super(TAG);
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        UseCaseHandler handler = UseCaseHandler.getInstance();
        UseCaseSaveHistory useCaseSaveHistory = new UseCaseSaveHistory();
        if (checkIfUnsynchHistoryExists()) {
            List<History> list = getUnsynchHistoryList();

            for (History history :
                    list) {
                UseCaseSaveHistory.RequestValues requestValues = new
                    UseCaseSaveHistory.RequestValues(history);
                handler.execute(useCaseSaveHistory, requestValues, null);
            }

        }
    }
}
```

8 Závěr

Cílem práce bylo vytvořit softwarový nastroj pro podporu učení studentů, aby všechny uživatele aplikace měli jednoduchou, rychlou a bezbariérovou možnost ověřování svých znalosti z určitých okruhů. Pro splnění tohoto cíle mel jsem vypracovat mnou definované vedlejší cíle.

Bakalářská práce je rozdělena do několika etap. Na začátku jsem se podíval na trh hotových řešení a zjistil jsem výhody jednotlivých aplikací. V dalším kroku jsem dozvěděl jak se vytváří testy pro kontrolu znalosti: jaké typy otázek se používají a jaké jsou pravidla pro jejich vytváření. Pak jsem prostudoval problematiku technologií spojených s vývojem aplikací pro nejpopulárnější mobilní platformu Android. Po získání těchto znalosti jsem udělal obecný návrh mobilní aplikace pro testování studentů a návrh webové aplikace pro správu otázek. Když návrh byl vytvořen mohl jsem začít vývojem samotné mobilní aplikace. Projekt byl rozdělen do třech vrstev: datová vrstva, vrstva obchodní logiky a prezentační vrstva. Každá vrstva řeší vlastní úkoly a jsou slabé závislé na sobě. Během vývoje aplikace byla otestována na několika reálných zařízeních s různou charakteristikou. Tím pádem jsem zajistil, že kód funguje podle mých očekávání a nevyvolává nečekané chyby.

Z toho důvodu, že škola zatím nemá nachystaný server pro tuto aplikaci, měl jsem rozjet svůj vlastní lokální server, který dovoloval mne simulovat práce s reálným serverem - registrace a přihlášení uživatele, načítání dát ze serveru a posílání zpátky, ovládat různé chyby, například, nedostupnost serveru atd. Možná po nasazení na reálný server aplikace potřebuje provést drobné úpravy, například nastavit správnou adresu pro komunikaci se službami externího serveru. Ale celkem mobilní aplikace spolupracuje se serverem v bezchybném režimu.

Určitě tato aplikace později může být rozšířena o chybějící funkčnosti. Například, poskytování přesnější statistiky nebo přihlášeni pres sociální sítě. Ale podle mne, užitečnější bude vytvořit modul, který dokáže ohodnotit odpovědí u tz. otevřených otázek. Zatím žádné řešení této úlohy pro češtinu jsem nenasel.

9 Reference

R. WILLIAMS, Apple iOS: a brief history [online]: Telegraph Media Group, 2015 [cit. 2016-10-22]. Dostupné z: http://www.telegraph.co.uk/technology/apple/11068420/Apple-iOS-a-brief-history.html.

C. LATTNER, Chris Lattner's Homepage [online]: Chris Lattner, 2014 [cit. 2016-10-22]. Dostupné z: http://nondot.org/sabre/.

TUTORIALSPOINT, Android Tutorial [online]: TutorialsPoint, 2016 [cit. 2016-10-22]. Dostupné z: https://www.tutorialspoint.com/android.

STATISTA INC., Statista - The Statistics Portal for Market Data, Market Research and Market Studies [online]: Statista Inc, 2016 [cit. 2016-10-22]. Dostupné z: https://www.statista.com/.

WINDOWNS DEV CENTER, Getting started with developing for Windows Phone 8 [online]: Microsoft, 2016 [cit. 2016-10-22]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/apps/ff402529(v=vs.105).aspx.

GOOGLE, Android Studio: The Official IDE for Android [online]: Google, 2016 [cit. 2016-10-22]. Dostupné z: https://developer.android.com/studio.

GENYMOBILE, Genymotion – Fast & Easy Android Emulator [online]: Genymobile, 2016 [cit. 2016-10-22]. Dostupné z: https://www.genymotion.com.

GOOGLE, Introduction to Android [online]: Google, 2016 [cit. 2016-10-22]. Dostupné z: https://developer.android.com/guide/index.html.

