FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Educhild – rodičovská část Android aplikace
Student:	Daniel Matoušek
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2021/2022

Pokyny pro vypracování

Cílem této práce je softwarový návrh a následná implementace rodičovského módu aplikace Educhild, která slouží jako pomůcka pro rodiče k motivaci plnění edukačních úkolů a limitaci jiných aktivit dítěte na mobilním zařízení s OS Android.

Postupujte v těchto krocích:
• Analyzujte požadavky potřebné pro cílové uživatele a stávající konkurenční řešení. Spolupracujte s Marekem Trzaskalikem, který řeší samotné uživatelské rozhraní a grafickou podobu.
• Na základě analýzy navrhněte vhodné funkcionality rodičovské části mobilní aplikace (nastavení, statistiky, odměny, časový rozvrh apod.)
• Proveďte vhodný a kompletní softwarový návrh rodičovské části Android aplikace.
• Implementujte navrženou část aplikace, neopomeňte řádně testovat.
• Proveďte vlastní hodnocení funkčnosti a použitelnosti implementované části aplikace včetně návrhu možných vylepšení.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 16. ledna 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Educhild – rodičovská část Android aplikace

Daniel Matoušek

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Hunka

12. května 2021

Poděkování

Na tomto místě bych rád poděkoval vedoucímu bakalářské práce Ing. Jiřímu Hunkovi za odborné vedení a cenné rady poskytované v průběhu psaní práce. Děkuji také ostatním členům týmu podílejícího se na projektu Educhild za spolupráci. V neposlední řadě chci poděkovat také rodině za trpělivost a podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 12. května 2021 . . . . . .

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2021 Daniel Matoušek. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.

Odkaz na tuto práci

Matoušek, Daniel. *Educhild – rodičovská část Android aplikace*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Obsahem této bakalářské práce je popis kompletního procesu vývoje rodičovského módu aplikace Educhild určené pro operační systém Android. Aplikace umožňuje rodičům v průběhu využívání mobilního zařízení dětmi zobrazovat naučné kvízy, případně omezit celkový čas, který děti na mobilním zařízení tráví. V rodičovském módu provádí rodič veškeré nastavení aplikace a zobrazuje statistiky o používání zařízení a úspěšnosti v kvízech. Aplikace je psána v programovacím jazyku Kotlin.

Výsledkem práce je funkční prototyp aplikace Educhild včetně rodičovského módu. Bakalářská práce představuje osnovu, podle které je možné vyvíjet aplikace na operační systém Android.

Klíčová slova mobilní aplikace, Android, rodičovský mód, vzdělávání dětí, Kotlin, Android Studio, Google Firebase

Abstract

The content of this bachelor thesis is a description of the complete process of developing the parent mode of the application Educhild designed for the Android operating system. The application allows parents to display educational quizzes while children use the mobile device, or to limit the total time children spend on a mobile device. In parent mode, the parent performs all application settings and displays statistics on device usage and success in quizzes. The application is written in the programming language Kotlin.

The result of the thesis is a functional prototype of the application Educhild, including parent mode. The bachelor thesis presents an outline according to which it is possible to develop applications for the Android operating system.

Keywords mobile application, Android, parent mode, education of children, Kotlin, Android Studio, Google Firebase

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Používání mobilních zařízení se stalo nedílnou součástí našeho života a velmi nám ho v určitých ohledech zjednodušuje. Stále více také přibývá dětí, které mají svůj vlastní chytrý telefon, nebo jim ho rodiče půjčují, aby na něm mohly například hrát hry nebo sledovat videa. Většina dětí mobilní zařízení dokáže hravě ovládat a často ho používá.

A právě na tom je založena aplikace Educhild. Díky ní budou moci děti svůj čas strávený na mobilním zařízení využívat kromě zábavy i ke vzdělávání. Rodiče získají okamžitý přehled o veškerých činnostech, jež jejich potomek během používání chytrého telefonu vykonal.

Toto téma jsem si zvolil, protože si myslím, že mobilní zařízení v rukou dětí by mohlo být využíváno také ke vzdělávání. Zároveň by v dnešní době rodičům velice pomohl alespoň jednoduchý přehled o tom, co děti na chytrém telefonu dělají.

Samotná aplikace Educhild bude rozdělena na rodičovský a dětský mód. V dětském módu se budou nacházet kvízy. Ty bude možné spustit samostatně, nebo se zobrazí automaticky po uplynutí nastaveného časového limitu. Za splnění kvízu bude dítěti časový limit obnoven na nastavenou hodnotu. V rodičovském módu se budou zobrazovat data o úspěšnosti dítěte v plnění kvízů a také statistiky o aktivitách, které byly získány během přepnutí aplikace do dětského módu. Bude zde možné nastavit rozvrh dítěte, časový limit nebo vybrat sledované aplikace. Pokud bude dítě trávit čas v jedné z nich, bude mu ubývat hrací doba a při úplném uplynutí limitu dojde k automatickému spuštění vzdělávací části. Rodič zároveň bude mít možnost přihlásit se ke stávajícímu účtu nebo si vytvořit nový.

Cílem této bakalářské práce je vytvoření rodičovského módu v Android aplikaci Educhild. Nejprve se zaměřím na analýzu dané problematiky včetně analýzy požadavků a konkurenčních řešení. Následovat bude návrh požadovaných funkcionalit, popis zvolené architektury a využitých technologií. V dalších kapitolách popíši proces realizace navržených funkcionalit a testování im-

plementovaných částí. V závěrečné kapitole zhodnotím dosavadní vývoj a navrhnu možná vylepšení, kterými by aplikace v budoucnu mohla disponovat. Výsledkem bude prototyp aplikace Educhild s funkční rodičovskou částí spustitelný na mobilních zařízeních s operačním systémem Android.

Kapitola 1

Analýza

1.1 Popis problematiky

Mobilní zařízení se v posledních několika letech stala neoddělitelnou součástí našich životů. Stále více se dostávají také do rukou dětí, a to už od velmi nízkého věku. Používání mobilních zařízení nečiní dětem sebemenší problémy zejména díky jednoduchému ovládání, které přinášejí dotykové obrazovky a přehledná rozhraní.

Studie z roku 2018 ze Spojených států amerických a Kuvajtu zabývající se používáním mobilních zařízení dětmi předškolního věku prováděná na vzorku 112 dětí udává, že pouze 17 dětí (15 %) mobilní zařízení vůbec nepoužívá. Většina dětí z testované skupiny už chytrý telefon nebo tablet někdy použila, případně jej využívá pravidelně. Za zmínku stojí, že ze skupiny dětí, které mobilní zařízení někdy použily, má svůj vlastní mobilní telefon, tablet nebo jiný druh mobilního zařízení 45 % dětí [6].

S jistotou mohu říci, že většina dětí ve věku od 3 do 5 let chytré telefony nebo tablety používá. S narůstajícím věkem se počet dětí, které nepoužívají mobilní zařízení, bude určitě ještě snižovat. Při stále pokračujícím rozmachu moderních technologií lze předpokládat, že počet dětí, které používají mobilní telefon, bude stále vyšší. Z toho je možné usuzovat, že do několika desítek let bude mobilní telefon nebo tablet vlastnit většina dětí.

Studie dále poukazuje na to, že většina dětí používá mobilní zařízení ke hraní her, nebo sledování videí. Jen malé procento dětí uvedlo, že používá mobilní zařízení za účelem vzdělávání, například ke čtení e-knih. I přesto se více než polovina z dotázaných dětí díky mobilnímu zařízení naučila rozeznávat písmena a číslice, nebo dokonce číst celá slova [6].

Z výše uvedeného vyplývá, že existuje jistý potenciál pro používání mobilních zařízení pro získávání nových znalostí a rozvoj schopností dítěte. Pokud by měl rodič možnost dítěti během hraní her, případně sledování videí, spouštět naučný kvíz, mohla by se mobilní zařízení stát efektivním zdrojem

vědomostí. Důležité je, aby se dítěti kvíz spouštěl automaticky a dítě tento kvíz nemohlo opustit před jeho vyplněním.

Pokud se na celou věc podíváme z pohledu rodičů, je potřeba, aby měli možnost kontroly úspěšnosti dítěte při plnění kvízů a zároveň získali přehled o množství času, které dítě strávilo v různých aplikacích. Na základě toho budou schopni vybírat, ve kterých aplikacích dítě tráví zbytečně moc času, a zobrazovat mu během jejich používání kvízy.

1.2 Postup vývoje

Předtím než začnu se samotnou analýzou požadavků a na ní navazujícími kapitolami, představím blíže celý projekt Educhild a způsob, jakým se postupně realizuje. Kromě toho popíši vybranou metodiku vývoje a případné alternativy, které je možné zvolit.

Educhild je projekt osmičlenného týmu pod vedením Ing. Jiřího Hunky a Ing. Libora Kudrny. Na Android aplikaci a částí s ní spojených se přímo podílí šest lidí, konkrétně se jedná o mě, Petra Šímu, Miloše Popoviče, Marka Trzaskalika, Tomáše Hanka a Terezu Langovou. Vývoj Android aplikace započal v loňském roce v rámci předmětu BI-SP1.

Zatímco Tomáš Hanko a Tereza Langová pracují na dětské části aplikace, respektive jejím návrhu, na rodičovském módu spolupracuji s Markem Trzaskalikem, který řeší uživatelské rozhraní pro rodiče, Milošem Popovičem, který se stará o backend rodičovské části a Petrem Šímou, který je zodpovědný za serverovou část podpory pro kvízy a jejich stahování do mobilního zařízení.

V navazujícím předmětu BI-SP2 se projekt rozrostl o webovou část, o kterou se stará Jan Stejskal a Barbora Kyseľová.

1.2.1 Metodika vývoje

Na začátku vývoje je třeba definovat metodiku, kterou se proces vývoje bude řídit. Metodikou vývoje rozumíme postup, který se používá pro vývoj software, v tomto případě Android aplikace Educhild. Jedná se o jakýsi souhrn pravidel, která jsou používána během analýzy, návrhu, implementace a dalších částí, s nimiž se během vývoje setkáme [7].

V současné době se nejvíce používají zejména tři metodiky a to waterfall neboli vodopád, iterativní a agilní metodika. Cílem všech je vývoj vysoce kvalitního software podle představ zákazníka v zadaném čase [7].

1.2.1.1 Vodopád

Vodopád je i dnes jednou z nejvíce používaných metodik. Tato metodika do jisté míry ovlivnila celou řadu dalších, které z ní vycházejí, případně ji lehce upravují [7].

Jednotlivé fáze procesu kopírují odpovídající fáze vývoje, které na sebe přímo navazují. Výstup každé fáze je nutný k zahájení následující.

Nevýhodou je, že při použití této metodiky je téměř nemožné reagovat na změny, které během vývoje nastanou.

1.2.1.2 Iterativní

Iterativní metodika vychází z předchozího vodopádu. Ve snaze, aby překonala klíčové limitace, které předešlá metodika má, zavádí iterativní metodika vývoj, který je rozdělen do několika iterací [7].

Stejně jako v předchozím případě jsou na začátku získány požadavky na software, ovšem jeho části jsou vyvíjeny postupně a v každé iteraci je přidána nová funkcionalita k již stávajícím, které byly implementovány v předchozích iteracích. Při použití iterativního programování je možné lépe reagovat na změny a případně je začlenit do dalšího vývoje.

1.2.1.3 Agilní

Poslední metodika, kterou představím, je nejnovější z uvedených. Nejedná se jako u předchozích pouze o jeden souhrn pravidel, ale o celou skupinu různých modelů, které s agilní metodikou přišly. Tyto modely se vyznačují velkou flexibilitou [7].

Mezi agilní metodiky patří extrémní programování, scrum a mnoho dalších. Každý model disponuje jinou výhodou oproti ostatním a je tak vhodný pro jiný projekt.

Na rozdíl od iterativního modelu přináší agilní metodika mnohem kratší jednotlivé iterace, kde na konci každé z nich nemusí být plně funkční nová část aplikace.

1.2.1.4 Shrnutí

Pro vývoj aplikace Educhild je vybrána iterativní metodika, která byla zvolena hlavně z důvodu snadnější reakce na změny, jež se v průběhu vývoje mohou objevit.

V rámci toho od začátku vývoje aplikujeme v týmu formu schůzek, které se pravidelně opakují. Díky tomu je možné rychle reagovat na nesrovnalosti, jež mohou během vývoje nastat, a ihned začlenit změny, které je řeší.

1.3 Analýza požadavků na rodičovský mód

V následující kapitole se zaměřím na analýzu požadavků, které jsou kladené na vyvíjenou aplikaci. Nejprve představím postupy, kterými je možné požadavky získávat a následně konkrétní způsob, který byl zvolen při provádění analýzy požadavků na aplikaci Educhild.

1.3.1 Způsob získávání požadavků

Protože získávání požadavků je jednou z nejdůležitějších činností, které vedou k vývoji software odpovídajícího představám zadavatele, vzniklo několik technik, které se právě tomuto problému věnují [8]. V současné době rozlišujeme tři typy technik, které se nejčastěji používají při sběru požadavků. Jedná se o komunikaci se zadavatelem, pozorování a analytickou techniku.

1.3.1.1 Komunikace se zadavatelem

Tato technika je postavena na komunikaci mezi dvěma a více lidmi [8]. Protože je konverzace přirozený způsob, jak vyjádřit nápady, případně pokládat dotazy nebo je zodpovídat, jedná se o efektivní způsob, jak je možné zachytit požadavky ve velice přesné formě. Pro realizaci této techniky lze využít jednoduché rozhovory, případně různé workshopy nebo brainstormingy. Další možností je použití dotazníku nahrazujícího přímou komunikaci mezi osobami.

1.3.1.2 Pozorování

Základ techniky tvoří pozorování rutinních aktivit, které člověk vykonává, což umožňuje porozumět procesům a pochopit, co se za nimi skrývá [8]. Tato technika je velice užitečná, pokud proces, který je skryt za požadavkem, není možné popsat jednoduše slovy, ale je mnohem snazší jej názorně ukázat na příkladu.

1.3.1.3 Analytická technika

U předchozích dvou metod jsou požadavky získávány přímo z lidského chování, případně z popsaných myšlenek. Oproti tomu analytická technika se zaměřuje na existující dokumentaci nebo znalosti a popis požadavků čerpá právě z nich [8]. Díky tomu je možné využít stávající požadavky nebo si studováním dříve zhotovené dokumentace a jejího obsahu vytvořit obraz o zkoumané problematice a na jeho základě získat požadavky nové.

1.3.1.4 Shrnutí

Velice často se v praxi používá také kombinace výše uvedených technik [8]. Pro správné zachycení požadavků je nutné zvolit vhodný typ techniky. Každá využívá specifické interakce mezi zadavatelem a člověkem, který analyzuje požadavky. Pokud existuje dokumentace je možné k získání požadavků využít také ji.

Během vývoje aplikace Educhild byla aplikována technika komunikace se zadavatelem, protože se v tomto ohledu jedná o techniku nejpřínosnější. Díky tomu, že se v rámci týmu konaly každotýdenní schůzky, bylo tímto způsobem nejjednodušší na základě popisu zadavatele vytvořit seznam požadavků, které bylo možné následně ověřit, případně je při nalezení neshody poupravit. Do

výsledného seznamu požadavků se kromě komunikace se zadavatelem promítl také průzkum uživatelů, který je blíže popsán v bakalářské práci Marka Trzaskalika. Požadavky byly tímto způsobem získávány a upravovány i po celou dobu tvorby návrhu, aby přesně odpovídaly představám zadavatele.

1.3.2 Vyhodnocování požadavků

U každého požadavku kromě jeho popisu uvádím i jeho kategorii, prioritu a složitost. Pro vyjádření priority a složitosti používám stupnici od 1 do 3, kde číslo 1 označuje nejdůležitější, respektive nejsložitější požadavek. Vysvětlení hodnot stupnice je v tabulce 1.1. Díky tomu je následně možné požadavky řadit a lépe odhadovat jejich časovou náročnost. Protože je vývoj aplikace samotné rozdělen na několik částí, zmiňuji zde pouze konkretní požadavky, které se přímo týkají rodičovského módu aplikace.

Hodnota stupnice	1	2	3
Slovní vyjádření	Vysoká	Střední	Nízká

Tabulka 1.1: Vyhodnocování priority a složitosti požadavků

1.3.3 Funkční požadavky

1.3.3.1 F1. Přihlašování a registrace

Uživatel, v tomto případě rodič, musí mít možnost přihlásit se ke svému stávajícímu účtu, případně se registrovat a vytvořit si účet nový. Kromě toho, musí být uživateli umožněno registraci a přihlášení přeskočit, aby mohl aplikaci využívat i bez vytvořeného účtu. Uživatelské účty jsou využity zejména při synchronizaci zařízení se serverem, což umožní použití více zařízení a zároveň také využití webového rozhraní, kde bude uživatel mít možnost vytvářet vlastní kvízy. Pokud uživatel zvolí využívání aplikace bez uživatelského účtu, o tyto možnosti přijde.

Kategorie: Funkčnost

Priorita: 1

Složitost: 2

1.3.3.2 F2. Prvotní nastavení

Během prvotního nastavení dojde k úvodnímu zadání potřebných hodnot po stažení aplikace. Uživatel zde bude nastavovat nejpotřebnější údaje důležité ke správnému běhu aplikace. Musí zde být obsaženo nastavení PIN kódu. Dále uživatel musí mít možnost vytvoření profilu dítěte. U tohoto profilu musí být

vyplněny základní informace o dítěti, jako jsou jeho jméno a rok a měsíc narození. Dále musí být uživateli nabídnuta možnost výchozího nastavení, ve kterém aplikace sama nastaví potřebná data. Zároveň musí mít uživatel možnost tato data nastavit sám podle svých preferencí.

Kategorie: Funkčnost

Priorita: 1

Složitost: 2

1.3.3.3 F3. Spravování sledovaných aplikací

Uživatel musí mít možnost spravovat seznam sledovaných aplikací. Jako sledované aplikace jsou označeny ty, během jejichž používání dochází ke spouštění kvízu po uplynutí časového limitu. Rodičovský mód tedy musí obsahovat způsob přidávání, případně odebírání aplikací v rámci skupiny sledovaných aplikací.

Kategorie: Funkčnost

Priorita: 1

Složitost: 2

1.3.3.4 F4. Zobrazování statistik

V rodičovském módu aplikace musí být shrnuta aktivita dítěte. Rodič musí mít možnost sledovat, kolik času tráví jeho potomek na mobilním zařízení a zejména, ve kterých aplikacích nejvíce, aby na tyto skutečnosti případně mohl reagovat a přesunout aplikace, o kterých si myslí, že jsou využívány nadbytečně, do skupiny sledovaných aplikací.

Kromě aplikačních statistik musí mít rodič možnost sledovat statistiky úspěšnosti dítěte ve vyplněných kvízech. Při každém plnění kvízů se musí zaznamenávat úspěšnost, tedy správně i chybně zodpovězené otázky. Tato aktivita dítěte v podobě úspěšnosti se musí zobrazovat v rodičovském módu, aby měl rodič představu o tom, jak se dítěti při vyplňování kvízů daří.

Kategorie: Funkčnost

Priorita: 2

Složitost: 1

1.3.3.5 F5. Spravování časového limitu

Časový limit je doba, kterou bude moci dítě strávit v aplikacích ze skupiny sledovaných. Rodič tedy musí mít možnost nastavit tento limit podle svých představ.

Kategorie: Funkčnost

Priorita: 1

Složitost: 3

1.3.3.6 F6. Časové rozvrhy

V aplikaci musí být možnost vytvářet pro dítě časové rozvrhy. Pokud bude časový rozvrh aktuálně aktivní, dojde k úplnému zablokování všech monitorovaných aplikací.

Rozvrhy bude možné vytvářet na základě času, aby se mohly spouštět jen v určitou denní dobu. Časový rozvrh by mělo být možné zároveň přiřadit pouze k určitým dnům v rámci týdne. Kromě toho musí aplikace podporovat mazání rozvrhů, ale také pouhé pozastavení jejich aktivity, aby nebylo nutné rozvrh pokaždé mazat a vytvářet znovu v případě, že se rodič rozhodne jej dočasně nepoužívat.

