VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ELEKTRONICKÝ NÁSTROJ PRO SPRÁVU INOVATIVNÍCH STUDIJNÍCH MATERIÁLŮ

ELECTRONIC TOOL FOR MANAGING INNOVATIVE STUDY MATERIALS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. ĽUBOŠ HLIPALA**
AUTHOR

VEDOUCÍ PRÁCE — **prof. Ing. ADAM HEROUT, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání diplomové práce

Student: **Hlipala Ľuboš, Bc.**
Program: Informační technologie Obor: Počítačová grafika a multimédia
Název: **Elektronický nástroj pro správu inovativních studijních materiálů**
Electronic Tool for Managing Innovative Study Materials
Kategorie: Uživatelská rozhraní

Zadání:

1. Vyhledejte a analyzujte existující nástroje pro učení a procvičování látky na mobilním telefonu.
2. Navrhněte inovativní systém pro strukturování znalostí a jejich prezentaci na mobilním telefonu.
3. Implementujte prototyp aplikace pro prezentování, studování a procvičování strukturovaných studijních materiálů.
4. Testujte jednotlivé vlastnosti řešeného způsobu vyučování na vhodných subjektech. Iterativně vylepšujte prototyp aplikace a shromažďujte poznatky o vyvíjeném způsobu učení a reprezentování znalostí.
5. Dokumentujte formát pro výměnu strukturovaných znalostí pro výuku a procvičování, demonstrujte formát na příkladech.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Steve Krug: Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability, ISBN: 978-0321657299
- Susan M. Weinschenk: 100 věcí, které by měl každý designér vědět o lidech, Computer Press, Brno 2012
- Joel Marsh: UX for Beginners: A Crash Course in 100 Short Lessons, O'Reilly 2016
- Frank Zammetti: Practical Flutter: Improve your Mobile Development with Google's Latest Open-Source SDK, Apress 2019

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3, značné rozpracování bodu 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 31. července 2020
Datum schválení: 3. března 2020

Abstrakt

Cieľom tejto diplomovej práce je vytvoriť mobilnú aplikáciu, ktorá umožňuje prehľadnou štruktúrou a zároveň vhodným a efektívnym spôsobom zdieľať, prezentovať a precvičovať znalosti rôzneho druhu. Dielo analyzuje niekoľko existujúcich aplikácií a systémov pre výuku a zdieľanie informácií a opisuje rozdielne prístupy k vývoju mobilných aplikácií so zameraním na multiplatformnosť. Práca sa tiež podrobne venuje technológii Flutter a charakterizuje vhodné princípy tvorby užívateľského rozhrania, na základe ktorých bola výsledná mobilná aplikácia vytvorená. Hlavná časť práce opisuje postupný vývoj aplikácie a dokumentuje vytvorený formát pre prezentovanie a zdieľanie výukových materiálov, s možnosťou praktického precvičovania nadobudnutých vedomostí. Výsledkom tejto diplomovej práce je mobilná aplikácia, ktorá je zverejnená na digitálnej distribučnej platforme Google Play.

Abstract

The aim of this diploma thesis is to create a mobile application that provides the users with structured, well-arranged and at the same time effective platform to share, present and practice the knowledge of various kinds. The work analyzes several existing applications and systems for education and sharing information and describes different approaches of mobile app development with the focus on multiplatformity. The main part of the work describes the gradual development of the application and documents the created format for the presentation and sharing of study materials, with the possibility of practical practice of the acquired knowledge. The result of this diploma thesis is a mobile application, which is available on the Google Play Store.

Kľúčové slová

grafické užívateľské rozhranie, GUI, užívateľské rozhranie, UI, užívateľská skúsenosť, UX, Flutter, Flutter SDK, mobilná aplikácia, multiplatformová mobilná aplikácia, výučba, vzdelanie, študijné materiály

Keywords

graphical user interface, GUI, user interface, UI, user experience, UX, Flutter, Flutter SDK, mobile application, multiplatform mobile application, education, study materials

Citácia

HLIPALA, Luboš. *Elektronický nástroj pro správu inovativních studijních materiálů*. Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Elektronický nástroj pro správu inovativních studijních materiálů

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána prof. Ing. Adama Herouta Ph.D. Uviedol som všetky literárne pramene, publikácie a dalšie zdroje, z ktorých som čerpal.

.....
Ľuboš Hlipala
31. júla 2020

Poďakovanie

V tejto sekcii by som chcel poďakovať školiteľovi tejto práce pánovi prof. Ing. Adamovi Heroutovi Ph.D za jeho odbornú pomoc, čas a pripomienky.

Obsah

Kapitola 1

Úvod

Výmena informácií predstavuje základný prvok v oblasti prežitia a ďalšieho napredovania ľudskej civilizácie už od dávnych čias. V prvopočiatkoch historického vývoja sa všetky nadobudnuté poznatky predávali medzi jedincami a skupinami výlučne ústnou formou. S postupom času a zvyšujúcim sa množstvom poznatkov nebolo už však pre vtedajších predstaviteľov ľudského druhu možné si tak značné kvantum vedomostí pamätať. Aj to doviedlo ľudstvo v danom období v tomto smere k veľkému míľniku v podobe vytvorenia prvých podôb prastarých písiem. Vďaka písmu bolo umožnené predmetné informácie zapisovať na rôzne formy kamenných tabúľ či dosiek a neskôr aj na iné nosiče, ako je napríklad dodnes používaný papier. Z rozsiahlych poznatkov následne začali vznikať knihy, ktoré sú doposiaľ najznámejším a zároveň aj najvyužívanejším médiom na predávanie informácií. Knihy dnes prakticky tvoria základ pre edukačné inštitúcie a slúžia ako referencie pre študijné materiály. Príchodom informačných technológií a internetu sa knihy, študijné materiály a v podstate všetky druhy informácií začali digitalizovať, čo so sebou prináša jednoduchší a rýchlejší prístup k týmto dátam. V dnešnej dobe je za pomoci internetu častokrát jednoduché nájsť k jednému poznatku veľké množstvo zdrojov, pričom každý z nich podáva detailný popis spôsobom osobitným sebe samému, niektoré sú prehľadnejšie, iné menej, no v konečnom dôsledku obe bližšie predstavujú detailnejší popis. Dnešným problémom teda nie je dostupnosť, ale skôr vhodný spôsob podania konkrétnej informácie. Preto som sa v zmysle uvedeného rozhodol pre vypracovanie predmetného zadania.

V úvode textu bude popísaná analýza existujúcich produktov, ktoré by mali priniesť vhodné poznatky o spracovanej problematike, následne text rozoberie prístupy a nástroje na vývoj natívnych a multiplatformových aplikácií, pomocou ktorých bude konečný produkt vytvorený. Hlavná časť práce bude detailne opisovať iteratívny proces a celkový progres v jednotlivých fázach vývoja aplikácie, ktorá je cieľom tejto práce. V samotnom závere budú zhodnotené výsledky ktoré sa podarilo dosiahnuť.

Kapitola 2

Existujúce riešenia

Pre rozšírenie nápadov a inšpiráciu na vytvorenie konečnej aplikácie, jej organizácie, rozloženia užívateľského rozhrania, či celkovej štruktúry aplikácie je vhodné analyzovať niekoľko existujúcich produktov, ktoré sú doposiaľ k dispozícii. Aplikácia, ktorá bude výstupom tejto práce, by mala byť vhodná pre výukový materiál z ľubovoľného odvetvia. Preto diplomová práca v tejto kapitole rozoberá aplikácie zamerané na výučbu predmetov z rôznych oblastí ako napríklad cudzie jazyky, biológia, chémia, história, technika ale aj aplikácie na výuku či zdieľanie širokospektrálnych informácií.

2.1 Duolingo

Duolingo[1] je s aktuálnym počtom 300 miliónov užívateľov [37] veľmi populárnou mobilnou aplikáciou na výuku cudzích jazykov. Za týmto úspechom stojí určite fakt, že produkt ponúka užívateľovi široký výber jazykov, v ktorých sa môže zdokonaľovať úplne zdarma. Pre anglicky hovoriacich používateľov je k dispozícii viac ako 30 cudzích jazykov. Aplikácia podporuje aj výuku zo slovenského či českého jazyka. Veľmi zaujímavou vlastnosťou je, že výuka jazyka pozostáva výhradne z praktických úloh, pričom sa aplikácia nezameriava primárne na teoretické vysvetlenie problematiky. Bližšie informácie k riešeniu danej úlohy je však možné získať pri diskusii k danej otázke, kde si môžu užívatelia navzájom odpovedať.

Štruktúra

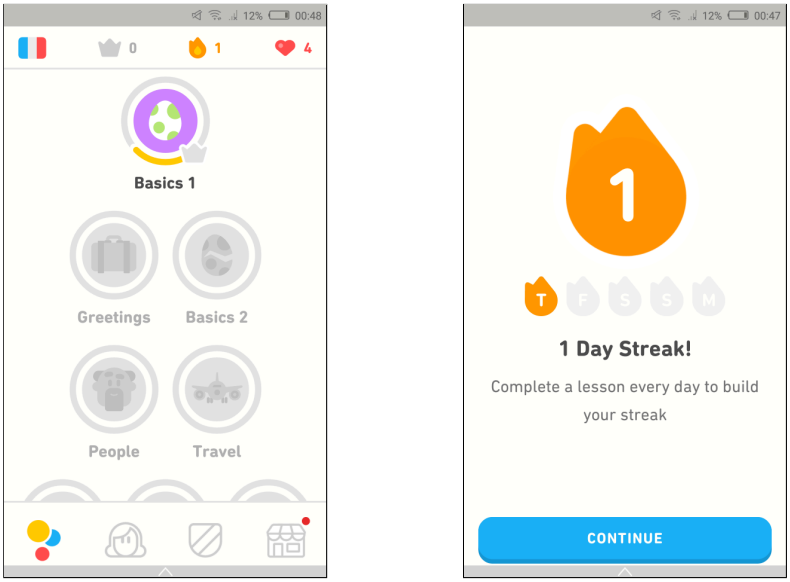
Štruktúra aplikácie je pre každý jazyk organizovaná do modulov, ktoré sa venujú určitej téme. Môže ísť o gramatiku, frázy, ale aj voľné témy ako voľnočasové aktivity, rodina, šport a iné. Moduly sú organizované tak, že sú na seba chronologicky naviazané a pre absolvovanie celého modulu je nutné splniť určitý počet lekcií. Po absolvovaní jedného alebo niekoľkých modulov, je sprístupnená nová sada modulov, ktoré vyžadujú znalosť a nadväzujú na už absolvované moduly. Štruktúra je teda prehľadná, jednoduchá a hierarchia kurzov nie je zanorovaná (viď obr. 2.1).

Gamifikácia

Neoddeliteľnou súčasťou Duolinga je gamifikácia procesu výučby. Aplikácia obsahuje veľa gamifikačných prvkov ako napríklad zbieranie odznakov za absolvovanie rôznych dlhodobých výziev, porovnávanie užívateľa s inými užívateľmi pomocou počtu získaných bodov

[1] Dostupné z: https://play.google.com/store/apps/details?id=com.duolingo

skúseností, či stratu života za nesprávnu odpoveď. Iným zaujímavým prvkom sú takzvané lingoty, ktoré užívateľ môže získať za absolvovanie krátkodobých výziev, ako je absolvovanie lekcie. Lingoty slúžia ako virtuálna mena, ktorú môže užívateľ vymeniť za doplnenie stratených životov, prémiové moduly a množstvo odlišných výhod. Na to, aby sa užívateľ každý deň k aplikácii vrátil, sa využíva systém, ktorý počíta počet dni štúdia bez prerušenia (anglický Streak – obr. 2.1).



Obr. 2.1: Aplikácia Duolingo. Jednoduchá hierarchia kurzu (vľavo). Gamifikačný prvok – aplikácia počíta počet dni štúdia bez prerušenia (vpravo)

2.2 Wikipedia

Wikipedia je známa online encyklopédií, ktorá ponúka prezentovanie obsahu pomocou webovej stránky[2] ale aj pomocou mobilnej aplikácie[3]. Tým, že nie je primárne určená iba na rýchly prehľad alebo precvičovanie problematiky, tak ide o rozsiahly zdroj znalostí, ktorý využíva veľké množstvo ľudí, rôznych vekových kategórií s ľubovoľným zameraním. Nanajvýš pozoruhodnou ideou je otvorenosť úprav obsahu všetkým užívateľom. Vzhľadom na to, že Wikipediu využíva veľké množstvo ľudí s rôznymi záujmami, musí poskytovať nástroje alebo prvky, pomocou ktorých je možné znalosti z rôznych odvetví prezentovať. Wikipedia z toho dôvodu využíva pre štrukturovanie textu značkovací jazyka WikiText.

WikiText [48] obsahuje niekoľko elementov, pomocou ktorých užívatelia môžu stránku štrukturalizovať a organizovať. Medzi najdôležitejšie elementy určite patria rôzne veľkosti a typy nadpisov, variácie štýlov textu (bold, italic...), citáty, číslované a nečíslované zoznamy, špeciálne znaky a symboly, grécka abeceda, tabuľky a taktiež matematické vzorce, ktoré využívajú zápis zvlášť podobný systému LaTeX. Čo sa týka vizualizačných prvkov, tak sú plne podporované statické a animované obrázky rôznych formátov, zvukové zdroje ale aj video formáty.

[2]Dostupné z: https://www.wikipedia.org
[3]Dostupné z: https://play.google.com/store/apps/details?id=org.wikipedia

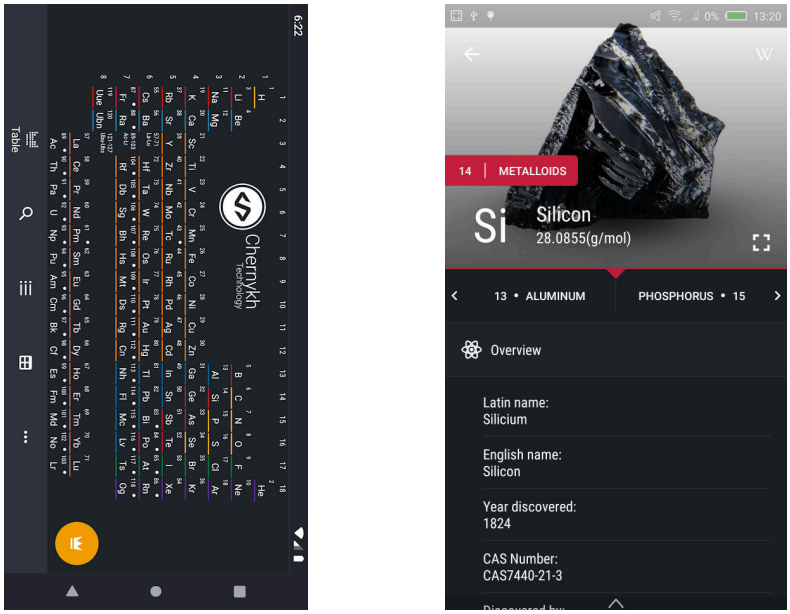
Keďže sa pojednáva o systém, ktorý slúži výhradne na poskytovanie informácií a jeho cieľom nie je sledovať a nútiť užívateľa v progres, tak Wikipedia neobsahuje žiadne gamifikačné prvky ani metódy pre praktické precvičovanie znalostí.

2.3 Periodic Table 2020

Aplikácia Periodic Table 2020[4] je určená pre študentov a nadšencov chémie. Ide o interaktívnu periodickú tabuľku, ktorá obsahuje prehľad informácií daných prvkoch. Veľmi dôležitá je prehľadnosť, organizovanosť a jednoduchosť podania informácie. Jednotlivé informácie nie sú podávané vo forme textu ale v podobe prehľadu kľúčových slov a ich hodnôt.

Ďalším dôležitým prvkom sú farby. Pre prehľadnosť využíva celá aplikácia farebné rozlíšenie prvkov na základe ich vlastností, alebo skupiny do ktorej patria (napríklad kovy, polokovy, nekovy, alkaloidy atď.). Taktiež umožňuje filtrovanie jednotlivých prvkov a skupín tabuľky. Vzhľad aplikácie, využívajúci tieto prvky je možné vidieť na obrázku 2.2.

Na rozdiel od aplikácie Duolingo neobsahuje žiadne prvky gamifikácie, predstavuje skôr veľmi vydarený prehľad prvkov spracovaný do dobre organizovanej formy. Aj bez gamifikačných prvkov sa najmä vďaka spomenutým vlastnostiam produkt môže pýšiť viac ako piatimi miliónmi stiahnutí[4].



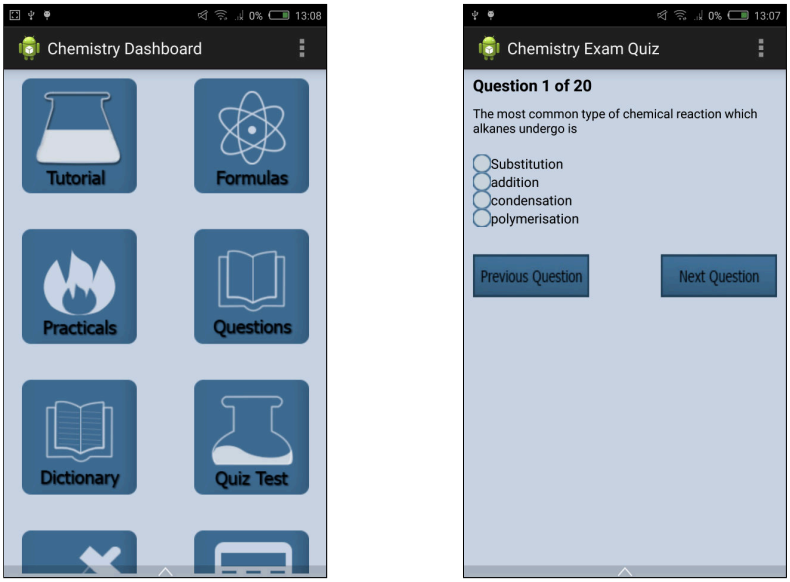
Obr. 2.2: Aplikácia Periodic Table 2020. Farebná periodická tabuľka chemických prvkov (vľavo). Detail chemického prvku (vpravo).

2.4 Complete Chemistry

Complete chemistry[5] odlišne od Periodic Table 2020 prezentuje svojim užívateľom komplexný prehľad základných konceptov chémie. Okrem periodickej sústavy chemických prvkov obsahuje prehľad rovníc pre základné, či pokročilé chemické výpočty, následne poskytuje

[4]Dostupné z: https://play.google.com/store/apps/details?id=mendeleev.redlime
[5]Dostupné z: https://play.google.com/store/apps/details?id=com.toscanyacademy.completechemistry

ku každej tématike určitú sadu otázok, ktoré znalosti z danej témy zoceľujú a precvičujú. Zaujímavým prvkom je aj to, že vývojár umožnil vytváranie vlastných poznámok, kde si užívateľ môže zapísať dôležité informácie alebo pojmy. V aplikácii sa taktiež nachádza slovník pre rýchly náhľad a vysvetlenie dôležitých chemických pojmov. Jedným z problémov aplikácie je, že využíva reklamy, ktoré sú umiestnené na miestach, kde prekrývajú niektoré dôležité informácie ako pri spodnej časti tabuľky prvkov. Medzi jemné prvky gamifikácie je možné zaradiť precvičovanie látky vo forme kvízu. Vzhľad aplikácie je vyobrazený na obrázku 2.3.

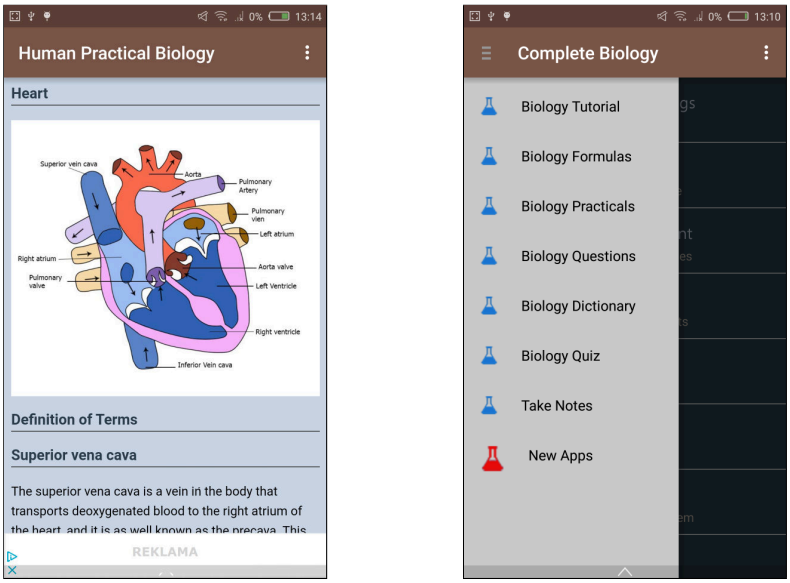


Obr. 2.3: Aplikácia Complete Chemistry. Hlavná stránka aplikácie (vľavo). Precvičovanie znalostí vo forme kvízov (vpravo)

2.5 Complete Biology

Aplikácia Complete Biology[6] patrí medzi aplikácie pre výuku biológie a je vytvorená rovnakým vývojárom ako produkt Complete Chemistry. Veľmi pravdepodobne aj to je dôvod, prečo sú aplikácie podobné a zdieľajú niektoré rovnaké vlastnosti. Aplikácia poskytuje zoznam biologických vzorcov a rovníc, slovník dôležitých biologických pojmov a otázok vo forme kvízu. Elementárne dôležitým prvkom aplikácie je, že bola prispôsobená oblasti, ktorú vyučuje, teda biológii. V tomto vedeckom odbore je značné množstvo javov, ktoré sú príliš ťažko vysvetliteľné a predstaviteľné iba za pomoci textu alebo rovníc. Aplikácia teda veľmi múdro využíva spôsob vysvetľovania problematiky pomocou grafických obrázkov s popisom (obr. 2.4). Tento štýl jednoznačne robí výuku danej problematiky prehľadnejšou a jednoduchšou.

[6] Dostupné z: https://play.google.com/store/apps/details?id=com.toscanyacademy.completebiology



Obr. 2.4: Aplikácia Complete Biology. Aplikácia využíva pri výučbe obrázky s popisom (vľavo). Slovník biologických pojmov (vpravo)

2.6 Khan Academy

Kahn Academy [33] je nezisková organizácia umožňujúca bezplatné vzdelávanie všetkým ľuďom s prístupom na internet. Kurzy sú dostupné cez webové rozhranie[7] a zároveň cez mobilnú aplikáciu[8]. Celý systém poskytuje širokú škálu kurzov v rôznych kategóriách ako matematika, prírodné vedy, humanitné vedy, technika, informačné technológie, ekonómia a financie, ale aj výtvarné umenie. Originálny je aj spôsob prezentovania vedomostí v jednotlivých kurzoch, ktorý je zhodný pre všetky kategórie.

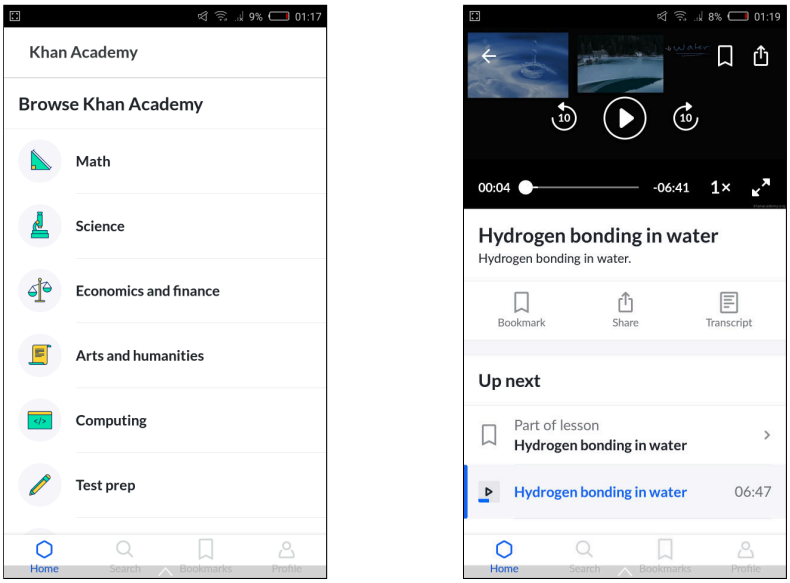
Štruktúra

Každý kurz pozostáva z niekoľkých sekcií, pričom každá sekcia obsahuje určitý počet lekcií. Samotná lekcia je väčšinou zložená iba z videa a diskusie (obrázok 2.5) a prípadne doplnená textom a prílohami. Video je vo väčšine prípadov pomerne krátke. Subjektívnym pozorovaním bolo možné zistiť, že priemerná dĺžka videí sa pohybuje v rozmedzí od 5 do 20 minút, výnimočne však aj dlhšie. Tento spôsob rozdelenia celistvej problematiky na čiastočné podproblémy predstavuje vhodnejšiu formu a dodáva výrazný prehľad k výučbovému procesu. Empiricky možno tvrdiť, že krátke videá dokážu udržať študenta viac v strehu a nedemotivujú ho hneď na začiatku tým, že bude musieť stráviť dlhý čas neefektívnym štúdiom. Po absolvovaní každej sekcie sa spúšťa možnosť vyplnenia kvízových otázok, ktoré kontrolujú, či študent pochopil danú matériu správne alebo by si mal niektoré lekcie zopakovať.

Gamifikácia

Systém začlenil aj jednoduché prvky gamifikácie ako získavanie rôznych odznakov za absolvovanie kurzov a pridávanie prospešných príspevkov do komunitného fóra a diskusie. Študent taktiež získava tzv. energetické body (Energy Points), ktoré sú akýmsi meradlom toho,

[7] Dostupné z: https://www.khanacademy.org
[8] Dostupné z: https://play.google.com/store/apps/details?id=org.khanacademy.android



Obr. 2.5: Aplikácia Khan Academy. Aplikácia a množstvo jej kurzov z najrôznejších odvetví (vľavo). Samotná lekcia je väčšinou zložená iba z videa a diskusie (vpravo)

koľko úsilia užívateľ investoval do štúdia a praktického precvičovania problematiky. Rovnako ako pri aplikácii Duolingo systém zobrazuje počet dní štúdia bez prerušenia (streak), čo podporuje systematickú prípravu študenta.

Kapitola 3

Prístupy k vývoju mobilných aplikácií

V tejto kapitole sa bude diplomová práca bližšie venovať jednotlivým prístupom k vývoju mobilných aplikácií a frameworku Flutter. Taktiež budú rozobrané všeobecné princípy návrhu užívateľského rozhrania a agilný prístup k vývoju aplikácií.

3.1 Natívne aplikácie

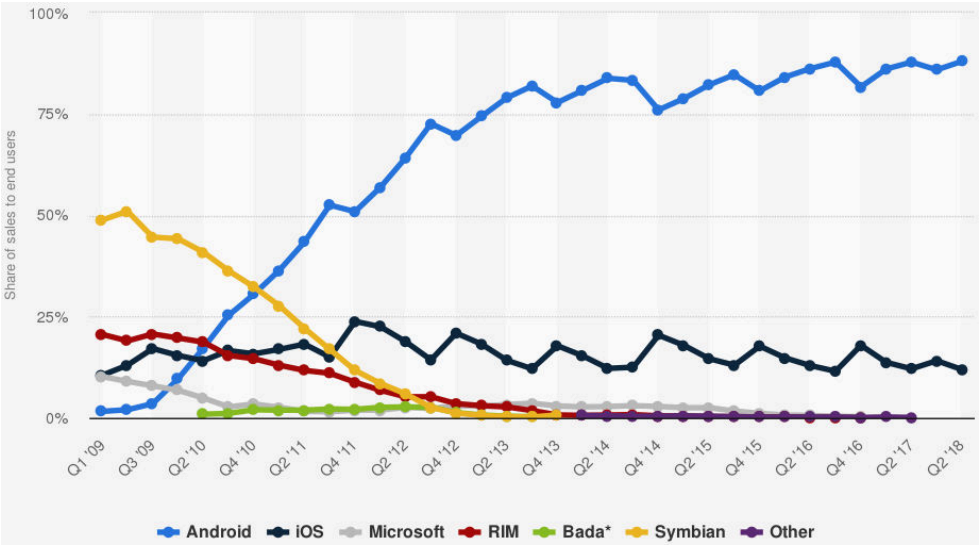
Každý tvorca operačného systému vo vlastnom záujme vytvára nástroje a prostriedky k tomu, aby vývojári aplikácií tvorili aplikácie práve pre jeho platformu. Počas doby smartfónov sa v obehu vyskytlo niekoľko rôznych operačných systémov, pričom niektoré boli úspešnejšie, iné zase menej. V minulosti bola najvyužívanejšia platforma [28] Symbian od Nokie, avšak časom ju prevalcoval Android od Google Inc. a iOS od firmy Apple. Za zmienku stojí aj Windows Phone (Microsoft), ktorý sa snažil začleniť do rozbehnutého trhového mechanizmu, no kvôli nevhodnému načasovaniu a následnému nedostatku užívateľov nebol atraktívny pre vývojárov aplikácií, čo viedlo k jeho postupnému zániku. Ak chceli vývojári vytvárať aplikáciu na zariadenia, tak museli využívať dostupné nástroje. Využitím týchto nástrojov vznikali tzv. natívne aplikácie [13], teda aplikácie fungujúce v súlade s funkcionalitou a užívateľskými návykmi pre danú platformu. Postupný vývoj popularity rôznych platforiem je vyobrazený na obrázku 3.1.

Android

Android [34] je open source projekt zastrešovaný firmou Google. Pre vývojárov je pripravené prostredie Android Studio, ktoré bolo vytvorené a upravené pre potreby vývoja aplikácií na Android v spolupráci s firmou JetBrains [46]. Pre vývoj aplikácií vznikla sada vývojových nástrojov Android SDK [42], ktorá podporuje jazyk Java, Kotlin a ponúka kvalitnú dokumentáciu. Pomocou týchto nástrojov je možné tvoriť natívne aplikácie pre platformu Android [29] a pristupovať k aplikačnému rozhraniu zariadenia.

iOS

S príchodom prvého iPhonu majiteľ spoločnosti Apple Inc. Steve Jobs nemal v úmysle umožniť vývoj natívnych aplikácií tretím stranám, a to kvôli obave z toho, že by takéto aplikácie mohli narušiť integritu alebo infikovať zariadenie vírusmi [32]. Tento názor však Steve Jobs



Obr. 3.1: Historický vývoj mobilných platforiem a ich podiel na svetovom trhu. [28]

následne zmenil [32] a tak je dnes možné pre vývoj natívnych aplikácií pre platformu iOS využiť iOS SDK [5]. Vývojová sada podporuje implementáciu aplikácií v jazykoch Objective-C [4] a Swift [3]. Spoločnosť Apple tiež ponúka vývojové prostredie XCode [6]. Nevýhodou je, že pre vytvorenie aplikácie pomocou nástroja XCode je nutné vlastniť zariadenie s operačným systémom od firmy Apple, pretože vývojové prostredie s inými zariadeniami nie je kompatibilné.

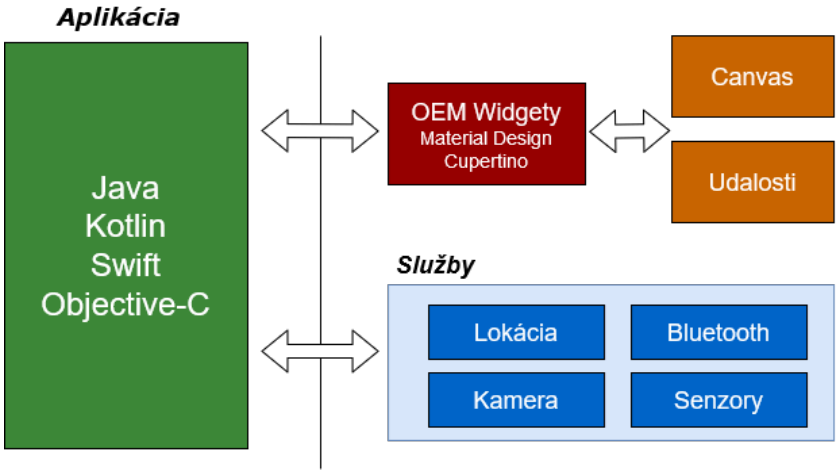
Vlastnosti natívnych aplikácií

Natívne mobilné aplikácie [7, 30] sú vytvárané a optimalizované pre špecifickú platformu. Výsledkom je, že aplikácia pracuje rýchlo a efektívne, a to z dôvodu, že pomocou natívneho programovacieho jazyka pre danú platformu je možné priamo pristupovať k službám zariadenia a natívnym prvkom užívateľského rozhrania (viď obr. 3.2). Keďže zámerom väčších aplikácií je pokryť trh čo najlepšie a vzhľadom na to, že počet užívateľov využívajúcich platformu iOS a Android nie je zanedbateľný, sú väčšinou vývojári takýchto aplikácií nútení vytvoriť aplikáciu pre každý z týchto operačných systémov zvlášť. To však má negatívny vplyv [30] na celý proces vývoja aplikácie najmä z hľadísk financovania prevádzky služby a údržby samotného softvéru.

3.2 Hybridné aplikácie

Kvôli hlavným nevýhodám ako je cena a náročnosť na čas začali postupne vznikať riešenia, ktoré by dokázali tieto nedostatky postupne odstrániť. Jednou z možností v medziach množiny potenciálnych riešení sú napríklad hybridné aplikácie.

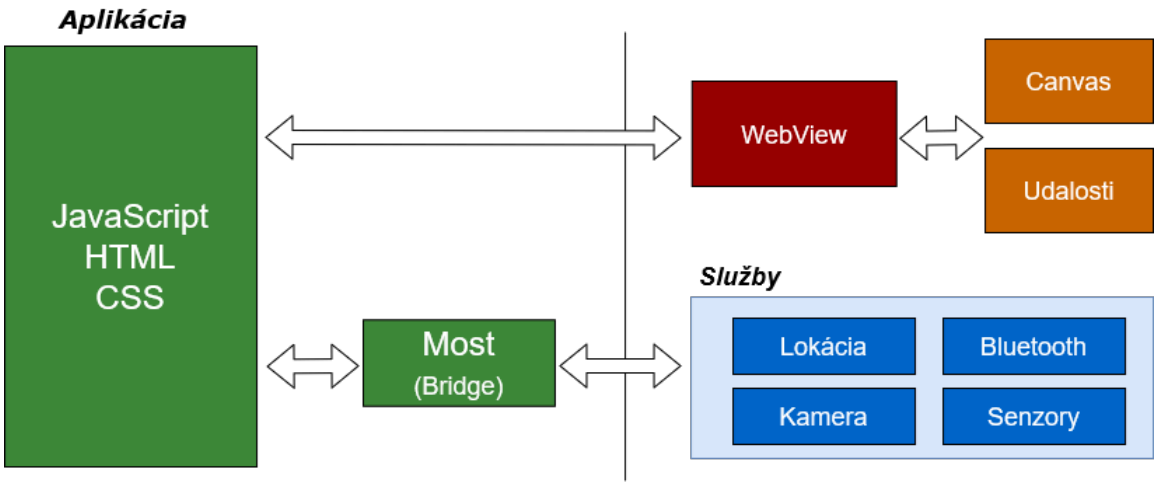
Hybridné aplikácie [30] dostali svoj názov kvôli tomu, že ide o webové aplikácie prispôsobené pre mobilné zariadenia, ktoré však majú prístup k niektorým natívnym funkciám operačného systému a hardvéru zariadenia. V podstate ide o webové aplikácie distribuovateľné cez AppStore a Google Play. Vytvorená webová aplikácia je zaobalená do WebView [30], čo je natívny widget, ktorý dokáže vykresľovať webové stránky resp. aplikácie.



Obr. 3.2: Princíp fungovania natívnych aplikácií. Zdroj: [43] spracovanie: vlastné

Vzhľadom na to, že aplikácia zdieľa rovnaký zdrojový kód pre všetky platformy, tak aj náklady na vývoj a následné udržiavanie aplikácie sú výrazne znížené [30]. Hybridné aplikácie [11, 30] využívajú webové technológie, ktoré sú značne rozšírené, preto je oveľa jednoduchšie nájsť programátorov a dizajnérov, ktorí sú v tomto smere skúsenejší a cenovo dostupní. Na druhej strane, zásadnou nevýhodou hybridných aplikácií je ich závislosť na natívnom vykresľovacom engine webových stránok, čo vedie k nekonzistentnosti výkonu aplikácie na zariadeniach.

Nevyhnutnou súčasťou frameworkov na vývoj takýchto aplikácií je technológia (tzv. bridge) [30], ktorá zabezpečuje komunikáciu so službami zariadenia danej platformy ako sú prístup ku kamere, lokácii zariadenia a iné. Princíp fungovania hybridných aplikácií ilustruje obrázok 3.3.



