FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Refactoring aplikace Myšák 2
Student:	Václav Čermák
Vedoucí:	Ing. Jiří Chludil
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce zimního semestru 2021/22

Pokyny pro vypracování

Myšák 2 je výuková aplikace pro děti předškolního věku, která napomáhá rozvoji grafomotoriky. Základ pro aplikace tvoří pracovní listy, které jsou v tištěné formě rozdány dětem. Aplikace pak za použití computer vision a rozšířené reality načte a vyhodnotí správnost řešení.

1) Analyzujte aplikaci Myšák 2 vytvořenou v rámci předmětů SP1 a SP2.
2) Knihovny pro rozpoznávání obrazu OpenCV a AR core podrobte analýze a zvolte nejvhodnější technologii pro načítání řešení úloh.
3) Pomocí metod softwarového inženýrství navrhněte novou verzi aplikace.
4) Implementujte novou verzi aplikace, zaměřte se na úpravu systému pro načítání řešení úloh.
5) Novou verzi aplikace podrobte vhodným testům.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 17. února 2020

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Refaktoring aplikace Myšák 2

Václav Čermák

Katedra Softwarového Inženýrství
Vedoucí práce: Ing. Jiří Chludil

4. června 2020

Poděkování

Převším bych chtěl poděkovat vedoucímu mé bakalářské práce Ing. Jiřímu Chludilovi za odborné vedení, pravidelné konzultace a pomoc při zpracování této práce. Dále bych chtěl poděkovat Ing. Ondřeji Slabému za pomoc při nastavení systému průběžné integrace a za věcné připomínky k práci. Chtěl bych poděkovat i Aleně Žižkové za pomoc při řešení potíží s kompatibilitou. Na závěr bych rád poděkoval za trpělovist všem, kteří se podíleli na akceptačním testování aplikace.

Prohlášení

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

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

V Praze dne 4. června 2020 ……

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Čermák, Václav. *Refaktoring aplikace Myšák 2*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Tato bakalářská práce se zabývá refaktoringem aplikace Myšák 2, vyvinuté v rámci stejnojmenného softwarového projektu. Aplikace je určena pro předškoláky a jejím cílem je pomáhat v rozvoji grafomotoriky. Aplikace má však mnoho nedostatků, kvůli kterým je v praxi nepoužitelná. Práce analyzuje tyto nedostatky, navrhuje řešení na jejich odstranění a vyvíjí novou verzi aplikace, zaměřenou na načítání pracovních listů, které se ukázalo být největší slabinou aplikace. Nedostatky aplikace Myšák 2 byly způsobeny nedostatečnou analýzou, špatným návrhem a s ním související špatnou volbou frameworků a knihoven. Při vývoji nové verze aplikace proto práce na analýzu a návrh klade velký důraz. Po vyhodnocení analýzy se s ohledem na nové technologie ukázalo, že je vhodnější zahodit existující řešení aplikace a vytvořit nové jádro zcela od začátku. Nová aplikace tedy nemá kromě estetické stránky a části algoritmů pro zpracování obrazu s původní aplikací téměř nic společného. Aplikace v současné době disponuje novým jádrem a zvládá načítat pracovní listy. Nové technologie a značně odlišná architektura navíc nabízí možnost ji v budoucnu rozšířit na platformu iOS. Práce se nezabývá rozšířením aplikace o nové úlohy a pracovní listy. Pouze analyzuje ty současné a popisuje algoritmy použité u jednotlivých kategorií úloh. Práce se nezabývá ani uživatelským rozhraním, postavami, dabingem, či vytvářením herního obsahu. Všechny tyto vyjmenované oblasti jsou též pouze analyzovány a návrh jejich řešení bude ponechán pro případné navazující práce.

Klíčová slova refaktoring aplikace, Android, grafomotorika, pracovní list, akceptační testování

Abstract

This bachelor thesis deals with the refactoring of the Myšák 2 application, developed within the software project of the same name. The application is intended for preschoolers and its goal is to help in the improvement of graphomotor skills. However, the application has many drawbacks that make it inapplicable in practice. The work analyzes these drawbacks, proposes solutions to eliminate them, and is developing a new version of the application, focused on scanning worksheets, which has proven to be the application's biggest weakness. The shortcomings of the Myšák 2 application were caused by insufficient analysis, poor design, and related bad choice of frameworks and libraries. Therefore, when developing a new version of the application, work on analysis and design places great emphasis. After evaluating the analysis, regarding new technologies, it turned out that it is more appropriate to discard the existing application solution and create a new kernel from scratch. Thus, the new application has almost nothing in common with the aesthetic aspect and part of the image processing algorithms with the original application. The application currently has a new kernel and can load worksheets. In addition, new technologies and a significantly different architecture offer the possibility to extend it to the iOS platform in the future.The work does not deal with extending the application with new tasks and worksheets. It only analyzes the current ones and describes the algorithms used in each task category. Thesis also does not deal with the user interface, characters, dubbing, or creating game content. All these listed areas are also only analyzed and the proposal of their solution will be left for possible follow-up work.

Keywords aplication refactoring, Android, graphomotorics, worksheet, acceptance testing

Obsah

Úvod **1**

1 Cíl práce **3**

2 Analýza původního stavu aplikace **5**
2.1 Charakteristické vlastnosti aplikace 5
2.1.1 Vyřazené účty 10
2.2 Frontend 11
2.2.1 Wireframy 11
2.2.2 Animace a postava myšáka 12
2.3 Databáze a backend 13
2.3.1 Databázový model 14
2.3.2 ORM 15
2.4 Architektura a business logika 16
2.4.1 Prezenční vrstva 18
2.4.2 Datová vrstva 22
2.4.3 Vrstva aplikační logiky 23
2.5 Dostupné úlohy 25
2.6 Testování původní aplikace 25
2.7 Zpracování obrazu 28
2.7.1 Algoritmy pro zpracování obrazu 29
2.7.2 Barevný model HSV 30
2.7.3 Modul pro načítání pracovních listů 31
2.8 Funkční a nefunkční požadavky 34

3 Analýza technologií pro novou verzi aplikace **37**
3.1 Programovací jazyk Kotlin 39
3.2 Analýza technologií pro detekci pozičních značek 40
3.2.1 ArUco knihovna 42

3.2.2 ArUco modul pro OpenCV . . . 43
3.2.3 ARToolKit . . . 44
3.2.4 Chilitags . . . 45
3.2.5 ReacTIVision . . . 46
3.2.6 AR Core . . . 47
3.3 Analýza frameworků a enginů pro GUI . . . 49
3.3.1 Android SDK s OpenGL ES . . . 50
3.3.2 Cocos 2d-x . . . 51
3.3.3 LWJGL . . . 53
3.3.4 jMonkey . . . 54
3.3.5 Unity . . . 55
3.3.6 LibGDX . . . 56
3.4 Analýza ORM frameworků . . . 59
3.4.1 Sugar ORM . . . 59
3.4.2 GreenDao . . . 60
3.4.3 Room . . . 61
3.4.4 OrmLite . . . 62
3.4.5 SQLDelight . . . 62
3.4.6 Realm . . . 63
3.5 Výsledná volba technologií . . . 64
3.5.1 GUI . . . 64
3.5.2 Poziční značky . . . 65
3.5.3 ORM . . . 66

