FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Dráček IV - Uživatelské rozhraní
Student:	Dominik Sivák
Vedoucí:	Ing. Jiří Chludil
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Dráček je vzdělávací aplikace využívající mobilní platformou Android. Jejím cílem je výuka a řešení problémů zábavnou formou. Tato práce navazuje na bakalářské práce z třech předchozích verzí.
1) Podrobte aplikaci uživatelskému testovaní s cílovou skupinou a výsledky vyhodnoťte.
2) Analyzujte chyby a nedostatky v uživatelském rozhraní s ohledem na uživatelskou přívětivost (UX).
3) Navrhněte metodiku řešící spolupráci s designéry, kteří nemají zkušenosti s platformou Android.
4) Pomocí metod softwarového inženýrství navrhněte novou podobu chování grafického rozhraní aplikace a to s ohledem na design dodaný spolupracující designérkou.
5) Implementujte navržené změny s ohledem na vícenásobné použití navržených komponent.
6) Hotové řešení podrobte uživatelským testům v UX laboratoři a vyhodnoťte výsledky.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 6. února 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalárska práca

Dráček IV – Uživatelské rozhraní

Dominik Sivák

Katedra softwarového inženýrství
Vedúci práce: Ing. Jiří Chludil

14. mája 2018

Poďakovanie

Chcel by som poďakovať vedúcemu mojej práce Ing. Jiřímu Chludilovi za priateľský prístup, pohodové vedenie práce a možnosť zúčastniť sa na projekte. Ďalej sa chcem poďakovať všetkým členom tímu Dráček, s ktorými som pracoval za ochotu a príjemnú spoluprácu; a priateľom, ktorí aj v čiernobielych časoch udržovali svet farebným. Špeciálne poďakovanie venujem svojej rodine za neustálu podporu počas celej dĺžky štúdia.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 14. mája 2018 .....

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2018 Dominik Sivák. Všetky práva vyhradené.
Táto práca vznikla ako školské dielo na FIT ČVUT v Prahe. Práca je chránená medzinárodnými predpismi a zmluvami o autorskom práve a právach súvisiacich s autorským právom. Na jej využitie, s výnimkou bezplatných zákonných licencií, je nutný súhlas autora.

Odkaz na túto prácu

Sivák, Dominik. *Dráček IV – Uživatelské rozhraní*. Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Táto bakalárska práca sa zaoberá analýzou, návrhom, implementáciou a testovaním úprav aplikácie, ktorá slúži ako jadro v modulárnom systéme aplikácií. Táto aplikácia je súčasťou projektu Dráček, ktorého cieľ je poskytnúť deťom v školskom a predškolskom veku učenie pomocou hry na platforme Android. V tejto práci najprv zisťujem stav a kvalitu produktov predchádzajúcich iterácií a venujem sa ich nedostatkom. Práca pokračuje návrhovou časťou, v ktorej sa nachádzajú nápady na vylepšenie aplikácie a stratégie, ako tieto nápady realizovať. Tieto nápady sú následne realizované a popísané v implementačnej časti spolu s príručkami ku produktu. Poslednou časťou je testovanie, v ktorej opisujem aké testy boli po realizácii zmien vykonané.

Klíčová slova mobilná aplikácia pre výuku detí, Android aplikácia, škola hrou, jadro aplikácie

Abstract

This thesis captures analysis, design, implementation and testing of changes made in application serving as core unit in modular system of applications. This application is a part of the project called Dráček (Dragon), which aims to deliver Android-based learning through play game. The thesis begins with

evaluation of products created by previous iterations of this project. The thesis then continues with the design phase, in which I mention the ideas of possible improvements along with strategies regarding their implementation. These ideas are then implemented in the implementation phase, which also contains product manuals. The last phase is testing, in which I describe tests that were executed after the implementation.

Keywords mobile application for children education, Android application, learning through play, application core

Obsah

Zoznam obrázkov

Úvod

Mobilné telefóny sú dnes neoddeliteľnou súčasťou moderného technologického života. V našich rukách sú niekoľko hodín denne a používame ich ako komunikačné médiá a zdroje noviniek a zábavy. Mobil, prípadne tablet, však môže slúžiť ako výborná učebná pomôcka hlavne vďaka svojej prenosnosti. Na tejto idei je založená aplikácia Dráček, ktorá sa zameriava na spojenie výuky a hry pre deti predškolského a školského veku.

Táto aplikácia bola vyvíjaná iteračne študentmi FIT ČVUT v rozsahu troch doterajších iterácií, pričom každá bola zameraná na inú časť celku. Moja práca bude popisovať naviazanie na aktuálny stav projektu a jeho vylepšenie. Výsledok práce ďalej posunie tento projekt k vydaniu a nasadeniu do prevádzky a zároveň bude slúžiť ako manuál pre nasledujúce iterácie projektu.

V práci sa konkrétne zameriavam na analýzu aktuálneho riešenia, komunikáciu z dizajnérom, návrh a implementáciu zmien a taktiež vyhodnotenie výsledkov užívateľského testovania po aplikácii týchto zmien.
Súbežne s mojou prácou prebieha aj práca môjho kolegu Martina Kameníka, ktorý má za úlohu navrhnúť a implementovať novú serverovú časť projektu. Niektoré časti mojej práce budú priamo závislé na zmenách, ktoré Martin implementuje.

Táto téma ma zaujala z dôvodu môjho dlhodobého záujmu o platformu Android v užívateľskom aj vývojárskom zmysle. Pred začiatkom tejto práce som sa venoval vývoju niekoľkých aplikácií zameraných na Android a táto skutočnosť mi pri písaní práce pomohla.

Kapitola 1

Cieľ práce

Túto prácu je možné okrem delenia na kapitoly rozdeliť aj na dve časti: analytickú a praktickú.
Cieľ analytickej časti je zistiť nedostatky aktuálneho riešenia týkajúce sa grafického rozhrania aplikácia. Do tejto časti patrí kapitola Analýza, v ktorej sa venujem technológiám, ktoré by nedostatky odstránili, prípadne aplikáciu vylepšili. Po analýze nutných predispozícií dizajnéra pracujúcom na tomto projekte nasleduje určenie funkčných a nefunkčných požiadaviek.
Cieľ praktickej časti je rozobrať možné spôsoby realizácie nápadov z prvej časti a následne ich implementovať. V kapitole Návrh, ktorá je jedna z kapitol praktickej časti, rozoberám rôzne možnosti implementácie riešení, uvádzam plusy a mínusy jednotlivých možností a na ich základe následne vyvodím výsledok. V tejto kapitole sa vyskytne aj porovnanie rôznych možností týkajúcich sa spolupráce medzi vývojárom a dizajnérom. Nasledujúca kapitola v tejto časti bude Implementácia, v ktorej popíšem priebeh realizácie návrhu a stav po nej. Záverečná kapitola tejto práce bude Testovanie, v ktorej čitateľovi priblížim, aké testy boli v rámci tejto práce vykonané.

1.1 Rozbor zadania

V tejto časti preberiem jednotlivé body zadania a v krátkosti popíšem, akým spôsobom sa ich budem snažiť dosiahnuť. Ciele práce sú po konzultácii s vedúcim práce mierne pozmenené kvôli absencii dizajnérky, ktorá bola pre tento projekt pripravená. Svoju absenciu ohlásila až po schválení zadania, čo viedlo k rozhodnutiu zadanie si ponechať a do maximálnej možnej miery ho splniť aj napriek tejto udalosti.

- **Podrobte aplikáciu užívateľskému testovaniu s cieľovou skupinou a výsledky vyhodnoťte**
  Pred realizáciou zmien je vhodné zistiť stav užívateľskej prívetivosti prostredníctvom užívateľského testovania. Testovanie po realizácii zmien sa

nachádza v zadaní tejto práce ako posledný bod. Z časových dôvodov bude v rámci tejto práce uskutočnené iba testovanie po realizácii zmien. Ako substitúciu po konzultácii s vedúcim práce vyhodnotím výsledky testovania mojich kolegov z predchádzajúcej iterácie.

- **Analyzujte chyby a nedostatky v užívateľskom rozhraní s ohľadom na užívateľskú prívetivosť (UX)**
  Tieto chyby a nedostatky sa aj napriek chýbajúcemu dizajnérovi budem snažiť odhaliť a analyzovať; zameriam sa však aj na nedostatky a nové nápady, ktoré nemusia priamo súvisieť s grafickou podobou aplikácie.

- **Navrhnite metodiku riešiacu spoluprácu s dizajnérmi, ktorí nemajú skúsenosti s platformou Android**
  Dobrá spolupráca je závislá od dobrej komunikácie a pochopenia práce. V tejto práci uvediem nástroje, ktoré sa na projekte používajú a navrhnem niekoľko spôsobov, akými by mohla práca medzi vývojárom a dizajnérom prebiehať.

- **Pomocou metód softwarového inžinierstva navrhnite novú podobu chovania grafického rozhrania aplikácie a to s ohľadom na design dodaný spolupracujúcou dizajnérkou**
  Vzhľadom na povahu situácie navrhnem niekoľko grafických komponentov spolu s ich logikou. Týmto komponentom nebude venovaná prílišná pozornosť týkajúca sa dizajnu kvôli efektivite práce, teda vyhnutie sa robeniu rovnakej úlohy dvakrát – raz bez dizajnéra a raz s ním.

- **Implementujte navrhnuté zmeny s ohľadom na viacnásobné použitie navrhnutých komponentov**
  Zmeny, ktoré zostanú po fázach analýzy a návrhu relevantné, budú implementované s dôrazom na jednotné správanie a viacnásobné použitie naprieč jadrom a modulmi.

- **Hotové riešenie podrobte užívateľským testom v UX laboratóriu a vyhodnoťte výsledky**
  Implementované zmeny budú podrobené užívateľským testom. K testom pripravím podklady, na základe ktorých budú potom prebiehať na cieľovej skupine aplikácie.

Kapitola 2

Analýza

V tejto kapitole sa najskôr zamieram na analýzu aktuálnej podoby projektu a jeho históriu. Po predstavení základných častí systému sa budem zameriavať na časť týkajúcu sa tejto práce – Jadro. V prvom rade vymenujem a popíšem nedostatky, ktoré táto časť obsahuje a následne sa budem venovať analýze možností opráv týchto nedostatkov.

2.1 Časti projektu

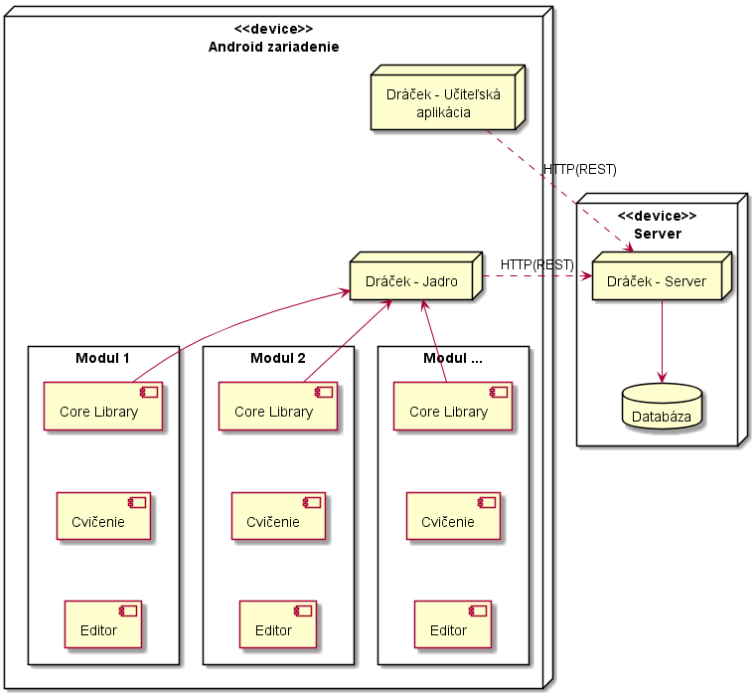
Ešte pred začatím analýzy konkrétnej časti projektu predstavím čitateľovi, ako celý projekt vyzerá a z akých časti sa skladá. Tieto časti sú znázornené na obrázku 2.1 a v nasledujúcich riadkoch ich stručne popíšem.

Server
: Na Dráčkovskom serveri beží serverová aplikácia, ktorá je napojená na databázu. S mobilnými aplikáciami komunikuje prostredníctvom protokolu HTTP a návrhového vzoru REST.

Učiteľská aplikácia
: Učiteľská aplikácia, ako názov napovedá, slúži pre učiteľov na plnenie rozličných úloh, medzi ktorými sú zobrazenie a správa žiakov, tried, cvičení a zadaní.

Jadro
: Aplikácia Jadro je hlavným gróm projektu Dráček. Je to vstupná brána do hier a správca zadaní pre deti. Stará sa o synchronizáciu výsledkov a zadaní so serverom, čo umožňuje fungovanie aj v offline móde.

Moduly
: Moduly obsahujú jednotlivé cvičenia vo forme hier, ktoré deti absolvujú. S jadrom komunikujú prostredníctvom knižnice *corelibrary*, ktorá slúži na posielanie výsledkov jadru. Okrem využitia deťmi slúžia moduly aj



Obr. 2.1: Diagram znázorňujúci aktuálnu podobu systému Dráček

pre učiteľov, ktorí v editačnej časti modulu môžu vytvárať nové otázky pre cvičenie.

2.2 História projektu

História projektu je sčasti voľne citovaná z práce Patrika Pavelca [2] a následne doplnená o aktuálne informácie.
Základy projektu Dráček boli položené v roku 2012 na predmete Softwarový týmový projekt 1 vyučovanom na FIT ČVUT. Tento projekt bol už od počiatku zamýšľaný ako dlhodobý, v rozsahu niekoľkých semestrálnych a bakalárskych prác. Na Dráčkovi sa podieľajú väčšinou študenti oboru Softwarové inženýrství, ktorí si vo svojich prácach vyskúšajú prácu v tíme, komunikáciu a riešenie problémov.

Dráček bol od svojho počiatku vytváraný v spolupráci so základnou školou Smečno, ktorá mala ideu poskytnúť žiakom alternatívne spôsoby učenia a to hlavne hravou formou. Pôvodný nápad výukovej aplikácie bol cielený na počítače, čo sa však neskôr zmenilo. Téma a maskot aplikácie vznikol z miestnej legendy o drakovi hľadajúcom poklad. Projekt mal od začiatku veľký potenciál jednak úspechu na spomínanej základnej škole a jednak expanzie na iné školy.

Prvá funkčná verzia vznikla v roku 2013 prácou prvej iterácie kladúcej základný kameň pre nasledujúci vývoj. Aplikácia vznikala v Jave a jej architektúra bola plánovaná ako modulárna – jedno jadro a viacero modulov cvičení. Zoznam členov prvej iterácie vrátane ich zameraní v rámci projektu je nasledovný:

Jan Bradáč	Tvorba jadra, menu a správa prihlásených užívateľov v aplikácii. [3]
Petr Kubišta	Správa zásuvných modulov. [4]
Ondřej Kužela	Perzistencia dát, serverová aplikácia a ukladanie výsledkov. [5]
Michael Bláha	Tvorba cvičení zameraných na prácu s textom a jeho vnímanie. [6]
Radek Tomšů	Tvorba cvičení zameraných na zrakové vnemy. [7]

Aplikácia bola úspešne nasadená a používaná. Jej používanie však malo jeden problém – nedostatok počítačov dostupných žiakom. Tým pádom bol Dráček dostupný len niektorým deťom.

V roku 2014 získala škola európsky grant na tablety so systémom Android v počte dostatočnom pre všetkých žiakov. Táto skutočnosť zmenila orientáciu projektu na operačný systém Android a vyprodukovala ďalšiu iteráciu, ktorá vznikla v rámci rovnakého predmetu ako v prvom prípade. Táto iterácia niesla názov Dráček II a zoznam členov vrátane ich zamerania je nasledovný:

Patrik Pavelec	Tvorba jadra klienta. [2]
Ondřej Filip	Perzistencia dát a serverová časť, rozhranie pre učiteľov. [8]
Miroslav Mazel	Vývoj modulov. [9]
Michal Bureš	Vývoj modulov. [10]
Karel Kovařovic	Gamifikácia a personalizácia. [11]
Pavel Podaný	Tvorba animovaného avatara. [12]

Výstup práce druhej iterácie bol síce funkčný, ale neucelený kód.

V roku 2016 sa do projektu znovu cez rovnaký predmet zapojila nová iterácia označovaná ako Dráček III. Táto iterácia sa zameriavala na vývoj nových modulov pre projekt. Zoznam jej členov spolu s témami ich prác je nasledovný:

Ondřej Slabý	Vývoj modulov zameraných na algoritmizáciu. [13]

Jaroslav Štěpán Vývoj modulov zameraných na fyziku. [14]

Jaroslav Ryba Vývoj modulov zameraných na matematiku. [15]

V čase, keď členovia tretej iterácie písali svoje bakalárske práce, som sa spolu s ďalšími kolegami pripojil do projektu a založili sme novú iteráciu – Dráček IV. Tá mala naviazať na produkt Dráčka II, zistiť jeho nedostatky a opraviť ich. Z pôvodného tímu piatich ľudí sme v Dráčkovi IV na bakalársku prácu ostali len dvaja. Členovia štvrtej iterácie sú tým pádom:

Dominik Sivák Zlepšenie užívateľského rozhrania aplikácie jadra.

Martin Kameník Vývoj serverovej časti. [16]

2.3 Analýza aktuálneho riešenia časti jadro

V tejto časti sa budem zaoberať aktuálnou verziou aplikácie Dráček – jadro, žiacka časť. Aplikácia bola vo svojej prvej generácii vyvíjaná na platforme Java SE s grafickým prostredím Swing. Vývojom sa zaoberal Peter Kubišta vo svojej bakalárskej práci [4]. V druhej generácii sa cieľová platforma zmenila na Android, a tým pádom prebiehal vývoj aplikácie od základov. Týmto vývojom sa vo svojej bakalárskej práci [2] zaoberal Patrik Pavelec. Pochopiteľne bola v tejto fáze vývoja priorita vybudovať funkčný a spoľahlivý backend, ktorý musel riešiť nové problémy a obmedzenia, ktoré prechod na túto platformu priniesol. Frontend tak bol až druhoradý a to sa podpísalo pod užívateľskú neprívetivosť a neatraktivitu aplikácie. Z vlastnej skúsenosti môžem povedať, že som mal problém sa v aplikácii zorientovať. Pri užívateľoch v predškolskom a rannom školskom veku, ktorí ešte nie sú zvyknutí na ovládanie aplikácií, je ešte dôležitejšie, aby bolo prostredie aplikácie intuitívne a užívateľsky prívetivé. Nedostatky v týchto oblastiach pripisujem neprítomnosti dizajnéra pri vývoji. V nasledujúcich bodoch vytýčim niekoľko nedostatkov, ktoré som používaním aplikácie objavil, prípadne sa o nich dočítal v práci Patrika Pavelca [2].