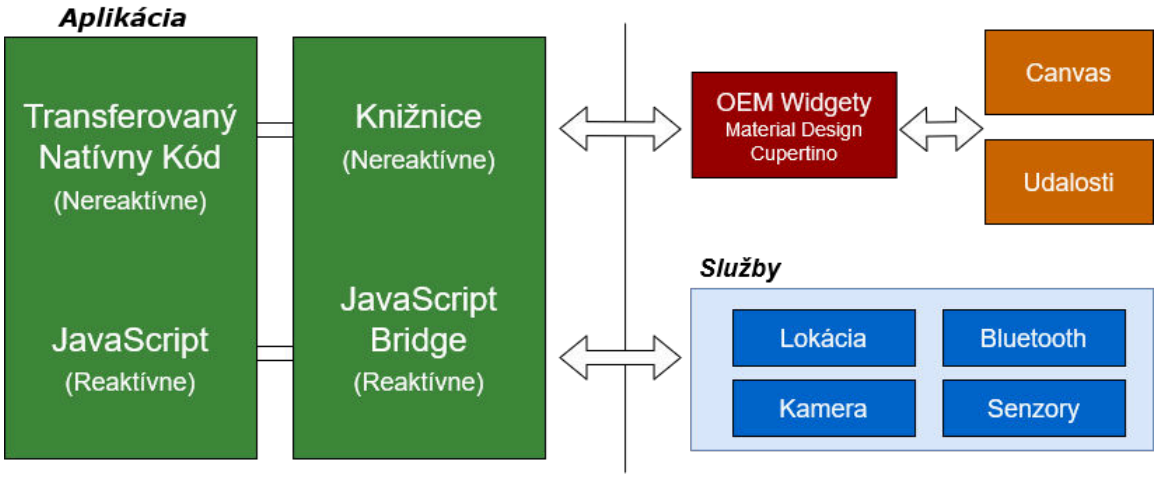
Obr. 3.3: Princíp fungovania hybridných aplikácií. Zdroj: [43] spracovanie: vlastné

Známe frameworky

Pre vytváranie hybridných aplikácií existuje hneď niekoľko použiteľných nástrojov. Jedným zo známych prostriedkov pre tvorbu hybridných aplikácií je Apache Cordova. Apache Cordova [30, 45] predstavuje framework umožňujúci tvorbu hybridných aplikácií pomocou známych webových technológií HTML, CSS, JavaScript. Ako už bolo spomenuté, tak tiež sprístupňuje programátorom možnosť využívať rôzne potrebné funkcie a služby zariadenia, ako je prístup ku kamere, geolokačným dátam, internetu, pamäti zariadenia a iným. Vzhľadom na to, že je zdrojový kód frameworku voľne dostupný, tak na jeho základe vznikli derivované frameworky akými sú PhoneGap či Ionic [30, 45]. Tieto frameworky ponúkajú rôzne rozšírenia, predpripravené prvky užívateľského rozhrania a zároveň ponúkajú rozličné nástroje a služby, ktoré zjednodušujú vytváranie a publikovanie aplikácií na Google Play či AppStore.

3.3 Multiplatformové natívne aplikácie

Ďalší spôsob vytvárania mobilných aplikácií sú multiplatformové natívne aplikácie [11], ktoré nevykresľujú užívateľské rozhranie iba do WebView ale pracujú priamo so všetkými natívnymi prvkami užívateľského rozhrania (widgetmi) dostupnými pre danú platformu (viď obrázok 3.4). Frameworky, ktoré umožňujú vytváranie takýchto aplikácií, slúžia ako vrstva medzi vývojárom a platformou, kedy framework zabezpečuje na základe popisu zo zdrojového kódu vytvorenie a ovládanie korešpondujúcich natívnych widgetov s príslušnou stavbou, funkcionalitou a hodnotami, ktoré majú zobrazovať [49].



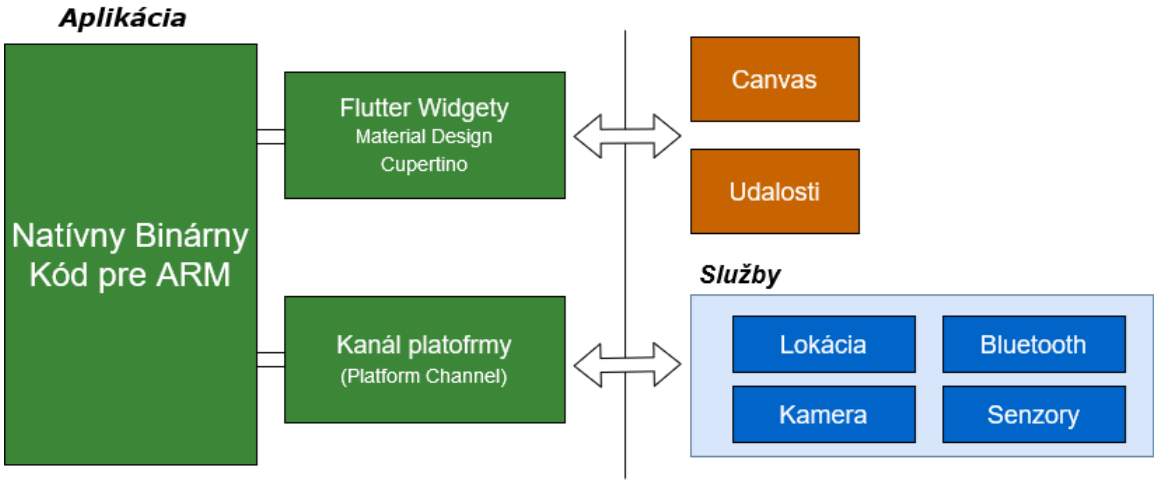
Obr. 3.4: Spôsob fungovania aplikácií vytvorených multiplatformovou technológiou. Zdroj: [43] spracovanie: vlastné

Vytvorenie natívnych widgetov, ktoré sa majú na obrazovke zobraziť, je možné zabezpečiť dvomi spôsobmi. Prvý z nich je realizovaný pomocou pripravených knižníc frameworku, ktoré sú implementované v natívnom programovacom jazyku pre každú podporovanú platformu [11] (tento prístup využíva napríklad framework Xamarin [31]). Negatívom tohoto spôsobu je, že v prípadoch, kedy programátor vyžaduje funkcionalitu, ktorú tieto knižnice nepodporujú, musí písať platformovo špecifický kód pre obe platformy [11]. Druhým spô-

sobom je komunikácia aplikácie s natívnym aplikačným rozhraním, ktoré vie zabezpečiť ich vykreslenie. Tento prístup využíva napríklad framework React Native [1, 41], ktorý na komunikáciu pre vykresľovanie natívnych widgetov využíva tzv. bridge. Nevýhodou takého prístupu je, že takýto spôsob komunikácie zvyšuje réžiu aplikácie, čo môže mať v konečnom dôsledku vplyv na rýchlosť behu aplikácie [11].

3.4 Flutter

Prvá verzia nástroja Flutter bola sprístupnená verejnosti roku 2017 [11] a prvá stabilná verzia v decembri roku 2018 [16]. Vzhľadom na tieto dátumy je zjavné, že nástroj je pomerne nový v príslušnom odbore. Technológia mení celkový prístup k vytváraniu multiplatformových natívnych aplikácií. Namiesto toho, aby Flutter [49] využíval natívne API platformy k vytvoreniu užívateľského rozhrania pomocou natívnych widgetov, tak dané widgety iba napodobňuje a vykresľuje do vlastného Skia kanvasu. V tomto smere sa vykresľovanie výsledného obrazu veľmi zjednoduší a odstránia sa problémy, ktoré nastávali pri vyššie spomenutých multiplatformových technológiách. Faktom je, že pri vytváraní aplikácií musí do aplikácie zaobaliť aj samotný vykresľovací engine, čím sa následne aj zväčšuje veľkosť samotnej aplikácie [49]. Prístup k senzorom, kamere, geolokačným dátam a iným službám je zabezpečený pomocou tzv. Kanálu platformy [26] (Platform channel), ktorý dokáže využívať natívne aplikačné rozhranie zariadenia na získanie potrebných informácií a aktivovaní prvkov. Bližší princíp fungovania aplikácie vytvorenej v technológii vyobrazuje obrázok 3.5.



Obr. 3.5: Princíp fungovania aplikácií vytvorených technológiou Flutter. Zdroj: [43] spracovanie: vlastné

Výhodou technológie Flutter [11, 49] je okrem vyššie spomenutého aj to, že dokáže zdrojový kód priamo preložiť do natívneho kódu procesoru a nemusí byť prekladaný za behu programu, ako to je v prípade jazyka JavaScript, ktorý používa napríklad framework React Native. Táto vlastnosť má tiež pozitívny vplyv na rýchlosť behu aplikácie.

Všetko má svoje výhody a nevýhody a ani technológia Flutter nie je výnimkou. Okrem už spomenutej zvýšenej veľkosti aplikácie, ktorá je dôsledkom zaobalenia vlastného vykresľovacieho systému, taktiež technológia nepracuje priamo s natívnymi widgetmi ale ich iba

napodobňuje [49]. Tým pádom je nutné vždy po aktualizácii vzhľadu natívnych widgetov, aby bol v technológii aktualizovaný aj korešpondujúci widget, ktorý ho napodobňuje. Preto sa môže stať, že aplikácia vytvorená pomocou tohto frameworku bude vyzerať ako zastaralá až do aktualizácie frameworku a aplikácie. Taktiež ide o novú technológiu, preto ešte nie sú vychytané všetky chyby, a občas sa pri vývoji môže objaviť problém, ku ktorému nie je možné nájsť jednoducho návod na vyriešenie. Tento problém je z časti kompenzovaný kvalitnou dokumentáciou.

3.5 Princípy návrhu užívateľského rozhrania

Vzhľadom na fakt, že nie som v odbore návrhu užívateľského rozhrania veľmi skúsený, tak som sa rozhodol preštudovať a prevziať určité odskúšané princípy od skúsenejších návrhárov a dizajnérov. Okrem správnych zásad užívateľského rozhrania pre mobilné zariadenia som sa taktiež rozhodol analyzovať zdroje popisujúce webové aplikácie, pretože kľúčové princípy návrhu užívateľského rozhrania sú využiteľné aj pre mobilné aplikácie. Každá z nasledujúcich zásad sa snaží podporiť v pozitívnom zmysle užívateľský prežitok (anglický User Experience) danej aplikácie.

Menej je niekedy viac

Existuje princíp použiteľnosti [36], ktorý hovorí, že ak niečo reálne vyžaduje alebo vyzerá, že potrebuje veľkú časovú investíciu, tak je menej pravdepodobné, že sa to bude používať. Dôkazom úspešnosti a pravdivosti tohto princípu sú aj aplikácie z kapitoly 2. Príkladom je produkt Duolingo, ktorého hlavná štruktúra pozostáva z veľkého počtu krátkych kvízov, alebo prístup k výuke aplikácie Khan Academy, kde celistvé učivo rozdeľujú na krátke časti, ktoré vyžadujú menej času.

Užívatelia neradi premýšľajú

Užívatelia by mali hneď na prvý pohľad pochopiť užívateľské rozhranie a rozloženie aplikácie [36], aby sa v nej vedeli ľahko orientovať. Ľudia nemajú radi, ak musia užívateľské rozhranie študovať, pretože im to zbytočne zaberá ich drahocenný čas. Preto musí byť užívateľské rozhranie navrhnuté tak, aby bolo na prvý pohľad jasné, na aký účel element užívateľského rozhrania slúži a čo je jeho úlohou. Podľa štúdie [38] sú užívatelia schopní už po 50 milisekundách určiť, či sa im stránka páči alebo nie.

Užívatelia skenujú, nie čítajú

Návštevníci vo väčšine prípadov venujú veľmi málo času čítaniu celého obsahu stránky a iba ho skenujú, pretože sú v zhone alebo jednoducho vedia, že nemusia všetko čítať [36]. Primárnym cieľom užívateľa je snaha dostať sa čo najskôr k chcenému obsahu. Preto je nutné, aby grafické užívateľské rozhranie dodržiavalo všeobecne známe a zažité konvencie pre užívateľa. Taktiež je veľmi dôležité a odporúčané, aby samotná hierarchia stránky alebo aplikácie bola jednoducho pochopiteľná [36]. To znamená, že užívateľské rozhranie musí byť intuitívne.

Bezpečnosť a pohodlnosť

Každý užívateľ sa musí cítiť bezpečne a pohodlne pri práci s aplikáciou, alebo webovou stránkou [36]. Niekedy sa napríklad môže užívateľ na stránke stratiť, alebo vykonať nechcenú akciu. Preto by malo užívateľské rozhranie umožniť návrat na hlavnú stránku z podstránky, alebo poskytnúť možnosť zrušenia alebo návratu vykonávania akcie, inak by to mohlo pre užívateľa znamenať nepríjemnú a nepohodlnú skúsenosť.

Dôležitosť testovania

Testovanie [35, 36] je dôležitým prvkom vývoja a to už v rannej časti vývoja webovej stránky alebo aplikácie. Vykonávanie testovania pomocou jedného užívateľa v počiatočnej dobe vývoja môže priniesť väčší úžitok ako testovanie 50 užívateľov v konečnej fáze vývoja. Nielenže sa odhadnú a odstránia dôležité chyby návrhu, ale taktiež ak je testovanie vykonané až v konečnej fáze, sú časové náklady na zmenu vyššie a zároveň môže dôjsť k nadväznosti chyby k ostatným častiam užívateľského rozhrania.

Všeobecne známe zásady

Pre úplnosť si pripomeňme ďalšie podstatné a všeobecne známe zásady, ktoré by malo užívateľské rozhranie spĺňať.

1. **Responzívnosť** – každá dobrá aplikácia by mala disponovať grafickým užívateľským rozhraním, ktoré sa vždy správne prispôsobí k rôznym rozmerom obrazovky a typu zariadenia.

2. **Komunikácia s užívateľom** – produkt by mal vždy oboznámiť užívateľa o stave aplikácie, o možných akciách, ktoré môže používateľ vykonať a prípadne o funkciách, ktoré sa aktuálne vykonávajú.

3. **Rýchla odozva** – nie je vhodné, ak užívateľ čaká na reakciu aplikácie, každá reakcia grafického užívateľského rozhrania musí byť okamžitá, prípadne v čo najkratšom čase.

4. **Zahltenie informáciami** – každá časť aplikácie by mala zobrazovať výhradne potrebné informácie. Pri zobrazovaní množstva nepotrebných informácií sa užívateľské rozhranie stáva neprehľadné. Tiež je nevhodné, aby aplikácia nútila užívateľa k tomu, aby si pamätal veľa informácií, ktoré sú potrebné na vykonanie jednoduchej akcie.

5. **Funkcionalita** – vysoká chybovosť a nečakané chovanie aplikácie vedie k zlej skúsenosti užívateľa s produktom, čo môže v konečnom dôsledku viesť k strate užívateľov. Preto je potrebné, aby bolo užívateľské rozhranie navrhnuté tak, aby užívateľ vždy vedel, aká má byť reakcia k vykonanej akcii, ktorú vykoná. Napríklad inkrementáciu počtu preskokov cez švihadlo v aplikácii nebude užívateľ vykonávať pomocou tlačidla s ikonou domčeka.

6. **Vhodný spôsob podania informácie** – vždy je nutné premýšľať, akými spôsobmi je možné podať chcenú informáciu a vybrať ten najvhodnejší. V niektorých prípadoch je dobrým výberom graf, v iných zase tabuľka, zoznam, obrázok, text či ich kombinácia.

3.6 Agilný vývoj

Existuje mnoho rozdielnych prístupov [44] k vývoju softvéru, ktoré sa líšia v mnohých vlastnostiach či ideológiách. Niektoré z nich fungujú lepšie pre veľké projekty, iné zase naopak pre menšie. Mnohé sa zameriavajú na detailnú analýzu a postupný návrh, implementáciu, testovanie a doručenie projektu (vodopádový model), pričom tieto kroky sa môžu iteračne opakovať (inkrementálny model). Jednou z vlastností týchto prístupov je, že sa najviac času venuje návrhu riešenia, pričom reálny produkt vzniká až na konci vývojového cyklu. Ak je produkt výborne navrhnutý, tak sú tieto prístupy vhodné. Na druhej strane, ak bol návrh nekvalitný, tak ďalšie úpravy aplikácie sú veľmi náročné na čas a peniaze.

Presným opakom je agilný prístup k vývoju aplikácií. Medzi tieto vývoje patria metodiky [44] ako Scrum, Extrémne programovanie, RUP (Rational Unified Processs), AUP (Agile Unified Process) a podobne. Každá z nich detailne popisuje vlastný spôsob vývoja aplikácie, no zdieľajú spoločné kľúčové vlastnosti a princípy, ktoré sú uvedené v manifeste agilného vývoja – Agile Manifesto [8, 44]. Medzi najdôležitejšie črty, ktorými sa vyznačuje, možno zaradiť:

- Funkčný produkt je dôležitejší ako kvalitná a podrobná dokumentácia.
- Najvyššou prioritou je uspokojiť zákazníka skorým a sústavným dodávaním hodnotného softvéru.
- Lepšie je rýchlo reagovať na zmenu ako sa držať navrhnutého plánu.
- Funkčný softvér je základným ukazovateľom progresu.
- Jednoduchosť – vykonať naozaj len to potrebné, je nevyhnutnosť.

Oproti spomenutým prístupom je agilný prístup vhodný pre malé a samoorganizované tímy. Vzhľadom na zadanie a vymedzený čas pre vývoj sa pre potreby diplomovej práce zdalo byť najvhodnejšie využitie agilného prístupu.

Kapitola 4

Vybrané technológie a nástroje

Pred návrhom a implementáciou užívateľského rozhranie je nutný výber technológií, pomocou ktorých bude konečná aplikácia realizovaná. Po porovnaní mnohých výhod a nevýhod rôznych prístupov a im prislúchajúcim technológiám padlo rozhodnutie využiť pri tvorbe diplomovej práce technológiu Flutter. Zároveň je veľkým pozitívom aj fakt, že táto technológia bola využívaná už v priebehu štúdia, čo umožnilo získanie miernych skúseností, a to by mohlo mať kladný vplyv na časovú náročnosť pri štúdiu kľúčových konceptov technológie, ale aj pri samotnej implementácii. V tejto kapitole bude poukázané na dôležité vlastnosti programovacieho jazyka Dart a frameworku Flutter.

4.1 Programovací jazyk Dart

Framework Flutter využíva programovací jazyk Dart predovšetkým pre jeho špecifické vlastnosti. Dart [11, 12, 47] je staticky typovaný a je jedným z mála programovacích jazykov, ktorého zdrojový kód môže byť spúšťaný dvomi spôsobmi. Jedným spôsobom je preklad zdrojového kódu pred spustením aplikácie (angl. AOT – Ahead of time compilation) do natívneho kódu pre chcenú platformu – Flutter využíva pri konečnom zostavení aplikácie (anglicky Release), a zároveň môže byť prekladaný počas behu programu (angl. JIT – Just in time compilation) – Flutter využíva pri zostavovaní aplikácie pre ladenie. Táto flexibilita prináša vlastnosť nazývanú ako „Hot Reload“ [9], ktorá umožňuje okamžité zobrazenie zmeny užívateľského rozhrania na základe zmeny zdrojového kódu, čo významne zrýchľuje celkový vývojový cyklus aplikácie.

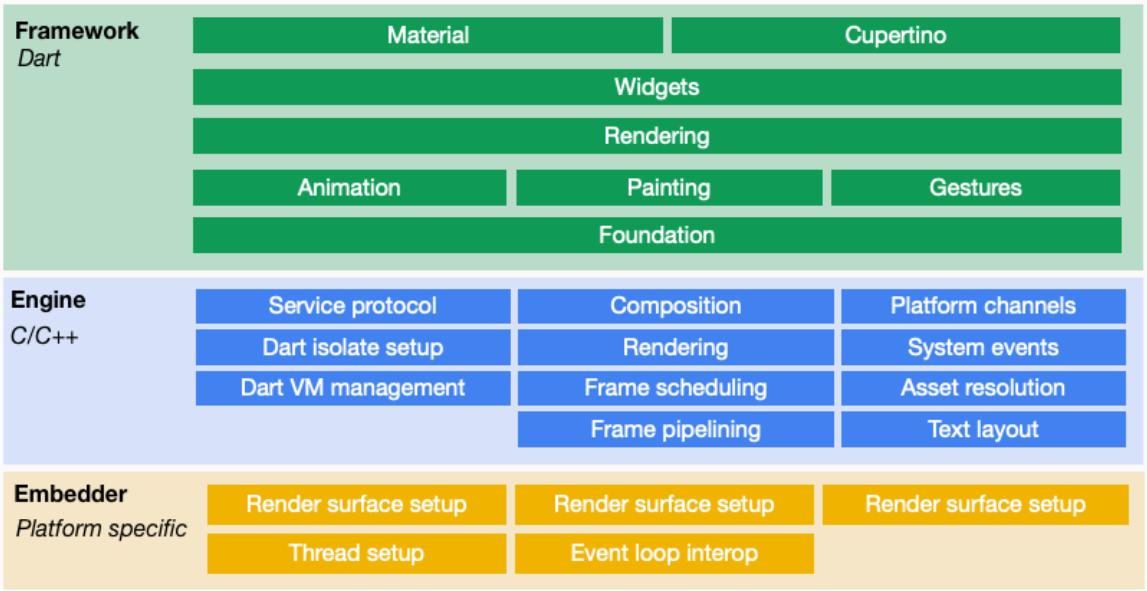
Ďalšou neoddeliteľnou výhodou jazyka pri implementácii užívateľských rozhraní je to, že ide o deklaratívny jazyk [18] a preto nie je nutné využívať na definovanie rozloženia grafického rozhrania rôzne šablónovacie alebo značkovacie jazyky (napríklad XML, HTML, JSON atď.) a iné nástroje na upravovanie layoutu (rozloženia). Zápis je prehľadný a jednoducho je zo zdrojového kódu možné vyčítať stavbu rozloženia. Dart zároveň poskytuje podporu pre rôzne typy testov [25] ako sú Unit testy, Widget testy a integračné testy.

4.2 Flutter Framework

Neoddeliteľnou súčasťou každej aplikácie vytvorenej pomocou tejto technológie je Flutter Engine [27, 40, 49], ktorý je implementovaný v jazyku C++. Táto časť technológie implementuje základné knižnice, ktoré zabezpečujú správny chod aplikácie. Slúžia na vykresľovanie aplikácie do Skia kanvasu, zabezpečujú vstupno výstupné operácie siete alebo

súborov, zabezpečujú spúšťanie aplikácie a tiež umožňujú prístup k natívnemu aplikačnému rozhraniu zariadenia. Ako už bolo spomenuté, tak Flutter Engine je súčasťou aplikácie, čo má za následok zväčšenie konečnej veľkosti aplikácie. Na druhej strane to zabezpečuje multiplatformitu výsledného produktu.

Flutter Framework umožňuje implementovať aplikácie pomocou jazyka Dart [49]. Medzi najdôležitejšie prvky frameworku sa radí knižnica Foundation [11, 49], ktorá poskytuje základné triedy a funkcie, používané programátorom pri vytváraní aplikácií. Jej dôležitosť spočíva v tom, že je pre programátora akýmsi rozhraním, ktoré dokáže komunikovať s Flutter Enginom a zjednocuje prístup k funkciám rôznych natívnych aplikačných rozhraní. Napríklad ak chce programátor zapnúť kameru na zariadení, tak zdrojový kód bude rovnaký bez ohľadu na to, či bude aplikácia spustená na zariadení s operačným systémom Android alebo iOS. Systémovú architektúru technológie Flutter je možné vidieť na nákrese 4.1.



Obr. 4.1: Systémová architektúra technológie Flutter. Zdroj: [24]

Základným stavebným prvkom každej aplikácie vytvorenej pomocou frameworku Flutter sú widgety [49]. Framework poskytuje dve základné témy – pre Android (Material) a pre iOS (Cupertino). Tieto predpripravené témy, obsahujú sady widgetov, ktoré napodobňujú natívne widgety danej platformy. Samozrejme je programátorovi alebo dizajnérovi umožnené definovať aj vlastné widgety a témy. Framework zároveň umožňuje definovať animácie [14] a programovanie callbackov pre rôzne užívateľské gestá [23] vykonané nad widgetmi. Widget [2] predstavuje predpis vzhľadu a rozloženia častí užívateľského rozhrania, ktoré opisuje. Túto hierarchiu a stavbu definuje pre každý widget metóda `build()` [40]. Dôležitou vlastnosťou je, že takmer všetko je widget a jednotlivé widgety sa môžu ľubovoľne zanorovať – využíva princíp kompozície[49]. Widgetom je napríklad tlačidlo, detektor gest, ba dokonca aj samotná aplikácia je považovaná za widget.
Na základe vlastností sa widgety rozdeľujú do troch typov:

- Bezstavové widgety (Stateless)
- Stavové widgety (Stateful)

- Zdedené widgety (Inherited)

4.2.1 Bezstavové widgety

Bezstavový widget [2, 11] predstavuje najjednoduchší a najzákladnejší typ widgetov. Tieto widgety sa používajú v prípade, kedy obsah a stavba widgetu je nemenná alebo vstupné parametre pre widget – napríklad reťazec, ktorý sa ma zobraziť, je konštantný. Bezstavový widget [17, 40] sa prekresľuje iba v prípade, že bol vložený do stromu widgetov, jeho rodičovský (parent) widget zmenil svoju konfiguráciu, alebo ak sa zmenila konfigurácia zdedeného (Inherited) widgetu na ktorom závisí. Viac informácií o zdedených widgetoch bude uvedených v sekcii 4.2.3. Jednoduchým príkladom bezstavového widgetu môže byť prvok užívateľského rozhrania, ktorý zobrazí chcený text na nemennom pozadí vybranej farby.

4.2.2 Stavové widgety

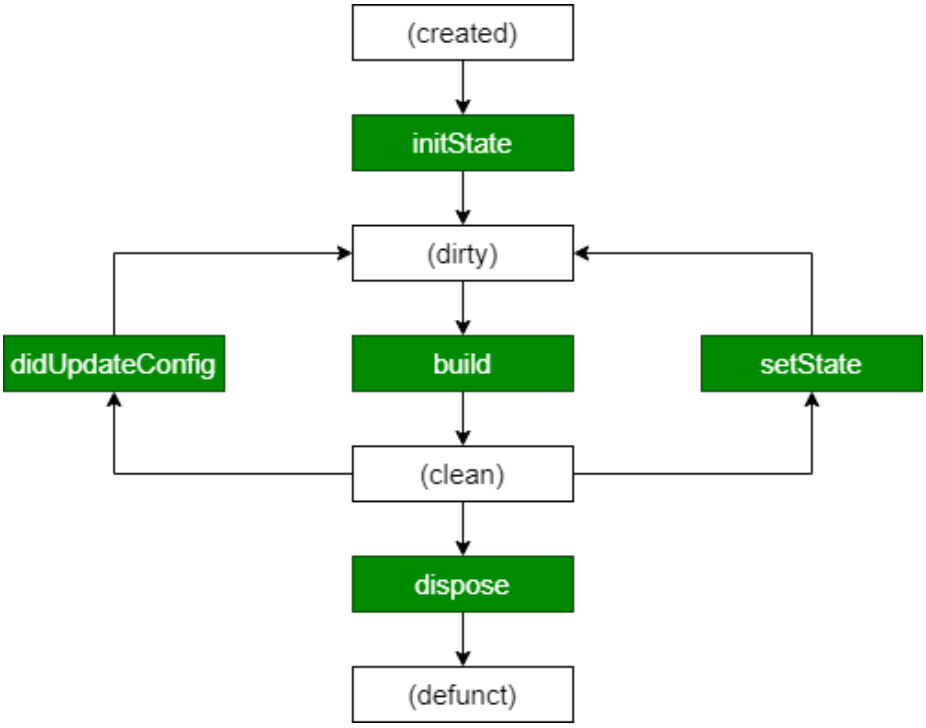
Stavové widgety [2, 11, 49] sa využívajú pre prípad, kedy má byť widget dynamický, teda časť užívateľského rozhrania, ktorú opisuje, sa môže počas životnosti meniť. Ide o rozšírenie bezstavového widgetu o triedu, ktorá definuje aktuálny stav widgetu v danom čase (je schopný uchovávať a meniť hodnoty stavových dát). Konštantné hodnoty sú predávané ako parametre cez konštruktor (rovnako ako pri bezstavovom widgete), a premenné, ktoré sú dynamické, sú definované v samotnom stave. Stav a samotný widget sú definované ako samostatné triedy [11]. Prepojenie widgetu so stavom zabezpečuje metóda `crateState()` v triede widgetu. Jednoduchým príkladom je počítadlo, ktoré dynamicky mení zobrazovanú hodnotu, keď naňho užívateľ klikne, tak sa inkrementuje jeho hodnota uložená v stave a prekreslí sa s novou aktuálnou hodnotou. Explicitné označenie widgetu, ktorý má byť prekreslený, je vykonané pomocou volania metódy `setState()`.

Životný cyklus stavového widgetu

Po vytvorení stavového widgetu je vždy evokovaná metóda `initState()` [21, 40], kde je možné inicializovať hodnoty daného stavu. Po inicializácii sa widget označí ako „dirty“. Tento stav označuje to, že daný prvok užívateľského rozhrania a všetkých jeho potomkov je nutné prekresliť. Pri prvom vykreslení ale aj opakovanom prekresľovaní widgetu je vyvolaná metóda `build()` [40], ktorá definuje samotnú stavbu a rozloženie daného prvku užívateľského rozhrania. Po vykreslení sa widget označí ako „clean“ [40], čo označuje, že ho nie je nutné v ďalšom snímku prekresľovať. V prípade, že sa zmení konfigurácia widgetu, alebo je programátorom vyvolaná funkcia `setState()`, tak sa vráti do stavu „dirty“ [40]. Pred samotným odstránením widgetu z hierarchického stromu widgetov sa volá metóda `dispose()` [21, 40], ktorú môže programátor využiť napríklad k uvoľneniu zdrojov, ktoré daný widget využíva. Následne sa widget označí ako „defunct“ [10] a nie je možné s ním ďalej pracovať. O takto označený objekt sa následne postará garbage collector, ktorý ho odstráni. Tento kolobeh znázorňuje obrázok 4.2.

4.2.3 Zdedené widgety

Tretím typom sú zdedené widgety [17]. Využívajú sa na efektívne šírenie hodnôt vlastností k widgetom, ktoré sa nachádzajú v stromovej hierarchii pod nimi. Namiesto toho, aby sa napríklad hodnota podávala ako parameter konštruktóru z widgetu na widget, tak môže ktorýkoľvek widget pristúpiť k vlastnosti najbližšieho dedeného widgetu v strome a získať



Obr. 4.2: Životný cyklus stavu stavového widgetu. Zdroj: [10] upravené

hodnotu chcenej vlastnosti. Pomocou týchto widgetov je možné implementovať napríklad tému písma, pričom k týmto hodnotám pristúpia iba widgety, ktoré ich potrebujú. Empiricky možno tvrdiť, že využívaním tohto typu widgetu sa stáva kód prehľadnejším.

4.3 Vývojové prostredie

Vytvárať mobilné aplikácie pomocou technológie Flutter je možné takmer pomocou ľubovoľného editora na text alebo vývojového prostredia. Avšak Android Studio [20], XCode [19] a Visual Studio Code [20] sú programy, ktoré sú oficiálne odporúčané na prácu s technológiou Flutter. Oproti ostatným produktom obsahujú všetky potrebné moduly a nástroje alebo prípadne poskytujú možnosť inštalácie rozšírení vývojového prostredia, ktoré uľahčujú a urýchľujú vývoj aplikácie. Osobne mám skúsenosť s vývojovým prostredím Android Studio a editorom Visual Studio Code, pričom preferujem druhú možnosť.

Kapitola 5

Návrh a štruktúra aplikácie

Táto kapitola sa bude vo svojich sekciách bližšie venovať vymedzeniu celkových požiadavok na predmetnú aplikáciu, opisu jej logickej štruktúry, štruktúre dát kurzu z pohľadu tvorcu, štruktúre dát z pohľadu aplikácie a analýze implementácie backendu aplikácie.

5.1 Vymedzenie požiadavok

Cieľom tejto diplomovej práce je vytvorenie elektronického nástroja pre správu inovatívnych študijných materiálov. Z toho dôvodu je nutné si vymedziť, aké kľúčové požiadavky by mal takýto elektronický nástroj obsahovať, a to predovšetkým za tým účelom, aby bolo možné v konečnom dôsledku po naplnení všetkých podmienok zhodnotiť, či došlo k naplneniu cieľa tejto diplomovej práce. Týmito požiadavkami sú:

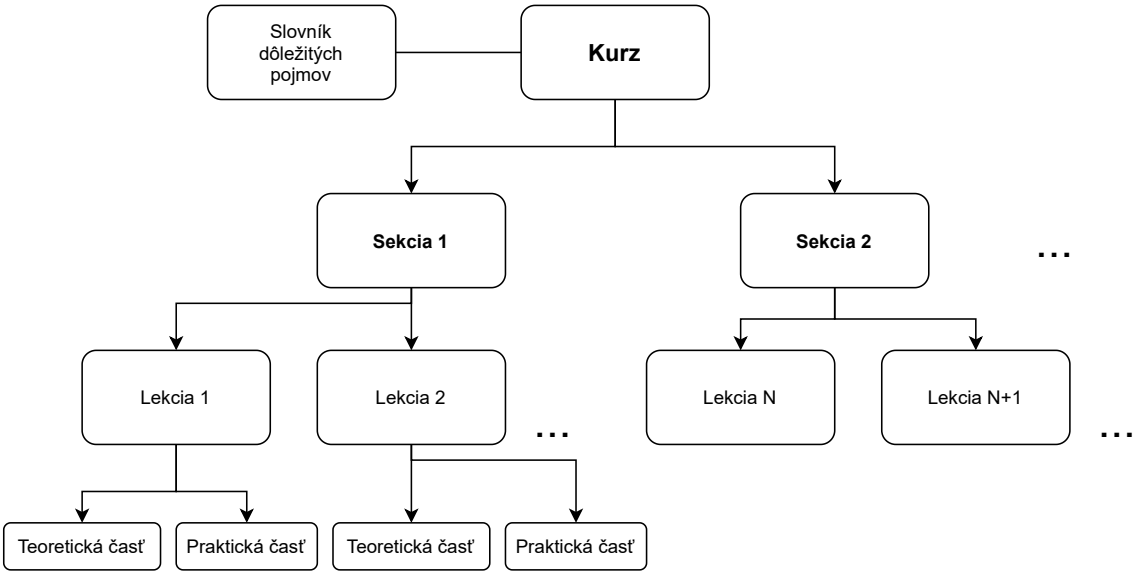
1. Multiplatformnosť aplikácie na zariadenia využívajúce operačné systémy Android a iOS, ako podmienka možnosti širšieho záberu na užívateľskom trhu.

2. Maximalizácia obšírnosti a rozličnosti vytvoreného obsahu, prostredníctvom vytvorenia takých vlastností aplikácie, ktoré dokážu pokryť čo najširšiu škálu predmetov vyučovaných na rôznych stupňoch školstva, s ohľadom na ich jednotlivé požiadavky a špecifikácie ako napríklad prehrávanie zvukov pre potreby správnej výslovnosti pri výuke jazykov, vykresľovanie rovníc v chemických a fyzikálnych vedných odboroch či matematike a podobne.

5.2 Logická štruktúra prezentovania látky

Na samom začiatku vývoja aplikácie bolo dôležité navrhnúť logickú štruktúru, ktorá by zabezpečila vhodné prostredie pre prezentovanie vyučovaných znalostí na mobilnom zariadení. Logická štruktúra pre prezentovanie látky pozostáva z troch častí – kurzu, sekcie a lekcie. Kurz možno definovať ako komplexný tematický okruh, teda ako celok, ktorý obsahuje konkrétne znalosti učiva prezentovanej problematiky, ktoré chce autor kurzu priblížiť jednotlivým užívateľom danej aplikácie. Následne je kurz rozčlenený do sekcií, ktoré predstavujú množiny obsahovo blízkych lekcií vyučovanej problematiky. Tým sa dostávame k pojmu lekcia, ktorú možno považovať za základnú jednotku celého kurzu, obsahujúcu konkrétne znalosti predloženého učiva, ktoré má užívateľ nadobudnúť za účelom poznania a porozumenia problematiky kurzu ako celku. Samotná lekcia pozostáva z dvoch častí – teoretickej, obsahujúcej teoretické poznatky predkladaného učiva, a časti praktickej, ktorá

užívateľom umožňuje preveriť si vedomosti na tému uvedenej lekcie. Osobitným inštitútom, nezávislým od usporiadania logickej štruktúry, nachádzajúcim sa v kurze, je slovník dôležitých pojmov, ktorý vo svojom obsahu prezentuje základné pojmy rozoberaného tematického okruhu. Diagram logickej štruktúry aplikácie je zobrazený na obrázku 5.1.



Obr. 5.1: Logická štruktúra kurzu

5.2.1 Priebeh vývoja logickej štruktúry

Počiatočný návrh logickej štruktúry bol čiastočne motivovaný a vytvorený na základe analýzy existujúcich riešení z kapitoly 2. Z predmetných riešení bolo následne možné vyvodiť dôležité prvky týchto aplikácií, ktoré sú aplikovateľné osobitne aj pre účely tejto diplomovej práce. V prvej verzii aplikácie logická štruktúra pozostávala iba z kurzu obsahujúceho základne prvky – lekcie. Postupom času sa však ukázalo, že pri kurzoch, ktoré disponovali zložitejšou štruktúrou, bolo nutné vložiť určitý vzťah usporiadania jednotlivých lekcií vo forme sekcií, čo v konečnom dôsledku umožnilo lepšie štrukturalizovať celkový obsah. Praktická časť lekcie vo forme preverovania nadobudnutých vedomostí bola súčasťou plánu logickej štruktúry už od samotného počiatku formovania prvotného plánu, pretože takúto formu interaktívnej účasti užívateľa pri prehľadávaní kurzu možno považovať za zaujímavú, no zároveň sa aj osvedčila v aplikácii Duolingo, ktorá je práve vďaka tejto užívateľskej možnosti známa na trhu edukačných prostriedkov používaných na mobilných zariadeniach. V samotnom závere procesu bol do logickej štruktúry aplikácie zaradený v časti kurzov aj slovník dôležitých pojmov, a to konkrétne z dôvodu potreby lepšej prehľadnosti existencie sumarizačného zoznamu dôležitých pojmov, ktoré sú spoločné pre celú rozoberanú problematiku predmetného kurzu. Pre demonštrovanie vhodnosti a širokého využitia navrhnutej štruktúry boli vytvorené tri kurzy z rôznych vedeckých oblastí – história, informačné technológie a biológia.