Kategorie: Funkčnost

Priorita: 2

Složitost: 2

1.3.3.7 F7. Nastavení

Rodič musí mít v aplikaci možnost přenastavení veškerých dříve zadaných hodnot. Proto musí aplikace obsahovat nastavení, kde bude možné upravovat uživatelský účet nebo základní informace o dítěti. V rámci uživatelského účtu musí mít rodič možnost odhlásit se nebo naopak přihlásit, případně registrovat, pokud využíval aplikaci bez uživatelského účtu. Samozřejmým požadavkem je změna PIN kódu nebo hesla k uživatelskému účtu.

Kategorie: Funkčnost

Priorita: 1

Složitost: 2

1.3.3.8 F8. Shrnující hlavní obrazovka

V aplikaci musí být vytvořena obrazovka, na které budou shrnuty veškeré podstatné informace důležité k jejímu využívání. Měly by zde být informace o aktuálně nastaveném časovém limitu, probíhajícím nebo nadcházejícím časovém rozvrhu a posledních aktivitách dítěte, zejména o úspěšnosti v plnění kvízů.

Kategorie: Funkčnost

Priorita: 3

Složitost: 2

1.3.3.9 F9. PIN kód

Aplikace musí při přepínání z dětského do rodičovského módu požadovat předem nastavený PIN kód. Díky tomu bude dítěti, které daný PIN kód nezná, zamezen přístup do rodičovského módu a nebude mít možnost měnit nastavení aplikace vytvořené rodičem.

Kategorie: Funkčnost

Priorita: 1

Složitost: 3

1.3.3.10 F10. Odměny

Aby dítě mělo motivaci kvízy plnit, musí aplikace obsahovat systém odměn. Ty bude mít možnost přidávat rodič a dítě si je bude moci zakoupit za virtuální měnu získanou při vyplňování kvízů. V rodičovském módu proto musí být obsažena možnost tyto odměny vytvářet, případně vybírat z předem vytvořených návrhů. Pokud dítě odměnu zakoupí, musí být rodič upozorněn, aby ji mohl splnit.

Kategorie: Funkčnost

Priorita: 3

Složitost: 1

1.3.4 Nefunkční požadavky

1.3.4.1 N1. Podpora Android verzí 5.0 a vyšší

Aplikace musí být určená pro operační systém Android. Tento operační systém je v současné době nejvyužívanějším operačním systémem pro mobilní zařízení. V roce 2020 byl operační systém Android nainstalován na celkově 84,8 % prodaných zařízeních v rámci globálního trhu. Podle prognóz by navíc na konci roku 2022 měl Android dosáhnout 86% podílu na trhu mobilních zařízení [1]. Každý rok vychází nová verze samotného operačního systému, jež se váže k určitému API. Vydané verze mohou obsahovat nové prvky, které nemusí být podporovány v předchozích verzích. Vyvíjená aplikace musí mít podporu od verze Android 5.0 s označením Lollipop. Verze 5.0 přinesla mnoho užitečných nových funkcí od aktualizace grafiky, kde přibyly Android Extension Pack a podpora Material design, až po úpravu notifikací a konektivity [9]. V dubnu roku 2020 podpora od verze API 21 pokrývala zhruba 94 % aktivních zařízení a tento procentuální poměr nadále roste s tím, jak lidé postupně přecházejí na zařízení s novějšími verzemi Android API [10]. Díky tomu bude zajištěna možnost využívat aplikaci na většině zařízení s operačním systémem Android.

Kategorie: Podporovatelnost

Priorita: 1

Složitost: 2

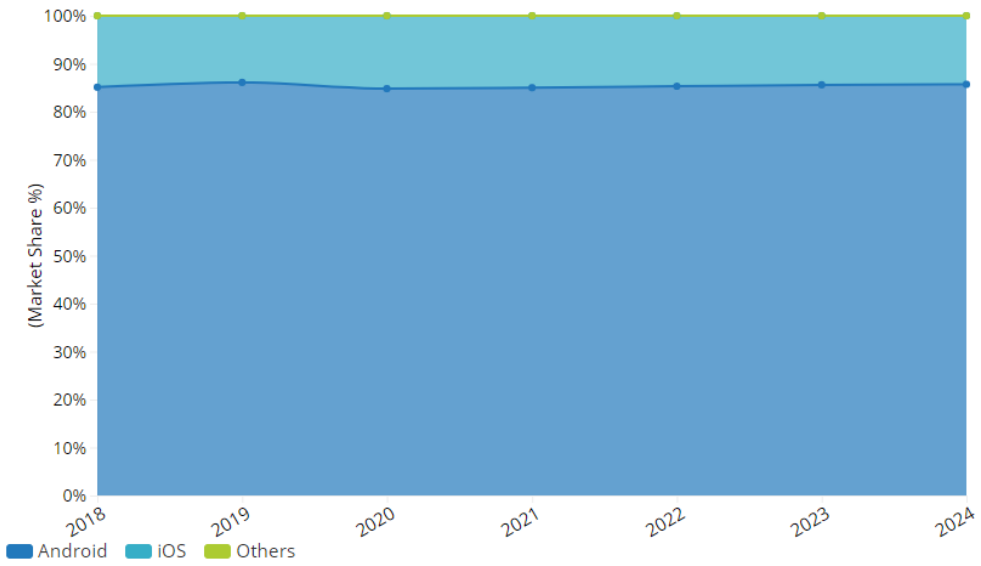
1.3.4.2 N2. Podpora různých velikostí obrazovek

Protože musí být možné aplikaci využívat na různých typech mobilních zařízení s různě velkými úhlopříčkami displejů, je potřeba vytvářet rozložení jednotlivých obrazovek tak, aby bylo možné aplikaci používat na mobilním telefonu a stejně tak na tabletu. I když většina zařízení má normální velikost displeje, jak je vidět na obrázku 1.2, existuje přibližně pětina mobilních zařízení, jejichž velikost displeje se řadí do velkých a extra velkých [2]. Proto není možné ani tato zařízení opomenout.

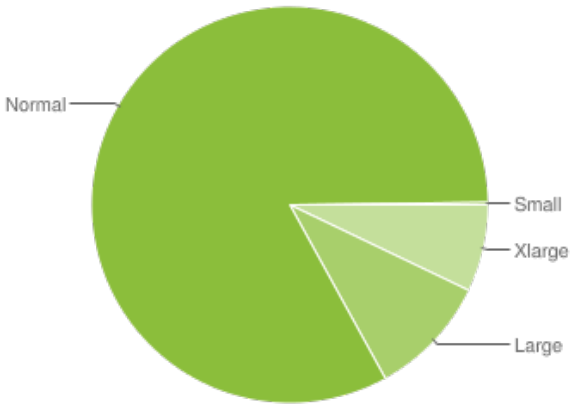
Kategorie: Použitelnost

Priorita: 1

Složitost: 3



Obrázek 1.1: Rozdělení trhu s mobilními zařízeními, převzato z [1]



Obrázek 1.2: Rozdělení mobilních zařízení dle velikosti displeje, převzato z [2]

1.3.4.3 N3. Lokalizace

Aplikace musí podporovat primárně anglický a český jazyk. Aplikaci zároveň musí být triviální přeložit do dalších jazyků.

Kategorie: Podporovatelnost

Priorita: 2

Složitost: 3

1.3.5 Shrnutí

Díky analýze jsem získal kompletní seznam podmínek, které jsou kladené na rodičovský mód vyvíjené aplikace. Tyto požadavky je nutné zohlednit při vytváření návrhu funkcionalit, kterému se budu věnovat v následujících kapitolách.

1.4 Analýza případů užití

Případy užití tvoří nedílnou součást vývoje jakéhokoliv software. Díky nim je možné popsat jednotlivé akce, které by vyvíjený systém měl na základě kolaborace s uživateli, tzv. aktéry, vykonávat. Každý případ užití by měl mít jasně daný výsledek [11].

1.4.1 Aktéři

Pod pojmem aktér si představujeme roli, ve které může být člověk, případně nějaký externí vliv (například čas), jež interaguje se systémem [11]. V rámci rodičovského módu aplikace se objevují dva aktéři, konkrétně rodič a dítě.

1.4.1.1 Rodič

Rodič jako aktér je entita, která bude v rámci rodičovského módu nejpodstatnější. Na základě jeho požadavků se bude spouštět většina případů užití, které jsou uvedeny níže.

1.4.1.2 Dítě

Dalším aktérem je dítě. Tento aktér nebude přímo interagovat s rodičovským módem aplikace, ale na základě jeho činností budou vznikat data, která budou podstatná pro předchozího aktéra.

1.4.2 Případy užití

V následujícím seznamu případů užití uvádím ty, které jsou podstatné pro rodičovskou část aplikace.

1.4.2.1 UC1. Registrace

V aplikaci bude rodiči po prvotním spuštění aplikace umožněno vytvořit si uživatelský účet na základě e-mailové adresy a hesla.

Alternativně se bude moci rodič registrovat a vytvořit si uživatelský účet, pokud bude aplikaci používat lokálně, bez uživatelského účtu. V tomto případě musí být uživateli nabídnuto migrování stávajících dat do nově vytvořeného účtu.

1.4.2.2 UC2. Přihlášení

Rodič bude mít po prvotním spuštění možnost přihlásit se ke svému stávajícímu účtu, který již má vytvořený. Alternativně musí být rodiči nabídnuta možnost přihlášení během používání aplikace v lokálním režimu.

1.4.2.3 UC3. Přeskočení autentizace

Rodiči bude umožněno využívat aplikaci i bez vytváření uživatelského účtu, proto musí mít možnost autentizaci v podobě přihlášení nebo registrace přeskočit.

1.4.2.4 UC4. Vytvoření nového PIN kódu

Rodiči bude umožněno vytvořit vlastní PIN kód, který bude nutné zadat při přechodu z dětského do rodičovského módu aplikace.

1.4.2.5 UC5. Zadání PIN kódu

Při přechodu z dětského do rodičovského módu bude po rodiči vyžadováno zadání jím nastaveného PIN kódu, aby bylo zamezeno přístupu dítěte do rodičovského módu.

1.4.2.6 UC6. Změna PIN kódu

Rodič bude mít možnost změnit stávající PIN kód. Pro bezpečnou změnu bude po rodiči požadováno nejprve zadání stávajícího kódu, za účelem ověření uživatele, až poté bude možné nastavit PIN kód nový.

1.4.2.7 UC7. Vytvoření profilu dítěte

Rodiči bude umožněno vytvoření profilu dítěte. Tento profil se bude skládat ze jména dítěte a jeho měsíce a roku narození.

1.4.2.8 UC8. Úprava dat z profilu dítěte

Rodiči musí být umožněno údaje z profilu dítěte změnit. Musí být možné nastavit jiné jméno, rok i měsíc narození.

1.4.2.9 UC9. Výchozí nastavení

Rodiči bude nabídnuto výchozí nastavení aplikace se základním časovým limitem a sledovanými aplikacemi.

1.4.2.10 UC10. Pokročilé nastavení

Kromě výchozího nastavení musí být rodiči nabídnuta možnost nastavení hodnot v rámci aplikace podle jeho preferencí. Rodič musí mít možnost projít seznamem sledovaných aplikací, nastavením časového limitu a vytvářením časových rozvrhů.

1.4.2.11 UC11. Zobrazení seznamu sledovaných aplikací

Rodič musí mít možnost zobrazit, které aplikace jsou aktuálně součástí seznamu sledovaných aplikací.

1.4.2.12 UC12. Úprava seznamu sledovaných aplikací

Rodiči musí být umožněno přidat do seznamu sledovaných aplikací další ze seznamu všech aplikací nainstalovaných v mobilním zařízení. Alternativou je odebrání aplikace ze seznamu sledovaných aplikací, pokud to rodič uzná za vhodné.

1.4.2.13 UC13. Zobrazení statistik o aktivitě dítěte

Rodiči musí být nabídnuta možnost zobrazení statistik o aktivitách, které dítě na mobilním zařízení vykonává. Těmito aktivitami se rozumí jednak zobrazení času stráveného v jednotlivých aplikacích a také v kategoriích, do kterých se veškeré aplikace rozdělují. Toto zobrazení bude možné přepínat mezi různými časovými obdobími. Statistiky jsou získávány během přepnutí aplikace do dětského módu, tedy během používání mobilního zařízení dítětem.

1.4.2.14 UC14. Zobrazení posledních aktivit dítěte v kvízech

Rodič musí mít možnost v aplikaci zobrazit shrnutí posledních aktivit dítěte v kvízech. Těmito aktivitami jsou myšleny zejména naposledy vyplněné kvízy a skóre dítěte v nich.

1.4.2.15 UC15. Změna časového limitu

Rodiči musí být umožněno změnit časový limit, po jehož uplynutí se dítěti spustí vybraný kvíz.

1.4.2.16 UC16. Zobrazení časových rozvrhů

V aplikaci musí být obsažena možnost zobrazení všech vytvořených časových rozvrhů.

1.4.2.17 UC17. Vytvoření časového rozvrhu

Aplikace umožní rodiči vytvářet časový rozvrh pro jeho dítě. V aktivním časovém rozvrhu je znemožněno spuštění kterékoliv sledované aplikace. Vytvoření časového rozvrhu se skládá z vyplnění názvu, zadání časového intervalu, ve kterém má být aktivní a vybrání jednoho, případně skupiny dní v týdnu.

1.4.2.18 UC18. Úprava časového rozvrhu

Rodič musí mít možnost upravit jakékoliv z dat zadaných při vytváření časového rozvrhu, musí mu být umožněno změnit název, časový interval, případně přidat nebo odebrat den z množiny, která udává, kdy je daný časový rozvrh aktivní.

1.4.2.19 UC19. Smazání časového rozvrhu

Aplikace musí rodiči umožňovat smazání vybraného časového rozvrhu.

1.4.2.20 UC20. Změna aktivnosti časového rozvrhu

Aplikace umožní rodiči zneaktivnit časový rozvrh tak, aby nebyl ve vybraném časovém intervalu a dni aktivován, ale zároveň nebylo nutné ho smazat.

Alternativně bude možné časový rozvrh, který je označen jako neaktivní, znovu aktivovat.

1.4.2.21 UC21. Změna hesla

Rodiči bude umožněno změnit heslo ke svému uživatelskému účtu. Aby ke změně došlo, bude nutné zadat stávající heslo za účelem ověření identity uživatele. Změna hesla je podmíněna používáním uživatelského účtu.

1.4.2.22 UC22. Odhlášení uživatele

Aplikace musí uživateli umožnit odhlásit se ze svého stávajícího uživatelského účtu, pokud je aktuálně přihlášen.

1.4.2.23 UC23. Zobrazení odměn

Rodič musí mít možnost v aplikaci zobrazit seznam odměn, které dříve vytvořil.

1.4.2.24 UC24. Přidání odměny

V aplikaci musí být pro rodiče připraveno rozhraní pro vytváření odměn pro dítě. Odměny by měly obsahovat jak jejich popis, tak počet celkové virtuální měny potřebné k jejímu zakoupení.

1.4.2.25 UC25. Zobrazení zakoupených odměn

Rodič musí být upozorněn v případě, že dítě zakoupilo nějakou odměnu, aby ji mohl zajistit.

1.4.2.26 UC26. Úprava odměny

Rodiči musí být umožněno upravit veškeré atributy odměny. Úpravy jsou podmíněny tím, že nebyla daná odměna dříve dítětem zakoupena, v tom případě její úprava není možná.

1.4.2.27 UC27. Smazání odměny

Aplikace bude umožňovat odstranění odměny. Podmínkou odebrání odměny je, že nebyla zakoupena dítětem. Pokud ano, není její smazání možné.

1.4.3 Diagram případů užití

Diagram případů užití tvoří grafické znázornění jednotlivých případů, které jsou popsané výše. Diagram je důležitý pro názornější popsání jednotlivých vztahů mezi případy užití.

Konkrétní diagram pro rodičovský mód aplikace Educhild, který je z důvodu větší přehlednosti rozdělen do dvou částí, se nachází na obrázcích 1.3 a 1.4.

V prvním případě se v diagramu neobjevuje aktér dítě, protože veškeré případy užití zde uvedené jsou čistě v režii aktéra rodič. Naopak v druhé části se aktér dítě stará o vytváření statistik a na základě jeho činnosti vznikají poslední aktivity.

1.4.4 Pokrytí případů užití

Protože jednotlivé případy užití zpravidla jsou součástí jednotlivých funkčních požadavků, je třeba ověřit, zda nebyl žádný vynechán. K tomu se používá tabulka pokrytí, v níž je jednoznačně označeno, který případ užití pokrývá který požadavek.

Tabulka 1.2 znázorňuje pokrytí pro předešlé případy užití v rámci rodičovského módu aplikace Educhild.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
UC1	✓						✓			
UC2	✓						✓			
UC3	✓									
UC4		✓								
UC5									✓	
UC6							✓			
UC7		✓								
UC8							✓			
UC9		✓								
UC10		✓								
UC11			✓							
UC12			✓							
UC13				✓						
UC14				✓				✓		
UC15					✓					
UC16						✓				
UC17						✓				
UC18						✓				
UC19						✓				
UC20						✓				
UC21							✓			
UC22							✓			
UC23										✓
UC24										✓
UC25										✓
UC26										✓
UC27										✓

Tabulka 1.2: Tabulka pokrytí případů užití

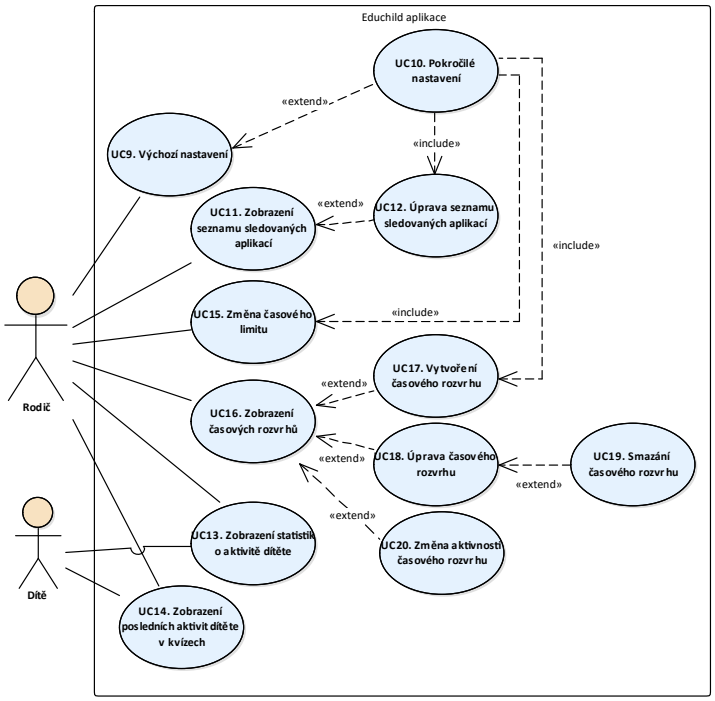
Obrázek 1.3: Diagram případů užití část 1.

1.5 Analýza konkurenčních řešení

V této sekci se věnuji analýze konkurenčních aplikací, které do jisté míry řeší výše popsanou problematiku. V rámci analýzy jsem se zaměřil především na aplikace, které jsou dostupné pro operační systém Android, neboť právě na ten cílí i aplikace Educhild. Konkurenční aplikace jsem vybíral na základě jejich popisu a počtu stažení v obchodě Google Play. U každé z aplikací se zaměřuji především na užitečné funkcionality, kterými disponují a díky kterým patří v komunitě uživatelů mezi oblíbené.

Cílem analýzy konkurenčních aplikací je nalezení stávajících řešení funkcionalit, jež byly uvedeny v předchozích částech, případně nových funkcionalit, kterými by rodičovský mód měl disponovat. Výsledkem je pro každou aplikaci seznam užitečných funkcionalit v ní obsažených. Díky tomu bude možné navrhovat funkcionality podle ověřených postupů a zaběhlých zvyklostí.

Popisované aplikace je možné rozdělit do dvou kategorií. První kategorii tvoří aplikace, které mohou být využity přímo k blokování jiné aplikace, případně jejich skupiny nebo celého mobilního zařízení, a to za účelem rodičovské kontroly nebo lepšího soustředění. V druhé skupině se nacházejí aplikace,



Obrázek 1.4: Diagram případů užití část 2.

které se zaměřují zejména na vzdělávání a vyplňování kvízů. V těchto aplikacích je možné vyplňovat povětšinou soubor otázek různého typu a s různým zaměřením.