4 Návrh nové architektury 67
4.1 Nové požadavky . . . 69
4.2 Jádro aplikace . . . 70
4.2.1 Logic . . . 70
4.2.1.1 Vyhodnocování úloh . . . 70
4.2.2 View . . . 74
4.2.3 Data . . . 77
4.3 Čtení dat z kamery . . . 78
4.4 Správa paměti . . . 80

5 Návrh modulu pro načítání pracovních listů 83
5.1 Geometrické transformace obrazu . . . 83
5.1.1 Převzorkování . . . 84
5.1.2 Warping . . . 85
5.2 Algoritmus načítání pracovního listu . . . 88
5.2.1 API OpenCV knihovny . . . 89

6 Implementace 91
6.1 Změny oproti návrhu . . . 91
6.2 Shrnutí a zhodnocení nových požadavků . . . 94

6.3 Správa aplikace 96
6.4 Návody 97
6.4.1 Kompilace OpenCV knihovny 97
6.4.1.1 Konfigurace CMake 98
6.4.1.2 Kompilace 101
6.4.1.3 Chyby při kompilaci 102
6.4.2 Nalinkování OpenCV SDK do Android Studia 104

7 Testování **107**
7.1 Porovnání rychlosti detekce 107
7.1.1 Rozdíly v implementaci 107
7.1.2 Měření 108
7.1.3 Další poznatky z testování 109
7.2 Uživatelské testování 110
7.2.1 Uživatelé a podmínky 111
7.2.2 Načítání pracovních listů 111
7.2.3 Výsledky a závěr z testování 112

Závěr **115**

Literatura **117**

A Seznam použitých zkratek **123**

B Obsah přiloženého CD **125**

Seznam obrázků

2.1 Ukázka pracovního listu 7
2.2 UseCase diagram použití aplikace 8
2.3 Activity diagram průchodu aplikací 9
2.4 Původní databázový model 14
2.5 Diagram balíčků současné architektury 17
2.6 Diagram tříd hierarchie. Červeně zvýrazněné vazby představují chyby v dědičnosti 19
2.7 Diagram tříd aktivit odvozených z hierarchie. Červené vazby představují chyby v dědičnosti 21
2.8 Diagram tříd fragmentů 22
2.9 Diagram tříd balíčku Data 23
2.10 Diagram tříd aplikační logiky (komunikace mezi C++ a Java částí aplikace) 24
2.11 Barevný model HSV. Obrázek byl převzat z práce [1] 31
2.12 Diagram tříd rozpracovaného modulu 32
2.13 Ukázka špatně nalinkované OpenCV knihovny 33

3.1 Ukázka různých druhů pozičních značek, převzatá z práce [2] . . . 40
3.2 Ukázka pozičních značek ReacTIVision 47

4.1 Diagram modulů a jejich závislostí 68
4.2 Diagram architektury, reprezentující spouštění aplikace na různých platformách 69
4.3 Hlavní třída a inicializace kontextu 71
4.4 Načtený pracovní list 72
4.5 Nový modul pro hodnocení úloh 73
4.6 Rozdělení View do balíčků podle zodpovědností 74
4.7 Stavy hlavní obrazovky 75
4.8 Návrh tříd balíčku screens 76
4.9 Nový databázový model 77

4.10 Přístup ke kameře a předávání dat do jádra 79
4.11 Sekvenční diagram čtení QR kódu z kamery 80
4.12 Mapování objektů z Java rozhraní OpenCV na jejich C++ implementaci 81
4.13 Zaplnění nativní haldy. Garbage Collector neví o nativní části haldy 82

5.1 Snímek s načtenými pozičními značkami obdržený z kamery. . . . 84
5.2 Snímek získaný transformací obrazu z 5.1. 85
5.3 Perspektivní transformace pomocí mapování souřadnic pozičních značek 87
5.4 Návrh tříd pro detekci pozičních značek 89
5.5 Načítání pracovních listů v OS Android 90

6.1 Diagram závislostí jednotlivých modulů na OpenCV knihovnu . . . 94
6.2 Ukázková konfigurace kompatibility pro Android 100
6.3 Nastavení cesty k extra modulům OpenCV knihovny 100
6.4 Knihovny pro cross-kompilaci na architekturu arm64-v8a (OS Linux) 103

7.1 Vyčerpání paměti při detekci značek. 109

Seznam tabulek

2.1 HSV rozsahy barev definované v třídě config.h 29
2.2 Závěrečné zhodnocení funkčních a nefunkčních požadavků za celý průběh softwarového projektu 34

3.1 Bodový zisk knihovny ArUco 42
3.2 Bodový zisk ArUco modulu pro OpenCV 44
3.3 Bodový zisk knihovny ARToolKit 45
3.4 Bodový zisk knihovny Chilitags 46
3.5 Bodový zisk OpenGL ES 51
3.6 Bodový zisk Cocos 2d-x 53
3.7 Bodový zisk jMonkey 55
3.8 Bodový zisk Unity 56
3.9 Bodový zisk LibGDX 58
3.10 Bodový zisk Sugar ORM 60
3.11 Bodový zisk GreenDao 60
3.12 Bodový zisk Room 61
3.13 Bodový zisk 62
3.14 Bodový zisk SQLDelight 63
3.15 Bodový zisk Realm 64
3.16 Výsledný bodový zisk frameworků a enginů pro GUI 64
3.17 Výsledný bodový zisk frameworků a knihoven detekce pozičních značek 65
3.18 Výsledný bodový zisk ORM frameworků 66

6.1 Závěrečné zhodnocení funkčních a nefunkčních požadavků nové verze aplikace 95

7.1 Vyhodnocení rychlosti čtení pozičních značek 108

Úvod

Nedostatečný rozvoj grafomotoriky u dětí předškolního věku má mnoho negativních dopadů, které se projevují při jejich následném nástupu do školy. Tyto děti mohou mít potíže se psaním, kreslením a rýsováním. Z toho plyne řada dalších problémů. Dítě si například nestihne včas zapsat poznámky. Nebo si je zapsat stihne, ale nepřečte pak svůj vlastní rukopis.

Myšák 2 je výuková aplikace pro děti předškolního věku, jejíž cílem je pomáhat v rozvoji jemné motoriky. Aplikace však obsahuje velkou řadu nedostatků, kvůli kterým je v praxi téměř nepoužitelná.

Tato bakalářská práce se zabývá analýzou těchto nedostatků, jejich odstraněním při zachování hlavních požadavků na systém a přidáním nových funkcionalit. Práce se dále zaměřuje na návrh nové architektury aplikace a vytvoření modulu pro usnadnění načítání pracovních listů. Tato práce se nezabývá vzhledovou stránkou aplikace. Grafická podoba aplikace a prvky uživateslkého rozhraní budou s výjimkou animací postavy myšáka přezaty ze softwarového projektu.

Kapitola 1

Cíl práce

Cílem této práce je identifikovat nedostatky aplikace Myšák 2 (ve stavu v jakém byla odevzdána v rámci předmětu BI-SP2), navrhnout řešení na jejich odstranění a na základě tohoto návrhu vyvinout novou verzi aplikace. Tuto nově vytvořenou verzi podrobit uživatelskému testování s dětmi a porovnat výsledky s výstupem testování, původní verze aplikace. Výstup této práce poslouží pro navazující práci, která se bude zabývat uživatelským rozhraním aplikace a rozšířením o nové úlohy.

Kapitola 2

Analýza původního stavu aplikace

Tato kapitola je věnována analýze původního stavu aplikace. Nejprve je vysvětlen princip práce aplikace, její použití a průchod od spuštění po ukončení. Poté je analyzována vizuální stránka aplikace, aplikační logika a nakonec datová část.