5.3 Štruktúra dát kurzu z pohľadu tvorcu

Cieľom tejto práce je tiež zabezpečiť to, aby si kurz mohol vytvoriť v podstate ktokoľvek bez rozdielu na to, či sa jedná o učiteľa, študenta alebo akékoľvek osoby, ktoré by chceli zdieľať svoje vedomosti s inými. Preto sa žiadalo navrhnúť takú štruktúru dát, ktorá by kurz ako celok reprezentovala logicky, a zároveň by bola čo najviac zhodná s logickou štruktúrou.

Kurz je distribuovaný ako archív vo formáte Zip, pozostávjúci z niekoľkých súborov, z ktorý niektoré sú povinné, iné sú naopak nepovinné. Avšak takmer všetky sa vyznačujú pevne danou vnútornou štruktúrou. V koreňovej zložke kurzu sa povinne musia nachádzať tieto súbory:

- `Course.json` – tvorca definuje základné informácie o kurze, ako názov, autora ale hlavne štruktúru resp. hierarchiu samotného kurzu.
- `Dictionary.json` – obsahuje kľúčové pojmy, ich definície a taktiež aj ich synonymá.

Detailný popis vnútornej štruktúry súboru `Course.json` je bude popísaný v sekcii 6.3 a popis súboru `Dictionary.json` v sekcii 6.4. Ostatné súbory definovateľné tvorcom kurzu sú opísané v kapitolách 6 a 7.

Ďalšou nutnosťou bolo nájdenie vhodného formátu súboru, ktorý by bol vhodný na definíciu dát kurzu a zároveň je nenáročný na pochopenie aj pre obyčajného človeka. Existuje niekoľko značkovacích jazykov (napr. XML, JSON, YAML ...), ktoré tieto požiadavky spĺňajú. No vzhľadom na prehľadnosť, čitateľnosť, popularitu či upísanosť formátov sa pre mňa javila ako najlepšia možnosť formát JSON.

Formát JSON, ktorý sa používa v tomto projekte, je navyše rozšírený o podporu odkazov (referencií) na externé súbory, pod ktorými možno rozumieť napríklad obrázkové súbory, zvukové ale aj textové súbory. Toto rozšírenie bolo vytvorené na základe myšlienky, ktorá by umožňovala v niektorých prípadoch tvorcovi podať informáciu aj iným spôsobom ako klasickým textom a to napríklad formou obrázka alebo zvuku. Zároveň sa ukázalo, že tento prvok navyše udržuje prehľadnosť dokumentu a celkovej štruktúry kurzu. Referencia sa označuje prefixom „@“, za ktorým nasleduje cesta k odkazovanému súboru. Ak sa cesta začína znakom „~“, tak sa jedná o definovanie absolútnej cesty vzhľadom ku koreňovému priečinku kurzu. V opačnom prípade sa berie cesta relatívna od súboru v ktorom sa referencia nachádza. Čo sa týka pohľadu aplikácie na vykresľovanie, tak v prípade detekcie referencie aplikácia určí typ tejto referencie na základe koncovky súboru na ktorý odkazuje (napr. obrázok – png, jpg, gif atď, alebo zvuk – mp3, wav, avi, ale aj text – txt). Príkladom môže byť nasledujúca ukážka znázornená v tabuľke 5.1.

5.4 Štruktúra dát z pohľadu aplikácie

Aplikácia uchováva dáta všetkých kurzov v priečinku `Courses`. Tento priečinok obsahuje priečinky všetkých kurzov nainštalovaných v aplikácii. Priečinky jednotlivých kurzov sú nazvané podľa unikátneho identifikačného reťazca pre daný kurz, ktorého bližšia charakteristika je uvedená v kapitole 6.1.1). Každý priečinok kurzu v sebe zaobaľuje štruktúru troch podpriečinkov a jedného súboru:

Definovaný JSON	Vykreslenie v aplikácii	Typ
`{ "show": "./Blue.png" }`	./Blue.png	Text
`{ "show": "@./Blue.png" }`		Obrázok
`{ "show": "@./Blue.txt" }`	Toto je obsah súboru Blue.txt	Text

Tabuľka 5.1: Príklad evaluovania referencií

- `info.educatorium` – súbor obsahuje URL, z ktorej bol kurz stiahnutý.
- `Download` – priečinok na dočasné ukladanie stiahnutých dát z internetu.
- `UserData` – priečinok obsahuje súbory, ktoré vytvoril tvoriteľ kurzu.
- `GeneratedData` – priečinok obsahuje súbory, ktoré si generuje aplikácia (napr. obrázky matematických rovníc).

Bližší popis využitia týchto súborov a priečinkov bude popísaný v kapitole 6.1.1.

5.5 Implementácia backendu aplikácie

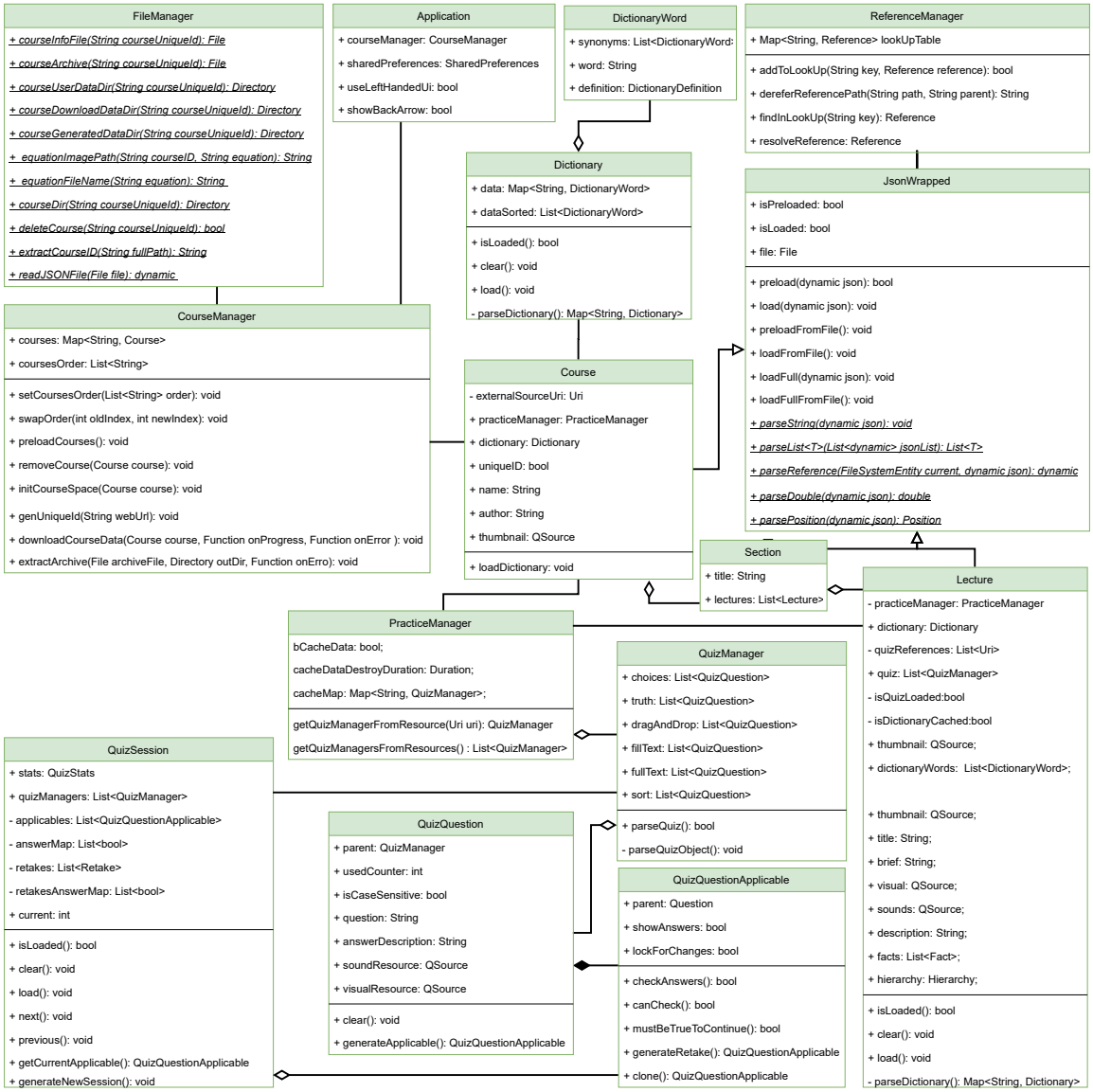
V tejto časti textu budú popísané základné triedy implementujúce hlavné funkčné jadro aplikácie. Relácie medzi jednotlivými triedami je možné vidieť v podobe diagramu tried na obrázku 5.2. Tento diagram obsahuje z dôvodu prehľadnosti iba tie najdôležitejšie triedy a ich najpodstatnejšie metódy a atribúty.

Trieda Application

Trieda `Application` je implementovaná podľa návrhového vzoru Singleton a slúži na zapuzdrenie a zjednodušenie prístupu k aktuálnemu stavu backendových dát z frontendovej časti aplikácie, a to pomocou ukazovateľa na inštanciu triedy `CourseManager`. Zároveň uchováva nastavenia užívateľského rozhrania aplikácie vybrané užívateľom, ktoré umožňuje dialóg nastavení z kapitoly 6.2:

- `useLeftHandedUi` – užívateľské rozhranie má byť prispôsobené pre osoby s preferenciou ovládania pomocou ľavej ruky.
- `showBackButton` – na potrebných miestach užívateľského rozhrania aplikácie zobrazí šípky v ľavom hornom rohu obrazoviek, aby bolo užívateľovi umožnené vykonať krok späť, tak ako je to dobrým zvykom na zariadeniach s operačným systémom iOS.

Atribút `sharedPreferences` obsahuje ukazovateľ k objektu typu `SharedPreferences` [22], ktorý umožňuje prístup a ukladanie jednoduchých dát na perzistentnom úložisku zariadenia.



Obr. 5.2: Diagram tried backendu aplikácie

Trieda FileManager

Táto trieda je tvorená výhradne statickými metódami, ktoré aplikácii umožňujú správu (sprístupnenie, vytváranie a odstraňovanie) hlavného pamäťového priestoru aplikácie a ich súborov definovaným v sekcii 5.4. Taktiež implementuje sadu metód pre prácu s cestami jednotlivých súborov – pri generovaní obrázkov matematických vzorcov (napríklad vytvorenie názvu súboru na základe popisu vzorca), vyvodenie unikátneho identifikátora kurzu z absolútnej cesty k súboru a iné.

Trieda CourseManager

Trieda `CourseManager` uchováva stav a informácie o jednotlivých kurzoch. Tvorí akýsi prístupový bod pre ich spravovanie (prístup, vymazanie a pridávanie), kde každý kurz je reprezentovaný pomocou triedy `Course`. Trieda obsahuje dve hlavne atribúty. Atribút `courses`

uchováva ukazovatele na dostupné inštancie kurzov pomocou dátového typu `Map<String, Course>`, kde každý kľúč je unikátny identifikátor kurzu a jeho hodnotou ukazovateľ na objekt typu `Course` s príslušným identifikátorom. Druhým atribútom je pole `coursesOrder`, ktoré predstavuje poradie zobrazovania kurzov na hlavnej stránke aplikácie opísanej v kapitole 6.2. Hodnoty tohto poľa sú nastavované užívateľom na základe reorganizácie kurzov na hlavnej stránke aplikácie. Táto trieda taktiež poskytuje funkcionalitu potrebnú pre inštalovanie jednotlivých kurzov na zariadenie a definuje funkcie, ktoré využíva reťazec sprístupnenia kurzu z kapitoly 6.1.1. Konkrétne sa jedná o stiahnutie súborov z internetovej siete a ich následná dekomprimácia a extrakcia do pamäte zariadenia. Celý proces riadenia reťazca sprístupnenia kurzu a postupné evokovanie spomenutých funkcií zabezpečuje trieda `CourseInstallPipeline`, ktorá na diagrame tried nie je vyobrazená.

Trieda JsonWrapped

Ide o abstraktnú triedu, ktorá poskytuje základnú funkcionalitu pre spracovanie súborov vytvorených tvorcami kurzov a zabezpečuje napríklad evaluáciu referencií upraveného formátu JSON. Trieda taktiež poskytuje rozhranie, ktoré rozdeľuje načítanie dát do dvoch fáz – prednačítanie a načítanie zvyšných dát. To, ktoré informácie sú načítané vo fáze prednačítania, je definované v metóde `preload` a načítanie zvyšných dát v metóde `load`. Rozdelenie načítania dát do týchto dvoch fáz bolo vytvorené výhradne pre zrýchlenie a zníženie pamäťovej náročnosti aplikácie. Jedným z príkladov využitia je nasledovná situácia: Užívateľ si vyberie kurz, ktorý chce prehľadávať. Po otvorení daného kurzu sa mu zobrazí obrazovka so zoznamom lekcií, pričom pre každú lekciu je nutné zobraziť iba jej názov. Až po samotnom otvorení obrazovky, na ktorej má byť vyobrazený kompletný popis teoretickej časti lekcie sa načítajú zvyšné dáta lekcie (napr. konkrétny výukový text). V metóde `preload` bude teda vykonané načítanie názvu danej lekcie a v metóde `load` zvyšných dáta.

Trieda Course

Trieda `Course` je derivátom triedy `JsonWrapped`, ktorá uchováva a spracováva informácie súborov definujúcich kurzy. Táto trieda zároveň riadi prednačítavanie jednotlivých lekcií (definované triedou `Lecture`). Taktiež inštancuje objekt typu `Dictionary` či `PracticeManager` a riadi ich chod. V prípade tejto triedy sa vo fáze prednačítania načítajú základné informácie o danom kurze vyobrazené na obrázku 6.6 (názov kurzu, autor, ilustračný obrázok) a v druhej fáze načítania je vykonané spracovanie súboru so slovníkom a vyvolanie funkcií prednačítania všetkých lekcií kurzu.

Trieda Lecture

Trieda `Lecture`, rovnako ako trieda `Course`, je derivátom triedy `JsonWrapped`, avšak uchováva a spracováva informácie súboru definujúceho lekciu. Vo fáze prednačítavania načíta názov lekcie a jej obrázok a následne vo fáze načítania načíta všetky zvyšné dosiaľ nenačítané časti. Jednotlivé lekcie sú zaobalené do objektu typu `Section` reprezentujúcom sekciu, ktorej daná lekcia prislúcha.

Trieda QuizManager

Praktická časť každej lekcie je realizovaná pomocou kvízov a kvízových otázok. Otázka alebo sada otázok sú definované ich tvorcami v jednotlivých súboroch. Každá inštancia

triedy `QuizManager` reprezentuje sadu otázok obsiahnutú v jedenom z nich. To znamená, že v prípade, keď užívateľ definuje 7 súborov s otázkami, tak je vytvorených 7 inštancií triedy `QuizManager`. Inštancia tento súbor vždy analyzuje pomocou metód `parseQuiz` a `parseQuizObject` a na základe obsahu ich rozdelí do 6 polí podľa typu jednotlivých otázok (Dopĺňanie textu, Zoradenie, Spájanie dvojíc a podobne). Konkrétne typy otázok sú uvedené v nasledujúcej časti textu.

Trieda QuizQuestion

Trieda `QuizQuestion` je abstraktná trieda, ktorá reprezentuje už základný stavebný prvok praktickej časti – otázku. Reprezentuje práve jednu otázku (nie sadu otázok) definovanú užívateľom. Aplikácia rozlišuje 6 typov otázok, ktoré sú implementované ako deriváty triedy QuizQuestion:

1. Otázka výberu z niekoľkých možností – derivovaná trieda `QuestionChoice`.
2. Doplňovanie do obrázku – derivovaná trieda `QuestionMap`.
3. Zoradenie – derivovaná trieda `QuestionSort`
4. Spájanie dvojíc – derivovaná trieda `QuestionPairs`.
5. Dopĺňanie textu – derivovaná trieda `QuestionFillText`.
6. Otázka s plnou odpoveďou – derivovaná trieda `QuestionFullText`.

Uvedené triedy nie sú znázornené na obrázku 5.2 z dôvodu lepšej prehľadnosti predmetného diagramu.

Trieda QuizQuestionApplicable

Vlastnosťou niektorých definovaných otázok, ktoré reprezentuje trieda `QuizQuestion` je, že z jednej definície otázky môže byť vytvorených niekoľko rôznych variácií. Napríklad pri otázke s výberom správnych odpovedí z niekoľkých možností, môže tvorca k jednému zadaniu definovať neobmedzený počet pravdivých tvrdení (napr. 15) a neobmedzený počet nepravdivých tvrdení (napr. 20). K jednému zadaniu by teda bolo dohromady definovaných 35 možností. Z týchto možností sa vždy vyberie iba určitý počet (napr. 4), ktoré sa užívateľovi zobrazia pri aktuálnej otázke vo fáze precvičovania látky. To znamená, že z jednej definície otázok môže byť vygenerovaných niekoľko rôznych realizácií. Trieda `QuizQuestionApplicable` teda predstavuje konkrétnu realizáciu otázky, pričom navyše uchováva odpovede užívateľa na danú otázku a zabezpečuje kontrolu týchto odpovedí.

Aplikácia ako taká rozlišuje medzi dvomi typmi otázok vzhľadom k odpovedi, kedy užívateľ:

- Musí odpovedať správne, aby mohol v kvíze pokračovať – jedná sa o otázky s obmedzeným možným počtom odpovedí (Zoradenie, Spájanie dvojíc, Otázka výberu z niekoľkých možností, Doplňovanie do obrázku).
- Nemusí odpovedať správne, aby mohol pokračovať v kvíze (Doplňovanie textu a Otázka s plnou odpoveďou).

Správanie jednotlivých typov otázok je implementované v derivátoch triedy `QuizQuestionApplicable` a to pomocou definície metód `mustBeTrueToContinue`, `checkAnswers`.

Trieda PracticeManager

Trieda `QuizManager` predstavuje množinu otázok definovanú v jednom súbore. Na tieto súbory môže byť odkazované v jednom kurze aj niekoľkokrát (Jedna sada otázok môže prislúchať viacerým lekciám). Preto musela byť vytvorená trieda `PracticeManager`, ktorá zabezpečuje to, aby nebol jeden súbor s množinou otázok spracovaný viac ako jedenkrát. Inak povedané, aby pre množinu otázok definovanú v rovnakom súbore nevznikla viac ako jedna inštancia triedy `QuizManager`. Tento princíp zároveň zaručuje unikátnosť definovaných otázok v rámci aplikácie. Objekt typu `PracticeManager` funguje na nasledujúcom princípe – obsahuje atribút `cacheMap` dátového typu `Map<String, QuizManager>`, kde kľúčom je absolútna cesta k súboru daného inštanciou objektu typu `QuizManager`. Pred každým ďalším inštancovaním objektu typu `QuizManager` je vykonaná kontrola, či predmetný súbor už nebol spracovaný – či premenná `cacheMap` neobsahuje kľúč v podobe absolútnej cesty k danému súboru. V prípade, že už spracovaný bol, tak sa inštanciovanie nevykoná ale poskytne už existujúci objekt. Tento spôsob implementácie zrýchľuje aplikáciu a zároveň tiež znižuje jej pamäťovú náročnosť – rovnaké dáta nie sú spracovávané a ukladané duplicitne.

Trieda QuizSession

Vstupom triedy `QuizSession` je pole objektov typu `QuizManager`, kde každý z nich obsahuje už spomenutú sadu otázok. Ak si chce užívateľ precvičiť danú látku, aplikácia mu vytvorí kvíz o určitom počte otázok. Trieda `QuizSession` slúži práve na túto funkciu, kedy zo všetkých otázok definovaných spomenutým vstupným poľom objektov vyberie určitú podmnožinu realizácií otázok, ktorú následne bude užívateľ precvičovať. Viac informácií o spôsobe výberu otázok je možné nájsť v kapitole 7.2.

Trieda Dictionary

Trieda Dictionary reprezentuje slovník dôležitých pojmov kurzu. Jeho štruktúra pozostáva z atribútu `data` a dátového typu `Map<String,DictionaryWord>`, kde kľúčom je daný pojem a prislúchajúcou hodnotou je ukazovateľ na objekt typu `DictionaryWord`, ktorý obsahuje detailnejšie dáta a definície daného pojmu.

Trieda DictionaryWord

Trieda `DictionaryWord` uchováva informácie o pojme zo slovníka dôležitých pojmov. Konkrétne pozostáva z atribútov:

- `word` – pojem v podobe reťazca
- `synonyms` – synonymá k danému pojmu v podobe zoznamu ukazovateľov na objekty typu `DictionaryWord`, ktoré tieto synonymá reprezentujú
- `definition` – odkaz na objekt udržujúci definíciu pojmu. Inštancia tohto objektu je medzi synonymami zdieľaná – to znamená, že pre všetky slová s touto definíciou je reťazec definície uchovávaný v celej aplikácii iba jedenkrát.

Kapitola 6

Kurz

V tejto kapitole bude bližšie analyzovaná najobšírnejšia časť logickej štruktúry aplikácie, a to práve kurz. Nasledujúce sekcie tejto kapitoly sa zamerajú na opis procesu sťahovania a spracovania dát jednotlivých kurzov, opis rozloženia hlavnej obrazovky aplikácie a kurzu či na analýzu slovníka dôležitých pojmov.

6.1 Sťahovanie a spracovanie dát

V samotnom počiatku tvorby systému bolo nutné sa zamerať na vyriešenie otázky dostupnosti kurzov a informácií, ktoré v sebe obsahujú. V tomto prípade sa naskytuje niekoľko možností, ako zabezpečiť distribúciu jednotlivých kurzov medzi užívateľov. Jednou z nich je zaobstarať server, ktorý by uchovával všetky vytvorené kurzy na jednom mieste (ako napríklad po vzore Wikipedie), čo by so sebou prinášalo výhody ako ľahšie vyhľadávanie, prehľad, či dostupnosť kurzov, no na druhej strane, táto možnosť so sebou prináša nevýhodu v podobe navýšenia nákladov spojených s obstarávaním, či údržbou takéhoto serveru. Preto aplikácia umožňuje sťahovanie kurzov z akejkoľvek lokality dostupnej pomocou internetovej siete, pričom je len na tvorcovi kurzu, kde dáta umiestni a na užívateľovi, odkiaľ si kurz stiahne. Na druhej strane, v súčasnosti existuje množstvo dostupných služieb, kde môže užívateľ tieto dáta zdarma ukladať (napríklad Github, Gitlab, Google drive a podobne).

6.1.1 Reťazec sprístupnenia kurzu

Reťazec sprístupnenia kurzu umožňuje užívateľovi nainštalovať nový kurz alebo aktualizovať už nainštalovaný kurz. Tieto akcie vykonáva v štyroch esenciálnych krokoch:

1. Alokácia a inicializácia dátového priestoru
2. Stiahnutie kurzu
3. Rozbalenie kurzu
4. Generovanie matematických vzorcov

Alokácia a inicializácia dátového priestoru

Pred samotným stiahnutím kurzu aplikácia vyhradí priestor na pamäťovom disku, kde budú súbory daného kurzu uložené a dostupné. Každý kurz, ktorý je nainštalovaný na zariadení, je definovaný svojim identifikátorom, ktorý je založený na myšlienke, že každá

lokalizácia z ktorej je kurz stiahnutý, je unikátna. To znamená, že tento identifikátor sa vytvára s ohľadom na zdrojovú URL adresu kurzu. Na základe tohto, už konkrétne daného identifikátora, je vytvorený podpriečinok pre budúci kurz v priečinku `Courses` aj spolu so základnou štruktúrou podpriečinkov a súborov definovanou v sekcii 5.4.

Stiahnutie kurzu

Ako už bolo v úvode tejto sekcie uvedené, tak aplikácia umožňuje užívateľovi stiahnuť ľubovoľný kurz z akejkoľvek internetovej lokácie a to konkrétne za pomoci protokolu HTTP alebo HTTPS. Implementácia samotného spojenia je realizovaná pomocou objektov `HttpClient`, `HttpClientRequest`, `HttpClientResponse`, ktoré sú súčasťou základného balíčka jazyka Dart `dart:io`. Tieto objekty poskytujú všetky potrebné funkcie, ktoré boli vyžadované a zároveň aj využité pri implementácii užívateľského rozhrania. Jedná sa napríklad o sledovanie zmeny pokroku sťahovania dát, či kontroly spojenia medzi zariadením a serverom s možnosťou odchytávania príslušných chýb. Informácie o zdroji sťahovania (URL) sú uložené v súbore `info.educatorium`. Súbor vo formáte Zip, ktorý je predmetom sťahovania, je umiestnený v podpriečinku `Download`.

Rozbalenie kurzu

Po stiahnutí dát nasleduje ďalšia fáza, v ktorej sa získaný súbor vo formáte Zip dekomprimuje a jeho obsah rozbalí do priečinku `UserData`. Na tento proces sú využité komponenty `Archive` a `ZipDecode` z knižnice `archive`[1]. Kľúčovým prvkom pri implementácii bolo spustiť tieto operácie mimo hlavného vlákna aplikácie. Dôvodom bolo to, že proces dekomprimácie a rozbaľovania sa ukázal natoľko výpočetne náročný, že spôsoboval blokovanie hlavného vlákna aplikácie, čo narušilo plynulosť chodu užívateľského rozhrania a jeho odozvy na užívateľa. Programovací jazyk Dart, pomocou ktorého je aplikácia implementovaná, ponúka možnosť paralelných výpočtov pomocou tzv. isolates [15], čo sú v podstate podobné pracovné jednotky ako vlákna, avšak s tým rozdielom, že navzájom nezdieľajú žiadny pamäťový priestor, ale komunikujú iba pomocou správ.

Generovanie matematických vzorcov

Na začiatku tejto fázy sa prehľadajú všetky súbory vo formáte JSON a pomocou regulárneho výrazu sa v týchto súboroch nájdu všetky matematické a chemické vzorce, ktoré je potrebné v danom kurze vykresľovať. Rovnice, ktoré sú výsledkom prehľadávania sú unifikované, a to tak, aby neboli zbytočne generované rovnaké vzorce viac ako jedenkrát. Vzhľadom na náročnosť týchto operácií bolo vyhľadávanie matematických vzorcov vykonané konkurentne pomocou triedy `isolate`, tak ako to bolo v predchádzajúcej fáze. Získané rovnice sú na základe ich predpisu následne vygenerované. V aplikácii sa matematické vzorce zobrazujú pomocou vygenerovaných obrázkov vo formáte PNG. Prečo je tomu tak, je vysvetlené v sekcii 6.1.1. Dôležitejšie však je, že tieto obrázky rovníc je potrebné nejakým spôsobom získať. Núkali sa 3 rôzne možnosti: Prvá možnosť bola použiť externé API na sťahovanie rovníc, akým je napríklad Google Charts API od spoločnosti Google Inc. V tomto prípade by však bola aplikácia závislá na inej službe poskytovanej treťou stranou, pričom ako tvorca aplikácie by som nemal absolútne žiadnu kontrolu nad funkčnosťou tohto aplikačného rozhrania, čo by v prípade zlyhania alebo vypnutia tejto služby znamenalo aj ochromenie využívania aplikácie. Druhou možnosťou bolo vytvorenie vlastného servera, ktorý by pomocou nejakej

[1] https://pub.dev/packages/archive

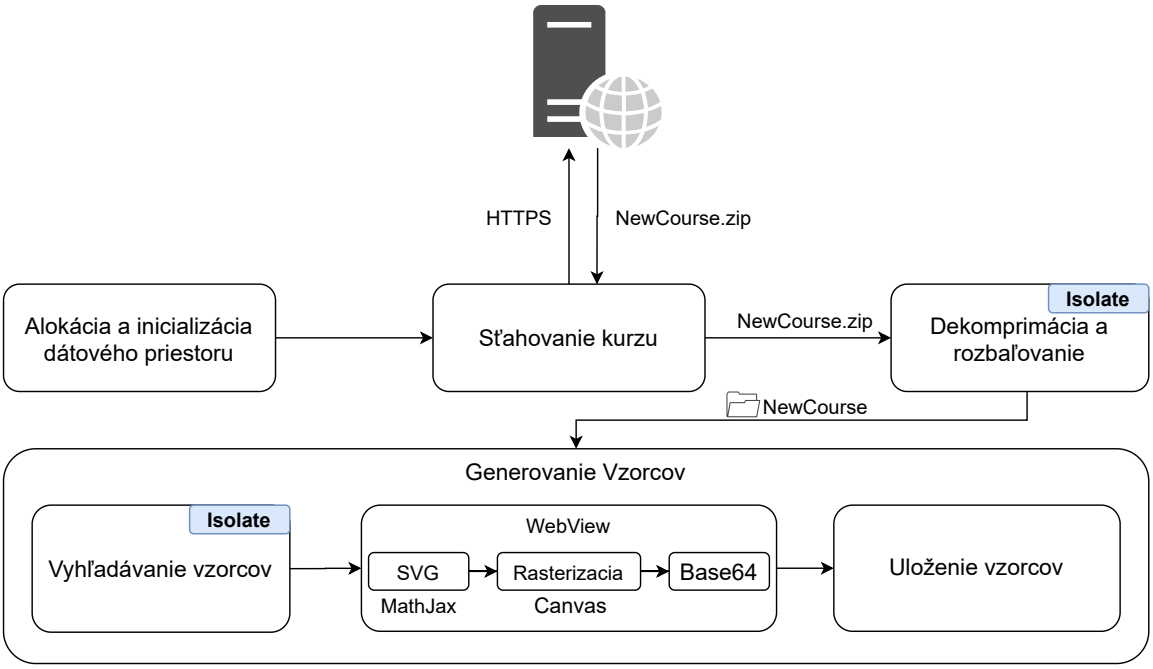
knižnice generoval tieto obrázky a zasielal ich naspäť aplikácii (vlastné API). Výhodou by bola plná kontrola nad týmto systémom, avšak na druhej strane by to vyžadovalo vyššie náklady na prevádzkovanie takéhoto servera. Poslednou a zároveň najideálnejšou cestou sa javilo vytváranie rovníc lokálne na mobilnom zariadení. Veľkým problémom však bol fakt, že na framework Flutter v tomto čase neexistovali žiadne základné, ani externé knižnice, ktoré by dokázali vykresľovať potrebné vzorce do obrázkov. Nakoniec som prišiel s dômyselným nápadom, ako tento problém vyriešiť. Existujú totiž rôzne knižnice napísané v jazyku JavaScript, ktoré slúžia na vykresľovanie matematických a chemických vzorcov, akými sú KaTeX[2] a MathJax[3]. KaTeX má veľkú výhodu v rýchlosti vykresľovania, no oproti knižnici MathJax neumožňuje vykresľovanie do formátu SVG, ani iného obrázkového formátu. Taktiež faktom je, že technológia Flutter pripúšťa inštancovanie natívnych widgetov zariadenia, medzi ktoré patrí `WebView`. `WebView` je widget, ktorý zabezpečuje vykresľovanie webových stránok alebo aplikácií na zariadení, a zároveň umožňuje evaluovať Javascriptové funkcie na danej stránke. Z toho dôvodu dané riešenie spočíva v tom, že som vytvoril lokálnu webovú aplikáciu pomocou jazykov HTML, Javascript a knižnice `MathJax`, na základe ktorej sa vykreslí chcený matematický, či chemický vzorec do formátu SVG. Následne je tento vektorový obrázok pomocou HTML elementu `Canvas` rasterizovaný do obrázku vo formáte PNG. Takýto obrázok však nemožno uložiť priamo pomocou widgetu `WebView`, preto bolo nutné výsledný obrázok rovnice zakódovať do textového reťazca vo formáte base64. Tento reťazec už je ďalej možné získať priamo do mobilnej aplikácie, znovu dekódovať a uložiť do pamäte zariadenia v chcenom formáte PNG. Takéto riešenie prináša so sebou nevýhodu, a to takú, že sa inštalačný reťazec stane priamo závislý na užívateľskom rozhraní, avšak na druhej strane prináša veľmi veľkú výhodu, ktorá robí aplikáciu nezávislú od externých služieb. Pre lepšiu predstavu je diagram sprístupnenia kurzu znázornený na obrázku 6.1. Pre zabezpečenie multiplatformity aplikácie natívny widget `WebView` v aplikácii vykresľovaný pomocou knižnice `webview_flutter_plus`[4], ktorá vytvára rozhranie na vykresľovanie tohto widgetu pre platformu Android aj iOS.

Vlastnosťou tohto reťazca, ktorú je vhodné spomenúť, je fakt, že v prípade, ak sa jedná o aktualizáciu už nainštalovaného kurzu, tak sa generujú iba matematické vzorce, ktoré neboli súčasťou predchádzajúcej verzie kurzu, čím sa výrazne zrýchli proces inštalácie kurzu.

6.1.2 Užívateľské rozhranie

Užívateľské rozhranie sprístupnenia kurzu je implementované vo forme nemodálneho dialógu, avšak s jedným špecifikom. Dialóg zmení svoje správanie na modálne (nie je možné ho zavrieť, kým sa nedokončí akcia) hneď, akonáhle sa užívateľ rozhodne vykonať hlavnú akciu dialógu. Je to z dôvodu, ktorý je opísaný v sekcii 6.1.1, kedy generovanie matematických vzorcov vyžaduje inštanciovanie a prácu s widgetom `WebView`, čím je celý proces priamo závislý na užívateľskom rozhraní aplikácie. Preto nemôže inštalácia prebiehať ako služba pracujúca v pozadí aplikácie. Indikácia, či je možné dialóg zavrieť, je realizovaná pomocou krátkeho bieleho textu „TAP ANYWHERE TO CLOSE“ pod hlavnou časťou dialógu. V prípade, že sa dialóg prepne do modálneho správania, tento text zmizne. Takéto správanie dialógu je možné vidieť na obrázku 6.3.

[2] https://katex.org/
[3] https://www.mathjax.org/
[4] https://pub.dev/packages/webview_flutter_plus



Obr. 6.1: Reťazec sprístupnenia kurzu zabezpečuje alokovanie priestoru pre kurz, sťahovanie kurzu, rozbalenie kurzu a vyhľadávanie či generovanie obrázkov matematických a chemických vzorcov obsiahnutých v kurze pomocou využitia knižnice MathJax.

Hlavná časť dialógu pozostáva z troch základných častí:

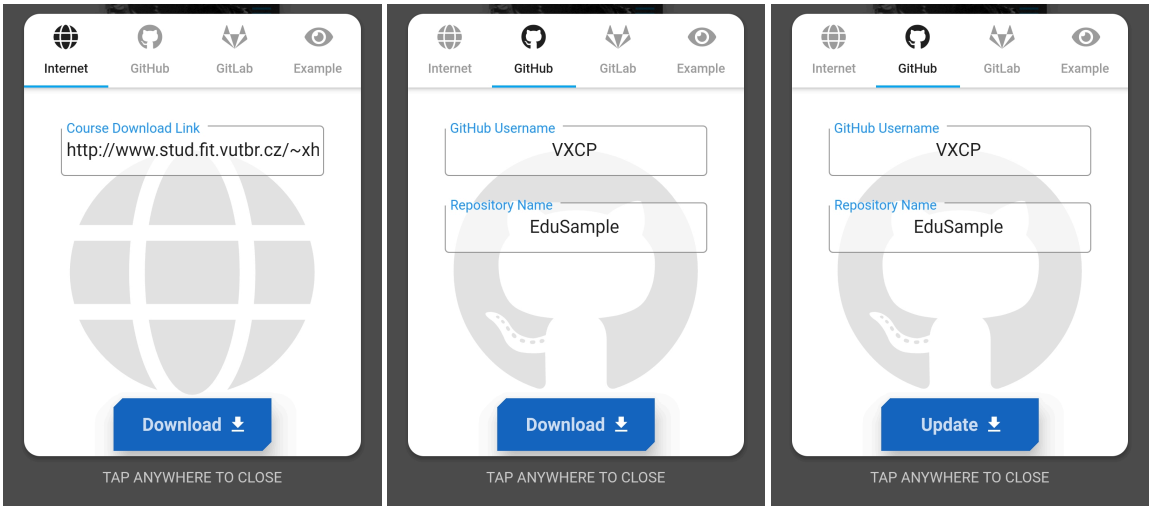
1. Základná lišta
2. Užívateľský vstup
3. Dynamické tlačidlo

Základná lišta a užívateľský vstup

Základná lišta obsahuje niekoľko tlačidiel, ktoré umožňujú definovať spôsob, akým bude užívateľ definovať internetovú lokalizáciu kurzu. Predvoleným vstupom je zadanie celej URL adresy kurzu.Taktiež boli pridané možnosti, kedy si užívateľ môže stiahnuť kurz uložený na určitých službách – konkrétne GitHub alebo GitLab zjednodušeným spôsobom. Namiesto toho, aby sa vyžadovalo vloženie celej URL adresy, tak stačí jednoducho vyplniť názov užívateľa (tvorcu kurzu) danej služby a názov vzdialeného repozitára, kde je daný kurz uložený. Na základe týchto informácií si aplikácia dokáže už automaticky vygenerovať URL adresu, z ktorej bude kurz sťahovať. Rôzne typy vstupov sú zobrazené na obrázku 6.2.

Dynamické tlačítko

Veľmi dôležitým prvkom tejto časti užívateľského rozhrania je dynamické tlačítko, ktorého obsah dynamicky prispôsobuje každej zmene aktuálneho vstupu. Konkrétne ide o to, že ak užívateľ zadá pomocou vstupu URL adresu kurzu, ktorý už je v aplikácii nainštalovaný, tak informuje užívateľa o tom, že dôjde len k aktualizácii kurzu, a nie k inštalácii nového



Obr. 6.2: Inštalácia kurzu umiestneného na ľubovoľnej internetovej lokácii (vľavo). Inštalácia kurzu umiestneného na serveroch služby GitHub (v strede). V prípade, že je už zadaný kurz nainštalovaný, tlačítko indikuje, že pôjde len o aktualizáciu (vpravo).