1.5.1 Blokovací aplikace

1.5.1.1 Qustodio

Qustodio je aplikace, která se zaměřuje hlavně na blokování dětem nevhodného obsahu, ať už se jedná o nebezpečné aplikace nebo nevhodné webové stránky [12]. Slouží jako pomůcka pro rodiče, kterým dává přehled o činnostech, jež děti na mobilním zařízení vykonávají, případně jim umožňuje tyto činnosti zakázat. Aplikace je rozdělena na dětskou a rodičovskou část. Rodičovská část obsahuje veškeré potřebné nastavení a zobrazení statistik a dalších dat získaných během používání mobilního zařízení dětmi, jak je vidět na obrázku 1.5. Aplikace je jak ve verzi pro Android a iOS, tak také pro Windows, aby mohla být použita i na noteboocích nebo stolních počítačích.

Ostatní aplikace v mobilním zařízení lze prostřednictvím této úplně zablokovat, případně jim nastavit časový limit, po jehož uplynutí je uživateli

zamítnuto jejich další využívání. Toho je docíleno díky vyskakujícímu upozornění, které jakoukoliv činnost znemožňuje.

Kromě blokování jsou pomocí aplikace zaznamenávány veškeré činnosti, které dítě na mobilním zařízení vykonává, od zpráv a hovorů až po aktivitu na sociálních sítích nebo vyhledávaní stránek v prohlížeči. Kromě toho je také zaznamenávána poloha mobilního zařízení, aby měl rodič o svém potomkovi kompletní přehled.

Po zablokování zařízení je přístupná pouze samotná aplikace Qustodio, kde se nachází vyobrazení časového přehledu pro dané zařízení. Rodič se může po zadání hesla přihlásit do rodičovské části, ovšem má povolený přístup pouze na jednu minutu, proto je tato aplikace vhodná spíše, pokud dítě má vlastní mobilní zařízení. Rodič má sám na svém zařízení nainstalovanou stejnou aplikaci, která je stále přepnutá do rodičovského módu.

Součástí aplikace je také zablokování jejího odstranění z mobilního zařízení. Dítě není schopno aplikaci odinstalovat a vyhnout se tak blokování svých oblíbených aplikací.

1.5.1.2 AppBlock

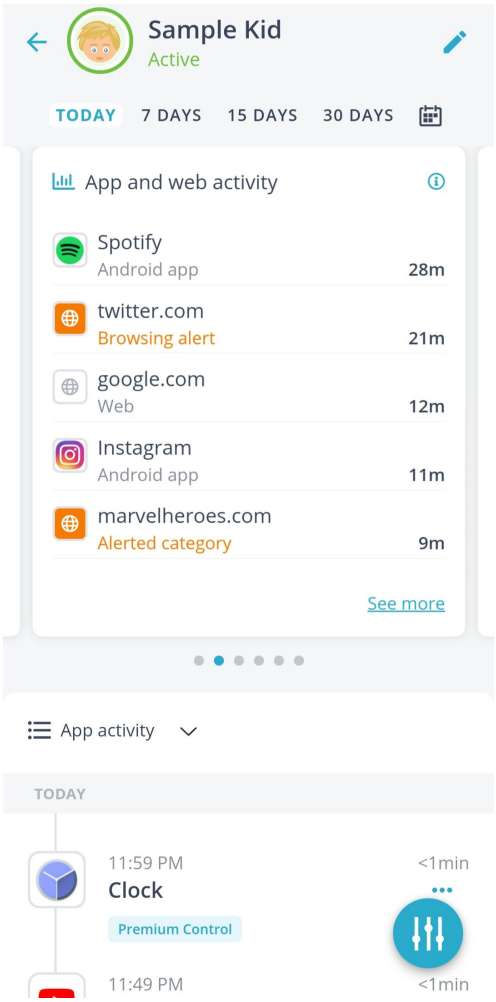
Aplikace AppBlock je další z aplikací určená zejména k blokování, v tomto případě je zaměřena na pomoc uživateli, aby se lépe soustředil na vykonávanou činnost, například na studium ve škole nebo na práci [13]. Pokud je aplikace přepnuta do „*strict*“ módu, je po uživateli požadováno zadání PIN kódu, aby došlo k odblokování zakázaných aplikací. Kromě aplikací samotných je možné blokovat také jejich notifikace, aby uživatele skutečně nic nerozptylovalo.

Hlavní výhodou aplikace je vytváření časových rozvrhů pro jednotlivé dny. V časovém rozvrhu je zvolena skupina aplikací, do kterých v dané době není umožněn přístup. To přináší možnost vytvořit například časový rozvrh na školní den, ve kterém je během dopoledních hodin zakázáno používání pro dítě rozptylujících aplikací. Blokování je možné nastavit nejen podle času, ale například i podle lokality, ve které se mobilní zařízení aktuálně nachází.

Kromě těchto funkcí aplikace obsahuje navíc velmi stručné zobrazení statistik o využívání mobilního zařízení za den nebo poslední týden.

1.5.1.3 Forest

Forest je aplikace, která sama o sobě žádné ostatní aplikace neblokuje, ale funguje na podobném principu. V aplikaci je možné nastavit, po jak dlouhou dobu se uživatel chce soustředit [14]. Poté jednoduše „*zasadí strom*“ a ten po nastavenou dobu roste a sílí. V případě, že uživatel během doby, kdy neměl být na mobilním zařízení aktivní, použije jinou aplikaci, jeho strom začne chřadnout.



Obrázek 1.5: Přehled o aktivitách dítěte v aplikaci Qustodio

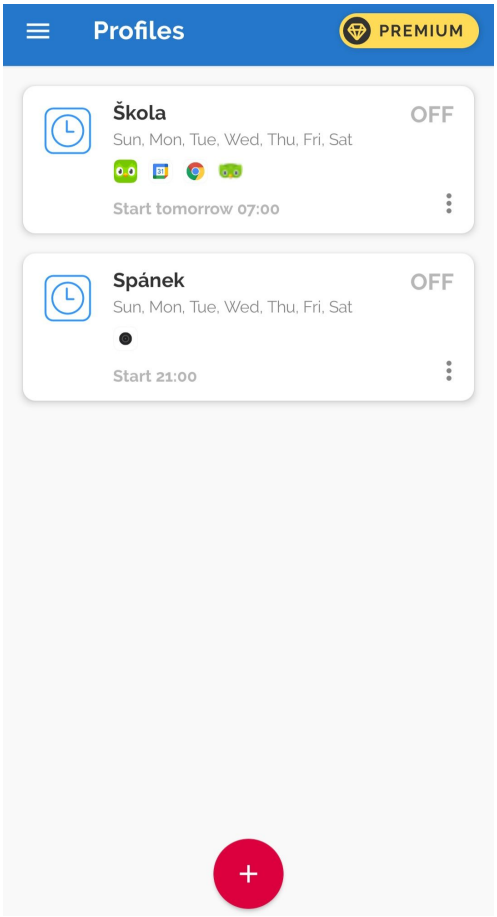
Aplikace tedy motivuje uživatele k tomu, aby se soustředil na vykonávanou činnost a vyhnul se rozptylujícímu mobilnímu zařízení, za což je v případě splnění odměněn. V opačném případě žádnou odměnu nezíská.

Tento nápad se těší velké oblibě. Aplikace má v současné době více než 10 milionů stažení a v letech 2015–2016 byla zvolena nejlepší aplikací roku v obchodě Google Play. [15]

1.5.2 Vzdělávací aplikace

1.5.2.1 Duolingo

Duolingo je jedna z nejznámějších vzdělávacích aplikací [16]. Aplikace je dostupná jak na zařízeních s operačním systémem iOS, tak pro uživatele s An-



Obrázek 1.6: Časové rozvrhy v aplikaci AppBlock

droid zařízeními, kde už dosáhla na více než 100 milionů stažení [17]. Tato aplikace je zaměřena pouze na vzdělávání v oblasti jazyků. Aplikace je založena na skutečnosti, že se chce uživatel přiučit něčemu novému, proto aplikace kromě zobrazení notifikací nijak uživatele nenutí k vyplňování kvízů.

Do aplikace je možné se přihlásit pomocí e-mailové adresy, stejně tak jako pomocí Google, případně Facebook účtu. V aplikaci si může uživatel nastavit připomínání procvičovaní. Zároveň je možné na e-mail odesílat informace o zlepšení a další tipy k učení.

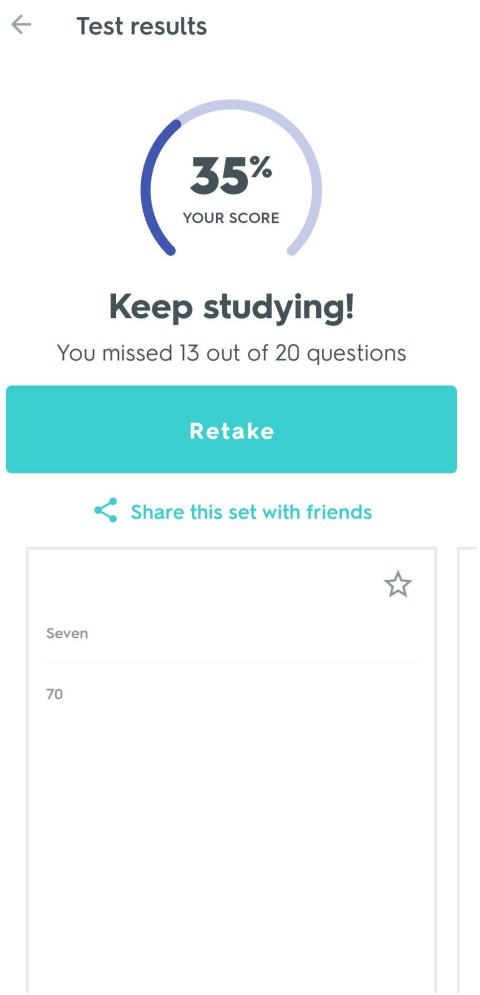
K motivaci uživatele k tomu, aby kvízy dobrovolně spouštěl, slouží systém statistik, kde je možné zjistit, jak si uživatel aktuálně vede. Za každý úspěšně dokončený kvíz se uživateli připočtou „*XP*“, díky čemuž se postupně dostává na stále lepší úroveň, a zároveň „*crowns*“, což je jakási měna, za kterou si následně může zakupovat nové kvízy. Kromě toho uživatel za plnění kvízů získává také výkonnostní ocenění.

Obrázek 1.7: Hlavní obrazovka aplikace Duolingo

1.5.2.2 Quizlet

Aplikace Quizlet je další z aplikací zaměřená čistě na vzdělávání [18]. To znamená, že v rámci jejího používání nedochází k automatickému spouštění vzdělávací části. Uživatel není nijak nucen ke spuštění kvízu, které tak závisí čistě na jeho vlastním svědomí a ochotě se něco nového naučit. Uživatel je na kvíz pouze upozorněn pomocí notifikace, a to v jím zvolenou dobu. V aplikaci se ke svému účtu stejně jako v předešlém případě přihlašuje pomocí e-mailové adresy nebo poskytovatelů identity Google a Facebook.

Z rodičovského hlediska stojí za zmínku zobrazení statistik o úspěšnosti v daném kvízu. Uživateli je v aplikaci zobrazeno nejen celkové skóre, ale má také možnost projít všechny otázky, aby viděl, na které odpověděl špatně a které otázky mu naopak nedělaly problém.



Obrázek 1.8: Zobrazení statistik v aplikaci Quizlet

1.5.3 Shrnutí

Při procházení konkurenčních aplikací jsem nalezl několik vhodných, užitečných a zajímavě zpracovaných funkcionalit, kterými by aplikace Educhild a její rodičovský mód měly disponovat.

Výsledný seznam nalezených funkcionalit vhodných pro aplikaci Educhild, případně rodičovský mód ke každé z analyzovaných aplikací je uveden v tabulce 1.3. Způsob, kterým jsou tyto funkcionality řešeny, jsem následně zohledňoval při návrhu funkcionalit rodičovského módu. Díky této tabulce je možné okamžitě zjistit, ve které z aplikací se nacházela konkrétní funkcionalita.

Název aplikace	Seznam nalezených funkcionalit
Qustodio	Vybírání nevhodného obsahu Zobrazení statistik o užívání zařízení Časový limit Blokování odinstalování aplikace
AppBlock	Výběr zakázaných aplikací PIN kód pro přepínání mezi módy Časové rozvrhy Blokování notifikací Blokování aplikací podle polohy
Forest	Odměny za nepoužívání zařízení
Duolingo	Systém odměn za plnění kvízů Odesílání e-mailů o úspěších
Quizlet	Zobrazování statistik o vyplňování kvízů Možnost procházení odpovědí

Tabulka 1.3: Shrnutí nalezených funkcionalit

Kapitola 2

Návrh

V této kapitole se budu věnovat návrhové části vývoje aplikace Educhild. Nejprve popíši jednotlivé entity a vztahy v rámci dané domény a představím doménový model. Poté shrnu postupy a technologie, které jsou důležité během vytváření aplikací, zejména pak pro operační systém Android. Technologie se objeví v závěrečných sekcích této kapitoly, kde popíši použitou architekturu spolu s možnými alternativami a využitý framework a jeho přínos.

2.1 Doménový model

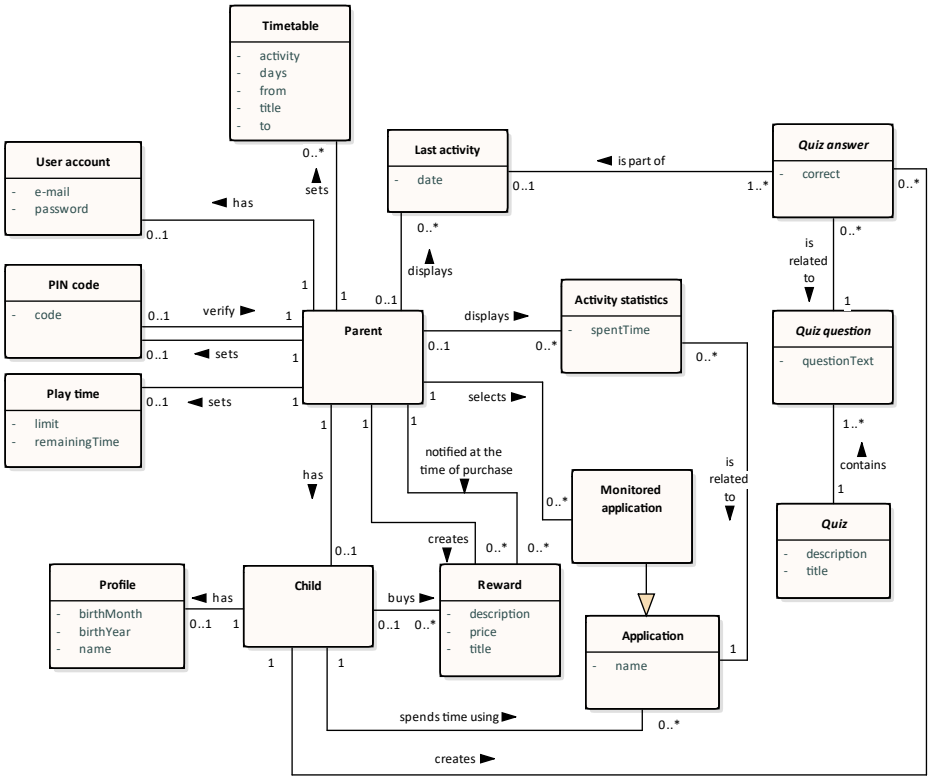
Další významnou částí procesu vývoje software je vytváření doménového modelu spolu s bližším popisem jednotlivých entit a vztahů.

Doménový model je organizovaná a strukturovaná znalost dané problematiky. Měl by reprezentovat nejdůležitější pojmy a klíčové koncepty, které se skrývají za problematikou domény [19]. Důležitou součástí doménového modelu je identifikace vztahů a vazeb mezi jednotlivými entitami, které se objevují v rámci popisované problematiky.

Doménový model tvoří jakousi páteř při dalším návrhu, ať už se jedná o třídy nebo interakce mezi nimi. Jde o vizuální reprezentaci konceptuálních tříd nebo prvků reálného světa, které se objevují v rámci analyzované problematiky. Díky němu je možné shrnout jak chování jednotlivých entit a interakci mezi nimi, tak data, která skrývají [19].

Doménový model pro rodičovskou část aplikace Educhild se nachází na obrázku 2.1.

Z hlediska kvízové části (entity označené kurzívou) se jedná o zjednodušenou variantu pouze s částmi, které jsou podstatné pro rodičovskou část. Kompletní varianta je v bakalářské práci Petra Šímy, který se právě kvízy zabývá.



Obrázek 2.1: Doménový model

2.1.1 Popis entit

V této kapitole se věnuji popisu jednotlivých entit, které se objevují v doménovém modelu. U každé entity je nutné vysvětlit jednotlivé atributy, protože ne vždy musí být zřejmé [19].

2.1.1.1 Parent

Parent je hlavní entita celého rodičovského módu. Jedná se o rodiče, který se stará o nastavování veškerých potřebných náležitostí. Rodič má vazbu na většinu ostatních entit, jelikož na něm závisí jak ostatní entity budou nakonfigurované.

2.1.1.2 Child

Další entitou, která se v modelu objevuje je entita Child. Entita označuje dítě předchozí entity, na které je primárně zaměřeno využívání aplikace. Dítě bude

hrát hlavní roli v dětském módu, ale i v doméně rodičovského módu má určitou funkci. Na základě jeho činnosti vznikají statistiky a poslední aktivity v kvízech, které jsou nedílnou součástí rodičovského módu. Další z činností dítěte je zakupování odměn. Child má vazbu na entitu Parent. V rámci současného návrhu je v aplikaci podporováno jediné dítě, to je důvod, proč se v modelu uvádí, že může mít rodič maximálně jedno dítě.

2.1.1.3 Profile

Entita Profile se vztahuje vždy k jednomu dítěti. Tvoří profil dítěte, který je použit napříč aplikací. Samotnou entitu vytváří rodič na základě informací o jeho potomkovi. Informace o měsíci a roku narození jsou využívány při doporučování vhodného kvízu rodiči.

Název atributu	Popis
name	Jméno dítěte
birthMonth	Měsíc, kdy se dítě narodilo
birthYear	Rok narození dítěte

Tabulka 2.1: Popis atributů entity Profile

2.1.1.4 User account

Tato entita označuje uživatelský účet rodiče. Účet vzniká na základě e-mailové adresy a hesla. Protože při využívání aplikace není nutné, aby byl uživatelský účet vytvořen, ani v doménovém modelu není pro entitu Parent povinná vazba na entitu User account.

Název atributu	Popis
e-mail	E-mailová adresa uživatele
password	Nastavené heslo k uživatelskému účtu

Tabulka 2.2: Popis atributů entity User account

2.1.1.5 PIN code

Entita PIN code reprezentuje PIN kód, který je nutný při přepínání z dětského do rodičovského módu. Rodič nejprve provede nastavení jím zvoleného kódu. Následně při přepínání do rodičovského módu je rodič požádán o zadání PIN kódu, aby byl zamezen přístup dítěti do části s nastavením aplikace.

Název atributu	Popis
code	Čtyřmístný číselný kód uživatele

Tabulka 2.3: Popis atributů entity PIN code

2.1.1.6 Play time

Play time označuje časový limit, po kterém bude dítěti spuštěna kvízová část. Z celkového časového limitu postupně ubývá čas, pokud má dítě spuštěnou některou ze sledovaných aplikací. Po uplynutí celkového časového limitu jsou dítěti zablokovány veškeré sledované aplikace, dokud nedokončí kvíz. Poté je zbývající čas resetován na nastavenou hodnotu.

Název atributu	Popis
limit	Celkový časový limit, který má dítě nastavený
remainingTime	Zbývající čas do spuštění kvízu

Tabulka 2.4: Popis atributů entity Play time

2.1.1.7 Timetable

Entita Timetable označuje časový rozvrh, který je nastaven pro dítě. Časový rozvrh vytváří rodič. Pokud je Timetable označen jako neaktivní, nedochází k jeho spuštění přesto, že by měl být podle časového intervalu a dne aktivován.