Následuje část uživatelského testování aplikace, které mělo proběhnout v rámci BI-SP2 začátkem letního semestru. Kvůli pandemii Covid-19 však bylo zrušeno, a proto bylo v rámci omezených možností provedeno v této bakalářské práci. Samostatná sekce je věnována systému načítání a hodnocení pracovních listů, jelikož se ukázal být největší slabinou aplikace. Závěr analýzy je věnován funkčním a nefunkčním požadavkům, které byly stanoveny v počátku prvotní analýzy aplikace, jejich rozboru a zhodnocení.

2.1 Charakteristické vlastnosti aplikace

V této sekci budou shrnuty základní pojmy a principy, na kterých je aplikace založena. Nejprve bude definována grafomotorika a popsán způsob, jakým aplikace pomáhá v jejím rozvoji.

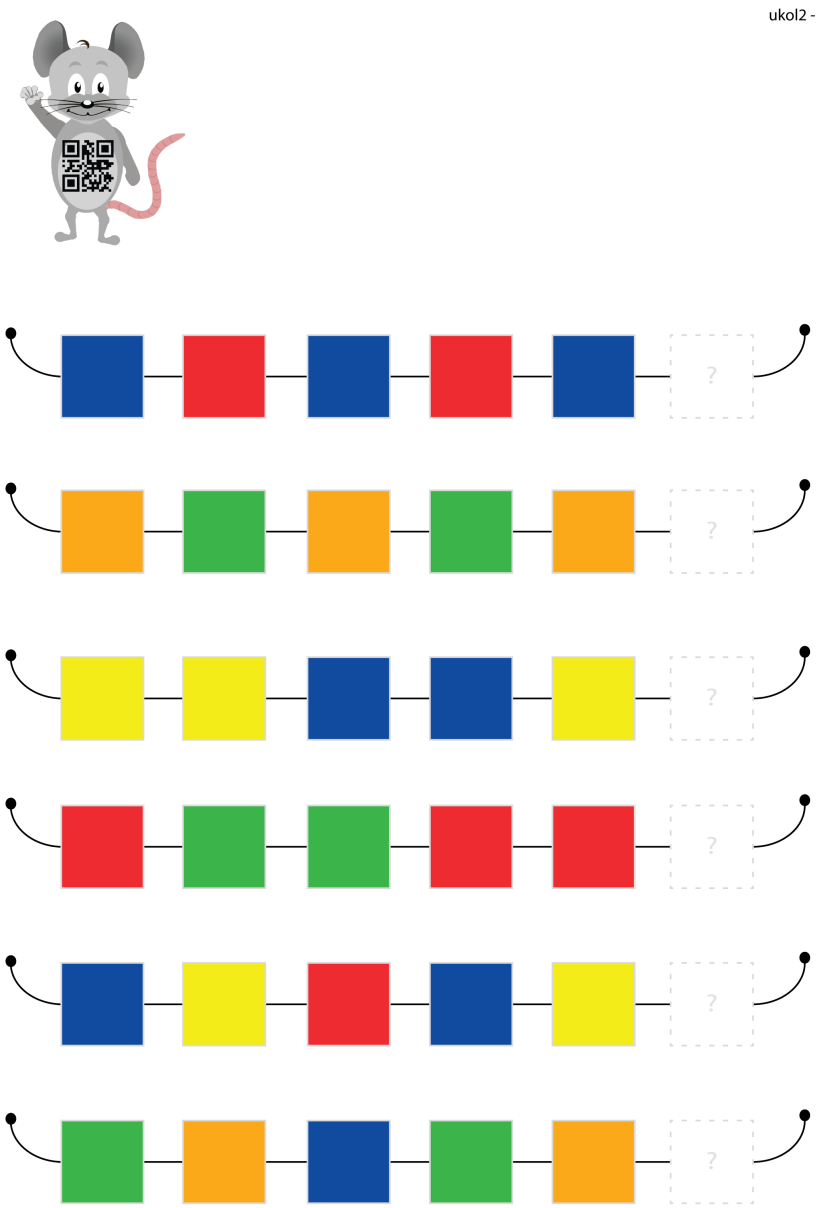
„Grafomotorika je hybnost jemných svalů dlaně a prstů, jejichž vzájemná koordinace umožňuje psát, kreslit, malovat či rýsovat.“ [3]. Jedna z věcí, které jsou důležité při rozvoji grafomotoriky, je rozlišování tlaku, který je vyvíjen na tužku, a správný úchop psacího náčiní [3]. Tato schopnost je důležitá při psaní.

Aplikace pro rozvoj grafomotoriky se na trhu vyskytují málo a grafomotoriku rozvíjejí jen omezeně. Příkladem může být GraphismeMaternelle [4]. Obsahuje grafomotorické úlohy, při kterých dítě obtahuje prstem na obrazovce tabletu různé tvary. Dotyková obrazovka ale neučí výše zmíněnému rozlišování tlaku působeném na tužku při psaní.

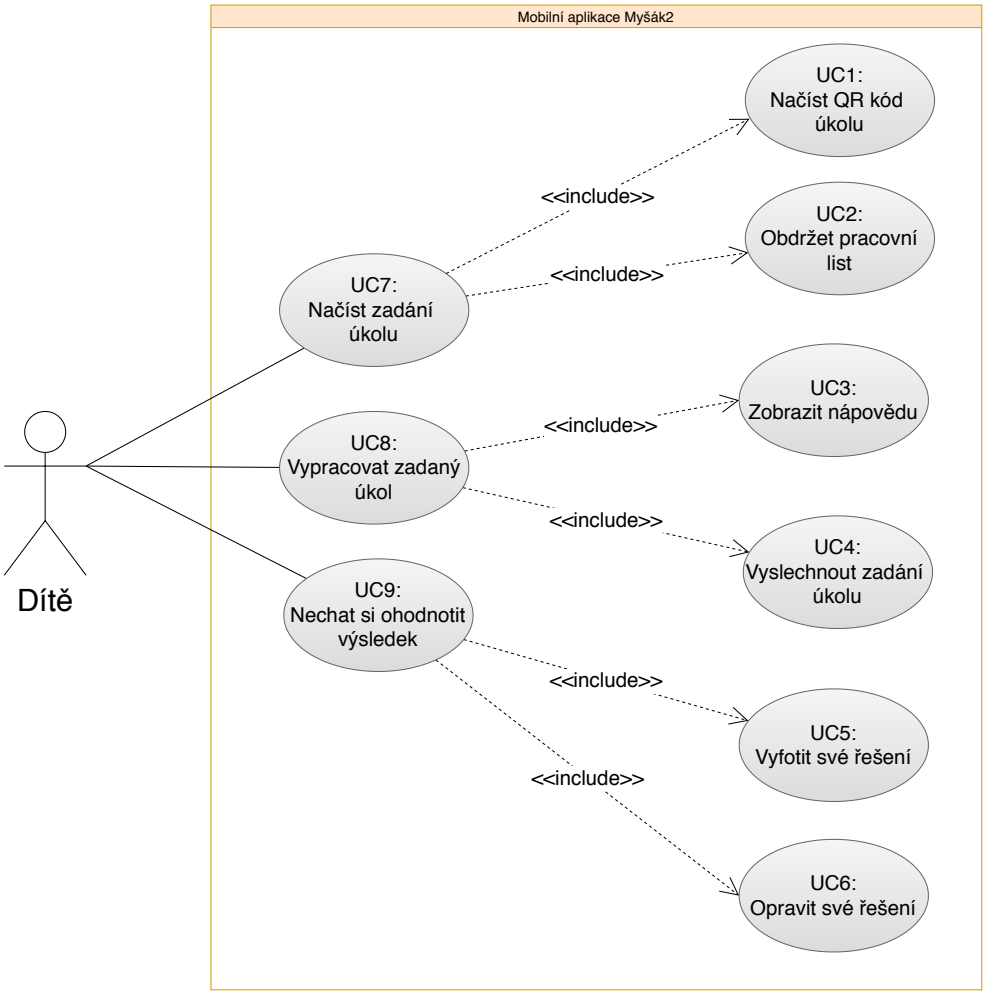
Procvičování grafomotoriky kreslením prstem po dotykové obrazovce není dostatečné a aplikace Myšák 2 se touto cestou nevydává. V aplikaci Myšák 2 jsou proto úlohy v tištěné formě pracovních listů a děti je vyplňují pomocí tužky, pastelek, či barevných fix. Aplikace slouží k vysvětlení zadání a ohodnocení těchto úloh pomocí postavy myšáka. Ten ve školce supluje roli učitele, který se může více věnovat pomalejším dětem, zatímco ostatní mohou pracovat samostatně. Aplikace není primárně určena pouze pro školky a může být používána i v domácím prostředí. V něm může roli učitele hrát rodič