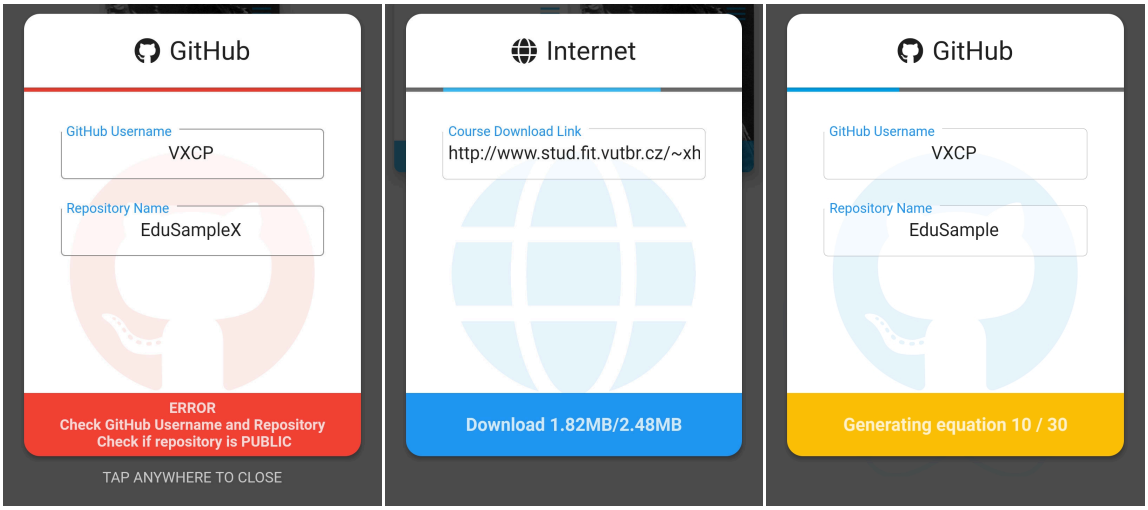
kurz viď obrázok 6.2. Ďalšou vlastnosťou je jeho dynamická zmena na základe stavu reťazca sprístupnenia kurzu, kedy sa po jeho stlačení spustí reťazec sprístupnenia a samotné tlačítko sa zmení na informačnú lištu, ktorá textovo a pomocou farieb zobrazuje aktuálne informácie o tomto stave. Konkrétne sa zobrazia jedná o tieto informácie:

1. Stav pripojenia na server, kde je kurz uložený
2. Aktuálny pokrok v sťahovaní kurzu
3. Indikuje, kedy prebieha extrakcia a dekompresia dát
4. Stav generovania matematických vzorcov
5. Chybové hlásenia

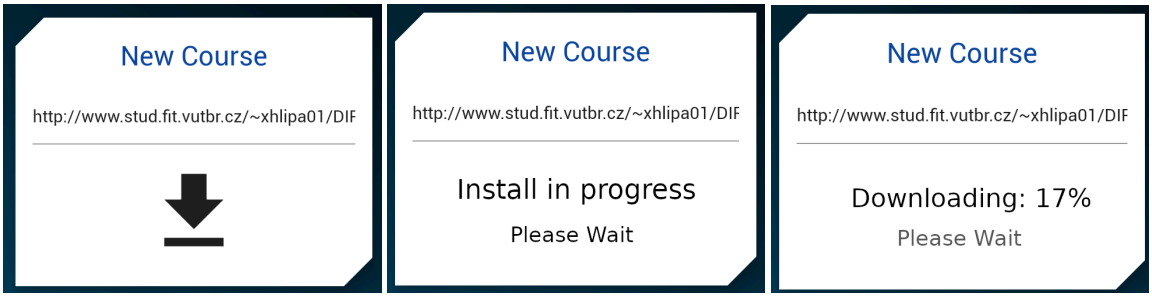
Celé rozloženie dialógu je doplnené o animovaný indikátor pokroku (angl. Progress bar), ktorý okrem toho, že naznačuje to, že aplikácia vykonáva nejakú akciu, zároveň slúži aj ako odozva užívateľovi, že nebol narušený chod aplikácie, čo užívateľovi dodáva istotu, že všetko funguje. Niektoré z týchto vlastností sú vyobrazené na obrázku 6.3.

6.1.3 Vývoj užívateľského rozhrania

Konečné užívateľské rozhranie, ktoré je vyobrazené v predchádzajúcej podsekcii, bolo zlepšované iteratívne v niekoľkých fázach. Na úplnom počiatku vytvárania tejto časti bolo rozloženie užívateľského rozhrania úplne jednoduché a pozostávalo iba zo vstupného textového poľa a tlačítka na stiahnutie. Chybou tohto návrhu bolo, že progres sťahovania bol zobrazovaný nedeterministicky, teda nebola priamo k dispozícii informácia napríklad o presnom percentuálnom postupe, ale iba informácia, že kurz sa aktuálne sťahuje a inštaluje, pričom užívateľ nemal ani predstavu, čo konkrétne aplikácia aktuálne vykonáva. Následne bol tento problém odstránený pridaním jednoduchého textu, ktorý užívateľa informoval o aktuálnom stave. Tento postupný vývoj dialógu je zobrazený na obrázku 6.4.



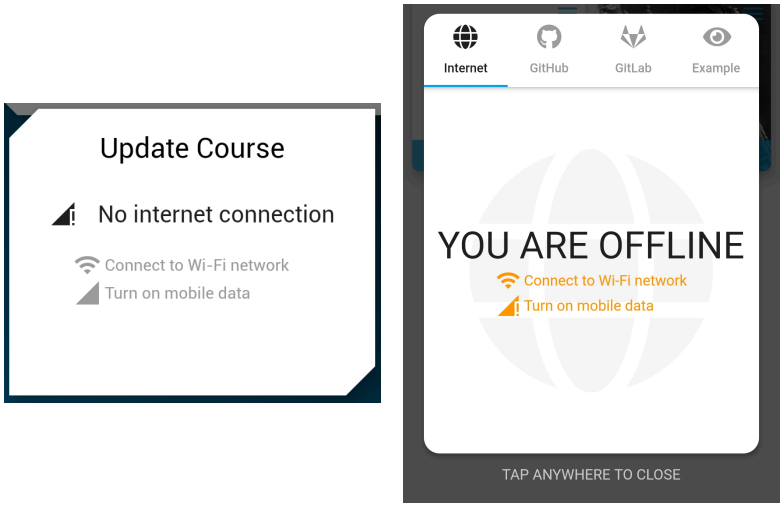
Obr. 6.3: Príklad indikácie chybového hlásenia (vľavo). Indikácia pokroku pri sťahovaní dát kurzu (v strede). Aktuálny stav generovania matematických vzorcov (vpravo).



Obr. 6.4: Prvá verzia dialógu (vľavo) Prvá verzia dialógu – nedostatočne informuje o pokroku inštalácie (v strede). Druhá verzia dialógu – informuje o stave inštalácie detailnejšie (vpravo).

Ďalším prvkom, ktorý sa ukázal postupným testovaním aplikácie ako nedomyslený, bola skutočnosť, že užívateľ častokrát zavrel dialóg ešte pred dokončením operácie, a to práve s pocitom, že sa inštalovanie dokončí na pozadí. Avšak toto správanie užívateľa narušilo hlavne fázu generovania rovníc, čím sa kurz nainštaloval nesprávne. To viedlo k vytvoreniu dialógu s už spomínaným špecifickým modálno-nemodálnym správaním. Následne bolo implementované automatické uzamknutie vstupného textového poľa, aby nebola URL adresa modifikovateľná počas inštalácie. Zároveň bola pridaná dynamická indikácia stavu pripojenia zariadenia do internetovej siete, kedy dialóg neumožnil sťahovanie kurzu, kým nebolo zaručené pripojenie na internet, pričom dialóg informoval užívateľa o nedostupnosti siete. Táto indikácia sa ukázala ako vhodná a preto bola zahrnutá aj do finálneho rozhrania zobrazeného na obrázku 6.5.

Spracovaním predchádzajúcich nedostatkov došlo k novému návrhu dizajnu tohto dialógu a zakomponovaniu animovaného indikátora pokroku. Na základe pozorovania plynulosti jeho animácie bolo nadobudnuté zistenie, že pri inštalácii kurzov malo práve extrahovanie dát a prehľadávanie súborov za účelom vyhľadávania rovníc vplyv na dynamiku užívateľského rozhrania a jeho odozvu. Po nadobudnutí tohto poznatku musel byť inštalačný reťazec upravený tak, aby boli tieto časti kódu vykonávané mimo hlavného vlákna



Obr. 6.5: Dynamická detekcia internetového spojenia – v staršej verzii dialógu (vľavo), bola zakomponovaná do finálnej verzie (vpravo).

aplikácie, tak ako je to popísané v sekcii 6.1.1. Posledným prvkom zmeny bola pridaná podpora sťahovania kurzov zo služieb Github a GitLab. Pri najbližšom testovaní však na základe tohto vylepšenia bola odhalená ďalšia vlastnosť, ktorú užívatelia považovali za zmätočnú. Išlo práve o to, že dialóg indikoval už nainštalovaný kurz rovnakým spôsobom, ako kurz nový, a to slovom „Download“, čo užívateľa zneistilo a prinútilo pred začatím akcie inštalácie kurzu si znovu overiť, či už daný kurz nemá nainštalovaný. Preto bolo pridané nové zlepšenie, ktoré každou zmenou vstupu kontroluje, či už daný kurz nie je nainštalovaný, a v prípade, že áno, tak tlačítko indikuje kurz slovom „Update“ a nie slovom „Download“ – viď obrázok 6.2.

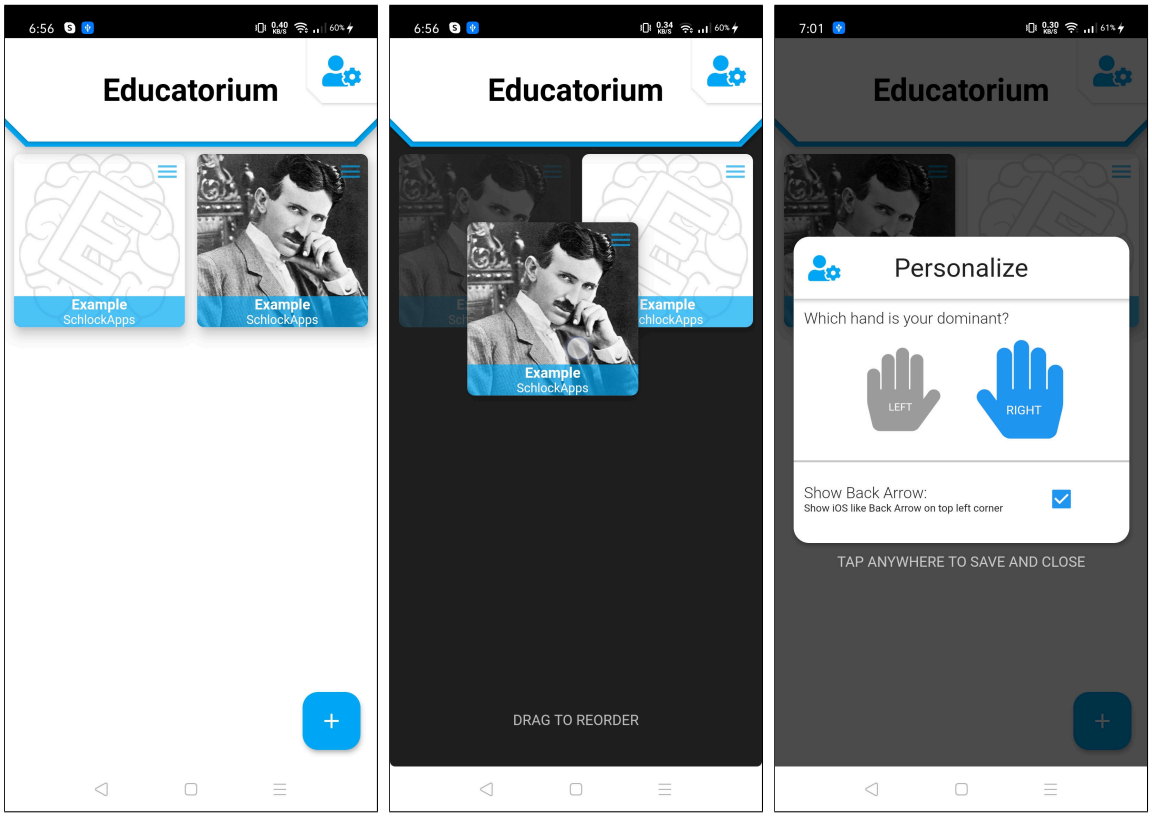
6.2 Hlavná obrazovka aplikácie

Hlavná obrazovka aplikácie vyobrazená na obrázku 6.6 je navrhnutá v duchu jednoduchosti a zaužívaných či užívateľom dobre známych prvkov užívateľského rozhrania z iných aplikácií. Štruktúra pozostáva z lišty, kde sa nachádza názov aplikácie s tlačítkom pre prístup k nastaveniam aplikácie, plávajúceho tlačítka akcie, ktoré slúži na zobrazenie dialógu pre inštaláciu opísaného v sekcii 6.1.2, a priestoru pre zobrazovanie zoznamu kurzov nainštalovaných na zariadení.

Každý kurz je reprezentovaný pomocou dlaždice v tvare štvorca, kde je na spodnej časti dlaždice vyobrazený názov kurzu (téma, ktorej sa venuje) a meno jeho autora. V pravom hornom rohu dlaždice sa nachádza tlačidlo, ktoré sprístupňuje menu pre spravovanie kurzu a pozostáva z týchto dvoch tlačidiel:

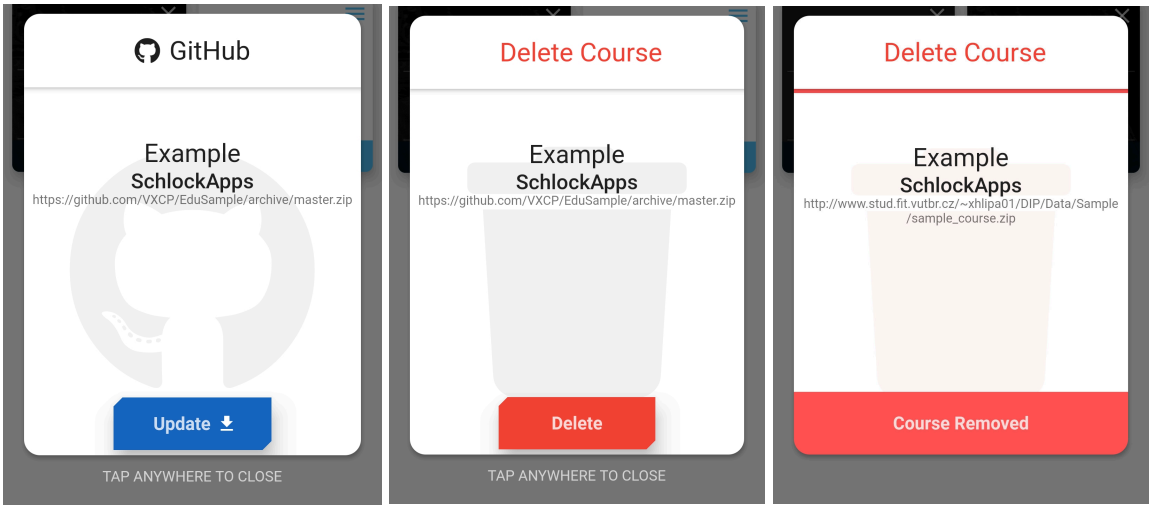
- Update – zobrazí dialóg na aktualizovanie kurzu.
- Delete – zobrazí dialóg na odstránenie kurzu.

Dialóg, ktorý slúži výhradne na aktualizáciu kurzu, funguje zhodne s dialógom opísaným v sekcii 6.1.2. S tým rozdielom, že už užívateľovi neumožňuje zadávanie internetovej lokácie pre tento kurz. Stlačením tlačidla „Update“ tohto dialógu sa spustí aktualizácia kurzu pomocou reťazca sprístupnenia kurzu. Pre odstránenie kurzu je vytvorený dialóg v podobnom



Obr. 6.6: Rozloženie hlavnej stránky aplikácie (vľavo). Jednotlivé kurzy je možno reorganizovať (v strede). Dialóg nastavení (vpravo)

štýle ako dialóg pre aktualizáciu, avšak jeho akciou je odstránenie daného kurzu z aplikácie. Obrázky zobrazujúce tieto dva dialógy je možné vidieť na obrázku 6.7.

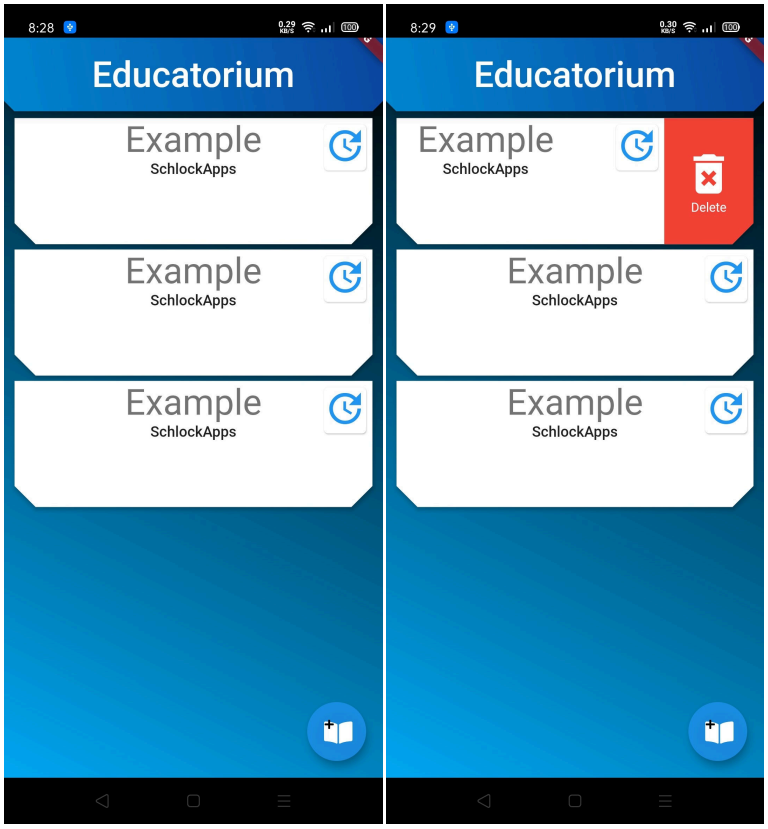


Obr. 6.7: Dialóg pre aktualizovanie kurzu (vľavo). Dialóg pre odstránenie kurzu (v strede). Dialóg oznamuje, že kurz bol úspešne odstránený (vpravo).

6.2.1 Vývoj užívateľského rozhrania

Počiatočná štruktúra užívateľského rozhrania bola (čo sa týka hlavného rozloženia prvkov na obrazovke) celkom podobná konečnému užívateľskému rozhraniu. Avšak grafická reprezentácia plávajúceho tlačidla pre pridávanie kurzov a užívateľskej skúsenosti s prácou s jednotlivými dlaždicami sa prvá verzia ukázala ako značne nevhodná. Prvotná verzia užívateľského rozhrania je vyobrazená na obrázku 6.8.

Prvým problémom bolo plávajúce tlačítko, ktoré každý z testovaných užívateľov prvotne stlačil, no s tým, že nie úplne porozumeli reprezentácii spojenia ikoniek (kniha a plus). Preto boli pôvodné ikonky zmenené iba na jednoduchý znak „+“, ktorý lepšie evokoval zámer, že užívateľ bude niečo nejakým spôsobom pridávať/sťahovať. Ďalšou pripomienkou užívateľov bolo vhodnejšie rozlíšenie farby samotného tlačítka od pozadia, ktoré spolu mierne splývali.



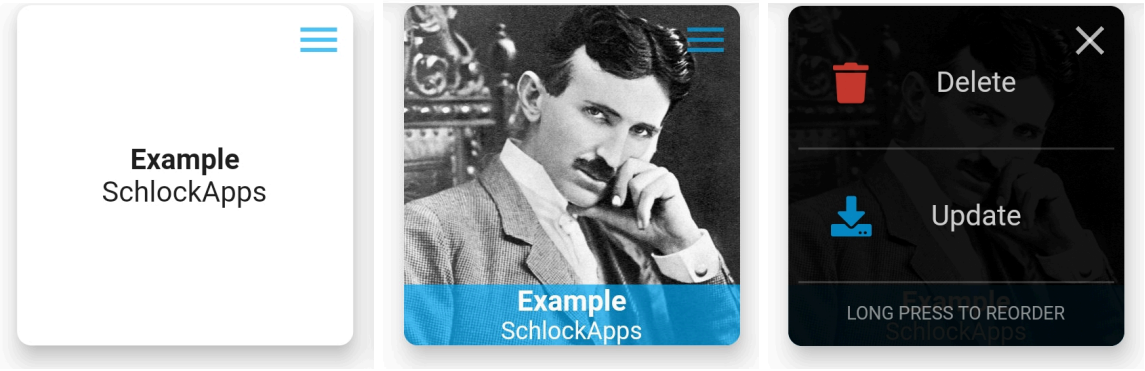
Obr. 6.8: Prvotné rozloženie hlavnej stránky aplikácie (vľavo). Odstránenie kurzu pomocou potiahnutia dlaždice do ľavej strany (vpravo).

Najväčším problémom však bola reprezentácia jednotlivých kurzov, ktoré boli pod sebou uložené v zozname ako je to vyobrazené na obrázku 6.8. V tomto prípade sa neosvedčil spôsob, akými sa jednotlivé kurzy spravovali. Aktualizácia pomocou tlačítka v pravom hornom rohu dlaždice sa ukázala ako neintuitívna (užívateľ síce vedel, že sa jedná o tlačítko, ale absolútne nevedel, že sa po jeho stlačení začne aktualizovať kurz). Ešte horším sa v praxi ukázala metóda odstránenia nainštalovaného kurzu spôsobom potiahnutia dlaždice smerom doľava, po ktorom sa odhalilo tlačidlo pre odstránenie kurzu (viď obrázok 6.8). Pri zadaní úlohy, kedy mal testovaný užívateľ odstrániť zo zoznamu daný kurz, sa stalo aj to, že

užívateľské rozhranie bolo úplne nepochopené a užívateľ na spôsob odstránenia neprišiel vôbec.

Zistenie týchto nedostatkov zabezpečilo kompletný redizajn užívateľského rozhrania dlaždíc pre jednotlivé kurzy. Výsledkom bola druhá verzia, ktorú je možné vidieť na obrázku 6.9. Táto verzia bola veľmi podobná finálnej verzii avšak jedným už z mála problémov bola nepripravenosť užívateľského rozhrania na ilustračné obrázky pre jednotlivé kurzy. Často sa stávalo, že čierna časť textu na tmavom pozadí splývala, a tak boli názov kurzu a meno autora nečitateľné. Preto bol pod text názvu a autora pridaný pás s modrým pozadím a text zmenený na kontrastnú bielu farbu.

V náväznosti na zmenu tvaru dlaždíc do štvorcovej podoby bol zmenený aj tvar plávajúceho akčného tlačítka pre inštalovanie nového kurzu z kruhového na štvorcový, a to z dôvodu, aby sémanticky ladil s tvarom dlaždíc, ktorými sú kurzy reprezentované.



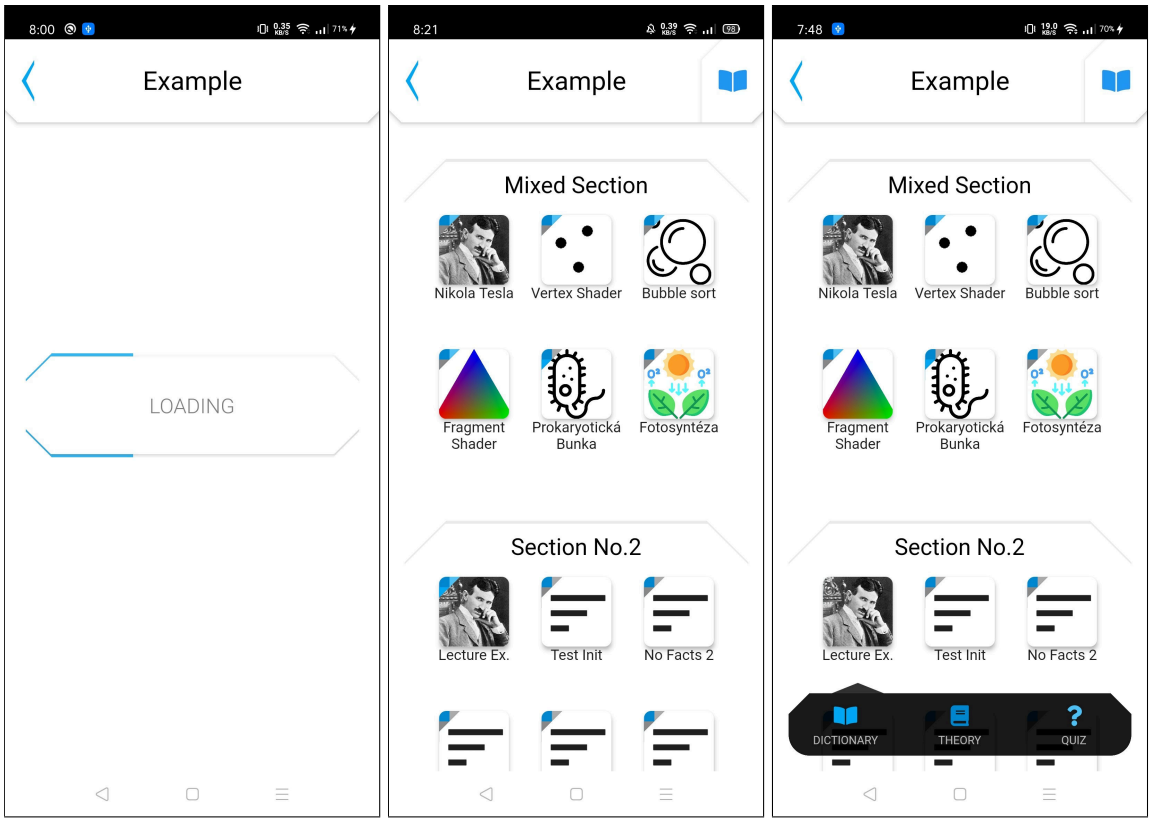
Obr. 6.9: Druhá verzia dlaždice – nebola pripravená na obrázok v pozadí (vľavo). Vylepšená verzia dlaždice – text je čitateľný aj v prípade obrázka (v strede). Menu spravovania kurzu (vpravo).

Užívateľovi je taktiež sprístupnená možnosť prispôsobenia užívateľského rozhrania aplikácie pomocou dialógu nastavení, ktorý je možné vidieť na obrázku 6.6. Tento dialóg poskytuje funkcie prispôsobenia užívateľského rozhrania na základe ľavorukosti resp. pravorukosti používateľov. Zároveň umožňuje pre zariadenia s operačným systémom Android zobrazenie šípky v hornom rohu disponujúcou schopnosťou vykonať krok späť, ktorá bola pridaná predovšetkým pre zariadenia s operačným systémom iOS. Finálny návrh, aký je znázornený na obrázku 6.6, je takmer totožný s pôvodným návrhom. Jedinou zmenou bola premena indikátora „TAP ANYWHERE TO CLOSE“ na „TAP ANYWHERE TO SAVE AND CLOSE“. Táto zmena bola vykonaná na základe pozorovania testovacích subjektov, kedy po uzatvorení tohto dialógu predmetní užívatelia opakovane kontrolovali, či boli zmeny v nastaveniach zariadením zaznamenané.

Na záver bola implementovaná funkcia reorganizácie dlaždíc kurzov v zozname (zmena poradia). Pre informovanie užívateľa o tejto možnosti bol pridaný text na spodnú časť dlaždice, ktorý informoval užívateľa, že túto funkciu môže vyvolať dlhým podržaním dlaždice viď obrázok 6.9. Táto funkcia sa ukázala ako vhodná, avšak spustenie tejto akcie nebolo nijako indikované, čo bolo pre užívateľa zmätočné. Preto sa spustením módu reorganizácie kurzov zmení farba pozadia hlavnej časti obrazovky a indikuje, že užívateľ môže dlaždice s kurzami reorganizovať. Túto vlastnosť užívateľského rozhrania je možné sledovať na obrázku 6.6 v strede.

6.3 Hlavná obrazovka kurzu

Definitívne rozloženie je zobrazené na obrázku 6.10 a pozostáva zo skrolovateľného widgetu a hlavnej lišty, ktorá je rozdelená do troch častí – v strede názov kurzu a dve tlačítka, z ktorý sa jedno nachádza v pravej časti a slúži na sprístupnenie slovníka daného kurzu popísaného v sekcii 6.4. Druhé tlačítko v tvare šípky, ktorého údelom je sprístupniť návrat užívateľa na hlavnú stránku aplikácie, sa nachádza v ľavej časti a bolo implementované najmä z dôvodu multiplatformnosti. Osobitosť tohto prístupu spočíva v tom, že oproti zariadeniam využívajúcim operačný systém Android, ktoré majú vždy k dispozícii systémové tlačítka dostupné z aplikácie, pomocou ktorých je možné vykonať krok späť, zariadenia operujúce na operačnom systéme iOS túto vymoženosť neposkytujú. Preto musela byť táto funkcionalita sprístupnená pomocou užívateľského rozhrania aplikácie, pretože v opačnom prípade by pre užívateľa s takýmto zariadením neexistovala cesta na návrat k hlavnej obrazovke aplikácie. Skrolovateľný widget v sebe zaobaľuje prístupové body k jednotlivým lekciám kurzu a vyobrazuje ich hierarchiu – teda ich rozdelenie do jednotlivých sekcií. Každý základný prvok kurzu (lekcia) je vyobrazený pomocou tlačidla štvorcového tvaru. Po stlačení tlačidla lekcie sa zobrazí rozcestník ktorý sprístupňuje jej jednotlivé súčasti – teoretickú časť, praktickú časť a slovník. Zobrazovanie a skrývanie rozcestníka je realizované pomocou zreťazenej animácie. Kvôli možnej zvýšenej vypočetnej náročnosti načítania veľkých kurzov bol pridaný indikátor načítania kurzu.



Obr. 6.10: Indikácia načítania potrebných dát (vľavo). Hlavné rozloženie obrazovky kurzu (v strede). Rozcestník sprístupňujúci jednotlivé súčasti lekcie (vpravo).

6.3.1 Štruktúra súboru JSON

Táto časť textu sa bude venovať spôsobu definícii hlavnej štruktúry kurzu a zároveň hlavnej obrazovky kurzu. Opis štruktúry bol pôvodne plánovaný v podobe formálnej definície pomocou gramatiky v podobe nonterminálov a terminálov, avšak po pokuse o tento spôsob zápisu bol zápis menej názorný ako to je v aktuálnej forme a to hlavne pri opise štruktúry kódu v kapitole 7. Preto som sa rozhodol o opis vo forme názornej ukážky a následnej definície jednotlivých kľúčov a im prislúchajúcich hodnôt, ktorý bude využívaný v celej práci. Každý kľúč a jeho hodnota bude definovaný vo forme „<key> (<data_type>):<definition>“, kde „<key>“ predstavuje názov kľúča, „<data_type>“ predstavuje datový typ hodnoty, ktorý môže kľúč nadobúdať a „<definition>“ predstavuje popis toho, čo kľúč a jeho hodnota predstavuje v kontexte aplikácie. V ďalšom texte sa rozlišujú okrem známych dátových typov (reťazec, objekt, bool, pole) aj tieto dátové typy:

- referencia – reťazec odkazujúci na súbor, ktorý nadobúda formu opísanú v kapitole 5.3.
- markdown – reťazec vo formáte markdown, ktorý umožňuje štruktúrovať text.
- číslo – číselná hodnota. V prípade definície číselného intervalu pomocou spojovníka „-“ sa bude jednať o množinu celých čísel v danom intervale a v prípade definície intervalu pomocou znakov „<“ a „>“ sa bude jednať o množinu reálnych čísel v danom intervale.

Text ktorý je možno štruktúrovať pomocou zápisu vo formáte markdown je implementovaný pomocou balíčka `flutter_markdown`[5]. Preto je forma zápisu zhodná so zápisom definovaným v dokumentácii tohto balíčka, avšak pre prispôsobenie balíčka na účel aplikácie boli vykonané tieto zmeny:

- Pridaná podpora vykresľovania matematických a chemických vzorcov pomocou formy zápisu podobnej ako LaTeX, tak ako ju definuje knižnica MathJax[6]. Zápis vzorca v rámci riadku (inline) začína a končí znakom „$“ a klasický zápis počína a končí reťazcom „$$“.
- Nie je podporovaná integrácia obrázkov do textu.
- Nie sú podporované internetové odkazy v texte.

Taktiež bola pre prispôsobenie potrieb aplikácie vykonaná zmena v zdrojovom súbore tohto balíčka. Hlavnú obrazovku aplikácie definuje súbor `Course.json`, ktorého štruktúra nadobúda následnú podobu:

```
{
    "name" :"Example Course",
    "authors": "Author Name",
    "thumbnail" :"@./CourseThumbnail.png",
    "enableInlineDictionary": false,
    "sections" :[
           "Section No.1",
           [
               "@./Lecture1.json",
               "@./Lecture2.json",
```

[5] https://pub.dev/packages/flutter_markdown/example
[6] https://docs.mathjax.org/en/latest/basic/mathematics.html

```
            "@./Lecture3.json",
        ],

        "Section No.2",
        [
            "@./Lecture4.json",
            "@./Lecture5.json",
        ]
        ...
    ]
}
```

Kde jednotlivým kľúčom prislúchajú tieto hodnoty:

- `name` (*reťazec*) – názov kurzu.
- `authors` (*reťazec*) – meno autora kurzu (reťazec), pričom je možné zadať viacerých autorov, ak je hodnotou pole reťazcov.
- `enableInlineDictionary` (*bool*) – aktivuje resp. deaktivuje rýchly prístup slovníka z textu, ktorý je opísaný v kapitole 7.1.2.
- `thumbnail` (*referencia*) – ilustračný obrázok vyobrazený na dlaždici kurzu na hlavnej stránke aplikácie zo sekcie 6.2.
- `sections` (*pole*) – definuje štruktúru kurzu (hierarchiu medzi sekciami a lekciami).

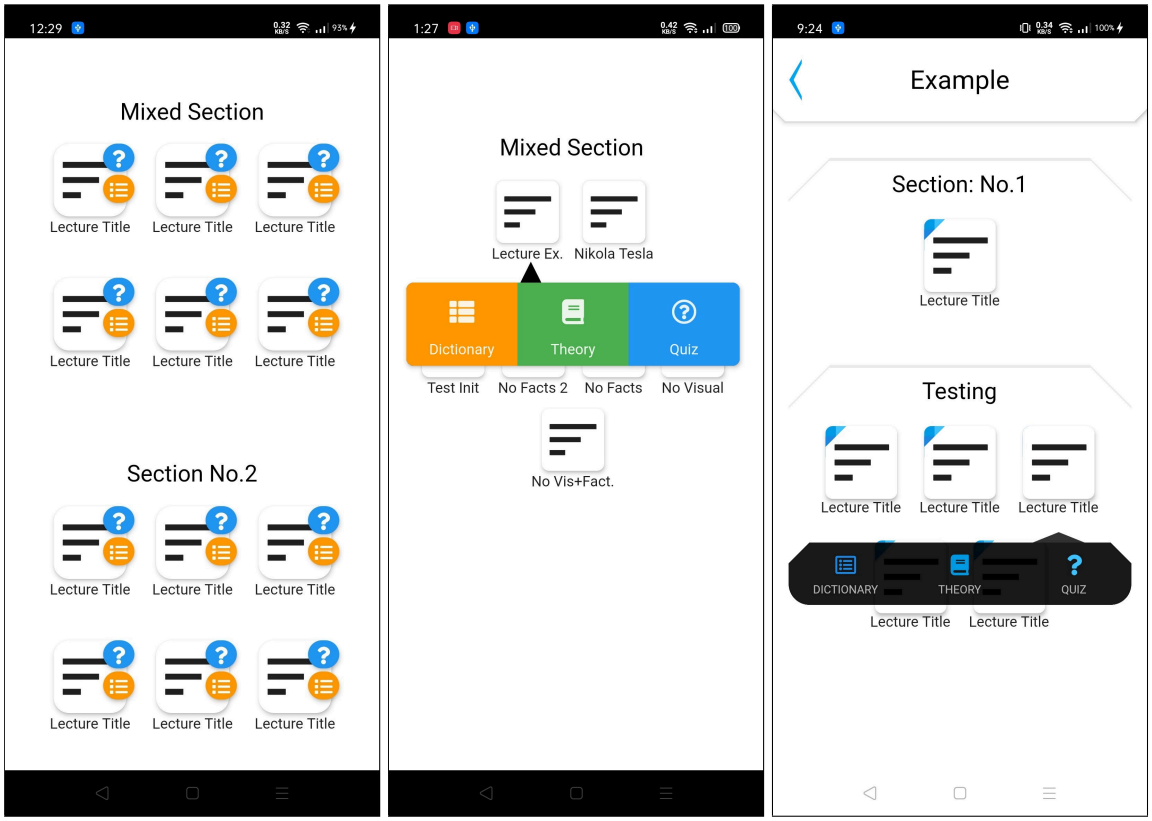
Kľúč `sections` definuje logickú štruktúru kurzu a rozdeľuje jednotlivé lekcie do sekcií. Každá lekcia je daná referenciou na súbor, v ktorom sa nachádza jej definícia. Štruktúra hodnoty kľúča `sections` je nasledovná. Hodnotou je vždy pole. Ak je prvkom poľa reťazec, tak je definovaná sekcia s názvom reťazca. Následne všetky referencie uložené v poli pod daným reťazcom sú brané ako odkazy na súbory s definíciou lekcie. Pokiaľ má tvorca záujem vytvoriť kurz bez rozdelenia do sekcií, tak hodnotou kľúča `sections` bude iba pole s referenciami na súbory lekcií, ktoré majú byť súčasťou kurzu.