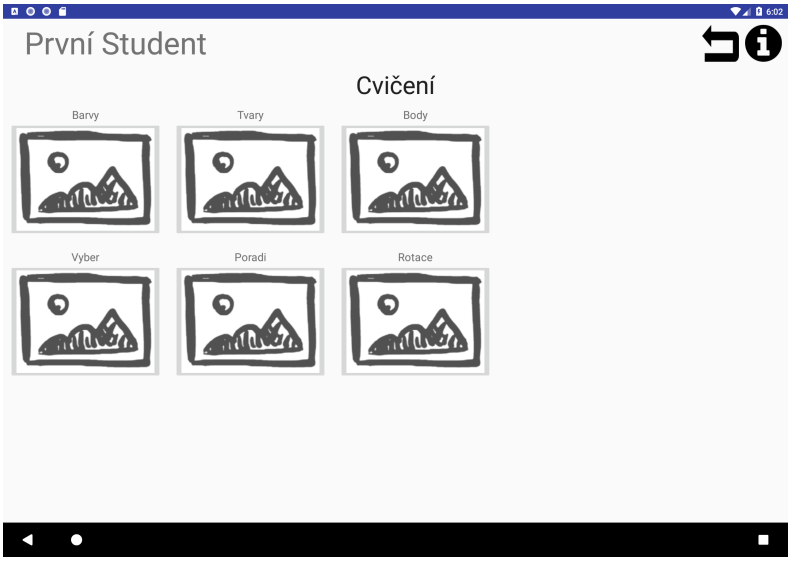
2.3.1 Nedostatky

Zložité prihlasovanie

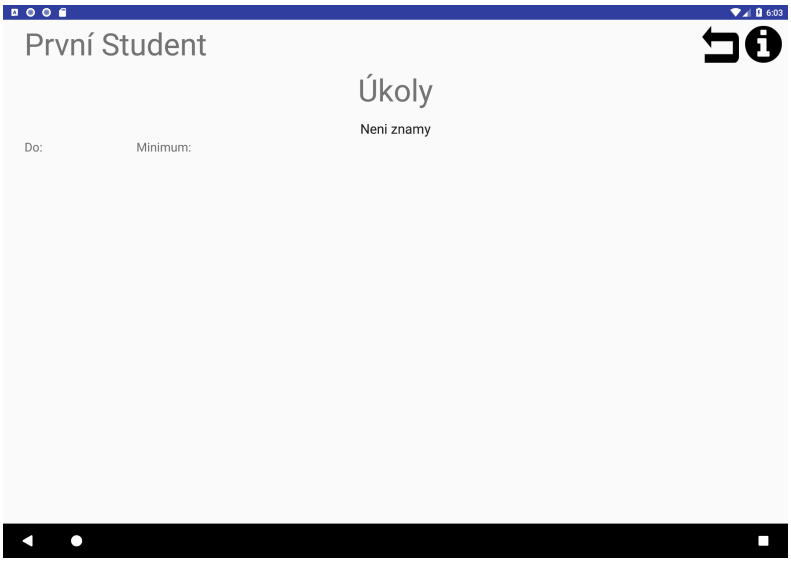
Do aplikácie sa prihlasuje pomocou používateľského mena a hesla. Aj keď to je tradičný spôsob prihlasovania do aplikácie, pre malé deti v predškolskom veku je tento spôsob nevhodný. Deti často ešte nemusia vedieť čítať alebo písať, a preto by bolo vhodnejšie zabrániť neoprávnenému prístupu jednoduchším spôsobom, ktorý bude zároveň pohodlný.

Nedokončené GUI

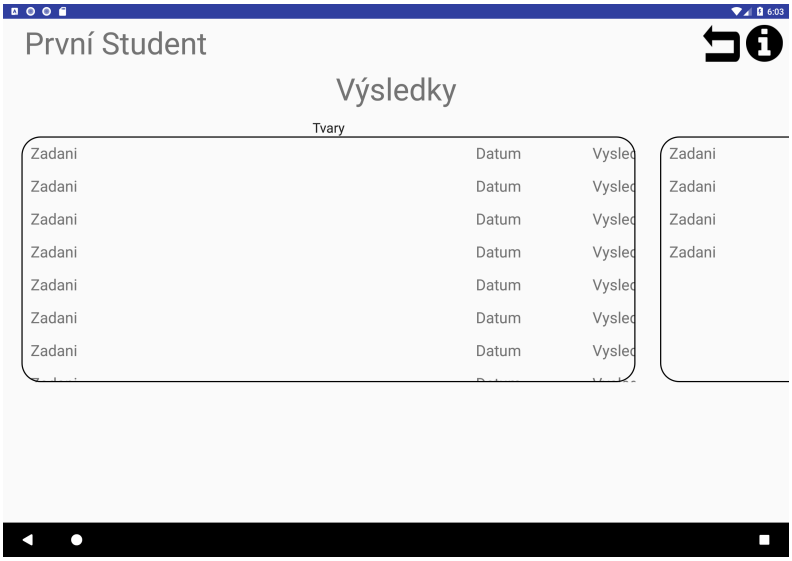
Aplikácia pôsobí už na prvý pohľad ako nedokončená – biele pozadia,



Obr. 2.2: Aktuálna podoba obrazovky s cvičeniami



Obr. 2.3: Aktuálna podoba obrazovky s úlohami



Obr. 2.4: Aktuálna podoba obrazovky s výsledkami

nevýrazné tlačidlá, čiernobiele farby. Na deti to môže vplývať odpudzujúco, pretože sú zvyknuté na farebné knihy a videá. Keďže cieľom tohto projektu je ukázať, že aj učenie môže byť zábavné, musí sa užívateľ cítiť príjemne pri prechádzaní obrazovkami. Ukážky aktuálnej podoby grafického rozhrania je možné vidieť na obrázkoch 2.2, 2.3 a 2.4.

Chýbajúci sprievodca prvým spustením

Užívateľ je do aplikácie nepríjemne *„vrhnutý“* a chýba mu znalosť ako pokračovať ďalej. Túto skutočnosť môže odstrániť fakt, že dieťa bude aplikáciu spúšťať v úzkej spolupráci s učiteľom, no aj tak je vhodné užívateľa všetkými časťami aplikácie previesť. Forma by mala byť rozprávková, aby ladila so zvyškom prostredia.

Inštalácia modulov

Aktuálna inštalácia modulov prebieha zložito a pre reálneho používateľa aplikácie neprívetivo. Samotné inštalačné súbory modulov sa nachádzajú na viacerých úložiskách a je nutné získať k ním prístup. Naviac dokiaľ sa užívateľ nepokúsi spustiť modul z príslušnej obrazovky, nie je žiadnym spôsobom oboznámený o tom, že modul na zariadení nie je nainštalovaný, prípadne nie je nainštalovaná posledná verzia.

Nefungujúca sekcia úspechov

Sekcia úspechy (achievements/achievementy) je momentálne prázdna a nevyužitá. Patrik Pavelec vo svojej práci [2] diskutoval o spôsobe implementácie achievementov, no k samotnej implementácii sa z časových a rozsahových dôvodov dostať nedokázal.

2.4 Analýza možností na vylepšenie aplikácie

V tejto časti budem analyzovať možnosti odstránenia nedostatkov aktuálneho riešenia, ktoré som v tejto práci rozpísal v časti 2.3.1.

2.4.1 Integrácia do služby Google Play

Jednou z ciest k jednoduchšiemu používaniu aplikácie je publikovanie aplikácie a príslušných modulov do platformy Google Play. Google Play je platforma určená k distribúcii digitálneho obsahu primárne na Android zariadenia. Táto platforma zastrešuje niekoľko služieb, z ktorých každá slúži na distribúciu iného obsahu. Medzi tieto služby patria:

- Aplikácie
- Hry – Google Play Games
- Hudba – Google Play Music
- Filmy – Google Play Movies
- Knihy – Google Play Books
- Noviny a časopisy – Google Play Newsstand

Pre náš prípad sú relevantné sekcie Aplikácie a Hry. Google Play momentálne obsahuje viac ako 3.5 milióna aplikácií [17] a nachádza sa skoro na všetkých zariadeniach bežiacich na platforme Android. Aj keď momentálne existuje niekoľko alternatív k sťahovaniu nových aplikácií, všetky vyžadujú povolené nastavenie inštalácie z neznámych zdrojov.

2.4.1.1 Nadchádzajúce zmeny v službe

Počínajúc druhou polovicou roku 2018 vstupujú do platnosti nové podmienky aplikácií v Google Play. V krátkosti teraz čitateľa oboznámim s pojmami spojenými s touto problematikou.

Každá aplikácia postavená pre Android musí pri kompilácii obsahovať tri čísla, ktoré sú obsiahnuté v manifeste aplikácie: *compileSdkVersion*, *minSdkVersion* a *targetSdkVersion*.

compileSdkVersion určuje, s akou sadou komponentov sa bude aplikácia kompilovať.

minSdkVersion určuje, aká najnižšia verzia systému bude možná aplikáciu nainštalovať a spustiť.

targetSdkVersion určuje, s akými zmenami v chovaní systému aplikácia počíta.

Všetky hodnoty sú vyjadrené vo verzii SDK Androidu. Z pomedzi týchto troch je relevantná hodnota *targetSdkVersion*, o ktorej uvediem príklad [18]: Vo verzii Androidu 6.0 (SDK 23) boli zavedené tzv. *Runtime Permissions*. Princíp tejto funkcie spočíval v explicitnom vyžiadaní aplikácie k prístupu k citlivým dátam. Ak je *targetSdkVersion* aplikácie nastavená na 23 a vyššie, musí túto zmenu vývojár implementovať. V opačnom prípade to neplatí a hodnota *compileSdkVersion* nemá na vyžadovanie systému tejto funkcie od aplikácie žiaden vplyv.

Hlavné zmeny týkajúce sa služby sa zameriavajú na bezpečnosť a kompatibilitu aplikácií [19]:

- Od **augusta 2018** bude od každej novej aplikácie pridanej do Google Play vyžadované, aby cielila (*targetSdkVersion*) na maximálne rok starú verziu SDK.
- Od **novembra 2018** bude rovnaké pravidlo uplatnené na nové aktualizácie existujúcich aplikácií.
- Od **augusta 2019** bude vyžadované, aby aplikácie používajúce natívne knižnice [20] boli poskytované jednak v 32 a jednak 64 bitovej variante. Toto pravidlo sa týka rovnako nových a aktualizácií existujúcich aplikácií.

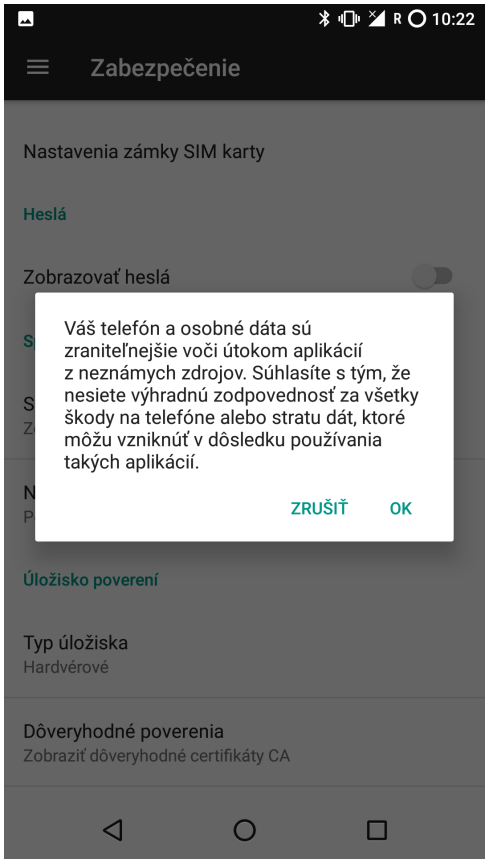
Ak je v pláne aplikáciu integrovať do Google Play, je nutné na tieto dátumy myslieť a prispôsobiť im stratégiu.

2.4.1.2 Google Play vs. Inštalácia z neznámych zdrojov

Na platforme Android existujú dve možnosti, ako aplikáciu nainštalovať. Jednou z nich je povolenie (anglický pojem permission sa však používa častejšie, preto ho ďalej budem používať) aplikácie inštalovať balíčky a druhá spočíva v spustení stiahnutého APK súboru. Spomínaný permission je konkrétne *android.permission.INSTALL_PACKAGES*, avšak jediná možná skupina, ktorá môže tento permission využívať, je skupina systémových aplikácií [21]. V operačnom systéme Android môže nainštalovaná aplikácia byť buď systémová, alebo užívateľská.

Systémová
: Systémové aplikácie sú uložené v priečinku */system/app*. Užívateľ má k tejto partícii počas behu systému iba **read-only** prístup, teda jej obsah sa bez root povolenia meniť nedá. Obsah tejto partície sa mení hlavne počas aktualizácie systému a počas aktualizácií týchto aplikácií. To znamená napríklad aj to, že užívateľ tieto aplikácie nemôže odinštalovať.

Užívateľská
: Užívateľské aplikácie sú naopak uložené v priečinku */data/app* a môžu byť voľne nainštalované a odinštalované podľa potreby.



Obr. 2.5: Dialóg, ktorý sa zobrazí pri zapnutí Inštalácie z neznámych zdrojov

Inštalácia z neznámych zdrojov je nastavenie, ktoré povoľuje užívateľovi zariadenia inštalovať aplikácie z APK súborov. Toto nastavenie sa nachádza v sekcii *Zabezpečenie* a pri povoľovaní tohto nastavenia je užívateľovi zobrazený dialóg, ktorým musí zmenu potvrdiť (viď obr. 2.5). Už samotný text dialógu môže rodičov detí odradiť od používania tejto aplikácie na ich zariadeniach.

Povolenie inštalácie z neznámych zdrojov je veľké bezpečnostné riziko vzhľadom na terajšiu agresívnu povahu reklám, za ktorými sa skrýva množstvo aplikácií obsahujúcich rôzny malware alebo spyware. Na druhú stranu mala aj platforma Google Play niekoľko káuz týkajúcich sa vírusov v aplikáciách poskytovaných touto platformou, ako napríklad DroidDream [22] či FalseGuide [23]. Riziko vystavenia zariadenia vírusu je však rádovo nižšie, ako pri inštalácii aplikácie voľne z internetu.

Patrik Pavelec vo svojej práci [2] porovnával tieto dva spôsoby, ale nakoniec kvôli chýbajúcemu prístupu na server ostala problematika inštalácie modulov neuzavretá. Viac sa tomuto problému budem venovať v časti Návrh.

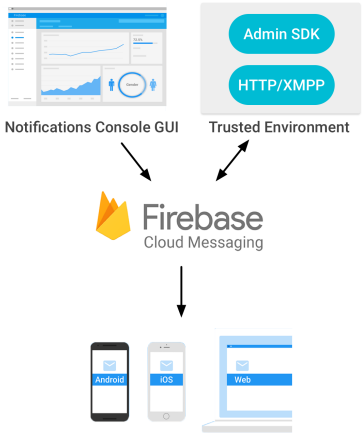
2.4.1.3 Google Play Console

Umiestnenie aplikácie do Google Play so sebou prináša okrem jednoduchšieho prístupu pre koncového užívateľa aj popis jeho správania v aplikácii a štatistiky dostupné z prostredia Google Play Console (GCP). GCP obsahuje niekoľko segmentov týkajúcich sa napríklad release managementu, či štatistík. Prístup je možný cez webovú aplikáciu aj cez aplikáciu na Android.

Štatistiky aplikácie ponúkajú niekoľko metrík dostupných k zobrazeniu a tie sa delia na dva druhy: metriky jednotlivých inštalácií a metriky aplikácie ako celku v Google Play. Uvediem niekoľko príkladov metrík:

- Jednotlivá inštalácia
  - Verzia OS Android na zariadení
  - Model zariadenia
  - Štát prislúchajúci užívateľovi
  - Jazyk nastavený na zariadení
  - Verzia a distribučný kanál aplikácie
- Aplikácia v Google Play
  - Počet inštalácií na aktívnych zariadeniach
    - Zariadenia, ktoré boli aktívne aspoň raz za posledných 30 dní a majú nainštalovanú konkrétnu aplikáciu
  - Počet odinštalovaní
  - Hodnotenia aplikácie
  - Peňažný zisk a počet kupcov
    - V prípade, že aplikácia je platená, alebo obsahuje mikrotranzakcie
  - Štatistiky FCM správ
    - Firebase Cloud Messaging (FCM) popíšem nižšie v tejto kapitole

Tieto a ďalšie metriky [24] je možné využiť k skvalitneniu poskytovania aplikácie. Medzi štatistiky patria aj správy o páde aplikácie a ANR (Application Not Responding = Aplikácia neodpovedá). Keďže vývoj aplikácie doteraz prebiehal a pravdepodobne aj naďalej bude prebiehať v iteráciách, medzi ktorými je malá komunikácia, je aplikácia vystavená väčšej chybovosti, ktorá sa pri používaní môžu vyskytnúť. Tieto správy majú oproti neodbornej spätnej väzbe veľkú výhodu vďaka obsiahnutému stack trace, ktorý umožní jednoduchšie hľadanie príčiny chýb. Užívateľ môže navyše k týmto správam pripojiť aj vlastnú správu popisujúcu napríklad spôsob, ako je možné pád reprodukovať.



Obr. 2.6: Jednoduché vyobrazenie fungovania služby FCM

Release management je ďalšia časť GPC. Vývojár v nej môže nastavovať publikovanie nových verzií. Medzi možnosti patrí A/B testovanie (určitej časti používateľov je distribuovaná kozmeticky alebo funkčne odlišná verzia aplikácie bez toho, aby o tom vedeli), alpha a beta distribučný kanál a vydanie novej verzie na produkčný kanál napríklad s možnosťou postupného uvoľňovania aktualizácie užívateľom.
Naviac môžu byť tieto kanály spárované s Continuous Integration[1](CI) a poskytovať tak nové verzie pre alpha kanál po každom commite, alebo denne.

