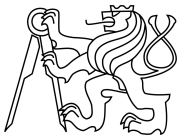
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Aplikace v OS Android pro tvorbu a vyplňování testů
Student:	Bc. Jakub Kalina
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Cílem práce je navrhnout a implementovat aplikaci v OS Android pro vzdělávání v rámci projektu Selftester. Aplikace umožní opakované vyplňování znalostních testů vytvořených v jiné aplikaci, kterou realizuje ve své magisterské práci Bc. Karolína Běhalová. Pomocí herních prvků bude uživatel motivován ke zlepšování výsledků.

Postupujte dle následujících kroků:
Proveďte analýzu alespoň 4 Android aplikací, které umožňují testování, a alespoň 3 dalších, které pro vzdělávání využívají herní prvky.
Na základě této analýzy navrhněte a implementujte vlastní funkční aplikaci, neopomeňte kvalitní návrh uživatelského rozhraní.
Aplikaci podrobte vhodným testům.
Na základě výsledku testů aplikaci upravte.
Využívejte správné postupy agilního vývoje mobilních aplikací, jednotlivé kroky dokumentujte.
Zhodnoťte přínosy aplikace a navrhněte další rozvoj.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 11. ledna 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Aplikace v OS Android pro tvorbu a vyplňování testů

Bc. Jakub Kalina

Vedoucí práce: Ing. Jiří Hunka

4. května 2017

Poděkování

Na tomto místě bych chtěl poděkovat svému vedoucímu Ing. Jiřímu Hunkovi za mnoho cenných rad při návrhu a realizaci práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 4. května 2017 .....

Odkaz na tuto práci

Kalina, Jakub. *Aplikace v OS Android pro tvorbu a vyplňování testů.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato diplomová práce je součástí projektu Selftester, jehož cílem je usnadnit přípravu na testy z různých oborů. V jejím rámci byla vytvořena mobilní aplikace pro OS Android umožňující opakované vyplňování a vyhodnocování znalostních testů vytvořených v diplomové práci Bc. Karolíny Běhalové.

V diplomové práci jsou rozebrány stávající konkurenční aplikace, výběr vývojové metodiky a její realizace.

Klíčová slova Android, mobilní aplikace, vzdělávání, testování

Abstract

This diploma thesis is part of the Selftester project, which aims to facilitate preparation for tests of various disciplines. Within this framework, a mobile application for Android OS has been created, which allows for the repeated completion and evaluation of the knowledge tests created in the thesis of Bc. Karolína Běhalová.

The diploma thesis deals with the existing competitive applications, the selection of the development methodology and its implementation.

Keywords Android, mobile application, education, testing

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V průběhu života se každý setkává s mnoha různými testy a zkouškami. Způsobů, jak se na ně připravovat je mnoho. Některé jsou efektivnější a některé méně. Tímto problémem se zabývalo již mnoho lidí a vytvořili několik metodik, jak se učit efektivně.

Projekt Selftester je založen na jedné z nejznámějších metod - PQRST. Skládá se z pěti jednotlivých fází: Preview, Question, Read, Summary, Test. V první fázi Preview probíhá seznamování s tématem. V druhé fázi Question si studující pokládá otázky, které se k tématu vztahují. Již tímto vymýšlením otázek probíhá učení dané problematiky. Ve třetí fázi Read probíhá jedna z nejdůležitějších částí učení - snaha nalézt odpovědi na otázky vymyšlené v předchozí fázi. Ve fázi Summary (shrnutí) by čtenář měl být schopen shrnout učené téma vlastními slovy. Pokud toho není schopný, měl by to zvládnout za pomoci seznamu klíčových slov. Posledním krokem je Test. V této fázi si čtenář pokládá jednotlivé otázky a zkouší na ně odpovědět. Pro kontrolu je často nutné nahlédnout opět do materiálů, čímž je dané téma lépe zapamatovatelné. [8]

V rámci diplomové práce Bc. Karolíny Běhalové bude možné ve webové aplikaci vytvořit vlastní testové otázky (fáze Question metody PQRST), které bude následně možné neomezeně testovat v mobilní aplikaci (fáze Test metody PQRST).

Tato diplomová práce dokumentuje celou tvorbu mobilní aplikace. V následující kapitole jsou rozebrány jednotlivé konkurenční aplikace pro srovnání a vyjasnění konkurenční výhody. Další kapitola se věnuje metodikám vývoje softwaru a výběru vhodné metodiky pro tuto mobilní aplikaci. Následující kapitoly rozebírají jednotlivé kroky návrhu, vývoje a testování vytvářené aplikace.

Kapitola 1

Analýza konkurenčních řešení

V této kapitole se diplomová práce zabývá rozborem aplikací, které jsou mobilnímu klientovi Selftester podobné. Umožňují prověřovat znalosti uživatelů. Aplikace byly vybrány na základě průzkumu trhu (českého i zahraničního).

Přímý konkurent projektu Selftester nebyl nalezen (možnost tvorby vlastních testů ve webové aplikaci a následné testování na mobilním klientovi). Konkurence však přijít může, neboť webová aplikace vyplnto.cz na svých stránkách již nyní uvádí, že brzy umožní automatickou kontrolu nad testy vytvořenými v této aplikaci. Nyní se zaměřuje pouze na vytváření průzkumů a dotazníků. Vyplnto.cz již mobilní aplikaci nabízí, až bude slibovaná funkcionalita vytvořena a spuštěna, bude pro mobilní aplikaci Selftester přímým konkurentem.

Analyzované aplikace jsou rozděleny na plně vzdělávací (kapitola 1.1) a aplikace pro vzdělávání využívající herní prvky pro lepší motivaci (kapitola 1.2).

V závěru této kapitoly je uveden přínos projektu Selftester oproti ostatním zmíněným aplikacím.

1.1 Aplikace umožňující testování

V této části jsou analyzovány aplikace, které slouží pro vzdělávání. Většinou jsou zaměřené na konkrétní téma a pomocí testovacích sad otázek uživatele učí. Analýza těchto aplikací se zaměřuje na následující oblasti:

- cílová skupina,
- obsahová stránka aplikací,
- uživatelské rozhraní,
- typy otázek,
- klady a zápory.

Aby bylo možné jednotlivé aplikace porovnat z pohledu uživatelského rozhraní, je u každé aplikace vyhodnocena Nielsenova heuristická analýza aplikací.

Nielsenova heuristická analýza se skládá z deseti jednotlivých heuristik. [9]

1. Viditelnost stavu systému
   - Systém by měl vždy uživateli ukazovat stav, co se děje.
2. Shoda mezi systémem a realitou
   - Systém by měl používat termíny z reálného světa, používat normální lidské fráze, ne systémové termíny.
3. Minimální zodpovědnost a stres
   - Často uživatel klikne na některé funkce omylem. Systém by měl umožnit vždy se vrátit a obnovit stav před nechtěnou uživatelskou akcí.
4. Shoda s použitou platformou a obecnými standardy
   - Aplikace by měla dodržovat obecné standardy používané na dané platformě.
5. Prevence chyb
   - Systém by se měl snažit chybám předcházet dřív než nastanou. Uživatel by neměl být schopen zadat hodnoty, které nejsou validní.
6. Kouknu a vidím
   - Systém by se měl snažit o snadnou orientaci uživatele. V každé části aplikace by mělo být uživateli zřejmé, kde se zrovna nachází a nemusel si pamatovat, jak se do dané části dostal.
7. Flexibilita a efektivita
   - Aplikace by měla být snadno ovladatelná pro nové uživatele. Těm, kteří používají aplikaci déle by měla umožnit pokročilejší možnosti nastavení a kontroly.
8. Minimalita
   - Aplikace by neměla obsahovat prvky, které jsou irelevantní nebo ne příliš často používané.
9. Smysluplné chybové hlášky

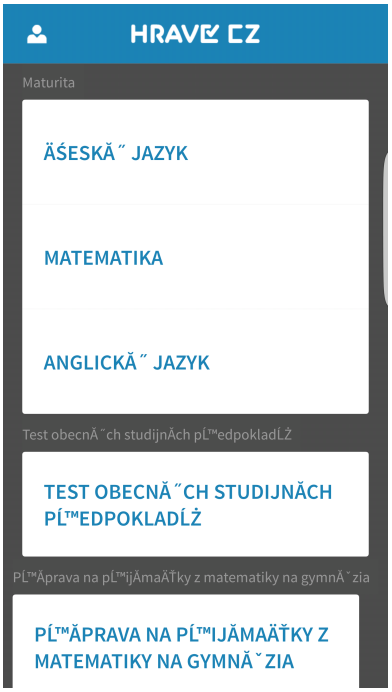
- Chybové hlášky by měly obsahovat pouze informace o problému a nabídnout řešení jak se problému vyvarovat. Neměly by obsahovat systémová hlášení.

10. Help a dokumentace
    - Aplikace by měla být natolik intuitivní, aby se uživatel obešel bez nápověd a dokumentace. Přesto je vhodné, aby dokumentace byla součástí dodávky; uživatel musí mít možnost v případě problému snadno najít řešení.

Pro každou z uvedených heuristik je pak nutné provést ohodnocení, jak aplikace danou heuristiku splňuje. Jednotlivé heuristiky jsou pak v závěru této kapitoly ohodnoceny body. Body jsou udělovány od nuly do tří, kde tři body znamenají bezproblémové splnění konkrétní heuristiky.

1.1.1 Zkoušky hravě - BETA[10]

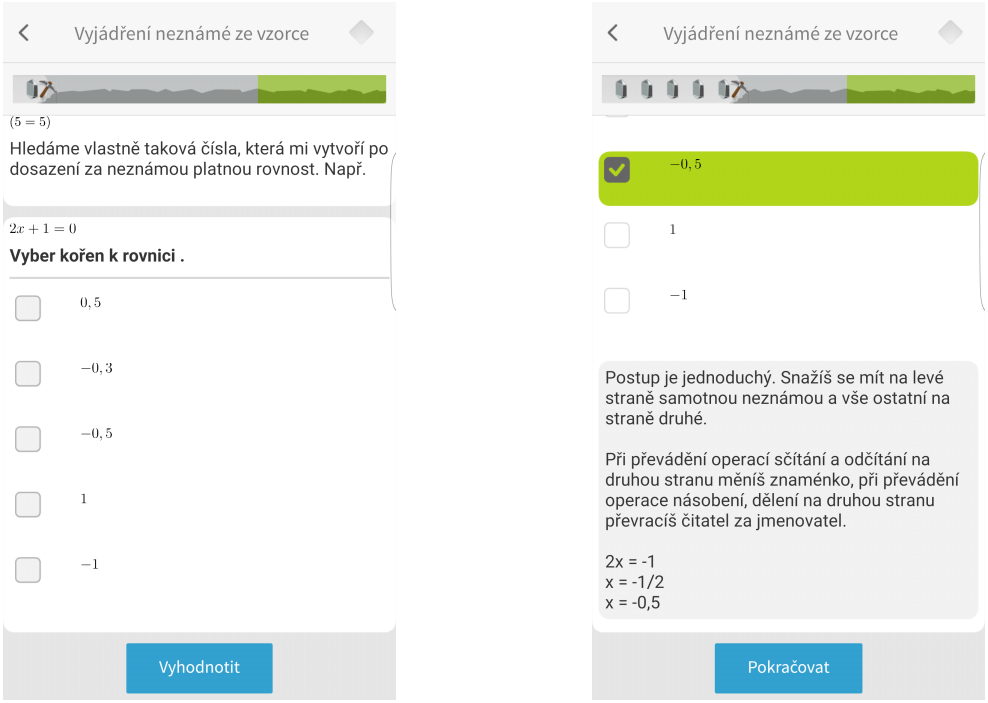
Tato aplikace je zaměřená na přípravu k maturitě a slouží uchazečům o studium na gymnáziích. Nachází se v ní cvičné testy ke státní maturitě - český jazyk, matematika a anglický jazyk. Dále obsahuje test obecných studijních předpokladů pro přípravu na NSZ (Národní srovnávací zkoušky). Poslední téma, které lze v aplikaci nalézt, je příprava na přijímací zkoušky z matematiky na gymnázia.



Obrázek 1.1: Aplikace Zkoušky hravě - BETA: menu

Uživatelské rozhraní je velmi jednoduché. Při prvním spuštění je vyžadováno přihlášení účtem, který je zaregistrován pro doplňující webovou aplikaci hrave.cz. Je však také možné se zaregistrovat přímo v mobilní aplikaci. Po úspěšném přihlášení se zobrazí menu s výběrem jednotlivých témat. Zde uživatel narazí na první velký nedostatek. Aplikace neumí v hlavním menu zobrazit českou diakritiku, přestože je lokalizována do českého jazyka. Při otevření některého z témat se již tento problém nevyskytuje.

Jednotlivá témata jsou velmi dobře zpracována a jsou rozdělena na konkrétní lekce. Ne všechny funkce aplikace jsou k dispozici zdarma, některé lekce jsou zpoplatněny. V mobilní aplikaci není možné tento zpoplatněný obsah zakoupit, nákup lze provést pouze na webové stránce. U každé lekce se nachází podrobné vysvětlení problematiky, které se týká. Za teorií následuje sada otázek, na kterou uživatel musí odpovědět. Všechny otázky jsou typu výběr jedné či více možností. Odpovědi jsou automaticky ohodnoceny. Vyhodnocení uživateli ukáže i podrobné vysvětlení, která odpověď je správně a proč.



(a) Zobrazení testu (b) Zobrazení výsledků

Obrázek 1.2: Aplikace Zkoušky hravě - BETA: test

1.1.1.1 Kladné stránky

V aplikaci jsou sady otázek definovány přímo autory. Pokud bude uživatel dobře ovládat zkušební testy zde, pravděpodobně bude velmi dobře připraven

i na naučené téma. Všechny odpovědi jsou velmi kvalitně vysvětlené.

1.1.1.2 Záporné stránky

V aplikaci není možné vytvářet vlastní testy. Tuto možnost nenabízí ani přidružená webová stránka. Testy lze stáhnout, avšak tato funkčnost není dobře zpracovaná. Bez internetu je aplikace velmi nestabilní, není tedy vhodné trénovat testy bez internetového připojení. Toto považuji za zásadní nedostatek, aplikace tak ztrácí jednu ze zásadních výhod mobilního telefonu, tedy jeho mobilitu.

1.1.1.3 Nielsenova heuristická analýza

1. Viditelnost stavu systému
   - Aplikace reaguje na uživatelské akce správně, pokud potřebuje něco načítat z internetu, ukazuje progres. Při stahování některého z témat se ukáže progres, avšak uživatel má stále možnost aplikaci ovládat. Pokud tak udělá, dochází k přerušení stahování (test není stažen, progres ale skončí jakoby stažen byl).
2. Shoda mezi systémem a realitou
   - Funkcionalita některých ikon není na první pohled zřejmá.
3. Minimální zodpovědnost a stres
   - Pokud je aplikace využívána při stabilním připojení na internet, nenachází se zde žádné problémy. K testům se lze libovolně vracet. Pokud však chce uživatel aplikaci využívat bez internetového připojení, má možnost si test stáhnout. Tento test je uložen trvale a nelze jej odstranit. Po zobrazení různých chybových hlášek z důvodu chybějícího internetového připojení aplikace často padá.
4. Shoda s použitou platformou a obecnými standardy
   - Aplikace využívá standardních prvků platformy Android.
5. Prevence chyb
   - Vzhledem k povaze otázek není možné zadat chybné hodnoty, které by způsobily nějaké problémy.
6. Kouknu a vidím
   - V horním navigačním panelu je vždy uvedeno, v jakém konkrétním tématu se uživatel nachází.
7. Flexibilita a efektivita

- Pro běžné použití aplikace (trénink testů) je vše velmi snadno přístupné. Aplikace umožňuje přidat si přátele, aby uživatel viděl jejich postup. Tato možnost je skrytá pod ikonkou v horním navigačním panelu.

8. Minimalita

   - Žádné grafické prvky v aplikaci nepřekáží pro běžné používání.

9. Smysluplné chybové hlášky

   - Pokud má uživatel nestabilní internetové připojení, aplikace začne informovat o chybách, že nelze načíst požadované testy. Chybové zprávy v podobě: „Lekci se nepodařilo načíst. IoErrorEvent type= ioError bubbles=false cancalable=false eventPhase=2 text=Error #2032 errorId=2032“ však uživateli příliš mnoho nevysvětlí.

10. Help a dokumentace

    - U všech testů jsou velmi dobře vysvětleny všechny otázky i odpovědi, i jakým způsobem by měl uživatel při učení postupovat.

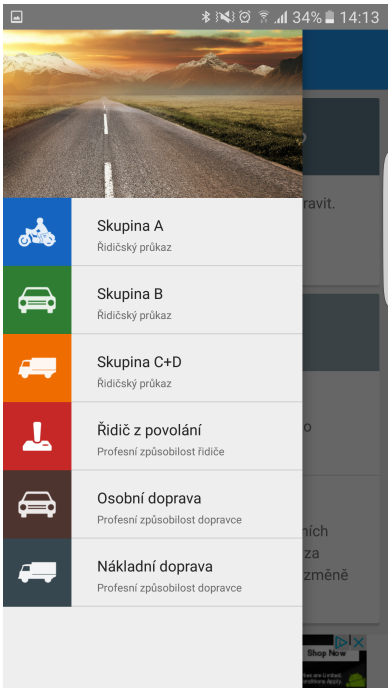
1.1.2 Autoškola[11]

Již z názvu plyne, že je tato aplikace velmi úzce zaměřena. Nabízí možnost vyzkoušet sadu oficiálních testovacích otázek při studiu teorie do autoškoly. Toto velmi úzké zaměření umožňuje, že je aplikace velmi dobře zpracována po stránce uživatelského rozhraní a celého designu.

V hlavním menu lze vybrat testy dle jednotlivých typů řidičských průkazů. Po provedení tohoto výběru je možné nastavit různé parametry pro vygenerování cvičného testu. Test lze spustit v módu, kde jsou zobrazeny správné odpovědi rovnou. Uživatel tak může procházet jednotlivé otázky a číst si vysvětlení pro odpovědi. Lze nastavit, zda jsou odpovědi vyhodnocovány po každé otázce, nebo až po celém testu dohromady. Dále lze zvolit oblast, ze které se otázky vybírají. K dispozici je sedm různých oblastí týkajících se znalostí z autoškoly. Kromě oblasti je možné pro generování otázek zvolit jednu z těchto možností:

- všechny otázky,
- nezodpovězené otázky,
- neznámé otázky,
- otázky, které umím,
- otázky, které neumím.