Každý pracovní list obsahuje QR kód, který slouží jako jednoznačný identifikátor úlohy. Dítě jej načte kamerou na tabletu, čímž spustí hlasové zadání úlohy. Po vyslechnutí zadání odloží tablet stranou a zpracovává úkol na pracovním listu. Při nedostatečném pochopení zadání si může tablet vzít a kliknout na nápovědu. Pokud ani nápověda nepomůže, zavolá na pomoc učitele, který zadání buď dovysvětlí nebo přidělí dítěti jiný pracovní list. Když je dítě se svým řešením spokojeno, může jej zkusit odevzdat. Odevzdání je prováděno vyfocením pracovního listu pomocí fotoaparátu zařízení.

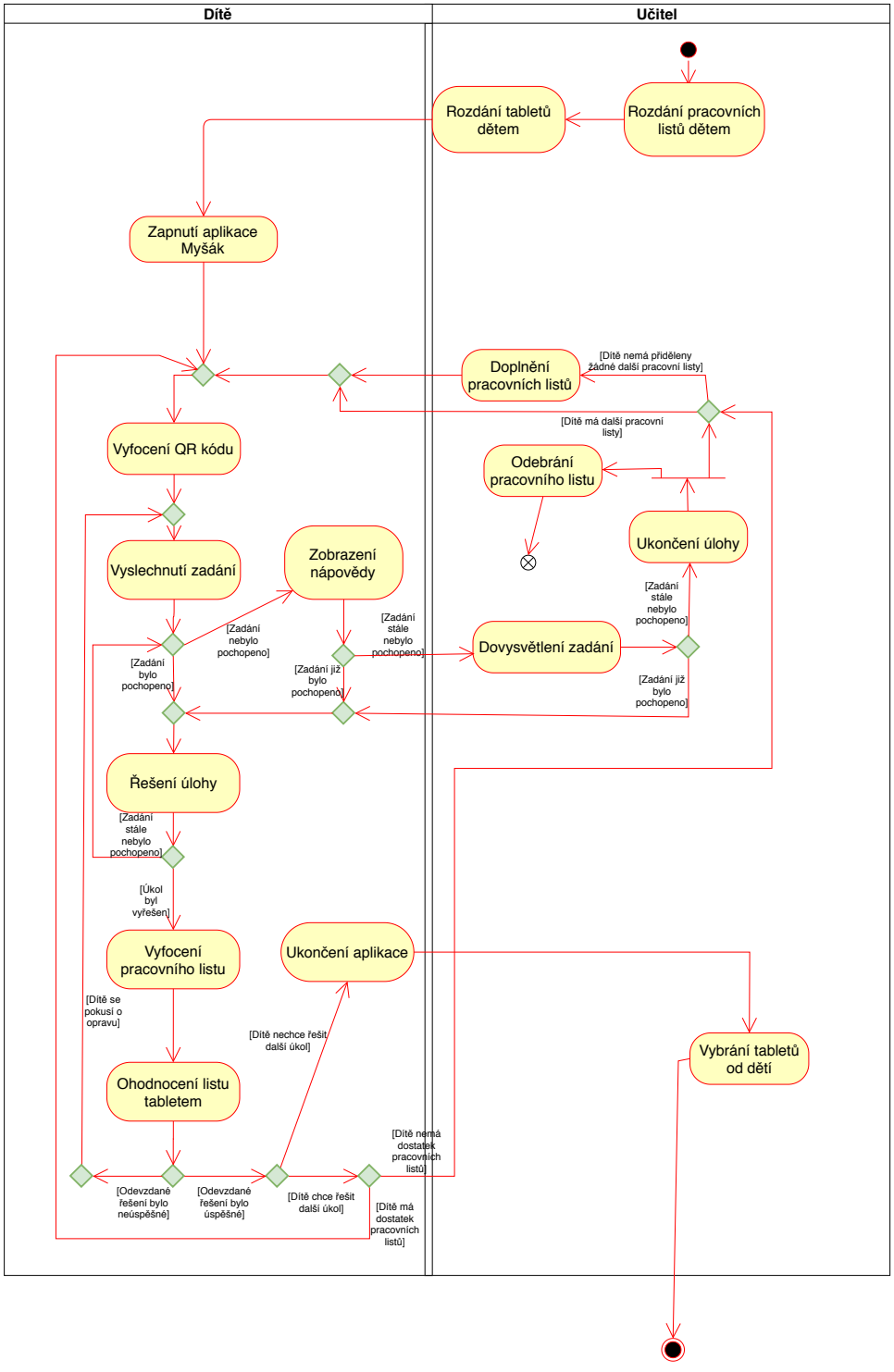
Aplikace pomocí knihovny pro rozpoznávání obrazu OpenCV analyzuje vyfocený snímek a sdělí uživateli, zda je jeho řešení uspokojivé, či ne. Dítě má poté možnost pokusit se řešení opravit nebo pokračovat na úvodní obrazovku a řešit další úlohu. Detailnější popis toho jak dětský uživatel interaguje s aplikací znázorňuje Use Case diagram 2.2. Průchod aplikace dětským uživatelem zachycuje Activity diagram 2.3.



Obrázek 2.1: Ukázka pracovního listu



Obrázek 2.2: UseCase diagram použití aplikace



Obrázek 2.3: Activity diagram průchodu aplikací

2.1.1 Vyřazené účty

Většina diagramů, které popisují původní aplikaci a jsou použity v této bakalřské práci při analýze, není převzata ze softwarového projektu, který byl realizován v rámci předmětů BI-SP1 a BI-SP2. Důvodem je to, že od konce předmětu BI-SP1 nebyly nijak aktualizovány, přestože se aplikace znatelně změnila. Jelikož tyto diagramy počítají s rozdělením aplikace do sekcí podle uživatelských účtů, stává se většina z nich neaktuální. Proto v rámci této bakalářské práce vznikly nové diagramy, které lépe reprezentují původní stav aplikace. Uživatelské účty, které aplikace obsahovala sestávaly ze 3 kategorií:

Child Profil dětského uživatele. V dětské sekci měl uživatel po přihlášení přístup k načtení úkolu a standardnímu průchodu aplikací, tedy vypracování, vyfocení a vyhodnocení úkolu. Odevzdaná řešení a úspěšnost uživatele se měly ukládat do jeho profilu (v databázi).