2.4.1.4 Firebase

Firebase je platforma pre vývoj webových a mobilných aplikácií fungujúca od roku 2011 a odkúpená v roku 2014 spoločnosťou Google [25]. Do tejto platformy spadá niekoľko služieb, z ktorých niektoré relevantné spomeniem.

Firebase Cloud Messaging (FCM)
FCM je bezplatné multiplatformové riešenie pre doručovanie správ na zariadenia užívateľov, ktoré nahradzuje Google Cloud Messaging (GCM) [26]. Medzi spôsoby použitia patrí napríklad oznámenie o novej verzii užívateľských dát na serveri. Fungovanie platformy je v jednoduchosti zobrazené na obr. 2.6.

Firebase Auth
Firebase Auth poskytuje taktiež multiplatformové riešenie pre autentifikáciu užívateľov [27]. Riešenie poskytuje backend služby, ktoré umožňujú

[1]Continuous Integration je spôsob vývoja SW, ktorý podoruje časté commity a automatické buildy za účelom rýchleho odhalenia chyby. [1]

verifikovať prístupové údaje, ale taktiež je možné využiť vlastný server na túto aktivitu a obmedziť rozsah dát ukladaných na serveroch Firebase. Medzi možnosti prihlásenia patrí aj prihlásenie pomocou tretích strán ako napríklad Google, Twitter alebo Facebook. Firebase Auth SDK tiež obsahuje napríklad anonymný login, či správu sessions.

2.4.1.5 Google Play Games

Služba Google Play Games vznikla v roku 2013 za účelom zjednodušiť vývoj hier na mobilné platformy [28]. Google Play Games v sebe obsahuje niekoľko funkcií, ktoré je možné využiť a vývojár ich teda nemusí vyvíjať znovu od začiatku. Medzi tieto funkcie patria [29]:

- Úspechy
- Rebríčky
- Ukladanie postupu hry do cloudového úložiska
- Režim viacerých hráčov

Použitie tejto služby je spojené s prihlásením do Google účtu na zariadení užívateľa. Pre tento projekt sú relevantné všetky vyššie uvedené funkcie, preto ich popíšem podrobnejšie.

Úspechy
Služba Google Play Games implementuje jednak logiku týkajúcu sa získavania úspechov a jednak frontend časť ako napríklad zobrazenie oznámenia o získaní úspechov alebo obrazovku so všetkými získanými úspechmi. Úspechy sa delia na dva druhy: štandardné a inkrementálne. Štandardné sú získané vyvolaním jednej akcie, naopak pri inkrementálnych sa ukladá pokrok postupne. Tieto úspechy sú viazané na príslušný prihlásený Google účet.

Rebríčky
Rovnako ako úspechy, tak aj rebríčky sú v Google Play Games implementované zo stránky backendu aj frontendu. Pre jednu hru je možné vytvoriť až 70 rôznych rebríčkov, ktoré fungujú v dvoch módoch: verejné (public) a spoločenské (social). Medzi týmito dvoma módmi môže prepínať užívateľ, keď si chce zobraziť svoju pozíciu buď v rámci všetkých hráčov tejto hry (public) alebo len hráčov, ktorí sú zároveň v kruhoch [30] užívateľa (social).

Ukladanie postupu do cloudového úložiska
Táto služba poskytuje aj možnosť ukladania postupu v hre na servery Google. Samotná služba sa stará o synchronizáciu postupu medzi zariadeniami. Vývojár musí zabezpečiť serializáciu a následnú deserializáciu

dát a tieto uložené dáta sa započítavajú do Google Drive kvóty účtu vývojára.

Režim viacerých hráčov
Režim viacerých hráčov (multiplayer) je dnes často používaná forma hry. Google Play Games poskytuje API, ktoré umožňuje jednoduché vytváranie virtuálnej miestnosti a ukladanie stavu miestnosti a účastníkov na servery Google. Taktiež je poskytované grafické prostredie, ktoré umožňuje hráčom vstúpiť do virtuálnej miestnosti, pozývať ostatných hráčov do tejto miestnosti a zobrazovať prijaté pozvánky.

2.4.1.6 Použiteľnosť na platforme Chrome OS

Android aplikácie je možné inštalovať na vybrané Chromebooky cez Google Play Store [31]. Chromebook je notebook bežiaci na Chrome OS, ktorý je založený na linuxovom jadre. Tento OS je zastrešovaný spoločnosťou Google a jeho využitie je cielené hlavne na vzdelávanie.
Android aplikácie môžu na tomto OS byť spustené bez väčších modifikácií. Používaním Dráčka na Chrome OS sa môže ešte viac rozšíriť cieľová skupina tejto aplikácie a zároveň sa spojí cieľ prvotnej iterácie Dráčka (vývoj aplikácie v Jave určenej pre počítače) a jeho druhej iterácie (vývoj aplikácie na platformu Android).
Tomuto nápadu sa ďalej v práci venovať nebudem; slúži skôr ako nápad do budúcna pre ďalšie iterácie tohto projektu.

2.4.2 Fungovanie v dvoch módoch

V sekcii týkajúcej sa nedostatkov aplikácie som popisoval medziiným aj zložité prihlasovanie sa do aplikácie. Miesto používania aplikácie je možné rozdeliť do dvoch prostredí, každé vyžadujúce iný stupeň ochrany pred neoprávneným prístupom:

Zdieľané prostredie je primárne priamo v triede, obecne na zdieľaných tabletoch. Tieto tablety sa môžu často dostať do nesprávnych rúk, či už náhodou (napr. žiak si nevšimne, že predchádzajúci držiteľ tabletu sa neodhlásil) alebo nie. Vyžaduje sa vyšší stupeň ochrany, napríklad odhlásenie žiaka pri ukončení aplikácie. V tomto prostredí je predpokladaná asistencia učiteľa pri prihlasovaní.

Súkromné prostredie je akékoľvek iné prostredie, v ktorom sa počíta s inštaláciou na vlastnom súkromnom zariadení. Toto zariadenie si proti neoprávnenému prístupu užívateľ (alebo rodič) ochráni sám, jednou z metód operačného systému.

Tieto dve prostredia sa premietnu do požiadaviek na aplikáciu a následne do implementácie.

2.4.3 Prihlasovanie pomocou rozpoznania tváre

Na trhu s mobilnými telefónmi existuje niekoľko riešení rozpoznávania tváre určené pre odomykanie zariadenia. Táto funkcia spočíva v tom, že zariadenie získa potrebné data na rozpoznanie príslušným senzorom a na základe týchto dát je následne vyhodnotená zhoda, či nezhoda aktuálneho obrazu s uloženým vzorom. Niekoľko riešení je vyvíjaných priamo výrobcami telefónov [32]:

Samsung
: Technológia na rozpoznávanie tváre od Samsungu obsiahnutá napríklad v modeloch Galaxy S8 alebo Note 8 využíva vzory v dúhovkách očí užívateľa. Podobne ako otlačky prstov sú dúhovky unikátne pre každého človeka. Rozpoznanie spočíva v nasvietení očí infračervenou diódou a následným snímaním špeciálnym fotoaparátom, ktorý dokáže zachytiť spektrum infračerveného svetla. Existujú však prípady, kedy sa túto technológiu podarilo hackerom prelomiť [33].

Apple
: Apple svoju technológiu Face ID predstavil spolu s modelom iPhone X. Princíp spočíva v nasvietení tváre rovnako infračerveným svetlom, ale namiesto očí sa táto technológia zameriava na celú tvár. Následne sa 3D model tváre vytvorí snímaním miniatúrnych pohybov užívateľa, vďaka čomu je táto technológia odolná voči útokom vykonaným 2D spôsobom (napr. umiestením fotografie pred snímač).

Google túto funkciu zabudoval priamo do Androidu už od verzie 4.0 (Ice Cream Sandwich) v roku 2011 [34]. Jej využitie je však možné nájsť len pri odomykaní obrazovky a tým pádom nie je dostupné žiadne verejné oficiálne API prístupné vývojárom. Medzi open-source knižnice použiteľné na Androide patrí napríklad *Android Face Recognition with Deep Learning – Test Framework* [35] či *FaceRecognitionLib* [36].
Takýto spôsob prihlasovania má v prípade aplikácie Dráček dve nevýhody: rôzne zariadenia a GDPR.

Použitie rozličných zariadení
: Vyššie vymenované možnosti, ktoré používajú výrobcovia telefónov majú jednu spoločnú vlastnosť – učenie a rozpoznanie tváre prebieha stále na jednom zariadení. Keďže aplikácia Dráček bude často využívaná na rozličných zariadeniach (v škole/doma), podpora týchto zariadení s rôznymi grafickými čipmi by značne sťažovala vývoj.

GDPR
: GDPR (General Data Protection Regulation) je sada pravidiel, akými by mali spoločnosti nakladať s osobnými informáciami užívateľov [37]. Vstupuje do platnosti 25. 5. 2018 a za nedodržanie týchto pravidiel sa organizáciám udeľujú veľké pokuty.

GDPR sa v tomto projekte týka už aj existujúcich dátových štruktúr v našej databáze (meno, priezvisko,...), preto je podriadenie sa regulácii ešte pred použitím v školách dôležitejšie, ako vytvárať nové sady užívateľských dát.

Kvôli týmto nevýhodám a malej pridanej hodnote na projekte tento nápad opúšťam v analytickej rovine a ďalej sa mu nebudem venovať.

2.5 Analýza metodiky spolupráce s dizajnérom

Spolupráca SW vývojára a dizajnéra sa nachádza v každom projekte obsahujúcom GUI. Mnohokrát, pri menších projektoch, môže obe role zastupovať jedna osoba – postup, ktorým prebiehali predchádzajúce iterácie tohto projektu. Tento postup je pochopiteľný v rámci vytvárania prototypu aplikácie. Ak však vývoj tejto aplikácie má postupovať do ďalších fáz, je nutné aby sa do aktívneho vývoja zapojil aj dizajnér. Preto v krátkosti predstavím nástroje používané pre vývoj tohto projektu a uvediem, aký význam majú pre dizajnéra z pohľadu vývojára:

Git je systém určený k správe revízií súborov [38] a v tomto projekte sa používa k verzovaniu zdrojového kódu. Medzi jeho výhody patrí hlavne možnosť súbežnej práce vývojárov, ktorú je možné následne spojiť.

Android Studio je oficiálne IDE určené pre vývoj Android aplikácií [39]. Okrem editácie kódu obsahuje aj iné funkcionality:

Grafický editor umožňuje náhľad grafického rozloženia bez nutnosti spúšťať aplikáciu akýmkoľvek spôsobom. Náhľad je možný v rôznych kombináciách veľkosti obrazovky, verzie Androidu, lokalizácie, témy a iných.

Android Virtual Device umožňuje emulovať Android zariadenie, na ktorom je možné následne spustiť skompilovanú aplikáciu. Taktiež je možné využiť rôzne kombinácie veľkosti obrazovky a verzie Androidu.

Okrem týchto nástrojov uvediem znalosti, ktoré by mal dizajnér ovládať:

Operačný systém Android bude dizajnér používať na vizuálnu kontrolu zmien. Preto dizajnér musí byť schopný nainštalovať APK súbor, ktorý sa automaticky vygeneruje na build portáli.

Formát XML je používaný jednak v súboroch rozloženia aplikácie a jednak aj v ďalších častiach, ktoré popisujem neskôr v tejto práci. Je vhodné, aby dizajnér bol oboznámený so základnou syntaxou použitou v tomto formáte.

2.6 Rozbor funkčných a nefunkčných požiadaviek

Funkčné a nefunkčné požiadavky budú sčasti prebraté z prác mojich predchodcov Petra Kubištu [4] a Patrika Pavelce [2], ktorí sa obaja v rámci svojich bakalárskych prác zaoberali vývojom jadra aplikácie pre žiacku časť. Nimi navrhnuté požiadavky budú do maximálnej možnej miery zachované tak, aby som bol schopný dosiahnuť cieľ svojej práce. K dosiahnutiu tohto cieľa pridám nové požiadavky, a upravím alebo v prípade neaktuálnosti odstránim jestvujúce.

2.6.1 Funkčné požiadavky

FP1 – Aplikácia bude obsahovať autentifikáciu užívateľa

- Aplikácia bude používaná v dvoch prostrediach – na zdieľanom a na súkromnom zariadení. Je preto nutné používať rozličné stratégie prihlásenia pre každé prostredie:
  - Zdieľané – vyžadované prihlásenie sa pomocou mena a hesla. Predpokladaná bude asistencia učiteľa pre ešte negramotné deti.
  - Súkromné – pri opätovnom prihlásení do uloženého účtu nie je od užívateľa vyžadované heslo.

FP2 – Aplikácia bude obsahovať sprievodcu pri prvom spustení a všadeprítomnú nápovedu

- Užívateľ bude pri prvom prihlásení sa do systému hravou formou sprevádzaný prostredím, kde zistí ako aplikáciu ovládať a čo je jej cieľom.
- Užívateľovi bude na každej stránke dostupná nápoveda, ktorá bude vysvetľovať, kde sa užívateľ nachádza a aké sú možné nasledujúce kroky.

FP3 – Z aplikácie bude možné spúšťať jednotlivé výukové moduly

- Užívateľ po prihlásení bude mať možnosť spúšťať moduly, ktoré sú mu priradené učiteľom. Ak konkrétny modul na zariadení nainštalovaný nie je, užívateľ o tom bude informovaný.

FP4 – Po dokončení modulu sa výsledok uloží

- Po ukončení práce s modulom sa výsledok uloží. Uloží sa teda aj v prípade, ak úloha ešte nie je dokončená.
- Aplikácia uloží výsledok aj v prípade, že zariadenie momentálne nemá pripojenie na internet. Synchronizácia znovu prebehne pri pripojení na internet.

FP5 – Užívateľ si bude môcť zobraziť výsledky

- Hlavné menu aplikácie bude obsahovať odkaz na obrazovku s výsledkami dokončených úloh. Tie budú rozčlenené podľa modulov a budú zobrazované v percentách.

FP6 – Užívateľ si bude môcť zobraziť úlohy

- Hlavné menu aplikácie bude obsahovať odkaz na obrazovku so zadanými úlohami, ktoré budú rozčlenené podľa vyučujúceho.
- Užívateľ bude môcť jednotlivé zadania spúšťať jedným kliknutím z tejto obrazovky.

FP7 – Užívateľ si bude môcť zobraziť úspechy

- Hlavné menu aplikácie bude obsahovať odkaz na obrazovku so získanými úspechmi.

2.6.2 Nefunkčné požiadavky

NP1 – Aplikácia bude cielená na platformu Android verzie 8.1

- Kvôli podmienkam spoločnosti Google týkajúcich sa umiestňovania obsahu na Google Play, je nutné aby aplikácia cielila (*targetSdkVersion*) na maximálne rok starú verziu Androidu. V dobe písania tejto práce je to Android 8.1 (*Oreo*), teda SDK verzie 27.
- Aplikácia bude kompilovaná (*compileSdkVersion*) v čase písania práce taktiež proti najnovšej verzii SDK (27).

NP2 – Beh aplikácie bude zaručený na zariadeniach s Android verziou 6.0 a vyššou

- Minimálna verzia Androidu potrebná na inštaláciu a spustenie aplikácie (*minSdkVersion*) bude 6.0, teda SDK verzie 23.

NP3 – Aplikácia bude dostupná k stiahnutiu na Google Play

- Aplikácia a jej moduly budú dostupné na portáli s aplikáciami Google Play.
- Beta vydania aplikácie budú sprístupnené určitej skupine ľudí.

NP4 – Užívateľské rozhranie bude navrhnuté pre deti základnej školy

- Grafické užívateľské rozhranie (UI) a užívateľský zážitok (UX) budú vytvárané s ohľadom na cieľovú užívateľskú skupinu – deti základnej školy. Prostredie bude mať rozprávkovú podobu a bude dostatočne jednoduché, aby sa v ňom vedeli orientovať deti spomínaného veku.

NP5 – Aplikácia bude fungovať aj v offline móde

- V režime offline (bez pripojenia k internetu) bude zabezpečené:
  - Spúšťanie stiahnutých modulov a vypracovanie úloh.
  - Ukladanie výsledkov, zadaných úloh a úspechov v zariadení, ktoré sa po pripojení k internetu synchronizujú so serverom.
  - Zobrazenie uložených výsledkov, zadaných úloh a úspechov.

NP6 – Inštalácia modulov bude jednoduchá

- Pri inštalácii modulov sa bude brať ohľad na cieľovú vekovú skupinu. Táto aktivita musí prebiehať jednoducho a s čo najmenším nutným zásahom užívateľa.

NP7 – Nedostupnosť modulov na zariadení a ich neaktuálnosť bude znázornená

- Obrazovka s výberom modulov bude informovať užívateľa o tom, že daný modul nie je nainštalovaný/je neaktuálny ešte pred tým, ako daný modul spustí.
- V prípade kliknutia na takýto modul bude užívateľovi ponúknutá inštalácia najnovšej dostupnej verzie.

Kapitola 3

Návrh

V tejto kapitole sa budem zaoberať návrhom implementácie odstránenia nedostatkov a pridaním nových funkcionalít. Nedostatky boli popísané v časti Analýza a nové funkcionality sú čerpané z požiadaviek na aplikáciu. Forma návrhu každej časti bude obsahovať predstavenie problému čitateľovi, navrhnutie možností implementácie, rozpísanie kladov a záporov každej možnosti a zhodnotenie, v ktorom sa vyberie možnosť, ktorá sa bude implementovať.

3.1 Prihlásenie žiaka

Jednou z požiadaviek na aplikáciu je fungovanie v dvoch prostrediach – na zdieľanom zariadení a na súkromnom zariadení. Pri fungovaní v škole na zdieľanom zariadení vzniká problém pri prihlasovaní ešte negramotných detí. Títo užívatelia ešte nie sú schopní zadať prihlasovacie meno a heslo na úvodnej obrazovke. Učiteľ tak musí obchádzať celú triedu s tabuľkou prihlasovacích mien a hesiel. Takéto uchovávanie hesiel je bezpečnostné riziko s predpokladom k neoprávnenému prístupu. Chaotické prostredie škôlky, prípadne školy kombinované so súťaživou povahou detí ešte viac zvyšujú pravdepodobnosť útoku.