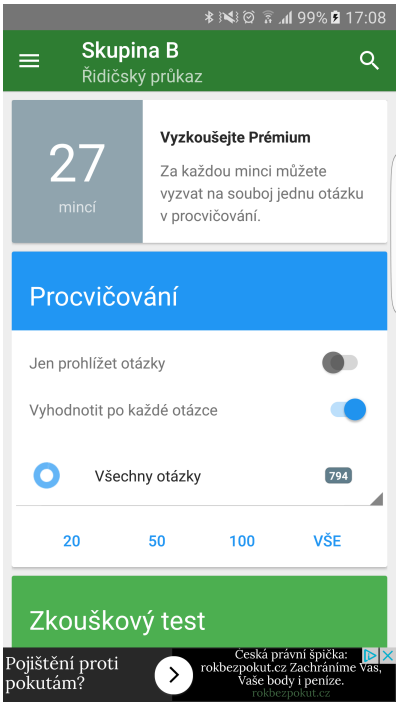
S těmito vybranými parametry pak lze spustit test buď se všemi otázkami, nebo náhodným výběrem 20, 50 nebo 100 otázek. Otázky jsou typu výběr jedné odpovědi z více možností, tedy shodný typ se skutečnými testy v autoškole.



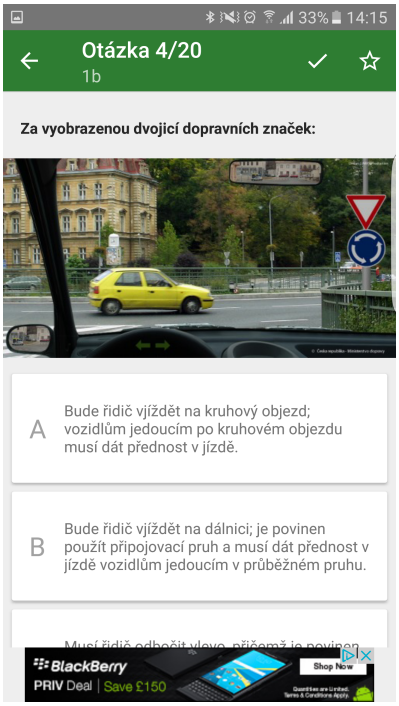
Obrázek 1.3: Autoškola - hlavní menu

Uživatelské rozhraní při testování je velmi přehledné a intuitivní. Stejně tak vyhodnocení testů je graficky dobře zpracované. Jediný nedostatek, který by se dal vytknout, je v menu. Pro spuštění testu musí uživatel skrolovat dolů, neboť pod nastavením je ještě mnoho dalších informací, než se dostane k samotnému tlačítku pro spuštění. Níže se pak nachází informace o zákonu, ke kterému se otázky vztahují. Ještě níže pod informacemi o zákoně se nachází statistika uživatele v podobě dlaždic - kolik otázek odpověděl správně, nesprávně, kolik ještě nezobrazil. Z těchto dlaždic lze spustit procvičování vybrané oblasti. Duplikují tak informace, které lze nastavit nahoře v nastavení testu.

Po celou dobu svého běhu ukazuje aplikace v dolní části banner s reklamou. Po nainstalování aplikace se v horní části zobrazí kartička s 50 mincemi. Každou vyzkoušenou otázkou se jedna mince odebere. Každou hodinu se doplní aplikace o jednu minci. Pokud si uživatel zakoupí plnou verzi, může zkoušet testové otázky bez omezení.



(a) Úvodní obrazovka



(b) Zobrazení otázek

Obrázek 1.4: Autoškola

1.1.2.1 Kladné stránky

Aplikace splňuje přesně to, co uživatel při učení na testy z autoškoly očekává. Díky pokročilému nastavení uspokojí i náročnější uživatele. Aplikace funguje plně bez připojení k internetu. Je velmi dobře udržovaná, otázky v ní jsou stále aktuální.

1.1.2.2 Záporné stránky

Navigace na hlavní stránce je kvůli velkému množství informací nepřehledná. Některá nastavení by bylo možné umístit lépe, aplikace by tak vyhověla i uživatelům, kteří nic nastavovat nepotřebují - chtějí test pouze spustit. Komplikované menu pravděpodobně také způsobuje velmi pomalý start aplikace.

1.1.2.3 Nielsenova heuristická analýza

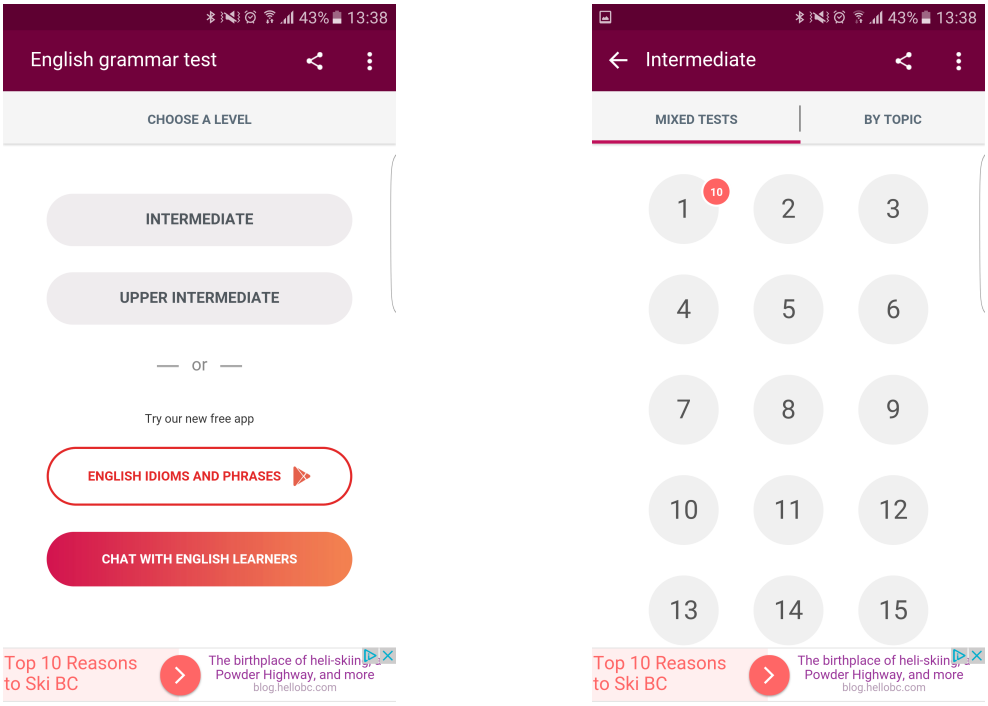
1. Viditelnost stavu systému
    - Aplikace pohotově reaguje na všechny uživatelské akce.
2. Shoda mezi systémem a realitou

- Grafika v aplikaci je zvolena dobře, vše je na první pohled jasné.

3. Minimální zodpovědnost a stres
   - Aplikace uchovává historii uživatelských odpovědí a lze se tak i zpětně vracet k těm špatně zodpovězeným.
4. Shoda s použitou platformou a obecnými standardy
   - Aplikace využívá standardních prvků OS Android.
5. Prevence chyb
   - Při zapomenutí vyplnění některých otázek je uživatel varován, jestli opravdu chce ukončit test. Pokud tento varovný dialog potvrdí, test je ohodnocen.
6. Kouknu a vidím
   - Při otevření aplikace je vždy hned jasné, kde se uživatel nachází.
7. Flexibilita a efektivita
   - Hlavní uživatelská akce v této aplikaci je procvičování otázek do autoškoly. Na hlavní stránce jsou však vidět podrobné možnosti nastavení, které většina uživatelů spíše nevyužije.
8. Minimalita
   - Při samotném testování se uživateli zobrazují správně pouze důležité informace. V hlavním menu je informací příliš, některé se snaží motivovat uživatele ke koupi premiové verze aplikace.
9. Smysluplné chybové hlášky
   - Během analýzy jsem nenarazil na žádné nesmyslné chybové hlášky.
10. Help a dokumentace
    - Vše je v aplikaci dobře vysvětlené a jednoznačné.

1.1.3 English Grammar Test[12]

Tato aplikace se zabývá výukou anglické gramatiky. Lze se v ní učit gramatiku obtížnosti intermediate nebo upper-intermediate. Po zvolení této obtížnosti se zobrazí možnost spustit jednotlivé testy. Všechny testy mají otázky typu výběr jedné správné odpovědi. Po vyplnění testu je uživateli zobrazen výsledek a nabídnuta možnost přečíst si vysvětlení k jednotlivým otázkám.



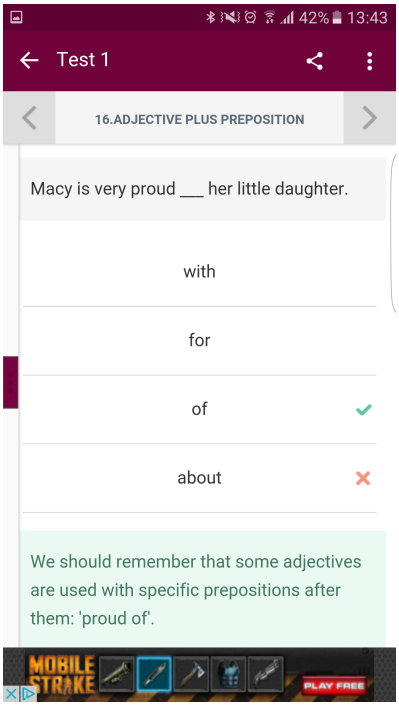
(a) Hlavní obrazovka

(b) Menu pro výběr testu

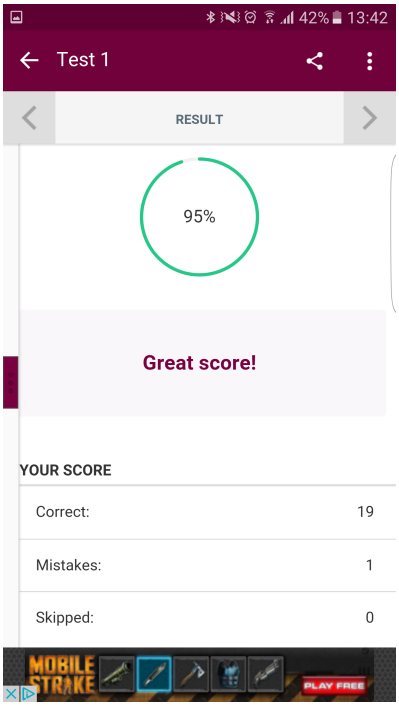
Obrázek 1.5: Aplikace English Grammer Test: navigace

Po stránce uživatelského rozhraní je aplikace velmi jednoduchá. Nepřehledný je pouze výběr jednotlivých testů. Uživatel má na výběr tlačítka s čísly 1 až 30 v každé z obou obtížností. Tlačítky se vybírá jeden z 30 předem určených testů. Pro zvýšení flexibility by aplikace mohla generovat testy náhodně z celé sady otázek, s případným zaměřením podle témat, úrovně studenta apod. Zobrazení výběru testů lze změnit. Pak jsou testy rozděleny podle témat, které se uživatel učí.

Po celou dobu běhu aplikace je v dolní části zobrazen banner s reklamou. Jednou za čas se zobrazí reklama přes celou obrazovku. Dále je zde zobrazeno mnoho reklam na různé jiné aplikace. Hned na hlavní obrazovce se autoři snaží propagovat svůj další produkt, ve kterém by se uživatel měl učit anglické idiomy a fráze. V horní liště je možné sdílet odkaz na English Grammer test v Google Play. V schovaném overflow menu se nachází reklamy na další aplikace, tentokrát třetích stran.



(a) Zobrazení otázek



(b) Zobrazení výsledků

Obrázek 1.6: Aplikace English Grammer Test: testování

1.1.3.1 Kladné stránky

Za kladné stránky považuji velké množství gramatických cvičení z anglického jazyka. Je zde celkem 60 testů, každý obsahuje 20 různých cvičení. Všechny odpovědi mají k dispozici podrobné vysvětlení.

1.1.3.2 Záporné stránky

Jak je zmíněno již výše, v aplikaci se nachází velké množství reklam. Použití se tak stává o něco méně pohodlné. Navigace by mohla být také o něco více atraktivní - například přepínání mezi jednotlivými otázkami pouze pomocí šipek v horní části obrazovky není příliš pohodlné.

1.1.3.3 Nielsenova heuristická analýza

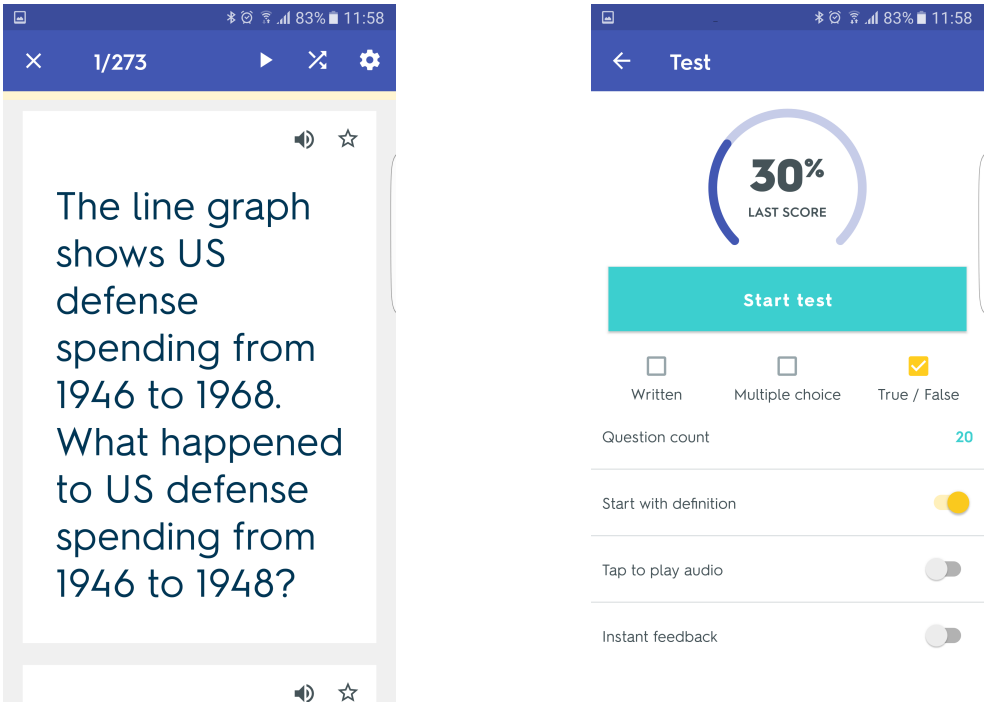
1. Viditelnost stavu systému
    - Aplikace reaguje okamžitě na všechny uživatelské akce. Otevření nastavení trvá déle. Aplikace zobrazuje progres.
2. Shoda mezi systémem a realitou

- V aplikaci je vše jasné, nepoužívá speciální systémové výrazy.

3. Minimální zodpovědnost a stres
   - Je nutné potvrdit dialog, pokud uživatel přeskočil některé z otázek a snaží se test dokončit.
4. Shoda s použitou platformou a obecnými standardy
   - Aplikace využívá standardních prvků OS Android.
5. Prevence chyb
   - V testu je možné vrátit se zpět a svou špatně zaškrtnutou odpověď opravit. Všechny testy lze spouštět znovu.
6. Kouknu a vidím
   - Vše je v aplikaci přehledné kromě výše zmíněného výběru testu z tabulky čísel.
7. Flexibilita a efektivita
   - Všechny testy jsou vytvořené předem autorem aplikace, pouští se vždy stejně. Není dostupné žádné pokročilé nastavení.
8. Minimalita
   - Uživateli se navíc ukazují reklamy a propagace na jiné aplikace vytvořené autorem. Tyto prvky kazí celkový dojem z aplikace, na druhou stranu bez těchto prvků by autor aplikace nic nevydělal.
9. Smysluplné chybové hlášky
   - Chybové hlášky v aplikaci jsou smysluplné.
10. Help a dokumentace
    - Jednoduchost aplikace nevyžaduje žádnou dodatečnou nápovědu.

1.1.4 Quizlet Learn With Flashcards[13]

Další velmi populární aplikací je Quizlet Learn With Flashcards. Oproti všem ostatním zde zmíněným aplikacím umožňuje vytvářet vlastní testovací sady ve webovém prostředí i v mobilní aplikaci. Blíží se tedy svojí funkčností Selftesteru. Pro studium využívá tzv. flashcards (kartičky, kde uživatel vidí otázku a po kliknutí se mu zobrazí správná odpověď).



(a) Zobrazení kartičky (b) Zobrazení výsledků

Obrázek 1.7: Aplikace Quizlet Learn With Flashcards: testování

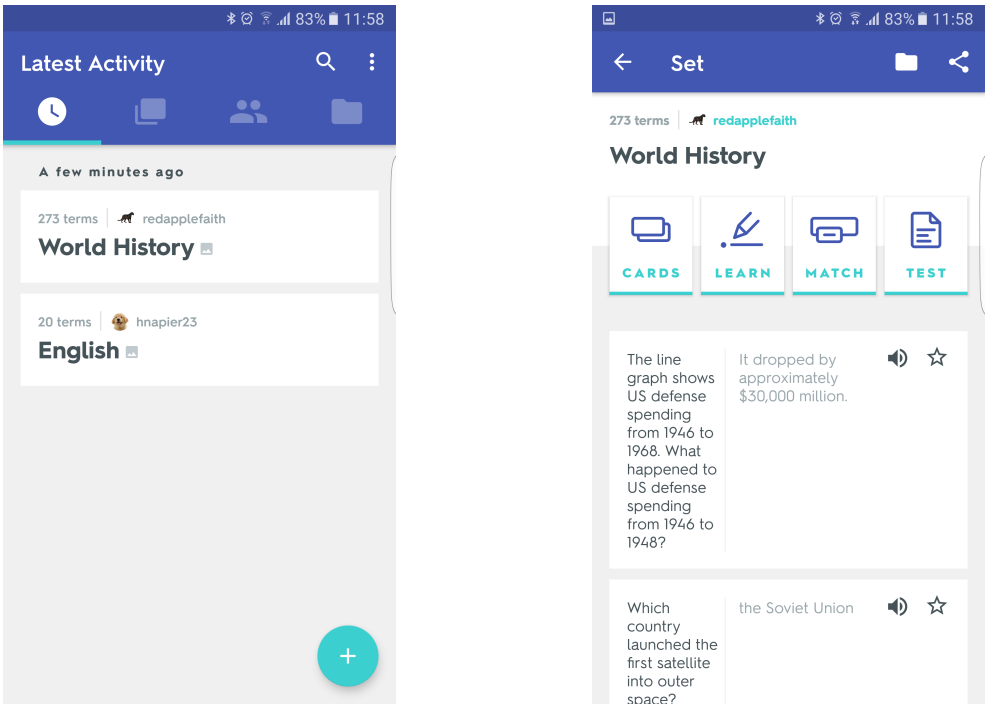
Aplikace nabízí pokročilé možnosti pro sdílení vytvořených sad kartiček. Pokud uživatel nic nezmění je tato sada vytvořena jako veřejná, dále je možné nastavit heslo pro vstup. Pro editaci je možné nastavit také heslo, kdokoliv jej zná, může tuto sadu dále upravovat. Je také možné vytvořit třídu, neboli uzavřenou skupinu uživatelů, kterým se vzájemně testovací sady sdílí.