Predchádzajúci príklad definuje kurz, ktorý bude zložený z 2 sekcií. Prvá sekcia bude niesť názov „Section No.1“ a bude obsahovať 3 lekcie definované v súboroch `Lecture1.json`, `Lecture2.json`, `Lecture3.json`. Druhá sekcia bude niesť názov „Section No.2“ a bude v sebe zahŕňať 2 lekcie definované v súboroch `Lecture4.json` `Lecture5.json`. Spôsob referencovania súborov je bližšie opísaný v kapitole 5.3.

6.3.2 Vývoj užívateľského rozhrania

Každá lekcia bola na začiatku reprezentovaná tlačidlom štvorcového tvaru, pod ktorým sa nachádzal názov lekcie. V tom čase však ešte nebola implementovaná podpora aplikácie pre praktickú časť kurzu a slovník, ale iba pre jej teoretickú časť. Z dôvodu doplnenia týchto častí do aplikácie bolo teda nutné vytvoriť spôsob, ktorý by umožnil prístup k jednotlivým súčastiam lekcie (teoretická časť, praktická časť a slovník) jednotlivo. Druhotný návrh počítajúci aj s touto funkcionalitou pozostával z princípu doplnenia pôvodného štvorcového tlačidla o dve menšie tlačidla uložených po jeho pravej strane (jedno pre slovník, druhé pre praktickú časť) viď obrázok 6.11 (vľavo). Tento návrh sa ukázal ako zle využiteľný, pretože tlačítka boli ťažko zasiahnuteľné a to hlavne na zariadeniach s menšími obrazovkami. To

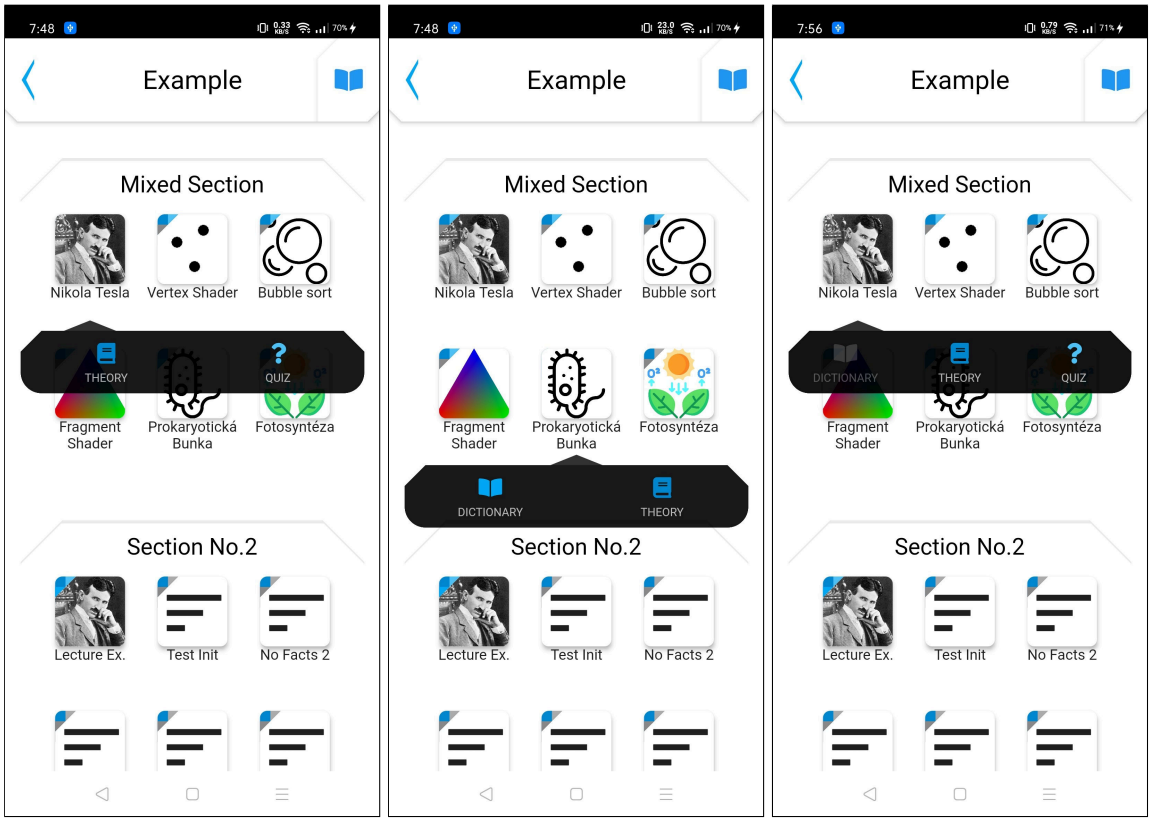
spôsobovalo nepresnosti a užívateľovi sa celkom často stávalo, že nepresným stlačením tlačítka pre otvorenie praktickej časti namiesto nej otvoril časť teoretickú. Veľkosť tlačidiel nebolo možné kvôli vyhradenému priestoru zväčšiť a preto bol tento návrh pozmenený do formy, ktorá využívala rozcestník v podobe vyskakovacieho pop-up menu, ktoré sa zobrazilo-/skrylo po stlačení tlačítka lekcie. Rozcestník pozostával z troch tlačidiel, ktoré umožňovali prístup k spomenutým prvkom danej lekcie. Pri implementácii bolo potrebné zisťovať aktuálnu pozíciu tlačítka danej lekcie, pretože pop-up menu bolo potrebné vykresliť priamo pri tlačidle lekcie s ukazovateľom, aby užívateľ pochopil, že sa menu vzťahuje k danej lekcii viď obrázok 6.11(v strede a vpravo).



Obr. 6.11: Tlačidlo lekcie s menšími tlačidlami po pravej strane (vľavo). Rozcestník s poziciovaním vzhľadom k hlavnému tlačidlu lekcie (v strede). Vylepšený návrh o animáciu a zmena farieb rozcestníka pre zachovanie farebnej konzistencie aplikácie (vpravo).

Tento návrh bol úspešnejší ako pôvodný, no stále nebol ideálny. Pri fáze testovania bola odhalená chyba, kedy po otvorení rozcestníka lekcie užívateľ nevybral žiadnu akciu z menu, ale pokračoval v skrolovaní kurzom ďalej. Samotný rozcestník však zostal rozbalený na pôvodnom mieste obrazovky, zatiaľ čo tlačidlo lekcie svoju pozíciu zmenilo. Tento fakt bol pre užívateľa veľmi zmätočný a preto v ďalšej fáze bola pridaná vlastnosť nasledovania tohto rozbaleného menu k tlačítku, ku ktorému prislúcha. Nakoniec sa však ukázalo, že implementácia tejto zmeny bola úplne zbytočnou stratou energie, pretože takmer vždy, keď užívateľ otvoril pomocou tlačítka rozcestník danej lekcie a následne začal skrolovať, tak vo veľkej väčšine prípadov sa k danej sekcii spätne nevrátil. Preto muselo byť správanie tlačidla a jemu prislúchajúcemu menu znovu zmenené. Riešením bolo vždy uzavretie aktuálne otvoreného menu pri každom započatí skrolovania kurzu. Taktiež bola pridaná

časť programu, ktorá zabezpečovala to, aby neboli v jednom čase otvorené rozcestníky pre dve rôzne lekcie. To znamená, že ak je otvorený rozcestník lekcie A a užívateľ sa rozhodne otvoriť rozcestník lekcie B, tak rozcestník lekcie A zmizne. V opačnom prípade nastávali stavy, kedy sa tieto rozcestníky prekrývali a užívateľské rozhranie bolo veľmi zle ovládateľné a neprehľadné. Ďalšou zmenou bolo farebné prispôsobenie rozcestníka z pôvodnej oranžovej, zelenej a modrej farby na odtiene modrej, z dôvodu zachovania farebnej konzistencia celej aplikácie. Posledným, avšak veľmi dôležitým zistením pri procese testovania bol fakt, že užívateľské rozhranie rozcestníka bolo nekonzistentné. Rozcestník bol implementovaný s úmyslom vylepšenia, tak aby zobrazoval vždy iba dostupné súčasti lekcie. Teda ak bola dostupná teoretická časť a praktická nie, tak rozcestník zobrazil iba časť teoretickú. To sa však ukázalo v praxi ako nevhodné, pretože testovací subjekt mal tendenciu otvárať jednotlivé časti z pamäti. Teda vždy keď chcel otvoriť teoretickú časť lekcie, tak vedel, že sa tlačítko nachádza v strede. V prípade, že boli tlačítka iba dve, tak sa stalo, že namiesto teoretickej časti otvoril napríklad slovník. Tento problém je možno vidieť na obrázku 6.12 (vľavo a v strede). Preto bola vykonaná zmena užívateľského rozhrania, kedy sú v rozcestníku zobrazené všetky časti lekcie. Avšak tie, ktoré sú nedostupné, sú indikované sivou farbou a zároveň nereagujú na užívateľské gestá. Túto zmenu je možné vidieť a porovnať na obrázku 6.12 (vpravo).



Obr. 6.12: Rozcestník sprístupňujúci slovník a lekciu (vľavo). Rozcestník sprístupňujúci lekciu a kvíz (v strede). Vylepšený návrh rozcestníka sprístupňujúceho lekciu a kvíz (vpravo).

Pre správne vykresľovanie museli byť rozcestníky implementované pomocou inštancií widgetu `OverlayEntry`, čo umožňovalo, aby boli vždy vykreslené nad všetky ostatné prvky užívateľského rozhrania.

Užívateľské rozhranie vo svojom pôvodnom návrhu neobsahovalo hornú lištu. Tá bola pridaná až v neskoršej fáze, a to z dôvodu, že musela byť pridaná podpora slovníka a spätnej šípky. V reakcii na tieto zmeny bol následne pridaný názov, aby sa vyplnil ničím nevyplnený priestor užívateľského rozhrania.

Čo sa týka skrolovacej časti v ktorej sa nachádzajú lekcie, tak v prvom návrhu neboli vôbec hierarchizované, no postupne sa ukázalo, že pri obšírnejších kurzoch je tento spôsob neprehľadný a neumožňuje dostatočne štrukturalizovať kurzy. Preto bola logická štruktúra kurzu opísaná v kapitole 5.2 rozšírená o sekcie.

V samotnom závere bol pridaný dekoračný prvok do ľavého horného rohu tlačidla každej lekcie, ktorý súvisí s animáciou otvárania a zatvárania rozcestníka. Zároveň je v ňom skrytá indikácia dostupnosti jednotlivých súčastí lekcie (kde každý trojuholník rôzneho odtieňa prestavuje jednu z nich). Ak je daná časť dostupná, tak je farba prislúchajúceho trojuholníka v odtieni farby modrej, inak v odtieni šedej farby. Aj keď táto indikácia nebola testovacími subjektmi vo väčšej miere spozorovaná, tak bola na svojom mieste ponechaná aspoň za účelom plnenia svojej dekoračnej funkcie.

6.4 Slovník

Na základe analýzy existujúcich riešení bol pridaný do štruktúry kurzu slovník. Hlavnou inšpiráciou pre zakomponovanie tejto pomôcky boli aplikácia Complete Biology, ktorá takýto slovník ponúka z dôvodu prehľadu dôležitých pojmov. Obsahuje zoznam pojmov, ktoré sú kľúčové pre prezentovanú problematiku. Slovník implementovaný v tejto práci nadobúda jednoduchú štruktúru, ktorá sa skladá z dvojíc pojmov (slovo alebo slovné spojenie) a ich definícií. Zároveň je možné jednej definícii priradiť viac ako jeden pojem, čím sa definujú synonymá.

Rozhranie slovníka je prístupné z hlavnej obrazovky kurzu opísaného v sekcii 6.3. Užívateľské rozhranie je možné vidieť na obrázku 6.13. Samotné rozloženie pozostáva zo zoznamu dôležitých pojmov zoradených v abecednom poradí, ktorý po stlačení prvku zoznamu predstavujúceho daný pojem zobrazí jeho definíciu. Definícia pojmu pokrýva dominantnú časť šírky okna slovníka, pričom okrem samotnej definície pojmu sa pri dolnom okraji hlavnej časti rozhrania zobrazujú synonymá zvoleného slova. Zvyšnú časť šírky doplňuje rýchly prístup k ostatným pojmom obsiahnutým v slovníku. Posledným prvokom je hlavná lišta, ktorá obsahuje nadpis, ktorý informuje užívateľa o tom, že pracuje s oknom slovníka a taktiež umožňuje návrat zo stavu v ktorom je zobrazená definícia pojmu späť na zoznam pojmov.

6.4.1 Štruktúra súboru JSON

Slovník definuje tvorca kurzu v súbore `Dictionary.json`, ktorého umiestnenie v štruktúre dát kurzu je opísané v kapitole 5.3. Formát tohto súboru je nasledovný:

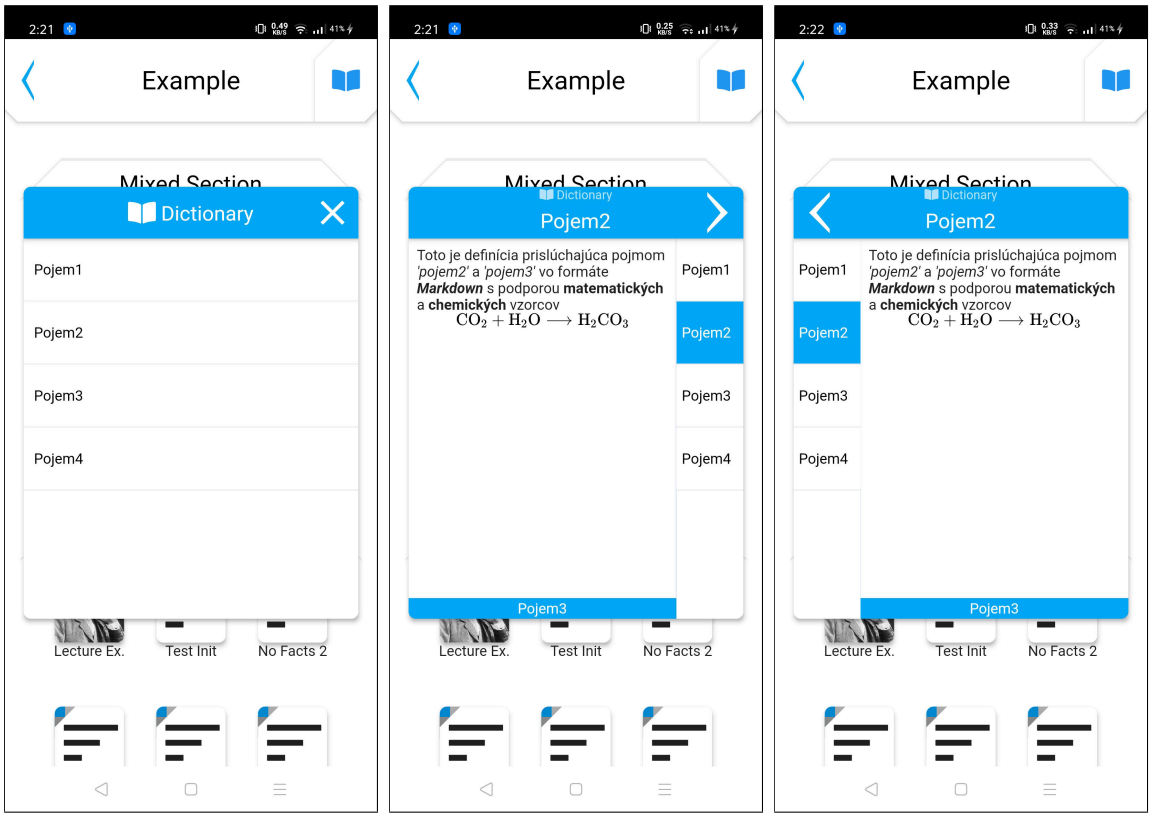
```
1 [
2     {
3         "words": ["Pojem1"],
4         "definition":"Toto je definicia prisluchajuca pojmu 'pojem1'"
5     },
6     {
7         "words": ["Pojem2", "Pojem3" ],
```

```
        "definition":"Toto je definícia pojmov 'pojem2' a 'pojem3' vo
            formáte Markdown s podporou vzorcov $$\\ce{CO2 + H2O -> H2CO3}$$
            "
    },
    ...
]
```

Súbor pozostáva z poľa obsahujúceho jednotlivé objekty definujúce dvojice pojem – definícia. Každý objekt pozostáva z dvoch atribútov.

- `words` (*pole reťazcov*) – obsahuje všetky pojmy prislúchajúce danej definícii.
- `definition` (*markdown*) – definícia prislúchajúca prvkom poľa `words`.

Na základe pojmov uložených v poli `words` sú tvorené aj vzťahy synoným. Návrh počíta s tým, že ak majú dve slová rovnakú definíciu, tak sa jedná o synonymá. V uvedenom príklade bude platiť, že slová „pojem2“ a „pojem3“ budú brané ako synonymá a „pojem1“ nemá žiadne synonymum.



Obr. 6.13: Zoznam pojmov (vľavo). Zobrazenie definície pojmu (v strede). Užívateľské rozhranie prispôsobné pre ľavorukých užívateľov (vpravo).

6.4.2 Vývoj užívateľského rozhrania

Počiatočný návrh bol takmer totožný s konečným návrhom. Po rozkliknutí pojmu sa však celý zoznam skryl a na celú šírku bola vyobrazená definícia daného pojmu. Voči tomuto

rozhraniu testovacie subjekty v podstate nemali žiadne výhrady, no aj tak som inicioval zmenu návrhu a rozšíril ho o spomenutý rýchly prístup pojmov. Pri ďalšom otestovaní na užívateľoch sa toto vylepšenie overilo s tým, že zrýchľovalo prácu so slovníkom a bolo teda zakomponované do konečného návrhu. Pridaním tejto vymoženosti však bolo užívateľské rozhranie taktiež prispôsobené aj pre ľavorukých užívateľov, hlavne pre ich lepší komfort práce s aplikáciou, kedy sa časť rýchleho prístupu prispôsobí ľavákovi tak, že sa jeho pozícia zmení na ľavú stranu. Prispôsobenie užívateľského rozhrania pre užívateľov s dominanciou ľavej ruky je zobrazené na obrázku 6.13.

Prvotným nápadom tiež bolo, že zo slov definovaných v tomto slovníku sa pre každú lekciu nájdu z jej obsahu slová, ktoré v sebe obsahuje. Teda na základe obsahu lekcie sa vytvorí tzv. lokálny slovník, ktorý obsahuje iba pojmy obsiahnuté v danej lekcii. Prístup k tomuto slovníku je zabezpečený pomocou rozcestníka lekcie spomenutom v sekcii 6.3. Pre tieto účely bolo poskytnuté rovnaké užívateľské rozhranie ako pre celý slovník kurzu, avšak v jeho zozname sa vyskytovali iba pojmy, ktoré sa nachádzali v obsahu tejto lekcie. Vzhľadom na to, že sa na hornej lište tohto rozhrania nachádza titulok „Dictionary“, tak sa užívateľom javilo, že sa jedná o jednu a tú istú vec a nechápali, prečo jeden slovník v jednom prípade obsahuje 1 pojem a v inom zase 3 pojmy. Preto bol výhradne pre lokálny slovník lekcie zmenený titulok na názov lekcie doprevádzaný ikonou reprezentujúcou slovník. Ukázalo sa, že tento na prvý pohľad bezvýznamný detail zlepšil pochopenie princípu lokálnych slovníkov.

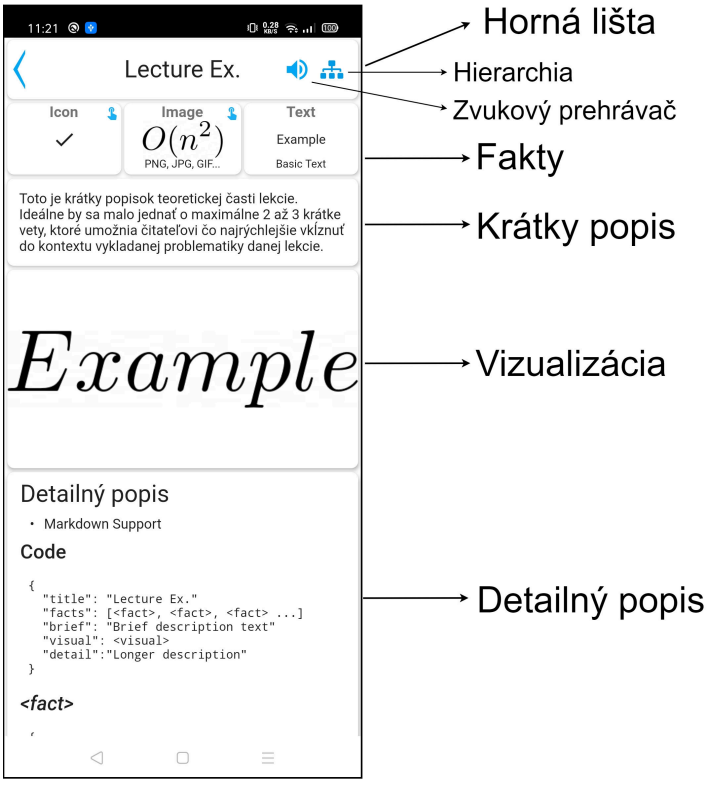
Kapitola 7

Lekcia

Táto kapitola sa bude bližšie venovať opisu užívateľského rozhrania najdôležitejšej základnej jednotky celej aplikácie, a to lekcie. Samotná lekcia pozostáva z dvoch častí – teoretickej a praktickej. Každá z uvedených častí bude detailnejšie analyzovaná v nasledujúcich sekciách tejto kapitoly.

7.1 Teoretická časť

Výsledné užívateľské rozhranie pre teoretickú časť lekcie je ukázané na obrázku 7.1. Predmetné rozhranie pozostáva z následujúcich súčastí: Krátky popis, Vizualizácia, Detailný popis, Fakty, Horná lišta, Hierarchia a Zvukový prehrávač.



Obr. 7.1: Užívateľské rozhranie teoretickej časti lekcie

Horná lišta

Horná lišta obsahuje názov lekcie a dvojicu tlačidiel pre zobrazenie hierarchie a zvukový prehrávač. Okrem týchto elementov poskytuje priestor pre tlačítko spätnej navigácie, odkazujúce na hlavnú obrazovku kurzu.

Fakty

Každý fakt pozostáva z nadpisu, obsahu, sekundárneho textu a detailného popisu faktu. Jedná sa o krátke jedno až dvojslovné prvky slúžiace pre veľmi rýchly prehľad najdôležitejších informácií, ktoré po dobu držania prstu na príslušnom fakte zobrazia jeho detailný popis. Demonštratívne možno uviesť nasledujúce prípady užitia: Ak sa kurz venuje problematike mikroorganizmov a lekcia prezentuje opis prokaryotickej bunky, kde za zásadnú informáciu môžeme považovať veľkosť takejto bunky, možno predať takúto informáciu vo forme faktu, kde názvom bude „Veľkosť“, obsahom „1-2μm“ a detailom, ktorý po kliknutí zobrazí tabuľku porovnania veľkostí jednotlivých typov mikroorganizmov (viď obr. 7.2 – vpravo). Alternatívou v odbore informačných technológii by mohla byť lekcia, ktorá sa venuje triediacemu algoritmu Quick Sort. V danom prípade za zásadnú informáciu môžeme považovať zložitosť tohto algoritmu, preto možno takúto informáciu predať užívateľovi ako fakt s nadpisom „Zložitosť“, obsahom „$O(nlogn)$“, podobsahom „Priemerná“ a detailom, ktorý sa zobrazí po kliknutí, by mohol byť text opisujúci základné typy časových zložitostí algoritmov alebo definícia časovej zložitosti algoritmu.

Krátky popis

Jedná sa o čo možno najkratší popis, ktorý uvádza čitateľa do problematiky alebo vyzdvihuje najdôležitejšie informácie z danej lekcie. Ideálne by sa malo jednať o maximálne 2 až 3 krátke vety, ktoré umožnia čitateľovi čo najrýchlejšie vkĺznuť do kontextu vykladanej problematiky danej lekcie.

Vizualizácia

Obrázok, ktorý ilustruje tému preberanú v danej lekcii alebo sa nejakým spôsobom viaže na výklad. Aplikácia podporuje vizualizáciu v podobe statických obrázkov (png, jpg, webp) alebo aj v podobe animácií (GIF), ktoré majú oveľa väčší vyjadrovací potenciál a zároveň sú vhodnejšie pri výklade niektorých tematických okruhov (napríklad triediace algoritmy).

Detailný popis

Mal by sa nejakým spôsobom viazať na vizualizáciu a obsahovať hlavný výkladový text k časti kurzu, ktorej sa venuje daná lekcia. Aby bol tento text čo najviac čitateľný, postupným rolovaním stránky dochádza k postupnému zakrývaniu hlavnej lišty a krátkeho popisu, čím sa vytvorí väčší priestor pre text výkladu (viď obr. 7.2 – v strede a vľavo)

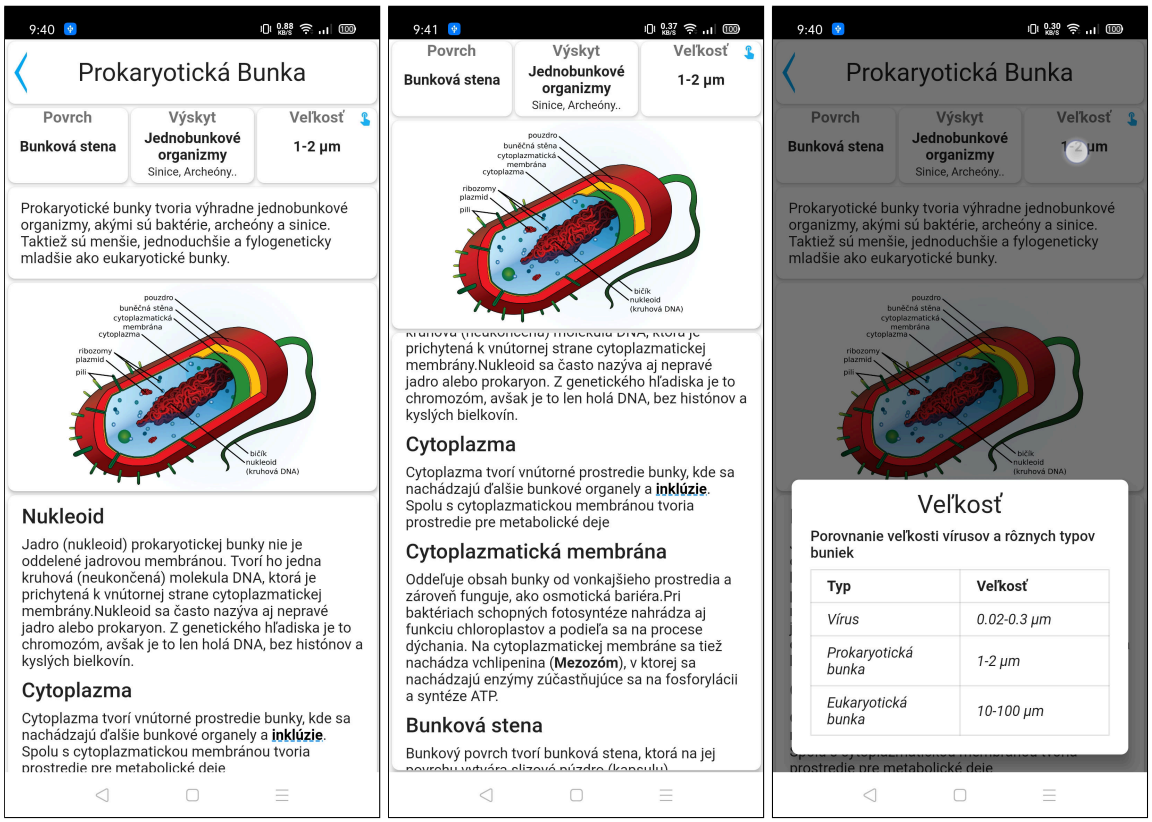
Hierarchia

Tento prvok slúži pre rýchle určenie kontextu lekcie v rámci výkladovej štruktúry kurzu. Príkladom môže byť nasledujúci prípad: Kurz sa venuje výkladu informácií o jednotlivých súčastiach zreťazeného spracovania OpenGL. Majme lekciu, ktorá obsahuje výklad o niektorej z týchto súčastí (napríklad Fragment Shader). Potom hierarchiou môže byť nákres

zoobrazujúci schému OpenGL pipeline, na ktorom je zvýraznená oblasť znázorňujúca „Fragment Shader“. To dá užívateľovi lepšie povedomie a prehľad o tom, ktorej súčasti sa aktuálne venuje a v akej relácii je táto súčasť so zvyšným obsahom kurzu (napríklad Vertex Shader).

Zvukový prehrávač

Prehrávač zvukov umožňuje integráciu zvukových súborov do lekcie. Bol implementovaný hlavne z dôvodu podpory výuky jazykov, kde je veľmi dôležitým prvkom výslovnosť či počúvanie nahrávok hovoreného slova. Okrem toho môže byť využitý aj za iným účelom, napríklad k výuke základov hudobnej výchovy – výuka rôznych tónov, alebo histórie hudby, kde by zvukovou nahrávkou mohla byť ukážka rôznych hudobných štýlov, hudby z rôznych období a iné.



Obr. 7.2: Rozloženie teoretickej časti lekcie (vľavo) s rolovaním textu mení veľkosť hlavného detailu pre lepšiu čitateľnosť textu (v strede). Zobrazenie detailu faktu po pordžaní prstom (vpravo).

7.1.1 Štruktúra súboru JSON

Táto časť textu sa bude venovať štruktúre súboru definujúceho teoretické časti lekcie. Platí, že každá lekcia je definovaná v osobitnom súbore. Tento súbor je zakomponovaný do štruktúry kurzu pomocou referencie, ktorá sa nachádza v poli `sections` v súbore `Course.json`, ktorého štruktúra je definovaná v sekcii 6.3.1. Štruktúra súboru definujúceho teoretickú časť lekcie má takúto podobu:

```
{
    "title": "Lecture Title",
    "thumbnail": "@./LectureThumbnail.png",
    "sounds": [
        {
            "title": "Audio 1",
            "credits": "Some Author or License 1",
            "source": "@~/Sounds/Sound1.mp3"
        },
        ...
    ],
    "hierarchy": {
            "source": "@~/Assets/Hierarchy.png",
            "position": "0.12:0.82"
    },
    "facts": [
        {
            "title" :"Fakt 1",
            "content": "Obsah faktu 1",
            "subcontent": "Podobsah faktu 1",
            "detail" :"Popis zobrazený po podržaní prstom"
        }
        ...
     ],
    "brief": "Toto je krátky popis, ktorý uvádza čitateľa do problematiky",
    "visual": {
        "source": "@~/Assets/Example.png"
    },
    "detail": [
        "# Detailny popis\n",
        "Podporuje notáciu vo formáte Markdown",
        ...
    ],
    "quiz": ["@~/QuestionSet.json", ...]
}
```

Kde jednotlivé kľúče znamenajú následovné:

- `title` (*reťazec*) – predstavuje názov lekcie vyobrazený na ovládacej lište lekcie a zároveň na hlavnej stránke kurzu 6.3.
- `thumbnail` (*referencia*) – predstavuje odkaz na náhľadový obrázok danej lekcie z hlavnej stránky kurzu.
- `hierarchy` (*objekt*) – predstavuje objekt definujúci Hierarchiu, kde kľúče objektu reprezentuje následovné:
  - `source` (*referencia*) – predstavuje odkaz na obrázok reprezentujúci hierarchiu.
  - `position` (*reťazec*) – určuje pozíciu terčíka v obrázku (viď obr. 7.5 – vľavo). Definícia je v tvare „<x>:<y>“, kde <x> určuje normalizovanú horizontálnu

súradnicu vzhľadom k šírke zdrojového obrázku a <y> určuje normalizovanú vertikálnu súradnicu vzhľadom k výške zdrojového obrázku. Normalizované súradnice nadobúdajú hodnoty v intervale <0,1>.

- `sounds` (*pole objektov*) – pole objektov predstavujúce zvukové prílohy lekcie. Každý objekt tohto poľa ma takúto štruktúr:
  - `title` (*reťazec*) – predstavuje názov zvukovej stopy.
  - `credits` (*reťazec*) – predstavuje meno autora zvukového súboru alebo licenciu.
  - `source` (*referencia*) – odkaz na zdrojový súbor so zvukovým obsahom.
- `facts` (*pole objektov*) – predstavuje pole objektov definujúcich prvok Fakt, kde štruktúra jedného objektu poľa je následnovná:
  - `title` (*reťazec*) – predstavuje nadpis faktu.
  - `content` (*reťazec alebo referencia*) – predstavuje hlavný obsah faktu a môže sa jednať o text, matematický vzorec alebo referenciu na statický alebo animovaný obrázkový súbor.
  - `subcontent` (*referencia*) – predstavuje sekundárny (dolný) text dlaždice .
  - `detail` (*markdown*) – predstavuje detailný popis faktu, ktorý sa zobrazí po podržaní prstom (viď obr. 7.2 – vpravo).
- `brief` (*markdown*) – predstavuje krátky popis lekcie.
- `visual` (*objekt*) – predstavuje objekt, ktorého hodnota kľúča `source` je referenciou na súbor obsahujúci obrázok predstavujúci hlavný vizualizačný prvok lekcie.
- `detail` (*markdown*) – predstavuje hlavný text lekcie. Text formátovaný podobne ako krátky popis pomocou zápisu Markdown. Nadobúda hodnotu reťazca alebo poľa reťazcov. Ako pole reťazcov je definovaný z dôvodu prehľadnejšieho zápisu pre tvorcu kurzu. Každý prvok poľa nepredstavuje nový riadok textu, ale všetky prvky sú konkatenované. Preto nový riadok výukového textu musí byť explicitne definovaný riadiacou sekvenciou „\n“.
- `quiz` (*pole referencií*) – predstavuje pole referencií na súbory obsahujúce sady otázok pre účel praktického precvičovania lekcie.

7.1.2 Vývoj užívateľského rozhrania

Táto sekcia sa bude venovať postupnému vývoju užívateľského rozhrania aplikácie zobrazujúcu teoretickú časť lekcie. Samotný vývoj užívateľského rozhrania bol vykonávaný v iteračných krokoch, ktoré sú detailne popísané v nasledujúcom texte.

Prvá iterácia

Prvotný návrh aplikácie pozostával zo štyroch v tom čase statických oblastí s pevne predpísanými parametrami veľkostí, ktoré prináležali prvkom: Horná lišta, Krátky popis, Vizualizácia a Detailný popis. Po vytvorení prototypu na základe prvotného návrhu boli vytvorené jednoduché lekcie pokrývajúce znalosti o jednoduchých triediacich algoritmoch akými sú

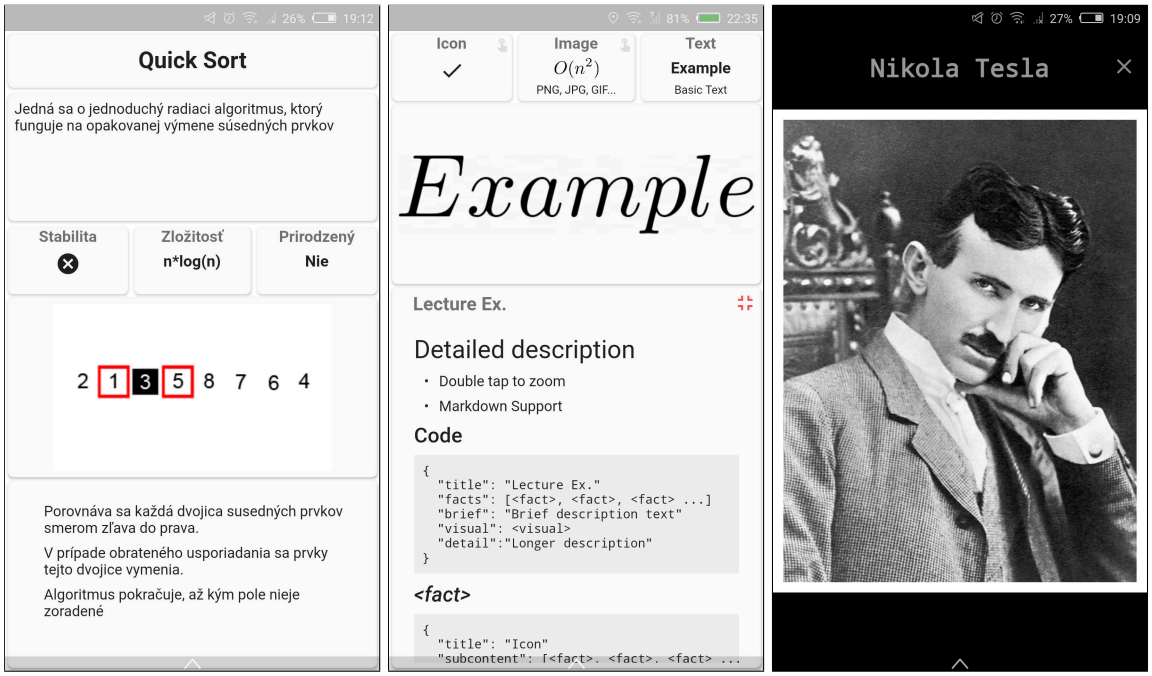
Bubble Sort a Quick Sort. Práve v súvislosti s ich tvorbou som premýšľal, ako podať užívateľovi informáciu o časovej zložitosti algoritmu, ktorá je často považovaná za dôležitú, a to bez toho, aby ju užívateľ musel hľadať vo výkladovom texte. Výsledkom tejto úvahy bolo zakomponovanie časti faktov do pôvodnej štvorprvkovej štruktúry. Ďalšiu potrebu, ktorú som na základe tvorby prvých lekcií vyvodil, bola nutnosť podpory animovaných obrázkov, ktoré sa ukázali pri výklade týchto algoritmov ako veľmi efektívny spôsob vizualizácie algoritmu. Poslednou zmenou bolo odstránenie bielych miest vo viacerých oblastiach, ktoré okupovali nemalú časť obrazovky a tak neostalo miesto pre potrebné informácie.