Tento problém je možné vyriešiť prihlasovaním žiaka cez dôvernú autoritu – učiteľa. Učiteľ tak bude mať oprávnenie prihlásiť do aplikácie akéhokoľvek žiaka z tried, ktoré vyučuje. Zneužitie tejto právomoci zo strany učiteľa je samozrejme vysoko nepravdepodobné. keďže nemá žiadnu motiváciu uškodiť žiakovi zlým vyplnením cvičenia.

Implementácia musí zabezpečiť to, aby po prihlásení žiaka bol znemožnený spätný prístup do učiteľského účtu. Server už momentálne poskytuje REST zdroj na získanie tried a žiakov prislúchajúcich týmto triedam. Vyhodnocovanie oprávnenia prihlásenia inej osoby bude prebiehať na serveri. Aktuálna implementácia prihlásenia funguje odoslaním REST POST požiadavky na server, kde požiadavka vo svojom tele obsahuje prihlasovacie meno a heslo. Ak sú tieto údaje správne, server odošle v odpovedi informácie o tokene a prihlá-

senom užívateľovi. Informácie o tokene konkrétne obsahujú samotnú hodnotu tokenu a čas expirácie. Nasledujúca ukážka použitá z práce Patrika Pavelca [2] znázorňuje požiadavku a odpoveď vo formáte JSON:

Listing 3.1: Požiadavka

```
--> POST http://185.88.73.6/dracek-nette/www/api/v1/token HTTP/1.1
    Content-Type: application/json; charset=UTF-8
    Content-Length: 36
    API-API-KEY: fc73b664-c712-0470-aa6a-bcb6048a5daf
    {
        "password":"1234",
        "username":"JDoe"
    }
--> END POST (36-byte body)
```

Listing 3.2: Odpoveď

```
<-- 200 OK http://185.88.73.6/dracek-nette/www/api/v1/token (386ms)
    Date: Sun, 22 Jan 2017 12:08:12 GMT
    Expires: Mon, 23 Jan 2017 10:00:00 GMT
    Content-Type: application/json;charset=utf-8
    {
        "token": {
                "token": "61bf3fcf99f6038f24aa39f6f7a3f63x",
                "expiration": 1487765292
                },
                "user": {
                    "id": 1,
                    "birthdate": null,
                    "role": "student",
                    "name": "John",
                    "surname": "Doe",
                    "username": "JDoe"
                }
    }
<-- END HTTP (186-byte body)
```

Prihlasovanie cez učiteľa bude prebiehať veľmi podobným spôsobom, keďže zmeny sa dotknú len požiadavky. Tá bude v tomto prípade obsahovať token prihláseného učiteľa a ID žiaka, ktorého chce prihlásiť. Na serveri sa vyhodnotí platnosť tokenu a oprávnenie prihlásiť konkrétneho žiaka a v prípade splnenia oboch požiadaviek bude naspäť zaslaná odpoveď v identickom formáte ako v ukážke, pričom informácia o tokene sa bude týkať žiaka, ktorého prihlásenie bolo vyžiadané.

Možnosti pre implementáciu grafického prostredia pre túto funkcionalitu sú dve: jednak priamo z aplikácie jadra a jednak cez učiteľskú aplikáciu.

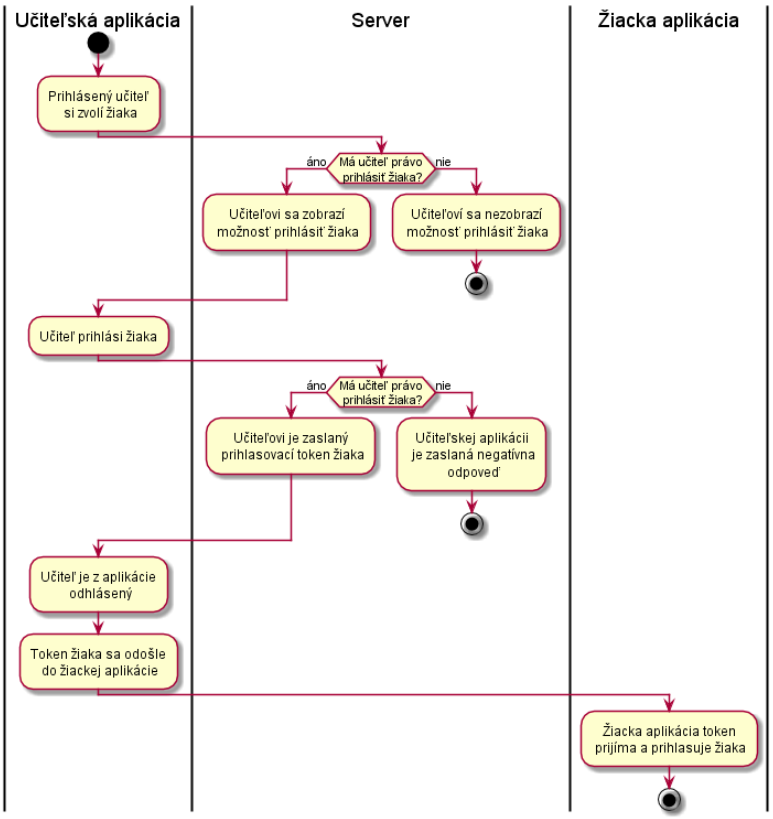
Prihlásenie z aplikácie jadra
Implementácia prihlásenia priamo z aplikácie jadra zahŕňa rozpoznanie role aktuálne prihláseného užívateľa a grafické prostredie pre výber žiaka. Výhoda tejto možnosti spočíva menšom zásahu do celého systému, vďaka nutnosti upravovať len jednu aplikáciu. Medzi nevýhody však patrí väčšia prácnosť pre implementáciu grafického prostredia a porušenie ustanovenia persón pre jednotlivé aplikácie.

Prihlásenie z učiteľskej aplikácie
Implementácia prihlásenia pomocou učiteľskej aplikácie výrazne zvyšuje intuíciu použitia tejto funkcie hlavne z dôvodu, že učitelia sú navyknutí používať učiteľskú aplikáciu, ktorá umožňuje správu používateľov. Na druhú stranu však vzniká problém komunikácie medzi dvoma aplikáciami, ktorý je ale pomerne jednoducho riešiteľný vďaka použitiu Android Intentov. Zásah do grafického prostredia učiteľskej aplikácie by bol minimálny vďaka už existujúcemu filtrovaniu a zobrazeniu detailu žiaka.

Z možností na návrh tejto funkcionality sa prikláňam k druhej možnosti – prihlásenie cez učiteľskú aplikáciu. Je to z dôvodu toho, že sa neporuší definovaná línia rozdeľujúca používateľov týchto dvoch aplikácií. Naviac celkový potrebný zásah do kódu bude nižší v porovnaní s prvým variantom. Na obrázku 3.1 je možné vidieť diagram aktivít popisujúci priebeh prihlásenia vo zvolenom variante. Ako je zrejmé, implementácia tejto funkcionality je závislá od zmien na serveri. Z tohto dôvodu bude implementácia prebiehať v spolupráci s Martinom Kameníkom, ktorý má na starosti novú verziu servera.

3.2 Grafické znázornenie stavu nainštalovaných modulov

Ďalšia pridaná nefunkčná požiadavka hovorí o poskytovaní informácie užívateľovi o tom, že sprístupnený modul momentálne nainštalovaný nie je, prípadne je dostupná nová aktualizácia tohto modulu. Moduly aplikácie nie sú priamo závislé na jadre, no neaktuálnosť modulov môže spôsobiť to, že žiak nebude mať možnosť vyplniť úlohu či zadanie. Aktuálna verzia API Google Services nepodporuje možnosť zistenia, či je aplikácia aktuálna ani získanie čísla verzie najnovšej aktualizácie z Google Play Store. Toto tvrdenie vychádza z nemožnosti nájsť akúkoľvek zmienku o tejto funkcionalite na vývojárskom portáli Androidu, naviac sa tento problém rozoberal aj na [40]. Neprítomnosť tejto funkcie v oficiálnom API môže byť nahradené dvoma spôsobmi: získavať číslo najnovšej verzie z web stránky Google Play, alebo si uchovávať toto číslo na vlastnom serveri a dotazovať sa na neho.



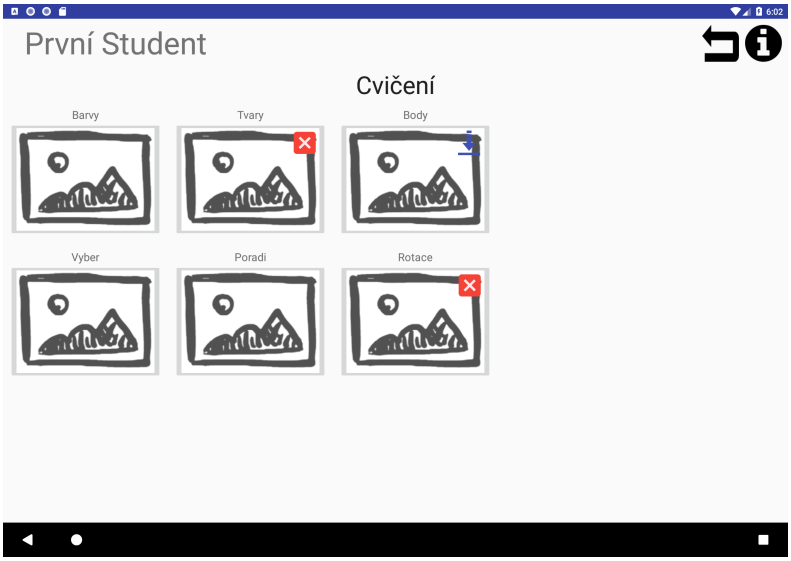
Obr. 3.1: Diagram aktivít znázorňujúci prihlásenie žiaka cez učiteľa

Použitie webu Google Play Store

Táto možnosť je z dvojice menej pracná a menej komplexná zmena, no jej funkčnosť nemôže byť zaručená. Možnosť implementácie spočíva v nájdení hodnoty v správnom HTML prvku stránky. Ak sa však štruktúra alebo názov prvku zmení, narúša to použitie.

Použitie vlastného servera

Použitie vlastného servera znamená v prvom rade upraviť chovanie a odpovede používaného servera. V aktuálnom riešení server posiela údaje o dostupných moduloch, ktoré následne môže užívateľ spustiť. Tým pádom je implementácia možná pridaním jedného parametru do odpovede od servera. Výhoda tohto spôsobu je plná prispôsobiteľnosť posielaných informácií. K ním môže patriť napríklad najnižšia možná spustiteľná verzia modulu, ktorá bude ovplyvnená napríklad zmenou v API. Nevýhoda spočíva v nutnosti manuálne aktualizovať číslo verzie pri vydaní aktualizácie. Zároveň však tento návrh bude nutné prediskutovať s kolegom Martinom Kameníkom, ktorý vyvíja server.



Obr. 3.2: Wireframe návrhu zobrazenia informácií o dostupnom module, ktorý nie je nainštalovaný, prípadne nie je aktualizovaný. (Autor použitých ikon – Icons8)

Z týchto dvoch možností navrhujem pre túto činnosť použiť vlastný server z dôvodu škálovateľnosti a nezávislosti na štruktúre stránky Google Play. Zároveň zavedenie minimálnej podporovanej verzie modulu uľahčí vývoj v tomto rannom štádiu, kedy ešte veľká časť projektu je stále nestabilná a prejde zmenami.

Na obrázku 3.2 je možné vidieť znázornenie dostupného modulu, ktorý nie je nainštalovaný (moduly *Tvary* a *Rotace*) a modulu, ktorý je nainštalovaný, ale nie v poslednej verzii (modul *Body*).

3.3 Sprievodca aplikáciou

Medzi novými požiadavkami sa nachádza aj sprievodca aplikáciou, ktorý používateľom predstaví jednotlivé časti aplikácie a v prípade nejasností im vysvetlí v akom stave sa nachádzajú a aké sú ďalšie možnosti pohybu v aplikácii. Samotný sprievodca bude mať podobu animovaného Dráčka, ktorý bude s užívateľom komunikovať buď formou textu, animácie, zvukových pokynov alebo akejkoľvek kombinácie týchto troch. Keďže medzi používateľov aplikácie môžu patriť aj negramotné deti, je vhodné použiť namiesto textovej nápovedy buď ľudský hlas, alebo dostatočne jasné animácie (napr. postupnosť blikajúcich prvkov na hracej ploche).

Keďže Dráček funguje modulárnym systémom (jadro + moduly, ako je možné vidieť na začiatku analýzy na obrázku 2.1), nastáva otázka šírky využitia tejto funkcionality. Možnosti využitia sprievodcu dosahujú v samotných

moduloch/hrách väčšie opodstatnenie ako v jadre z dôvodu rozličnosti hier a herných postupov. Jedným zo spôsobov, ako túto funkcionalitu sprístupniť pre všetky moduly, je použitie spoločnej knižnice. Takáto knižnica by mohla obsahovať všetky triedy a zdroje potrebné k fungovaniu nápovedy. V nasledujúcich sekciách sa budem venovať viacerým častiam knižnice, ktorých implementácia je dosiahnuteľná viacerými možnosťami.

3.3.1 Základný návrh

V tejto časti načrtnem základnú ideu rozdelenia a návrhu tried. Tento návrh bude po vyhodnotení nasledujúcich častí upresnený na konci tejto sekcie.

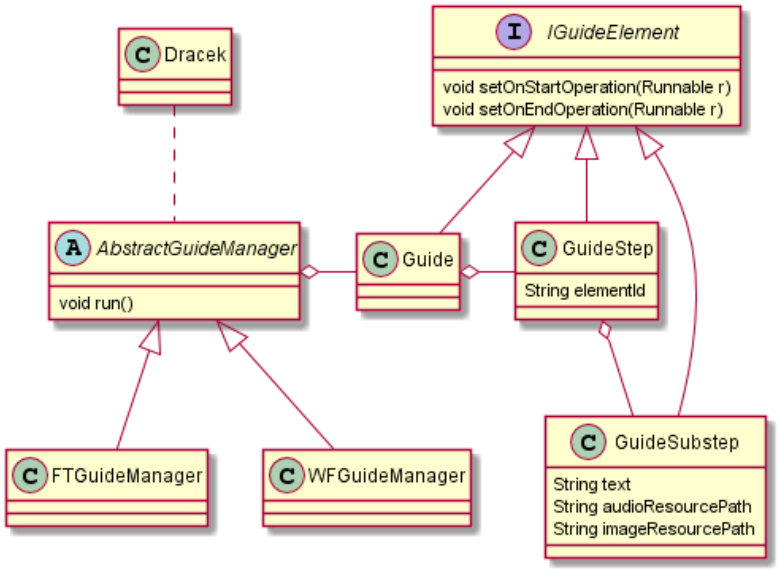
Sprievodca aplikáciou bude fungovať dvojako: jednak ako sprievodca prvým spustením a jednak ako nápoveda pre užívateľa. Sprievodca prvým spustením je spustený užívateľom ideálne raz a má za úlohu previesť užívateľa základnými ovládacími prvkami na obrazovke. Aktuálny stav užívania aplikácie nemá na spustenie žiadny vplyv. Nápoveda je na druhú stranu závislá od toho, čo užívateľ do času spustenia vykonal. Pre uľahčenie pochopenia budem prvky týkajúce sa prvého spustenia označovať skratkou FT (first time/prvý krát) a prvky nápovedy skratkou WF (workflow/pracovný postup). Základný návrh vyzerá nasledovne:

Každá upravená Android Activity[2]má vlastného FT a WF manažéra, ktorý v sebe obsahuje konkrétnu definíciu sprievodcu. Sprievodca (Guide) je prvý základný prvok a obsahuje kroky. Krok (Step) je entita, ktorá obsahuje id grafického prvku na obrazovke, ktorého sa daný krok týka. Na základe tejto informácie tak bude avatar Dráček presunutý na lokalitu v blízkosti prvku. Krok obsahuje neprázdny zoznam podkrokov, ktorými sa v rámci jedného kroku prechádza. Podkrok (Substep) v sebe nesie textovú, zvukovú alebo vizuálnu informáciu. Akákoľvek kombinácia týchto informácií je možná. Každý z troch základných prvkov obsahuje aj možnosť spustiť ľubovoľný počet metód pred začatím a po skončení konkrétneho prvku. Na obrázku 3.3 je možné vidieť prvotný návrh architektúry tejto funkcionality.

3.3.2 Ukladanie spoločných zdrojov

Medzi zdroje obsiahnuté v knižnici pre OS Android sa podľa [41] radia napríklad reťazce (strings), rozloženia (layouts), grafické prvky (drawables) a špeciálny druh zdrojov – asset (kvôli absencii vhodného prekladu názov ponechaný v AJ). Zdroje sú náročné na úložný priestor – už len v práci Karla Kovařovica [11] je možné vidieť plánovanie niekoľkých variant animovanej postavičky Dráčka, čo znamená množstvo kombinácií statických obrázkov a animácií. V nadväznosti na tento problém rozoberiem dve možnosti, akým je možné uložiť spoločné zdroje potrebné k fungovaniu knižnice:

[2]Activity je Java trieda reprezentujúca jednu obrazovku v Androide majúca svoj vlastný životný cyklus. Viac na https://www.tutorialspoint.com/android/android_acitivities.htm.



Obr. 3.3: Prvotný návrh tried pre sprievodcu aplikáciou

Uloženie zdrojov v knižnici

Ako som už načrtol, jedna možnosť je uchovávať zdroje priamo v knižnici. Pri každej kompilácii modulu alebo jadra sa zostaví posledná verzia knižnice, ktorá je následne pripojená k modulu/jadru v jednom inštalačnom balíčku. Táto možnosť má svoje výhody v aktuálnosti použitých zdrojov a jednoduchšej implementácii. Avšak problém, ktorý vidím ako kritický, je už spomínaná robustnosť grafických zdrojov. Ak sa k tomu pripočítajú aj zvukové zdroje (zvukové pokyny), veľkosť každého modulu môže stúpnuť až o niekoľko desiatok MB.