Procházení kartiček a pročítání si správných odpovědí není jediným způsobem, který je zde využit pro rychlejší a snadnější osvojení znalostí. Druhým způsobem, je zobrazování jedné strany kartiček, kde musí uživatel odpovědi ručně psát. Dalším způsobem je zobrazení několika kartiček naráz. U každé se zobrazí obě strany a uživatel musí spojit kartičky do správných dvojic. Poslední možností je spuštění sady kartiček v podobě testu. Zde se kartičky zobrazují jako testové otázky typů:

- psaní přesné odpovědi,
- výběr správné odpovědi z několika možných,
- pravda nebo nepravda.

Pro typ otázky na výběr správné odpovědi je vždy zobrazena jedna strana kartičky jako otázka, a k ní čtyři různé odpovědi z vybrané sady, pouze jedna odpověď je správná.

Uživatelské rozhraní je velmi dobře navržené. I přes pokročilé možnosti nastavení lze vše pohodlně nalézt.



(a) Hlavní obrazovka (b) Detail testu

Obrázek 1.8: Aplikace Quizlet Learn With Flashcards: navigace

1.1.4.1 Kladné stránky

Za kladné stránky aplikace považuji možnost vytváření vlastních testovacích sad, stejně jako pokročilé nastavení jejich sdílení. V okamžiku rozšíření aplikace v rámci nějaké školy můžou studenti velmi dobře spolupracovat na vytváření dobrých studijních materiálů.

1.1.4.2 Záporné stránky

Aplikace se zaměřuje na typ učení pomocí kartiček, tvorba testů je tedy omezená na tento konkrétní typ. Pro určité testy by toto mohlo být komplikací (například pro sadu otázek, kde má každá přímo dané možnosti na výběr a uživatel musí volit z nich).

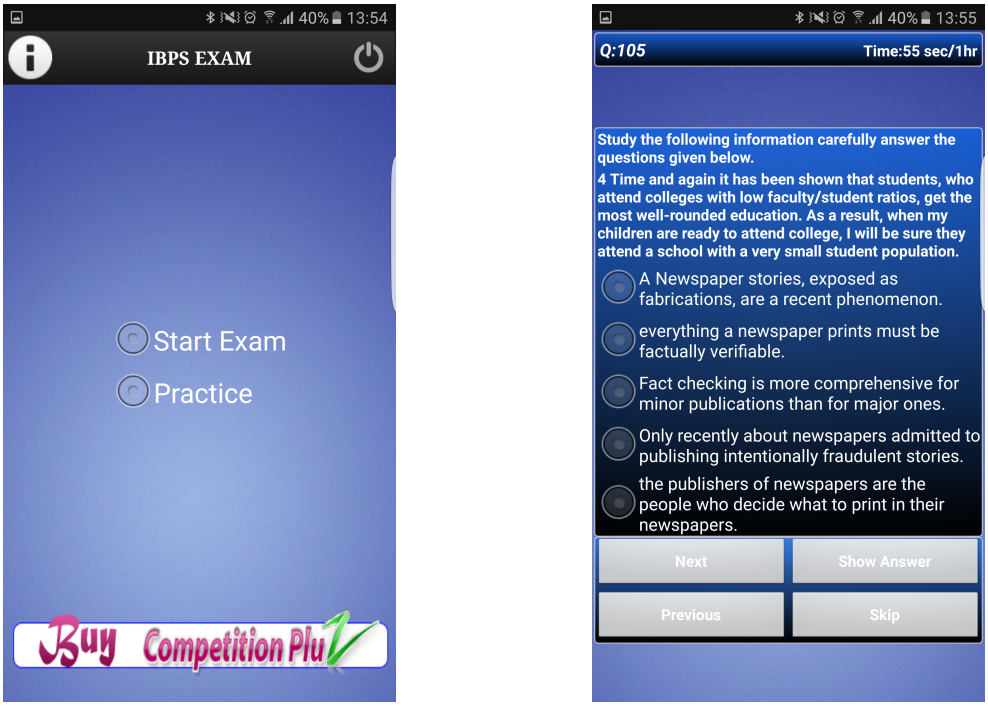
V aplikaci také chybí možnost zaslat tvůrci sady zpětnou vazbu. V případě špatné odpovědi v určité sadě otázek se tak stává pro uživatele tato sada nepoužitelnou.

1.1.4.3 Nielsenova heuristická analýza

1. Viditelnost stavu systému
   - Aplikace reaguje pohotově na jakoukoliv akci.
2. Shoda mezi systémem a realitou
   - Aplikace využívá grafických prvků velmi dobře a jen z ikonek je hned jasné co tlačítka znamenají.
3. Minimální zodpovědnost a stres
   - Aplikace ukládá kompletní historii, aby uživatel mohl trénovat jen s kartičkami, které ještě neumí. Pokud uživatel aplikaci opustí uprostřed testu a nechá ji systémem ukončit, dochází ke ztrátě postupu v testu.
4. Shoda s použitou platformou a obecnými standardy
   - Aplikace využívá nejmodernějších grafických prvků OS Android.
5. Prevence chyb
   - Při opouštění testu je zobrazen potvrzovací dialog.
6. Kouknu a vidím
   - Aplikace je velmi přehledná, ihned je jasné, kde se uživatel nachází.
7. Flexibilita a efektivita
   - Aplikace nabízí i možnost tvorby kartiček přímo v mobilním telefonu, vytváření složek a třízení jednotlivých témat. Vše dobře graficky umístěné. Uživatelům, kteří složitější funkce nevyužijí, tak ovládací prvky pro ně nepřekáží.
8. Minimalita
   - Aplikace nezobrazuje žádné rušivé elementy.
9. Smysluplné chybové hlášky
   - Všechny hlášky uživatele jsou dobře specifikované.
10. Help a dokumentace
    - Všechny položky v aplikaci jsou dobře popsané. V nastavení lze nalézt odkaz na Help sekci na webových stránkách, kde je vše dobře a podrobně popsané.

1.1.5 IBPS Exam Training[14]

Jako poslední ze zde rozebíraných aplikací byla zvolena IBPS Exam Training. IBPS je zkouška prováděna v Indii při vstupních pohovorech do jakékoliv indické banky. Cílová skupina uživatelů je tedy velmi konkrétní. Aplikace však byla zvolena z důvodu jiného zaměření než na školní znalosti, případně znalosti jazyků.



(a) Menu (b) Test

Obrázek 1.9: Aplikace IBPS Exam Training

Uživatelské rozhraní působí velmi zastarale. I vytvoření horní navigační lišty proběhlo autory bez použití standardních prvků pro OS Android. Po otevření aplikace jsou zobrazeny dvě zaškrtávací tlačítka:

- Start Exam (začít zkoušku),
- Practise (procvičovat).

Obě po zmáčknutí zobrazí sadu testovacích otázek. Rozdíl mezi spuštěním zkoušky a procvičováním najdeme v počítání skóre a v časovém limitu. U zkoušky má uživatel časový limit jedné hodiny a započítává se mu skóre, které si může kdykoliv během vyplňování testu zobrazit. Skóre započítává počet správně a špatně zodpovězených otázek z celkově vyplněného počtu otázek. U procvičování má uživatel možnost vybrat správnou odpověď nejprve sám, poté si její

správnost může ověřit. Nedostatkem této aplikace však je, že se při ověření otázek označí pouze správná odpověď a původní již není označena.

U obou variant se nachází možnost procházení otázek pomocí tlačítek Next (Další) a Previous (předchozí). Dále se zde nachází tlačítko Skip (Přeskočit), které však duplikuje funkci tlačítka Next.

Aplikace je plně zadarmo, neobsahuje žádné reklamy ani zpoplatněný obsah.

1.1.5.1 Kladné stránky

Aplikace poslouží svému účelu. Je v ní velké množství otázek na procvičení. Funguje plně offline a díky jednoduchosti je provedení libovolné akce v aplikaci velmi svižné.

1.1.5.2 Záporné stránky

Největším mínusem je jednoznačně uživatelské rozhraní (popsané problémy výše) a nestabilita. V hodnocení na Google Play si uživatelé stěžují na pády aplikace.

1.1.5.3 Nielsenova heuristická analýza

1. Viditelnost stavu systému
    - Aplikace na jakékoliv změny reaguje okamžitě, pro žádné funkce nevyžaduje internetové připojení, tedy ani neobsahuje části, kde by se muselo čekat a zobrazovat progres.
2. Shoda mezi systémem a realitou
    - Některá tlačítka v aplikaci provedou neočekávanou akci.
3. Minimální zodpovědnost a stres
    - Po spuštění testu uživatel prochází otázkami. Pokud však aplikaci opustí a systém aplikaci ukončí, okamžitý stav vyplnění testu není nikde uložen a tak dochází ke ztrátě všech vyplněných odpovědí.
4. Shoda s použitou platformou a obecnými standardy
    - Aplikace vůbec nepoužívá standardní prvky OS Android.
5. Prevence chyb
    - Vzhledem k povaze typů otázek není možné zadat nevalidní data do odpovědí.
6. Kouknu a vidím

- Nejasnost se nachází pouze v hlavním menu, kde po spuštění testu pomocí start exam nebo practice způsobí téměř to samé a tak není jasné, jak se vlastně liší.

7. Flexibilita a efektivita
   - Aplikace je jednoduchá a nenabízí žádné pokročilé nastavení.
8. Minimalita
   - Aplikace nezobrazuje žádné rušící grafické elementy.
9. Smysluplné chybové hlášky
   - Při opouštění testu je akci nutné potvrdit.
10. Help a dokumentace
    - Aplikace žádný návod nevyžaduje.

1.1.6 Srovnání aplikací

V následující tabulce jsou srovnány mezi předchozími analyzovanými aplikacemi jednotlivé klíčové vlastnosti. Dále je zde pro každou aplikaci uvedena její velikost (počet instalací) a popularita (hodnocení). Data jsou získána z obchodu Google Play.

Jednotlivé aplikace jsou označeny čísly dle popisovaného pořadí. Tyto čísla v tabulce určují sloupec, ve kterém jsou hodnoty pro danou aplikaci.

- Zkoušky hravě - BETA = 1,
- Autoškola = 2,
- English Grammar Test = 3,
- Quizlet Learn With Flashcards = 4,
- IBPS Exam Training = 5.

	1	2	3	4	5
Počet instalací	1000+	100000+	1M+	5M+	50000+
Uživatelské hodnocení	4.2	4.3	4.6	4.5	3.8
Min. verze OS Android	2.2	3.0	4.0	různé	2.0
Funguje bez internetu	Ne	Ano	Ano	Ano	Ano
Obsahuje placené funkce	Ano	Ano	Ne	Ano	Ne
Obsahuje reklamy	Ne	Ano	Ano	Ne	Ne
Vysvětlování odpovědí	Ano	Ano	Ano	Ne	Ne

Tabulka 1.1: Porovnání aplikací umožňujících testování ze znalostí

1.1.7 Vyhodnocení Nielsenovy heuristické analýzy

	Zkoušky Hravě	Autoškola	English Grammar	Quizlet	IBPS
1	2	3	2	3	3
2	2	3	3	3	1
3	1	3	3	2	0
4	3	3	3	3	0
5	3	3	3	3	3
6	2	3	2	3	2
7	3	1	1	3	3
8	3	2	2	3	3
9	0	3	3	3	3
10	3	3	3	3	3

Tabulka 1.2: Vyhodnocení Nielsenovy heuristické analýzy

V jednotlivých řádcích jsou uvedeny konkrétní body Nielsenovy heuristické analýzy.

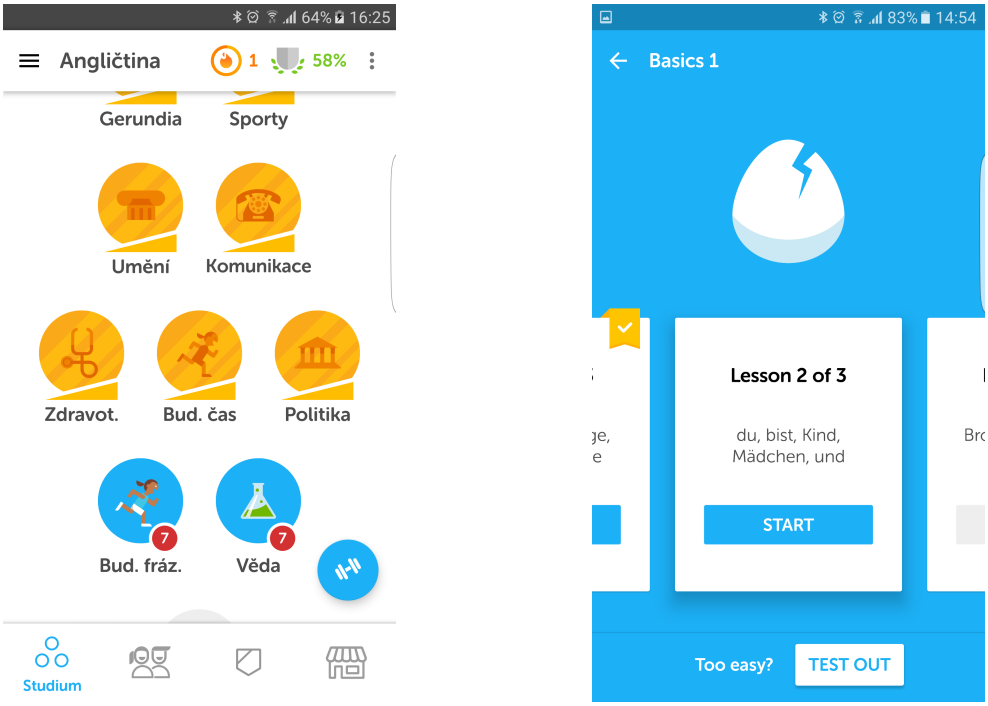
1.2 Aplikace pro získávání znalostí za pomoci herních prvků

Aby bylo získávání znalostí pomocí aplikace Selftester zábavnější, rozhodl jsem se využít herních prvků. V této části diplomové práce jsou analyzovány další nejoblíbenější aplikace, které pro vzdělávání herní prvky využívají také. I malé zapojení zábavné části učení velmi zpříjemní, což je vidět přímo na počtu instalací a hodnocení těchto aplikací. Analýza je zde zaměřena převážně na motivační část.

1.2.1 Duolingo[15]

Tato aplikace slouží k výuce cizích jazyků. Na začátku si uživatel zvolí jazyk, který chce studovat. Pokud již má nějaké znalosti, je mu nabídnut vstupní test. Podle tohoto testu aplikace rozpozná jazykovou úroveň uživatele a nechá ho přeskočit základy při učení. Duolingo je velmi propracovaná aplikace ve všech směrech. Na hodnocení a počtu instalací je to jasně vidět. Přesáhla 50 milionů instalací s průměrným hodnocením 4.7 hvězdičky.

Duolingo staví na velmi propracovaném systému otázek a odpovědí. Nachází se zde velké množství druhů úkolů, které musí uživatel splnit. Aplikace dokonce využívá i rozpoznávání zvuku, takže na některé otázky je nutné odpovědět do mikrofonu. Uživatelské rozhraní je velmi profesionálně vytvořené. Při studiu uživatel postupně prochází jednotlivými tématy, jak pro slovní zásobu, tak pro gramatiku.



(a) Hlavní menu

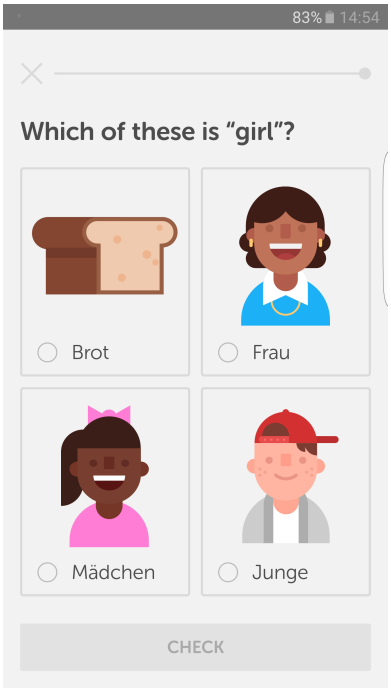
(b) Výběr lekce

Obrázek 1.10: Aplikace Duolingo: navigace

1.2.1.1 Motivační prvky

Výuka jazyků je dnes velmi důležitá, díky velmi kvalitnímu zpracování a hravému designu aplikace obdržela mnoho kladných referencí.

V průběhu učení jazyka se postupně odemykají další a těžší témata. Za každé úspěšně splněné téma uživatel získává „zkušenosti“. Má možnost se připojit k Facebooku a vidět, jak jsou na tom jeho kamarádi, kteří tuto aplikaci rovněž používají. V okamžiku, kdy některý z kamarádů získá více zkušeností, zobrazí se o tom notifikace. Díky bonusovým zkušenostem je uživatel motivován ke každodenní výuce pomocí této aplikace.



Obrázek 1.11: Duolingo: ukázka testové otázky

1.2.2 World Geography[16]

World Geography je hra, která využívá testy pro výuku zeměpisných znalostí. Obsahuje 6000 otázek týkajících se základních informací o jednotlivých zemích na světě.

V obchodě Google Play dosahuje hodnocení 4.7 hvězdičky s celkovým počtem více jak milion instalací.

V menu aplikace může uživatel přejít do následujících částí:

- hra,
- nastavení,
- nápověda.

Dále je možné se ihned z tohoto menu přihlásit pomocí Google účtu nebo Facebook účtu. Po přihlášení tak uživatel neztratí svůj postup ve hře.

Po spuštění hry je již potřeba jen zvolit světadíl, nebo i více světadílů naráz, kterého se budou týkat otázky. Uživatelské rozhraní je jednoduché a přehledné. V horní části aplikace se zobrazuje banner s reklamou. Pomocí in-app nákupů je možné tyto reklamy vypnout.

(a) Hlavní menu

(b) Herní menu

Obrázek 1.12: Aplikace World Geography: navigace

1.2.2.1 Motivační prvky

Na začátku má hráč 5 životů, je na úrovni 1 a nemá žádné zkušenosti. Každé spuštění testu způsobí ztrátu jednoho života. Jeden život se doplní jednou za 15 minut. Správně zodpovězené otázky přidávají uživateli zkušenosti, za které pak dosáhne na vyšší úrovně. Ve vyšších úrovních se mu postupně otevírají nové otázky a nový obsah v aplikaci. Hra tak dosahuje motivace se k ní stále vracet. Pokud má uživatel zájem o urychlení obnovování životů, či odemčení nějakého obsahu předem (zisk zkušeností), má možnost v obchodě v rámci aplikace pomocí in-app nákupů tyto výhody získat.

Dalším způsobem motivace v aplikaci jsou ocenění. Ocenění se získávají za správně zodpovězené otázky, splněné úkoly, výzvy (obsah, který se odemkne později). Je ale také možné získat ocenění za časté používání nápovědy. Následně jsou zde ocenění za experta na jednotlivé kontinenty. Získaná ocenění urychlují zisk zkušeností pro postup do vyšších úrovní.