Výsledkom testovania prvotného návrhu bola nutnosť zväčšenia prvku obsahujúceho detailný text a zároveň rozšírenie funkcionality jednotlivých faktov tak, aby po stlačení sprístupňovali bližší popis vybraného faktu. Dôvodom pridania tejto funkcionality bolo, že v určitých situáciách užívateľ cítil potrebu dohľadať bližší význam daného faktu. V tomto bode vývoja sa tiež núkala zmena spôsobu štrukturalizácie výukového textu, kedy mal v tom čase podobu jednoduchého textu, kde jediným spôsobom štrukturalizácie bolo oddelenie jednotlivých častí textu novým riadkom. Preto bolo cieľom ďalšej verzie túto nedokonalosť vyriešiť.

Problém štrukturovania výukového textu

Na začiatku bola vykonaná analýza dostupných balíčkov umožňujúcich štrukturalizovanie textu, z ktorý som zvolil dva, ktoré sa javili pre tento účel ako najvhodnejšie. Jednalo sa konkrétne o balíček `flutter_tex`[1] (štrukturalizuje text pomocou zápisu vo forme LaTeX) a `flutter_markdown`[2] (štrukturalizuje text pomocou zápisu vo formáte Markdown). Knižnica `flutter_tex` sa javila ako vhodnejšia pre použitie, pretože nielen že dokázala štrukturalizovať text, ale taktiež v sebe zaobaľovala vykresľovanie matematických a chemických rovníc, čo by veľmi uľahčilo vývoj aplikácie. Jej problém však spočíval v tom, že vykresľovanie textu bolo realizované pomocou natívneho widgetu `WebView`, čo so sebou prinášalo kľúčovú nevýhodu, kedy widget neposkytoval možnú paletu funkcií na zrealizovanie navrhnutého užívateľského rozhrania, ktoré by podporovalo dynamické prispôsobovanie veľkostí jednotlivých prvkov na základe textového obsahu. Bolo tomu tak z dôvodu, že nebolo možné programovo získať veľkosť natívneho widgetu `WebView`, a v prípade alokovania menšieho priestoru ako rozmeru textového obsahu, bol daný prvok rolovateľný, čo bolo neprípustné. Preto je v aktuálnej verzii aplikácie využívaný balíček `flutter_markdown`, ktorý touto negatívnou vlastnosťou nedisponuje. Problémom tohto balíčka je neschopnosť vykresľovania rovníc. Preto bola na stole myšlienka spojiť silu týchto dvoch knižníc dohromady a text štrukturalizovať pomocou knižnice `flutter_markdown` a rovnice vykresľovať pomocou balíčka `flutter_tex`. Tento spôsob kombinovania knižníc sa ukázal ako nepoužiteľný pre účely tejto aplikácie, a to hlavne kvôli absencii možnosti synchronizovať veľkosti písma vykresľovaných rovníc a textu. Nakoniec teda musela byť zvolená cesta, kedy sa rovnice integrujú do štrukturalizovaného textu vo forme obrázkov, ktorá bola implementovaná pomocou preťaženia funkcie `imageBuilder` widgetu `Markdown` obsiahnutom v balíčku `flutter_markdown`. Táto komplikácia so sebou priniesla nutnosť rozšírenia reťazca sprístupnenia kurzu z kapitoly 6.1.1 o fázu generovania matematických vzorcov.

[1] https://pub.dev/packages/flutter_tex
[2] https://pub.dev/packages/flutter_markdown



Obr. 7.3: Verzia so statickými veľkosťami jednotlivých prvkov (vľavo). Verzia s umožneným zväčšovaním časti detailného popisu (v strede). Užívateľské rozhranie umožňuje zväčšiť vizualizáciu na celú obrazovku (vpravo).

Druhá iterácia

V druhej iterácii bol pridaný spôsob štrukturalizácie textu opísaného v predchádzajúcom texte a zároveň bol odstránený problém nedostatočného priestoru pre detailný text a to spôsobom, kedy je hlavná lišta a krátky popis lekcie skrytý a časť zobrazujúca detailný text tento pôvodný priestor nadobudne. Celý tento proces je pre lepšiu dynamiku užívateľského rozhrania realizovaný pomocou zreťazenej animácie, ktorá je implementovaná pomocou tried `AnimationController`, `Animation` a `Tween`. Pre správnu funkčnosť tejto animácie musel byť vždy vypočítaný rozdiel medzi aktuálnym a výsledným priestorom, ktorý bude prvok Detailného popisu pokrývať a to so zreteľom na aktuálne rozmery ostatných prvkov rozhrania. Animácia sa zároveň musela prispôsobiť skutočnosti, že niektoré elementy nadobúdajú dynamické rozmery na základe ich obsahu. Preto bolo kľúčové získavanie týchto rozmerov pre potrebné výpočty animácie. Rozmery jednotlivých widgetov sú získavané pomocou triedy `GlobalKey` a metód `currentContext`, `findRenderObject`. Indikovanie možnosti zväčšenia detailného popisu bolo indikované ikonkou v pravom hornom rohu prvku obsahujúceho detailný text, tak ako to je vyobrazené na obrázku 7.3 (v strede).

V tejto iterácii bola zároveň vytvorená teoretická časť lekcie, ktorá sa venuje biografii známej osobnosti (Nikola Teslu). Vytvorením tohto výukového prvku bol odhalený ďalší nedostatok užívateľského rozhrania, kedy vyhradený priestor pre vizualizáciu nebol vhodný pre obrázky, ktorých šírka bola menšia ako výška. Dôsledkom tohto nedostatku bol znova neefektívne využitý priestor, viď obrázok 7.4 (vľavo). Preto bola pre takýto prípad implementovaná úprava rozloženia jednotlivých prvkov a to tak, že v prípade obrázkov spomenutých rozmerov sa presunul panel faktov z pozície nad vizualizáciou na pravú stranu od vizualizácie. To zároveň umožnilo zväčšiť výšku vizualizáciu o pôvodnú výšku panela faktov.

Taktiež bola pridaná možnosť zväčšenia vizualizácie na celú obrazovku zariadenia. Tieto zmeny je možné vidieť na obrázkoch 7.4 (vpravo) a 7.4 (v strede a vľavo).

Tretia iterácia

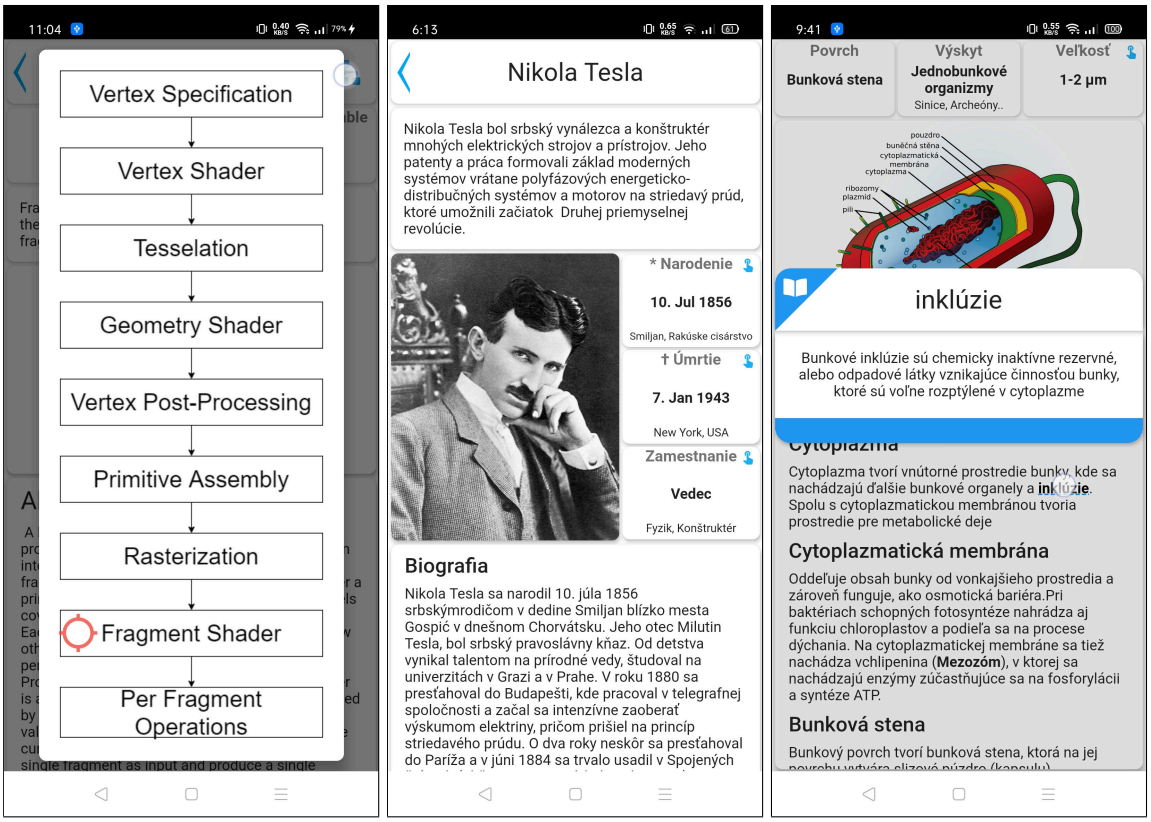
Predchádzajúca verzia preukázala nedostatky, kedy sa testovacie subjekty zhodli na tom, že zväčšovanie priestoru detailného textu by bolo vhodnejšie implementovať pomocou ťahu prstom (postupného rolovania) detailného textu smerom nahor. Spomenuté zlepšenie bolo následne implementované pomocou widgetu `GestureDetector` a jeho metódy `onPanUpdate`, predstavujúcich funkciu, ktorá je vyvolaná s každou udalosťou ťahu prstom užívateľa. Táto funkcia umožnila výpočet kroku animácie na základe normalizácie derivácie pozície ťahu prstom vzhľadom k rozdielu počiatočnej a chcenej výšky prvku obsahujúceho detailný popis. Zároveň sa ukázalo, že pre ľavorukých užívateľov je umiestnenie faktov na pravej strane od vizualizácie nekomfortné a keďže je zobrazenie detailu faktu realizované gestom podržania prsta na danom fakte, tak si užívateľ prekrýval obrazovku rukou. Toto zistenie bolo kľúčové pre zavedenie nastavení, ktoré umožňujú prispôsobenie užívateľského rozhrania aj pre ľavorukých užívateľov (viď obr. 7.4 – vpravo).

V tejto časti vývoja aplikácie bola vytvorená lekcia, ktorá sa venuje popisu práce Fragment Shaderu, kde vznikol koncept vyobrazovania hierarchie, kedy bolo vhodné znázorniť pozíciu tohto prvku vzhľadom k celej štruktúre zreťazeného spracovania OpenGL, a ktorá by mohla byť zároveň využiteľná aj v iných výkladoch. Do prvku hierarchie bol pridaný ukazovateľ v podobe blikajúceho červeného terčíka, ktorý slúži na označenie pozície v rámci hierarchie. Využitie hierarchie je zobrané na obrázku 7.5 (vľavo).

Obr. 7.4: Verzia, kde užívateľské rozhranie neefektívne využíva priestor v prípade obrázkov s väčšou výškou ako šírkou (vľavo). Efektívnejšie rozloženie prvkov v prípade portrétov (v strede). Užívateľské rozhranie prispôsobené aj pre ľavorukých užívateľov (vpravo).

Štvrtá iterácia

Štvrtá iterácia prebiehala v duchu miernych zmien, kedy sa menili iba detailné prvky užívateľského rozhrania. Prvým z nich bola zmena implementácie zväčšovania priestoru pomocou widgetu `GestureDetector`, na realizáciu pomocou objektu `ScrollView` a využitím triedy `ScrollNotification` a `ScrollPhysics`. Táto zmena zlepšovala pôvodnú verziu v tom, že užívateľ jedným ťahom prsta nielenže zväčšil priestor detailného popisu ale zároveň mohol tým istým ťahom prsta pokračovať v rolovaní predmetného text. V implementácii pomocou triedy `GestureDetector` užívateľ najprv zväčšil text a až opätovným ťahom mu bolo umožnené rolovanie textu, čo bolo pre užívateľov otravné a zmätočné. Ďalším prvkom zlepšenia bolo pridanie zvukového prehrávača a indikovanie interaktívnych prvkov pomocou modrej farby (viď obrázok 7.5 – v strede). Spomenuté zlepšenia sa osvedčili a tým vznikla takmer finálna verzia rozloženia a ovládateľnosti pre teoretickú časť lekcie.



Obr. 7.5: Zobrazenie prvku hierarchie s blikajúcim červeným terčíkom na zvýraznenie potrebnej časti obrázka (vľavo). Zvýraznenie interaktívnych prvkov modrým sfarbením (v strede). Inovatívny spôsob využívania slovníka (vpravo).

Piata iterácia

Piata iterácia prebiehala v procese integrácie inovatívneho prvku, ktorým mal byť slovník dostupný priamo z textu výkladu. Ideou bolo vyhľadať pojmy definované v slovníku kurzu, ktoré sú zároveň obsiahnuté vo výklade lekcie a následne umožniť užívateľovi zobraziť príslušnú definíciu pojmu podržaním prsta nad týmto slovom. Preto, aby užívateľ pri držaní prsta neprekrýval svojou rukou zobrazenú definíciu, musel byť implementovaný

výpočet umiestnenia vyobrazenia tejto definície na základe polohy stlačenia prstom. Možnosť interakcie s takými slovami bola indikovaná podčiarknutím modrou farbou. Voči tejto funkcionalite, neboli vznesené žiadne výhrady no riešenie trpí nasledujúcimi problémami:

- Citlivosť na veľkosť písma – vždy je zvýraznené iba slovo, ktoré je presnou zhodou pojmu definovanom v slovníku. Napríklad ak je v slovníku definovaný pojem „WINE“, tak slovo „wine“, ktoré sa nachádza v texte, nebude poskytovať možnosť zobrazenia definície. Dôvodom toho je práve viacznačnosť takýchto slov, kde slovo „WINE“ s veľkými písmenami značí skratku pre emulátor určený na spúšťanie windowsových aplikácií v linuxovom prostredí a slovo „wine“ znamená víno.
- Nepracuje so skloňovaním slova.

Preto zatiaľ nie je možné považovať túto funkcionalitu za plnohodnotnú. Túto špeciálnu vymoženosť slovníka zobrazuje obrázok 7.5 (vpravo).

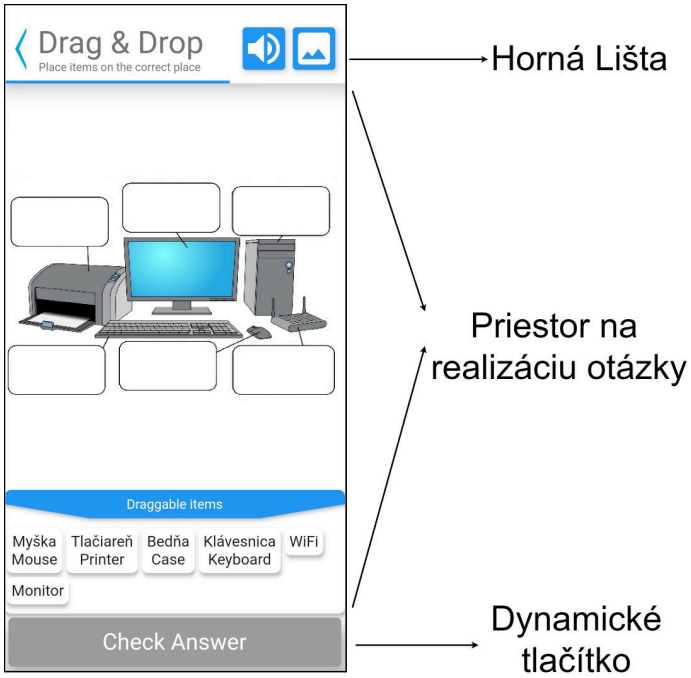
7.2 Praktická časť

Na základe analýzy existujúcich riešení bola zvolená forma praktického precvičovania teoretickej časti lekcie vo forme kvízových otázok. Pre každú lekciu môže tvorca kurzu definovať otázky pomocou súboru vo formáte JSON obsahujúcom pole definícií otázok, pričom pole otázok v jednom takomto súbore predstavuje sadu otázok. To, ktorá sada resp. sady otázok sú precvičované v danej lekcii, je určené na základe hodnoty kľúča `quiz` v súbore definujúcom lekciu, bližšie opísaného v kapitole 7.1.1. Platí, že jednej lekcii môže byť priradený ľubovoľný počet súborov definujúcich rôzne sady otázok.

Z pohľadu užívateľa prebieha praktické precvičovanie látky vo forme krátkych sérií o dĺžke 10 až 15 náhodne vybraných otázok z príslušnej sady/sád otázok. Vytváranie série z množiny otázok nie je úplne náhodné ale berie do úvahy aj fakt, koľkokrát už bola daná otázka užívateľom precvičovaná. Táto informácia je uchovávaná o každej otázke pomocou akumulátora v inštancii reprezentujúcej danú otázku, ktorá je v rámci aplikácie unikátna viď kapitola 5.5. Na základe hodnôt týchto akumulátorov je následne určená pravdepodobnosť výberu danej otázky a to spôsobom, kedy otázka s najväčšou hodnotou akumulátora má najnižšiu pravdepodobnosť výberu, a to z dôvodu, že bola veľakrát precvičovaná.

7.2.1 Užívateľské rozhranie kvízu

Séria otázok, ktorá je riešená užívateľom, je prezentovaná vo forme kvízu. Pre prezentovanie jednotlivých otázok bola vytvorená šablóna, ktorá spája a umožňuje prístup k spoločným prvkom užívateľského rozhrania potrebných pre každý typ otázky. Táto šablóna je zobrazená na obrázku 7.6 a pozostáva z hlavnej lišty, priestoru pre realizáciu otázky a dynamického tlačidla. Horná lišta zobrazuje, o aký typ otázky sa jedná s prípadnou doplňujúcou informáciou k spôsobu riešenia. Ďalej lišta obsahuje dve tlačidlá pre sprístupnenie zvukových a obrázkových príloh, návratovú šípku pre ukončenie kvízu (ktorá zároveň zobrazuje dialóg s potvrdením ukončenia kvízu, aby užívateľ neukončil kvíz omylom) a prvok znázorňujúci pokrok v rámci postupu užívateľa daným kvízom. V priestore pre realizáciu otázky sa nachádza časť užívateľského rozhrania zobrazenej otázky, ktoré sa líši od typu otázky. Dynamické tlačítko predstavuje hlavný ovládací prvok kvízu, ktorý zabezpečuje správny prechod medzi jednotlivými otázkami a umožňuje užívateľovi vykonať kontrolu zadaných odpovedí.



Obr. 7.6: Popis rozloženia šablóny pre užívateľské rozhranie otázky.

Ukazovateľ pokroku

Ukazovateľ pokroku nebol vôbec súčasťou prvotného návrhu tejto časti aplikácie, no postupným testovaním aplikácie sa ukázal ako potrebný, pretože užívatelia cítili potrebu mať prehľad o tom, koľko otázok im ešte v danej sérii zostáva vyriešiť. Na základe tejto požiadavky bol pridaný do užívateľského rozhrania. Prvotným návrhom bol ukazovateľ pokroku, ktorý poskytoval presný prehľad pokroku pomocou oddelených obdĺžnikov, kde každý obdĺžnik reprezentoval práve jednu otázku (viď obrázok 7.7). Tento návrh vyzeral v poriadku, no časom sa ukázalo, že informovanie užívateľa takýmto spôsobom splnilo svoj účel, no niektorých užívateľov rozptyľoval v prípade, že bola definovaná séria o väčšom počte otázok. V takomto prípade ukazovateľ obsahoval veľa obdĺžnikov, čo často vyvolávalo stav, kedy bol užívateľ rozptýlený faktom, že ho čaká ešte veľa otázok na vypracovanie. Preto bola vykonaná zmena, kedy konečným riešením je spojitý ukazovateľ, ktorý síce informuje o pokroku, no pri väčšom počte otázok je z neho ťažšie presne vyčítať presný počet otázok, ktoré musia byť vypracované. Spomenuté dva typy ukazovateľov sú vyobrazené na obrázku 7.7.

Obr. 7.7: Ukazovateľ pokroku s indikáciou pokroku pomocou obdĺžnikov zobrazujúci sériu otázok o dĺžke 10 (vľavo). Ukazovateľ pokroku s indikáciou pokroku pomocou obdĺžnikov zobrazujúci sériu otázok o dĺžke 25 (v strede). Spojitý ukazovateľ pokroku (vpravo).

Pravdou však je, že testovanie ukazovateľov pokroku prebiehalo aj so sériami otázok dlhšími, ako je aktuálna maximálna dĺžka série generovaná aplikáciou. Aplikácia v svojej konečnej verzii generuje sériu o veľkosti maximálne pätnástich otázok.

Dynamické tlačidlo

Dynamické tlačidlo je dôležitým prvkom šablóny a má hneď niekoľko funkcií:

1. Informuje užívateľa o tom, či je otázka vypracovaná dostatočne na to, aby mohla byť vykonaná jej kontrola (napríklad neumožní kontrolu otázky, pokiaľ nie sú spojené všetky dvojice prvkov).
2. Umožňuje vykonať kontrolu vypracovania danej otázky.
3. Indikuje správnosť odpovede.
4. Umožňuje zobrazenie návodu k správnemu riešeniu danej otázky, ktorý poskytuje tvorca kurzu. (Napríklad pri riešení matematickej úlohy zobrazí medzivýpočty, ktoré vedú k chcenému výsledku).

Realizácia toľkých funkcií jedného tlačítka bola implementovaná pomocou stavového widgetu s vlastnosťou dynamickej zmeny podoby na iné prvky. Z počiatku je tlačidlo v stave, kedy indikuje, či je možné vykonať kontrolu nad vypracovaním otázky. To, že túto kontrolu otázky nie je možné vykonať, tlačítko indikuje sfarbením do sivej farby a nereaguje na žiadne užívateľské gestá. V opačnom prípade znázorňuje možnosť stlačenia zmenou sfarbenia do modrej farby. Po stlačení a spustení funkcie kontroly sa riešenie otázky vyhodnotí. V prípade, že bola odpoveď nesprávna, tak následné správanie aplikácie záleží od typu otázky. Ak sa jedná o otázku s konečnou množinou odpovedí, tak nesprávnosť odpovede aplikácia vyobrazí začervenaním obrazovky a animáciou horizontálneho zatrasenia obrazovky. Tým užívateľovi naznačí, že musí zmeniť svoje riešenie. Takto aplikácia reaguje, až kým nie je riešenie správne. Ak sa jedná o otázku s nekonečnou množinou možných odpovedí (Doplňovanie do textu, Otázka s plnou odpoveďou), tak bez ohľadu na to, či je odpoveď správna alebo nesprávna, tlačítko morfuje do podoby dvoch tlačidiel. Jedno z tlačidiel predstavuje možnosť zobrazenia návodu na riešenie úlohy, ktoré poskytuje tvorca kurzu a druhé predstavuje možnosť pokračovania na ďalšiu otázku série. V prípade, že je odpoveď nesprávna, tak táto otázka bude musieť byť riešiteľom na konci kvízu opätovne vypracovaná.

Obr. 7.8: Tlačítko indikuje, že otázka nebola ešte dostatočne vypracovaná (vľavo). Tlačidlo indikuje, že je možné vykonať kontrolu otázky (v strede). Tlačítko zmení svoju podobu do podoby tlačidla pre zobrazenie návodu na riešenie a tlačítka pre pokračovanie v kvíze indikujúce správnu (zelená) resp. nesprávnu (červená) odpoveď (vpravo)

Prvotne bola indikácia správnosti či nesprávnosti odpovede realizovaná pomocou dočasne zobrazovanej lišty implementovanej pomocou triedy `SnackBar`[3], ktorá zobrazovala správy „Correct“ na zelenom pozadí a „Incorrect“ na červenom pozadí. Následne bol s pribúdajúcim typom otázok tento spôsob vyhodnocovania nedostatočný a to z dôvodu, že neposkytoval užívateľom dostatočne detailnú informáciu o správnosti jeho odpovedí.

Zvukový prehrávač

Tento prvok užívateľského rozhrania umožňuje užívateľovi prehrať jeden alebo aj viacero zvukových súborov. Vzhľadom na počet zvukových príloh pracuje v dvoch módoch. V prí-

[3] https://api.flutter.dev/flutter/material/SnackBar-class.html

pade, že je k dispozícii iba jeden zvukový súbor, tak prehrávač kliknutím na tlačidlo začne prehrávať tento zvuk. V prípade, že má k dispozícii viac ako jeden takýto súbor, tak sa zobrazí zoznam prehrávačov, ktorý umožňuje výber spomedzi súborov, ktoré majú byť prehraté. Jednotlivé prehrávače prislúchajúce týmto súborom pozostávajú z tlačítka pre spustenie resp. zastavenie prehrávania a ukazovateľa časového pokroku zvukovej stopy. Zvukový prehrávač bol na základe testovania vylepšený o tieto vlastnosti. Jednotlivé prehrávače sú v prípade viacerých prehrávačov navzájom synchronizované tak, že ak je aktivovaný jeden prehrávač, tak iný prehrávač, ktorý ešte nedokončil prehrávanie zvuku, svoju činnosť zastaví. Tento prvok bol implementovaný z dôvodu, že užívatelia mali tendenciu spúšťať iné zvuky ešte pred tým, kým sa prehrávanie pôvodného zvuku neukončilo, čo spôsobilo paralelné prehrávanie týchto zvukov. Zároveň bol pod samotné hlavné tlačítko zvukového prehrávača pridaný ukazovateľ pokroku prehrávania, a to pre prípad, keď užívateľ počas prehrávania zvuku zoznam prehrávačov minimalizoval. Posledná zmena tohto prvku bola vykonaná preto, že správanie tlačidla bolo pre užívateľa zmätočné. Jednalo sa o reakciu na užívateľa, kedy v jednom prípade sa stisnutím tlačítka hneď prehrával zvuk a v inom sa zobrazil zoznam prehrávačov. Pre zlepšenie predikcie tohto správania bol pridaný indikačný prvok vo forme ikony v tvare zoznamu na pravej strane tlačidla.

Štruktúra vo formáte JSON

V tejto časti bude opísaný spôsob definície prvkov, ktoré sú spoločné pre všetky typy otázok. Jedná sa konkrétne o zvukové prílohy a obrázkové prílohy, ktoré sú dostupné pomocou tlačidiel umiestnených na hornej lište, a návod k riešeniu úlohy poskytnutý tvorcom. Každý z kľúčov prislúchajúcich týmto prvkov sa nachádza na najvyššej úrovni JSON objektu definujúceho danú otázky. Definícia objektov reprezentujúcich jednotlivé otázky budú opísané v ďalšom texte.

```
1   {
2       "type": "<typ_otazky>",
3       "visual" :["@~/Image.png"]
4       "sounds": [
5              {
6                  "title": "Audio 1",
7                  "credits": "Some Author or License 1",
8                  "source": "@~/Sounds/Sound1.mp3"
9              },
10             ...
11         ],
12      "answer_detail": "Návod k riešeniu otázky",
13       ...
14  }
```

Kde jednotlivé kľúče predstavujú nasledovné:

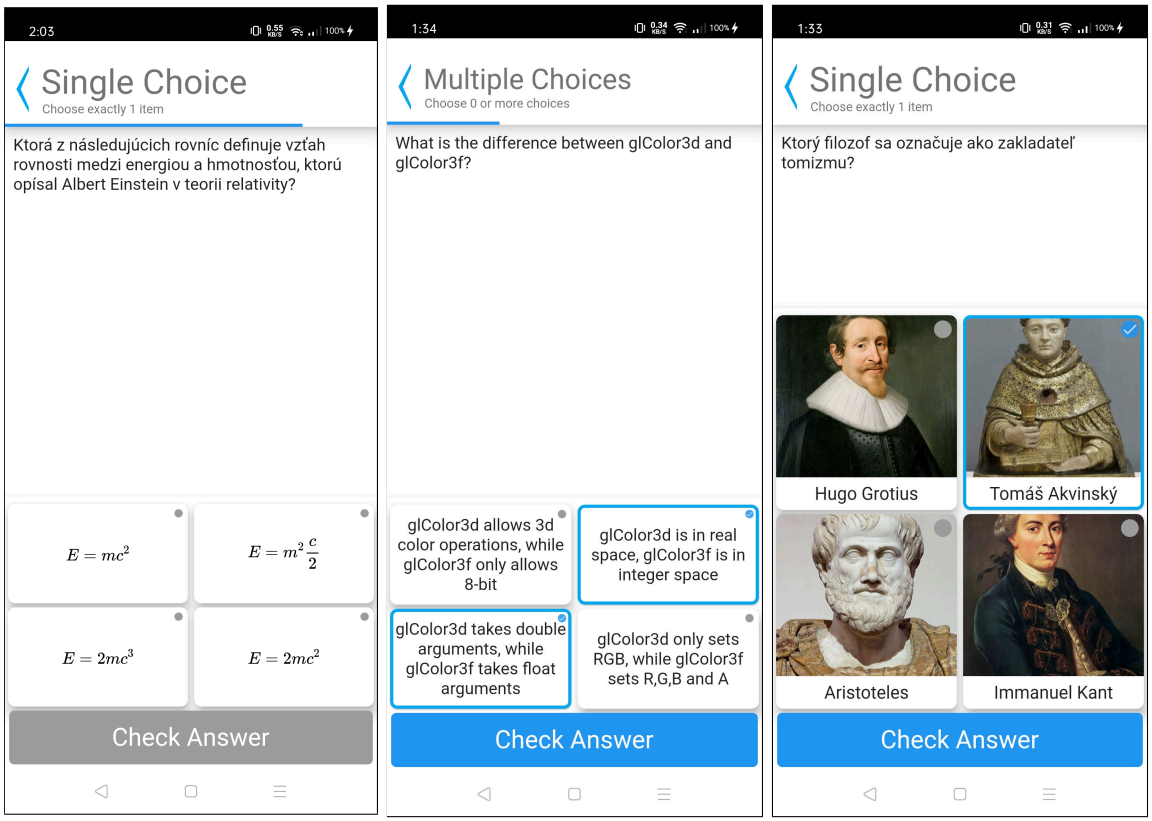
- `type` (*reťazec*) – definuje typ otázky. V tomto prípade je zobrazený iba pre lepšie pochopenie umiestnenia kľúčov pre spoločné prvky otázok. Viac informácii o tomto poli je uvedených v ďalšom texte.
- `visual` *(pole referencií)* – predstavuje odkazy na obrázkové súbory, ktoré majú byť priložené k danej otázke.

- `sounds` (*pole objektov*) – pole objektov predstavujúce zvukové prílohy lekcie. Každý objekt tohto poľa ma takúto štruktúr:
  - `title` (*reťazec*) – predstavuje názov zvukovej stopy.
  - `credits` (*reťazec*) – predstavuje meno autora zvukového súboru alebo licenciu.
  - `source` (*referencia*) – odkaz na zdrojový súbor so zvukovým obsahom.
- `answer_detail` (*markdown*) – detailné vysvetlenie správnej odpovede, ktoré je dostupné pomocou dynamického tlačítka po vyhodnotení otázky.

7.2.2 Výber jednej alebo viacerých možností

Otázka s výberom jednej alebo viacerých možností patrí medzi všeobecne známy spôsob precvičovania znalostí, ktorá prebieha vo forme, kedy je užívateľovi predložená množina výrokov a na základe zadania musí vyberať z tejto množiny tie správne. Užívateľské rozhranie reprezentujúce tento typ otázky pozostáva z dvoch častí – prvou je text zadania otázky a druhou je časť, kde užívateľ vyberá správne odpovede z množiny odpovedí. Prvotná verzia užívateľského rozhrania umožňovala reprezentovať jednotlivé možnosti iba v podobe textu, no následne bola pridaná podpora zvukov a obrázkov. S príchodom podpory obrázkov musel byť vyriešený problém prepočítavania voľného priestoru vzhľadom na počet možností, z ktorých užívateľ vyberá tie správne, tak aby bolo dostupné miesto využité čo najefektívnejšie. Taktiež sú definované dva typy správania užívateľského rozhrania, ktoré sa menia na základe toho, či má užívateľ vybrať práve jednu odpoveď (single-choice) alebo ľubovoľný počet (multiple-choice). Hlavným rozdielom je to, že pri výbere práve jednej možnosti nie je užívateľovi umožnené vykonať kontrolu odpovede, kým neoznačí práve jednu možnosť a zároveň aplikácia neumožní zvolenie viacerých možností. Užívateľské rozhranie otázky a spomenuté vlastnosti je možno sledovať na obrázku 7.9.

Časť užívateľského rozhrania, ktorá sa pri testovaní ukázala ako najviac problematická, bola indikácia zvolenej odpovede. Prvá verzia využívala vlastnosti material designu, kedy bolo označenie indikované výhradne na základe výšky a príslušného tieňa dlaždice reprezentujúcej jednu z možností. Táto metóda pri menších počtoch možností dostatočná, avšak nie ako veľmi ideálna sa javila v prípade väčšieho počtu prvkov, kedy bol tento spôsob indikácie na prvý pohľad užívateľa neprehľadný. Preto bol v druhej iterácii pridaný indikátor vo forme modrého krúžku s ikonou fajky. Aktuálna verzia využíva na zvýraznenie zvoleného prvku obe spomenuté indikačné prvky, no zároveň musela byť doplnená aj o zvýraznenie zvolenej dlaždice pomocou orámovania dlaždice. Bolo to najmä z dôvodu, kedy pri možnostiach, ktoré obsahovali obrázky bol modrý krúžok s ikonkou málo viditeľný a zároveň pri textových odpovediach čiastočne prekrýval text odpovede pričom ho pre zachovanie dobrej viditeľnosti nebolo možné bez inej zmeny zmenšiť. Preto bolo pridané modré orámovanie, ktoré je priestorovo menej náročnejšie ako kruhový indikátor a zároveň aj viac viditeľné. Na základe úlohy s výberom z možností bol do šablóny pridaný krátky popis otázky pod názov typu otázky, kedy užívatelia často nevedeli určiť minimálny počet možných odpovedí. Taktiež bola pridaná funkcia automatického prispôsobenia veľkosti textu na základe vymedzeného priestoru dlaždice. Pre lepšiu estetiku bola následne doplnená synchronizácia týchto textov vrámci všetkých dlaždíc. To znamená, že ak je nejaký text dlhý a musí zmenšiť svoju veľkosť, tak aj texty v ostatných dlaždiciach zmenia svoju veľkosť na rov-



Obr. 7.9: Pri výbere jednej možností užívateľské rozhranie neumožní kontrolu otázky, kým nie je zvolený aspoň jeden prvok (vľavo). Otázka s výberom viacerých možnosti (v strede). Možnosti môžu byť realizované aj pomocou obrázkov (vpravo).

nakú. Automatické prispôsobovanie textu a synchronizácia bola implementovaná pomocou balíčka `auto_size_text`[4] a jeho tried `AutoSizeText` a `AutoSizeGroup`.

Štruktúra vo formáte JSON

Táto časť textu sa bude venovať štruktúre definície otázky s výberom jednej alebo viacerých možností. Definícia otázky môže byť definovaná v súbore definujúcom sadu otázok alebo v samostatnom súbore. Definícia tohto typu otázky má nasledujúcu štruktúru:

```
{
    "type": "choice",
    "text": "Which of the following statements are true?",
    "singleChoice" :true,
    "choices": [4,6],
    "imageFit" :"cover",
    "true": [
        {
            "text": "True 1",
            "image": "@~/True1.png",
            "sound": "@~/True1.mp3"
```

[4] https://pub.dev/packages/auto_size_text

```
        },
        ...
    ],
    "false": [
        {
            "text": "False 1",
            "image": "@~/False1.png",
            "sound": "@~/False1.mp3"
        },
        ...
    ]
},
```

Hlavná štruktúra súboru:

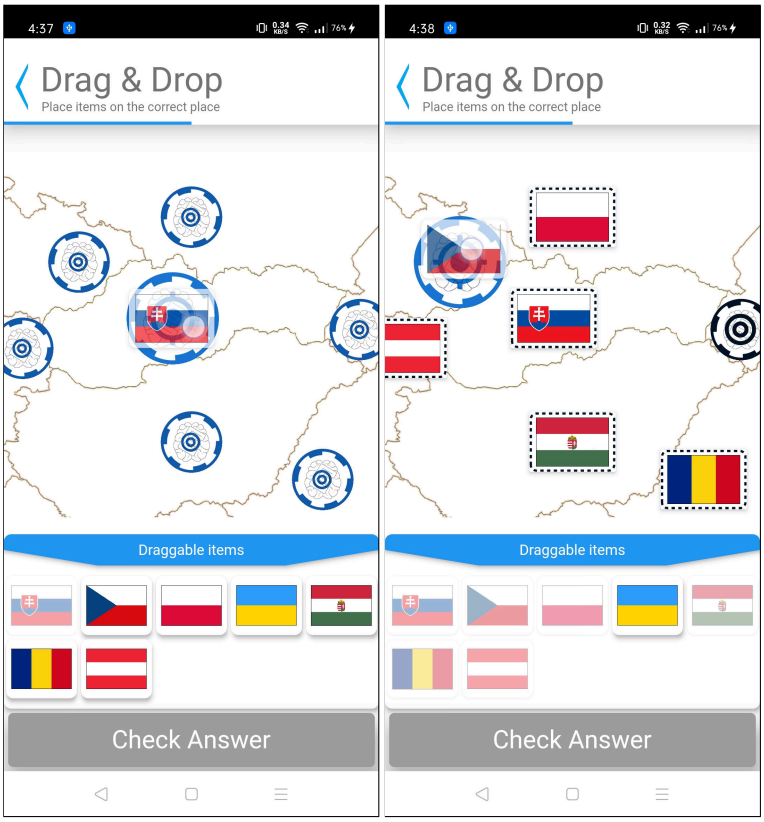
- `type` (*reťazec*) – definuje typ otázky. Pre otázku typu „Výber jednej alebo viacerých možností“ musí nadobúdať hodnotu „choice“.
- `text` (*markdown*) – určuje textové zadanie úlohy.
- `imageFit` (*reťazec*) – určuje správanie roztahovania obrázkov v dlaždiciach a môže nadobúdať hodnoty „fill“ (roztiahne obrázok vzhľadom na veľkosť dlaždice – zmení pomer strán), „cover“ (vyplní obrázok vzhľadom na veľkosť dlaždice – nezmení pomer strán obrázka), „contain“ (neroztiahne obrázok, iba zmenší tak, aby sa zmestil do dlaždice – predvolená hodnota).
- `choices` *(pole čísel)* – umožňuje definovať počet možnosti z ktorý bude užívateľ vyberať.
- `singleChoice` *(pole objektov)* – ak nadobúda hodnotu `true`, tak bude užívateľ vyberať z množiny odpovedí, kde je iba jedna správna, ak nadobúda hodnotu `false`, tak užívateľ môže vybrať ľubovoľný počet odpovedí z danej množiny.
- `true` *(pole objektov)* – pole obsahujúce objekty, ktoré sú správnou odpoveďou.
- `false` *(pole objektov)* – pole obsahujúce objekty, ktoré sú nesprávnou odpoveďou.