Název atributu	Popis
title	Název daného časového rozvrhu
from	Čas, od kdy je rozvrh aktivní
to	Čas, do kdy je rozvrh aktivní
days	Množina dní, kdy se daný rozvrh spouští
activity	Označení, jestli se časový rozvrh má spustit v danou dobu

Tabulka 2.5: Popis atributů entity Timetable

2.1.1.8 Application

Entita označuje veškeré nainstalované aplikace v mobilním zařízení. K aplikacím se vztahují jednotlivé aplikační statistiky, které jsou získávány při přepnutí zařízení do dětského módu.

Název atributu	Popis
name	Název aplikace

Tabulka 2.6: Popis atributů entity Application

2.1.1.9 Monitored application

Entita reprezentuje aplikaci, která je označena jako sledovaná. Během trávení času ve sledované aplikaci dítěti ubývá čas z celkového limitu. Po jeho vypršení dojde ke spuštění kvízu.

2.1.1.10 Activity statistics

Entita Activity statistics vyjadřuje jednotlivé statistiky, které vznikají během využívání mobilního zařízení dítětem. Jedná se o statistiku, která udává čas strávený dítětem v jednotlivých aplikacích, případně jejich kategoriích.

Název atributu	Popis
spentTime	Čas strávený při dané činnosti

Tabulka 2.7: Popis atributů entity Activity statistics

2.1.1.11 Last activity

Tato entita označuje poslední aktivitu dítěte, která by měla být zobrazena rodiči. V aktuálním návrhu je poslední aktivita dítěte reprezentována pouze informacemi o posledních vyplněných kvízech.

Název atributu	Popis
date	Datum záznamu o aktivitě

Tabulka 2.8: Popis atributů entity Last activity

2.1.1.12 *Quiz*

Entita Quiz označuje kvíz, který dítě vyplňuje. Kvíz si dítě může spustit samo nebo mu je spuštěn automaticky po uplynutí časového limitu.

Název atributu	Popis
title	Název kvízu
description	Popis kvízu (zaměření, doporučený věk)

Tabulka 2.9: Popis atributů entity Quiz

2.1.1.13 *Quiz question*

Quiz question reprezentuje jednu otázku, která je součástí kvízu. Na tyto otázky dítě vytváří odpovědi.

Název atributu	Popis
questionText	Text kvízové otázky

Tabulka 2.10: Popis atributů entity Quiz question

2.1.1.14 *Quiz answer*

Entita vyjadřuje odpověď dítěte na určitou otázku v rámci kvízu. V této entitě je uvedeno, zda je daná odpověď na otázku správná nebo ne.

Název atributu	Popis
correct	Udává, zda je daná odpověď správná

Tabulka 2.11: Popis atributů entity Quiz answer

2.1.1.15 Reward

Entita Reward označuje odměnu pro dítě. Odměnu přidává rodič za účelem motivování dítěte. Zároveň je rodič upozorněn, pokud dítě danou odměnu zakoupilo.

Název atributu	Popis
title	Název dané odměny
description	Bližší popis odměny
price	Cena, za kterou dítě může odměnu zakoupit

Tabulka 2.12: Popis atributů entity Reward

2.1.2 Shrnutí

Díky doménovému modelu jsem získal základní strukturu, která je následně využita při samotné implementaci. Na první pohled jsou vidět vazby mezi entitami, které se objevily během analyzování problematiky. Model zároveň obsahuje seznam dat, která jsou v rámci jednotlivých entit podstatná a tvoří základ pro atributy tříd.

2.2 Využité technologie

V následující kapitole se zaměřím na technologie, které jsou použity při vývoji Android aplikace Educhild. Představím vybrané vývojové prostředí a programovací jazyk, případně jejich nejvýznamnější alternativy, které je také možné zvolit. Následovat budou nejdůležitější komponenty, které jsou využity během vývoje Android aplikací a kterým je třeba porozumět.

2.2.1 Vývojové prostředí

Dnes je možné si při vývoji Android aplikace vybrat z několika alternativ vývojových prostředí. Výběr správného IDE, může být obecně náročný, nicméně sám Google doporučuje při vývoji Android aplikace využití pouze jedné z nich, konkrétně Android Studia [20].

2.2.1.1 Android Studio

Jedná se o vývojové prostředí primárně určené pro vývoj aplikací pro operační systém Android, které je založené na dále popsaném IntelliJ IDEA. Na rozdíl od něj přidává mnoho užitečných funkcí, které výrazně usnadňují vývoj pro operační systém Android a zvyšují efektivitu. Řadí se mezi ně flexibilní systém založený na Gradle, který umožňuje build, rozšířené testovací nástroje a frameworky, nástroje typu Lint pro kontrolu problémů s výkonností, použitelností a kompatibilitou verzí a v neposlední řadě také emulátor, díky němuž může vývojář testovat vyvíjenou aplikaci bez nutnosti mít vlastní zařízení s operačním systémem Android. Emulátor zároveň dovoluje simulovat různé konfigurace. Další užitečnou komponentou je Visual layout editor, díky kterému je možné vytvářet rozsáhlejší a komplexnější rozložení obrazovek s ohledem na jejich velikost [20].

2.2.1.2 IntelliJ IDEA

IntelliJ IDEA je IDE od společnosti JetBrains, ve kterém je také možné vytvářet Android aplikace. Podobně jako předchozí Android Studio i IntelliJ IDEA disponuje UI designerem, který slouží k vytváření složitějších rozvržení obrazovek. Na rozdíl od Android Studia není toto IDE dostupné zcela zdarma [21].

2.2.1.3 Visual Studio

Další možností je využití produktu od společnosti Microsoft. Stejně jako Android Studio disponuje Visual Studio emulátorem pro testování. Výhodou tohoto IDE je možnost vyvíjení aplikace současně pro různé operační systémy. Díky tomu je možné zároveň vyvíjet aplikaci pro Android, stejně tak jako pro iOS, případně i Windows. Další výhodou je možnost využití Unity nástrojů pro snadnější práci při vývoji graficky náročnějších aplikací, případně her [22].

2.2.1.4 Shrnutí

Kromě výše uvedených nejvýznamnějších IDE existují ještě další alternativy, které je možné využít k vývoji Android aplikací, ale žádná nepřináší podstatnou výhodu.

Na počátku vývoje aplikace Educhild bylo rozhodnuto o využívání Android Studia. K tomuto rozhodnutí vedly především dva důvody. Prvním z nich je, že Android Studio disponuje užitečnými funkcionalitami a prvky, které během vývoje výrazně usnadňují práci. Druhým důvodem je, že se jedná o vývojové prostředí, které je primárně určené pro vývoj Android aplikací a i Google jeho využití doporučuje.

2.2.2 Programovací jazyk

Před začátkem implementace je nutné zvolit vhodný programovací jazyk, ve kterém bude psán kód aplikace. V současné době se pro vývoj Android aplikací běžně používá zejména Java nebo Kotlin. Oba tyto jazyky jsou podporovány v rámci Android Studia [23]. Pro aplikaci Educhild byl vybrán programovací jazyk Kotlin především z níže popsaných důvodů.

2.2.2.1 Java

Java byla v minulosti oficiálním jazykem pro vývoj Android aplikací [24]. I v současné době se ovšem jedná o velice oblíbený a využívaný jazyk. Největší výhody tvoří rozsáhlá uživatelská základna, která za roky její existence vznikla, velký počet frameworků a množství knihoven.

2.2.2.2 Kotlin

Je to právě Kotlin, který v roce 2019 nahradil Javu v roli oficiálního jazyku pro vývoj Android aplikací [24]. Jedná se o poměrně nový programovací jazyk, který tvoří alternativu k dlouhou dobu využívané Javě. V současné době Kotlin používá přibližně 60 % Android vývojářů [25]. Stejně jako Java i Kotlin cílí na JVM, proto je možné, aby v rámci jedné aplikace koexistovaly. To umožňuje využití knihoven, které jsou napsané v Javě, i v projektu, jež je primárně psán v jazyku Kotlin. Největší výhodou Kotlinu oproti Javě je jeho stručnost. Existuje velký rozdíl v celkovém množství kódu, které mají obdobné projekty napsané v Javě a v Kotlinu [26].

Kotlin zároveň obsahuje Coroutines, které usnadňují psaní asynchronního a neblokujícího kódu. Jedná se o jakási lehká vlákna, díky nimž je možné spravovat dlouhodobě běžící úkoly na pozadí, které by jinak blokovaly hlavní vlákno [27].

2.2.3 Programování pro Android

V následující kapitole se zaměřím na nejdůležitější komponenty, se kterými se během vývoje Android aplikace vývojář setká. Tyto prvky tvoří základ každé Android aplikace a bez správného pochopení, jak jednotlivé části fungují, není možné vytvořit produkt, který by se choval tak, jak je zamýšleno.

2.2.3.1 Activity

Activity tvoří důležitou část Android aplikací. Způsob, kterým jsou Activity spouštěny a jak jsou složeny dohromady, tvoří základ celé aplikace. Na rozdíl od zaběhlých programovacích paradigmat, ve kterých jsou aplikace spouštěny za pomoci metody `main()`, systém Android inicializuje kód v instanci Activity voláním specifických callback metod, které odpovídají konkrétním fázím životního cyklu [28].

Activity poskytuje jakési okno, do kterého může aplikace zobrazovat prvky UI. Typicky toto okno pokrývá celou obrazovku, ovšem není to podmínkou, je možné, aby bylo menší než obrazovka a bylo zobrazeno v popředí ostatních.

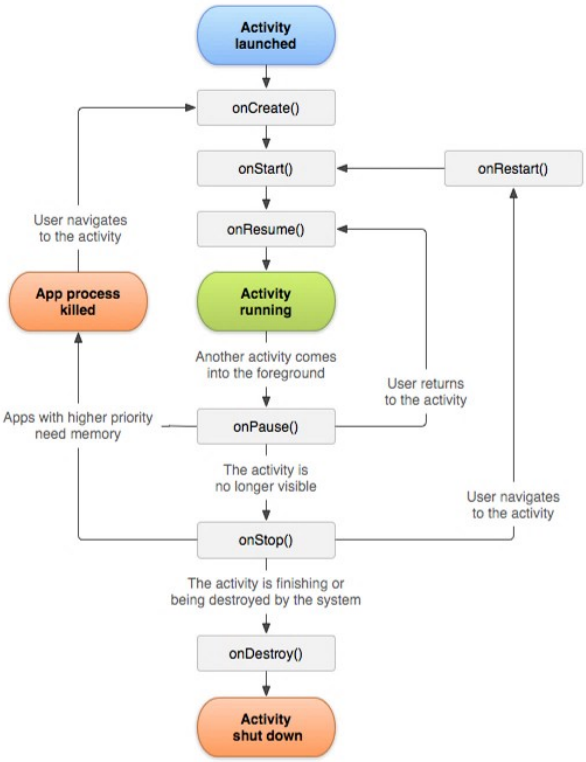
Ve většině případů je jedna Activity označena jako *Main Activity*. Ta se objeví jako první po spuštění aplikace. Každá Activity poté může spouštět libovolný počet dalších. I přesto je díky volné provázanosti docílena minimální závislost mezi nimi [28].

V průběhu toho, jak uživatel prochází v rámci Activity, se postupně mění její stav [3]. Ke spravování přechodů mezi jednotlivými stavy je opět použita série callback metod. Tyto metody spolu s odpovídajícími stavy jsou vyobrazené na obrázku 2.2.

2.2.3.2 Fragment

Fragment reprezentuje znovupoužitelnou část UI v rámci aplikace. Definuje a spravuje své vlastní rozložení, může reagovat na vstupy a podněty od uživatele a definuje i svůj vlastní životní cyklus [29]. Přináší možnost rozdělit UI celé Activity do určitých částí. Je vhodné v rámci jedné komponenty Fragment spravovat UI jedné obrazovky, případně její části. Ke každé komponentě Fragment se váží View, které reprezentují základní stavební blok všech prvků v rámci UI. Životní cykly Fragment a View a jejich vzájemná závislost je zobrazena na obrázku 2.3.

Fragment nemůže v rámci aplikace existovat sám o sobě, vždy je vázán na Activity případně jiný Fragment. Pokud je Activity v rámci svého životního cyklu ve fázi `STARTED` nebo vyšším, je možné komponenty Fragment přidávat, nahrazovat nebo odstraňovat. V rámci Activity je pak vhodné definovat navigaci mezi jednotlivými komponentami Fragment [29].



Obrázek 2.2: Životní cyklus Activity, převzato z [3]

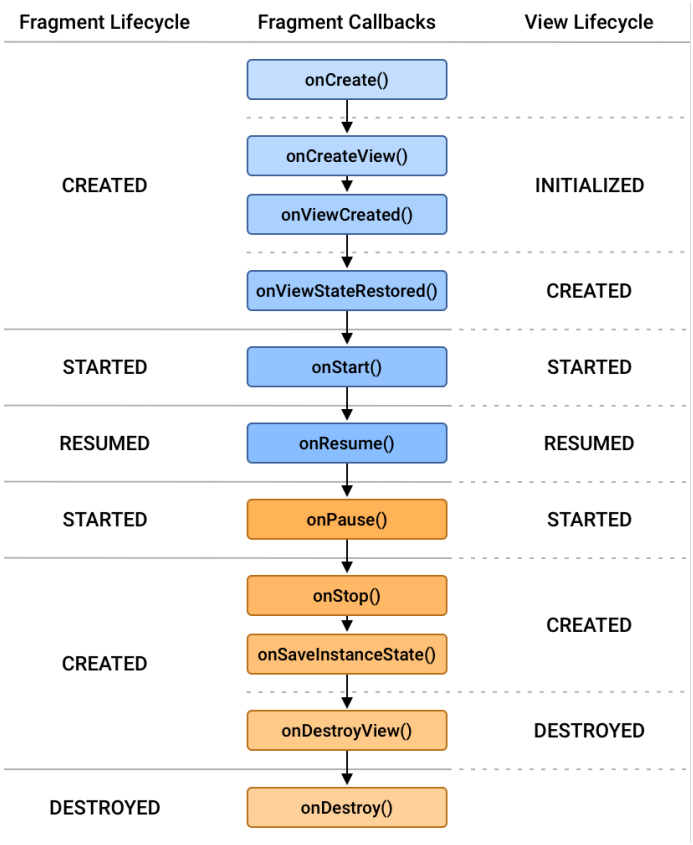
2.2.3.3 Service

Service je součást aplikace, která vykonává dlouhodobě běžící procesy na pozadí. Na rozdíl od komponent Activity a Fragment nezajišťuje Service uživatelské rozhraní. Pokud je Service spuštěná, může zůstat běžet na pozadí, a to i poté, co uživatel změní aktuálně zobrazovanou aplikaci [30].

Aktuálně se při vytváření Android aplikací využívají zejména tři typy Service [30]:

- **Foreground** - Tato Service zajišťuje operace, které jsou pozorovatelné pro uživatele. Podmínkou je zobrazení Notification, aby uživatel věděl, že na pozadí stále běží daná Service.

- **Background** - Díky této Service je možné provádět operace, které nejsou přímo pozorovatelné pro uživatele.



Obrázek 2.3: Životní cyklus Fragment a View, převzato z [4]

- **Bound** - Service je *bound*, pokud je připnutá k jiné komponentě aplikace pomocí příkazu `bindService()`. Tato komponenta běží tak dlouho, dokud existuje komponenta, ke které je připnutá.

2.3 Architektura

Volba správné architektury je jednou z nejpodstatnějších částí vývoje [31]. Klíčem jednoduchého vývoje aplikace je, aby veškeré komponenty byly správně organizované. Mezi nejčastější chyby, které vedou k neúspěchu projektu, patří zvolení nevhodné, případně žádné architektury.

Pokud základ aplikace netvoří dobrá architektura, s téměř stoprocentní jistotou nastanou během vývoje problémy, na které se bude jen velice těžko hledat přijatelné řešení. Nesprávně zvolená architektura vede k špatné organizaci komponent, z čehož vyvstává mnoho problémů. Kvůli velké závislosti mezi prvky není možné jednoduše odstraňovat chyby, které přirozeně během vývoje vznikají, případně není možné komponenty pokrývat základními unit

testy. Pro většinu vývojářů bude nesmírně složité udržovat aplikaci, případně přidat i jednoduchou funkčnost. Díky tomu je téměř nemožné postupné rozšiřování vývojářského týmu.

Z předchozího vyplývá, že pro vývoj kvalitní aplikace je klíčové vybrat správnou architekturu. Výběr je potřeba důkladně zvážit kvůli výsledné jednoduchosti, snadné testovatelnosti a údržbě aplikace.

V současné době jsou nejpoužívanější tři typy architektonických vzorů, které jsou vhodné k využití v rámci Android aplikace. Jedná se o MVC (Model-View-Controller), MVP (Model-View-Presenter) a MVVM (Model-View-ViewModel) [31].

2.3.1 MVC

Tento architektonický vzor se skládá ze tří vrstev. Vznikl kvůli oddělení business logiky od uživatelského rozhraní. Vzor MVC byl rozšířen v aplikacích zejména v úplných začátcích Androidu. V prvních Android aplikacích se jednalo o nejpřirozenější volbu, neboť MVC byl jeden z nejpopulárnějších vzorů v té době [32].

Vrstva s označením Model je vrstva datová. Stará se o spravování business logiky a zároveň řízení komunikace s databázovým API, případně network komunikace. View je vrstva, která se stará o vizualizaci dat, jež získává přímo z modelové vrstvy. V Android světě by ji měly tvořit komponenty Activity, Fragment a View. Vrstva Controller zabezpečuje správnou reakci na chování uživatele tím, že podle definovaných pravidel upravuje Model dle potřeb.

Mezi největší výhody tohoto vzoru patří možnost separace jednotlivých záležitostí do daných vrstev, čímž je zajištěna snadnější testovatelnost vrstev Controller a Model a celková rozšiřitelnost aplikace.

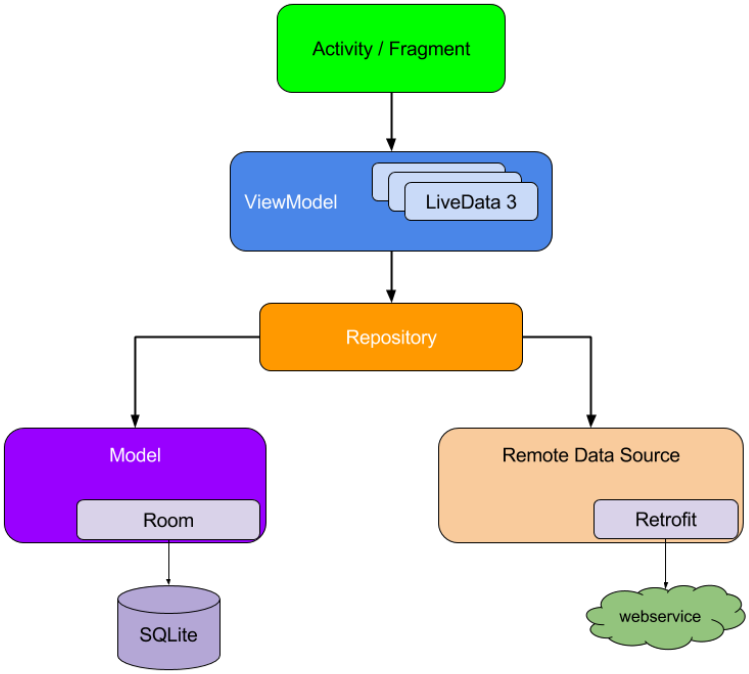
Nevýhody pramení ze závislosti View na vrstvách Controller a Model. Současně díky tomu, že vrstva View přijímá data od vrstvy Model, musí do jisté míry obsahovat UI logiku, čímž je téměř znemožněno její unit testování.

2.3.2 MVP

Vzor MVP je derivací architektonického vzoru Model-View-Controller, který se v dnešní době používá mnohem častěji. Na rozdíl od něj zde Presenter tvoří jakéhosi „prostředníka“ mezi dalšími dvěma vrstvami. Díky tomu může být vrstva View úplně oddělena od vrstvy Model [33].

Presenter je tedy zodpovědný jak za interakci s uživatelem, tak za formátování dat pro View. Jeho úkolem je úprava dat v rámci vrstvy Model, ale stejně tak i dat pro vrstvu View, která je zbavena nutnosti zpracovávání UI logiky a stará se jen o zavolání správné metody z Presenter pokaždé, když uživatel provede akci.