Poslední motivační částí jsou celosvětové žebříčky uživatelů. Hráč vidí žebříčky top hráčů z celého světa a kromě svého umístění může bojovat i za lepší umístění země, ze které pochází. Žebříčky jsou řazeny podle získaných zkušeností ve hře.

(a) Ocenění

(b) Žebříček top hráčů ve světě

Obrázek 1.13: Aplikace World Geography: herní motivace

1.2.3 Quiz of knowledge[17]

Tato aplikace učí uživatele všeobecným znalostem. Nabízí kvízové otázky z jednotlivých kategorií znalostí (např. zeměpis, dějepis, hudba, sport, vesmír, a spoustu dalších), nebo lze vygenerovat kvízy s náhodnými kategoriemi. Po vygenerování zvoleného kvízu jsou zobrazeny otázky. Je omezen čas na odpověď. Při spuštění testu uživatel začíná se třemi životy. Ke ztrátě života dochází při špatně zodpovězené otázce. Naopak za každých pět správně zodpovězených jeden život přibývá. Snahou je dosáhnout co největšího skóre, tedy na omezený počet životů zodpovědět na co nejvíce otázek. Čím rychleji jsou otázky zodpovězeny, tím větší skóre uživatel získá.

V obchodě Google Play má tato aplikace více jak milion instalací s celkovým hodnocením 4.3 hvězdičky. Nachází se v ní přes 4000 otázek. Během používání se jednou za čas ukáže reklama přes celou obrazovku. Při průběhu testu je v dolní části obrazovky ukazován reklamní banner.

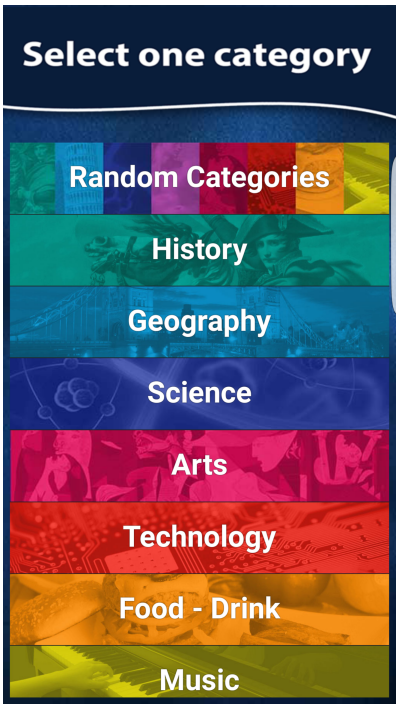
1.2.3.1 Motivační prvky

Tato aplikace používá velmi jednoduchou a přímočarou motivaci - získávání co největšího skóre. V každé z jednotlivých kategorií je motivací se stále zlepšovat

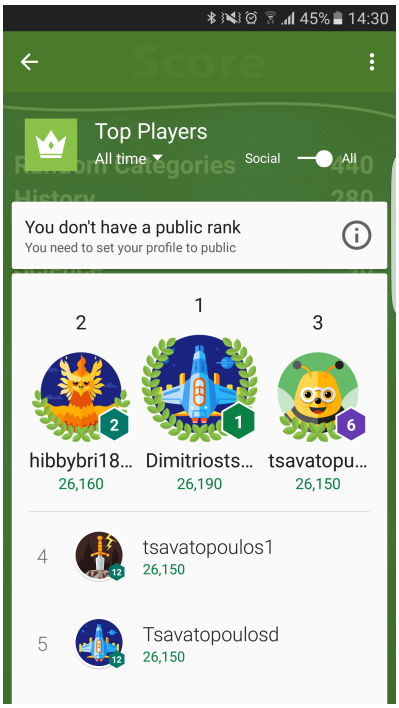
a překonávat.

Pro porovnání s ostatními hráči lze zobrazit online žebříčky. Zde se nachází sečtené skóre ze všech kategorií dohromady. Podle tohoto součtu skóre jsou hráči seřazeni buď do žebříčku celosvětového nebo jen s přáteli na Google Plus.

Autoři aplikace v popisu na Google Play uvádí, že uživatel může získávat různé ocenění, avšak tato možnost nebyla nikde nalezena.



(a) Výběr kategorie



(b) Žebříček top hráčů

Obrázek 1.14: Aplikace Quiz of knowledge

1.3 Vyhodnocení

Z výsledků hledání na trhu a analýzy aplikací podobných Selftesteru je zřejmé, že prostor pro možnost tvorby vlastních testů ještě není vůbec zaplněn. Dále je také zřejmé, že zapojení, ať už sebemenších herních prvků do aplikace způsobí větší zájem a motivaci se dále snažit učit a získávat nové znalosti. Pro Selftester je velmi důležité, aby v něm vznikalo velké množství obsahu. Pomocí výše popsaných motivačních prvků u konkurenčních aplikací je tak možné na uživatele zapůsobit, aby se zapojili a pomáhali tvořit i samotný obsah.

Kapitola 2

Metodiky vývoje aplikací

Metodiky vývoje aplikací popisují, jak vyvíjet a následně udržovat vytvářený software. Dnes již existuje mnoho různých metodik. Mezi nejznámější patří:[18]

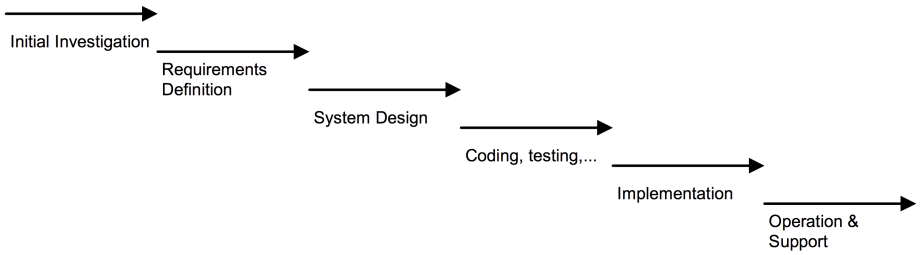
- waterfall,
- prototyping,
- incremental,
- spiral,
- RAD (Rapid Application Development),
- Extreme Programming.

Ne všechny metodiky lze použít na vývoj libovolného informačního systému. Závisí nejen na mnoha vlastnostech dané aplikace, ale i na organizačních a týmových možnostech. Každý přístup má některé výhody a nevýhody.

2.1 Waterfall[1]

Waterfall (vodopád) je sekvenční přístup k celému vývoji. Celý proces se dělí do jednotlivých fází, po kterých se postupuje. Veškeré plánování je provedeno na začátku a dále už se nic nemění. Řízení vývoje je prováděno pomocí rozsáhlých dokumentů, které vše podrobně popisují.

Tento přístup je vhodný pouze pro projekty, kde jsou předem velmi přesně definované požadavky. Velké projekty, kde se dají očekávat v průběhu externí změny, změny financí nebo rychle se měnící technologie jsou pro tento model velmi nevhodné.



Obrázek 2.1: Waterfall: fáze vývoje

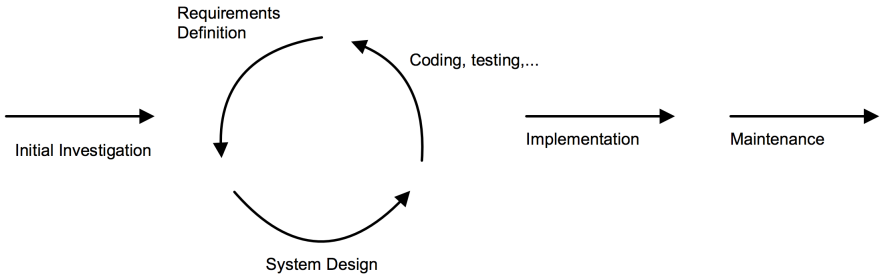
2.1.1 Výhody

Waterfall je velmi snadný na použití a je tak vhodný pro méně zkušené týmy a projektové manažery. Přísná kontrola jednotlivých kroků zajišťuje kvalitu, spolehlivost a udržitelnost vyvíjeného softwaru. Postup jednotlivých fází je dobře měřitelný.

2.1.2 Nevýhody

Model je velmi pomalý a špatně reagující na změny. Není možné se příliš vracet zpět v procesu. Při plánování nemusí být ještě plně zřejmé všechny požadavky. Výkon systému je možné měřit až po plné implementaci.

2.2 Prototyping[1]



Obrázek 2.2: Prototyping: fáze vývoje

Prototyping (prototypování) je iterativní přístup k vývoji. Dochází zde k vytváření neúplných verzí softwaru (tzv. prototypů). Snaží se snížit množství rizik pomocí rozdělení aplikace na menší části, které usnadní implementaci případných změn v softwaru. Uživatel je tak více zapojen do procesu narozdíl od modelu Waterfall, kde produkt viděl až ve finální podobě. Prototypy

jsou implementovány s očekáváním, že budou zahozeny, někdy jsou použity ve finálním produktu.

Prototypování je vhodné pro projekty, které na začátku nejsou jasně specifikované. Také se hodí pro velké projekty, kde je mnoho specifikací od různých uživatelů a je potřeba snížit rizika.

2.2.1 Výhody

Prototypování přináší možnost rychlého dodání experimentální verze aplikace a tak uživatelům usnadní lepší specifikování požadavků pro celý systém. Zlepšuje komunikaci mezi vývojem a manažery, případně zákazníkem.

2.2.2 Nevýhody

U prototypování není možné přesně dokumentovat a řídit celý proces. V průběhu se mohou požadavky velmi změnit. Rychlé prototypování jednotlivých funkcí může způsobit v budoucnu problémy, protože není čas se příliš zabývat celkovým obrazem aplikace. Prototyp může vést k mylným informacím, přes to, že vypadá jako hotový, může chybět některá ze základních funkcionalit. Budoucí rozšiřitelnost aplikace vzniklé z prototypu je velmi komplikovaná.

2.3 Incremental[1]

Tato metodika kombinuje sekvenční a iterativní přístup. Cílem je snížit rizika rozdělením aplikace na menší části. Pro tyto menší části systému jsou vytvářeny pomocí metodiky Waterfall. Nejdříve jsou specifikovány požadavky pro celkový systém a následně začne probíhat sekvenční vývoj malých vodopádů.

Inkrementální metodika je vhodná pro velké systémy, kde nejsou požadavky plně zřejmé nebo se v průběhu mění kvůli externím změnám nebo rychle se měnící technologii. Nehodí se na malé projekty.

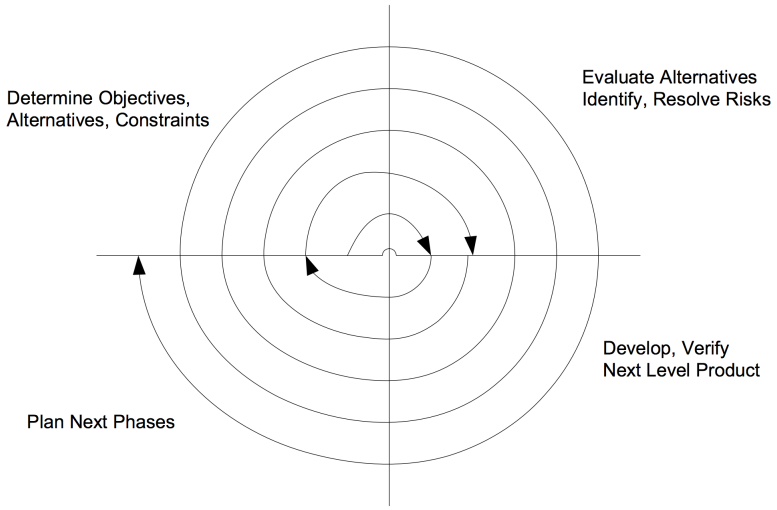
2.3.1 Výhody

Možnost zužitkovat znalosti získané v počátečním plánování v pozdějších iteracích. Je zachována kontrola nad projektem během procesu. Během životního cyklu mohou zákazníci vidět stav projektu.

2.3.2 Nevýhody

Při plánování malých vodopádů často dochází k zanedbání u propojení jednotlivých částí a aplikace jako celku. Některé části jsou implementovány dříve než jiné a tak je těžké správně navrhnout rozhraní, kterými se budou části propojovat. Složitější části jsou posouvány do pozdější implementace, aby bylo možné dříve ukazovat úspěch managementu.

2.4 Spiral[1]



Obrázek 2.3: Spiral: fáze vývoje

Spirálová metodika se obdobně jako inkrementální zaměřuje na kombinování sekvenčního a iterativního přístupu. Soustředí se na analýzu rizik a jejich minimalizaci rozdělením aplikace do menších částí. Během procesu umožňuje zanesení změn. V průběhu vývoje se stále vyhodnocují rizika a podle nich se další vývoj přizpůsobuje.

Každý cyklus spirály postupuje podle stejných kroků pro každou část aplikace:

- analýza,
- vyhodnocení, identifikace a řešení rizik,
- vývoj aplikace a kontrola výsledků,
- plánování další iterace.

Tato metodika je vhodná pro projekty, kde je nutné se co nejvíce vyhnout rizikům, nebo systémy běžící v reálném čase.

2.4.1 Výhody

Jak už bylo řečeno, spirálová metodika velmi zvyšuje šanci vyhnout se rizikům. Je užitečná pro výběr metodiky pro samotný vývoj. Umožňuje pro jednotlivé iterace využít libovolnou z již zmíněných metodik (vodopád, prototypování, iterování) a libovolně tyto metody kombinovat.

2.4.2 Nevýhody

Vyžaduje velmi zkušeného projektového manažera. V každém cyklu je nutné určovat přesné složení použitých vývojových metodik. Nic neřídí přesun do dalšího cyklu, tedy předchozí cyklus může vytvořit pro následující víc práce. Nelze určit deadline pro jednotlivé cykly.

2.5 RAD[1]

V metodice RAD (Rapid Application Development) dochází k iterativnímu vyvíjení prototypů. Cílem je rychlé dosažení kvalitních systémů při nízkých nákladech. RAD aktivně zapojuje uživatele a automatizované vývojové nástroje (např. generátory GUI, generování kódu). Nedochází k vytváření pokaždé nových prototypů, ale prototyp se upravuje a vzniká z něj postupně produkční software.

2.5.1 Výhody

Funkční verze aplikace vzniká mnohem rychleji než u vodopádu, iterativní, či spirální metodiky. Software vyvíjený pomocí RAD bývá kvůli své rychlosti levnější. Nabízí velmi blízkou interakci s uživatelem a reaguje velmi rychle na zadané změny.

2.5.2 Nevýhody

Velmi rychlý vývoj může způsobit menší celkovou kvalitu aplikace. Ve finálním produktu může být více funkcí než bylo původně zadáno. Rychlé změny v průběhu procesu mohou způsobit nekonzistenci ve finálním designu. Také další rozšiřitelnost a udržovatelnost softwaru je problematická.

2.6 Extreme Programming[2]

Metodika extrémního programování staví na základních principech vývoje a všechny osvědčené postupy žene do extrému. Extrémní programování se tak lépe přizpůsobuje změnám a dodává software vyšší kvality.

Je vyžadováno dodržovat následující principy.:

- Během celého vývoje je nutné udržovat velmi kvalitní komunikaci nejen v týmu, ale i se zákazníkem.
- Programuje se pouze to, co je zadané. Extrémní programování počítá s tím, že se kterýkoliv den mohou požadavky změnit a jakákoliv práce navíc by byla zbytečná.

- Aplikace se neustále testuje. Programátoři píší jednotkové testy, zákazník pak software kontroluje pomocí svých akceptačních testů. Stále tedy ví, v jakém stavu se aplikace nachází.
- Programátoři musí být odvážní dělat velké změny, stejně jako zahodit řešení, které nefunguje a zkusit problém vyřešit jiným způsobem.
- Programátoři nesmí vytvářet kód, který rozbije současný stav aplikace. Musí vzájemně respektovat a oceňovat kvalitu vznikajícího kódu a snažit se hledat nejlepší řešení pro všechny problémy.

2.6.1 Výhody

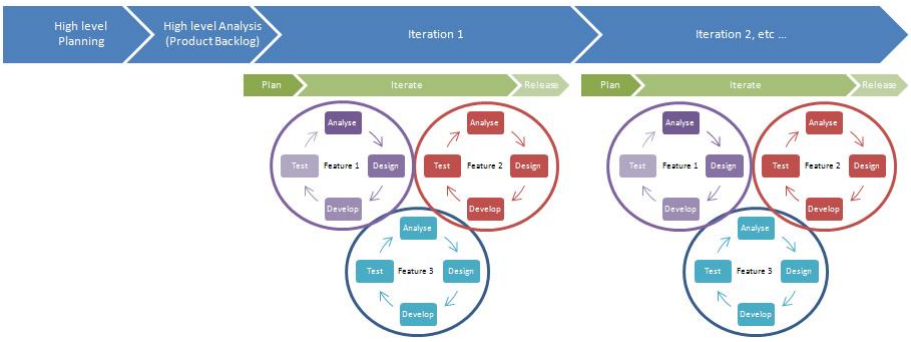
Extrémní programování je velmi přizpůsobivé změnám v průběhu celého procesu. Požadavky jsou zadávány inkrementálně a tak nemůže vzniknout špatné pochopení, či specifikování výsledného produktu.

2.6.2 Nevýhody

Extrémní programování funguje pouze v menších týmech, u větších je nutné je dále členit. Je vyžadována neustálá spolupráce zákazníka, nejlépe přímo na místě. Všichni musí navštěvovat velké množství schůzek. Chybí kvalitní dokumentace, kvůli tomu může dojít ke zvýšení rozsahu práce, než bylo původně plánováno.

2.7 Agilní vývoj

Agilní vývoj zahrnuje skupiny iteračních a inkrementálních metodik vývoje software. Umožňuje tak rychle vyvíjet software a zároveň rychle reagovat na změny v požadavcích. [3]



Obrázek 2.4: Agilní vývoj: vývojový cyklus[6]

Na začátku probíhá plánování a analýza celkového projektu. Následuje vývojový cyklus, který se skládá z jednotlivých iterací, které obvykle trvají

jeden týden až 30 dní. Na konci každé iterace by měl být tým schopný dodat nový stabilní release. Cyklus se skládá z následujících částí: analýza, návrh, implementace a testování.[6]

Mezi nejznámější patří výše zmíněné extrémní programování, dále Scrum, FDD (Feature-driven development), TDD (Test-driven development).

2.7.1 Scrum[19]

Scrum je metodika fungující na základě iterací (sprintů) o délce jeden týden až 30 dní, obvykle trvá dva týdny. Každý sprint začíná planningem, kde je určena práce na celý sprint. Každý den probíhá denní scrum (stand-up), kdy každý z členů týmu informuje ostatní o tom co dělá, jestli má nějaké problémy a podobně. Sprint končí jeho zhodnocením, kde se vyhodnotí postup. Na konci sprintu také vznikne demo, které je předvedeno zákazníkovi, který poskytne zpětnou vazbu.

2.7.2 FDD[20]

FDD (Feature-driven development) se skládá z pěti částí.