Teacher Profil učitele. Jeho sekce měla sloužit ke správě dětských profilů v aktuální třídě. Učitel měl podle úspěšnosti daného žáka vybrat pro něj vhodný úkol. Ten nepřiděloval prostřednictvím aplikace, ale rozdáním příslušných pracovních listů k úkolům, které měl mít v aplikaci možnost zobrazit.

Guest Byl zamýšlen pro vpuštění do aplikace bez nutnosti přihlášení. Průchod aplikací byl totožný s profilem dítěte. Neukládal odevzdaná řešení ani žádné statistiky.

Účty s ohledem na cílovou skupinu uživatelů přinášely mnoho komplikací a nakonec byly po diskuzi se zadavatelem projektu ve druhé iteraci BI-SP2 zahozeny [5]. Účty měly původně sloužit k větší individualizaci uživatelů při řešení úloh. Měly umožňovat zobrazení statistik a úspěšnosti jednotlivých dětí a úloh. Posláním aplikace je však řešení úloh na pracovních listech, tedy kreslení, a k tomu nejsou uživatelské účty klíčové.

Hlavní komplikací bylo přihlašování dětí, protože většina předškoláků neumí číst ani psát. Pekárková ve své knize uvádí: *„Před 6. rokem dítě projevuje zájem o čísla, případně písmena“* [3], z čehož nelze čtení a psaní považovat jako prerekvizitu předškoláka. Přihlašování pomocí jména a hesla je proto nevhodné.

Jako řešení se nabízelo přihlašování pomocí QR kódu. Každé dítě by mělo svůj unikátn, který by ho po načtení tabletem automaticky přihlásil do profilu. Toto řešení bylo dlouho považováno za uspokojující, než se naskytla otázka, jak řešit některé krajní situace. Jednou z nich může být to, když dítě načte cizí QR kód [5]. Vyskytuje se v cizím profilu. Ve stejnou chvíli se může pokusit přihlásit i skutečný majitel účtu. Tato situace by byla obtížně detekovatelná i řešitelná a pro učitele by to znamenalo další práci navíc. Pokud by dítě QR kód ztratilo, polilo nebo ho jakkoliv znehodnotilo, musel by učitel zjistit, o jaký kód se jednalo, a znovu ho vytisknout, což by zdržovalo od plnění úkolů.

Mohlo by se navíc stát, že učitel sám vytiskne špatný kód. QR kódy jsou od sebe těžko rozeznatelné a selhání tohoto druhu by se řešilo obtížně.

Ve druhé závěrečné zprávě se píše *„Implementaci a celkové review uživatelských účtu tedy ponecháme do navazujících bakalářských pracích.“* [5]. Tato práce se další analýzou a návrhem řešení účtů zabývat nebude. Účty jsou zmíněny pouze proto, že jejich předchozí začlenění do projektu a následné odstranění zneplatnilo veškeré diagramy a závěry pocházející z projektu až do druhé iterace SP2. Pozůstatky uživatelských účtů navíc přežily i přes jejich vyřazení do finální podoby aplikace a budou ještě dále zmiňovány.

2.2 Frontend

Sekce se zaměřuje na grafické rozhraní aplikace a technologie pro něj použité. Pozornost je věnována především estetické stránce aplikace, tedy tomu, co vidí uživatel aplikace.

2.2.1 Wireframy

Tato sekce se zabývá detailnějším popisem jednotlivých obrazovek v aplikaci. Popsané nedostatky uživatelského rozhraní budou následně ověřeny uživatelským testováním.

Grafická část aplikace v poslední vydané verzi vychází z wireframů vytvořených v 1. iteraci předmětu BI-SP2, kdy byly definitivně odstraněny uživatelské účty. Wireframy jsou k dispozici jako přílohy této práce. Dále je možné je nalézt na adrese `https://tinyurl.com/MysakWireframes`. V aplikaci se obrazovky od wireframů podstatně neliší. Většina obrazovek používá společný layout. Ten představuje pozadí, dva fragmenty určené pro interaktivní obrázky, animace a dolní lištu, která slouží k navigaci mezi obrazovkami. Fragment animací slouží postavě myšáka, která je do něj vykreslována.

Při spuštění aplikace se zobrazí úvodní obrazovka, na které se myšák představí. Po kliknutí kamkoliv na obrazovce se uživatel dostane do hlavního menu, ve kterém může aplikaci buď skončit nebo spustit výběr úkolu Na další obrazovce myšák vyzve uživatele, aby načetl QR kód, který se nachází v levém horním rohu papíru. Popisuje ho jako magickou známku, protože dítě v předškolním věku nemusí vědět, coto QR kód je. Na animaci pak názorně ukáže, jak tuto magickou známku načítá. Pokud doje k úspěšnému načtení kódu, zobrazí se obrazovka zadání. myšák zadání řekne a čeká až uživatel úkol dokončí. Při řešení je možné zobrazit nápovědu k zadání. Na další obrazovce myšák předvede, jak vyfotit své řešení. Poté je vpuštěn do fotoaparátu, ve kterém má vyfotit svůj vyřešený pracovní list. Po úspěšném vyfocení myšák ohodnotí úkol.

Pokud je načten nevalidní náhodný QR kód, který se neváže k žádné úloze, je zobrazena písemná hláška. Dětský uživatel neumí číst a pokud by náhodou k této situaci došlo, nejspíše by byl zmaten.

S ohledem na cílovou skupinu uživatelů je aplikace málo komunikativní. Na úvodní obrazovce se představí hlavní postava myšáka a čeká, až se uživatel dotkne obrazovky. To mu není nijak sděleno a mohl by čekat na další instrukce myšáka. Následující obrazovka představuje něco jako hlavní menu, ve kterém uživatel může vybrat novou úlohu, či ukončit aplikaci. Na tuto obrazovku se uživatel vrací po vyřešení úlohy.

V obrazovce před focením myšák názorně předvede uživateli, jak má načíst QR kód. Měl by ale uživateli říci, aby následně klikl na ikonku fotoaparátu na dolní liště, jelikož animace s úkolem se přehrává pořád dokla a uživatel by mohl očekávat další instrukce od myšáka.

Problém s nepochopením toho, co má uživatel dále udělat, kvůli nedostatečnému vysvětlení navigace, může nastat na každé obrazovce. Zkušenější uživatel aplikace si poradí, ale pro nezkušeného uživatele může být ovládání neintuitivní. S ohledem na cílovou skupinu uživatelů aplikace by bylo vhodné přidat funkci, která bude globálně zapínat přídatné vysvětlivky. S nimi bude myšák vysvětlovat navigaci na každé obrazovce (říkat uživateli co má dělat po každém kliknutí), případně vysvětlovat úlohy konkrétněji. Pro zkušenější uživatele by naopak měla existovat možnost přeskakovat většinu instrukcí. Pokud již vyřešil dvacet úloh, je zbytečné, aby mu bylo znovu vysvětleno, jak má načítat QR kód.