Díky tomu je jednodušší vytvářet jednotkové testy i přesto, že jsou zachovány veškeré výhody návrhového vzoru MVC.



Obrázek 2.4: Diagram MVVM architektury, převzato z [5]

2.3.3 MVVM

Posledním architektonickým vzorem, který představím je MVVM [5]. Tento vzor Google uvádí jako doporučený architektonický vzor pro vývoj Android aplikací. Stejně jako u předchozích i zde Model představuje vrstvu s veškerou business logikou a daty, se kterými aplikace pracuje. View je vrstva uživatelského rozhraní.

ViewModel obsahuje data pro specifické UI komponenty, která jsou součástí vrstvy View, a zároveň obsahuje logiku pro práci s daty nutnou pro komunikaci s vrstvou Model. ViewModel na rozdíl od předchozích vzorů neobsahuje napojení na vrstvu View. Naopak data jsou zde reprezentována pomocí komponent LiveData, které mohou být monitorovány prvky z vrstvy View.

LiveData je komponenta, která je schopna uchovávat data. Ostatní komponenty v aplikaci mohou monitorovat změny, které se s daty dějí bez nutnosti vytvoření explicitní a rigidní závislosti mezi nimi. Proto mohou být komponenty v rámci View vrstvy informovány, při aktualizaci dat a mohou na tuto změnu reagovat jejich zobrazením [5].

2.3.4 Shrnutí

Pro vývoj aplikace Educhild byl vybrán architektonický vzor MVVM. V současné době se jedná o nejrozšířenější a nejvíce používaný vzor, který lze aplikovat v rámci vývoje Android aplikace. Zároveň jde o oficiální vzor pro vývoj Android aplikací doporučený společností Google [5]. Díky MVVM vzoru a oddělení jednotlivých vrstev, které přináší, je zajištěna nenáročná rozšiřitelnost a udržovatelnost aplikace.

2.4 Framework

Framework je softwarová sada nástrojů, která vývojářům umožňuje sestavit výsledný produkt podle požadavků. Tvoří jakousi páteř aplikace, která je pouze doplněna o specifické funkce, případně grafickou část. Framework je vždy navržen tak, aby zjednodušil proces vývoje aplikace a usnadnil správu a opravy chyb, které se později objeví [34].

V současné době existuje nepřeberné množství různých frameworků, které disponují jedinečnými užitečnými funkcionalitami. Tyto funkcionality se mohou hodit pro jeden typ aplikace, ovšem pro jiný nemusí být vůbec vhodné. Proto je nutné, aby byl správný framework vybrán na základě konkrétních požadavků na danou aplikaci.

Pro aplikaci Educhild byl na základě dříve definovaných požadavků vybrán framework Teanity, který obsahuje řadu užitečných prvků popsaných dále.

2.4.1 Teanity

Teanity poskytuje širokou škálu pomocných nástrojů a podpůrných architektonických návrhů. Fungování tohoto frameworku, který je psán výhradně v programovacím jazyku Kotlin, plně závisí na Coroutines a Android Lifecycle knihovnách. Teanity rozděluje celou aplikaci do několika modulů [35]:

- **Components module** - Díky tomuto modulu je možné vytvářet jednodušší a testovatelnější kód v rámci aplikace. Veškerá logika, která není spojená s UI je rozdělena do jednotlivých UseCase, kde každý má specifický účel. Tyto UseCase je možné dále skládat dohromady do složitějšího UseCase za účelem řešení náročnějších problémů.
- **Core module** - Tento modul zpracovává komunikaci mezi View a ViewModel a poskytuje pokyny k tomu, jak by měla tato komunikace vypadat. Je zde podpora pro tři typy komponent, konkrétně Activity, Fragment a DialogFragment.
- **DI module** - Zde jsou zajištěny veškeré potřebné závislosti tak, aby výsledná provázanost zůstala co nejnižší. Doporučené je využívání *Koin* závislostí obvyklých pro vývoj Android aplikací.

- **Network module** - Network module poskytuje nejčastěji využívané prvky, které jsou použity při volání API, jako například zpracování výjimek a chyb, které mohou nastat.
- **Persistance module** - Nejdůležitějším prvkem v tomto modulu je poskytování interface BaseDao. Na jeho základě je možné následně automaticky generovat implementace konkrétních tříd, které slouží jako vstupní bod do databáze.
- **UI module** - Poslední modul poskytuje nástroje pro tvorbu UI a vytváření animací.

Kapitola 3

Realizace

Po dokončení fází analýzy a návrhu mám vypracované veškeré části, jež jsou podstatné pro úspěšné zvládnutí další fáze, kterou je realizace. Cílem této fáze je převést prvky, které byly dříve navrženy, na funkční systém splňující dané požadavky [36].

Protože se jedná o rozsáhlý projekt, byly do první verze aplikace za účelem včasného vydání vybrány v rámci implementace jen některé prvky. V této kapitole se budu věnovat zejména popisu implementace jednotlivých částí první verze aplikace Educhild. Bližší popis první verze se nachází v kapitole 3.2. Nejprve představím obecné schéma, ze kterého budou vycházet implementace všech částí rodičovského módu a celkově aplikace Educhild.

3.1 Obecné schéma

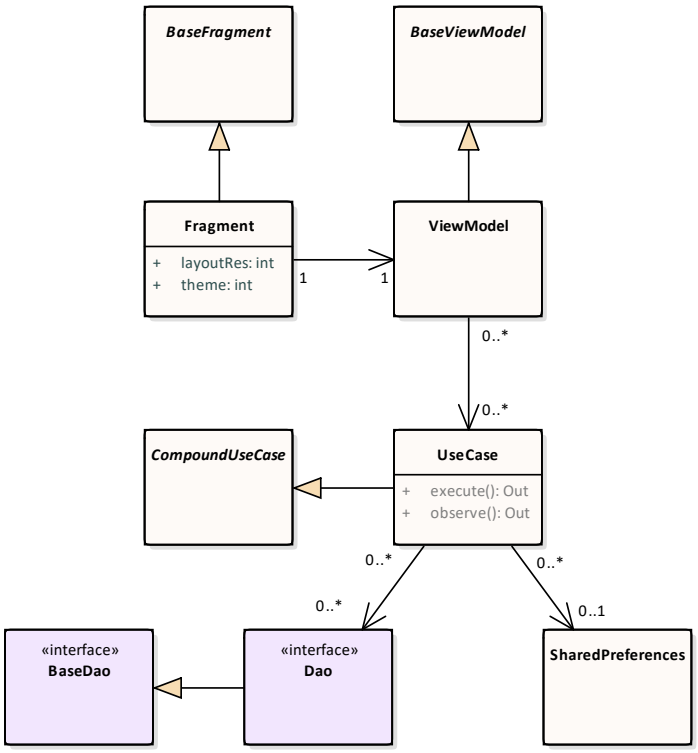
Než začnu s vlastním popisem implementace, představím třídy a další komponenty, které využívám během realizace jednotlivých navržených funkcí a obrazovek. Obecné schéma přidávám za účelem snazšího pochopení vztahů mezi jednotlivými třídami a rozdělení funkčnosti. Zjednodušený diagram tříd je na obrázku 3.1.

Závislosti mezi třídami jsou řešeny za pomoci *dependency injection*. Díky tomu je zajištěna menší provázanost a také snazší testovatelnost jednotlivých tříd [37].

V následujících sekcích se zaměřím na popis konkrétních tříd, které se v rámci diagramu tříd objevují, a představím jejich nejpodstatnější prvky, jež se při práci s nimi využívají.

3.1.1 Fragment

Jak bylo popsáno v předešlé kapitole, Fragment reprezentuje znovupoužitelnou část uživatelského rozhraní v rámci aplikace [29]. Z pravidla je implementována jedna třída `Fragment` ke každé obrazovce. Základ tříd tvoří abstraktní



Obrázek 3.1: Zjednodušený model tříd

třída `BaseFragment`, ze které dědí veškeré ostatní třídy Fragment. V této třídě je přepsána metoda `onCreateView`, za účelem nastavení správného grafického tématu aplikace, který je definován v rámci souboru *styles*. V jednotlivých instancích odpovídajících obrazovkám stačí pouze přepsat proměnnou `theme` a nastavit téma této obrazovky, které odpovídá požadovanému umístění v rodičovském módu.

Dalším typem komponenty Fragment, který se v aplikaci Educhild objevuje, je `DialogFragment`. Tento typ nezabírá celou obrazovku, ale pouze její část. Využívá se k zobrazení dialogových oken v rámci aplikace. I zde je vytvořena abstraktní třída `BaseDialogFragment`.

Každý Fragment má deklarované rozložení obrazovky, které je označováno jako Layout. Ten definuje strukturu pro uživatelské rozhraní v rámci aplikace. Veškeré komponenty, které jsou součástí Layout, používají View a ViewGroup hierarchii. View obvykle představuje komponentu, která se přímo vykresluje uživateli a interaguje s ním, například `TextView`, nebo `Button`. Naopak ViewGroup je jakýsi neviditelný *„kontejner“*, který definuje strukturu jednotlivých

View, případně dalších ViewGroup [38]. Hlavním ViewGroup, který je v aplikaci využíván, je `ConstraitLayout`. Jeho největší výhodu tvoří možnost vytváření rozsáhlých rozložení, bez nutnosti používání dalších ViewGroup. Rozestavení jednotlivých View v rámci `ConstraitLayout` je realizováno pomocí relativních vzdáleností mezi nimi. Díky Layout Editor, kterým disponuje i Android Studio, se jedná o nejjednodušší způsob vytváření Layout, protože každou komponentu je možné správně umístit pomocí jednoduchého „*drag-and-drop*“ namísto upravování XML souboru s rozložením obrazovky [39].

3.1.2 ViewModel

ViewModel slouží k ukládání a spravování dat, která jsou spojena s UI. V této třídě vždy dochází k přípravě dat, jež mají být uživateli zobrazena. Druhou funkcí je reagování na vstupy uživatele nejčastěji spouštěním asynchronních akcí [40].

V rámci projektu je podobně jako u komponenty Fragment definována abstraktní třída `BaseViewModel`, ze které dědí všechny třídy ViewModel, jež jsou v rámci aplikace implementovány.

K zobrazování dat je využita *Data Binding* knihovna. Tato knihovna umožňuje propojení UI komponent se zdroji dat [41]. Díky komponentám, které podporují *Data Binding*, je možné výrazně zjednodušit implementace Activity a Fragment, čímž se stávají jednodušší na údržbu. Knihovna umožňuje propojení atributů příslušného ViewModel přímo s jednotlivými View tak, že automaticky generuje třídy, které jsou potřebné pro toto propojení. Proměnné, které mají být využity jako výrazy pro *Data Binding*, jsou v rámci Layout definované uvnitř `data` komponenty, která je na stejné úrovni jako základní ViewGroup. Oba tyto elementy jsou zabalené pomocí tagu `layout`.

Knihovna také podporuje využití *Two-way data binding*, které umožňuje zároveň zobrazovat správná data při aktualizaci atributů, ale také přímo reagovat na změny, jež provádí uživatel, a případně podle toho upravovat data v atributech [41].

Pro každý prvek Layout existuje *Binding adapter*, který nastavuje danou hodnotu, případně reaguje na vstupy uživatele. Kromě těch, které jsou už připravené v implementaci jednotlivých View, je možné vytvářet vlastní adaptéry, které budou nastavovat potřebné hodnoty [41]. Příklad vytvořeného adaptéru pro nastavení textu do komponenty `TextView` obsahuje ukázka zdrojového kódu 1.

K vykonání asynchronních akcí, které jsou v rámci ViewModel obvykle spouštěny kvůli provedení jednoho nebo více UseCase, jsou využity *Coroutines*. Jedná se o *Concurency* návrhový vzor, který byl přidán do programovacího jazyku Kotlin za účelem zjednodušení kódu vykonávaného asynchronně [27]. V rámci tříd ViewModel jsou *Coroutines* spouštěny za pomoci `viewModelScope`, konkrétně funkce `launch`, která vytvoří danou *Coroutine* a odešle funkci, jež je součástí jejího těla, k příslušnému *dispatcher*, kde do-

```
@BindingAdapter("android:text")
fun TextView.setText(text: Text?) {
    text?.let {
        setText(text.getText(context.resources))
    }
}
```

Ukázka zdrojového kódu 1: *Binding adapter* pro nastavení textu

jde k jejímu vykonání. `ViewModelScope` je předdefinovaný `CoroutineScope`, který se stará o spravování jedné nebo více *Coroutine* [27]. Každá *Coroutine* musí běžet v daném *scope*.

3.1.3 UseCase

V rámci UseCase je obsažena logika, která není přímo spojena s UI. Každý UseCase má na starosti vykonávání právě jednoho úkolu [35]. UseCase je nejčastěji používán ke správě uložených dat pomocí interface DAO, případně přímo ze Shared Preferences. Další možností je implementace složitějšího UseCase, který v sobě používá jeden nebo více jiných UseCase, za účelem rozložení složitého problému a tedy vykonávání právě jednoho úkolu. Každý implementovaný UseCase dědí z abstraktní třídy `CompoundUseCase`.

V týmu bylo rozhodnuto o využívání *Room persistance library* k ukládání dat. Bližší informace o této knihovně a jejím používání je možné nalézt v bakalářské práci Miloše Popoviče. K přístupu k uloženým datům v rámci *Room* je vždy implementováno příslušné DAO. Každé DAO obsahuje způsob abstraktního přístupu k datům uloženým v databázi. V době kompilování *Room* automaticky generuje na základě těchto abstraktních deklarací konkrétní implementace jednotlivých DAO [42]. V rámci aplikace je obdobně jako pro UseCase a předešlé třídy definováno základní interface `BaseDao`.

Druhou možností k ukládání dat jsou Shared Preferences. Ty slouží k uložení relativně malé kolekce klíčových dat. Objekt `SharedPreferences` obsahuje jednoduché metody, určené k zápisu a čtení dat, které jsou součástí souboru s klíčovými hodnotami [43]. Nejjednodušší způsob, jakým je možné `SharedPreferences` v rámci aplikace realizovat, je knihovna *Kotpref* [44]. Jedná se o knihovnu určenou pro programovací jazyk Kotlin, která se stará o jednoduché spravování Shared Preferences v rámci aplikace. Základní modul knihovny *Kotpref* obsahuje podporu standardních typů proměnných, které se v Shared Preferences objevují, jako jsou `String`, `Int` nebo `Long`, ale existují i další rozšiřující moduly [45]. V rámci aplikace je použit rozšiřující modul, který přidává podporu pro objekty typu `Enum`.

3.2 První verze

Protože je projekt Educhild velmi rozsáhlý, není možné v tak krátkém čase vyvinout kvalitní aplikaci se všemi požadovanými funkcemi. Proto byla na základě rozhodnutí týmu společně se zadavateli pečlivě vybrána první verze aplikace, ve které nejsou implementovány všechny požadavky, jež se během analýzy objevily. Funkcionality, které vznikly na základě těchto požadavků, budou součástí následující verze aplikace Educhild, proto již byla zpracována předchozí analýza společně s nimi.

Konkrétně v rodičovském módu není v rámci první verze implementován systém odměn, který není pro fungování aplikace nezbytně nutný, ale jedná se spíše o další motivaci pro děti, aby plnily kvízy i samostatně. Přestože implementace odměn není součástí této verze, je vše připraveno ke snadnému přidání potřebných funkcionalit. Hlavní důvod, proč zatím nebyly odměny do rodičovského módu přidány, přestože se nejedná o náročný úkol, je prozatím chybějící podpora na serveru v první verzi, díky čemuž je do jisté míry omezeno jejich používání.

Součástí první verze z pohledu rodičovského módu zůstaly následující funkcionality:

- přihlašování a registrace,
- nastavení a zadávání PIN kódu,
- úvodní nastavení aplikace (včetně základního a pokročilého nastavení),
- výběr monitorovaných aplikací,
- zobrazování aplikačních statistik,
- spravování časového limitu,
- vytváření a správa časových rozvrhů,
- zobrazení posledních aktivit dítěte,
- další nastavení (nastavení uživatelského účtu, spravování profilu dítěte apod.).

V následujících kapitolách popisuji implementaci jednotlivých funkcionalit, kterými disponuje rodičovský mód v rámci první verze aplikace Educhild. Protože popis veškeré implementace těchto funkcionalit by byl velice zdlouhavý, zaměřuji se především na vybrané detaily, které jsou pro jejich uskutečnění nejpodstatnější, případně takové, které je nutné blíže představit.

3.2.1 Přihlašování a registrace

Úvodní obrazovky, které jsou uživateli zobrazeny po povolení potřebných oprávnění a odesílání anonymních dat, jsou autentizační obrazovky. Na první z nich uživatel volí, zda se chce přihlásit, v případě že uživatelským účtem již disponuje, registrovat, pokud si chce vytvořit nový uživatelský účet, nebo autentizaci přeskočit a pokračovat pouze s lokálním účtem.

Pokud uživatel zvolí registraci, je po něm požadováno vyplnění e-mailové adresy a hesla do určených textových polí realizovaných pomocí komponenty uživatelského rozhraní `EditText`. V případě emailové adresy je zde nastaven atribut `android:inputType` jako `textEmailAddress`, naopak v případě hesla jako `textPassword`. Díky tomu se jednotlivá písmena z hesla zobrazují pouze jako černé body, aby nebylo možné vidět zadané heslo a to zůstalo soukromé. Zadání hesla je z důvodu nechtěného překlepu požadováno dvakrát, jak bývá obvyklé při každé registraci.

V případě, že se uživatel chce přihlásit ke svému stávajícímu účtu, musí vyplnit do obdobných textových polí jako v předchozím případě e-mailovou adresu a heslo zadané při registraci. Pokud uživatel zapomene heslo ke svému stávajícímu účtu je možné přejít na obrazovku *Forgotten password*, kde je možné po zadání e-mailové adresy své heslo obnovit.

Samozřejmostí je reagování na chybně zadané heslo, e-mailovou adresu nebo na chybu, která během autentizace může nastat v podobě errorových hlášení, jež jsou zobrazována nad textovými poli, kterým navíc přibude červené ohraničení, aby bylo zvýrazněno, kde se daná chyba nachází.

Samotná realizace autentizace je v aplikaci Educhild řešena za pomoci Firebase Authentication.

3.2.1.1 Firebase Authentication

Firebase Authentication poskytuje backend služby, SDK a UI knihovny, které jsou využity během autentizace uživatele v rámci aplikace [46]. Podporuje přihlášení pomocí několika možností, od klasického s využitím e-mailové adresy a hesla, až po přihlášení pomocí telefonního čísla nebo poskytovatele identity, kterými jsou například Google, Facebook nebo Twitter. Po úspěšném přihlášení nebo registraci je možné využít poskytnutý autentizační token k ověřování identity v backend službách aplikace [46].

V rámci aplikace Educhild je implementována podpora přihlášení pomocí e-mailové adresy a hesla. Poté, co jsou tyto osobní údaje získány od uživatele jsou předány Firebase SDK, kde dojde k jejich ověření. Výsledná implementace UseCase starajícího se o přihlášení pomocí Firebase Authentication s využitím `suspendCoroutine` se nachází v ukázce zdrojového kódu 2. Registrace uživatele probíhá obdobně, pouze je použita jiná metoda z Firebase Authentication SDK, která jako parametry také přijímá e-mail uživatele a zadané heslo, konkrétně metoda `createUserWithEmailAndPassword(email, password)`.

```
class SignInWithFirebaseUseCase(
    private val firebaseAuth: FirebaseAuth
) : CompoundUseCase<Pair<String, String>, Unit>() {
    override suspend fun execute(
        input: Pair<String, String>
    ) = suspendCoroutine<Unit> {
        val email = input., first
        val password = input.second
        firebaseAuth.signInWithEmailAndPassword(
            email,
            password
        ).addOnCompleteListener { task ->
            if (task.isSuccessful) {
                it.resume(Unit)
            } else {
                it.resumeWithException(
                    task.exception ?: Exception()
                )
            }
        }
    }
}
```

Ukázka zdrojového kódu 2: Přihlášení pomocí Firebase Authentication

Firebase Authentication také podporuje resetování hesla, pomocí zaslání odkazu na e-mailovou adresu, kde je možné nastavit heslo nové. Tato funkcionalita je implementována obdobně jako v předchozích případech přihlášení a registrace, pouze je použita jiná metoda, kterou Firebase Authentication disponuje, konkrétně se jedná o metodu `sendPasswordResetEmail(email)`, které je jako parametr předána emailová adresa uživatele. Pokud uživatel skutečně vlastní danou e-mailovou adresu, dostane přístup k obnově hesla, pomocí kterého se následně může přihlásit.