Na začátku probíhá průchod celým projektem na vysoké úrovni. Pomocí menších týmů jsou vytvořeny doménové modely pro jednotlivé modelovací části aplikace. Jsou odprezentovány a sloučeny do celkového modelu.

Druhou částí je určení jednotlivých vlastností aplikace. Celý doménový model je rozdělen do jednotlivých částí podle vlastností. Vlastnosti by neměly být časově náročnější než na dva týdny.

V třetí části se plánují vlastnosti do vývojového plánu jako třídy programátorům.

Ve čtvrté části je vytvořen návrh pro každou vlastnost. Architekt zvolí, které vlastnosti budou během následujících dvou týdnů implementovány.

V poslední části jsou implementované vlastnosti připojeny do hlavního buildu.

Vzhledem k tomu, že featury jsou poměrně malé úkoly jsou určeny tzv. milestony, které určují postup v celém procesu.

2.7.3 TDD[21]

TDD (Test-driven development) je metodika, kde se první píšou testy a následně teprve samotný program. Píše se pouze nezbytné množství kódu, který test splní.

2.8 Výběr metodiky pro mobilní aplikaci Selftester[3]

Pro dobrý rozvoj mobilních aplikací je nutné rychle reagovat na změny uživatelských potřeb, často je aktualizovat. Mobilní platformy se v současné době také velmi rozvíjí, každý rok vychází nová verze OS Android[22] a tak je pro vývoj mobilních aplikací vhodné využívat agilních metodik.

Mobilní aplikace Selftester tak bude pro svůj vývoj používat správných postupů agilních metodik.

Kapitola 3

Analýza požadavků na aplikaci

V této kapitole jsou rozebrány jednotlivé požadavky na aplikaci.

3.1 Výběr mobilní platformy

Platforma, na kterou bude mobilní klient vytvořen, byla vybrána dle celosvětových statistik rozšířenosti jednotlivých operačních systémů. Dnes je OS Android nejrozšířenější mobilní platformou, a proto byl zvolen pro klienta projektu Selftester. Následující tabulka ukazuje procentuální výskyt na trhu jednotlivých platforem po kvartálech od konce roku 2015 do třetího kvartálu roku 2016.[23]

	Android	iOS	Windows Phone	Ostatní
2015Q4	79.6%	18.7%	1.2%	0.5%
2016Q1	83.5%	15.4%	0.8%	0.4%
2016Q2	87.6%	11.7%	0.4%	0.3%
2016Q3	86.8%	12.5%	0.3%	0.4%

Tabulka 3.1: Procentuální rozdělení mobilních platforem

Dále bylo potřeba vybrat minimální verzi OS Android, která bude aplikací podporována. Tento výběr proběhl dle oficiálně uvedených statistik, kolik zařízení se na trhu v dané verzi ještě vyskytuje. Omezení je nutné provést, protože zachovávání kompatibility se staršími verzemi při implementaci způsobuje komplikace. Doporučuje se podporovat přibližně 90% zařízení na trhu.[24]

Následující tabulka ukazuje procentuální výskyt jednotlivých verzí. Selftester bude tedy podporovat OS Android verze 4.1 a výše, pokryje tak celých 97.9% zařízení na trhu.

Verze	Název	API	Procent na trhu
2.3.3 - 2.3.7	Gingerbread	10	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.1%
4.1.x - 4.3	Jelly Bean	16 - 18	11.6%
4.4	KitKat	19	22.6%
5.0 - 5.1	Lollipop	21 - 22	33.4%
6.0	Marshmallow	23	29.6%
7.0 - 7.1	Nougat	24 - 25	0.7%

Tabulka 3.2: Rozdělení verzí OS Android na trhu[7]

3.2 Funkční a nefunkční požadavky

Pro mobilní aplikaci Selftester jsou v rámci této diplomové práce zvoleny následující funkční a nefunkční požadavky.

3.2.1 Funkční požadavky

3.2.1.1 Vyhledání testu

Testy půjdou vyhledávat pouze v případě internetového připojení. Uživatel zadá libovolný řetězec, který se odešle na server, kde bude zpracován. Uživateli se zobrazí seznam testů odpovídající zadanému vyhledávacímu řetězci.

3.2.1.2 Zobrazení seznamu testů

Po prvním spuštění aplikace se uživateli zobrazí seznam testů vybraných z API. Dále bude možné volit mezi procházením všech testů, naposledy použitých testů, nebo stažených testů.

3.2.1.3 Zobrazení podrobností o testu

U každého testu musí aplikace kromě krátkého popisu v seznamu umožnit zobrazit také podrobné informace o něm. Uživateli se tak navíc zobrazí celý popis testu. Dále uvidí komentář a diskuze od ostatních uživatelů k danému tématu.

3.2.1.4 Stažení testu

Testy nebudou stahovány automaticky. Mohou obsahovat velké množství dat a tak by uživateli vyčerpaly velmi rychle datový limit. Pokud má uživatel zájem si test stáhnout tak, aby fungoval kompletně bez internetu, bude mu to umožněno jak na hlavní obrazovce v seznamu testů, tak v detailu testu.

Pro rychlý přístup test spuštěný uživatelem stáhne pouze otázky vygenerované pro uživatele, nestahuje se však celá testovací sada automaticky.

3.2.1.5 Smazání staženého testu

Stažený test zabírá místo v zařízení uživatele. Díky tomu, že test může obsahovat větší množství dat je nutné uživateli umožnit uložený test smazat pro uvolnění místa v úložišti. Mazání bude možné u stažených testů ve stejném místě jako bylo předtím možnost stažení.

3.2.1.6 Seřazení testů

Libovolný ze zobrazených seznamů testů půjde seřadit podle následujících způsobů:

- podle oblíbenosti,
- podle jména,
- podle poslední aktualizace.

3.2.1.7 Procházení správných odpovědí

Aplikace umožní procházet přímo správné odpovědi a číst si tak výsledky jednotlivých testů. Uživateli se zobrazí celá testovací sada bez náhodného promíchávání odpovědí či otázek. Zobrazí se správné odpovědi s vysvětlením, pokud je k dispozici.

3.2.1.8 Spuštění testu

Testy lze spouštět přímo po vyhledání. Stáhne se menší počet otázek (ne celá testovací sada) pro rychlý přístup. Uživatel si tak bude moc rychle a snadno test vyzkoušet. Pokud uživatel test stáhne, bude se spouštět vždy offline verze tohoto testu. Aplikace bude udržovat testy v aktuální podobě pomocí pravidelných aktualizací.

3.2.1.9 Možnost vyplnění jednotlivých typů otázek

Uživatel bude moci vyplnit jednotlivé typy otázek:

- výběr jedné správné odpovědi,
- výběr více správných odpovědí,
- zadání uživatelského textového vstupu.

Zadání uživatelského textového vstupu vyžaduje napsání přesně stejného textu jako uvádí autor testu. Pro zjednodušení psaní přesného textu na mobilním telefonu nebudou při vyhodnocování rozlišována malá a velká písmena. Dále nebude kontrolována diakritika nad písmeny.

3.2.1.10 Zobrazení výsledků

Po vyplnění libovolného testu budou zobrazeny výsledky. Aplikace zobrazí počet správných/špatných odpovědí a vypočítá výsledné procentuální skóre.

3.2.1.11 Zobrazení správných odpovědí s vysvětlením

Po zobrazení výsledků bude možné projít znovu celý test, kde uživatel uvidí špatné i správné odpovědi. Pokud se bude u otázky nacházet speciální vysvětlení, bude také zobrazeno.

3.2.1.12 Nahlášení chybné odpovědi

Při procházení správných/špatných odpovědí bude možné přímo nahlásit autorovi testu chybu. Aplikace umožní označit odpověď, o které si uživatel myslí, že je špatně, a následně přidat komentář, proč to tak je. Autorovi testu přijde notifikace s nahlášením chyby a ten pak bude moci dle svého uvážení chybu v testu opravit.

3.2.1.13 Sdílení testů

Uživateli bude umožněno jednotlivé testy nasdílet na sociální síť Facebook. Tato možnost se bude nacházet v menu v seznamu jednotlivých testů a dále v menu u zobrazení detailu testu.

3.2.1.14 Sdílení výsledků

Po dokončení testu uživatel bude moci nasdílet výsledky přímo z testu do sociální sítě Facebook. Tato možnost se bude nacházet v menu při zobrazení výsledků.

3.2.1.15 Zobrazení historie svých pokusů

Aplikace umožní prohlížet svou historii zkušebních pokusů. Na jednom místě tak uživatel uvidí přehledně svůj postup.

3.2.1.16 Ohodnocení kvality testu

Každý test, který uživatel minimálně jednou zkusil, bude možné ohodnotit. Hodnocení bude probíhat pomocí pěti hvězdiček, kde obdobně jako v Google Play je pět hvězdiček nejvíce.

3.2.1.17 Přidání komentáře k testu

Ke každému testu bude možné přidat libovolné množství komentářů. Přidávání komentářů však bude umožněno pouze přihlášeným uživatelům.

3.2.1.18 Zaslání zpětné vazby na aplikaci

Pro zlepšování aplikace bude možné zaslat přímo zpětnou vazbu na aplikaci nebo i obecně na jednotlivé testy. V nabídce budou následující tři možnosti zpětné vazby:

- chyba v testu,
- návrh na zlepšení aplikace,
- chyba v aplikaci.

Uživatel bude moci napsat zprávu a případně svůj e-mail pokud by si přál dostat odpověď. Chybu v testu bude možné nahlásit pouze pro testy, které uživatel již minimálně jednou absolvoval.

3.2.2 Nefunkční požadavky

3.2.2.1 Verze OS Android

Mobilní aplikace je omezena na OS Android verze 4.1 a vyšší z důvodů uvedených v kapitole 3.1. Na zařízeních splňující tyto požadavky musí Selftester fungovat.

3.2.2.2 Jazyk

Aplikace bude dostupná v českém a anglickém jazyce.

3.2.2.3 Google Play Services

Pro zobrazení žebříčků je využíváno Google Play Services API. Aplikace tak bude fungovat pouze na zařízeních, kde jsou Google Play Services nainstalovány. Uživatel také musí mít nainstalovanou aktuální verzi Google Play Games, pokud bude chtít vidět výsledky svých přátel, nebo všech uživatelů aplikace.

3.2.2.4 Zobrazení pouze na výšku

Aplikace bude vynucovat zobrazení pouze na výšku. Pro zachování přehlednosti v aplikaci je nutné kvůli zobrazení na šířku vytvářet jiné návrhy s jiným rozložením prvků uživatelského rozhraní na obrazovce. Dále by zobrazení na šířku v rámci celé aplikace vyžadovalo ošetřit problémy s rotací ve všech stavech aplikace.

Dle průzkumu „How Do Users Really Hold Mobile Devices?“[25] 90% uživatelů využívá svůj mobilní telefon na výšku a pouze zbylých 10% na šířku. Pro většinu uživatelů je tedy zobrazení pouze na výšku dostačující.

3.2.2.5 Internet vyžadovaný pro některé funkce

Pro následující funkce aplikace je vyžadováno internetové připojení:

- vyhledávání testů,
- stahování nových testů,
- hodnocení nebo komentování testů,
- sdílení výsledků,
- zaslání zpětné vazby aplikace.

Při používání těchto funkcí, pokud je internetové připojení nedostupné, bude aplikace uživatele informovat o problému, kvůli kterému tyto funkce nelze využít.

3.3 Herní prvky v aplikaci

Pro větší motivaci uživatelů bude aplikace využívat některých herních prvků.

V každém testu bude možné získávat medaile. Medaili uživatel získá, pokud zodpoví ve vybraném testu 100% otázek správně. Za první dosažení 100% skóre získá uživatel medaili bronzovou, za druhé stříbrnou a za třetí zlatou. 100% skóre však musí získat v řadě za sebou. Pokud při dalším pokusu získá uživatel méně než 100% ztrácí tak jakoukoliv zatím dosaženou medaili. Medaile tak uživatele motivují k velmi dobré znalosti daného tématu. Získané medaile budou vidět na první pohled jak v seznamu testů, tak v detailu u jednotlivých testů a také v historii u jednotlivých pokusů.

Dalším motivačním prvkem v aplikaci Selftester budou žebříčky uživatelů. Za každou správnou odpověď v libovolném testu dostane uživatel bod. Žebříček bude ukazovat uživatele s největším počtem získaných bodů.

Kapitola 4

Návrh

4.1 Návrh uživatelského rozhraní

Pro návrh jsem využil studia Sketch. Tento program představuje profesionální studio pro návrh uživatelského rozhraní. Pomocí dalších nástrojů umožňuje pohodlné sdílení návrhů s vývojáři a dalšími členy týmů ve firmách. V návrzích pak vývojáři vidí přímo přesné hodnoty velikostí jednotlivých prvků na obrazovce, i přesné rozložení, jak se na sebe jednotlivé prvky váží.

Ve studiu Sketch vznikla první verze návrhu (jednotlivé obrazovky jsou vidět v příloze). Podle těchto návrhů byl naimplementován prototyp aplikace. Tento prototyp však plně neodpovídá prvním návrhům, protože již při implementaci vyplynuly nejasnosti v návrhu. Tyto nejasnosti tak v prototypu byly hned opraveny. Na tomto prototypu následně proběhlo uživatelské testování.

Testeři byli vybíráni z cílové skupiny uživatelů aplikace, tedy lidé, kteří se ve svém životě setkávají s různými znalostními testy. Uživatelské testování se skládá ze tří částí:

- pre-test,
- uživatelský test,
- post-test.

V rámci pre-testu byli testeři seznámeni s aplikací a jejím zaměřením. Testeři byli také seznámeni s průběhem testu. Během celého testu je nahrávána obrazovka mobilního zařízení. Testování probíhá na konkrétním zařízení s OS Android verze 6, kvůli komplikovanější instalaci Selftesteru, stabilitě prototypu, ale i možnosti snadno nahrávat obrazovku v průběhu testu. U každého testera je uvedeno, zda již používá jinou podobnou aplikaci, a pokud ano, tak jakou.

Poté proběhlo samotné testování podle zadání uvedeného dále.

Po dokončení testování byly testerovi v rámci post-testu položeny následující otázky, které ověřovaly pocity z používání aplikace během testu.

1. Bylo používání aplikace příjemné? Ohodnoťte na stupnici 1 - 5, kde pět je nejvíce bodů.
2. Přemýšlíte, že aplikaci využijete?
3. Chybí Vám v aplikaci nějaká další funkcionalita? Jaká?

4.1.1 Zadání uživatelských testů

První dva testeři (tester 1 a tester 2) dostali následující zadání:

1. Vyhledejte test Autoškola, ten spusťte, vyplňte otázky a následně odevzdejte. Zobrazte si správné a špatné odpovědi a následně se vraťte na hlavní obrazovku.
2. Při průchodu správnými a špatnými odpověďmi z vyplněného testu se domníváte, že jedna z odpovědí je špatně označena jako správná. Nahlašte tuto odpověď autorovi testu.
3. U Vámi vybraného testu zobrazte správné řešení pro všechny otázky.
4. Vyberte si test a ten si stáhněte do offline řežimu. Následně stažený test smažte.
5. Zobrazte historii svého testování.
6. Zašlete zpětnou vazbu na aplikaci.
7. Přihlašte se na Facebook účet. Zobrazte žebříček nejlepších uživatelů aplikace.
8. Ohodnoťte libovolný test hvězdičkami, a přidejte k testu komentář.

Při průchodu tímto zadáním nemohly být některé chyby zjištěny, protože zadání příliš přesně popisovalo činnost, jakou mají testeři vykonávat. Po prvních dvou testerech bylo zadání z tohoto důvodu upraveno na následující podobu:

1. Chcete si procvičit své znalosti z testů do autoškoly. Projděte tímto testem a zjistěte své skóre.
2. Ve svých odpovědích ve vyplněném testu se domníváte, že jedna z odpovědí je špatně označena jako správná. Nahlašte tuto odpověď autorovi testu.
3. U vámi vybraného testu se podívejte na správné řešení pro všechny otázky.
4. Vyberte si test a ten si uložte, abyste ho následně mohli trénovat bez připojení k internetu. Po chvíli učení jste se rozhodli, že tento test již nebudete potřebovat, a proto ho tedy odstraňte.

5. Zobrazte historii svého testování.
6. Zašlete zpětnou vazbu na aplikaci.
7. Přihlašte se na Facebook účet. Zobrazte žebříček nejlepších uživatelů aplikace.
8. Ohodnoťte libovolný test nejlepším možným hodnocením a přidejte k testu komentář.

4.1.2 Správné řešení uživatelských testů

1. Průchod testováním
   - Uživatel otevře aplikaci a pomocí ikonky lupy zobrazí pole pro hledání. Napíše název testu, který chce vyhledat (Autoškola) a potvrdí vyhledání. Dále test spustí pomocí tlačítka uvSpustit test. Celý test vyplní a dokončí. Pro přechod mezi jednotlivými otázkami má možnost využít tlačítek předchozí/další v dolní části obrazovky, nebo pomocí swipování. Při zobrazení poslední otázky se zobrazí dokončovací tlačítko, kterým uživatel test odevzdá. Dále pomocí tlačítka „Projít odpovědi“ zobrazí své špatné a správné odpovědi v testu. Na závěr již jen dokončí kontrolu svých odpovědí a vrátí se na hlavní obrazovku.
2. Nahlášení chybné odpovědi
   - Scénář se podobá předchozímu průchodu testováním. Při procházení správných či špatných odpovědí zde uživatel musí stisknout trojtečkové menu a zvolit možnost nahlásit test. U nahlašované otázky pak zadá řešení, o kterém si myslí, že je správně a připíše komentář s odůvodněním. Dále již pokračuje stejně jako v předchozím scénáři a procházení odpovědí ukončuje a vrací se zpět na hlavní obrazovku.
3. Průchod správnými odpověďmi
   - V tomto scénáři uživateli stačí kliknout na trojtečkové menu u libovolného testu v seznamu a zvolit možnost Ukázat odpovědi. Druhou možností je otevřít detail testu pomocí kliknutí na libovolný test a zde vybrat Ukázat odpovědi. Obě tyto řešení jsou optimální.
4. Offline používání
   - Scénář musí probíhat v návaznosti na předchozí. Na obrazovce s výsledky testu si uživatel pravděpodobně mohl všimnout možnosti stažení testu. Správné řešení však není projít celý test kvůli možnosti stáhnout, ale kliknout na trojtečkové menu a vybrat stažení

testu. Při zobrazení detailu testu se nachází v horní navigační liště přímo ikonka pro stažení. Tento způsob je také v pořádku. Následné odstranění staženého testu provede uživatel stejným způsobem jako stažení (ikonka a položka v menu se pro stažené testy změní na smazání).