Štruktúra objektu polí `true` a `false`:
Každý objekt v poli `true` resp. `false` definuje dlaždicu reprezentujúcu pravdivú resp. nepravdivú hodnotu, ktorú musí užívateľ na základe jej pravdivosti označiť resp. neoznačiť.

- `text` (*reťazec*) – textový nápis alebo matematický vzorec zobrazený na dlaždici.
- `image` (*referencia*) – odkaz na obrázkový súbor, ktorého obsah je vyobrazený na dlaždici.
- `sound` (*referencia*) – odkaz na zvukový súbor, ktorého obsah je spustený pri stlačení dlaždice prstom.

7.2.3 Doplňovanie do obrázka

Otázka umožňuje precvičovanie znalostí na základe doplňovania textu (obr. 7.11) alebo obrázkov (obr. 7.10) na správnu pozíciu v zdrojovom obrázku. Tvorca kurzu môže okrem zdrojového obrázka a prvkov, ktoré majú byť doplnené do tohto obrázka, definovať aj masku, ktorá slúži na prekrytie častí, ktoré majú byť užívateľovi skryté. Túto masku je možné definovať dvomi spôsobmi a to buď pomocou obrázka alebo pomocou vykresľovania maskovacích elementov, ktorých definícia je bližšie opísaná v časti definujúcej súbor JSON. Užívateľské rozhranie pozostáva z časti doplňovacej (zdrojový obrázok a maska) a z časti



Obr. 7.10: Prijímací slot zväčší svoju veľkosť dvojnásobne (vľavo). Prvky, ktoré nie sú ešte doplnené do obrázka, tak sú v dolnej časti zobrazené výraznejšie (vpravo).

prvkov, ktoré sú do obrázka doplňované užívateľom. Podobne ako pri otázke s doplňovaním textu je doplňovanie prvkov do obrázka realizované pomocou gesta „potiahni a pusť“. Po začatí gesta „potiahni a pusť“ sú v časti zdrojového obrázka vyobrazené terčíky zobrazujúce pozície obrázka, na ktoré je možné jednotlivé preťahované prvky položiť (ďalej len sloty). Veľkosť týchto slotov bola prvotne statická, no vzhľadom na fakt, že sa užívateľom ťažko odhadovalo, kedy daný slot prijíma preťahovaný prvok, tak bola implementovaná zmena, kedy sa veľkosť prijímacieho slotu zväčší oproti ostatným slotom dvojnásobne. Zároveň sa preťahovaný prvok, alebo prvky, ktoré už sú na svojích miestach indikujú v spodnej časti užívateľského rozhrania zvýšenou priehľadnosťou aby boli užívateľovi zvýraznené prvky, ktorým ešte nepriradil miesto v zdrojovom obrázku. Takéto správanie je možné sledovať na obrázku 7.10.

Na základe testovania bola tiež doplnená funkcionalita, ktorá zabezpečuje prvotné načítanie a zobrazenie masky zdrojového obrázka a až následne zobrazí zdrojový obrázok.

Dôvodom bolo to, že pri starších zariadeniach, ktoré mali oneskorenie zobrazenia väčších obrázkov sa stávalo, že najprv bol zobrazený zdrojový obrázok s odpoveďami a až následne jeho maska. To na krátky moment umožnilo vidieť užívateľovi riešenie úlohy. Príklad využitia masky je znázornený na obrázku 7.11. Druhá časť užívateľského rozhrania zobrazuje prvky, ktoré majú byť vložené na správne miesto zdrojového obrázka. Umiestnenie týchto prvkov bolo pôvodne zabezpečené widgetom `Wrap`, no s postupnou podporou doplňovania obrázkov do zdrojového obrázka a príchodom neuniformity výšok jednotlivých prvkov bolo umiestňovanie pomocou tohto widgetu nevhodné, pretože sa ukázalo ako neefektívne. Pre lepší spôsob rozloženia bol využitý widget `StaggeredGridView` z balíčka `flutter_staggered_grid_view`[5], ktorý umožňoval efektívne rozloženie prvkov s neuniformnou veľkosťou. Testovaním preťahovateľných prvkov bolo tiež zistené, že prvky pozostávajúce výhradne z textu pri procese preťahovania neboli viditeľné, pretože užívateľ si ich prekrýval prstom. Pre odstránenie tohto nedostatku bol pridaný ofset umiestnenia pre textové prvky počas preťahovania prstom. Túto zmenu možno sledovať na obrázku 7.11.

Obr. 7.11: Príklad využitia masky (znázornená ružovou farbou) pre prekrytie správnych odpovedí v obrázku (vľavo). Ofset pri preťahovaní textového prvku zabezpečuje prevenciu proti prekytiu textu prstom (vpravo).

[5] https://pub.dev/packages/flutter_staggered_grid_view

Štruktúra vo formáte JSON

Táto časť textu sa bude venovať štruktúre definície otázky s doplňovaním prvkov do obrázka. Definícia otázky môže byť definovaná v súbore definujúcom sadu otázok alebo v samostatnom súbore. Definícia tohto typu otázky musí nadobúdať nasledujúcu štruktúru:

```
{
    "type": "map",
    "map": "@~/sourceImage.png",
    "mask": "@~/sourceImageMask.png",
    "hide": [
        "c: 0.0:0.5 :0.5",
        "r: 0.04:0.72 :0.3:0.9 | b-beveled:1:1:0:0 | clr:255:40:120:1.0",
    ],
    "answers": [{
            "position": "0.5:0.8",
            "size": "0.3 :0.2",
            "sound": "@~/Sounds/alarm.ogg",
            "image": "@~/Image.png",
            "style": "clr:70:70:70:1.0 | font-clr:0:255:0 | b-beveled: 0.2:
                0.05 :0.05:0.0 "
        },
        {
            "text": "Text prvku",
            "position": "0.5:0.2",,
            "style": "clr:70:70:240:0.5 | font-clr:0:255:0:0.5"
        },
        ...
    ]
}
```

Hlavná štruktúra súboru:

- `type` (*reťazec*) – definuje typ otázky. Pre otázku typu „Doplňovanie do obrázka“ musí nadobúdať hodnotu „map“.
- `map` (*referencia*) – odkaz na zdrojový obrázok, do ktorého užívateľ doplňuje prvky.
- `mask` (*referencia*) – odkaz na súbor, ktorý je vyobrazený nad zdrojovým obrázkom a slúži na prekrytie tých častí zdrojového obrázku, ktoré by mohli obsahovať odpoveď.
- `hide` (*pole reťazcov*) – pole reťazcov popisujúcich maskovacie objekty, ktoré sú vyobrazené nad zdrojovým obrázkom a slúžia na prekrytie časti zdrojového obrázku, ktorá by mohla obsahovať odpoveď. Jedná sa o alternatívu k súboru definovanom kľúčom `mask`, avšak tvorca kurzu môže využiť obe spôsoby maskovania.
- `answers` *(pole objektov)* – pole definujúce jednotlivé objekty, ktoré majú byť doplnené do zdrojového obrázku užívateľom.

Štruktúra objektu poľa `answers`:
Predstavujú jednotlivé prvky, ktoré užívateľ preťahuje do zdrojového obrázku.

- `text` (*referencia*) – textový nápis zobrazený na prvku.
- `image` (*referencia*) – odkaz na súbor obrázku, ktorý je vyobrazený na prvku. Ak je definovaný obrázok aj text, tak je zobrazený iba obrázok.
- `sound` (*referencia*) – odkaz na zvukový súbor, ktorého obsah je prehratý s počiatkom gesta „ťahaj a pusť“.
- `size` (*reťazec*) – určuje priestor, ktorý má prvok zabrať na zdrojovom obrázku. Definícia je v tvare „<width>:<height>“, kde <width> určuje normalizovanú šírku prvku vzhľadom k šírke zdrojového obrázku a <height> určuje normalizovanú výška vzhľadom k výške zdrojového obrázku. Normalizovaná šírka nadobúda hodnoty v intervale <0,1>.
- `position` (*reťazec*) – určuje pozíciu prvku v zdrojovom obrázku. Definícia je v tvare „<x>:<y>“, kde <x> určuje normalizovanú horizontálnu súradnicu vzhľadom k šírke zdrojového obrázku a <y> určuje normalizovanú vertikálnu súradnicu vzhľadom k výške zdrojového obrázku. Normalizované súradnice nadobúdajú hodnoty v intervale <0,1>.
- `style` (*reťazec*) – umožňuje definovať štýl zobrazenia prvku. Jednotlivé vlastnosti sú oddelené pomocou znaku „|“. Tvorcovi kurzu je umožnené definovať tieto vlastnosti:
  - farba pozadia – v tvare „clr:<red>:<green>:<blue>:<opacity>“, kde <red>, <green> a <blue> predstavujú jednotlivé farebné zložky a môžu nadobúdať hodnotu 0-255 a <opacity> predstavuje prehľadnosť, ktorá nadobúda hodnoty v intervale <0,1>.
  - farba písma – v tvare „font-clr:<red>:<green>:<blue>:<opacity>“, kde <red>, <green> a <blue> predstavujú jednotlivé farebné zložky a môžu nadobúdať hodnotu 0-255 a <opacity> predstavuje prehľadnosť, ktorá nadobúda hodnoty v intervale <0,1>.
  - zaokrúhlenie rohov – v tvare „b-<type>:<tl>:<tr>:
:<bl>“, kde <type> určuje spôsob zaokrúhlenia rohov a môže nadobúdať hodnoty: „beveled“(hranaté), „rounded“(oblé) alebo „none“(bez zaokrúhlenia) a <tl>,<tr>,<bl>,
 predstavujú mieru zaokrúhlenia jednotlivých rohov pričom nadobúdajú nadobúdajú hodnoty v intervale <0,1> (tl – ľavý horný roh, tr – pravý horný roh, br – pravý dolný roh, bl – ľavý dolný roh).

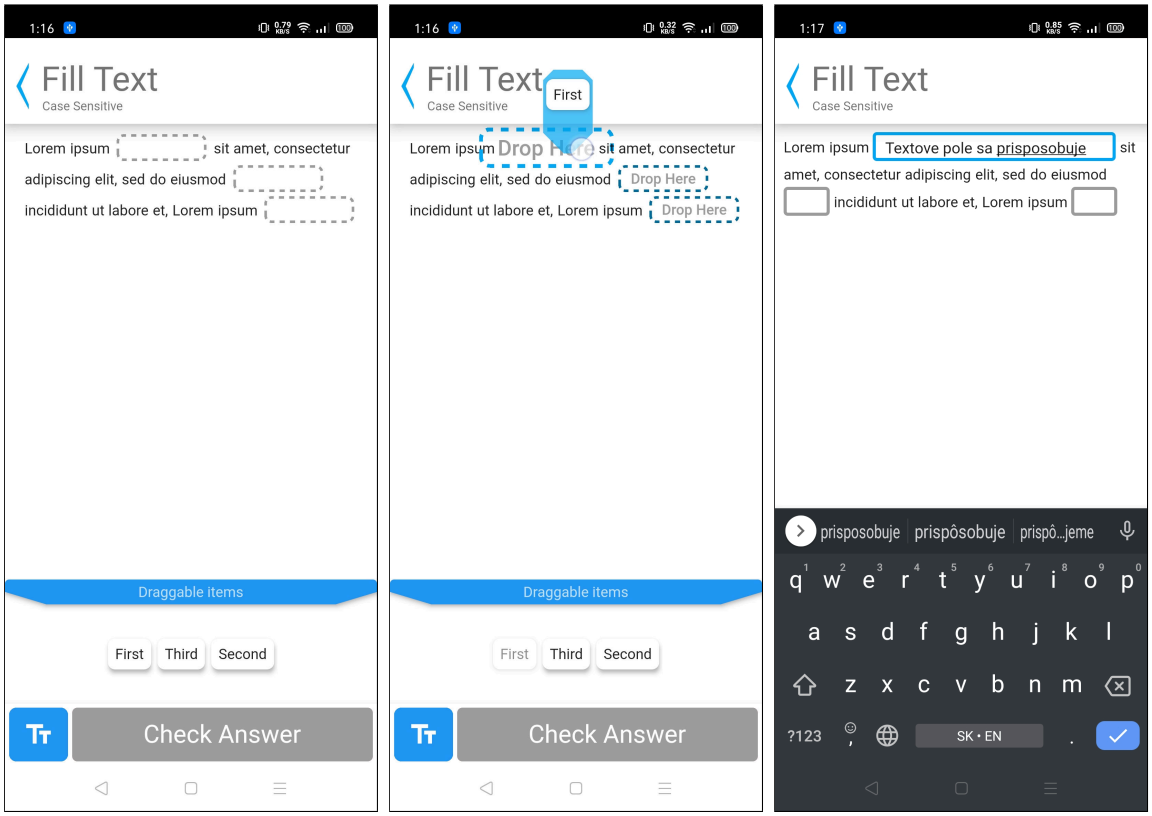
Štruktúra reťazca poľa `hide`: Pomocou týchto reťazcov môže tvorca kurzu definovať objekty prekrývajúce zdrojový obrázok. Je možné definovať dva rôzne typy objektov:

- Kruh – kde definícia je v tvare „c:<sx>:<sy>:<p>“, kde <sx>,<sy> sú súradnice stredu kruhu normalizované vzhľadom na rozmery zdrojového obrázka a <p> je priemer kruhu normalizovaný vzhľadom k výške obrázku.
- Obdĺžnik – kde definícia je v tvare „r:<ax>:<ay>:<bx>:<by>“, kde <ax>,<ay> sú normalizované súradnice ľavého horného bodu obdĺžnika a <bx>, <by> sú normalizované súradnice pravého dolného rohu obdĺžnika.

Jednotlivé objekty je možné štýlovať pomocou definície vlastnosti oddelených znakom „|“, kde definícia jednotlivých vlastnosti je zhodná s definíciou uvedenou v predošlom texte. Kruhovému objektu je možné nastaviť farbu pozadia a obdĺžnikovému objektu je možné nastaviť zaokrúhlenie rohov a farbu pozadia.

7.2.4 Doplňovanie textu

Tento typ otázky umožňuje doplňovať chýbajúce slová alebo menšie časti textu do celistvého textu. Spôsob doplňovania týchto častí textu je v konečnej verzii umožnený pomocou dvoch spôsobov – doplňovanie textu pomocou vstupného poľa a klávesnice, alebo výberom a preťahovaním odpovedí na správne miesta v texte. Tieto rôzne prístupy je možné sledovať na obrázku 7.12.



Obr. 7.12: Doplňovanie textu výberom z odpovedí pomocou preťahovania prvkov (vľavo a v strede). Doplňovanie textu pomocou vstupného poľa a klávesnice, pričom vstupné pole prispôsobuje svoju veľkosť na základe aktuálneho vstupu (vpravo).

Samotný proces vývoja prebiehal v dvoch hlavných fázach. Cieľom prvej fázy bolo vytvorenie verzie pre zadávanie vstupov pomocou klávesnice do textových polí. Najťažšou úlohou bola integrácia vstupného poľa do textu zadania úlohy. Realizácia tejto časti spočívala v rozdelení textu zadania otázky na jednotlivé slová, kde každé z nich je zobrazené ako osobitný widget o rozmere slova. Widgety reprezentujúce jednotlivé slová sú vložené so vstupnými poľami pre doplňované slová do objektu triedy `Wrap`, ktorý umožňuje zlepenie týchto súčastí čo najbližšie k sebe, a ak už nie je na obrazovke priestor, tak ich vloží na nový riadok. Týmto spôsobom sa vyriešilo zakomponovanie interaktívneho widgetu medzi text, no keďže prvky obsahujúce slová, ktoré boli zložené výslovne z malých písmen, mali menšiu výšku ako tie s veľkými písmenami, tak riadkovanie textu bolo nerovnomerné. Tento problém bol vyriešený stanovením uniformnej výšky pre všetky slová a vstupné polia. Ďalším problémom s ktorým bolo nutné sa vysporiadať, bolo vytvorenie vstupných polí, ktoré by sa svojou šírkou prispôsobili obsiahnutému textu. Triedy `TextField` a `TextInputField`, ktoré sú súčasťou frameworku Flutter a sú široko využívané na účel vpisovania textu, touto

možnosťou automatického prispôsobovania šírky nedisponujú. Preto bola táto vlastnosť realizovaná spôsobom, kedy je widget typu `TextFormField` zaobalený do prvku s fixnou šírkou a s každou zmenou vstupu je prepočítavaná šírka na základe vstupného textu. Vypočítanie šírky tohto textu funguje na princípe vykreslenia obsahu vstupného textu do widgetu `Container`, ktorý svoju šírku prispôsobuje obsahu. Pomocou inštancie objektu triedy `GlobalKey` je táto šírka po vykreslení získaná a následne aplikovaná na vstupné pole. Z dôvodu potreby vykresľovania textu do iného widgetu a následného aplikovania šírky je šírka poľa prispôsobená s oneskorením jedného framu, čo však pre ľudské oko nie je viditeľné. Druhá fáza predstavovala implementáciu zjednodušenej možnosti doplňovania textu a to pomocou gesta preťahovania prvkov z množiny možností priamo do textu. Postup pri realizácii bol takmer totožný s realizáciou prvej fázy s tým rozdielom, že bol uľahčený o implementáciu prepočítavania šírky vstupného poľa na základe vstupu. Na druhej strane bolo nutné programovo realizovať dopĺňanie textu slovami pomocou gesta „potiahni a pusť“.

Jedným z dôležitých zistení vo fáze testovania bolo, že pri verzii s preťahovaním možností, kde sú jednotlivé možnosti uložené v spodnej časti obrazovky (viď obr. 7.12 – vľavo a v strede), neboli užívateľmi zbadané, takže určitú chvíľu trvalo, kým sa zorientovali. Tento problém bol vyriešený pridaním tmavo modrej kontrastnej lišty s nápisom „Draggable items“. Taktiež bol pridaný prvok zobrazenia vyhodnotenia doplneného slova, kedy podržaním prsta môže užívateľ vidieť správnosť jeho odpovede. Tento prvok je možné sledovať na obrázku 7.13. Zároveň bolo nutné riešiť viditeľnosť tohto prvku a pozíciu prepočítavať vzhľadom k pozícii stlačenia prstom.

Obr. 7.13: Rýchly náhľad detailu vyhodnotenia nesprávne doplneného slova (vľavo) a správne doplneného slova(vpravo).

Pri úvahe, ktorý zo spôsobov vyplňovania textu je vhodnejšie do finálnej verzie zakomponovať, som sa rozhodol, že bude užívateľovi umožnený výber z týchto možností a to tak, že v prípade otázky bude prvotne zobrazení ťažší variant (vstup pomocou klávesnice) a v prípade, že bude mať užívateľ záujem o menej náročnú verziu, bude mu umožnené prepnutie riešenia na spôsob preťahovania možností do textu. Prepínanie medzi jednotlivými variantmi je možné vykonať pomocou tlačidla v ľavom dolnom rohu, ktoré je možné vidieť na obrázku 7.12 (vľavo a v strede).

Štruktúra vo formáte JSON

Táto časť textu sa bude venovať štruktúre definície otázky s doplňovaním textu. Definícia otázky môže byť definovaná v súbore definujúcom sadu otázok alebo v samostatnom súbore. Definícia otázky s doplňovaním textu musí nadobúdať nasledujúcu štruktúru:

```
1  {
2      "type": "filltext",
3      "case_sensitive": true,
4      "text": "Lorem ipsum ??First?? sit amet, consectetur ... ??Second??
           ... ut labore et. Lorem Ipsum ??Third??"
5  }
```

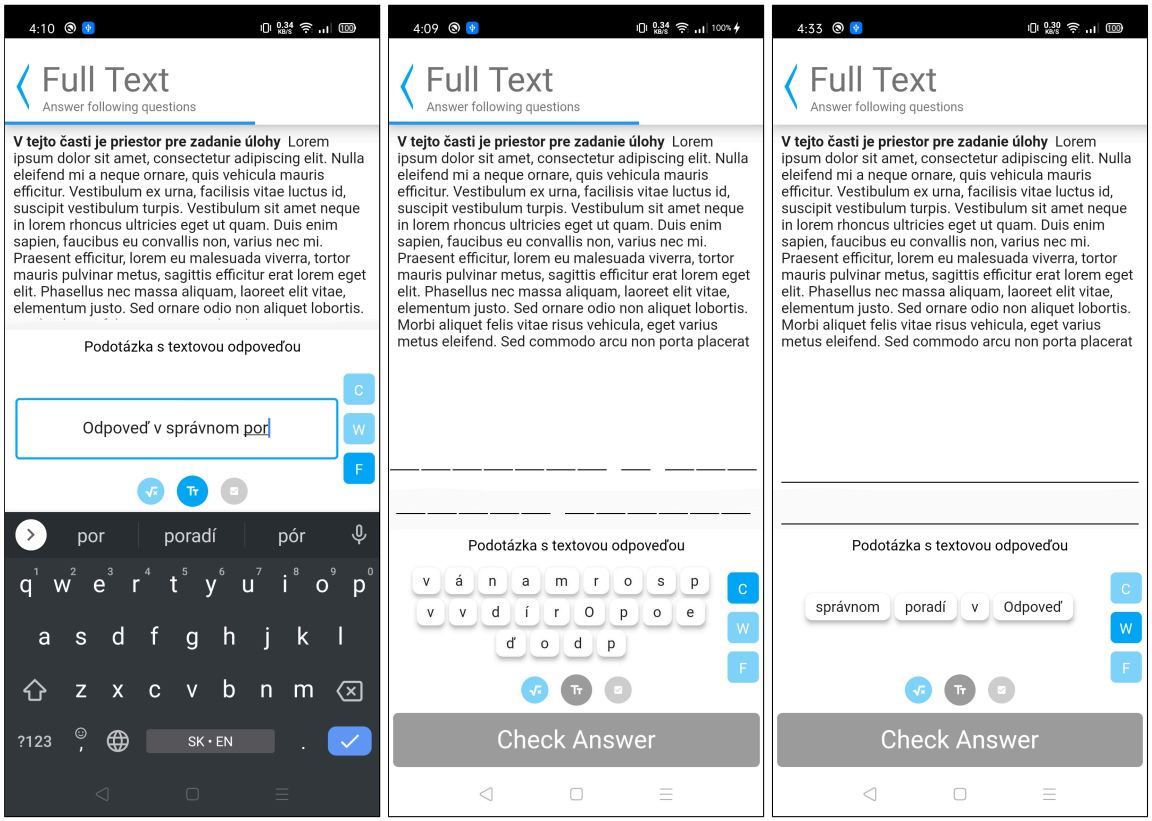
- `type` (*reťazec*) – definuje typ otázky. Pre otázku typu „Doplňovanie textu“ musí nadobúdať hodnotu „filltext“.
- `case_sensitive` (*bool*) – určuje, či je kontrola odpovedí vykonaná s ohľadom na veľkosti jednotlivých znakov.
- `text` – zadanie úlohy, kde podreťazec začínajúci prefixom „??“ a končiaci sufixom „??“ je podreťazec, ktorý má užívateľ do textu doplniť.

7.2.5 Otázka s plnou odpoveďou

Otázka s plnou odpoveďou predstavuje úlohu, kde užívateľ odpovedá na otázku vo forme textových odpovedí. Užívateľské rozhranie tohto typu otázky pozostáva zo zadania úlohy a zoznamu podotázok s priestorom na odpoveď. Tvorcovi je umožnené vytvoriť ľubovoľný počet podotázok, a to z dôvodu, že k jednému zadaniu je možné vykonať viacero čiastočných výpočtov za účelom získania konečného výsledku. A práve pre možnosť overenia správnosti aj týchto medzi výpočtov je poskytnutá táto vymoženosť. Úloha rozlišuje medzi troma typmi odpovedí, ktorými sú:

1. Číselná odpoveď
2. Textová odpoveď
3. Pravdivostná hodnota (pravda/nepravda)

Spôsob vypracovania textovej odpovede je umožnený pomocou troch rôznych vstupov, a to konkrétne zložením textovej odpovede z množiny znakov, zložením odpovede skladbou prvkov z množiny slov alebo vstupom pomocou klávesnice. Uvedené typy vstupov je možné sledovať na obrázku 7.14. Tieto módy predstavujú 3 stupne náročnosti odpovede a je iba



Obr. 7.14: Užívateľ si môže zvoliť rôzne tri vstupy pre vypracovanie textovej odpovede – klávesnica (vľavo), skladanie odpovede z množiny znakov (v strede) a skladanie odpovede z množiny slov (vpravo).

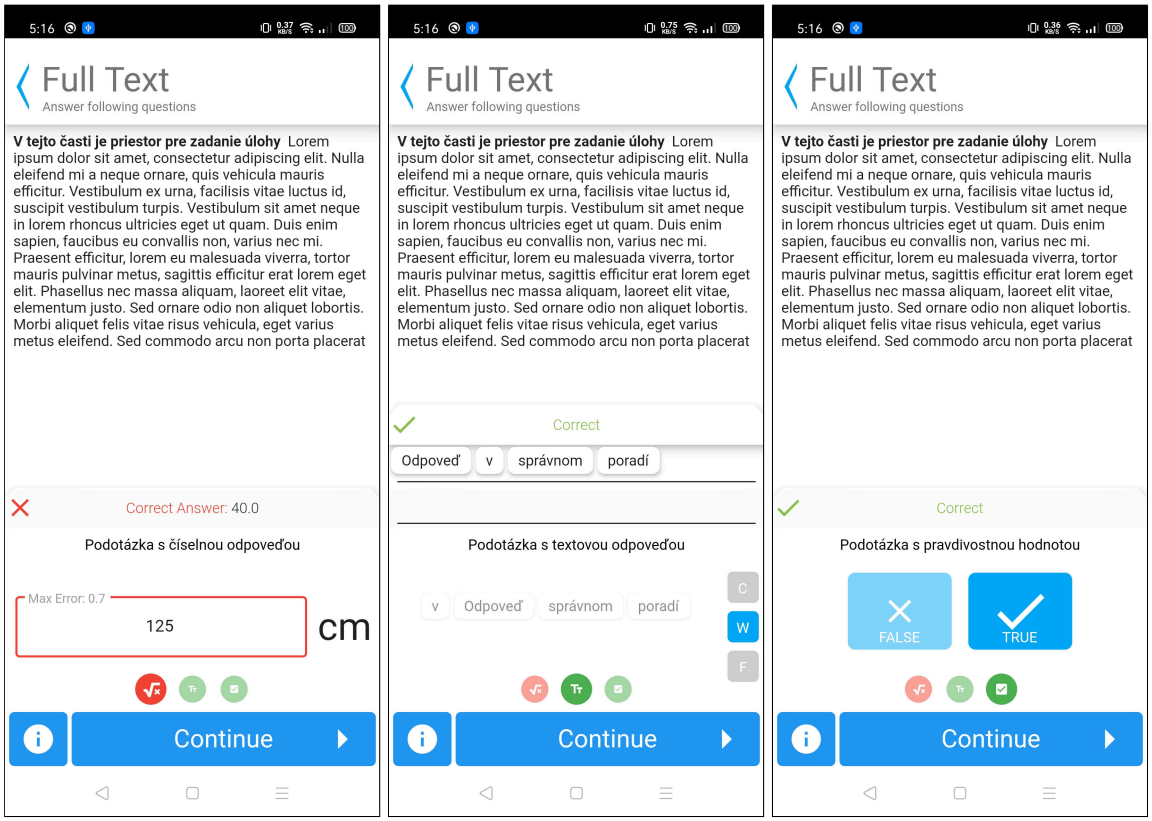
na užívateľovi, ktorú z týchto náročností si zvolí a na tvorcovi kurzu, ktorú z nich pri vypracovaní otázky povolí.

Číselné odpovede poskytujú možnosť vstupu výhradne za pomoci klávesnice a ich vstup je obmedzený iba na číselné hodnoty. Pri každej číselnej odpovedi je možné definovať jednotku, v ktorej má byť výpočet vykonaný. Je to z dôvodu, že výpočet užívateľa môže byť správny, avšak uskutočnený v iných jednotkách (napríklad teplota – Kelvin, stupne Celzia).

Pravdivostná hodnota je najjednoduchším typom odpovede, a to určením, či je daná podotázka resp. výrok pravdivý alebo nie.

V súvislosti s testovaním užívateľského rozhrania bola pridaná indikácia zoznamu podotázok v podobe kruhových objektov uložených v riadku. Každý tento prvok predstavuje jednu podotázku a mení svoje sfarbenie na základe toho, či je odpoveď danej podotázky užívateľom zadaná (modrá farba) alebo ju ešte nevyplnil (sivá farba). Taktiež slúži ako indikátor správnosti odpovede podotázky a to tak, že v prípade, ak je odpoveď na danú podotázku správna, tak je zafarbený na zeleno a v prípade, že je nesprávna, dôjde k jej sfarbeniu na červeno. So spôsobom vyhodnocovania bol taktiež pridaný prvok, ktorý v prípade zlej odpovede ukáže jej správnu odpoveď. Užívateľské rozhranie zobrazujúce aj vyhodnotenie otázky je zobrazené na obrázku 7.15. V počiatočnej verzii bolo vyobrazená iba informácia, či je odpoveď správna alebo nie, to sa však ukázalo ako nedostatočné a to hlavne pri otázkach s číselnou odpoveďou. Samotné sfarbenie dynamického tlačítka pre pokračovanie v kvíze opísaného v kapitole 7.2.1 je zmenené tak, že v prípade, ak je niektorá z podotázok

vypracovaná nesprávne, tak je zafarbená na modrú farbu a nie na červenú. Táto zmena bola vykonaná na základe zistenia, kedy prehľadávaním jednotlivých podotázok po vyhodnotení nastával stav, že odpoveď podotázky bola správna, no tlačítko bolo zafarbené na červeno, a to kvôli tomu, že iná podotázka bola zodpovedaná nepravdivo. To malo za následok, že užívateľské rozhranie bolo neprehľadné a zmätočné. Ďalšou veľmi dôležitou zmenou, čo sa týka užívateľskej skúsenosti s aplikáciou, bolo implementovanie automatického zobrazovania a skrývania klávesnice pri prechode medzi jednotlivými podotázkami, čo vyriešilo problém štandardného správania aplikácie. Týmto problémom bolo, že vždy, keď nastal prechod medzi jednotlivými podotázkami, tak sa klávesnica deaktivovala a zmizla a zároveň nebolo vstupné pole pre aktuálnu otázku aktívne. To nútilo užívateľa pri každej zmene podotázky stlačiť prstom vstupné pole, aby mu bol opätovne umožnený textový/číselný vstup.



Obr. 7.15: Užívateľské rozhranie zobrazuje vyhodnotenie jednotlivých podotázok – číselnej (vľavo), textovej (v strede) a pravdivostnej hodnoty (vpravo).

Štruktúra vo formáte JSON

```
{
    "type": "fulltext",
    "text": "Zadanie úlohy ..."
    "answers": [{
        label": "Podotázka s číselnou odpoveďou.",
        "value": [40.0],
        "max_error": 0.1,
```

```
        "units": "cm"
    },
    {
        "label": "Podotázka s textovou odpoveďou.",
        "case_sensitive": true,
        "value": ["Odpoveď v správnom poradí", "Answer" ],
        "fulltext": true,
        "characters": true,
        "words": true,
    },
    {
        "label": "Podotázka s pravdivostnou hodnotou",
        "value": true
    },
    ]
}
```

Hlavná štruktúra súboru:

- `type` (*reťazec*) – definuje typ otázky. Pre otázku typu „Otázka s plnou odpoveďou“ kľúč musí nadobúdať hodnotu „fulltext“.
- `text` (*markdown*) – určuje textové zadanie úlohy.
- `answers` *(pole objektov)* – pole objektov definujúcich podotázky.

Štruktúra objektu poľa `answers`:
Každý objekt tohto poľa definuje podotázku. Ako už bolo spomenuté v predošlom texte, tak aplikácia rozlišuje medzi tromi rôznymi typmi podotázok a to konkrétne medzi podotázkou s textovou odpoveďou, podotázkou s číselnou odpoveďou a podotázkou s pravdivostnou hodnotou (pravda/nepravda). To, o aký typ odpovede sa jedná, definuje tvorca otázky pomocou hodnoty kľúča `value`

- **Podotázka s textovou odpoveďou**
  - `label` (*reťazec*) – textové znenie podotázky.
  - `value` (*pole reťazcov*) – určuje množinu reťazcov, ktoré sú správnou odpoveďou.
  - `case_sensitive` (*bool*) – určuje, či je kontrola odpovede na podotázku vykonaná s ohľadom na veľkosti jednotlivých znakov.
  - `fulltext` (*bool*) – určuje, či je užívateľovi sprístupnená možnosť zadávania odpovede formou klávesnice a vstupného poľa.
  - `characters` (*bool*) – určuje, či je užívateľovi sprístupnená možnosť zadávania odpovede formou skladania z množiny znakov.
  - `words` (*bool*) – určuje, či je užívateľovi sprístupnená možnosť zadávania odpovede formou skladania slov.
- **Podotázka s číselnou odpoveďou**
  - `label` (*reťazec*) – textové znenie podotázky.
  - `value` (*pole čísel*) – určuje číselnú množinu hodnôt správnych odpovedí.

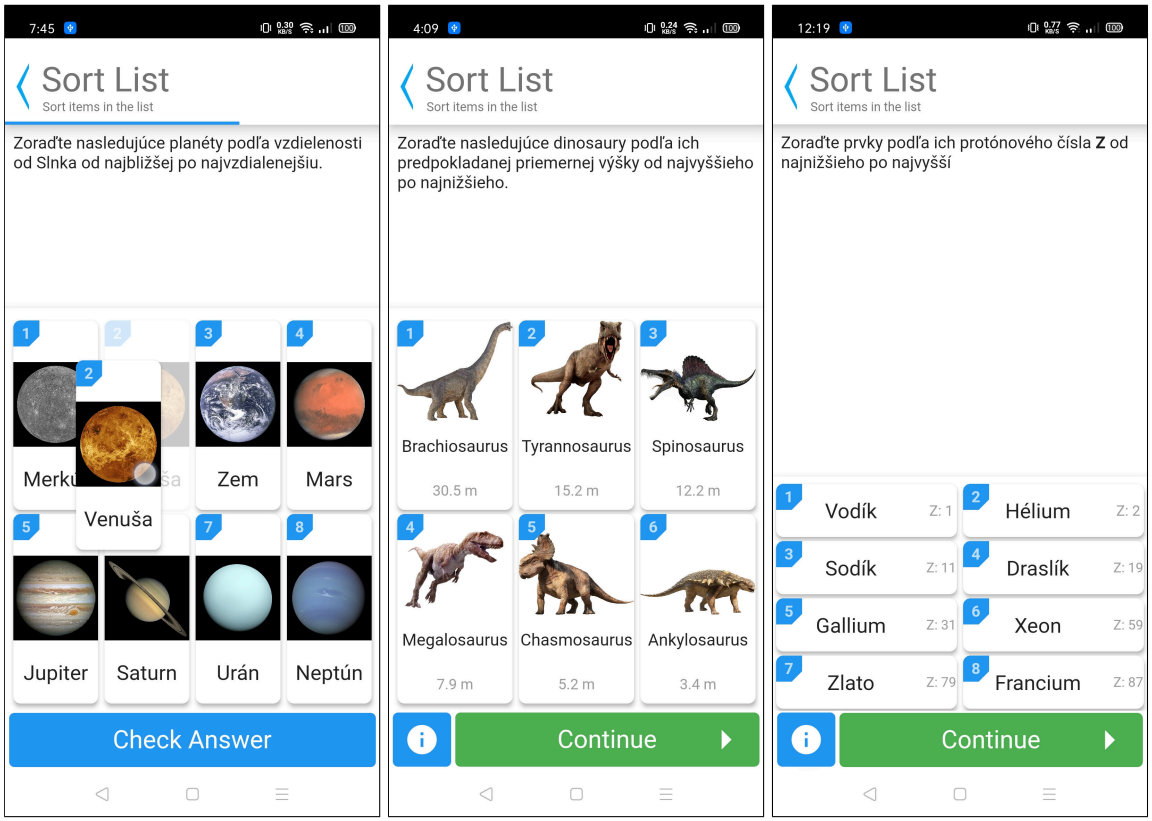
  - `units` (*reťazec*) – jednotky v ktorých má byť číselná hodnota odpovede zadaná.
  - `max_error` (*číslo*) – maximálna odchýlka, ktorú môže užívateľ výpočtom dosiahnuť oproti hodnotám definovaných kľúčom `values`.