3.2.2 PIN kód

PIN kód reprezentovaný čtyřmístným číslem, je v rámci aplikace požadován, pokud se uživatel chce přepnout z dětského do rodičovského módu. Díky tomu je zamezeno přístupu dítěte do rodičovského módu, kde se nachází nastavení aplikace.

I přesto, že uživatel zadává čtyřmístné číslo, je PIN kód na úrovni implementace reprezentován jako `String`, který je uložen v Shared Preferences. Díky tomu je možné využít při zadávání číslic prosté přidávání znaků do stá-

vajícího řetězce místo složitého přepočítávání. K zadávání PIN kódu byla vytvořena jednoduchá klávesnice společně s reprezentací kódu v podobě bodů, které mění tvar a barvu, aby uživatel přesně věděl, kolik čísel již zadal. Na tuto změnu je vytvořen vlastní *Binding adapter* pro komponentu `ImageView`, který na základě příznaku nastaví odpovídající pozadí.

Protože musí existovat několik obrazovek, na kterých se zadání PIN kódu bude objevovat (nastavení PIN kódu, změna PIN kódu a samotné zadání PIN kódu pro ověření), bylo pro klávesnici vytvořeno samostatné View, které je možné pomocí tagu `<include>` přidat do Layout souboru dané obrazovky. Aby bylo možné realizovat binding mezi View a ViewModel bylo nutné vytvořit abstraktní třídu `BasePinViewModel`, kterou následně rozšiřují třídy ViewModel, jež odpovídají daným obrazovkám, kde se View nachází. Konkrétní ViewModel je následně při přidávání View předán pomocí atributu `bind:viewModel`.

3.2.3 Úvodní nastavení

V úvodním nastavení uživatel prochází několika obrazovkami. Nejprve je mu zobrazena obrazovka s vyplněním informací k profilu dítěte. Následuje obrazovka s rozhodnutím, zda mají být nastaveny základní hodnoty aplikace automaticky nebo zda chce uživatel projít přes pokročilé nastavení. V případě, že si vybere *Advanced settings* je mu zobrazena navíc sekvence obrazovek, na kterých ručně vybírá monitorované aplikace a nastavuje časový limit a rozvrh pro dítě.

3.2.3.1 Profil dítěte

První z obrazovek úvodního nastavení je vytváření dětského profilu. Každý profil, jak bylo zmíněno již během tvorby návrhu, obsahuje jméno a rok a měsíc narození dítěte. Údaje o věku dítěte jsou podstatné během stahování a doporučování vhodných kvízů rodičům. K zadávání roku a měsíce narození je využito vlastního dialogu s komponentou `NumberPicker`. Aby bylo možné přistupovat k zadávaným datům zároveň z komponent, které náleží dialogu, a z komponent, které patří samotné obrazovce k vytváření profilu, využil jsem `SharedViewModel`. Díky tomu je možné, aby dvě a více komponent Fragment, které spolu potřebují komunikovat, sdílely jeden ViewModel [40].

Kromě uvedených dat je možné dítěti vybrat jeden z přednastavených profilových obrázků, případně nahrát obrázek vlastní. K zobrazení obrázku jsem využil knihovnu *CircleImageView*, která umožňuje vyobrazení kruhových obrázků, bez nutnosti jejich ořezávání [47]. Díky tomu je možné zobrazit obrázky, které mají klasický čtvercový, případně obdélníkový tvar, v kruhovém výřezu.

Nahrávání vlastního obrázku je podmíněno povolením příslušného oprávnění, v tomto případě možnosti číst externí úložiště. Výsledná funkce, která zajišťuje vybrání obrázku včetně žádání o oprávnění, je v ukázce zdrojo-

```
fun pickImage() = viewModelScope.launch {
        val grantedPermission = ActivityResultContracts.
        RequestPermission().await(
        android.Manifest.permission.READ_EXTERNAL_STORAGE
        )
        if (grantedPermission) {
            uploadedPhoto = ActivityResultContracts.
            GetContent().await("image/*")
        }
    }
```

Ukázka zdrojového kódu 3: Nahrání profilového obrázku

vého kódu 3. Nahraný obrázek je následně uložen do vnitřní paměti aplikace a v rámci třídy `Child`, která představuje profil dítěte, je zaznamenána cesta k tomuto obrázku, aby bylo možné jej následně načíst.

3.2.3.2 Primární nastavení

Součástí úvodního nastavení je také výběr, zda chce uživatel zvolit základní nastavení, které za něj provede nastavení časového limitu a označí aplikace s kategoriemi hry a sociální sítě jako monitorované, nebo pokročilé nastavení, ve kterém je uživatel proveden nastavením časového limitu, výběrem monitorovaných aplikací a tvorbou časového rozvrhu.

Při výběru základního nastavení je pro dané dítě aplikováno výchozí nastavení `MonitoredAppSettings`, ve kterém je zahrnuto automatické označení aplikací jako monitorovaných u těch, které mají danou kategorii. Kategorie jsou získávány pomocí `ApplicationInfo`, kde jsou zahrnuty informace, které lze o dané aplikaci získat. Tyto informace jsou získávány přímo z `AndroidManifest` souboru, kterým každá aplikace disponuje [48]. Konkrétně jde o informace uvedené v rámci tagu `<application>`, kde jedním z atributů je právě kategorie aplikace. V případě základního nastavení jsou jako monitorované aplikace brány ty s atributem kategorie `CATEGORY_GAME` a `CATEGORY_SOCIAL`.

V případě výběru pokročilého nastavení je vytvořena speciální navigace mezi třídami Fragment pomocí navigačních argumentů. Díky nim je možné předat v rámci navigace data mezi dvěma třídami Fragment [49]. Použití `SharedViewModel` zde není na místě, jelikož dané komponenty Fragment spolu přímo nesouvisejí a je nutné, aby existovaly zcela samostatně. Díky navigačním argumentům je předán příznak, zda je daná obrazovka aktuálně součástí průchodu pokročilým nastavením, a na základě toho může být upraven vzhled obrazovek a funkčnost některých komponent.

3.2.4 Monitorované aplikace

Jak jsem popsal už u doménového modelu, dítěti během spuštění monitorovaných aplikací ubývá čas z celkového limitu a po jeho úplném vypršení je mu spuštěn kvíz. Aby rodič mohl vybrat aplikace, které budou označeny jako monitorované, musí mít možnost zobrazit všechny nainstalované aplikace v zařízení.

Seznam všech aplikací je získáván pomocí třídy `PackageManager`, která je dostupná díky rozhraní `context` a jeho metodě `getPackageManager`. Třída `PackageManager` má metodu `queryIntentActivities`, která vrací seznam všech *Activity*, v tomto případě aplikací, které mají `Intent`, jež je předán jako parametr. `Intent` představuje abstraktní popis operace, která má být vykonána [50].

Aby byly vráceny veškeré aplikace, které jsou v zařízení nainstalované, a zároveň se nejednalo o systémové aplikace bez grafického rozhraní, jež není třeba monitorovat, je potřeba předat `Intent` se správnou akcí a nastavenou kategorií. Díky nastavené akci `ACTION_MAIN` získám veškeré aplikace. Pokud je zároveň přidána kategorie `CATEGORY_LAUNCHER` pomocí metody `addCategory`, jsou vybrány pouze aplikace, které jsou zobrazeny v top-level launcheru, to znamená přes celou obrazovku, a disponují svým vlastním grafickým rozhraním [50].

Díky předešlému získám seznam `ResolveInfo`, kde každá položka odpovídá jedné aplikaci v zařízení. Posledním krokem je získání `ApplicationInfo`, jež je součástí právě `ResolveInfo`, které obsahuje veškeré podstatné informace o aplikacích, například jejich název, *packageName* a případně kategorii. Aby v seznamu nebyla také samotná aplikace Educhild, je nutné odebrat aplikaci, jejíž *packageName* je shodný s tím, který je uveden v `context.packageName`, kde je dostupný vlastní *packageName* vyvíjené aplikace.

K zobrazení aplikací je využita komponenta `RecyclerView` pro monitorované a nemonitorované aplikace. `RecyclerView` slouží k zobrazení větších množin dat. Jeho úkolem je dynamicky vytvářet prvky na základě dodaných dat a definovaného vzhledu itemu. Typicky je možné scrollovat seznamem prvků. Pokud prvek není aktuálně zobrazen na obrazovce, nedochází k jeho úplnému zničení, ale jeho View je znovu použito na vytvoření dalšího prvku. Tato „*recyklace*“ razantně zlepšuje výkon a responzivitu aplikace a snižuje spotřebu energie [51].

Seznamy vytvořených itemů s monitorovanými a nemonitorovanými aplikacemi jsou pomocí *DataBinding* předány do příslušných `RecyclerView`, které se starají o jejich následné zobrazení. Itemy obsahují také ikony jednotlivých aplikací. Tyto ikony aplikací jsou získávány na základě *packageName* opět pomocí třídy `PackageManager`, konkrétně díky metodě `getApplicationIcon`, kterou disponuje.

3.2.5 Časový rozvrh

Časové rozvrhy udávají, kdy jsou vybrané aplikace blokovány úplně. Pokud je časový rozvrh aktivní, nemá dítě možnost využívat monitorované aplikace. Každý časový rozvrh disponuje názvem, časovým intervalem a množinou dní, ve kterých je aktivní.

Celkově tvorbu a správu časových rozvrhů tvoří dvě hlavní obrazovky. Na první obrazovce je uveden seznam všech vytvořených časových rozvrhů za pomoci komponenty `RecyclerView` podobně jako u zobrazování monitorovaných aplikací. U každého itemu z komponenty, tedy každého časového rozvrhu, je možné měnit jeho aktivnost pomocí klasické komponenty `switch`, jež spouští `UpdateTimetableActivityUseCase`, který na základě ID časového rozvrhu změní jeho aktivnost.

Pokud chce uživatel změnit parametry daného časového rozvrhu, dostane se pomocí tlačítka *Edit* do detailního přehledu, kde je možné parametry upravovat, případně časový rozvrh úplně smazat. Přechod ze seznamu na konkrétní časový rozvrh zajišťuje předání jeho ID pomocí navigačního argumentu, aby mohl být načten správný časový rozvrh.

Název časového rozvrhu zadává rodič do komponenty `EditText`. Pokud nechá toto pole prázdné, je jako název brán příslušný `String` ze souboru s texty, jež odpovídá aktuálně vybranému jazyku. Díky tomu je zajištěn správný překlad názvu pro všechny podporované jazyky. Pro výběr dnů je opět využita komponenta `switch` s výrazně změněným stylem, aby odpovídal designovému návrhu.

3.2.5.1 Časový interval

Zadávání časového intervalu je možné vykonat dvěma způsoby. První možností je využití textových polí `EditText`, kam uživatel zadává čas, od kdy a do kdy má být časový rozvrh aktivní. Aby bylo možné zadat jen korektní čas ve formátu `HH:MM`, případně `H:MM` je zde využita implementace objektu `TextWatcher`. Pokud je zmíněný objekt připojen ke komponentě, která rozšiřuje rozhraní `Editable`, čímž je i komponenta `EditText`, jsou volány odpovídající metody, když je změněn text [52]. Konkrétně se jedná o metody `beforeTextChanged`, `onTextChanged` a `afterTextChanged`. Jestliže uživatel zadá hodnotu, která neodpovídá formátu stanovenému pomocí regulárního výrazu, je přidaný znak odebrán.

Druhou možností zadání časového intervalu je využití komponenty kruhového posuvníku, který je realizovaný za pomoci knihovny *CircularRangeSlider*. Tato komponenta obsahuje dva posuvníky, se kterými je možné pohybovat v kruhu a přesně nastavit počáteční a koncový čas včetně správného intervalu mezi nimi [53]. Změny je možné zachytit pomocí `OnRangeChangeListener` objektu, ve kterém je přepsána metoda `onRangeChange`, aby změna dat přepsala i hodnoty atributů v odpovídající třídě `ViewModel`. Protože obrazovka detailu

```
circularRangeSlider.setOnTouchListener { v, event ->
    v!!.parent.requestDisallowInterceptTouchEvent(true)
    when (event!!.action and MotionEvent.ACTION_MASK) {
        MotionEvent.ACTION_UP ->
        binding.nestedScrollViewTimetable.
        requestDisallowInterceptTouchEvent(false)
    }
    false
}
```

Ukázka zdrojového kódu 4: Zakázání scrollování

časového rozvrhu je scrollovací, aby se na ni celé vytváření časového rozvrhu vměstnalo, bylo nutné implementovat zákaz scrollování, pokud uživatel nastavuje hodnotu na komponentě posuvníku, aby nedocházelo zároveň ke scrollování obrazovky a změnám počátečního nebo koncového času na posuvníku. K tomu byla využita metoda `setOnTouchListener` kruhového posuvníku, ve které je definován zákaz reagování na dotek komponentě `NestedScrollView`, jež je použita právě jako ViewGroup určený ke scrollování. Implementace řešení tohoto problému je v ukázce zdrojového kódu 4. Výsledkem je nemožnost scrollování obrazovky, pokud uživatel interaguje s komponentou kruhového posuvníku.

3.2.6 Statistiky

Aplikační statistiky je možné zobrazit ve třech časových intervalech, konkrétně statistiky denní, týdenní a desetidenní. Týdenní statistiky jsou vždy brány od pondělí do dne, kdy jsou zobrazovány.

Aby byl zajištěn jednoduchý a plynulý přechod mezi obrazovkami s různými časovými intervaly, využívám komponentu `ViewPager2` a `TabLayout`. Společné využití těchto komponent vytváří známé uživatelské prostředí pro navigaci mezi jednotlivými View [54].

V rámci aplikačních statistik jsou zobrazeny tři nejčastěji využívané aplikace během daného časového intervalu, případně je možné zobrazit detailní seznam se všemi používanými aplikacemi. Další možnost tvoří zobrazení statistik na základě seskupení aplikací do jednotlivých kategorií, jako jsou například hry a sociální sítě. Jedná se o obdobné kategorie, jejichž získávání bylo vysvětleno dříve v sekci *Primární nastavení*.

Zobrazení aplikačních statistik a statistik kategorií, na což je využita komponenta `RecyclerView`, je doplněno také o grafické znázornění, které je realizováno za pomoci knihovny *MPAndroidChart*. Tato knihovna přináší možnost přidávat View v podobě několika typů grafů [55]. V aplikačních statistikách je konkrétně využit *PieChart*, neboli koláčový graf, na kterém je názorněji zob-

razeno celkové rozdělení stráveného času. Z hodnot, které mají být součástí grafu, je vytvořen nejprve seznam `PieEntry`, kterým je předána hodnota, jež reprezentují. Z tohoto seznamu je vytvořen `PieDataSet`. Aby se v grafu objevovaly dané barvy musí zde být prostřednictvím atributu `colors` předán seznam vybraných barev. Posledním krokem je nahrání tohoto setu dat do samotného grafu a pomocí příkazu `invalidate()` jeho opětovné vykreslení s novými daty, na což je vytvořen vlastní *Binding adapter*.

3.3 Automatické sestavení a verzování

Android build systém, který stojí za sestavením aplikace, se stará o zkompilování zdrojů aplikace společně se zdrojovým kódem a jejich zabalením do výsledného APK balíčku, který je možné dále testovat, nasadit nebo distribuovat. Android Studio k tomuto úkolu využívá Gradle, což je pokročilý nástroj, jež slouží k automatizaci a správě procesu sestavení podle konfigurace, kterou je možné definovat [56].

Protože na projektu pracujeme v týmu bylo nutné pro správný průběh vývoje používat vhodný verzovací systém. Pro tento úkol byl zvolen distribuovaný verzovací systém Git, jelikož se jedná v současné době o nejvíce využívaný systém. Nad samotným systémem Git existují nástavby, které výrazně usnadňují a zpřehledňují jeho použití a přinášejí navíc užitečné funkce. Jednou z nejznámějších nástaveb je velice rozšířený GitHub. V rámci projektu Educhild ovšem využíváme GitLab, který je dostupný všem členům týmu díky fakultním účtům. GitLab je platforma, která je plně postavená na systému Git. Přináší mnoho užitečných funkcí nad rámec verzovacího systému Git, jež usnadňují vývoj [57].

Jednou z výhod, kterou GitLab disponuje, je využití CI fungujícím na principu spouštění *pipeline* pokaždé, když dojde k nahrání jakéhokoliv bloku kódu. *Pipeline* obsahuje skript, kterým je spouštěno sestavení a testování aplikace. Díky tomu je možné validovat kód předtím, než dojde k *merge* s hlavní větví repozitáře [58].

Samotnou konfiguraci automatického sestavení i nastavení repozitáře projektu v GitLab a CI realizoval Petr Šíma.

3.4 Dokumentace

Aby bylo možné implementované části průběžně spravovat a rozšiřovat, případně zvětšovat vývojářský tým, který se o aplikaci stará, je důležité, aby byla vytvořena odpovídající dokumentace, na kterou je možné se obrátit v případě nesrovnalostí. Dobrá dokumentace dokáže v budoucnu ušetřit mnohem více času, než zabere její napsání [59].

Protože je aplikace psána v programovacím jazyku Kotlin, jasnou volbou pro vytvoření dokumentace je *KDoc* a *Dokka*. *KDoc* je jazyk určený k do-

kumentování kódu napsaného v programovacím jazyku Kotlin [60]. Jedná se o ekvivalent *JavaDoc*, který se používá při psaní dokumentace ke kódu napsanému v Javě. Samotná syntaxe *KDoc* je obdobou syntaxe *JavaDoc*, pouze rozšířena o prvky, které jsou specifické pro Kotlin.

Ke generování dokumentace se využívá nástroj *Dokka*. Tento nástroj disponuje pluginem pro Gradle, takže je možné generování integrovat přímo do procesu sestavení [60]. Generování dokumentace se provádí příkazem `./gradlew dokkaHtmlMultiModule`.

V rámci rodičovského módu aplikace jsem vytvářel dokumentaci zejména pro jednotlivé třídy UseCase, protože se jedná o znovupoužitelné části kódu. Proto je nutné vytvořit dokumentaci tak, aby byl v daném kontextu použit vždy pouze správný UseCase.

3.5 Monitoring problémů

I když jednotlivé části aplikace prošly testovacím procesem, který je popsaný dále, nikdy není možné odhalit veškeré potencionální problémy. Proto klíčovou částí aplikace, která zajišťuje její další růst, je možnost monitorování a hlášení problémů uživatelů. Ve stále se rozrůstající základně aplikací znamená aplikace, která zdánlivě bezdůvodně padá, téměř jistý neúspěch. Jedním z nejlepších způsobů, jak zajistit, aby aplikace neobsahovala chyby, je okamžitě vědět o jakýchkoliv problémech. Identifikace a stanovení priority zásadních selhání umožňují včasné řešení a díky tomu zmírnění celkového dopadu na uživatele [61].

3.5.1 Sentry

Sentry je jedním z nástrojů, který monitoring problémů umožňuje. Jedná se o nástroj, který dokáže hlásit pády aplikace včetně detailů, kterými může být i název souboru a řádek kódu, kde problém nastal. Díky tomu není nutné další pátrání a dohledávání příčiny problému [62].

Užitečnou funkcí může být také filtrování a eliminování nepodstatných hlášení. Dalším důležitým prvkem Sentry je možnost odhalování, ve které verzi se chyba poprvé objevila, aby bylo možné přesněji odhalit dobu a místo jejího vzniku.

3.5.2 Firebase Crashlytics

Dalším z nástrojů, které umožňují získat jasné a přehledné informace o těchto problémech, je Firebase Crashlytics [63]. Tento nástroj umožňuje nahlašovat a sledovat problémy, které zhoršují výslednou kvalitu aplikace.

Aby nedošlo k úplnému zahlcení, Firebase Crashlytics disponuje funkcí sdružování obdobných problémů do jednoho. Tento nástroj dokáže poskytovat

informace o kontextu, za kterého daný problém vznikl, a definovat závažnost a rozsah zasažených uživatelů, aby bylo možné rozhodovat o prioritě řešení.