Uloženie zdrojov v jadre

Druhá možnosť návrhu spočíva v uložení zdieľaných zdrojov v jadre a následnom pristupovaní k týmto zdrojov z modulov. Na Android je problém zdieľania súborov medzi aplikáciami vyriešený použitím triedy FileProvider[3], ktorej použitie je pomerne jednoduché na implementáciu. V knižnici tým pádom bude naimplementovaná logika prístupu k týmto súborom. Výhody tejto možnosti spočívajú v znížení veľkostí modulov a väčšej konzistentnosti zdrojov (rovnaká verzia obrázku pre každý modul). Väčší zásah do prepojenia medzi knižnicou a zdrojmi v jadre môže vyvolať nepriaznivé správanie neaktualizovaného modulu. Spolu s

[3]Android trieda umožňujúca sprístupniť súbory jednej aplikácie ostatným. Prijímajúca aplikácia získava prístup k obsahu súboru bez toho, aby získal prístup k súboru samotnému. Viac na https://developer.android.com/reference/android/support/v4/content/FileProvider.html.

nutnosťou udržovať jadro aktuálne pri pridaní alebo zmene zdrojov to beriem ako nevýhody tejto možnosti.

Z týchto dvoch možností vyberám pre implementáciu funkcionality druhú možnosť – uloženie spoločných zdrojov v jadre. Je to z dôvodu šetrenia voľným miestom na zariadení deduplikáciou grafických prostriedkov. Zdroje využívané iba na jednom mieste v rámci celého systému je vhodné ukladať do príslušného inštalačného balíčka.

3.3.3 Vstup pre sprievodcu

Medzi časti funkcionality, ktoré je možné implementovať viacerými spôsobmi, patrí aj vstup pre sprievodcu. Pod týmto pojmom si čitateľ môže predstaviť atribúty znázornené na obrázku 3.3, ako napríklad text zobrazený používateľovi, prípadne id grafického prvku na obrazovke. Tento vstup môže byť v budúcnosti často vytváraný ľuďmi, ktorí nie sú programátorsky schopní, a preto sa zameriam hlavne na jednoduchosť pridávania a úpravy návodov. V nasledujúcich riadkoch rozoberiem možnosti, akými sa dá k implementácii tejto časti pristupovať:

Vstup v zdrojovom kóde
Prvá z možností je definovanie obsahu sprievodcu priamo v kóde aplikácie, konkrétne v triedach Activity. Princíp spočíva v použití návrhového vzoru Builder[4], vďaka ktorému je výsledný kód prehľadnejší. Medzi výhody tejto možnosti patria jednoduchosť, dobrá prispôsobiteľnosť a prakticky neobmedzené možnosti pre triedu objektu vstupného parametru funkcií. Avšak ako som avizoval vyššie, hlavné kritérium výberu je jednoduchosť manipulácie so vstupmi. Keďže na vývoji sa budú často podieľať ľudia bez programátorských zručností, táto možnosť sa javí ako neprívetivá.

Vstup z osobitného súboru
Druhá možnosť je zameraná na splnenie hlavného kritéria spomínaného vyššie za pomoci vopred definovaného spôsobu zápisu dát. Medzi dva známe spôsoby patria JSON a XML, ktoré sú dobre spracovateľné počítačom. Obidva spôsoby poskytujú dobrú čitateľnosť pre vývojára a oproti predchádzajúcemu spôsobu odoberajú nutnosť používať samotný kód k úprave vstupov. Medzi nevýhody však patria nutnosť tieto súbory spracovať a obmedzenosť dátových typov, ktoré môžu byť využité. Pre použitie tejto možnosti je nutný výber jedného spôsobu z dvojice XML a JSON:

[4]Builder je návrhový vzor, ktorý rieši konštrukciu komplexných objektov systémom krok po kroku. Viac na https://www.tutorialspoint.com/design_pattern/builder_pattern.htm.

XML
Formát XML je oproti JSON obsiahlejší kvôli nutnosti používať uzatváracie značky. Medzi jeho výhody podľa [42] patria možnosti transformácie dokumentu (napr. kvôli zmene verzie) pomocou XSL a jednoznačné určenie štruktúry dokumentu pomocou XML Schema. Keďže vstup sprievodcu sa radí medzi zdroje Android aplikácie, považujem použitie formátu XML ako výhodu v ponechaní integrity formátu zdrojov, keďže veľká väčšina ostatných zdrojov je zapísaná vo formáte XML.

JSON
Formát JSON má oproti XML podľa [43] výhody vo veľkosti, rýchlosti čítania a zápisu a možnosti využitia polí. Keďže tieto výhody majú svoje uplatnenie hlavne vo webovej komunikácii, nemajú v tomto prípade veľký dopad na zvýšenie výkonu aplikácie.

Zachovanie integrity a použitie XML Schema považujem za dve veľké výhody XML, kdežto výhody JSON sú v používaní tejto funkcionality zanedbateľné. Ako formát vstupu pre sprievodcu v tejto možnosti volím XML.

Kombinácia predchádzajúcich dvoch možností
Obe možnosti majú svoje výhody a nevýhody a keďže sa vzájomne nevylučujú (súbor z druhého spôsobu bude spracovaný na ekvivalent z prvého spôsobu), je možné tieto možnosti spojiť a využiť tak výhody oboch. Po spracovaní vstupného súboru na inštanciu triedy v Jave je možné doplniť chýbajúce atribúty programátorsky. Medzi tieto atribúty tak môžu patriť napríklad funkcie, ktoré sa spustia pred/po určitom kroku alebo podkroku. Podobným spôsobom je možné na platforme Android pracovať s grafickými prvkami, kedy sa v súbore rozloženia definuje prvok a následne sa k nemu v kóde definuje onClick funkcia (kód ktorý sa vykoná po kliknutí na prvok). Príklad je možné vidieť na [44]. Nevýhoda tejto možnosti je hroziaca roztrieštenosť definícií prvkov v súbore a v kóde.

Tretia možnosť zachováva výhody prvých dvoch možností a pridáva nevýhodu, ktorá však pri organizovanom vývoji kritická nie je. Konzistentnosť a návrhový postup podobný fungovaniu v systéme Android považujem ako ďalšiu výhodu. Z tohto dôvodu volím ako najvhodnejšiu tretiu možnosť – tvorba základných vstupov vo formáte XML, ktoré budú následne obohatené o ďalšie funkcie priamo v kóde. V budúcnosti je možné zaoberať editorom vstupov s grafickým rozhraním pre jednoduchosť používania. Tento nápad je však nad rámec rozsahu tejto práce.

3.3.4 Grafická podoba sprievodcu

Grafické náčrty približného chovania sprievodcu je možné vidieť na obrázkoch 3.4, 3.5 a 3.6. Ako avatar je použitý obrázok z práce Karla Kovařovica [11] a tento avatar je vyobrazený v niekoľkých stavoch. Stav na obrázku 3.4 vykresľuje Dráčka v základnej pozícii – vpravo dole, nevykonávajúci žiaden proces, čakajúci na používateľovu akciu. Po spustení nápovedy sa Dráček presúva pomocou animácie na prvý krok, ktorý je spojený s určitým elementom na obrazovke. Užívateľ má možnosť ďalej postupovať nápovedou, alebo nápovedu zrušiť dvoma tlačidlami. Na obrázku 3.5 je to element pre textový vstup prihlasovacieho mena. Dráček sa nachádza hneď vedľa elementu a zobrazuje text z podkroku. Ak Dráček práve zobrazuje posledný podkrok posledného kroku, graficky sa tento podkrok odlíši napríklad zmenou ikony ako na obrázku 3.6.

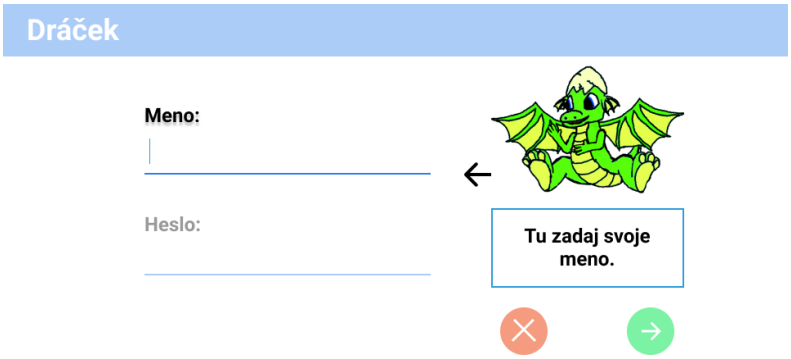
Pohyb Dráčka po obrazovke bude zabezpečovať špeciálna trieda, ktorá pozíciu vedľa želaného vypočíta a na dané miesto ho pomocou animácie presunie. Krok vypočítavania pozície bude musieť rátať s rôznymi relatívnymi pozíciami voči elementu, ak predvolená pozícia (vpravo) by spôsobila, že Dráček sa presunie mimo obrazovku.

Obr. 3.4: Náčrt základného stavu sprievodcu. Dráček je k dispozícii.

3.3.5 Akcie sprievodcu

WF a prípadne aj FT nápovedu bude nutné určitým spôsobom spúšťať akciou od užívateľa. V súčasnom stave aplikácie sa akciou užívateľa vyvoláva Dialog[5], ktorý užívateľa informuje o aktuálnej obrazovke. Ikona tejto akcie sa nachádza

[5]Dialog je vyskakovacie okno, ktoré nezaberá celú obrazovku a zobrazuje sa v strede. Zvyčajne užívateľa informuje o nejakej skutočnosti alebo žiada od neho akciu. Viac na https://developer.android.com/guide/topics/ui/dialogs.html.

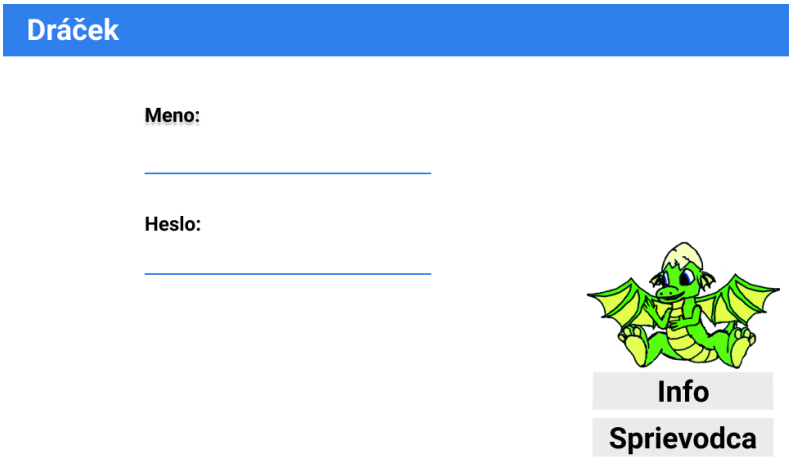


Obr. 3.5: Náčrt sprievodcu ukazujúceho na prvok na zadanie mena. Ďalší krok je k dispozícii.

Obr. 3.6: Náčrt sprievodcu ukazujúceho na prvok na zadanie hesla. Je to zároveň posledný krok.

v pravom hornom rohu, ako je možné vidieť na obrázku 2.2. Pridávanie ďalších ikon týmto štýlom nemusí byť pre užívateľa intuitívne a so zobrazenými grafickými prvkami by sa malo (hlavne u malých detí, ktorým to môže odvádzať pozornosť) šetriť. Na základe tohto problému vznikol nápad využiť avatar aj ako menu akcií, ktoré nebudú v základnom stave zobrazené. Rôzne Android Activity môžu poskytovať rôzne akcie pre používateľov (napríklad Info popísané vyššie je spoločná akcia pre všetky Activity). Tieto akcie sa budú pridávať práve do menu akcií sprievodcu a toto menu bude možné vyvolať kliknutím

na Dráčka. Grafický náčrt otvoreného menu je možné vidieť na obrázku 3.7.



Obr. 3.7: Náčrt menu akcií sprievodcu.

3.4 Grafická knižnica

Jedným z nedostatkov aplikácie preberaných v časti 2.3.1 je nedostatočné grafické rozhranie aplikácie. Keďže, ako som spomínal v časti 1.1, na práci nemám možnosť pracovať s priradeným dizajnérom, je tento nedostatok riešiteľný v tejto práci len čiastočne. Aktuálny problém grafického rozhrania spočíva aj v nekonzistentnosti použitých elementov v rámci systému (učiteľská aplikácia, žiacka - jadro, moduly) a aj v rámci jednotlivých aplikácií.

Tento problém sa dá riešiť vytvorením spoločných grafických prvkov v jednej spoločnej zdieľanej knižnici a ich následným použitím v jednotlivých aplikáciách. Taktiež je možné vytvoriť jednotný grafický štýl (Style[6]) a tému (Theme[7]), ktoré budú tieto aplikácie používať. Užívateľ aj napriek modulárnemu systému aplikácií nemusí spoznať, že sa nachádza v inej aplikácii.

Z dôvodu absencie dizajnéra na projekte sa nebudem zaoberať širokým spektrom spoločných elementov. Navrhnem a v aplikácii použijem 2 elementy, ktoré budú slúžiť ako príklad pre ďalšie spoločné elementy vyvíjané nasledujúcimi iteráciami. Tieto dva elementy budú konkrétne **Toolbar** a **Dialog**, ktoré sú v aktuálnom stave používané v každej Activity jadra.

[6]Style je kolekcia atribútov, ktoré špecifikujú dizajn elementu. Viac na https://developer.android.com/guide/topics/ui/look-and-feel/themes.html.

[7]Theme je špeciálny druh štýlu (Style), ktorý aplikuje atribúty na všetky elementy aplikácie. Viac na https://developer.android.com/guide/topics/ui/look-and-feel/themes.html.

Toolbar bude ladený do zelenej farby (zmena tejto farby bude možná) a pri offline stave aplikácie (bez pripojenia k internetu) sa bude zobrazovať v červenej farbe spolu s informačným nápisom. Tlačidlá akcií sa presunú do menu akcií Dráčka a jediné zostávajúce tlačidlo bude akcia *Späť*.

Dialog bude taktiež ladený do zelenej farby a okrem štandardného obsahu (nadpis, text a tlačidlá) bude tiež obsahovať aj avatar Dráčka. Tento avatar tak bude vytvárať dojem komunikácie s Dráčkom.

Pri vytváraní týchto prvkov budem dbať na možné neskoršie zmeny navrhnuté dizajnérom.

3.5 Google Play

V kapitole Analýza v časti 2.4.1 som popisoval platformu Google Play, aké služby je možné využiť pri umiestnení aplikácie na Google Play a aké majú výhody. Tieto výhody boli dostatočne lákavé na to, aby sa táto idea pridala do nefunkčných požiadaviek a postúpila do kapitoly Návrh.
Spoločnosť Google sa dostatočne venuje písaniu dokumentácie Androidu, jeho elementov, príkladov použitia, alebo návodov v štýle *všetko v jednom*. Medzi takéto návody patrí napríklad aj zoznam procesov, ktorý je vhodný vykonať pred umiestnením aplikácie na Google Play. Z tohto zoznamu vyberiem a okomentujem niekoľko procesov, pričom celý zoznam je možné nájsť na [45].

Pripraviť vývojársky účet
K splneniu tohoto kroku je nutné vytvoriť Google účet, pod ktorým sa bude aplikácia vydávať. Aby bolo možné umiestniť aplikáciu na Google Play, je nutné zaplatiť jednorazový registračný poplatok vo výške $25 USD.

Vyhodnotiť splnenie pokynov týkajúcich sa kvality
Google poskytuje zoznam kvalitatívnych vlastností, ktoré by mala aplikácia spĺňať. Okrem základných vlastností je možné nájsť aj vlastnosti pre aplikácie určené na tablet, Wear OS (smart hodinky), Android TV a Android Auto. V našom prípade by mala aplikácia spĺňať jednak základné vlastnosti a jednak aj rozšírené vlastnosti pre tablet.

Skompilovať APK súbory pre rôzne vydávacie kanály
V prípade produkčného kanálu to znamená medziiným aj obmedzenie správ, ktoré sa posielajú do logu. Okrem tohto kanálu poskytuje Google Play aj alpha a beta kanál. Rovnaké obmedzenia ako pre produkčný kanál by mali platiť aj pre beta kanál. Alpha kanál môže byť ustanovený na využitie vývojármi a môže tým pádom ostať v debug móde.

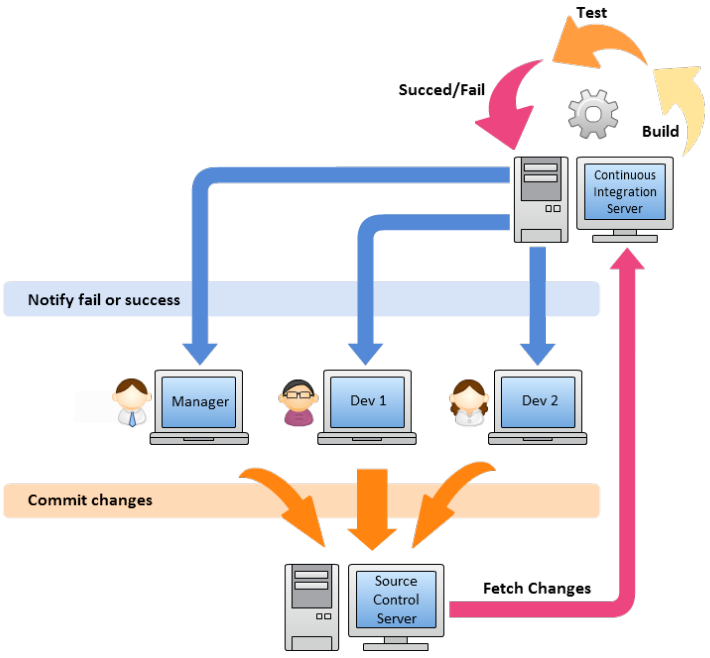
Ak sú tieto procesy splnené, aplikácia je pripravená na umiestnenie na Google Play. V čase písania návrhu je však otázne, či bude vývojársky účet dostupný

do ukončenia realizačnej fázy. V kladnom prípade budú tieto procesy vyhodnotené v kapitole Implementácia.

3.6 Spolupráca s dizajnérom

V kapitole Analýza v časti 2.5 som popisoval, ovládanie akých nástrojov a technológií je k práci na projekte Dráček potrebné. V tejto časti vyberiem a popíšem niekoľko možných spôsobov spolupráce dizajnéra a vývojára. V týchto spôsoboch sa okrem skratky **AS** pre Android Studio bude používať aj skratka **CI** pre Continuous Integration.