Nejnedostatečnější je poslední obrazovka, na které probíhá vyhodnocení vypracovaného úkolu. myšák v ní pouze oznámí, jestli bylo řešení správné či nikoliv a v případě neúspěchu nabídne uživateli, jestli chce svou úlohu opravit. Uživatel v tuto chvíli ví, že je jeho řešení špatné, ale nezjistí, v čem udělal chybu, a tedy bude těžko své řešení opravovat. Pokud uživatel udělal chybu, měl by mít možnost zobrazit správné řešení.

2.2.2 Animace a postava myšáka

Hlavní postava myšáka je limitována svým technickým řešením. Myšák je zobrazován v podobě video animací, které se přehrávají na obrazovce ve smyčce. Tyto animace lze rozdělit na dvě kategorie. Animace myšáka obsahují pouze model myšáka. Vytvořeny byly v 3D grafickém softwaru (Maya), a následně rendrovány jako video do formátu mp4. Druhá kategorie animací jsou nápovědy k úkolům, které myšáka nezobrazují. Na každé má myšák kolem sebe zelené pozadí, které je při zobrazování v aplikaci odfiltrováno pomocí knihovny AlphaMovieView. Tento způsob animací přináší mnoho kritických omezení a komplikací.

Mezi tato omezení patří:

Nesjednocený vzhled postavy Myšák nevypadá na všech animacích stejně. Například při mluvení je větší, než při kreslení. To mohlo být způsobeno tím, že animace byly dodávány postupně a mezitím se upravoval a zdokonaloval i model myšáka. Z čehož plyne další omezení.

Rozšiřitelnost Pokud by byl model myšáka upraven, musely by být všechny animace všech videí znovu přerendrovány, aby se zabránilo nesjednocenému vzhledu postavy (viz předchozí bod). Samotný model myšáka se navíc v souborech aplikace nevyskytuje.

Použité technologie Knihovna pro odfiltrování zeleného pozadí z videa, bez které by se animace nezobrazovaly správně, je téměř tři roky neaktualizovaná, přestože obsahuje několik chyb. Nejvýraznější z nich je probliknutí černého snímku před spuštěním animace. Tato chyba se projevuje především na starších a pomalejších zařízeních.

Animace nápověd nezobrazují myšáka a napovídají uživateli ukázkovým postupem řešení. Například pro úlohu s vybarvováním korálků je názorně ukázáno vyplnění posledního barevného čtverečku. Zobrazují se na nové obrazovce při stisknutí tlačítka s nápovědou. Tyto animace pro zobrazení nevyžadují problematickou knihovnu AlphaMovieView, a mohou být proto ponechány do nové verze aplikace.

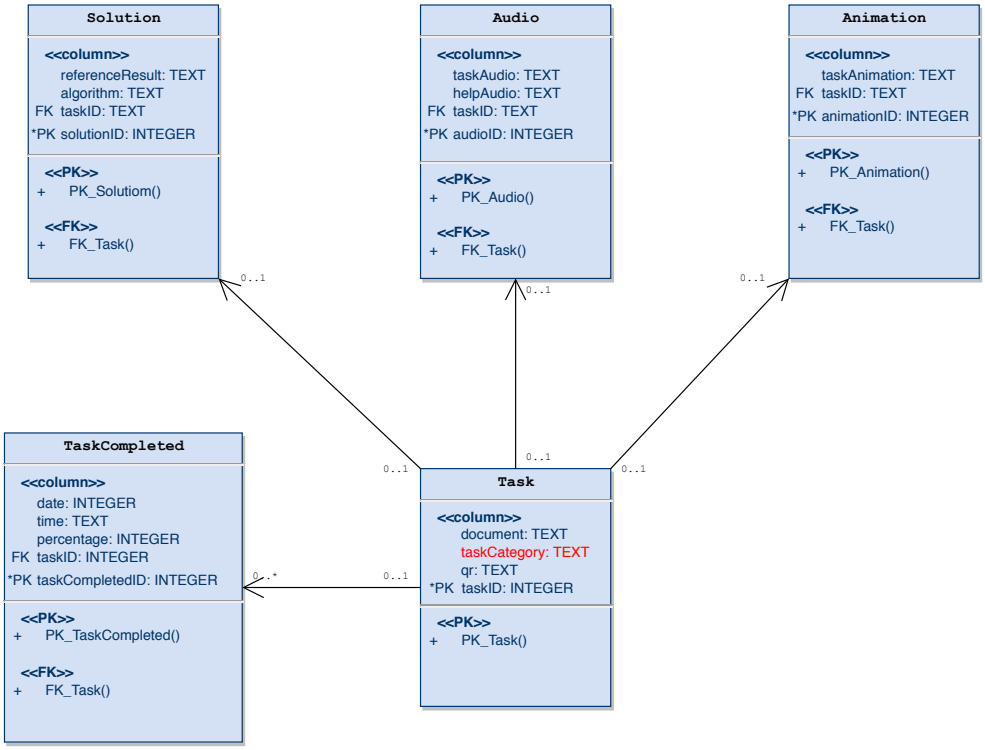
Oproti tomu animace postavy myšáka vyžadují volbu jiného řešení. Nejvhodnější je vykreslovat na obrazovku přímo 3D model myšáka, k čemuž je potřeba zvolit vhodnou technologii. Volbou této technologie se zabývá sekce 3.3 z následující kapitoly.

2.3 Databáze a backend

Aplikace používá lokální relační databázi SQLite k uložení některých souvisejících dat. Všechny tabulky v databázi se týkají zadání úloh a jejich odevzdaných řešení. Ke každé úloze je uloženo hlasové zadání, animace a řešení. Tabulky animací a hlasového zadání neuchovávají data samotná, ale pouze cesty k multimediálním souborům, které jsou umístěny ve složce assets. Řešení obsahuje serializovaná data ve formátu JSON. Hlavním účelem databáze je usnadnění načtení všech dat, která se váží ke konkrétnímu zadání úkolu.

2.3.1 Databázový model

Databázový model se v průběhu softwarového projektu neaktualizoval, přestože datová část aplikace za svůj vývojový cyklus prošla několika zásadními změnami. V něm i v databázi samotné zůstaly i do finální verze tabulky uživatelských účtů a tabulky s nimi související. V databázi zůstaly i další nepoužité tabulky jako například tabulka `Tag`, jejíž účel nelze dohledat. V rámci analýzy byl vytvořen databázový model 2.4, který odpovídá původnímu stavu aplikace. Nepoužité tabulky v něm byly vynechány.



Obrázek 2.4: Původní databázový model

Databázový model 2.4 popisuje tyto tabulky:

Solution Vzorové řešení, konkrétní úlohy. Jeho formát je různý pro každou kategorii. Ukládá se jako textová položka ve formátu JSON. Atribut `algorithm` měl sloužit k ukládání stringové reprezentaci své funkce, která by se poté volala pomocí reflexe. Od toho bylo nakonec upuštěno, jelikož algoritmus řešení úlohy je pro danou kategorii úloh stejný.

Audio Obsahuje cestu ke zvukovému souboru, který je uložen ve složce assets.

Animation Obsahuje cestu k video souboru, které je uloženo ve složce assets.

Task Reprezentuje pracovní list. Uchovává informace o zadání úlohy.

TaskCompleted Odevzdané řešení úlohy. Ukládat se měla všechna řešení úloh, tedy i ta špatná. Implementace ukládání úloh nebyla realizována.