Další podstatnou funkcí jsou upozornění v reálném čase, tedy okamžitě, jakmile se objeví nový problém, stávající problém se začne rychle zvětšovat nebo jiný problém vyžaduje okamžitou pozornost [63].

Protože v rámci aplikace používáme i Firebase Authentication, jehož nastavování se provádí obdobně jako u Firebase Crashlytics skrze *Firebase console*, bylo v týmu rozhodnuto o využívání Firebase Crashlytics namísto ostatních monitorovacích systémů, aby se veškerý přehled nacházel na jednom místě.

Samotná implementace Firebase Crashlytics spočívá v integrování Crashlytics SDK. To je možné pomocí přidání Crashlytics do Gradle jako *dependency* v rámci souboru `build.gradle`. V případě, že je ve *Firebase console* vytvořen odpovídající projekt, začne Firebase Crashlytics okamžitě sbírat hlášení. Každé hlášení obsahuje jak informace o zařízení, tak i konkrétní výjimku, případně výjimky, které problém způsobily.

Kapitola 4

Testování

Předtím, než dojde na finální vydání aplikace, musí dojít k ověření správnosti, funkčnosti a použitelnosti aplikace. Aby bylo možné tyto prvky ověřit, je nutné aplikaci podrobit testování.

Testování tvoří další nedílnou součást procesu vývoje aplikace. Díky němu je možné získat rychlou zpětnou vazbu v případě chyb a zejména včasnou detekci selhání, aby bylo možné odstranit nedostatky z finální verze, kterou budou používat reální uživatelé [64].

Rodičovská část Android aplikace Educhild byla během vývoje podrobena několika typům testů. Základ testování tvořilo ověření správnosti vytvořeného UI a jeho funkčnosti pro každou přidanou funkcionalitu na různých typech zařízení s různými verzemi operačního systému Android a jednotkové testy, kterými jsou pokryty jednotlivé UseCase, aby bylo ověřeno, zda fungují jak mají.

Kromě těchto dvou typů testování, byla aplikace podrobena ještě uživatelskému testování, které měl na starost Marek Trzaskalik. Výsledky a bližší popis testování se nachází v jeho bakalářské práci. Na základě testování došlo k úpravě podoby několika obrazovek. Nejčastější opravou bylo přepsání málo výstižného textu popisujícího danou obrazovku. Další změnou, kterou testování přineslo, bylo zvětšení dotykové plochy pro některá tlačítka, aby byla jednodušeji aktivována.

Za účelem zjištění chyb, které nebyly zachyceny testováním a objeví se na produkci, je v aplikaci implementován monitoring problémů pomocí Firebase Crashlytics, jež byl popsán v sekci 3.5.

4.1 Testování na různých zařízeních

Protože je nutné, aby výslednou aplikaci bylo možné spustit a využívat na velkém počtu zařízení, musí být každá nově implementovaná funkcionalita otestována na různých typech zařízení s různými verzemi operačního systému

Název zařízení	Verze Android	Level API	Velikost obrazovky (v palcích)	Poměr stran
Xiaomi MI9	10	29	6,39	19,5:9
Huawei P9 lite	7.0	24	5,2	16:9
Emulátor	11	30	5,7	19:9
Emulátor	5.0	21	5,7	19:9

Tabulka 4.1: Testovací zařízení

Android. Toto testování je zaměřeno především na správné zobrazení a fungování prvků, které souvisí s UI aplikace, to znamená správné rozmístění prvků na menších i větších obrazovkách a korektní chování prvků na různých verzích operačního systému Android.

Zařízení používaná k tomuto testování se nachází v tabulce 4.1. Tato zařízení byla vybírána tak, aby testování ověřilo funkčnost a vzhled obrazovek na co největší skupině zařízení.

První dvě zařízení byla používána zejména pro ověření správného rozložení prvků na různě velkých obrazovkách. Jedná se o reálná zařízení, neboť právě ta poskytují vyšší věrohodnost testů. Pro testování bylo vybráno jedno zařízení s relativně velkým displejem a druhé s na dnešní dobu malou úhlopříčkou displeje. Obě zařízení mají odlišné poměry stran displeje. Díky této volbě je zaručena použitelnost aplikace na zařízeních se širokou škálou úhlopříček displeje.

Virtuální zařízení v podobě emulátoru byla použita pro testování zpětné kompatibility jednotlivých prvků s různými verzemi operačního systému Android. Jedná se o hardwarově totožná zařízení s odlišnými verzemi operačního systému Android. Jde o verzi aktuálně nejnovějšího operačního systému Android, to znamená Android 10, a verzi, která je v rámci nefunkčních požadavků definovaná jako minimální podporovaná. Protože novější verze Android musí podporovat veškeré funkcionality předchozích verzí, je díky tomuto výběru ověřena funkčnost na zařízeních s operačním systémem Android 5.0 s označením *Lollipop* a novějších.

Díky testování bylo zjištěno pár minoritních nedostatků, které se týkaly rozložení prvků, zejména pak na malých typech zařízení s menší úhlopříčkou displeje. Tyto nedostatky byly odstraněny pomocí úprav rozložení prvků v odpovídajících souborech jednotlivých obrazovek tak, aby docházelo ke správnému zobrazování na mobilních zařízeních se všemi běžně používanými velikostmi displeje. Ukázky nalezených nedostatků se nacházejí na přiloženém médiu.

4.2 Unit testy

Unit testy, neboli jednotkové testy, jsou základní typy testů zaměřující se na ověření funkčnosti malých komponent. Díky těmto testům je možné snadno zajistit správnost logiky dané komponenty. Spouštění jednotkových testů při každém sestavení aplikace umožňuje rychlé zachycení a opravení chyb, které vznikají změnami kódu. To zaručuje bezpečnější údržbu aplikace a přidávání nových funkcionalit do budoucna [65].

Unit testy se zaměřují na ověření funkcionalit nejmenších částí kódu, jakými jsou typicky metody nebo celé třídy. Při vytváření unit testů v rámci rodičovského módu se soustředím zejména na třídy UseCase, které obsahují nejpodstatnější část logiky aplikace.

4.2.1 JUnit

Za účelem zjednodušení testovacího procesu je velmi výhodné využít některý z vytvořených frameworků. Testování v aplikaci Educhild je na základě rozhodnutí týmu realizováno pomocí frameworku *JUnit4*. Jedná se o velice populární a rozšířený framework určený pro psaní jednotkových testů [66].

Testy se podle pravidel tohoto frameworku nachází v samostatné testovací třídě, která obsahuje testovací metody. Ty jsou označeny anotací `@Test` a obsahují kód, který má za účel ověřit funkčnost testované komponenty. V případě, že je potřeba vykonat nějakou činnost před, případně po provedení samotné testovací metody, jsou využity anotace `@Before`, respektive `@After`. První z uvedených anotací používám během testování k inicializaci a aplikaci potřebných závislostí.

4.2.2 MockK

Typicky je testovaná komponenta testována v izolaci, aby testy působily právě na ni a ověřovaly její funkčnost. Aby bylo možné komponentu ověřit v izolaci, jsou během testování využity knihovny, které připravují potřebné závislosti. Jednou z takových knihoven je například *Mockito*, nebo právě v aplikaci použitá knihovna *MockK*.

Jedná se o open source knihovnu, která se zaměřuje právě na obstarávání závislostí při testování. Knihovna má podporu pro Gradle, je vytvořená speciálně pro programovací jazyk Kotlin včetně podpory pro Coroutines [67]. To jsou hlavní důvody proč byla vybrána na úkor ostatních.

Knihovna obsahuje funkce `every`, případně `coEvery` (obdoba `every` pro Coroutines), ve kterých dochází k deklaraci chování prvků, které jsou předány v rámci závislostí testované komponentě. Díky funkcím `verify` a `coVerify` může být ověřeno správné volání požadovaných prvků. K zjednodušení vytváření objektů, které obstarávají závislosti, je využita anotace `@MockK`. Jednot-

livé závislosti jsou následně správně aplikovány primárně na základě jména proměnné, případně pomocí odpovídajícího typu třídy nebo nadtřídy [67].

4.2.3 Shrnutí

Testováním za pomoci jednotkových testů pokrývám zejména třídy UseCase, ve kterých je obsažena podstatná část logiky aplikace.

Během vytváření jednotkových testů jsem se zaměřoval na nejdůležitější části kódu, abych měl jistotu, že fungují správně. U jednotlivých tříd jsem kontroloval, zda jejich metody vracejí odpovídající hodnoty a zda jsou volány korektní funkce.

Výsledné pokrytí implementovaných tříd UseCase jednotkovými testy v závislosti na jednotlivých modulech, ve kterých se třídy nacházejí, je uvedeno v tabulce 4.2.

Modul	Pokrytí tříd	Pokrytí kódu
App	100 %	100 %
Data	100 %	97 %
Persistace	100 %	96 %

Tabulka 4.2: Pokrytí jednotkovými testy

Kapitola 5

Hodnocení vývoje a pokračování v projektu

Vytváření aplikace začalo na jaře roku 2020. Celkově dosavadní vývoj trval přibližně rok. Prozatímním výsledkem je plně funkční prototyp aplikace Educhild včetně rodičovského módu, ve kterém jsou implementovány a otestovány požadované funkcionality. Přestože rodičovský mód je na vydání připraven, aplikace v době psaní této práce ještě není dostupná pro veřejnost přes obchod Google Play.

Součástí výsledného rodičovského módu jsou následující funkcionality:

- přihlašování, registrace, přeskočení autentizace s vytvořením lokálního účtu,
- nastavení a zadávání PIN kódu pro přepínání z dětského do rodičovského módu,
- vytváření a spravování profilu dítěte,
- základní a pokročilé nastavení,
- přidávání nebo odebírání monitorovaných aplikací,
- zobrazování aplikačních statistik včetně detailního přehledu a seskupení podle kategorií,
- nastavení časového limitu,
- vytváření a upravovaní časových rozvrhů,
- shrnující hlavní obrazovka se zobrazením posledních aktivit dítěte, běžícího nebo nadcházejícího rozvrhu a aktuálně nastaveného časového rozvrhu,
- další nastavení (změna PIN kódu a hesla, úprava profilu dítěte, úprava odesílání anonymních dat, smazání účtu).

Navíc kromě těchto funkcionalit byla za účelem snazšího pochopení jednotlivých prvků u časových rozvrhů, monitorovaných aplikací a časového limitu vytvořena nápověda.

Výslednou podobu implementovaných a otestovaných částí rodičovského módu aplikace je možné vidět v příloze A. Veškeré zdrojové kódy se nacházejí v repozitáři: `https://gitlab.fit.cvut.cz/simapet4/educhild-android`.

Celkově dosavadní vývoj trval více než 500 hodin. Bližší informace jsou k nahlédnutí v nástroji Redmine, v němž byl vykazován strávený čas [68]. Práce na projektu Educhild mi umožnila přenést teoretické poznatky získané během studia do praxe. Osvojil jsem si dovednosti, které jsou nutné při vyvíjení Android aplikací. Na projektu bych rád pracoval i v budoucnu, abych tyto znalosti mohl nadále rozšiřovat.

5.1 Možná vylepšení

V rámci projektu Educhild a konkrétně rodičovské části Android aplikace se nabízí několik možných vylepšení vhodných do dalších verzí aplikace, která by ji více zdokonalila a rozšířila možnosti pro rodiče.

5.1.1 Odměny

Odměny již byly zpracovány v rámci analýzy a návrhu a s jejich začleněním do aplikace se počítá i v rámci stávající implementace, kde je do jisté míry na přidání této funkcionality vše připraveno.

Odměny nabídnou rodiči další možnost motivace dítěte k plnění kvízů. V první verzi aplikace sice může dítě samovolně spustit kvíz, ale není k tomu nijak významně motivováno. Aby rodič mohl podněcovat dítě k samostatnému spouštění kvízů, nabídne mu aplikace přidání odměny, kterou si dítě může vysloužit za jejich plnění. Samotná interpretace podoby odměny je na rodiči, může se jednat o nějaký výlet nebo malý dárek.

5.1.2 Přihlášení a registrace

V současné době je v aplikaci realizováno přihlášení a registrace pouze za pomoci e-mailové adresy a hesla. Firebase Authentication ovšem nabízí další způsoby autentizace, které by mohly být využity v aplikaci Educhild. Konkrétně se jedná zejména o přihlašování a registraci pomocí poskytovatelů identity, kterými jsou například společnosti Google, Facebook nebo Twitter.

5.1.3 Podrobnější statistiky kvízů

Dalším vylepšením pro rodiče je možnost nahlížet do odpovědí dítěte tak, aby si mohli udělat představu o tom, které otázky dělají dítěti problémy a jaké naopak jsou pro něj triviální.

V aplikaci jsou v současné době implementovány poslední aktivity dítěte, v nichž je možné vidět dosažené skóre, které dítě získalo během plnění jednotlivých kvízů, ale již zde není možnost, aby si rodič blíže prohlédl odpovědi na otázky a zjistil, kde přesně dítě udělalo chybu.

5.1.4 Podpora pro více dětí

Nejzásadnější změnou, kterou je nutné v dalších verzích aplikace vykonat je možnost přidávání více dětí v rámci jednoho rodičovského účtu, tak aby rodič mohl mít pohromadě přehled o všech svých dětech.

Aktuálně je aplikace zaměřena na používání zařízení pouze jedním dítětem, proto nelze přidat další dítě. Aby bylo možné v rámci rodičovského módu zahrnout podporu pro více dětí, je nutné, aby byly upraveny současné obrazovky. Rodič musí mít možnost vybrat, které dítě bude aktuálně mobilní zařízení využívat při přepínání do dětského módu, případně kterému dítěti je určen vytvořený časový rozvrh, doba na hraní nebo nastavený seznam monitorovaných aplikací. Stejně tak je nutné vybrat dítě, jehož statistiky si chce rodič aktuálně zobrazit.

5.1.5 Zákaz odinstalování

Posledním vylepšením, které v rámci této kapitoly uvádím, je zakázání odinstalování aplikace, pokud zařízení bude přepnuto do dětského režimu. Protože se dnešní děti na mobilních zařízeních orientují dostatečně dobře na to, aby byly schopny samy aplikaci, která jim automaticky spouští kvízy během hraní hry, případně hraní her úplně zakazuje, odinstalovat, je nutné tuto možnost omezit.

Závěr

Cílem bakalářské práce bylo vytvoření funkčního rodičovského módu v rámci prototypu mobilní aplikace Educhild. Rodičovská část měla uživateli umožňovat přihlašování a registraci, prvotní nastavení, zadávání pinu, zobrazování statistik a nastavení časového limitu, monitorovaných aplikací a rozvrhů dítěte. Vytyčené cíle byly naplněny, výsledný prototyp aplikace Educhild obsahuje plně funkční rozhraní pro rodiče s požadovanými funkcionalitami.

V úvodní kapitole jsem se věnoval analýze dané problematiky, požadavků na aplikaci a konkurenčních řešení. Následovala kapitola s návrhem požadovaných funkcionalit, jejíž součástí byl jak doménový model a popis dané domény, tak představení možných architektur a technologií využívaných při vývoji Android aplikací. V další kapitole jsem se zaměřil na popis implementace navržených funkcionalit a s tím spojených náležitostí. V závěrečných kapitolách jsem popsal proces testování aplikace, zhodnotil dosavadní vývoj a navrhl možná vylepšení do budoucna.

Výsledkem je prototyp Android aplikace Educhild s plně funkčním rodičovským módem, který obsahuje požadované funkcionality. Po spuštění aplikace je rodiči nabídnuta možnost přihlášení, registrace nebo přeskočení autentizace. Následuje nastavení PIN kódu, který je potřeba zadávat při přepínání mezi dětským a rodičovským módem tak, aby dítě nemohlo měnit nastavení aplikace. Na dalších obrazovkách rodič vyplňuje informace o dítěti a volí základní nebo pokročilé nastavení. V případě základního jsou nastaveny automaticky potřebné hodnoty, kterými jsou časový limit a seznam monitorovaných aplikací. Pokud se rodič rozhodne pro pokročilé nastavení, je proveden nastavením aplikace, kde může změnit hodnoty podle svých preferencí. Součástí rodičovského módu je vybírání monitorovaných aplikací ze seznamu aplikací nainstalovaných v zařízení, nastavování časového limitu pro dítě a vytváření a spravování časových rozvrhů. Rodič má možnost zobrazit, ve kterých aplikacích, případně jejich kategoriích, dítě tráví nejvíce času. Další možností je sledování posledních aktivit dítěte, které vznikly během vyplňování kvízů. Veškeré zdrojové kódy dosavadního vývoje jsou uloženy na fakultním GitLab.

Kromě výše popsaných funkcionalit nabízí projekt ještě řadu dalších vylepšení, kterým bych se rád věnoval i do budoucna. Jedním z nich je možnost přihlášení a registrace pomocí poskytovatelů identity, kterými jsou například společnosti Google, Facebook nebo Twitter. Dalšími způsoby, kterými je možné aplikaci zlepšit, jsou například podrobnější zobrazení statistik k jednotlivým kvízům nebo zákaz odinstalování aplikace. Nejvýznamnější vylepšení do budoucna tvoří integrace podpory pro více dětí, což rodiči umožní spravování všeho podstatného na jednom místě.

V neposlední řadě se také nabízí vývoj obdobné aplikace pro další používané operační systémy, zejména pak pro iOS. Díky tomu by bylo možné aplikaci Educhild využívat i na zařízeních s jiným operačním systémem než je Android.

Literatura

[1] IDC: *Smartphone Market Share - OS* [online]. [cit. 2021-3-2]. Dostupné z: https://www.idc.com/promo/smartphone-market-share/os

[2] Android Developers: *Distribution dashboard : Android Developers* [online]. [cit. 2021-3-2]. Dostupné z: https://developer.android.com/about/dashboards/index.html

[3] Android Developers: *Understand the Activity Lifecycle : Android Developers* [online]. [cit. 2021-3-19]. Dostupné z: https://developer.android.com/guide/components/activities/activity-lifecycle

[4] Android Developers: *Fragment lifecycle : Android Developers* [online]. [cit. 2021-3-19]. Dostupné z: https://developer.android.com/guide/fragments/lifecycle

[5] Android Developers: *Guide to app architecture : Android Developers* [online]. [cit. 2021-3-8]. Dostupné z: https://developer.android.com/jetpack/guide

[6] Dashti, F. A.; Yateem, A. K.: Use of Mobile Devices: A Case Study with Children from Kuwait and the United States. *International Journal of Early Childhood* [online], 2 2018, doi:10.1007/s13158-018-0208-x, [cit. 2021-02-27]. Dostupné z: https://link.springer.com/article/10.1007/s13158-018-0208-x

[7] Akbar, M. A.; Sang, J.; Khan, A. A.; aj.: Improving the Quality of Software Development Process by Introducing a New Methodology–AZ-Model. *IEEE Access* [online], 12 2017, [cit. 2021-3-9]. Dostupné z: https://ieeexplore.ieee.org/abstract/document/8241771

[8] Zhang, Z.: Effective requirements development-A comparison of requirements elicitation techniques. *British Computer Society* [online], 2 2007: s.