Continuous Integration je postup softwarového vývoja, v ktorom členovia vývojárskeho tímu integrujú svoje prídavky (inkrementy) často (aj niekoľko krát za deň) za účelom znížiť riziko konfliktov vzniknutých spájaním inkrementov práce. Inkrementy sú po automatickej kompilácii podrobené testom na serveri.
Na projekte Dráček fungujú základy CI, ktoré založil bývalý člen tímu – Jaroslav Kravec. Aktuálny stav poskytuje po každom commite automaticky výstup v podobe APK súborov dostupných v repozitári. Tieto súbory sú pripravené na inštaláciu na Android zariadenie.



Obr. 3.8: Znázornenie životného cyklu CI podľa [1].

Nasledujúce spôsoby predstavujú jednu iteráciu aplikovania zmien dizajnérom. Táto iterácia spočíva v:

1. nájdení súboru určeného k zmene
2. úprave súboru
3. kompilácii
4. uložení a commite
5. spustení na zariadení

Spôsoby vykonania iterácie môžu byť nasledovné:

Bez AS, s CI
Tento spôsob je najmenej náročný na výkon PC dizajnéra a je postavený hlavne na funkčnom CI. Nájdenie súboru, uloženie a commit sú bez AS výrazne pomalšie, pričom kompiláciu v tomto spôsobe úplne vylučujem. Bez AS musí dizajnér pracovať s nástrojom git a zmeny je možné skontrolovať až po úspešnom automatickom zostavení na serveri a následným stiahnutím a nainštalovaním výsledného APK súboru na vlastné zariadenie. Tento proces je pomerne zdĺhavý.

AS doplnené CI
Použitím AS sa výrazne ušetrí čas na projekte vďaka nástrojom, ktoré sú v ňom obsiahnuté. V časti Analýza som popisoval napríklad použitie grafického editoru rozloženia a emulátoru. Zatiaľ čo prvý nástroj z uvedených umožňuje náhľad vykonaných zmien bez nutnosti kompilácie a spúšťania aplikácie, emulátor pomáha vyskúšať si chovanie zmien na viacerých veľkostiach obrazovky a verziách systému. Medzi ďalšie výhody patrí napríklad aj vstavaný grafický nástroj git. Dizajnér bude týmto spôsobom commitovať na server vizuálne skontrolované zmeny a predíde sa tak veľkému množstvu commitov venujúcich sa jednej zmene. Tento spôsob však má svoju nevýhodu hlavne v požiadavkách na výkon PC dizajnéra. Emulátor a editor rozloženia spotrebúvajú pomerne veľké množstvo systémových zdrojov a miesta na disku.

Emulátor bežiaci na serveri
Posledný spôsob, ktorý uvediem, slúži skôr ako doplnok pre predchádzajúce možnosti. Keďže možnosť automaticky zostaviť aplikáciu na serveri existuje, je rovnako možné túto aplikáciu nasadiť automaticky. V princípe by to znamenalo bežiaci emulátor na serveri, ku ktorému by existoval vzdialený prístup napríklad pomocou protokolu vnc. Na tento emulátor by v prípade úspešného zostavenia boli nainštalované všetky aplikácie a dizajnér by tak mohol zmeny vidieť aj bez nutnosti vlastniť Android zariadenie, či používať emulátor. Zriadeniu tejto možnosti sa však kvôli obmedzenému rozsahu práce venovať nebudem.

Zo zostávajúcich dvoch možností odporúčam pre prácu na projekte druhú možnosť a to použitie IDE Android Studio. Návod ako IDE pripraviť a nastaviť na projekt Dráček rozpíšem v kapitole Implementácia.

Kapitola 4

Implementácia

V tejto kapitole čitateľa zoznámim s výslednou podobou zmien popísaných v kapitole Návrh. Táto kapitola bude zároveň slúžiť ako manuál pre ďalšie iterácie tohto projektu – hlavne pre budúcich kolegov, ktorí budú na moju prácu naväzovať. Vyhnem sa popisovaniu kódu, ktoré by zabralo niekoľko strán a v konečnom dôsledku by bolo neprehľadné. Namiesto toho využijem nástroj Javadoc, ktorý z komentárov v zdrojovom kóde vytvorí HTML stránky obsahujúce dokumentáciu kódu. Stránky budú uložené na CD nosiči priloženom k tejto práci.

4.1 Priebeh vývoja

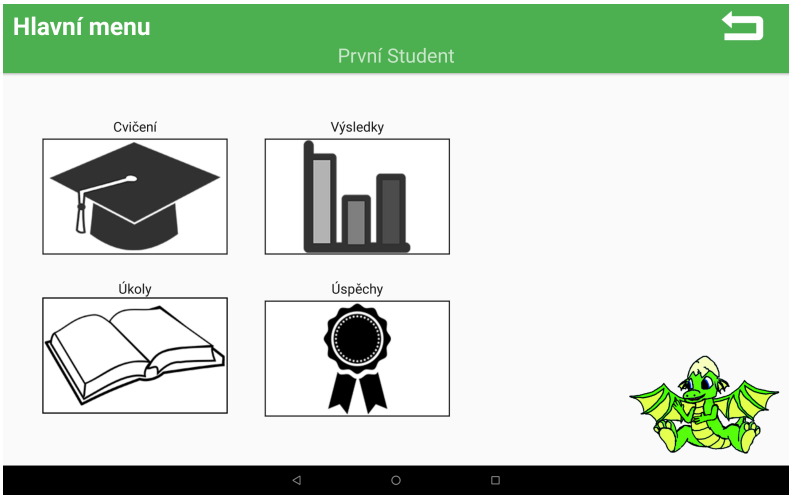
Vývoj prebiehal na mojom osobnom počítači bežiacom na operačnom systéme Windows 10. Primárny nástroj použitý k programovaniu bol Android Studio a zmeny boli testované jednak na virtuálnom zariadení Android (veľkosť obrazovky 10 palcov, Android API P) a jednak na zapožičanom tablete Google Nexus 10 s verziou Androidu 8.1. Vizuálne výsledky práce je možné vidieť na obrázkoch 4.1, 4.2, 4.3, 4.4 a 4.5.

4.2 Podoba systému

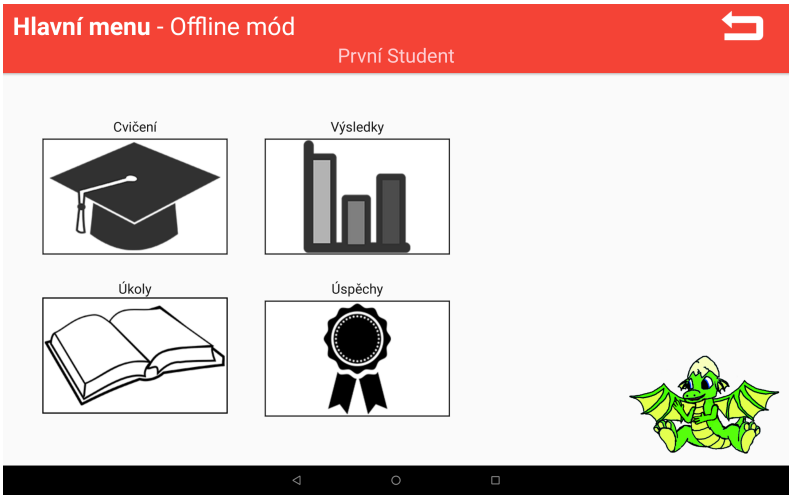
Aktuálna podoba systému sa oproti verzii aktuálnej pred začatím tejto práce (viď obr. 2.1) líši len minimálne. Jediná zmena oproti starej verzii je pridanie spoločnej knižnice *commonlibrary* ako závislosť jednak aplikácie Jadra a jednak knižnice Core Library.

4.3 Vývojárska príručka

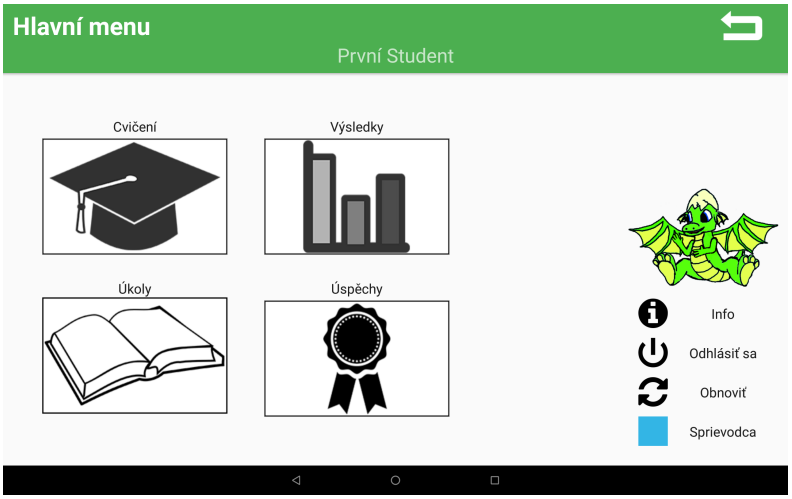
V tejto sekcii prenechám svojim nástupcom na tomto projekte svoje skúsenosti a rady, ktoré môžu pomôcť k urýchleniu vývoja aplikácie. V texte sa bude



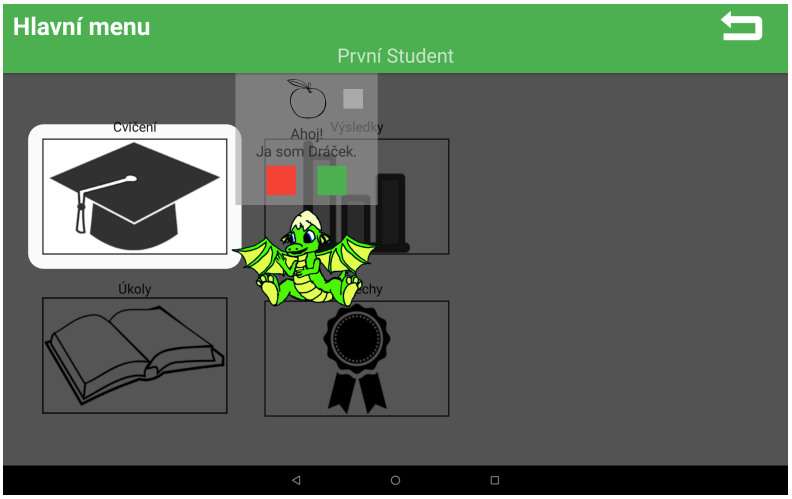
Obr. 4.1: Výsledná podoba: Základný vzhľad.



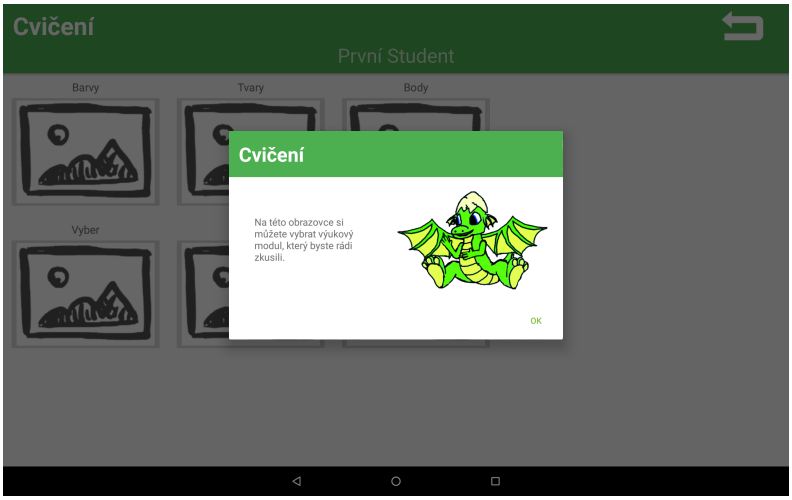
Obr. 4.2: Výsledná podoba: Signalizácia offline režimu.



Obr. 4.3: Výsledná podoba: Menu akcií Dráčka.



Obr. 4.4: Výsledná podoba: Dráček sprevádza aplikáciou. Zobrazuje sa vlastný obrázok a prehráva sa audio.



Obr. 4.5: Výsledná podoba: Nový vzhľad prvku Dialog.

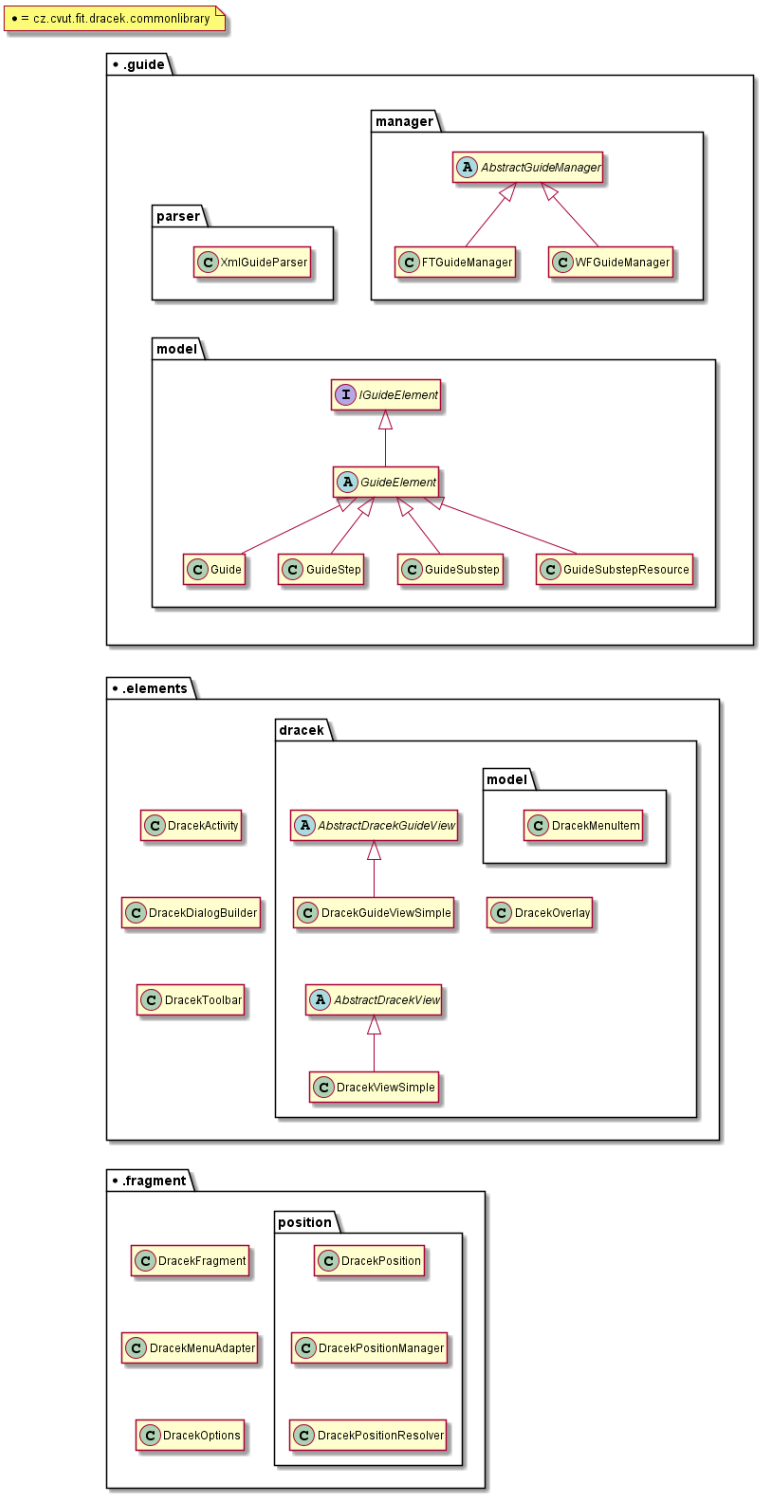
vyskytovať vývojársky slang a anglické slová, ktoré uľahčujú pochopenie a skracujú vysvetľovanie daného prvku.

4.3.1 Repozitár kódu

V čase, kedy sme začali na projekte Dráček spolu s tímom pracovať, existovalo niekoľko rôznych repozitárov kódu; každý z nich založený inou generáciou projektu. Zásluhou Jaroslava Kravca sa tieto projekty podarilo spojiť do jedného repozitára a pomocou systému Gradle medzi nimi jednoducho prepínať. Repozitár je momentálne prístupný na školskej stránke Gitlab na adrese https://gitlab.fit.cvut.cz/kravejar/dracek. Taktiež zásluhou Jaroslava Kravca funguje v repozitári forma Continuous Integration – build po každom commite. Artefakty – teda výstupné súbory – sú po úspešnom builde dostupné v jednom ZIP archíve.

4.3.2 Knižnica commonlibrary

Veľká väčšina kódu vyvinutého v tejto práci je obsiahnutá v knižnici commonlibrary. Mojou snahou pri vývoji tejto knižnice bolo do maximálnej možnej miery abstrahovať chovanie a podporovať mnohonásobné použitie bez duplicitného kódu. Keďže aplikácie boli do tejto doby vyvíjané samostatne, existovala len malá spoločná „základňa“ kódu. To ma priviedlo k nápadu vytvoriť spoločnú knižnicu ako základ pre aplikáciu jadra, všetkých modulov a do budúcna prípadne aj učiteľskú aplikáciu.
Štruktúra tejto knižnice je znázornená na obrázku 4.6. V krátkosti popíšem funkciu niektorých tried obsiahnutých v tejto knižnici.



Obr. 4.6: Štruktúra knižnice commonlibrary

.fragment
Balíček fragment je zameraný na logiku a správanie Dráčka v aplikácii. Základný kameň Dráčka je trieda DracekFragment, ktorá ako názov napovedá, dedí od triedy Fragment. Pri vytváraní fragmentu je posielaná inštancia triedy DracekOptions, ktorá obsahuje konfiguráciu pre Dráčka vytvorenú v Activity. Pozíciu Dráčka na obrazovke spravuje trieda DracekPositionManager, ktorá nové miesto presunu získa od triedy DracekPositionResolver. Trieda DracekMenuAdapter slúži na vytvorenie menu akcií Dráčka.