MASUMI NAKAMURA, Marko Gargenta. Learning Android, 2nd Edition. Farnham: O'Reilly, 2014. ISBN 9781449336226.

BEN CLAY. Is This a Trick TrickQuestion? A Short Guide to Writing Effective Test Questions [online]: Kansas Curriculum Center, 2001 [cit. 2016-11-8]. Dostupné z: https://www.k-state.edu/ksde/alp/resources/Handout-Module6.pdf.

CYNTHIA J. BRAME. Writing good multiple choice test questions [online]. Vanderbilt University, 2013 [cit. 2016-11-8]. Dostupné z: https://cft.vanderbilt.edu/guides-sub-pages/writing-good-multiple-choice-test-questions.

ISPRING SOLUTIONS, 8 Tips for Writing Good Fill-in-the-Blank Questions in E-Learning Courses [online]: iSpring Solutions, 2016 [cit. 2016-10-22]. Dostupné z: http://www.ispringsolutions.com/blog/8-tips-for-writing-good-fill-in-the-blank-questions-in-e-learning-courses.

QUANTCAST, Quizlet.com Traffic and Demographic Statistics by Quantcast [online]: Quantcast, 2016 [cit. 2016-11-12]. Dostupné z: https://www.quantcast.com/quizlet.com.

ALEXANDER KLIMOV, Learn Android easily [online]: alexsanderklimov, 2016 [cit. 2016-11-19]. Dostupné z: http://developer.alexanderklimov.ru/android.

CODEPATH, CodePath Android Cliffnotes [online]: CodePath, 2016 [cit. 2016-11-19]. Dostupné z: https://guides.codepath.com/android.

VOGELLA, Android Development [online]: Vogella, 2016 [cit. 2016-11-21]. Dostupné z: http://www.vogella.com/tutorials/android.html.

MOCKITO, Mockito framework [online]: Mockito, 2016 [cit. 2016-11-21]. Dostupné z: http://site.mockito.org.

MICROSOFT, Three-Layered Services Application [online]: Microsoft, 2016 [cit. 2016-12-8]. Dostupné z: https://msdn.microsoft.com/en-us/library/ff648105.aspx.

RETROFIT, A type-safe HTTP client for Android and Java [online]: Square, 2016 [cit. 2016-12-8]. Dostupné z: https://square.github.io/retrofit/.

OKHTTP, An HTTP & HTTP/2 client for Android and Java applications [online]: Square 2016 [cit. 2016-12-8]. Dostupné z: https://square.github.io/okhttp/.

A. PHILLIPS, Software Development Methodologies [online]: CodeProject, 2010 [cit. 2016-12-9]. Dostupné z: https://www.codeproject.com/articles/124732/software-development-methodologies.

P. REJNKOVÁ, Příklady použití diagramů UML 2.0 [online]: Praha, 2009 [cit. 2016-12-9]. Dostupné z: http://uml.czweb.org/index.html.

GOOGLE, Material design [online]: Kalifornie, 2016 [cit. 2016-12-9]. Dostupné z: http://material.google.com.

MICROSOFT, Základy návrhu databáze [online]: Redmond, 2010 [cit. 2016-12-9]. Dostupné z: https://support.office.com/cs-cz/article/Z%C3%A1klady-n%C3%A1vrhu-datab%C3%A1ze-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5.

S. HORNÝ, Návrh uživatelského rozhraní webové aplikace [online]: Praha, 2014 [cit. 2016-12-9]. Dostupné z: http://gml.vse.cz/data/oppa-webdesign/ui.html.

C. WODEHOUSE, Server-Side Scripting: Back-End Web Development Technology [online]: Upwork Global Inc., 2015 [cit. 2016-13-9]. Dostupné z: https://www.upwork.com/hiring/development/server-side-scripting-back-end-web-development-technology/.

C. ZAPPONI, GitHut - Programming Languages and GitHub: London, 2014 [cit. 2016-13-9]. Dostupné z: http://githut.info/.

H. BOPARAI, 10 Reasons Why Python Scores Over PHP for Web Development: Net Solutions, 2014 [cit. 2016-13-9]. Dostupné z: https://www.netsolutionsindia.com/blog/10-reasons-why-python-scores-over-php-for-web-development/.

JETBRAINS, Kotlin Programming Language: JetBrains, 2016 [cit. 2016-13-9]. Dostupné z: https://kotlinlang.org/.