5. Zobrazení historie
   - Uživatel otevře menu aplikace a zde vybere položku historie.
6. Poslání zpětné vazby
   - Uživatel otevře menu aplikace a zde vybere položku zpětné vazby. Následně vyplní libovolnou zpětnou vazbu k aplikaci a tu odešle.
7. Přihlašování a účty
   - Uživatel se přihlásí nejprve pomocí Facebook účtu (otevře menu a zvolí Pokračovat přes Facebook). Je nutné potvrdit přihlášení a povolit přístup k informacím z Facebook účtu. Při dalším spuštění aplikace se uživatel odhlásí. Odhlášení je možné na stejném místě jako bylo předtím přihlášení. Odhlášení je opět nutné potvrdit v dialogu.
     Žebříčky nejlepších uživatelů aplikace tester otevře pomocí ikonky pohárku na hlavní obrazovce. Tato funkce je dostupná po přihlášení pomocí Google Games, které je provedeno automaticky při zobrazení.
8. Hodnocení testu
   - Hodnocení testu lze provést po otevření detailu testu. Zde má uživatel možnost přidělit hvězdičky. Uživatel musí přidělit nejlepší možný výsledek (tedy pět hvězdiček). Dále se pokusí přidat komentář k testu. Zde se ověřuje, že přidávání komentářů je na správném místě. Samotné přidávání komentářů není pro tento uživatelský test implementováno.

4.1.2.1 Tester 1

- Student všeobecného lékařství na 3. lékařské fakultě Karlovy Univerzity v Praze
- Nyní využívá pouze školní materiály, z aplikací pak jedině na výuku jazyků Duolingo.

Po otevření aplikace při zadání prvního scénáře došlo na chvíli k zaváhání, kde najít tlačítko pro hledání. Po krátkém prohlížení aplikace toto tlačítko bylo nalezeno a správně použito.

Další problém se vyskytl u scénáře, který ověřoval offline používání aplikace. Stažení testu proběhlo bez problémů, avšak možnost smazání testu nebyla velmi dlouho nalezena. Testera vůbec nenapadlo přejít pro tuto možnost do podrobností o testu.

4.1.2.2 Tester 2

- Lektor odpoledního programu na ZŠ Livingston
- Tester používal jedinou podobnou aplikaci - Mozkovnu.

Při průchodu testem tester nepoznal, o jaký typ otázky se jedná, jestli jde o výběr jedné správné odpovědi, nebo je možné zaškrtnout více. Dále došlo k zaváhání při odevzdávání testu. Vzhledem k tomu, že jediná možnost odevzdání testu je pomocí tlačítka, které se zobrazuje až po zobrazení poslední otázky, odevzdání proběhlo nakonec v pořádku. Dle testera by však bylo lepší tlačítko přímo s nápisem „Odevzdat“.

Tester očekával, že po kliknutí na tlačítko „Stáhnout“ test bude stažen jako nějaký dokument přímo do telefonu. Tester navrhoval přejmenovat stahování testů na ukládání.

Stejně jako u prvního testera došlo k problému u mazání stažených testů. V seznamu testů nešlo odlišit online test od testu staženého.

4.1.3 Tester 3

- Student otevřené informatiky na fakultě elektrotechnické na ČVUT v Praze
- Tester používá pouze materiály dodávané školou případně jiné materiály, které si studenti sami vytváří.

Při dokončování testu testera mátlo tlačítko další na poslední otázce, které již nespouštělo žádnou akci. Uvítal by minimálně tlačítko vizuálně označit, že již na něj nelze klikat.

Při odesílání hlášení špatných odpovědí, nebo i zpětné vazby aplikace chyběla testerovi zpráva o úspěšném, či neúspěšném odeslání.

Hodnocení testu se tester snažil provést přímo v seznamu testů pomocí kliknutí na hvězdičky. To však nelze, protože zde je zobrazen celkový průměr uživatelského hodnocení. Dle testera by bylo vhodné tento průměr alespoň barevně odlišit od místa, kde je možné test skutečně ohodnotit.

4.1.4 Tester 4

- Student obecné matematiky na matematicko-fyzikální fakultě Univerzity Karlovy v Praze

- Tester při studiu na autoškolu využíval aplikace Autoškola, dále na výuku cizího jazyka Duolingo.

Při dokončení testu (vyplnění všech otázek) tester očekával místo tlačítka další tlačítko dokončit. Poté co na toto tlačítko nešlo kliknout, tester správně zmáčkl tlačítko pro odevzdání.

K dalšímu problému došlo při stažení testu. Chyběla identifikace, který test je a není stažený a tak testerovi nebylo jasné, který test se vlastně stáhnul. Tester by očekával informaci v notifikaci nebo někde v aplikaci, že stažení testu bylo dokončeno.

Historii a také zpětnou vazbu aplikace se tester snažil hledat pod ikonkou pro řazení testů. Na poprvé neočekával menu na levé straně. Po zjištění, kde se menu skutečně nachází, již další problém nebyl.

Při hodnocení testu byl tester překvapen, že může test ohodnotit bez alespoň jednoho průchodu daným testem. Hodnocení musí tedy být uživateli umožněno až po prvním průchodu daným testem, aby bylo validní.

4.1.5 Tester 5

- Student gymnázia Špitálská
- Tester využívá aplikaci Quizlet, avšak nebaví ho nové sady kartiček vytvářet a tak využití příliš aktivní není.

Jako u předchozích testerů došlo k nedorozumění u tlačítka pro odevzdání testu.

U dalších scénářů došlo k chybnému nalezení tlačítka, které otevře menu. Ikonka tlačítka pro řazení je velmi podobná a tak došlo ke zmýlení.

Poslední nedostatek se objevil při snaze ohodnotit test. Vůbec testera nenapadlo kliknout na test, aby se zobrazil jeho detail.

4.1.6 Vyhodnocení uživatelského testování

Z uživatelského testování vyplynulo několik chyb.

1. Nejasné tlačítko pro odevzdání testu.
2. Je možné klikat na tlačítka předchozí a další pro pohyb mezi otázkami i na první popř. poslední otázce.
3. Není možné poznat, jestli je test stažený, či nikoliv.
4. Uživatelé nejsou informováni o úspěchu nebo neúspěchu po dokončení některých akcí.
5. Test je možné ohodnotit bez spuštění a tedy bez jeho znalosti.

Chyby ve výsledné aplikaci budou opraveny následujícím způsobem.

1. Tlačítko na odevzdání testu bude obsahovat nápis odevzdat místo pouze nejasné ikony.
2. Tlačítko Další na poslední otázce a Předchozí na první otázce bude vizuálně odlišeno tak, aby působilo, že je nepřístupné.
3. V seznamu testů bude na první pohled vidět, zda je test stažený, či nikoliv.
4. Při provádění akcí, které odesílají data přes internet (odeslání zpětné vazby, nahlášení chyby v testu, ohodnocení testu hvězdičkami), bude viditelné, zda proběhly v pořádku nebo ne.
5. Hodnocení testu bude možné provést až po prvním průchodu testem.

4.1.6.1 Vyhodnocení post-testu

V následující tabulce se nacházejí odpovědi od jednotlivých testerů k první a druhé otázce z post-testu.

Tester 1	Tester 2	Tester 3	Tester 4	Tester 5
4	3	4.5	4	4
ano	ano	ano	ano	ano

Tabulka 4.1: Vyhodnocení otázky 1 a 2 z post-testu

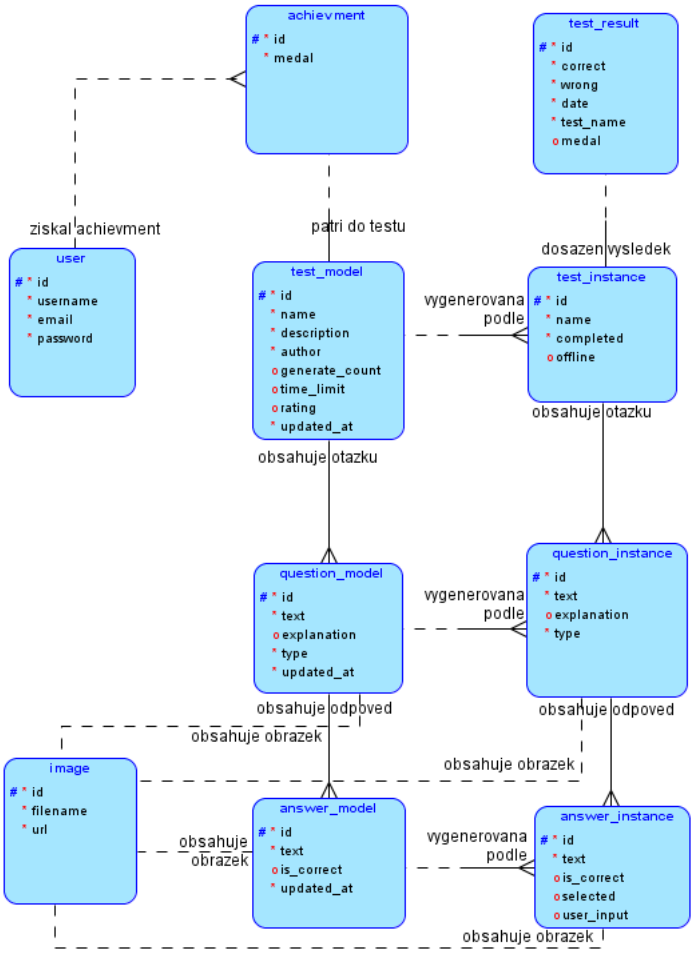
Někteří testeři měli zajímavé nápady na možné další rozšíření aplikace (otázka 3 z post-testu).

- Pro každý test by bylo možné samostatné nastavení počtu vygenerovaných otázek a vlastních časových limitů.
- Umožnit v menu předem dané vlastní filtry podle klíčových slov a možné vyhledávání v takto vyfiltrovaných testech.
- Možnost zvolit z předem definovaných kategorií (obecných témat jako matematika, historie, zeměpis, informatika, ...) ty, které mě zajímají a přímo tyto kategorie v menu zobrazovat s možností vlastního vyhledávání jen v rámci těchto zvolených kategorií.

4.2 Návrh databáze

Pro projekt Selftester byla navržena databáze, která se bude nacházet na backendu aplikace. Databáze byla implementována na backend v rámci diplomové práci Bc. Karolíny Běhalové. Z databázové struktury byla odvozena databáze

pro mobilní aplikaci, která bude uložena lokálně a umožní offline používání aplikace. Schéma databáze v mobilní aplikaci je zobrazeno na obrázku 4.1 níže.



Obrázek 4.1: Logický datový model v aplikaci

Základem datového modelu je databáze jednotlivých testů. Tabulka test_model je šablona pro generování jednotlivých testů. Obdobně jsou použity tabulky question_model (šablona pro otázky) a answer_model (šablona pro odpovědi). Tyto modely jsou udržovány ve stejné struktuře jako na backendu a jsou periodicky aktualizovány, pokud dojde k nějakým změnám od autora testu. Slouží pro uložení celých testovacích sad plně offline.

Při spuštění testu dochází k vygenerování test_instance. Tato tabulka obsahuje všechny informace o současném spuštěném testu. Naváží se na ní vybrané otázky (question_instance) s příslušnými odpověďmi (answer_instance). V těchto „instance“ tabulkách vzniknou kopie původních modelů. Při změně původního modelu tak zůstane historie testování zachována. Všechny tabulky instancí si zachovávají vazbu na jejich původní modely kvůli možnosti hlášení špatných odpovědí (je potřeba informovat, které původní otázky se daná chyba týká). Vzniklé smyčky mezi instancemi a modely testů tak problematické nejsou.

Každá z tabulek modelů a instancí se váže na tabulku image. V této tabulce jsou uloženy informace o přiložených obrázcích k jednotlivým otázkám, či odpovědím. Každá otázka nebo odpověď může obsahovat maximálně jeden unikátní obrázek.

Po odevzdání testu dochází k zápisu do tabulky test_result. Zde se uchová počet správných a špatných odpovědí a datum a čas, kdy byl test dokončen. Přesné odpovědi, které uživatel zodpověděl jsou zapsány přímo do instancí jednotlivých odpovědí. Pokud je dosaženo skóre 100%, dochází k zápisu také do tabulky achievment. Zde se ukládají získané medaile pro zvolený test_model. Pokud je získána nová medaile, je do tabulky test_result uložena stejná hodnota medaile jako do tabulky achievment. Tato hodnota však není duplicitou. V achievment tabulce je ukládána pouze současná získaná medaile pro celý test model, tedy při dalším nedosažení 100% skóre dochází k smazání hodnoty v této tabulce. Do tabulky s výsledky se tato hodnota kopíruje, aby při zobrazení historie bylo možné získat získat informace, kdy uživatel jakou medaili získal, popřípadně ztratil.

Poslední tabulkou v lokální databázi je user. Zde jsou uloženy základní údaje přihlášeného uživatele, které slouží pro autentikaci.

Lokální databáze neobsahuje tabulky, které bez připojení k internetu nemají význam - uživatelská hodnocení, komentáře a správu přístupu. V mobilní aplikaci se ukazuje pouze souhrnný průměr z uživatelských hodnocení, a tak je zde ukládána pouze tato souhrnná hodnota. Uživatelská interakce v podobě komentářů není přístupná bez připojení na internet z důvodu, aby uživatelé nereagovali na již neaktuální komentáře. Stahování všech komentářů k používaným testům by též zabíralo uživatelům zbytečnou paměť. Přístup k jednotlivým testům bude vyřešen již na backendu, a není tedy potřebné se v mobilní aplikaci touto entitou zabývat.

4.3 Návrh datových struktur testů

Pro účel prototypu, který byl používán k uživatelskému testování, byl vytvořen hrubý základ datových struktur. Existoval jeden test s otázkami, který byl uložen v paměti, nebyl však uzpůsoben tomu, aby se ukládal do databáze, či se dokonce posílal nebo přijímal ze serveru. Pro účel prototypu tato třída testu

stačila. Dále již bylo potřebné tuto datovou strukturu upravit, aby nezačala být příliš komplikovanou a nepřehlednou.

Z navržené databáze bylo nutné rozdělit test na dva možné případy: model testu a instance testu. Dále pro lokální databázi byly implementovány třídy, které obsahují atributy s anotacemi určujícími správné ukládání do lokální databáze (třídy, které končí svůj název slovem Table). Pro možnost stahovat testy ze serveru byly vytvořeny třídy, které pomocí anotací umožňovaly serializaci a deserializaci příchozích a odchozích dat (třídy začínají svůj název Serialized).

Těchto několik typů testových tříd bylo potřeba spojit dohromady. K tomu slouží v prostřední části diagramu třídy, které se používají pro zobrazení, a jsou vytvořeny buď za pomoci objektů z lokální databáze, nebo serializovaných objektů, které přijdou ze serveru. Class diagram ukazuje vzájemné vztahy mezi zmíněnými třídami. V diagramu není zobrazena většina atributů, protože přímo vycházejí z návrhu databáze a tak by tento diagram pouze komplikovaly. Zmíněný class diagram je zobrazen na obrázku „class_diagram.png“ v příloze.

4.4 Návrh architektury[4]

Pro aplikace s uživatelským rozhraním je nutné dodržovat některý z návrhových vzorů architektury aplikace. Kód je pak mnohem přehlednější. Mezi nejznámější patří:

- MVC (Model View Controller)
- MVP (Model View Presenter)
- MVVM (Model View ViewModel)

4.4.1 MVC (Model View Controller)

Tento návrhový vzor se skládá ze tří částí. Model je nezávislý na uživatelském rozhraní, řeší se v ní veškerá logika aplikace, přístup k datům. View zajišťuje vykreslování uživatelského rozhraní a informuje controller při uživatelské interakci. Controller spojuje jednotlivé komponenty dohromady. Na základě příchozích uživatelských akcí příchozích z View mění model. U Android aplikací je controller nejčastěji reprezentován aktivitou nebo fragmentem.

Tento model není pro Android aplikace příliš vhodný. Controller je velmi úzce spojen s Android API a vzniká v něm velké množství kódu, není proto příliš dobře udržovatelný.

4.4.2 MVP (Model View Presenter)

Model zde zachovává stejnou funkci jako u MVC. Aktivita/fragment se stává součástí View. Presenter je narozdíl od Controlleru u MVC na View připojený pouze pomocí rozhraní. Neměl by mít žádné závislosti na Android API.

4.4.3 MVVM (Model View ViewModel)

Model je opět stejný jako u předchozích návrhových vzorů. View se váže na hodnoty a akce poskytované ViewModelem. ViewModel tak nabízí data z modelu, které se mají zobrazit.

4.4.4 Výběr pro Selftester

Oba návrhové vzory MVP i MVVM je vhodné pro vývoj Android aplikací použít. Pro aplikaci Selftester jsem zvolil využívání vzoru MVP.

4.5 Návrh komunikace s API

Pro komunikaci s API aplikace bude využívat REST protokolu. Pro snadnou implementaci tohoto protokolu se využije knihovna Retrofit (více v kapitole 5.3.8). Na základě požadavků byly navrženy následující volání API:

- registrace a přihlášení uživatele,
- zobrazení seznamu testů,
- zobrazení podrobnostech o testu,
- stažení testu,
- vygenerování nové instance testu,
- získání instance testu,
- ohodnocení testu,
- zobrazení komentářů k testu,
- přidání komentáře k testu,
- odeslání výsledků instance testu,
- odeslání instance testu.

Kapitola 5

Implementace

5.1 Implementace pro OS Android

Android poskytuje bohatý framework pro vytváření aplikací prostřednictvím jazyka Java. Základem všech aplikací jsou následující komponenty:

- activity,
- service,
- content provider,
- broadcast receiver.

Activity odpovídá obrazovce, kterou uživatel vidí, service umožňuje práci na pozadí, content provider zprostředkovává přístup k datům a broadcast receiver přijímá oznámení. Mezi jednotlivými komponenty je nadále možné komunikovat pomocí intentů. Vytvoří se objekt intentu, který definuje zprávu, jakou komponentu aktivovat. Do této zprávy je možné přidat další libovolná data a jiné informace, které je potřebné nové komponentě předat.