.elements
Balíček elements je zameraný na grafické prvky aplikácie. DracekActivity je trieda, od ktorej dedia všetky Activity naprieč modulmi a celou aplikáciou jadra. DracekDialogBuilder a DracekToolbar sú triedy, ktoré vznikli na základe navrhnutých grafických prvkov v predchádzajúcej kapitole. AbstractDracekView a AbstractDracekGuideView predstavujú abstraktné triedy pre avatar Dráčka, resp. rozloženie pre sprievodcu. Obe triedy majú svoje konkrétne triedy s koncovkou Simple, ktoré slúžili na vývoj a po grafickej stránke sú nedostačujúce.

.guide
Balíček guide pozostáva z tried, ktoré slúžia ako logika sprievodcu. Trieda XmlGuideParser slúži na parsovanie XML súboru, z ktorého sa vytvoria inštancie tried z balíčku model. AbstractGuideManager slúži hlavne na spustenie sprievodcu.

4.3.3 Inštalačná príručka

Knižnica je v aktuálnom stave kódu pridaná do všetkých modulov, keďže bola pridaná ako závislosť pre knižnicu *corelibrary*, ktorá slúži primárne na komunikáciu modulov s jadrom. V prípade že knižnicu bude vhodné do inej časti projektu pripájať, je nutné pridať nasledujúci riadok do *build.gradle* súboru konkretného Gradle[8]modulu:

```
implementation project(':core:commonlibrary')
```

Ako postupovať pri integrácii jednotlivých častí knižnice popíšem nižšie v príslušných sekciách.

4.3.4 Vytváranie novej Activity

Vytváranie novej Activity by malo prebiehať jedným z dvoch spôsobov - novo vytvorená trieda bude priamym alebo nepriamym potomkom triedy DracekActivity. Po analýze kódu, ktorý som mal k dispozícii na začiatku tejto

[8]Gradle je open-source build nástroj, ktorý sa zameriava na automatizovanie procesov a je podporovaný IDE Android Studio.

práce, bolo mnoho metód duplicitných, prípadne len ľahko obmenených. Radím svojim nástupcom, aby sa tomuto spôsobu vyhli. Nová Activity trieda by mala v svojom XML layout súbore obsahovať <include> na Toolbar a FragmentContainer (viď Activity v Core aplikácii). Odporúčaným top layoutom je CoordinatorLayout. V súbore AndroidManifest.xml konkrétneho modulu je možné nastaviť tému Activity na CommonAppTheme a CommonAppTheme.FullScreen pre klasickú spoločnú a prípadne aj full screen tému.

4.3.5 Vytváranie obsahu pre sprievodcu

Vstup pre sprievodcu sa vytvára a následne spracováva vo formáte XML. V zdrojovom kóde je obsiahnutý vzorový vstup, ktorý sa spúšťa v hlavnej Activity jadra. Vstup má niekoľko prvkov, ktorých štruktúra je nasledovná: Každý

```
1..1 <guides>
├─ 1..1 <FTGuide>
│  └─ 1..* <step>
│     └─ 1..* <substep>
│        ├─ 0..1 <text>
│        ├─ 0..1 <img>
│        └─ 0..1 <audio>
└─ 1..1 <WFGuides>
   └─ 1..* <WFGuide>
      └─ rovnako ako vo FTGuide
```

prvok štruktúry okrem koreňového má nepovinný atribút ID, pomocou ktorého sa dá konkrétny prvok vyhľadať. To je vhodné napríklad pri použití triedy ElementListener, ktorá umožňuje spúšťať kód pri začiatku a konci konkrétneho prvku. Pri prvku <(WF|FT)Guide> je to spustenie a koniec sprievodcu, pri prvku <substep> je to klik užívateľa na tlačidlo nasledujúceho kroku a pri prvku <step> je to splnenie všetkých podkrokov.

4.3.5.1 Vstupný súbor

Trieda DracekActivity obsahuje metódu, ktorá, ak je to možné, načíta a parsuje súbor uložený na defaultnej ceste uložený pod defaultným názvom. Táto cesta je:

```
res/raw/guide_*.xml
```

kde * reprezentuje názov triedy v nasledujúcom formáte (pre príklad použijem vstupnú triedu MainMenuActivity.class):

- **Odstránenie prípony súboru**
  => MainMenuActivity

- **Odstránenie prípony *Activity***
  => MainMenu
- **Snake case**
  => main_menu
- **Pridanie predpony *guide_* a prípony *.xml***
  => guide_main_menu.xml

Ak je tento súbor existuje, je parsovaný a použije sa ako vstup pre sprievodcu Dráčka.

4.3.5.2 Zdroje

Sprievodca Dráček bol implementovaný s podporou pre tri rôzne typy zdrojov: textový, obrázkový a zvukový. Všetky tieto zdroje je možné používateľovi sprístupniť v konkrétnom podkroku. Textový zdroj funguje pomerne jednoducho - text z XML súboru sa použije ako text v podkroku. Pri zvyšných dvoch typoch je to však zložitejšie, keďže sa v XML deklaruje cesta k súboru, ktorý má byť použitý. Tento súbor môže byť uložený dvoma spôsobmi a tie odlišuje atribút *common* prvkov <audio> a <img>. Oba spôsoby majú spoločnú štruktúru uloženia súborov a to:

```
assets/guide/audio
assets/guide/img
```

Čo ich však rozdeľuje, je umiestnenie *assets* zložky. Ak je common atribút nastavený na hodnotu **false** (čo je jeho základná hodnota), použije sa *assets* zložka v konkrétnej aplikácii. V opačnom prípade (**true**) sa do zložky *assets* pristupuje cez FileProvider poskytovaný aplikáciou Core. Iným slovami, hľadá sa v zložke *assets* projektu *Core*.
Tento spôsob umožňuje viacnásobné použitie jedného súboru, avšak prináša so sebou nutnosť aktualizovať Core pri pridaní nového *common* zdroja.

4.3.6 Nedokončené časti

Medzi vylepšeniami, ktoré som rozpisoval v kapitole Návrh, sa nachádzajú aj také, ktoré sa v priebehu realizácie ukázali ako momentálne nesplniteľné. Tieto časti rozpíšem nižšie s cieľom uľahčiť prácu nasledujúcim iteráciám projektu.

Prihlásenie žiaka cez učiteľskú aplikáciu
(viď časť 3.1)
V dobe realizovania návrhu ešte nebola nová serverová aplikácia mojím kolegom Martinom Kameníkom vyvinutá. Keďže táto zmena vyžadovala zásah aj do serverovej časti projektu, nechal som toto vylepšenie v návrhovej fáze kvôli prípadným budúcim vylepšeniam v návrhu.

Grafické znázornenie stavu nainštalovaných modulov
(viď časť 3.2)
Podobne ako pri predchádzajúcom rozšírení aplikácie bolo aj toto ponechané v návrhovej fáze kvôli neprítomnosti nového serveru v čase realizácie.

Google Play
(viď časť 3.5)
Integráciu aplikácie do služby Google Play nebolo možné vykonať z dôvodu chýbajúceho vývojárskeho Google účtu. Aj napriek snahe môjho vedúceho práce, pána Ing. Jiřího Chludila, sa tento platený účet nepodarilo získať do uzavretia implementačnej fáze mojej práce.

Dva režimy aplikácie
(viď časť 2.4.2)
Implementácia dvoch režimov aplikácie sa javila zo začiatku ako dobrá možnosť zabezpečenia aplikáciu voči neoprávnenému prístupu. Po dlhšom uvážení a plánovaní do budúcna som nastavil tomuto nápadu oveľa nižšiu prioritu ako bolo pôvodne zamýšľané. Je to z dôvodu nedostatočnej praktickosti tohto nápadu. Ak by dieťa zamklo obrazovku, bolo by nutné aby učiteľ obiehal celú triedu a žiaka prihlásil (počítam s prípadom, kedy dieťa ešte nie je schopné zadať svoje prihlasovacie údaje samo). Tento nápad do budúcnosti však úplne nezahadzujem a navrhujem implementovať ho po tom, ako sa do aplikácie integruje jednoduchší spôsob prihlásenia.

4.3.7 Tipy na vylepšenie

V tejto časti uvediem niekoľko nápadov na vylepšenie, ktoré mi boli známe už v priebehu písania analýzy a návrhu alebo vznikli počas realizácie návrhu. Okrem nedokončených časti popísaných vyššie medzi ne patria aj:

Grafické vylepšenia
Keďže som na tejto práci nemal možnosť spolupracovať s dizajnérom, grafická podoba aplikácie je stále nedostačujúca a je potrebné ju vylepšiť. Medzi tieto vylepšenia by mali patriť aj produkty mojej práce a to napríklad avatar Dráček a grafické vyobrazenie sprievodcu.

Umiestnenie na Google Play
Okrem aplikácie Core je vhodné umiestniť na Google Play zároveň aj ostatné moduly a vhodným spôsobom ich spravovať.

Synchronizácia v pozadí
Aktuálny spôsob synchronizácie aplikácie so serverom môže byť v niektorých prípadoch nedostačujúci. Namiesto toho, aby sa aplikácia synchronizovala so serverom až po spustení, je možné vytvoriť službu, ktorá

bude túto činnosť vykonávať na pozadí. K tomuto vylepšeniu odporúčam použiť už existujúce riešenia namiesto vytvárania nových.

Používanie Android best practices
Začiatky práce na projekte pre mňa boli náročné z dôvodu náročnej adaptácie na kód aj napriek predchádzajúcim skúsenostiam s touto platformou. Je pochopiteľné, že kvalita kódu od vývojára bez skúseností s platformou je nižšia, ale s pribúdajúcou robustnosťou projektu to predstavuje problém pre nových členov projektu. Uvedomujem si, že na kvalitu kódu neexistuje žiadne objektívne kritérium, ale odporúčam vyhľadať si best practices od dôveryhodných zdrojov na internete. Do rozumnej miery odporúčam aj používanie knižníc, ktoré môžu uľahčiť a urýchliť vývoj.

Dokumentácia
Podobne ako v predchádzajúci bod, aj tento sa môže časom začať prejavovať ako nedostatok projektu. Aj jednovetná poznámka môže novému členovi projektu výrazne urýchliť pochopenie kódu.

4.3.8 Prechod na nový server

Jednou z požiadaviek pre nasledujúcu generáciu bude pochopiteľne upraviť aplikáciu tak, aby fungovala spolu s novým serverom vyvýjaným Martinom Kameníkom. Preto poskytnem krátky a stručný návod, aké kroky budú potrebné k vykonaniu tejto úlohy. Krátke a stručné budú z dôvodu malej znalosti kódu, ktorý zabezpečuje komunikáciu so serverom.

Zistiť zmeny v API
Z práce Martina Kameníka [16] bude nutné získať zmeny parametrov HTTP požiadavky a odpovede.

Upraviť modelové triedy
Modelové triedy sa nachádzajú v balíčku *model/rest*. Je nutné, aby sa členské premenné týchto tried zhodovali s prichádzajúcimi odpoveďami zo servera.

Upraviť logiku
V prípade zmeny URL adresy servera je nutné upraviť triedu *RestApiGenerator*. Zmeny v dotazoch je nutné premietnuť do tried v balíčku *rest/api* a logiku spracovania odpovedí upraviť v triedach balíčku *rest/tasks*.

4.4 Dizajnérska príručka

V tejto sekcii uvediem základný postup pre nových účastníkov vývoja na platforme Windows. Tento postup pokryje kroky potrebné k vykonaniu zmien a

počíta so základnou znalosťou SW spomenutého v kapitole Analýza v časti 2.5. Granularita návodu bude zvolená väčšia ako pre zručného človeka, no dostatočne malá, aby bol návod stručný a výstižný.

Repozitár a Git

Repozitár s kódom je uložený na adrese

https://gitlab.fit.cvut.cz/kravejar/dracek.

Keďže sa jedná o repozitár uložený na službe poskytovanej fakultou, je nutné pre prístup do repozitára použiť prihlasovacie údaje od FIT ČVUT. Repozitár má v rámci služby riadený prístup, a tak je nutné o prístup požiadať mňa, prípadne iného člena projektu Dráček IV. Aby bolo možné s týmto repozitárom pracovať, je nutné stiahnuť a nainštalovať utilitu Git dostupnú z odkazu

https://git-scm.com/download/win.

Po získaní prístupu a nainštalovaní Gitu je možné prejsť na nasledujúci krok.

Java a Android Studio

V tomto kroku bude nutné stiahnuť a nainštalovať Java SDK (v čase písania práce je najvhodnejšia verzia *8u172*) z tohto odkazu:

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html.

IDE Android Studio je možné stiahnuť z odkazu

https://developer.android.com/studio/.

Po inštalácii (v ktorej začiarkneme všetky komponenty) a spustení sa užívateľovi zobrazí dialógové okno vyzývajúce k založeniu, prípadne importu projektu. V našom prípade zvolíme možnosť *Check out project from Version Control* a v nasledujúcom zozname vyberieme *Git*. Do políčka s *URL* zadáme

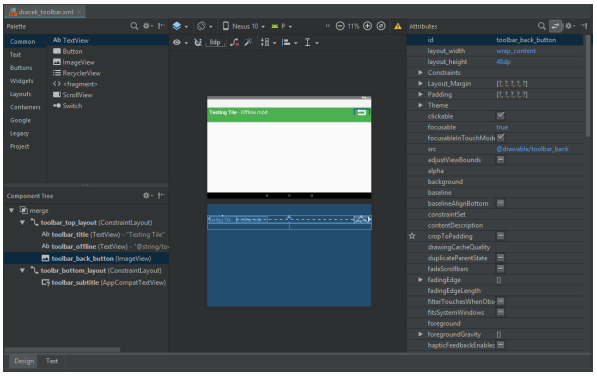
git@gitlab.fit.cvut.cz:kravejar/dracek.git

a vyberieme *Clone*. V nasledujúcich prípadoch zvolíme potvrdzovacie tlačidlo (väčšinou *Next*). Po potvrdení posledného kroku sa dostávame do editora kódu a v spodnej časti je možné vidieť priebeh *Gradle Sync*. Je možné, že pri synchronizácii budú hlásené neprítomné build nástroje potrebné ku kompilácii; v tom prípade je možné opraviť tieto chyby klikom na link pod samotnou chybou. Po úspešnej synchronizácii je možné editovať kód.

Úprava kódu

Kód je možné upravovať na mnohých miestach, pričom existuje mnoho spôsobov, akými sa požadovaná časť kódu upraviť dá. Z tohto dôvodu uvediem jeden ukážkový postup, ktorý napomôže k základnej intuícii.

Požadovaná časť určená k zmene bude v tomto prípade obrázok tlačidla návratu v Toolbare aplikácie. Toto tlačidlo je pre referenciu možné vidieť napríklad na obrázku 4.1 vpravo hore. Toolbar je súčasťou knižnice *common-library*, ktorú si v ľavej časti obrazovky rozklikneme. Objaví sa niekoľko priečinkov z ktorých rozklikneme *res* a po rozkliknutí vidíme pre nás dva relevantné priečinky: *layout* a *drawable*. V priečinku *layout* otvoríme súbor *dracek_toolbar.xml*. V spodnej časti aktuálneho panelu môžeme vidieť prepnutie medzi *Design* a *Text*. Tieto dva režimy sú priamo prepojené tak, že elementy v režime Text sa graficky zobrazia v režime Design. V režime Design je možné tieto elementy taktiež upravovať a zmeny sa prejavia v režime Text a zobrazia v režime Design. V našom prípade zvolíme režim Design a ak sa náhľad zobrazuje v režime na výšku, je možné ho prepnúť na šírku tlačidlom  . Vpravo od tohto tlačidla je možné vybrať veľkosť zobrazovaného displeja vyjadrenú v uhlopriečkach viacerých zariadení. Pre náš projekt je vhodné používať Nexus 10, prípadne iný model tabletu. Na obrázku 4.7 je možné vidieť stav, v ktorom



Obr. 4.7: Grafický náhľad rozloženia v IDE Android Studio.

by sme po nasledovaní návodu mali byť. Po kliknutí na požadovanú ikonu tlačidla je možné vpravo vidieť atribúty patriace tomuto elementu. Medzi nimi je aj atribút *src*, ktorý v tomto prípade vyjadruje zdroj, ktorý sa má zobraziť. Jeho hodnota je nastavená na *@drawable/toolbar_back*, čo naznačuje, že zdrojový súbor budeme hľadať v priečinku spomínanom vyššie – *drawable*. Po otvorení tohto priečinku tam skutočne môžeme nájsť súbor *toolbar_back.png*. K splneniu našej úlohy je možné buď tento súbor zameniť za iný, alebo zmeniť hodnotu atribútu *src*.

Build

K overeniu vykonaných zmien na zariadení je nutné aplikáciu skompilovať. Tento krok je za pomoci IDE pomerne jednoduchý a vykonáva sa tlačidlom  v hornej časti obrazovky. Ešte pred kliknutím tohto tlačidla je nutné vybrať modul, ktorý chceme kompilovať a následne spustiť. Pre náš prípad vyberieme aplikáciu Jadro - teda modul Core [upozornenie – nemýliť si **Gradle** modul (tento prípad) a **výukový** modul aplikácie Dráček]. Po stlačení tlačidla sa objaví dialóg, ktorý vyzýva k zvoleniu zariadenia, na ktorom bude aplikácia spustená. Zariadenie môže byť jednak fyzické – pripojené USB káblom, jednak emulátor. Pre použitie fyzického zariadenia je najprv nutné povoliť *adb* na zariadení. Vysvetlenie a návod, ako tento krok docieliť, je možné nájsť na [46]. V prípade použitia emulátora je nutné zariadenie najprv vytvoriť kliknutím na tlačidlo *Create New Virtual Device*, ktorý otvorí sprievodcu. Jeho prvý krok je výber uhlopriečky (znovu je odporúčané pre tento projekt použiť tablet), druhý výber OS (odporúčané je použiť najnovší dostupný) a tretí obsahuje možnosť upraviť niektoré parametre. Po vykonaní jedného z týchto dvoch postupov by sa malo zariadenie zobraziť v dialógovom okne. Build a spustenie na zariadení zahájime kliknutím na tlačidlo *OK*. Priebeh buildu je možné vidieť v spodnej časti obrazovky a v prípade, že je úspešný, je aplikácia spustená na želanom zariadení.