Nejkritičtějším bodem databáze jsou zadání úloh. Každá úloha má unikátní umělý klíč, textovou reprezentaci seedu pro QR kód a vazby na tabulky se vzorovým řešením, animací a zvukovou stopou pro zadání úkolu. Zadání se dělí do kategorií, podle kterých je zvolen algoritmus na zpracování jejich řešení. Tyto kategorie jsou v databázi řešeny přidáním textového atributu `TaskCategory` do tabulky `Task`.

Animace s nápovědou a hlasové zadání stopy se reálně váží pouze ke kategorii a pro všechny úlohy z kategorie jsou tudíž stejné. Toto řešení znepřehledňuje relace mezi entitami a způsobuje mnoho zbytečných vazeb, kdy všechny úkoly se stejnou kategorií mají v databázi uložen cizí klíč ke stejnému hlasovému zadání.

Zavádějící je i vazba mezi úkolem a zvukovou stopou daného zadání. V tabulce `Audio`, která slouží k reprezentaci zvukové stopy, jsou uloženy dvě cesty k zvukovým souborům – jedna k hlasovému zadání a druhá k nápovědě. Tato tabulka má však reprezentovat konkrétní zvukovou stopu, která nutně nemusí patřit k zadání. Chyba je způsobena nedostatečnou abstrakcí. Přirozenějším řešením by byla existence tabulky `Help`, která by reprezentovala nápovědu. Zadání (Task) by mělo několik nápověd (Help) a každé zadání i nápověda by měly vlastní hlasovou stopu.

Nedostatky v databázi nejsou pro chod aplikace kritické, ale jsou limitující především pro její budoucí rozšiřování. Databázový model bude proto přepracován.

2.3.2 ORM

Objektově-relační mapování (dále jen ORM) je programovací technika, která se používá při konverzi relačních dat na objekty v objektově orientovaném programování [6]. ORM tak odstiňuje programátory od psaní SQL dotazů a ručního mapování relačních dat na objekty. Aplikace myšák pro toto mapování používá framework SugarORM [7]. Tabulky v databázi jsou reprezentovány entitními třídami. K jejich mapování na tabulky využívá framework reflexi a odstiňuje uživatele nejen od psaní SQL dotazů, ale dokonce snižuje počet anotací, které je nutno k entitním třídám psát.

Jedinou povinou anotací je `@Entity` se jménem příslušné tabulky. Atributy entitních tříd musí dodržovat stejné názvy jako jejich příslušné sloupce v tabulkách v databázi (není-li explicitně určeno jinak pomocí volitelných anotací). Framework používá pro presistenci dat modifikovaný návrhový vzor Row Data Gateway, kdy Data Access Object (dále DAO) i entitní třída splývají v jeden objekt. Nabízí jednoduché api, které každé entitní třídě poskytne

předdefinované CRUD operace. Api je jednoduché, intuitivní a to i pro programátory bez zkušeností s ORM. Toto řešení komplikuje testování. Framework má několik nedostatků, které budou nyní popsány.

Absence podpory vazeb

Podpora vazeb One to Many a Many to Many je slabá. Přestože Many To Many vazby byly autorem knihovny již v roce 2012 označeny jako brzy dostupné, nebyly doposud implementovány. Mapování pro tyto vazby je tedy potřeba naprogramovat ručně, k čemuž však není poskytnuto kvalitní api.

V rámci softwarového projektu se tento nedostatek neprojevil, protože od všech složitejších vazeb, navržených v databázovém modelu, bylo nakonec upuštěno a jsou používány pouze vazby One To One.

Dokumentace

Dokumentace pro SugarORM je nedostatečná a jediné informace lze dohledat na oficiálních stránkách [7]. Ty obsahují odkaz na repozitář frameworku, kde uživatel nalezne, ve kterém uživatel nenalezne o moc více informací. Největším zdrojem informací je zde issue tracker repozitáře. V oficiální dokumentaci je popsán pouze jeden ukázkový příklad mapování vazby One to One.

Není ani vysvětleno, jak funguje reflexe, pomocí které se tabulky mapují. Programátor bez zkušeností s tímto frameworkem se snadno dostane do nesnází, pokud se pokusí namapovat již existující databázi. Nikde není vysvětleno, na jakém principu funguje načítání vazeb typu One to One. Po několika pokusech v rámci této bakalářské práce bylo zjištěno, že se uplatňuje eager loading, avšak v dokumentaci to není nikde zmíněno. Do jaké hloubky se eager loading uplatňuje, též nelze dohledat. Nelze dohledat, ani jak se anotují vazby One to Many, o kterých autor knihovny tvrdí, že jsou podporovány.

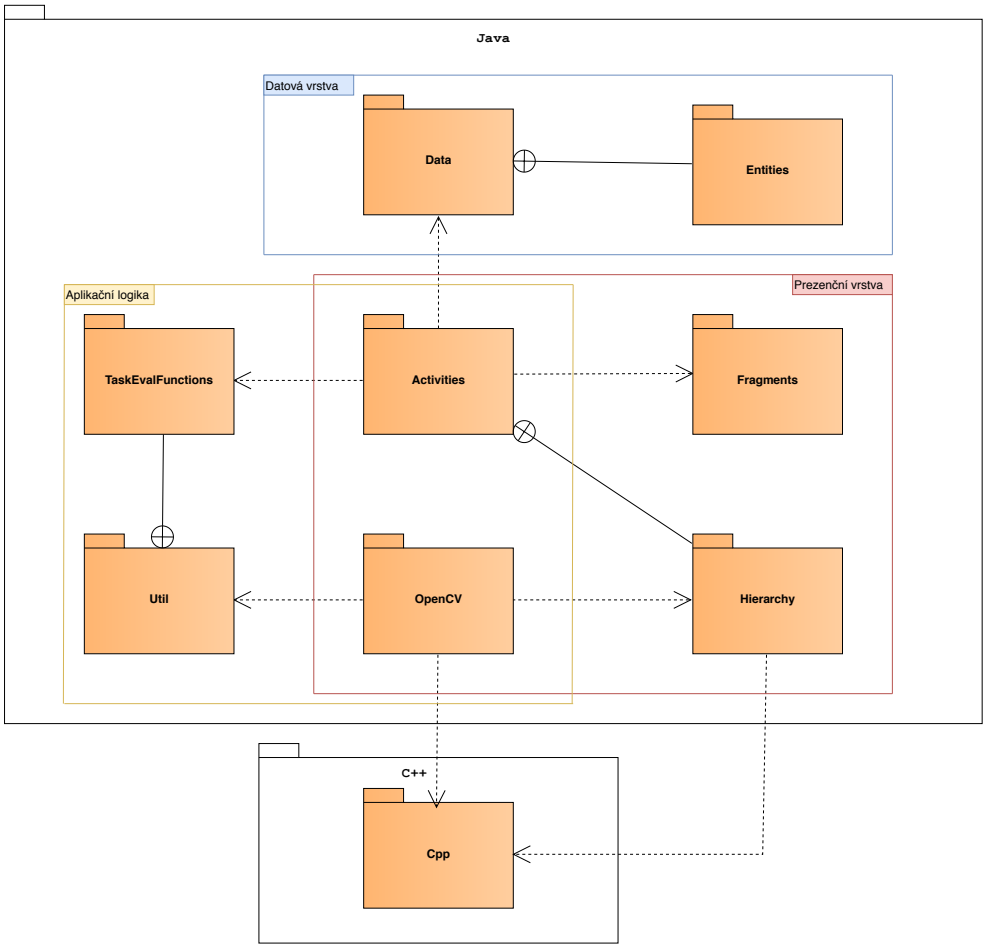
Do nové verze aplikace je vhodné zvážit náhradu frameworku. Proto bude proveden průzkum konkurenčních technologií, na jehož základě bude zvolena vhodná náhrada. Podrobněji se tímto průzkumem zabývá sekce 3.4.