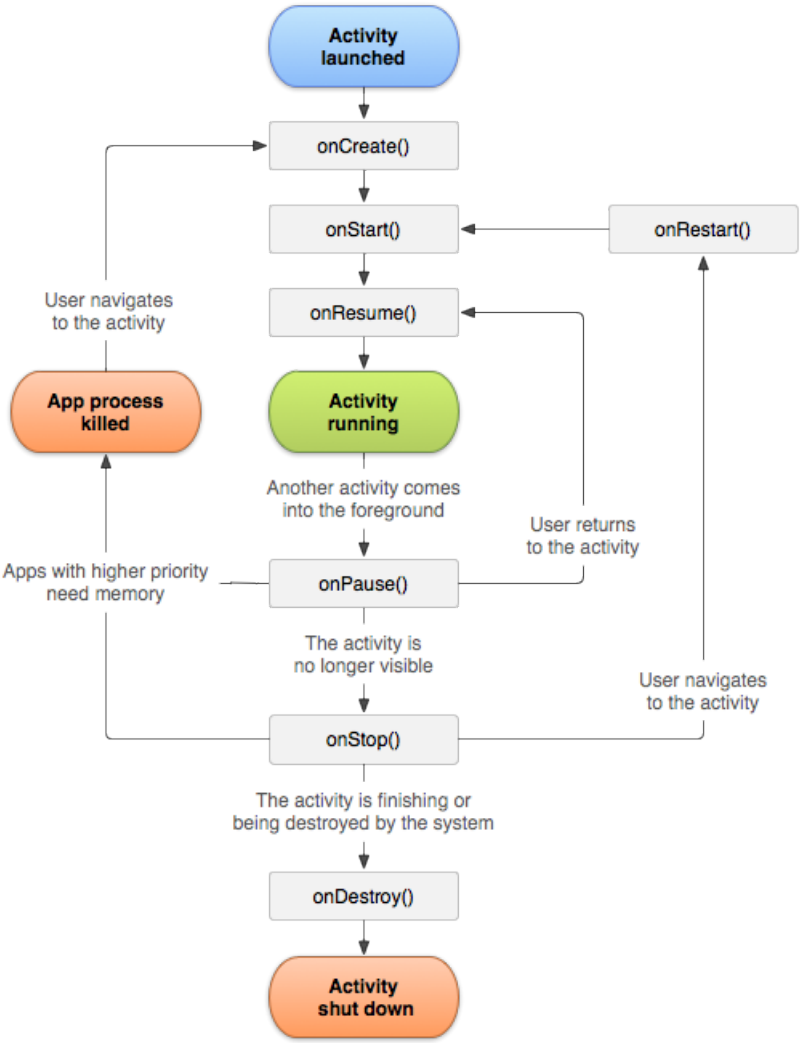
Všechny komponenty aplikace musí být definovány v souboru AndroidManifest.xml. V tomto souboru se dále definují oprávnění, které aplikace dostane, minimální API level i hardwarové a softwarové vlastnosti, jestli aplikace bude využívat bluetooth, kameru, gyroskop a podobně.[26]

5.1.1 Activity[27]

Třída aktivity je nejdůležitější komponentou Android aplikací. Způsob, jakým jsou aktivity spouštěny, je základní součástí platformy Android. Narozdíl od Javy není aplikace spouštěna v metodě main, ale prochází postupně celým životním cyklem jednotlivých aktivit. Na základě tohoto průběhu jsou volány callback funkce ke konkrétním částem životního cyklu. Jedna aktivita představuje jednu obrazovku v aplikaci. Aplikace obsahují většinou více jednotlivých

aktivit. Jedna z nich je označena jako hlavní, která se spustí při otevření aplikace. Ostatní aktivity můžou být vytvářeny a zobrazovány na základě uživatelských akcí. Jednotlivé aktivity mají na ostatních minimální závislost. Tato nezávislost umožňuje otvírat konkrétní aktivity i z jiných aplikací (například libovolná aplikace může spustit aktivitu pro sdílení obsahu na sociálních sítích v příslušných aplikacích).

5.1.1.1 Cyklus aktivit[28]



Obrázek 5.1: Životní cyklus aktivit

Při spuštění dochází postupně k volání tří hlavních metod: onCreate, onStart a onResume. Pokud začne uživatel aktivitu opouštět, jsou volány další tři me-

tody: onPause, onStop a onDestroy. Při různých situacích nejsou volány vždy všechny tyto metody. Android automaticky aktivity z paměti neodstraňuje ihned a příští start tak může být rychlejší. Uživatel uvidí aktivitu v přesně stejném stavu, v jakém ji opustil. Pokud však paměť dojde, dochází k zabíjení aktivit systémem v závislosti na stavu, v jakém se nacházely.

Při tvorbě aktivity je povinné implementovat metodu onCreate, kde dochází k nastavení základní logiky této aktivity. Z příchozího Bundle (parametr této metody) je možné získat stav, ve kterém se má aktivita nacházet. Pokud ještě neexistovala, je tento parametr null. V případě aplikace Selftester v této metodě dochází k inicializaci základního layoutu a dat potřebných při spuštění jednotlivých aktivit.

Metoda onStart je volána, když aktivita začne být viditelná uživateli. Zde dochází k nastavování uživatelského rozhraní. Po onStart je vždy volána metoda onResume.

V onResume metodě je aktivita již na popředí a viditelná pro uživatele. V tomto stavu aktivita zůstává, pokud není přerušena jinou akcí, která vezme její focus (například příchozím hovorem). Dochází k inicializaci prvků, které jsou uvolněny v metodě onPause.

Metoda onPause slouží k přerušení animací, přehrávání zvuků a jiných operací, které by ve stavu paused měli být pozastaveny, protože tato aktivita nemá focus. Tato metoda by neměla být používána pro ukládání uživatelských dat, protože je očekáván rychlý návrat zpátky do aktivity. Měla by tedy proběhnout velmi rychle.

OnStop metoda je zavolána až poté, kdy uživatel již nevidí žádné uživatelské rozhraní této aktivity. Slouží pro uvolňování dat, které nejsou potřeba, je-li je aktivita na pozadí. Tato metoda může trvat delší dobu a tak je zde možné například ukládat data do databáze a provádět jiné náročnější operace. Aktivita je následně uložena do paměti, kde drží informace o jednotlivých view a prvcích v této aktivitě. Následuje buď obnovení aktivity zavoláním onRestart a následně onStart, nebo ukončení aktivity - onDestroy.

Poslední volanou metodou je onDestroy. Zde dochází ke kompletnímu odstranění aktivity z paměti - buď příčinou ukončení pomocí zavolání odděděné metody finish, nebo pokud systému došla paměť a aktivitu musí ukončit. Uvolní se zde všechny prostředky, které nebyly uvolněny v metodě onStop.

Správné používání jednotlivých metod životního cyklu aplikace je velmi důležité pro plynulý chod aplikace při běžném používání zařízení s OS Android.

5.1.1.2 Aktivity v Selftesteru

Selftester se skládá z následujících aktivit:

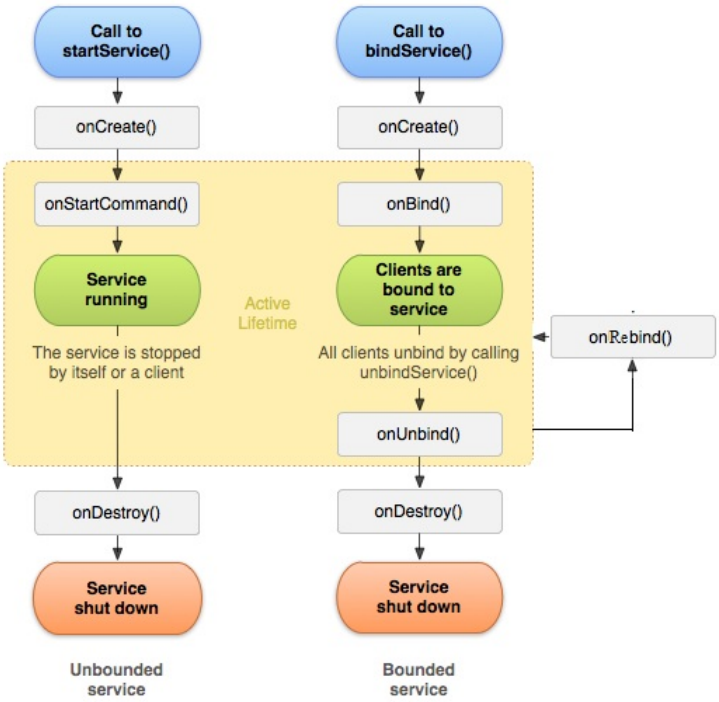
- MainActivity (hlavní obrazovka),
- TestActivity (obrazovka pro zobrazení testů),

- TestDetailActivity (obrazovka, kde jsou zobrazeny podrobné informace o testu),
- TestResultActivity (obrazovka, kde se zobrazují výsledky),
- HistoryActivity (obrazovka, kde se zobrazuje historie testování),
- FeedbackActivity (obrazovka umožňující zpětnou vazbu),
- SettingsActivity (obrazovka s nastavením).

5.1.2 Service[29]

Service je komponenta, která umožňuje provádět dlouho trvající operace na pozadí bez uživatelské interakce. Service je vhodná pro operace vyžadující internetové připojení, přehrávání hudby, nebo i náročné operace se soubory. Service má jako aktivita svůj životní cyklus.

5.1.2.1 Životní cyklus service



Obrázek 5.2: Životní cyklus service

Jak je z obrázku patrné, service může nabývat dvou různých forem.

První z nich (vlevo) je unbounded service. Tato service běží na pozadí neomezeně dlouho, není závislá na žádných dalších komponentách aplikace. Po

skončení činnosti by se sama měla ukončit. Vhodné využití pro tuto service je například stahování, či uploadování souborů přes internet. V aplikaci Selftester je využita přesně pro tento účel - stahování nových testů případně jejich aktualizací.

Druhá forma (vpravo) je bounded service. Tato service je spuštěna při připojení k jiné komponentě. Je možné připojit i více různých komponent. Service je pak ukončena v okamžiku odpojení všech připojených komponent.

5.1.2.2 Service v Selftesteru

Aplikace využívá následujících service:

- NotificationService (slouží pro příjem všech notifikací),
- UpdateService (zajišťuje aktualizaci testů pokud jsou upraveny na webu).

5.1.3 Content provider[30]

Content providery pomáhají aplikacím spravovat data. Zapouzdřují je a poskytují mechanismy k zajištění bezpečnosti. Umožňují také sdílet data mezi více aplikacemi. Pokud aplikace nesdílí data s jinými procesy, není nutné content providery implementovat.

V Aplikaci Selftester content providery implementovány nejsou. Pro přístup k datům je využita knihovna OrmLite. O řešení správy dat se tak stará tato knihovna. Selftester neumožňuje sdílet data s jinými aplikacemi, tedy v rámci této diplomové práce není potřebné se tímto sdílením zabývat.

5.1.4 Broadcast receiver[31]

Aplikace v OS Android můžou přijímat a vysílat broadcast zprávy. Posílají se, pokud nastane nějaká událost. Například OS Android vysílá broadcast zprávy o spuštění systému, připojení zařízení do nabíječky a spoustu dalších. Při implementaci aplikace se v souboru AndroidManifest.xml definuje, jaké konkrétní broadcasty chce aplikace přijímat.

5.1.4.1 Broadcast receivery v Selftesteru

V aplikaci Selftester se broadcast receivery používají pro všechny notifikace, které uživateli chodí. Jedná se o notifikaci připomínající procvičování testů a notifikaci informující o aktualizaci testů.

5.2 Implementace databáze[5]

V OS Android je nejsnazší implementovat databázi SQLite. Android pro ni nabízí přímo nativní podporu a není nutná složitá konfigurace. Stačí pouze

napsat create a update SQL skripty pro vytvoření a aktualizování databáze, o zbytek se už postará systém sám. Tato databáze podporuje pouze tři základní datové typy: TEXT (textový řetězec, podobné jako String v jazyce Java), INTEGER (celé číslo, podobné jako Long v jazyce Java) a REAL (reálné číslo, podobné jako Double v jazyce Java). Celá databáze je uložena v jednom souboru.

Pro snadnou implementaci je využita opensource knihovna OrmLite. Nebylo již ani nutné psát ručně SQL skripty pro správu tabulek v databázi. Knihovna pomocí ORM usnadní práci a stačí pouze zadefinovat přímo v datových třídách, o jaké tabulky se jedná, a které členské proměnné jsou atributy. Toto definování je prováděno pomocí anotací, kterým lze specifikovat i další parametry (jak se atribut v databázi bude nazývat, jestli se jedná o klíč a různé další). Pro tvorbu a údržbu databáze bylo potřebné implementovat třídu, která dědí od OrmLiteConfigUtil. V této třídě jsou definovány třídy, které slouží jako objekty mapované na relační tabulky. Zavoláním metody writeConfigFile se do daného konfiguračního souboru zapíše struktura databáze, která se následně vygeneruje.

5.3 Použité nástroje a knihovny

K vývoji pro OS Android je možné použít dvě různá vývojářská studia: Eclipse s ADT Pluginem a Android Studio. Pro ADT Plugin však v srpnu roku 2016 skončila podpora a tak je dnes jediné podporované studio pro vývoj nativních aplikací Android Studio. [32] Staví na velmi propracovaném vývojářském studiu IntelliJ Idea.

Pro práci s projekty využívá Gradle pluginu. Díky němu lze velmi snadno spravovat projekty, používané knihovny a další nástroje. Pro rychlejší vývoj nabízí tzv. Instant Run - možnost předat do běžící aplikace pouze nové změny bez nutnosti překompilovat celý kód.[33]

Aplikace Selftester využívá následující knihovny: Play Services, Facebook Android SDK, Firebase Core a Firebase Crash, Butterknife, OrmLite, EasyPrefs, Gson, Picasso, Retrofit a OkHttp.

Dále využívá support knihoven přímo od Googlu z důvodu zachování zpětné kompatibility se staršími verzemi OS Android.

5.3.1 Play Services

Knihovna Play Services je využívána pro přihlašování uživatelů pomocí Google účtů. Pomocí této knihovny je také aplikace připojena na Google Games API, které umožňuje zobrazení žebříčků, jak jsou uživatelé úspěšní při vyplňování testů. Díky propojení s Google Games uživatel hned vidí, jak na tom jsou jeho přátelé v síti Google Plus.

5.3.2 Facebook Android SDK

Knihovna Facebook Android SDK umožňuje přihlášení na Facebook. Nabízí tak jeden ze dvou způsobů, jakým se uživatel může přihlásit do aplikace, a vidět tak stejný obsah jak ve webové, tak mobilní aplikaci.

5.3.3 Firebase Core a Firebase Crash

Knihovny Firebase Core a Firebase Crash slouží pro získávání dat o tom, jak uživatelé aplikaci používají. Firebase Core umožňuje sběr podrobnějších dat o uživatelích než Google Play Developer Console, kde jsou vidět jen hrubá data o instalacích a odinstalacích. Firebase Core je také nezbytnou knihovnou, pokud je v aplikaci vyžívána Crash knihovna. Firebase Crash knihovna pomáhá lépe odchytit všechny pády aplikace, které se za běhu vyskytnou. O jednotlivých pádech je pomocí tohoto nástroje vidět mnohem více informací. Díky velké diverzitě zařízení s OS Android je vhodné tato data sbírat a dělat tak aplikaci stabilnější pro více uživatelů.

5.3.4 Butterknife

Pro přístup k jednotlivým view v rámci aktivit je nutné vždy podle klíče vyhledat chtěné view a následně přetypovat na konkrétní typ, jestli se jedná o textové pole, layout, nebo libovolný jiný prvek z uživatelského rozhraní. Tato knihovna velmi usnadňuje přístup pomocí anotace @BindView. Žádné přetypování již není nutné řešit.

5.3.5 OrmLite

Knihovna OrmLite slouží pro usnadnění přístupu k lokální databázi. Nabízí jednoduchou možnost, jak do databáze mapovat objekty v jazyce Java. Pomocí anotací tak lze definovat jednotlivé tabulky přímo jako třídy v kódu. Není tak nutné psát ručně komplikované SQL příkazy.

5.3.6 Gson

Gson je knihovna, která slouží pro serializaci JSON objektů. Pomocí anotace @SerializedName lze tak jednotlivé atributy tříd mapovat přímo na JSON objekt. Json objekty jsou v aplikaci používány pro komunikaci s API.

5.3.7 Picasso

Knihovna Picasso umožňuje snadno stahovat obrázky z internetu a vykreslit je až v okamžiku kdy se načtou. Knihovna také umožňuje načítání obrázků i z úložiště. Vše provádí na pozadí a nezatěžuje vlákno pro vykreslování.

5.3.8 Retrofit

Retrofit poskytuje prostředky pro připojení k Api. Komunikace probíhá pomocí REST protokolu. Jednotlivé volání HTTP metod tak lze psát přímo do rozhraní v jazyce Java. Interface v aplikaci Selftester byl naimplementován na základě navržených volání v následující podobě.

```
@POST("users")
Call<SerializedUser> createAccount
        (@Body UserRegistrationCredentials credentials);

@POST("login")
Call<SerializedUser> login
        (@Body HashMap<String, String> credentials);

@GET("testModels")
Call<ArrayList<SerializedTestModel>> getTestList
        (@QueryMap Map<String, String> options);

@GET("testModels/{id}")
Call<SerializedTestModel> getTestModelDetail
        (@Path("id") String id);

@GET("testModels/{id}/download")
Call<SerializedTestModel> downloadTestModel
        (@Path("id") String id);

@GET("tests/models/{id}")
Call<SerializedTestInstance> generateTestInstance
        (@Path("id") String testModelId);

@GET("tests/{id}")
Call<SerializedTestInstance> getTestInstance
        (@Path("id") String id);

@POST("testModels/{id}/ratings")
Call<SerializedRating> rateTest
        (@Header("Authorization") String token,
         @Body Rating rating,
         @Path("id") String testModelId);

@POST("testModels/{id}/comments")
Call<SerializedComment> addComment
        (@Header("Authorization") String token,
```

```
            @Body CommentToSend commentToSend,
            @Path("id") String testModelId);

@PUT("tests/{id}/result")
Call<SerializedTestInstance> saveTestResults
        (@Header("Authorization") String token,
         @Path("id") String id,
         @Body TestResultToSend testResultToSend);

@POST("tests")
Call<SerializedTestInstance> sendTestInstance
        (@Header("Authorization") String token,
         @Body TestInstanceToSend testInstance);
```

5.3.9 OkHttp

OkHttp spolu s knihovnou Retrofit nabízí další možnosti jak s REST API pracovat. V knihovně retrofit lze OkHttp nastavit jako klienta pro příchozí data. Pro přístup pak lze nastavit například timeout.

5.4 Rozšíření nad rámec požadované funkcionality

V mobilní aplikaci Selftester bylo implementováno několik dalších funkcí, které v zadání této diplomové práce nebyly vyžadovány. Jedná se o následující funkce:

- přihlašování přes Facebook,
- možnost použití obrázků v otázkách,
- možnost nahlásit chybné odpovědi autorovi testu.

Všechny tři funkce implementované navíc jsou závislé na dostupném API, které však v současné době tyto možnosti nepodporuje. Ve finální verzi aplikace dodávané k této diplomové práci tedy není možné je korektně zobrazovat uživatelům.

Pro přihlašování přes Facebook by bylo nezbytné, aby toto přihlašování umožňovala i webová aplikace. Ověřování uživatelů by pak probíhalo pomocí autentikačních tokenů získaných z Facebook účtů. V současné verzi aplikace je tento problém vyřešen provizorně tím, že z Facebook účtu se o uživateli získá jeho přihlašovací jméno a e-mail a následně je vyzván zadat heslo, kterým se do účtu bude přihlašovat. Účet, kterým se může uživatel přihlásit následně i do webové aplikace se tedy skládá z jména na Facebooku a hesla, které uživatel zadal.

Databáze v mobilní aplikaci i zobrazení otázek u jednotlivých testů je připraveno na podporu obrázků. Obrázky jsou stahovány na pozadí pomocí knihovny Picasso. Zobrazené obrázky lze otevřít pomocí kliknutí v dostupné galerii na zařízení. Kvůli chybějící podpoře v API však v aplikaci obrázky nalézt nelze.