Commit a CI

Po vykonaní a otestovaní zmien je vhodné zmeny commitovať. Zmenené súbory majú názov zobrazený v modrej farbe; nové buď v červenej alebo zelenej farbe (záleží na tom, či sú pridané do gitu). Výber súborov určených pre commit zahájime kliknutím na tlačidlo  v hornej lište. Zobrazí sa dialóg, ktorý umožňuje začiarknúť súbory, ktoré budú súčasťou commitu. Po zvolení správnych súborov a krátkom popísaní zmien textom namierime kurzor na tlačidlo *Commit* a zvolíme možnosť *Commit & Push*. Nasledujúce dialógové okno potvrdíme tlačidlom *Push*, a týmto sa commit odošle na centrálny repozitár popísaný v prvom kroku. Na tomto repozitári je zriadené Continuous Integration v podobe automatického buildu pri novom commite. Po niekoľkých minútach je možné stiahnuť APK súbory všetkých modulov projektu zabalené v jednom ZIP súbore. Priebeh a výsledok je možné vidieť na adrese repozitára spomenutej vyššie v časti *Pipelines* so spomínanými súbormi dostupnými po kliknutí na tlačidlo .

Kapitola 5

Testovanie

Testovanie nových funkcionalít v kóde prebiehalo v čase písania implementácia a následne aj po ňom. Keďže vývoj bol zameraný hlavne na asistenta Dráčka, rozsah testov nebol tak obšírny ako v prípade vývoja novej aplikácie. Veľká väčšina testov prebiehala manuálne spočiatku na emulátore a neskôr na zariadení Google Nexus 10. V prípade emulátoru som používal hlavne verziu API 27 – Oreo a aplikácia bola odskúšaná aj na API P, čo je vývojársky náhľad na novú verziu Androidu a v čase písania tejto práce najnovšia verzia API. Keď som sa dostal k tabletu Google Nexus 10, analyzoval som možnosti aktualizácie na najnovšiu možnú verziu operačného systému. Nakoniec som na tomto tablete testoval aplikáciu na operačnom systéme s verziou API taktiež 27. Testovanie priamo na tablete malo v mojom prípade väčšiu výpovednú hodnotu, keďže emulátor mal – aj napriek pomerne výkonnému počítaču, na ktorom bežal – oveľa nižší grafický výkon, čo sa najviac prejavilo na animáciách. Rovnaký model sa používa aj na ZŠ Smečno, čo pridalo ďalšie plus k testovaniu na tablete.

Počas práce som na aplikácii vykonal niekoľko druhov testov, ktoré nižšie popíšem. Tie manuálne boli vykonané pomocou nasledujúcich prihlasovacích údajov:

Prihlasovacie meno: PStude

Heslo: dracek

5.1 Unit testy

Unit (jednotkový) testing je vývojársky proces, v ktorom sú najmenšie možné časti (jednotky) nezávisle na sebe kontrolované za účelom overenia správnosti fungovania [47]. Často sú automatizované a používané napríklad pre kontrolu správnosti výpočtu, alebo spracovania určitých dát.

Prípadov využitia unit testov v mojom príspevku do kódu som mnoho nenašiel, preto som do projektu pridal len dva unit testy. Jeden z nich kon-

troluje správnosť spracovania XML súboru pre vstup sprievodcu. Medzi referenčné data som okrem správnych vstupov zaradil aj nesprávne s negatívnym očakávaným výstupom.

Druhý sa zameriava na správnu transformáciu mena Activity pre automatické hľadanie vstupu pre sprievodcu v projektových súboroch.

5.2 Integračné testy

Integračné testy rozhodujú, či skupina nezávisle vyvíjaných jednotiek spolu komunikuje správne [48]. Túto skupinu testov som vykonával len manuálne, keďže vytváranie tohto typu testov je časovo náročnejšie hlavne na platformu Android. Do integračného testovania patrilo napríklad pozorovanie správania triedy DracekPositionManager a DracekPositionResolver v debug móde.

5.3 Užívateľské testy

Užívateľské testovanie je spôsob, ako odhaliť mieru jednoduchosti použitia aplikácie používateľom. Títo ľudia sú vyzvaní k splneniu rôznych úloh a pri tom sú pozorovaní moderátorom, ktorý si všíma ich narazenie na problémy a mieru zmätenia [49]. Určené pre tieto testy je na ČVUT zriadené Usability laboratórium, ktoré disponuje pokročilou výbavou a preto bude vhodné uskutočniť užívateľské testy v ňom.

5.3.1 Vyhodnotenie testovania Dráček III

Ako som uviedol v rozbore zadania na začiatku práce, po konzultácii s vedúcim práce bolo užívateľské testovanie aplikácie pred akýmkoľvek zásahom nahradené vyhodnotením testov predchádzajúcej iterácie. Po konzultácii s touto iteráciou a analýze výsledkov som však zistil, že výstupné dáta sú pre túto prácu irelevantné, keďže sa zameriavajú hlavne na pochopenie zadania a ovládacích prvkov cvičení. Z tohto dôvodu nie je možné tieto poznatky využiť a úsilie bude smerované na testovanie aplikácie po vykonaní zmien.

5.3.2 Testovanie Dráček IV

Testovanie bude po dohode s vedúcim práce dohodnuté na deň detí, kedy na škole bude viacero subjektov v cieľovej skupine. Po získaní súhlasu od rodičov a vyplnení vstupného dotazníku bude testovaná intuitívnosť základnej navigácie v aplikácii a prívetivosť nového sprievodcu - Dráčka. Súčasťou formulára predloženého rodičom bude niekoľko oznámení týkajúcich sa GDPR s ktorými musia následne rodičia súhlasiť. Medzi tieto oznámenia patria napríklad:

- Audio a video z testovania bude uchovávané.

- Anonymizované výsledky budú využité v tejto a iných vedeckých prácach
- Obrázok zachycujúci testovanie môže byť použitý pri obhajobe práce

Po podpísaní tohto dotazníku čaká pre testovací subjekt – dieťa – vstupný dotazník, ktorý bude obsahovať otázky napríklad na aktuálny ročník v škole/škôlke a skúsenosti s Androidom, mobilnými zariadeniami a hrami na nich. Vyplnením vstupného dotazníku môže začať samotné testovanie, ktoré bude prebiehať podľa vopred pripraveného scenára v dvoch variantoch:

Pre moderátora bude pripravené rozšírenie scenára, ktorý popisuje čoho sa držať v prípade, že nastane neočakávaná udalosť.

Pre testovanie bude určený klasický scenár, ktorý bude popisovať inštrukcie moderátora pre subjekt.

Po skončení testovania bude deťom predložený výstupný dotazník, ktorý zozbiera pocity a spätnú väzbu subjektov. Tieto podklady spoločne s výsledkami testovania po ich vykonaní budú dostupné jednak na priloženom pamäťovom médiu a jednak na adrese

https://goo.gl/g6AwZN.

Pod touto adresou je uložený zdieľaný priečinok služby Google Drive a pre prístup doň je nutný účet FIT ČVUT.

5.3.3 Výsledky testovania

Z dôvodu vykonania testov až po odovzdaní práce, budú po dohode s vedúcim práce výsledky spracované a predložené až pri obhajobe práce.

Záver

V práci som sa zaoberal analýzou, návrhom, implementáciou a testovaním vylepšení aplikácie Jadro projektu Dráček.

V analýze som zhodnotil aktuálny stav projektu a aplikácie Jadro a uviedol som prebraté a zistené nedostatky. Na základe týchto nedostatkov som vytýčil niekoľko nápadov, ktoré by tieto nedostatky odstraňovali a preskúmal som ich možnosti. Na základe týchto poznatkov som vytvoril zoznam funkčných a nefunkčných požiadaviek. V tejto časti som ešte uviedol niekoľko nástrojov a základných znalostí, ktoré by mal dizajnér vstupujúci do tohto projektu ovládať.

Požiadavkám z analýzy som sa venoval v návrhu, kde som uviedol a rozpísal niekoľko možností realizácie návrhu. Ku každej možnosti som uviedol plusy a mínusy, na základe ktorých som potom vyvodil záver a zvolil možnosť, ktorá bola neskôr implementovaná. Do tejto časti patril aj návrh rôznych možností spolupráce vývojára s dizajnérom.

Po implementácii funkcionalít podľa spôsobov zvolených v časti návrh som popísal, ako nové časti kódu fungujú a ako je možné ich používať. Okrem vývojárskej príručky som vytvoril aj príručku pomenovanú ako dizajnérska, ktorá popisuje kompletný proces úpravy kódu od inštalácie potrebných nástrojov až po commit. Táto príručka bola písaná tak, aby jej rozumel aj človek, ktorý s vývojom Android aplikácií nemá žiadne skúsenosti.

Implementované zmeny boli testované jednotkovými a integračnými testami predtým, ako boli pripravené na test užívateľský. Z časových dôvodov však tento test prebehne až po odovzdaní tejto práce, avšak podklady preň boli už pripravené a popísané v časti testovanie.

Ciele tejto práce boli zväčša splnené. Aj napriek neprítomnosti dizajnérky bolo možné prácu tejto skutočnosti prispôsobiť a splniť jej zadanie. Nápady na zlepšenie postupovali dvoma fázami - analýza a návrh. Ak sa nápad v jednej z týchto časti javil ako nevhodný, bol spolu s vysvetlením opustený. V opačnom prípade bol buď implementovaný, alebo popísaný v časti 4.3.6 aj s vysvetlením, prečo jeho implementácia nebola možná.

Pevne verím, že pre budúce generácie projektu bude táto práca prínosná a ušetrí čas a nervy pri vývoji. Možností vylepšenia aplikácie je mnoho, z ktorých niektoré sú popísané v časti 4.3.7.

Literatúra

[1] Pecanac, V.: What Is Continuous Integration and Why Do You Need It? *Code Maze*, február 2016, [cit. 2018-04-26]. Dostupné z: `https://code-maze.com/what-is-continuous-integration/`

[2] Pavelec, P.: *Výuková aplikace Dráček II – Vývoj jádra aplikace pro žákovskou část*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

[3] Bradáč, J.: *Výuková aplikace Dráček – Jádro klienta*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2013.

[4] Kubišta, P.: *Výuková aplikace Dráček – Správa zásuvných modulů*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2013.

[5] Kužela, O.: *Výuková aplikace Dráček - Serverová část*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2013.

[6] Bláha, M.: *Výuková aplikace Dráček Vývoj vybraných modulů*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2013.

[7] Tomšů, R.: *Výuková aplikace Dráček – Vývoj vybraných aplikačních zásuvných modulů*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2013.

[8] Filip, O.: *Výuková aplikace Dráček II – Serverová část a rozhraní pro učitele*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

[9] Mazel, M.: *Dragon II - plugins I.* Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

[10] Bureš, M.: *Výuková aplikace Dráček II - zásuvné moduly II.* Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

[11] Kovařovic, K.: *Gamifikace a personalizace výukové aplikace Dráček.* Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

[12] Podaný, P.: *Animovaný avatar pro výukovou aplikaci Dráček.* Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

[13] Slabý, O.: *Zásuvné moduly aplikace Dráček III - výuka základů algoritmizace.* Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

[14] Štěpán, J.: *Zásuvné moduly aplikace Dráček III - výuka fyziky.* Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

[15] Ryba, J.: *Zásuvné moduly aplikace Dráček III - výuka matematiky.* Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

[16] Kameník, M.: *Dráček IV - serverová část.* Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

[17] Number of available applications in the Google Play Store from December 2009 to December 2017. *Statista*, december 2017, [cit. 2018-04-30]. Dostupné z: `https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/`

[18] Lake, I.: Picking your compileSdkVersion, minSdkVersion, and targetSdkVersion. *Medium*, január 2016, [cit. 2018-03-12]. Dostupné z: `https://medium.com/google-developers/picking-your-compilesdkversion-minsdkversion-targetsdkversion-a098a0341ebd`

[19] Improving app security and performance on Google Play for years to come. *Android Developers Blog*, december 2011, [cit. 2018-03-12]. Dostupné z: `https://android-developers.googleblog.com/2017/12/improving-app-security-and-performance.html`

[20] Android: *Android NDK Native APIs.* [cit. 2018-03-12]. Dostupné z: `https://developer.android.com/ndk/guides/stable_apis.html`

[21] Android: *Manifest.permission*. [cit. 2018-05-01]. Dostupné z: https://developer.android.com/reference/android/Manifest.permission#INSTALL_PACKAGES

[22] The Mother Of All Android Malware Has Arrived: Stolen Apps Released To The Market That Root Your Phone, Steal Your Data, And Open Backdoor. *Android Police*, marec 2011, [cit. 2018-03-08]. Dostupné z: https://www.androidpolice.com/2011/03/01/the-mother-of-all-android-malware-has-arrived-stolen-apps-released-to-the-market-that-root-your-phone-steal-your-data-and-open-backdoor/

[23] FalseGuide malware victim count jumps to 2 million. *ZDNet*, apríl 2017, [cit. 2018-03-08]. Dostupné z: https://www.androidpolice.com/2011/03/01/the-mother-of-all-android-malware-has-arrived-stolen-apps-released-to-the-market-that-root-your-phone-steal-your-data-and-open-backdoor/

[24] Google: *View app statistics*. [cit. 2018-03-10]. Dostupné z: https://support.google.com/googleplay/android-developer/answer/139628

[25] Wikipedia: *Firebase*. [cit. 2018-03-10]. Dostupné z: https://en.wikipedia.org/wiki/Firebase

[26] Google: *Firebase Cloud Messaging*. [cit. 2018-03-10]. Dostupné z: https://firebase.google.com/docs/cloud-messaging/

[27] Google: *Firebase Authentication*. [cit. 2018-03-10]. Dostupné z: https://firebase.google.com/docs/auth/

[28] Wikipedia: *Google Play Games*. [cit. 2018-03-10]. Dostupné z: https://en.wikipedia.org/wiki/Google_Play_Games

[29] Google: *Google Play Games Services*. [cit. 2018-03-10]. Dostupné z: https://developers.google.com/games/services/

[30] Google: *Google+ Circles*. [cit. 2018-03-11]. Dostupné z: https://en.wikipedia.org/wiki/Google+#Circles

[31] Google: *Install Android apps on your Chromebook*. [cit. 2018-04-19]. Dostupné z: https://support.google.com/chromebook/answer/7021273?hl=en

[32] Facial recognition technology explained. *Android Authority*, september 2017, [cit. 2018-03-11]. Dostupné z: https://www.androidauthority.com/facial-recognition-technology-explained-800421/

[33] Samsung Galaxy S8 iris scanner fooled by German hackers. *The Guardian*, máj 2017, [cit. 2018-03-11]. Dostupné z: https://www.theguardian.com/technology/2017/may/23/samsung-galaxy-s8-iris-scanner-german-hackers-biometric-security

[34] Face Unlock — Android 4.0 Ice Cream Sandwich 4.0's Most Personal Feature. *Android Authority*, október 2011, [cit. 2018-03-11]. Dostupné z: https://www.androidauthority.com/face-unlock-android-4-0-ice-cream-sandwich-most-personal-feature-27693/

[35] GitHub: *Android Face Recognition with Deep Learning - Test Framework*. [cit. 2018-03-12]. Dostupné z: https://github.com/Qualeams/Android-Face-Recognition-with-Deep-Learning-Test-Framework

[36] GitHub: *Face Recognition Library*. [cit. 2018-03-12]. Dostupné z: https://github.com/Lauszus/FaceRecognitionLib

[37] Bhatia, P.: A summary of 10 key GDPR requirements. *EU GDPR Academy*, [cit. 2018-04-30]. Dostupné z: https://advisera.com/eugdpracademy/knowledgebase/a-summary-of-10-key-gdpr-requirements/

[38] git: *git*. [cit. 2018-04-19]. Dostupné z: https://git-scm.com/

[39] Android: *Android Studio*. [cit. 2018-04-19]. Dostupné z: https://developer.android.com/studio/index.html

[40] Lindhout, R.: Programmatically check Play Store for app updates. [online], december 2015, [cit. 2018-04-04]. Dostupné z: https://stackoverflow.com/revisions/25201516/3

[41] Linuxtopia: *Resources and Assets*. [cit. 2018-04-02]. Dostupné z: http://www.linuxtopia.org/online_books/android/devguide/guide/topics/resources/index.html

[42] Bugayenko, Y.: Stop Comparing JSON and XML. november 2015, [cit. 2018-04-02]. Dostupné z: http://www.yegor256.com/2015/11/16/json-vs-xml.html

[43] w3schools.com: *JSON vs XML*. [cit. 2018-04-02]. Dostupné z: https://www.w3schools.com/js/js_json_xml.asp

[44] Android: *Button*. [cit. 2018-04-02]. Dostupné z: https://developer.android.com/reference/android/widget/Button.html

[45] Android: *Launch checklist*. [cit. 2018-04-19]. Dostupné z: https://developer.android.com/distribute/best-practices/launch/launch-checklist.html

[46] Android: *Android Debug Bridge (adb)*. [cit. 2018-05-03]. Dostupné z: https://developer.android.com/studio/command-line/adb

[47] TechTarget: *unit testing*. [cit. 2018-05-02]. Dostupné z: https://searchsoftwarequality.techtarget.com/definition/unit-testing

[48] Fowler, M.: IntegrationTest. *MartinFowler.com*, január 2018, [cit. 2018-05-05]. Dostupné z: https://martinfowler.com/bliki/IntegrationTest.html

[49] Experience UX: *What is usability testing?* [cit. 2018-05-03]. Dostupné z: https://www.experienceux.co.uk/faqs/what-is-usability-testing/

Dodatok **A**

Zoznam použitých skratiek

GUI Graphical user interface

XML Extensible markup language

JSON Javascript object notation

IDE Integrated development environment

APK Android application package

SDK Software development kit

Dodatok **B**

Obsah priloženého média

- `readme.txt` ..... stručný popis obsahu média
- `apk` ..... adresár so spustiteľnou formou implementácie
- `doc` ..... dokumentácia implementácie
- `src`
  - `impl` ..... zdrojové kódy implementácie
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
- `text` ..... text práce vrátane príloh
  - `BP_Sivak_Dominik_2018.pdf` ..... text práce vo formáte PDF
  - `BP_Sivak_Dominik_2018.ps` ..... text práce vo formáte PS
  - `annexes` ..... prílohy práce