- **Podotázka s pravdivostnou hodnotou**
  - `label` (*reťazec*) – textové znenie podotázky.
  - `value` (*bool*) – určuje pravdivostnú hodnotu správnej odpovede. Môže nadobúdať boolové hodnoty „true“ alebo „false“.

7.2.6 Zoradenie postupnosti

Precvičovanie zoradenia prvkov na základe postupnosti môže mať široké využitie v rôznych odboroch. Môže byť napríklad využitá na chronologické zoraďovanie historických udalostí, matematických konštánt podľa ich hodnoty alebo na zoraďovanie chemických prvkov na základe ich protónového čísla a množstva iného. Užívateľské rozhranie pozostáva z dvoch súčastí – prvou z nich je znenie zadania a druhou je priestor na zoraďovanie jednotlivých prvkov. Tento typ úlohy umožňuje reprezentovanie jednotlivých prvkov v podobe textu, obrázkov ale aj ich kombinácií (viď obr. 7.16). Podobne ako pri úlohe s výberom možností, tak aj v tomto prípade musel byť braný ohľad na prepočítavanie rozmerov jednotlivých prvkov vzhľadom k dostupnému priestoru na obrazovke. Pre samotnú implementáciu zoraďovania prvkov bol v prvej verzii podporujúcej iba textové prvky využitý balíček `flutter_reorderable_list`[6], ktorý poskytoval rozhranie pre zoraďovanie prvkov uložených v stĺpci. Samozrejme vzhľadom na spomenutú podporu obrázkov bol tento balíček zamenený za balíček `reorderables`[7], ktorý poskytuje rozhranie k zoraďovaniu prvkov uložených aj v inej formácii ako je stĺpec. Finálne užívateľské rozhranie, ktoré je možné vidieť na obrázku 7.16, pôvodne nedisponovalo indikovaním poradia jednotlivých prvkov. Táto zmena bola vykonaná na základe testovania užívateľského rozhrania. Pri prvotnom testovaní, v ktorom bola implementovaná podpora čisto textových prvkov, tento detail nebolo nutné do rozloženia pridávať, pretože každý užívateľ rozumel, že prvok uložený na vrchu stĺpca je prvý v poradí a prvok na spodku je v poradí posledný. Nutnosť tejto indikácie sa teda zvýraznila kvôli podpore obrázkov, kedy jednotlivé prvky nie sú zobrazené v stĺpci ale v podobe dlaždíc poskladaných vedľa seba do formátu mriežky. Pôvodný návrh počítal s tým, že užívateľovi bude jasné, že prvá dlaždica v poradí je tá na ľavom hornom okraji mriežky a poslednou v poradí dlaždica v pravom dolnom okraji mriežky. To sa v praxi ukázalo v celku pochopiteľné, avšak nastal aj prípad, kedy sa užívateľ snažil o tomto pravidle uistiť bližším dopytovaním. Z toho dôvodu teda bola pridaná indikácia poradia. Problémom pri implementácii tohto rozšírenia bol fakt, že každou zmenou poradia bolo nutné všetky prvky zoznamu prekresľovať, aby sa aktualizovala ich indikácia pozície v rámci zoradenia. Balíček `reorderables` neposkytoval funkciu pre získanie poradia prvkov pri každej zmene poradia počas preťahovania prvku. Poskytoval iba funkciu, ktorá informovala o zmene poradia až po uvoľnení vybranej dlaždice a preto bolo pre prispôsobenie potrieb aplikácie nutné upraviť zdrojový kód tohto balíčka. Samotná aktualizácia a prekresľovanie jednotlivých prvkov bola realizovaná pomocou triedy `StreamBuilder`, ktorá umožňuje prekresľovanie widgetu vždy pri zavolaní funkcie, ktorej naslúcha. V tomto prípade sa jednalo o spomínanú funkciu, ktorá bola doplnená do implementácie balíčka `reorderables`. Takisto bolo

[6] https://pub.dev/packages/flutter_reorderable_list
[7] https://pub.dev/packages/reorderables



Obr. 7.16: Zoradzovanie prvkov je realizované pomocou gesta „potiahni a pusť“ (vľavo). Po správnom zoradení sú zobrazené samotné hodnoty na základe ktorých bolo nutné prvky zoradiť (v strede). Jednotlivé prvky je možné definovať aj bez obrázkov (vpravo).

pozmenené pôvodné správanie zoraďovania dlaždíc pomocou aktivácie preraďovania dlaždice prostredníctvom dlhého podržania prsta a následného potiahnutia prvku. To sa ukázalo ako nevhodné a následne bolo toto správanie zmenené iba na podobu potiahnutia prstom bez potreby dlhého podržania. Posledným dôležitým míľnikom bolo pridanie čiastočných odpovedí zobrazovaných na každej dlaždici po vyhodnotení otázky, ktoré ukazujú hodnoty na základe ktorých mali byť jednotlivé prvky zoradené (viď obr. 7.16 – v strede a vpravo).

Štruktúra vo formáte JSON

Táto časť textu sa bude venovať štruktúre definície otázky zoraďovania postupnosti. Definícia otázky môže byť definovaná v súbore definujúcom sadu otázok alebo v samostatnom súbore. Definícia tohto typu otázky musí nadobúdať nasledujúcu štruktúru:

```
{
    "type": "sort",
    "text": "Zoraďte dinosaury podľa výšky od najvyššieho po najnižšieho.",
    "sort": [
        {
            "text": "Brachiosaurus",
            "partial": "30.5 m",
            "visual": "@./brachiosaurus.png",
```

```
            "sound": "@./sound.mp3"
        },
        {
            "text": "Tyrannosaurus",
            "partial": "15.2 m",
            "visual": "@./tyrannosaurus.png",
            "sound": "@./sound2.mp3"
        },
        ...
    ]
},
```

Hlavná štruktúra súboru:

- `type` (*reťazec*) – definuje typ otázky. Pre otázku typu „Zoradenie postupnosti“ musí nadobúdať hodnotu „sort“.
- `text` (*markdown*) – určuje textové zadanie úlohy.
- `sort` (*pole objektov*) – pole definujúce jednotlivé objekty, ktoré majú byť zoradené. Kontrola správnosti zoradenia funguje na základe zoradenia jednotlivých prvkov v tomto poli, nie na základe ich hodnôt.

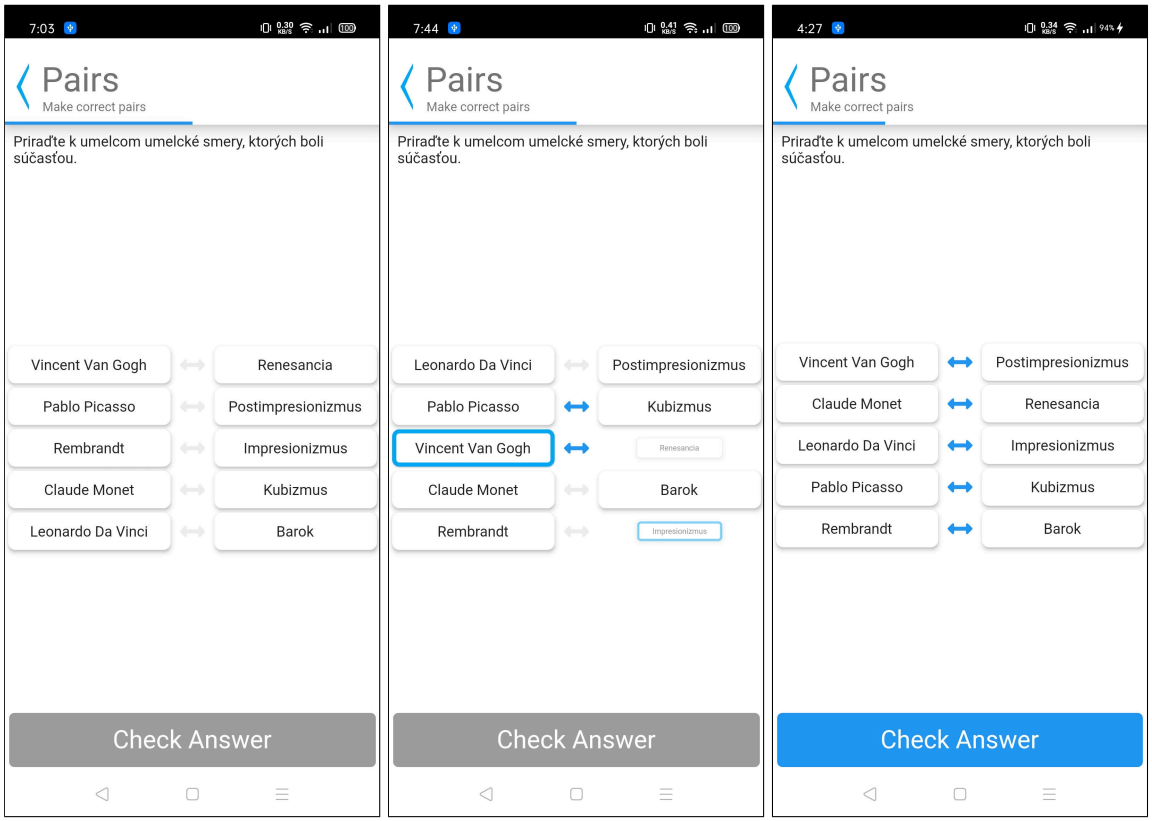
Štruktúra objektu poľa `sort`:
Tieto objekty definujú jednotlivé dlaždice, ktoré majú byť užívateľom zoradené.

- `text` (*reťazec*) – textový nápis alebo matematický vzorec zobrazený na dlaždici.
- `partial` (*reťazec*) – čiastočná odpoveď odhalená po vyhodnotení otázky.
- `visual` (*referencia*) – odkaz na obrázkový súbor, ktorého obsah je vyobrazený na dlaždici.
- `sound` (*referencia*) – odkaz na zvukový súbor, ktorého obsah je spustený pri stlačení dlaždice prstom.

7.2.7 Spájanie dvojíc

Tento typ otázky umožňuje precvičovať znalosti pomocou spájania dvojíc, ktoré sú v určitej relácii. Logiku relácií popisuje tvorca popisom zadania, pričom úlohou užívateľa je tieto prvky na základe definovanej relácie spojiť do príslušných dvojíc. Užívateľské rozhranie pozostáva z textu zadania a samotnej časti vypracovania, ktorá obsahuje dva stĺpce. Každý z týchto stĺpcov obsahuje práve jeden prvok z dvojice. Medzi prvkami oboch stĺpcov sa nachádza dvojsmerná šípka, ktorá indikuje farbou spojenie (modrá) alebo nespojenie (sivá) dvojíc. Zároveň nie je užívateľovi umožnená kontrola otázky, kým nie sú vytvorené spojenia medzi všetkým dvojicami prvkov. Pre lepšiu predstavu je užívateľské rozhranie zobrazené na obrázku 7.17.

Pri návrhu boli vytvorené dve verzie užívateľského rozhrania, kde v prvej z nich bolo realizované spájanie prvkov pomocou gesta „potiahni a pusť“ a v druhej pomocou jednotlivého označovania prvkov stlačením vzájomne prináležiacich dvojíc. Druhá verzia sa ukázala ako vhodnejšia, no na základe testovania bola navyše rozšírená o prvok možnosť zrušenia prepojenia jednotlivých dvojíc stlačením šípky, čo umožňovalo iný spôsob spájania dvojíc a



Obr. 7.17: Rozloženie obrazovky otázky spájania dvojíc (vľavo). Zámena prvkov v stĺpci je vykonaná animáciou (v strede). Užívateľské rozhranie umožní kontrolu odpovedí, až vtedy, ak sú vytvorené spojenia medzi všetkými dvojicami (vpravo).

rozpájanie už existujúcich spojení medzi jednotlivými prvkami. V pôvodnej verzii bola tiež vykonaná zámena dvoch prvkov a ich spojenie hneď na následujúci snímok obrazovky. To spôsobovalo situácie, kedy užívateľ túto zmenu nezaregistroval. Preto bola následne implementovaná zámena prvkov vrámci jedného stĺpca pomocou animácie. Pokus o čo najlepšie vyobrazenie tejto animácie pomocou obrázku možno vidieť na obrázku 7.17 (v strede). Zároveň bola pridaná funkcionalita, kedy sa pri spájaní dvojíc automaticky doplní posledné spojenie v prípade, že užívateľ vytvorí spojenie medzi predposlednou nespojenou dvojicou prvkov.

Štruktúra vo formáte JSON

Táto časť textu sa bude venovať štruktúre definície otázky spájania dvojíc. Definícia otázky môže byť definovaná v súbore definujúcom sadu otázok alebo v samostatnom súbore. Definícia tohto typu otázky musí nadobúdať nasledujúcu štruktúru:

```
{
    "type": "pairs",
    "text": "Priraďte k umelcom umelecké smery, ktorých boli súčasťou.",
    "pairs": [
        [
            {"text": "Pablo Picasso", "sound":"@./sound.mp3" },
```

```
            {"text": "Kubizmus", "sound":"@./sound2.mp3"}
        ],
        [
            {"text": "Leonardo Da Vinci"},
            {"text": "Impresionizmus"}
        ],
        ...
        ]
    ]
}
```

Hlavná štruktúra súboru:

- `type` (*reťazec*) – definuje typ otázky. Pre otázku typu „Spájanie dvojíc“ kľúč musí nadobúdať hodnotu „pairs“.
- `text` (*markdown*) – určuje textové zadanie úlohy.
- `answer_detail` (*markdown*) – detailné vysvetlenie správnej odpovede, ktoré je dostupné po vyhodnotení otázky.
- `sort` *(pole polí objektov)* – pole polí obsahujúcich dvojice objektov, ktoré sú považované za pár.

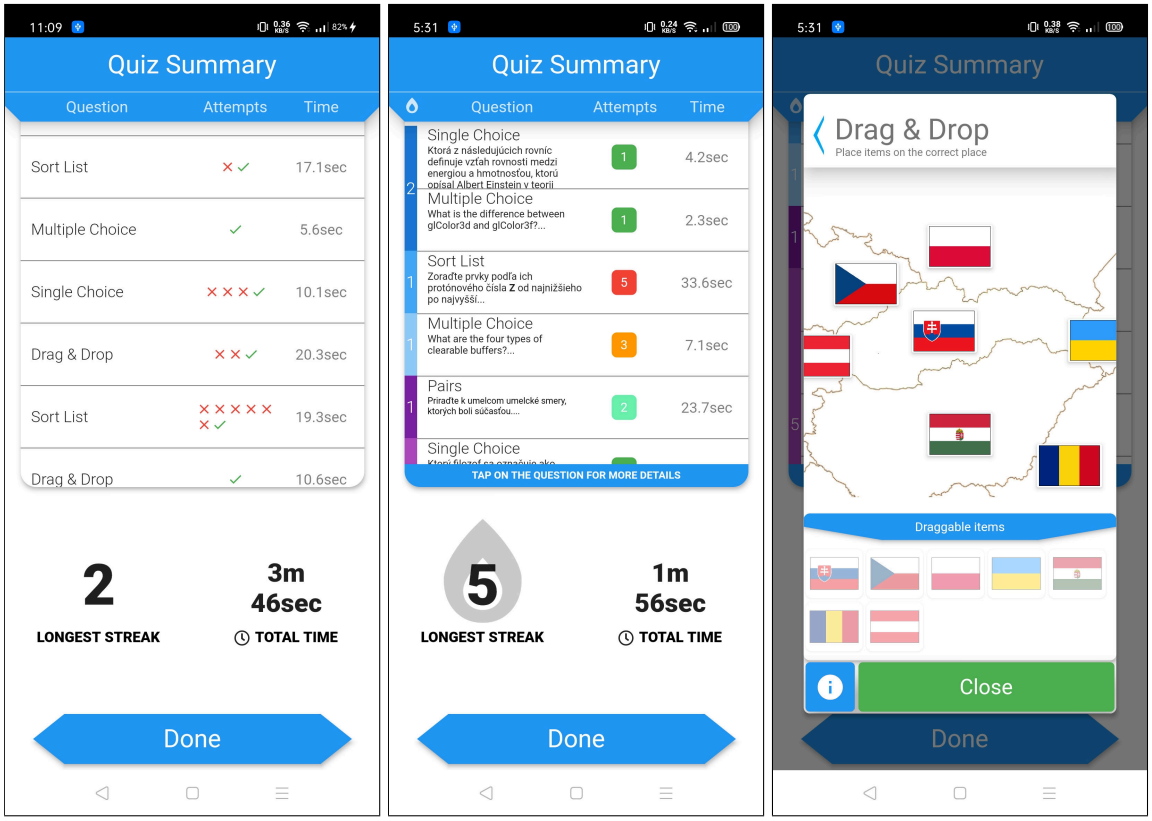
Štruktúra objektu poľa v poli `pairs`:
Každý takýto objekt definuje užívateľské rozhranie jedného prvku zo vzájomne prislúchajúcich dvojíc.

- `text` (*reťazec*) – textový nápis alebo matematický vzorec zobrazený na prvku reprezentujúcom daný objekt.
- `sound` (*referencia*) – odkaz na zvukový súbor, ktorého obsah je spustený pri stlačení prvku prstom.

7.2.8 Vyhodnotenie kvízu

Po vypracovaní všetkých otázok zo série je užívateľovi zobrazená stránka vyhodnotenia otázok. Táto obrazovka pozostáva z tabuľky, ktorá umožňuje náhľad ku každej zo zodpovedaných otázok a zobrazenie odpovedí užívateľa (umožňuje mu spätne vyobrazenie chýb) a zároveň zobrazuje štatistiky k jednotlivým otázkam vo forme počtu neúspešných pokusov užívateľa pri vypracovaní otázky a času, ktorý strávil riešením danej otázky (viď obr. 7.18). Na obrazovke sú zobrazené aj štatistiky k celému priebehu kvízu ako je celkový čas vypracovania otázok a najdlhší počet správne zodpovedaných otázok v poradí. Čo sa týka postupného vývoja užívateľského rozhrania tejto časti aplikácie, tak hlavné rozloženie nepodstúpilo veľkým zmenám. Niektoré prvky však zmenené byť museli. Konkrétne bola zmenená reprezentácia jednotlivých riadkov danej tabuľky, kde každému riadku tabuľky prislúcha práve jedná otázka. Riadok pozostáva z typu otázky a zobrazenia zadania, počtu pokusov a času riešenia. Prvá verzia na rozdiel od finálnej verzie neobsahovala zadanie otázky ale iba typ otázky, kedy bolo pre užívateľa problematické nájsť spätne chcenú otázku v prípade, že sa v precvičovanej sérii nachádzalo viac otázok rovnakého typu. Taktiež bola rozličná reprezentácia počtu neúspešných pokusov riešenia danej otázky, kedy boli realizované za pomoci ikoniek v podobe červených krížikov o počte neúspešných pokusov a zelených

fajok. Tento spôsob zobrazovania počtu pokusov sa ukázal ako nevhodný a preto bola vykonaná zmena do podoby, kedy je počet pokusov znázornený číslom na farebnom pozadí. Posledným prvkom zmeny bol fakt, že užívateľ nevedel o funkcionalite, kedy boli jednotlivé riadky tabuľky interaktívne a pomocou stlačenia riadku prstom bolo možné zobrazenie riešenia užívateľa. Preto bol na spodok tabuľky pridaný text „TAP ON THE QUESTION FOR MORE DETAILS“. Tieto zmeny možno pozorovať na obrázku 7.18 (vľavo a v strede).



Obr. 7.18: Staršia verzia užívateľského rozhrania pre vyhodnotenie kvízu (vľavo). Finálna verzia užívateľského rozhrania pre vyhodnotenie kvízu (v strede). Spätný náhľad na vybranú vypracovanú otázku (vpravo).

Kapitola 8

Testovanie

Proces kreovania aplikácie diplomovej práce pozostával z opakovaného testovania jednotlivých verzií užívateľského rozhrania, na základe ktorého následne vznikali nové verzie. To viedlo až ku sformovaniu prezentovaného konečného riešenia. Predchádzajúce kapitoly tejto práce vo svojich sekciách už obsahujú bližší opis vývoja a testovania jednotlivých častí výslednej aplikácie a to z dôvodu lepšieho priblíženia iteratívneho spôsobu vývoja tejto aplikácie. Preto táto kapitola neobsahuje úplný, ale len stručný popis testovacieho procesu, ktorý svojim obsahom dopĺňa informácie, ktoré už boli uvedené v predchádzajúcich kapitolách.

Testovanie bolo vykonávane na siedmich testovacích subjektoch. Samotný spôsob testovania mobilnej aplikácie bol vykonaný v dvoch formách. Prvá forma testovania spočívala v osobnom sledovaní konkrétneho správania testovacieho subjektu pri práci s aplikáciou a vykonávaní jednotlivých zadaných úkonov. Druhá forma testovania aplikácie bola vykonávaná dištančne, a to pomocou distribúcie aplikácie pomocou služby Google Play a následného videohovoru, zdieľania obrazovky smartfónu a konečného verbálneho hodnotenia či prípadných pripomienok.

Podpora platforiem

Jednou z požiadaviek na túto aplikáciu bola aj požiadavka na zabezpečenie jej multiplatformnosti. V procese vývoja a implementácie bola táto požiadavka riadne zohľadnená. Avšak z dôvodu absencie prístupu k zariadeniam, ktoré disponujú operačným systémom iOS, nebolo možné zabezpečiť aj overenie správnosti jej fungovania formou individuálneho testovania na týchto zariadeniach. No aj napriek tomuto nedostatku by predmetná aplikácia mala na základe konečnej implementácie riadne fungovať aj na zariadeniach so systémom iOS. Aplikácia bola teda vzhľadom na vyššie uvedené testovaná výlučne na zariadeniach s operačným systémom Android, a to konkrétne:

- Realme 5 Pro (API level 29)
- Xiaomi A2 (API level 29)
- Samsung J5 (API level 29)
- Asus ZenFone 2 (API level 28)
- Xiaomi Redmi 8 (API level 28)
- Xiaomi Redmi Note 4 (API level 24)

- Huawei P20 Lite (API level 23)
- Lenovo A536 (API level 22) – zariadenie vykazovalo problém s generovaním matematických a chemických vzorcov. Dôvodom bolo využitie funkcií balíčka `flutter_webview_plus`, ktorý nepodporoval túto verziu API.
- Huawei G700 (API level 17) – zariadenie vykazovalo rovnaký problém ako zariadenie Lenovo A536.

Testovanie na jednotlivých zariadeniach zároveň ukázalo, že aplikácia správne funguje na zariadeniach s vyšším aplikačným rozhraním operačného systému Android ako je verzia 23 (vrátane). Aplikácia by teda mala byť kompatibilná s približne 84,9% [39] zariadení využívajúcimi operačný systém Android.

Kapitola 9

Záver

Cieľom práce bolo vytvoriť mobilnú aplikáciu pre prezentovanie, študovanie a precvičovanie znalostí. Na samom počiatku boli analyzované už existujúce alebo podobné riešenia, rôzne prístupy a technológie pre vývoj natívnych a multiplatformových mobilných aplikácií. Boli naštudované zásady dobrého návrhu užívateľského rozhrania a princípy agilného prístupu k vývoju softvéru, ktorý bol nosným prvkom celkového vývoja aplikácie tejto diplomovej práce.

Na základe tejto analýzy a získaných vedomostí bola následne vytvorená základná štruktúra pre prezentovanie a precvičovanie znalostí. Zároveň bolo vytvorené užívateľské rozhranie, ktoré jednotlivé časti štruktúry reprezentovalo. Postupným testovaním, vytváraním výukových materiálov a následným vytváraním nových návrhov užívateľského rozhrania aplikácie, spoločne s pridávaním nových vlastností (ako napríklad podpora vykresľovania matematických a chemických vzorcov, podpora obrázkov či prehrávanie zvukových stôp a podobne) boli vytvárané nové verzie. Tie boli následne na základe vyššie spomínaného testovania iteratívne zlepšované, čo v konečnom dôsledku viedlo k vytvoreniu finálnej verzie aplikácie. Finálna verzia aplikácie je zároveň zverejnená na digitálnej distribučnej platforme Google Play.

Konečná štruktúra pre prezentovanie a precvičovanie znalostí má podobu kurzu rozdeleného do jednotlivých sekcií, lekcií a slovníka dôležitých pojmov. Lekcie sú následne rozdelené na teoretické a praktické časti. Teoretická časť lekcie predstavuje jadro prezentovaných znalostí v kurze. Preto musel byť v počiatku vyriešený problém štrukturovania textu a zakomponovania podpory matematických a chemických vzorcov do textu pre možnosť adekvátneho prezentovania príslušných vedomostí z matematických, chemických, či iných technických, prírodovedeckých alebo humanitnovedných odborov. Praktická časť lekcie je vykonávaná vo forme kvízu tvoreného rozličnými druhmi otázok (doplňovanie do textu, výber jednej alebo viacerých možností, doplňovanie do obrázka, otázka s plnou odpoveďou, spájanie dvojíc a zoraďovanie postupnosti).

Pre demonštrovanie širokého využitia aplikácie a spôsobu, akým štruktúruje výukové materiály boli vytvorené ukážkové kurzy z rôznych odvetví akými sú história, biológia či informačné technológie spolu so sériou praktických otázok na precvičovanie prezentovaných znalostí.

Pokračovanie projektu

Aktuálny spôsob tvorby kurzu a jeho súčastí je realizovaný formou vytvárania súborov vo formáte JSON, na základe postupov popísaných v predchadzajúcom texte. Jediným spôso-

bom, akým môže tvorca skontrolovať aktuálnu podobu vytváraného kurzu, je uloženie jeho zdrojových súborov na webový server, ktoré musí následne stiahnúť a zobraziť prostredníctvom mobilnej aplikácie. Tento spôsob vytvárania kurzov môže byť v niektorých prípadoch časovo náročný. Preto jedným z návrhov na vylepšenie je realizácia editora, ktorý by umožňoval aktuálny náhľad na vytváraný kurz už počas jeho tvorby. Využitie frameworku Flutter v tomto projekte považujem za výhodné riešenie z hľadiska predpokladaného vývoja uvedeného editoru. Pri jeho implementácii tak budú môcť byť využité už hotové časti kódu, ktoré sú súčasťou vytvorenej aplikácie a to bez ohľadu na cieľovú platformu.

Ďalším vylepšením by mohlo byť vytváranie nových typov otázok umožňujúcich iné, zatiaľ nepodporované spôsoby precvičovania novonadobudnutých vedomostí. Mohlo by sa jednať napríklad o úlohu, kde by bola aplikácia schopná rozoznávať na základe rukopisu užívateľa rôzne symboly. Pre lepšie priblíženie, užívateľ nakreslí písmeno ruskej azbuky a aplikácia overí, či je daný znak napísaný v správnej podobe. V budúcnosti by mohla byť implementovaná otázka zameraná na kontrolu správnej výslovnosti, ktorá by na pozadí využívala algoritmy pre spracovanie reči. Obdobnú funkcionalitu dnes už poskytuje aj aplikácia Duolingo.

V samotnom závere predikcie budúceho pokračovania daného projektu a zlepšovania jeho obsahu nie je možné opomínať ani základnú úlohu aplikácie, a to práve sprostredkovanie informácií a výučbového materiálu bežným užívateľom. Preto je nutné, za účelom zvyšovania celkovej kvality mobilnej aplikácie, sa zamerať nielen na zlepšovanie užívateľského rozhrania a vytváranie nových typov kvízových otázok, ale taktiež aj na jej obsahovú stránku a kvantitu (samozrejme v súbehu s kvalitou) vytvorených kurzov. Len v tom prípade, ak sa aplikácia dostane do povedomia a obľuby spotrebiteľov na trhu digitálnych služieb, bude možné viesť ďalšiu adekvátnu diskusiu o potenciálnych zlepšeniach, ktoré budú reakciou na samotné požiadavky jednotlivých užívateľov.

Literatúra

[1] Akshat Paul, A. N. *React Native for Mobile Development: Harness the Power of React Native to Create Stunning iOS and Android Applications*. 2. vyd. Apress, jún 2019. ISBN 978-1484244531.

[2] Alessandria, S. *Flutter Projects: A practical, project-based guide to building real-world cross-platform mobile applications and games*. 1. vyd. Packt Publishing, apríl 2020. ISBN 9781838642532.

[3] Apple Inc.. *Swift | Apple Developer Documentation* [online]. Apple Inc. [cit. 2020-01-12]. Dostupné z: https://developer.apple.com/swift/.

[4] Apple Inc.. *Using Objective-C Runtime Features in Swift* [online]. Apple Inc. [cit. 2020-01-12]. Dostupné z: https://developer.apple.com/documentation/swift/using_objective-c_runtime_features_in_swift.

[5] Apple Inc.. *What's New in the iOS SDK – Apple Developer* [online]. Apple Inc. [cit. 2020-01-12]. Dostupné z: https://developer.apple.com/ios/whats-new.

[6] Apple Inc.. *Xcode | Apple Developer Documentation* [online]. Apple Inc. [cit. 2020-01-12]. Dostupné z: https://developer.apple.com/documentation/xcode.

[7] Armour, B. *5 Key Benefits of Native Mobile App Development* [online]. ClearBridgeMobile.com, august 2018 [cit. 2020-01-13]. Dostupné z: https://clearbridgemobile.com/benefits-of-native-mobile-app-development/.

[8] Beck, K., Beedle, M., Bennekum, A. van, Cockburn, A., Cunningham, W. et al. *Manifesto for Agile Software Development* [online]. 2001 [cit. 2020-01-12]. Dostupné z: https://agilemanifesto.org/.

[9] Biessek, A. *Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter and Dart 2*. Packt Publishing, september 2019. ISBN 978-1788996082.

[10] Cao, N. *Why I move to Flutter* [online]. A Medium Corporation, marec 2018 [cit. 2020-01-10]. Dostupné z: https://medium.com/@nhancv/why-i-move-to-flutter-34c4005b96ef.

[11] Clow, M. *Learn Google Flutter Fast: 65 Example Apps*. Publikované nezávisle, marec 2019. ISBN 9781092297370.

[12] Dart. *The Dart type system* [online]. [cit. 2020-01-13]. Dostupné z: https://dart.dev/guides/language/sound-dart.

[13] Dossey, A. *A Guide to Mobile App Development: Web vs. Native vs. Hybrid* [online]. ClearBridgeMobile.com, jún 2019 [cit. 2020-01-13]. Dostupné z: https://clearbridgemobile.com/mobile-app-development-native-vs-web-vs-hybrid.

[14] Flutter. *Animations overview* [online]. [cit. 2020-01-13]. Dostupné z: flutter.dev/docs/development/ui/animations/overview.

[15] Flutter. *Dart:isolate library* [online]. [cit. 2020-02-30]. Dostupné z: https://api.flutter.dev/flutter/dart-isolate/dart-isolate-library.html.

[16] Flutter. *Flutter SDK releases* [online]. [cit. 2020-01-13]. Dostupné z: https://flutter.dev/docs/development/tools/sdk/releases?tab=windows.

[17] Flutter. *InheritedWidget class* [online]. [cit. 2020-01-13]. Dostupné z: https://api.flutter.dev/flutter/widgets/InheritedWidget-class.html.

[18] Flutter. *Introduction to declarative UI* [online]. [cit. 2020-01-13]. Dostupné z: https://flutter.dev/docs/get-started/flutter-for/declarative.

[19] Flutter. *MacOS install* [online]. [cit. 2020-01-13]. Dostupné z: https://flutter.dev/docs/get-started/install/macos.

[20] Flutter. *Set up an editor* [online]. [cit. 2020-01-13]. Dostupné z: https://flutter.dev/docs/get-started/editor.

[21] Flutter. *StatefulWidget class* [online]. [cit. 2020-01-13]. Dostupné z: https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html.

[22] Flutter. *Store key-value data on disk* [online]. [cit. 2020-02-30]. Dostupné z: https://flutter.dev/docs/cookbook/persistence/key-value.

[23] Flutter. *Taps, drags, and other gestures* [online]. [cit. 2020-01-13]. Dostupné z: https://flutter.dev/docs/development/ui/advanced/gestures.

[24] Flutter. *Technical Overview* [online]. [cit. 2020-01-13]. Dostupné z: https://flutter.dev/docs/resources/technical-overview.

[25] Flutter. *Testing Flutter apps* [online]. [cit. 2020-01-13]. Dostupné z: https://flutter.dev/docs/testing.

[26] Flutter. *Writing custom platform-specific code* [online]. [cit. 2020-01-13]. Dostupné z: https://flutter.dev/docs/development/platform-integration/platform-channels.

[27] Flutter. *Flutter Engine* [online]. GitHub, 2020. Dostupné z: https://github.com/flutter/engine.

[28] Gartner Inc.. *Mobile OS Market Share* [online]. Statista, jún 2019 [cit. 2020-01-07]. Dostupné z: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems.

[29] Google Inc. a Open Handset Alliance n.d.. *Platform Architecture*. [cit. 2020-01-04]. Dostupné z: https://developer.android.com/guide/platform.

[30] Griffith, C. *Mobile app development with Ionic, revised edition: Cross-platform apps with Ionic, Angular and Cordova*. 1. vyd. O'Reilly Media, 2017. ISBN 978-1-491-99812-0.

[31] Hermes, D. *Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals*. 1. vyd. Apress, jún 2015. ISBN 978-1484202159.

[32] Isaacson, W. *Steve Jobs*. New York, NY: Simon & Schuster, 2011. ISBN 978-1-4516-4853-9.

[33] Khan Academy. *About* [online]. 2020 [cit. 2020-1-7]. Dostupné z: https://www.khanacademy.org/about.

[34] King, B. *Is Android Really Open Source? And Does It Even Matter?* [online]. MakeUseOf, marec 2016 [cit. 2020-01-12]. Dostupné z: https://www.makeuseof.com/tag/android-really-open-source-matter/.

[35] Krug, S. *Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability Problems*. 3. vyd. Berkeley, California: New Riders, 2010. ISBN 978-0-321-65729-9.

[36] Krug, S. *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability*. 3. vyd. San Francisco: New Riders, 2014. ISBN 9780321965516.

[37] Lardinois, F. *Duolingo hires its first chief marketing officer as active user numbers stagnate but revenue grows* [online]. TechCrunch, august 2018 [cit. 2019-1-7]. Dostupné z: https://techcrunch.com/2018/08/01/duolingo-hires-its-first-chief-marketing-officer-as-active-user-numbers-stagnate/.

[38] Lindgaard, G., Fernandes, G., Dudek, C. a Brown, J. Attention web designers: You have 50 milliseconds to make a good first impression! *Behaviour & Information Technology*. Taylor & Francis. 2006, roč. 25, č. 2, s. 115–126. Dostupné z: https://doi.org/10.1080/01449290500330448.

[39] Mishaal Rahman. *Android Version Distribution statistics will now only be available in Android Studio* [online]. XDA Developers, apríl 2020 [cit. 2020-05-27]. Dostupné z: https://www.xda-developers.com/android-version-distribution-statistics-android-studio.

[40] Napoli, M. *Beginning Flutter: A Hands On Guide to App Development*. Indianapolis, Indiana: John Wiley & Sons, Inc., 2019. ISBN 978-1-119-55082-2.

[41] Novick, V. *React Native - Building Mobile Apps with JavaScript: Build real-world iOS and Android native apps with JavaScript*. 2. vyd. Packt Publishing, august 2017. ISBN 978-1787282537.

[42] Ostrander, J. *Android UI Fundamentals: Develop and Design*. 1. vyd. Peachpit Press, február 2012. ISBN 9780132929035.

[43] Singh, R. *Mobile Development Approaches and Flutter Architecture: FLUTTER PART-I* [online]. A Medium Corporation, jún 2019 [cit. 2020-01-10]. Dostupné z: https://medium.com/gradeup/mobile-development-approaches-and-flutter-architecture-flutter-part-i-a7e08838c97a.

[44] Stober, T. a Hansmann, U. *Agile Software Development: Best Practices for Large Software Development Projects*. 1. vyd. Springer, 2009. ISBN 978-3-540-70830-8.

[45] The Apache Software Foundation. *Apache Cordova* [online]. [cit. 2020-01-13]. Dostupné z: https://cordova.apache.org/.

[46] Toporov, E. *IntelliJ IDEA is the base for Android Studio, the new IDE for Android developers* [online]. JetBrains, máj 2013 [cit. 2020-01-12]. Dostupné z: https://blog.jetbrains.com/blog/2013/05/15/intellij-idea-is-the-base-for-android-studio-the-new-ide-for-android-developers/.

[47] Wen, D. *Getting to know Flutter*. [cit. 2020-01-09]. Preložené pomocou Google Translate. Dostupné z: https://book.flutterchina.club/chapter1/flutter_intro.html.

[48] Wikipedia Contributors. *Help:Wikitext – Wikipedia, The Free Encyclopedia* [online]. Január 2020 [cit. 2020-1-5]. Dostupné z: https://en.wikipedia.org/wiki/Help:Wikitext.

[49] Zammetti, F. *Practical Flutter: Improve your Mobile Development with Google's Latest Open-Source SDK*. 1. vyd. Apress, 2019. ISBN 978-1484249710.

Príloha A

DVD

Priložené DVD obsahuje zdrojové súbory tohto textu a systému, ktorý bol implementovaný pre účely tejto práce. Priložené DVD má nasledujúcu štruktúru:

- `src` – zložka obsahuje zdrojové súbory textu práce, vytvorenej aplikácie a ukážkových kurzov.
- `plagat.png` – digitálna verzia propagačného plagátu.
- `video.mp4` – prezentačné video aplikácie.

Príloha B

Plagát

Súčasťou odovzdávanej práce je vytlačený propagačný plagát o veľkosti A2. Digitálnu formu plagátu je možné nájsť na priloženom pamäťového médiu v súbore s názvom `plagat.png`.

Príloha C

Video

Video je umiestnené v hlavnej koreňovej zložke pamäťového média ako súbor s názvom `video.mp4`.