Při průchodu otázkami je implemetována možnost reportovat otázky autorovi testu. Tato možnost se nachází v menu v navigační liště. Po zvolení nahlášení odpovědi má uživatel možnost zvolit odpověď, o které si myslí, že je správně, připsat komentář s důvodem a následně nahlášení odeslat. Vzhledem k tomu, že tuto funkci API nepodporuje, odeslání reportu nic neprovede.

Kapitola 6

Testování

Testování mobilních aplikací lze rozdělit na dvě části:

- lokální testy,
- instrumentační testy.

Lokální testy jsou spouštěny na lokální JVM a nemají přístup k funkcím Android API.

Instrumentační testy slouží k testování především uživatelského rozhraní. Tyto testy se spouští na Android emulátorech, nebo na skutečných zařízeních. Umožňují přímo volat metody, které vyvolávají uživatelské akce a modifikovat pole stejně jako uživatel aplikace. Pomocí těchto testů lze automatizovat testování uživatelské interakce.[34]

6.1 Lokální testy

Lokální testy ověřují správnost kódu - jednotlivé metody vrací správné hodnoty, v kódu se nenachází žádné strukturální problémy. Pro testování aplikace Selftester byl využíván nástroj Lint pro statickou analýzu kódu[35], a pro testování některých tříd unit testy napsané ve frameworku pro javu JUnit 4[36].

6.1.1 Lint

Lint je nástroj nabízený přímo v Android Studiu. Analyzuje kód aplikace a hledá potencionální chyby v kódu, možnosti optimalizace, bezpečnostní a další chyby, které se v kódu mohou nacházet a je možné je odhalit bez spouštění aplikace nebo psaní testů.

6.1.2 Unit testy

V projektu vytvořeném v Android studiu je nutné všechny Unit testy ukládat do *src/test/java/*. Při vytvoření projektu je tento adresář vytvořen automa-

ticky. Každý Unit test je vytvořený v samostatné JUnit 4 test třídě. Testy se vždy skládají ze zadání vstupních testovacích dat jednotlivým metodám a ověřují, zda metody vrátí pro tyto testovací data očekávanou hodnotu.

6.2 Testování uživatelského rozhraní

Testování uživatelského rozhraní mobilních aplikací pro OS Android je velmi komplikované. V současné době se na trhu nachází velké množství různých telefonů, které jsou různě veliké, mají různé rozlišení obrazovky a v neposlední řadě různé verze OS Android. Pro důkladné testování aplikace je vždy nutné vybrat zařízení s různými velikostmi a různými verzemi OS Android.

Pro testování je kromě skutečných mobilních telefonů možné využívat také Android emulátorů. Testování na emulátorech není příliš cenově nákladné a je velmi rychlé. Emulátory však nenabízí tolik druhů prostředí, která se vyskytují na zařízeních od jednotlivých výrobců. Není možné se spoléhat pouze na emulátory.

Aplikace Selftester byla v průběhu celého vývoje od vzniku prvního prototypu stále manuálně testována autorem na skutečném zařízení i na dostupném emulátoru. Dále na prvním prototypu byly provedeny uživatelské testy, na základě kterých došlo k úpravám aplikace. Toto uživatelské testování je podrobněji popsáno v kapitole 4.1.

Manuální testování aplikace na velkém množství různých zařízení je velmi náročné, proto je vhodné testování automatizovat. Automatizace testů však vyžaduje velké množství dalších prostředků pro vytvoření takových testů.[37]

6.2.1 Automatizace

Automatické testy musí být uloženy v adresáři *src/androidTest/java*. Gradle plugin pro Android vytvoří testovací aplikaci postavenou na kódu testů. Následně tuto aplikaci spustí na zařízení, kde je i cílová aplikace, která se má otestovat. Testovací aplikace dále simuluje uživatelskou interakci nad cílovou aplikací.

Pro automatizaci je možné využít dvou nabízených frameworků, které jsou dodávané v Android Testing Support Library (UiAutomator a Espresso).[38]

V aplikaci Selftester je napsán automatický test pouze pro základní průchod testem, tedy nejdůležitější část aplikace. Využívá frameworku UiAutomator.

6.2.1.1 UiAutomator

Pomocí frameworku UiAutomator je možné procházet nejen cílovou aplikaci, ale je možné ověřovat i interakce s aplikacemi jinými. Testy v UiAutomatoru mohou být spouštěny pro verzi OS Androidu 4.3 a vyšší. V Android SDK je dodáván nástroj UiAutomatorViewer, který umožňuje prohlížet hierarchii

jednotlivých view na obrazovce, pomocí kterých test na jednotlivé prvky přistupuje. UiAutomator však neumožňuje přistupovat na jednotlivé prvky ve webview (zobrazení webových stránek), není tedy vhodný pro testování aplikací, jejichž součástí webview jsou.[39]

6.2.1.2 Espresso

Druhý framework Espresso nabízí testování pouze cílové aplikace. Není možné interagovat s jinými aplikacemi v zařízení. Pro běh Espresso testu je nutné vypnout v aplikaci všechny animace, mohly by způsobit neočekávané problémy. Narozdíl od UiAutomatoru umožňuje testovat i webové stránky. Ty se v aplikaci Selftester nevyskytují, proto tento framework není nutné využít. [40]

6.3 Testy na větším množství uživatelů

Před vydáním hotové aplikace je možné ji testovat na větším množství uživatelů. K těmto účelům slouží alfa a beta skupina uživatelů již přímo v Google Play.

Alfa verze aplikace bývá velmi nestabilní, je proto nejprve vydávána do této velmi omezené skupiny uživatelů, od kterých se přímo očekává kvalitní zpětná vazba. Do této skupiny byla aplikace Selftester vydávána od svého prvního prototypu pro získání zpětné vazby od několika uživatelů, kteří byli ochotní aplikaci testovat. Už v průběhu vývoje tak díky této skupině byla aplikace testována na dalších zařízeních:

- Asus zenfone 4, Android 4.4
- Samsung Galaxy J5, Android 6
- Nexus 5, Android 6

Do Beta verze se aplikace vydává v okamžiku, kdy by už neměla obsahovat žádné známé chyby, avšak je potřeba získat některé kvalitativní informace. Aplikace se zde dostane na větší množství typů zařízení, na kterých se případné chyby mohou objevit.

Pro samotné vydávání se v Google Play využívá ještě dalšího kroku - stage rollout. Aplikace je první vydána pro menší množství uživatelů. Teprve po ověření, že se v ní nevyskytují žádné závažné chyby, je vydána pro všechny uživatele. Tato metoda je využívána pro rozšíření aktualizací aplikace, nikoliv pro první verze.[41]

6.4 Další možnosti testování

Následující testy nabízí další možnosti, jak mobilní aplikace testovat. V rámci této diplomové práce však nebyly implementovány. Jsou to testy, které se hodí

převážně až pro další vývoj a ne první verzi aplikace. Jediný monkey test byl na aplikaci spuštěn, avšak i tento test je vhodné zautomatizovat aby se spouštěl například pro každou další aktualizaci opakovaně.

6.4.1 Performance testy

Performance testy sledují rychlost aplikace s každou další aktualizací. V těchto testech je vhodné sledovat rychlost spuštění aplikace, rychlost načítání jednotlivých obrazovek, rychlost stahování dat ze serveru a různé další informace o výkonu, které jsou o konkrétní aplikaci zajímavé.[42]

6.4.2 Monkey testy

Pro ověření stability aplikace se používají monkey testy. Monkey je program, který na konkrétním zařízení vytváří pseudonáhodné uživatelské akce jako klikání, gesta a podobně. Při spuštění je možné nastavit následující parametry:

- počet událostí, které má test vykonat,
- omezení na konkrétní části aplikace,
- typy událostí a jejich frekvence,
- nastavení debugování.[43]

6.4.3 A/B testy

Pro validní výsledky je tyto testy možné provádět až na vydané aplikaci s větším množstvím uživatelů. Test porovnává dvě různé verze některých částí aplikace a sleduje, která z nich je lepší. Může se jednat například o dva různé texty na tlačítku. Následně vyhrává tlačítko s textem, které uživatelé používají častěji. Tyto testy se využívají hlavně pro zisk z aplikace. Tedy proto, aby se dosáhlo nejlepších výsledků a produkt si koupilo co nejvíce uživatelů.[44]

Závěr

V rámci této diplomové práce byla úspěšně navržena a následně naimplementována aplikace v OS Android Selftester. Celý projekt Selftester přináší nové možnosti jak se připravovat na libovolné znalostní zkoušky nejen studentům. Selftester velmi usnadní přípravu kdekoliv a kdykoliv.

Aplikace obsahuje všechny zadané požadavky a dokonce i některé funkce navíc. V současné době je aplikace vydána do veřejné testovací alfa skupiny na Google Play (`https://play.google.com/store/apps/details?id=cz.selftest.selftester`). Pro plnohodnotné vydání aplikace na veřejnost by bylo vhodné zrealizovat některé další kroky, které nejsou realizovány v rámci této diplomové práce. Jednalo by se především o návrh a implementaci sběru dat z používání aplikace, aby bylo možné podrobně sledovat, jak uživatelé aplikaci využívají. Na základě těchto dat by pak bylo možné aplikaci příslušně opravovat a vylepšovat. Pro vydání by taktéž bylo vhodné opravit funkce, které nefungují optimálně kvůli chybějící podpoře v API (kapitola 5.4).

Kromě zlepšování kvality používání aplikace se zde nabízí velké množství různých funkcí, které by bylo možné realizovat.

Budoucnost aplikace

Vzhledem k zájmu o podobnou aplikaci bych rád i po skončení diplomové práce tento projekt rozvíjel. Jako první by bylo potřebné dokončit a sjednotit funkce mobilní a webové aplikace, aby si plně odpovídaly a byly více propojené. Dále by po dodělání výše zmíněných nedostatků, kvůli kterým není aplikace nyní vydaná plnohodnotně v Google Play, došlo k jejímu vydání. Po získání zpětné vazby od širší skupiny uživatelů by mohlo začít docházet k pravidelným aktualizacím s novými funkcemi a opravami chyb.

Jako první rozšíření bych zvolil možnost vytvářet a následně testovat různé další typy otázek a odpovědí.

- číselná odpověď

- spojování dvojic
- komplikovaná odpověď, kterou by si uživatel sám hodnotil zda je správně či špatně obdobně jako kartičky u aplikace Quizlet Learn With Flashcards (kapitola 1.1.4) ’

Dále by bylo možné do otázek a odpovědí přidat podporu matematických formulí, aby byla aplikace dostačující i na technicky zaměřených školách, kde jsou tyto formule často nezbytné.

Dalším způsobem jak zefektivnit přípravu v této aplikaci by bylo zlepšení generátoru otázek. Uživatel by přednostně dostával otázky, které buď ještě neviděl, nebo je v předchozích pokusech zodpovědě špatně. Na tuto funkci bude aplikace již připravená, protože je ukládána kompletní historie testování a odpovědí uživatele.

Pro snazší vyhledávání testů by pak bylo vhodné implementovat některou z funkcí, které navrhovali testeři při testování uživatelského rozhraní v prototypu (kapitola 4.1.6.1) - přidávání vlastních filtrů do menu, předem definované kategorie, do kterých by testy byly při tvorbě ve webové aplikaci přiřazovány.

Literatura

[1] Centers for Medicare & Medicaid Services, Office of Information Services.: Selecting a Development Approach. [ONLINE]. 2008. Dostupné z: https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf

[2] University of San Francisco.: Extreme Programming. [ONLINE]. Dostupné z: http://www.cs.usfca.edu/~parrt/course/601/lectures/xp.html

[3] Andrei Cristian Spataru.: Agile Development Methods for Mobile Applications. [ONLINE]. 2010. Dostupné z: https://www.inf.ed.ac.uk/publications/thesis/online/IM100767.pdf

[4] Eric Maxwell.: MVC vs. MVP vs. MVVM on Android. [ONLINE]. 2017. Dostupné z: https://realm.io/news/eric-maxwell-mvc-mvp-and-mvvm-on-android/

[5] Lars Vogel.: Android SQLite database and content provider - Tutorial. [ONLINE]. 2014. Dostupné z: http://www.vogella.com/tutorials/AndroidSQLite/article.html

[6] Kelly Waters.: Agile Development Cycle. [ONLINE]. 2011. Dostupné z: http://www.allaboutagile.com/agile-development-cycle/

[7] Google, Inc.: Platform Versions. [ONLINE]. Dostupné z: https://developer.android.com/about/dashboards/index.html

[8] Bc. Monika Černá, Mgr. Michal Černý.: Metody efektivního čtení. [ONLINE]. Dostupné z: http://clanky.rvp.cz/clanek/c/G/14531/metody-efektivniho-cteni.html/

[9] Jakob Nielsen.: 10 Usability Heuristics for User Interface Design. [ONLINE]. 1995. Dostupné z: https://www.nngroup.com/articles/ten-usability-heuristics/

[10] Educasoft s.r.o.: Zkoušky Hravě - BETA. [ONLINE]. Dostupné z: https://play.google.com/store/apps/details?id=air.cz.educasoft.hravemobile

[11] James Deer.: Autoškola 2017. [ONLINE]. Dostupné z: https://play.google.com/store/apps/details?id=cz.jamesdeer.free.autotest

[12] SevenLynx.: English Grammar Test. [ONLINE]. Dostupné z: https://play.google.com/store/apps/details?id=english.grammar.test.app

[13] Quizlet LLC.: Quizlet Learn With Flashcards. [ONLINE]. Dostupné z: https://play.google.com/store/apps/details?id=com.quizlet.quizletandroid&rdid=com.quizlet.quizletandroid

[14] Appsoftindia.: IBPS Exam Training. [ONLINE]. Dostupné z: https://play.google.com/store/apps/details?id=com.zayanInfotech.IbpsExam

[15] Duolingo.: Duolingo: Learn Languages Free. [ONLINE]. Dostupné z: https://play.google.com/store/apps/details?id=com.duolingo

[16] Atom Games Ent.: World Geography - Quiz Game. [ONLINE]. Dostupné z: https://play.google.com/store/apps/details?id=com.age.wgg.appspot

[17] educ8s.com.: Quiz of Knowledge - Free game. [ONLINE]. Dostupné z: https://play.google.com/store/apps/details?id=com.educ8s.triviaquiz2015

[18] Wikimedia Foundation, Inc.: Software development process. [ONLINE]. Dostupné z: https://en.wikipedia.org/wiki/Software_development_process

[19] *The Scrum Culture*. Springer, 2015.

[20] *A Practical Guide to Feature-driven Development*. Prentice Hall PTR, 2002.

[21] *Test Driven Development: By Example*. 2002.

[22] Jerry Hildenbrand.: Inside the different Android Versions. [ONLINE]. 2016. Dostupné z: http://www.androidcentral.com/android-versions

[23] IDC.: Smartphone OS Market Share, 2016 Q3. [ONLINE]. 2016. Dostupné z: http://www.idc.com/promo/smartphone-market-share/os;jsessionid=EEEC88027847B9748BDD39E0219F63EC

[24] Google, Inc.: Supporting Different Platform Versions. [ONLINE]. Dostupné z: https://developer.android.com/training/basics/supporting-devices/platforms.html

[25] Steven Hoober: How Do Users Really Hold Mobile Devices?. [ONLINE]. 2013. Dostupné z: http://www.uxmatters.com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices.php

[26] Google, Inc.: Application Fundamentals. [ONLINE]. Dostupné z: https://developer.android.com/guide/components/fundamentals.html

[27] Google, Inc.: Introduction to Activities. [ONLINE]. Dostupné z: https://developer.android.com/guide/components/activities/intro-activities.html

[28] Google, Inc.: The Activity Lifecycle. [ONLINE]. Dostupné z: https://developer.android.com/guide/components/activities/activity-lifecycle.html

[29] Google, Inc.: Services. [ONLINE]. Dostupné z: https://developer.android.com/guide/components/services.html

[30] Google, Inc.: Content Providers. [ONLINE]. Dostupné z: https://developer.android.com/guide/topics/providers/content-providers.html

[31] Google, Inc.: Broadcasts. [ONLINE]. Dostupné z: https://developer.android.com/guide/components/broadcasts.html

[32] Google, Inc.: ADT Plugin (DEPRECATED). [ONLINE]. Dostupné z: https://developer.android.com/studio/tools/sdk/eclipse-adt.html

[33] Google, Inc.: Meet Android Studio. [ONLINE]. Dostupné z: https://developer.android.com/studio/intro/index.html

[34] Google, Inc.: Getting Started with Testing. [ONLINE]. Dostupné z: https://developer.android.com/training/testing/start/index.html

[35] Google, Inc.: Improve Your Code with Lint. [ONLINE]. Dostupné z: https://developer.android.com/studio/write/lint.html

[36] Google, Inc.: Building Local Unit Tests. [ONLINE]. Dostupné z: https://developer.android.com/training/testing/unit-testing/local-unit-tests.html

[37] Jaspreet Singh.: 7 Ways To Win At Mobile Application Testing. [ONLINE]. 2016. Dostupné z: https://www.axelerant.com/blog/7-ways-win-mobile-application-testing

[38] Google, Inc.: Automating User Interface Tests. [ONLINE]. Dostupné z: https://developer.android.com/training/testing/ui-testing/index.html

[39] Google, Inc.: Testing UI for Multiple Apps. [ONLINE]. Dostupné z: https://developer.android.com/training/testing/ui-testing/uiautomator-testing.html

[40] Google, Inc.: Testing UI for a Single App. [ONLINE]. Dostupné z: https://developer.android.com/training/testing/ui-testing/espresso-testing.html

[41] Google, Inc.: Launch. [ONLINE]. Dostupné z: https://developer.android.com/distribute/best-practices/launch/index.html

[42] Developer.com Staff.: Android App Performance Testing: An End-to-end Approach. [ONLINE]. 2012. Dostupné z: http://www.developer.com/ws/android/development-tools/android-app-performance-testing-an-end-to-end-approach.html

[43] Google, Inc.: UI/Application Exerciser Monkey. [ONLINE]. Dostupné z: https://developer.android.com/studio/test/monkey.html

[44] VWO.: The Complete Guide to A/B Testing. [ONLINE]. Dostupné z: https://vwo.com/ab-testing/

Příloha **A**

Seznam použitých zkratek

PQRST Preview, Question, Read, Summary, Test

RAD Rapid Application Development

GUI Graphical User Interface

FDD Feature-driven development

TDD Test-driven development

MVC Model View Controller

MVP Model View Presenter

MVVM Model View ViewModel

JVM Java Virtual Machine

API Application Programming Interface

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `apk` ..... adresář s instalačním balíčkem aplikace
- `uzivatelske testovani`
  - `videa` ..... adresář s videi z testování
  - `prototyp.apk` ..... prototyp aplikace pro uživatelské testy
- `design` ..... adresář s návrhem uživatelského rozhraní
- `src`
  - `android` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `thesis.pdf` ..... text práce ve formátu PDF
- `class_diagram.png` ..... class diagram z kapitoly 4.3