225–240, [cit. 2021-3-10]. Dostupné z: https://www.researchgate.net/publication/228717829_Effective_Requirements_Development-A_Comparison_of_Requirements_Elicitation_Techniques

[9] Android Developers: *Android 5.0 APIs : Android Developers* [online]. [cit. 2021-3-2]. Dostupné z: https://developer.android.com/about/versions/android-5.0.html

[10] xda: *Android Version Distribution statistics will now only be available in Android Studio* [online]. [cit. 2021-4-25]. Dostupné z: https://www.xda-developers.com/android-version-distribution-statistics-android-studio/

[11] UML graphical notation overview, examples, and reference.: *UML Use Case Diagrams* [online]. [cit. 2021-3-11]. Dostupné z: https://www.uml-diagrams.org/use-case-diagrams.html

[12] Qustodio: *Best Parental Control Software* [online]. [cit. 2021-2-28]. Dostupné z: http://www.qustodio.com/en/

[13] AppBlock: *Stay focused at work* [online]. [cit. 2021-2-28]. Dostupné z: https://www.appblock.app/

[14] Seekrtech: *Forest - Stay focused, be present* [online]. [cit. 2021-2-28]. Dostupné z: http://www.forestapp.cc/

[15] Google: *Forest: Stay focused - Apps on Google Play* [online]. [cit. 2021-2-28]. Dostupné z: https://play.google.com/store/apps/details?id=cc.forestapp&hl=en_US&gl=US

[16] Duolingo: *Duolingo: Learn Languages Free* [online]. [cit. 2021-2-28]. Dostupné z: https://www.duolingo.com/

[17] Google: *Duolingo: Learn Languages Free - Apps on Google Play* [online]. [cit. 2021-2-28]. Dostupné z: https://play.google.com/store/apps/details?id=com.duolingo&hl=en&gl=US

[18] Quizlet Inc.: *Quizlet - learning tools & flashcards* [online]. [cit. 2021-2-28]. Dostupné z: https://quizlet.com/

[19] Culttt: *What is the Domain Model in Domain Driven Design?* [online]. [cit. 2021-3-15]. Dostupné z: https://culttt.com/2014/11/12/domain-model-domain-driven-design/

[20] Android Developers: *Meet Android Studio : Android Developers* [online]. [cit. 2021-3-9]. Dostupné z: https://developer.android.com/studio/intro

[21] IntelliJ IDEA Help: *Tutorial: Create your first Android application - Help: IntelliJ IDEA* [online]. [cit. 2021-3-9]. Dostupné z: https://www.jetbrains.com/help/idea/create-your-first-android-application.html

[22] Visual Studio: *Android Development: Visual Studio* [online]. [cit. 2021-3-9]. Dostupné z: https://visualstudio.microsoft.com/vs/android/

[23] Android Developers: *Application Fundamentals : Android Developers* [online]. [cit. 2021-3-9]. Dostupné z: https://developer.android.com/guide/components/fundamentals?hl=en

[24] GeeksforGeeks: *Top Programming Languages for Android App Development* [online]. [cit. 2021-3-9]. Dostupné z: https://www.geeksforgeeks.org/top-programming-languages-for-android-app-development/

[25] Android Developers: *Kotlin and Android : Android Developers* [online]. [cit. 2021-3-9]. Dostupné z: https://developer.android.com/kotlin

[26] Ardito, L.; Coppola, R.; Malnati, G.; aj.: Effectiveness of Kotlin vs. Java in android app development tasks. *Information and Software Technology* [online], 11 2020, doi:10.1016/j.infsof.2020.106374, [cit. 2021-3-9]. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0950584920301439

[27] Android Developers: *Kotlin coroutines on Android : Android Developers* [online]. [cit. 2021-3-25]. Dostupné z: https://developer.android.com/kotlin/coroutines

[28] Android Developers: *Introduction to Activities : Android Developers* [online]. [cit. 2021-3-19]. Dostupné z: https://developer.android.com/guide/components/activities/intro-activities

[29] Android Developers: *Fragments : Android Developers* [online]. [cit. 2021-3-19]. Dostupné z: https://developer.android.com/guide/fragments

[30] Android Developers: *Services overview : Android Developers* [online]. [cit. 2021-3-19]. Dostupné z: https://developer.android.com/guide/components/services

[31] Medium: *Architecture patterns in Android - Android architecture design* [online]. [cit. 2021-3-8]. Dostupné z: https://medium.com/android-news/architecture-patterns-in-android-abf99f2b6f70

[32] Medium: *Android Architecture Patterns Part 1: Model-View-Controller* [online]. [cit. 2021-3-8]. Dostupné z: https://medium.com/upday-devs/

android-architecture-patterns-part-1-model-view-controller-3baecef5f2b6

[33] Medium: *Model View Presenter(MVP) in Android with a simple demo project* [online]. [cit. 2021-3-8]. Dostupné z: https://medium.com/cr8resume/make-you-hand-dirty-with-mvp-model-view-presenter-eab5b5c16e42

[34] Tateeda: *Top 3 Frameworks for Android App Development in 2020* [online]. [cit. 2021-3-21]. Dostupné z: https://tateeda.com/blog/top-3-frameworks-for-android-app-development-in-2020

[35] GitHub: *Android-Kotlin "framework"designed to reduce boilerplate* [online]. [cit. 2021-3-21]. Dostupné z: https://github.com/skoumalcz/teanity

[36] Maryland: Department of Information Technology: *Phase 6: Development - COTS Single Release Project* [online]. [cit. 2021-3-28]. Dostupné z: https://doit.maryland.gov/SDLC/COTS/Pages/Phase06Single.aspx

[37] Android Developers: *Dependency injection in Android : Android Developers* [online]. [cit. 2021-4-2]. Dostupné z: https://developer.android.com/training/dependency-injection

[38] Android Developers: *Layouts : Android Developers* [online]. [cit. 2021-4-2]. Dostupné z: https://developer.android.com/guide/topics/ui/declaring-layout

[39] Android Developers: *Build a Responsive UI with ConstraintLayout : Android Developers* [online]. [cit. 2021-4-2]. Dostupné z: https://developer.android.com/training/constraint-layout

[40] Android Developers: *ViewModel Overview : Android Developers* [online]. [cit. 2021-3-30]. Dostupné z: https://developer.android.com/topic/libraries/architecture/viewmodel

[41] Android Developers: *Data Binding Library : Android Developers* [online]. [cit. 2021-4-3]. Dostupné z: https://developer.android.com/topic/libraries/data-binding

[42] Android Developers: *Accessing data using Room DAOs : Android Developers* [online]. [cit. 2021-4-3]. Dostupné z: https://developer.android.com/training/data-storage/room/accessing-data

[43] Android Developers: *Save key-value data : Android Developers* [online]. [cit. 2021-4-3]. Dostupné z: https://developer.android.com/training/data-storage/shared-preferences

[44] Medium: *Kotpref, The Easier Way To Do Shared Preferences* [online]. [cit. 2021-4-3]. Dostupné z: https://medium.com/dot-intern/kotpref-the-easier-way-to-do-shared-preferences-77e89b9c58a

[45] DEV Community: *Kotpref: Easy SharedPreferences library for Kotlin android* [online]. [cit. 2021-4-3]. Dostupné z: https://dev.to/chibatching/kotpref-easy-sharedpreferences-library-for-kotlin-android-c68

[46] Google: *Firebase Authentication* [online]. [cit. 2021-3-29]. Dostupné z: https://firebase.google.com/docs/auth

[47] GitHub: *hdodenhof/CircleImageView* [online]. [cit. 2021-3-30]. Dostupné z: https://github.com/hdodenhof/CircleImageView

[48] Android Developers: *ApplicationInfo : Android Developers* [online]. [cit. 2021-3-30]. Dostupné z: https://developer.android.com/reference/android/content/pm/ApplicationInfo

[49] Android Developers: *Pass data between destinations : Android Developers* [online]. [cit. 2021-3-30]. Dostupné z: https://developer.android.com/guide/navigation/navigation-pass-data

[50] Android Developers: *Intent : Android Developers* [online]. [cit. 2021-4-1]. Dostupné z: https://developer.android.com/reference/android/content/Intent

[51] Android Developers: *Create dynamic lists with RecyclerView : Android Developers* [online]. [cit. 2021-4-1]. Dostupné z: https://developer.android.com/guide/topics/ui/layout/recyclerview

[52] Android Developers: *TextWatcher : Android Developers* [online]. [cit. 2021-3-31]. Dostupné z: https://developer.android.com/reference/android/text/TextWatcher

[53] GitHub: *bikcrum/CircularRangeSlider-Android* [online]. [cit. 2021-3-31]. Dostupné z: https://github.com/bikcrum/CircularRangeSlider-Android

[54] Android Developers: *Create swipe views with tabs using ViewPager2 : Android Developers* [online]. [cit. 2021-3-31]. Dostupné z: https://developer.android.com/guide/navigation/navigation-swipe-view-2

[55] GitHub: *PhilJay/MPAndroidChart* [online]. [cit. 2021-3-31]. Dostupné z: https://github.com/PhilJay/MPAndroidChart

[56] Android Developers: *Configure your build : Android Developers* [online]. [cit. 2021-4-15]. Dostupné z: https://developer.android.com/studio/build

[57] Gitlab: *Git* [online]. [cit. 2021-4-15]. Dostupné z: https://docs.gitlab.com/ee/topics/git/

[58] Gitlab: *GitLab CI/CD* [online]. [cit. 2021-4-15]. Dostupné z: https://docs.gitlab.com/ee/ci/

[59] Software Quality Blog - SubMain Software: *Code Documentation: The Complete Beginner's Guide - SubMain* [online]. [cit. 2021-4-15]. Dostupné z: https://blog.submain.com/code-documentation-the-complete-beginners-guide/

[60] Kotlin Help: *Document Kotlin code: KDoc and Dokka: Kotlin* [online]. [cit. 2021-4-15]. Dostupné z: https://kotlinlang.org/docs/kotlin-doc.html

[61] Android Developers: *Debug crashes quickly using Android vitals and Firebase Crashlytics* [online]. [cit. 2021-4-10]. Dostupné z: https://developer.android.com/distribute/best-practices/launch/debug-crashes

[62] Sentry: *Android Error and Performance Monitoring* [online]. [cit. 2021-4-14]. Dostupné z: https://sentry.io/for/android/

[63] Google: *Firebase Crashlytics* [online]. [cit. 2021-4-10]. Dostupné z: https://firebase.google.com/docs/crashlytics

[64] Android Developers: *Test apps on Android : Android Developers* [online]. [cit. 2021-4-11]. Dostupné z: https://developer.android.com/training/testing

[65] Android Developers: *Build effective unit tests : Android Developers* [online]. [cit. 2021-4-19]. Dostupné z: https://developer.android.com/training/testing/unit-testing

[66] Android Developers: *Build local unit tests : Android Developers* [online]. [cit. 2021-4-19]. Dostupné z: https://developer.android.com/training/testing/unit-testing/local-unit-tests

[67] MockK: *MockK : mocking library for Kotlin* [online]. [cit. 2021-4-19]. Dostupné z: https://mockk.io/

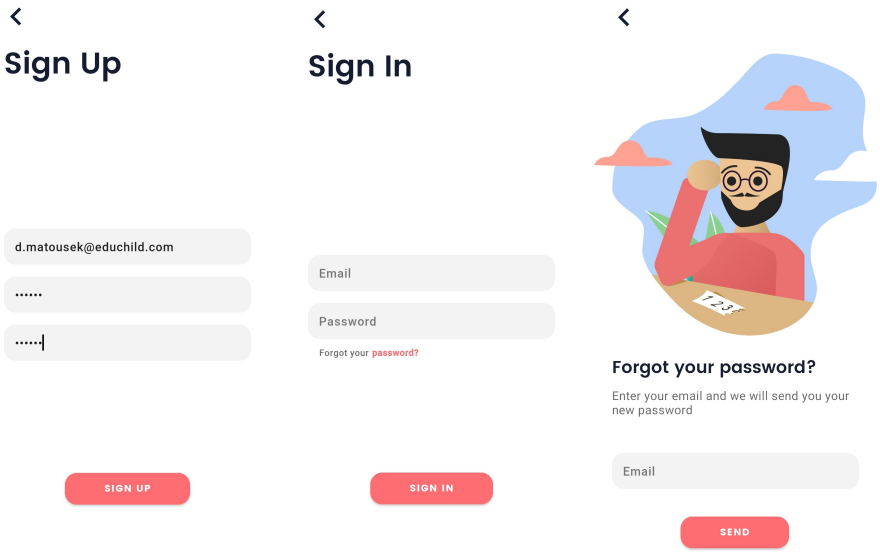
[68] Redmine: *Redmine* [online]. [cit. 2021-4-13]. Dostupné z: https://www.redmine.org/

Příloha A

Výsledná podoba aplikace

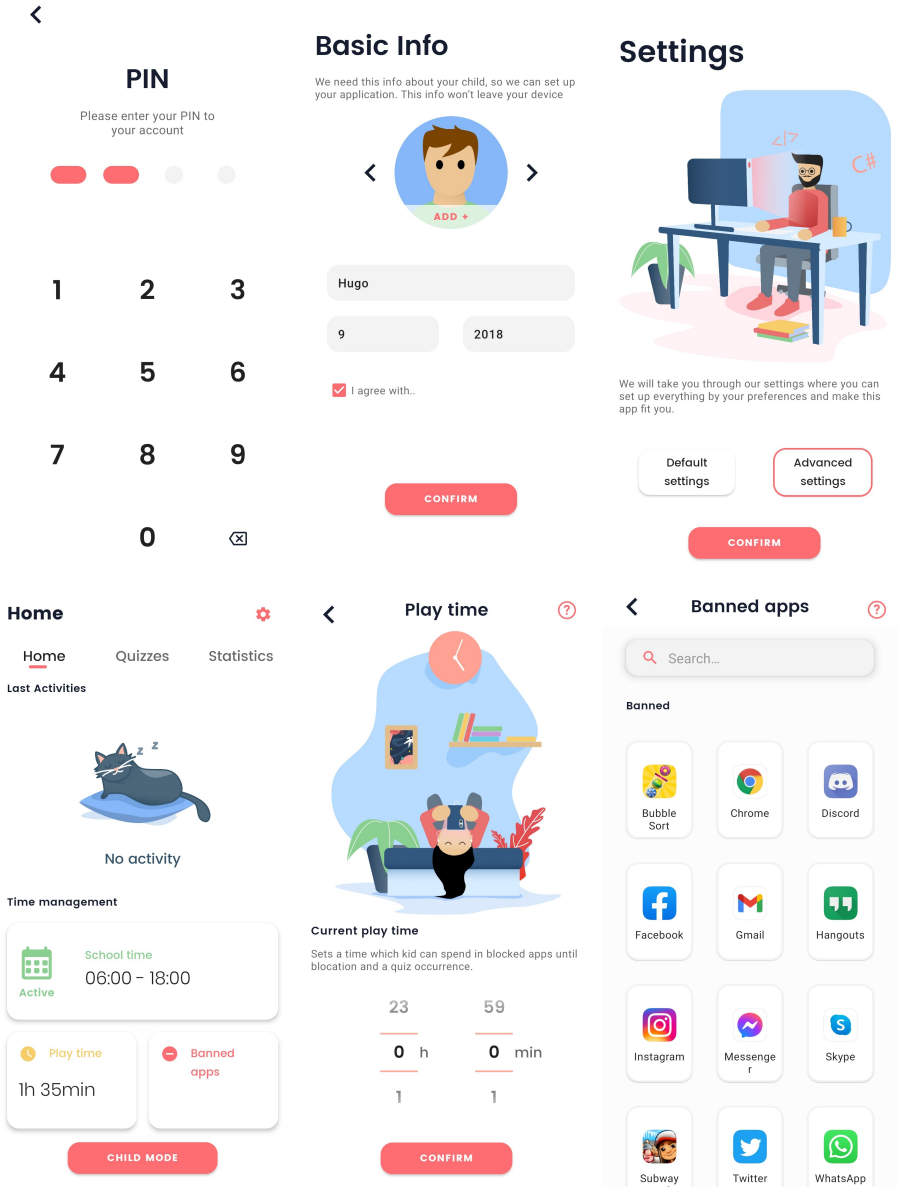
Přílohu tvoří snímky obrazovky z rodičovského módu Android aplikace Educhild. Všechny snímky byly pořízeny na mobilním zařízení Xiaomi MI9 s operačním systémem Android 10 (level API 29), displejem s velikostí 6,39 palce a poměrem stran 19,5:9. Jako primární jazyk mobilního zařízení je nastavena angličtina.

V ukázce A.1 se nacházejí autentizační obrazovky. Konkrétně se jedná o registraci, přihlášení a obrazovku pro resetování hesla, jestliže ho majitel uživatelského účtu zapomněl.



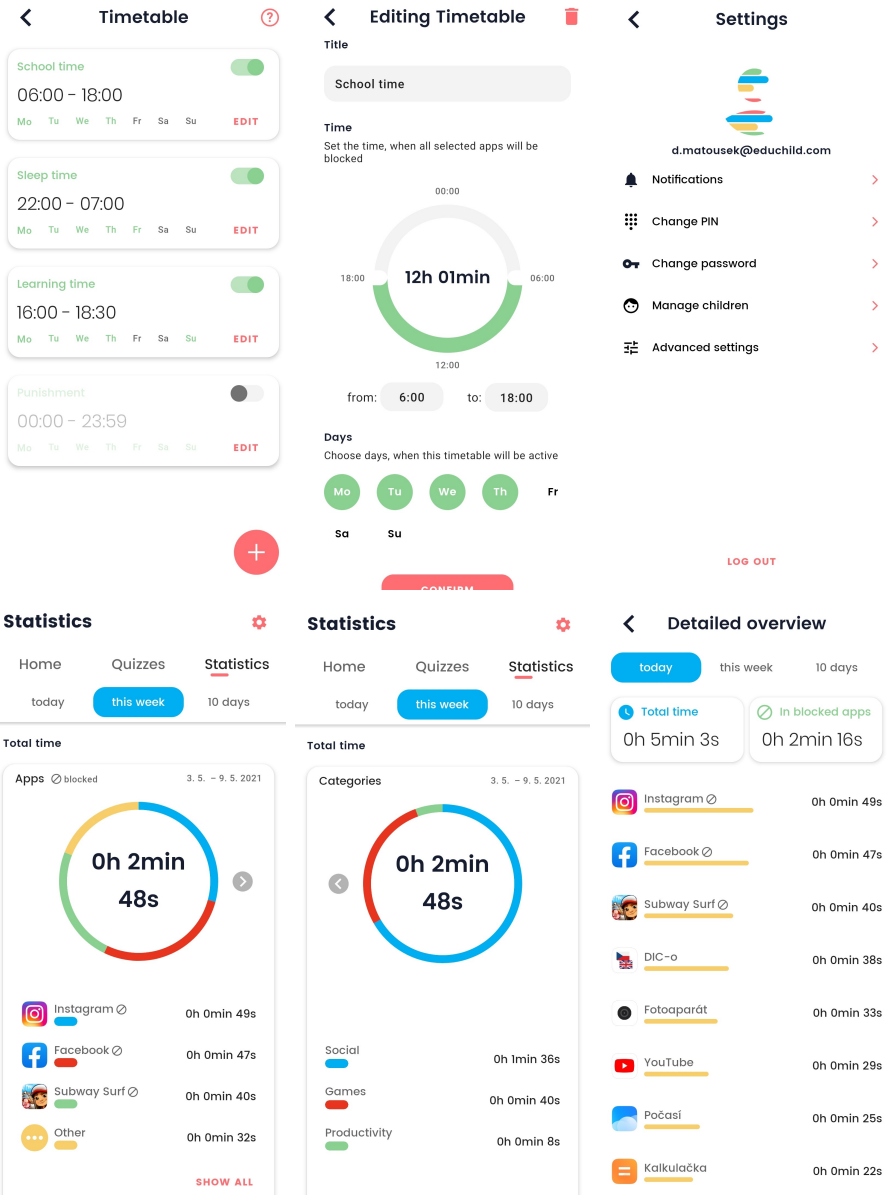
Obrázek A.1: Ukázka výsledné aplikace č. 1

Na obrazovkách v ukázce A.2 se nachází zadávání PIN kódu, vytváření profilu pro dítě a vybírání základního nebo pokročilého nastavení. Dále je zde ukázka domovské obrazovky rodičovského módu, nastavení časového limitu a výběru monitorovaných aplikací.



Obrázek A.2: Ukázka výsledné aplikace č. 2

V závěrečné ukázce A.3 se nachází obrazovky pro zobrazení seznamu časových rozvrhů, úpravu dat jednoho z nich a další nastavení aplikace. Na druhé řádce se nachází zobrazení statistik. První jsou statistiky aplikační, následuje shrnutí podle kategorií a poslední snímek zachycuje detailní přehled všech aplikací. V horní liště je možné přepínat mezi časovými obdobími.



Obrázek A.3: Ukázka výsledné aplikace č. 3

Příloha B

Seznam použitých zkratek

API Application Programming Interface

IDE Integrated Development Environment

JVM Java Virtual Machine

UI User Interface

DI Dependency Injection

SDK Software Development Kit

DAO Data Access Object

APK Android Package

CI Continuous Integration

Příloha C

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `apk` ..... adresář se spustitelnou formou implementace
- `src`
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
- `screenshots` ..... adresář s ukázkami výsledné podoby obrazovek
- `testing`
  - `multiple_devices` ..... adresář se snímky obrazovek se špatným rozložením nalezených při testování na různých zařízeních
  - `unit` ..... adresář s výsledky a údaji o pokrytí unit testů