2.4 Architektura a business logika

Tato sekce se zabývá architekturou aplikace, jednotlivými komponentami, rozhraním tříd, jejich vnitřní logikou a vzájemnými vztahy. Většina tříd v aplikaci je napsána v Javě. Pro hodnocení pracovních listů je použita knihovna pro zpracování obrazu OpenCV [8]. Ta je do projektu linkována staticky, poskytuje zkompilované binární soubory pro C++ a Java rozhraní, které se kompiluje současně s aplikací. Třídy, které tuto knihovnu v aplikaci používají, jsou napsány v C++ a volány z nativních funkcí definovaných v Java části aplikace.

Aplikace je rozdělena do tří vrstev:

1. Prezenční
2. Vrstva aplikační logiky
3. Datová vrstva



Obrázek 2.5: Diagram balíčků současné architektury

Implementace je kromě toho rozdělena ještě podle použitých programovacích jazyků na balíčky `Java` (Aplikační část) a `Cpp` (Modul zpracování obrazu). Balíček `Cpp` svým způsobem také patří do vrstvy aplikační logiky. Obsahuje však specifické funkce, které se neváží na zbytek projektu. Mohl by být z aplikace izolován a použit jinde se zachováním své funkcionality, a proto se v diagramu vyskytuje samostatně.
Prezenční vrstva a vrstva aplikační logiky se v některých částech projektu prolínají. To je znázorněno na diagramu balíčků 2.5. Více bude tato problematika rozepsána v následujících kapitolách.

V aplikaci byly ponechány i nepoužité třídy, které byly z implementace vyřazeny. Například se jedná o třídy uživatelských účtů, GUI starého menu, či pomocná třída pro přehrávání animací jakožto sekvence obrázků. Tyto jsou anotovány jako `@Deprecated` a nemají v aplikaci žádnou funkci. Budou proto vynechány ze všech diagramů.

Ve všech diagramech tříd (v celé této práci) budou rozhraní a třídy barevně rozlišeny, a to následovně:

Červené budou třídy, které jsou součástí Android Software Development Kit (dále SDK).

Modré budou třídy, které jsou součástí externích knihoven (SugarOrm).

Zelené budou všechny vlastní třídy aplikace Myšák 2.

2.4.1 Prezenční vrstva

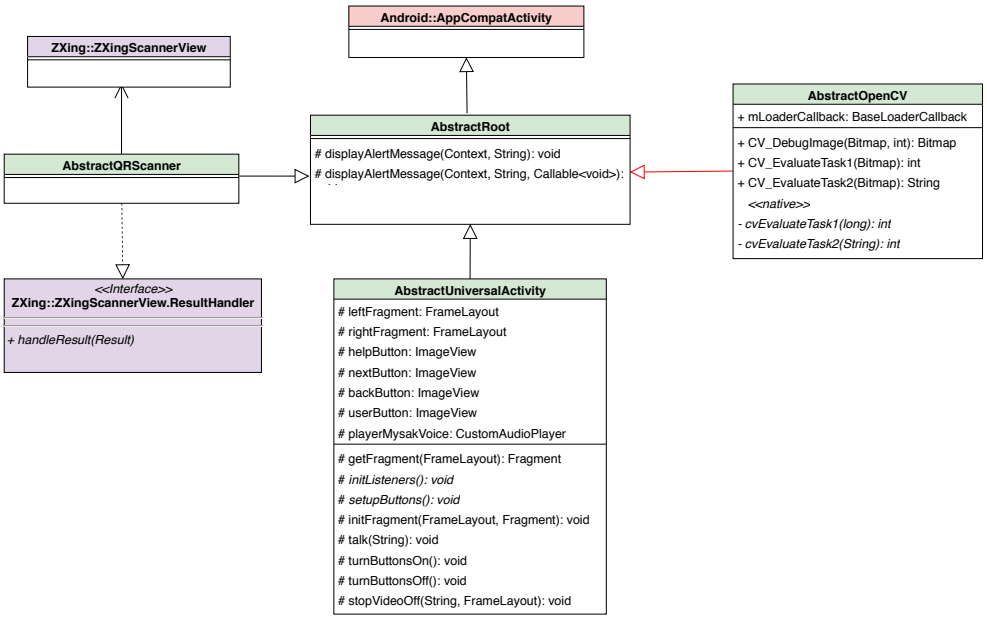
Prezenční vrstvu reprezentují třídy z balíčků `Activities` a `Fragments` (viz 2.5). Definují GUI aplikace, avšak některé jejich třídy do značné míry obsahují i byznys logiku. To je způsobeno špatným návrhem hierarchie dědičnosti.

Activities

Jedná se o balíček obsahující třídy aktivit, které jsou z převážné většiny implementací wireframů a pokrývají implementaci UI. Některé z nich obsahují i business logiku. Každé obrazovce wireframů odpovídá jedna aktivita, přestože je většina obrazovek velmi podobná. Aktivity mají v aplikaci definovanou vlastní hierarchii dědičnosti, aby bylo zamezeno duplikaci kódu. Ta obsahuje mnoho vad. Některé jsou méně závažné, některé způsobují značné komplikace.

Hierarchy

Tento balíček je součástí `Activities` a obsahuje všechny abstraktní třídy použité v hierarchii aktivit. Jediná třída `AbstractUniversalActivity` obsahuje abstraktní metody. Ostatní třídy tohoto balíčku, přestože jsou deklarovány jako abstraktní, žádné abstraktní metody ani atributy nemají. Mohly by proto být obyčejnými třídami.



Obrázek 2.6: Diagram tříd hierarchie. Červeně zvýrazněné vazby představují chyby v dědičnosti

Mateřskou třídou všech aktivit je třída `AbstractRoot`, ze které dědí každá aktivita v aplikaci. Svou nadtřídu `AppCompatActivity` rozšiřuje pouze o metody pro výpis dialogů s varováním, které nejsou k ničemu užitečné a nikde v aplikaci nejsou použity. Je tedy zbytečná a aktivity by mohly dědit rovnou od `AppCompatActivity`. Od `AbstractRoot` dědí abstraktní třídy `AbstractUniversalActivity` a `AbstractOpenCV`, na které se zaměřím kvůli jejich špatnému návrhu.

AbstractOpenCV a narušení vrstev

`AbstractOpenCV` je aktivita, která umožňuje načíst zkompilovanou OpenCV knihovnu do paměti a poskytuje rozhraní pro hodnocení úloh. Návrh třídy je špatný z několika důvodů. Aktivity slouží v aplikaci pouze pro uživatelské rozhraní (dále UI) a neměly by obsahovat žádnou výpočetní logiku, jako je zpracování a ohodnocení pracovních listů. Přestože neobsahuje třída samotné algoritmy pro zpracování obrazu, definuje jejich rozhraní a v aplikaci slouží výhradně k poskytování funkcí pro zpracování obrazu. `AbstractOpenCV` načítá OpenCV knihovnu v rámci svého životního cyklu aktivity v metodě `OnCreate`. Pro zpracování fotky s řešením úkolů je proto nutné definovat aktivitu, která dědí od `AbstractOpenCV`, instancovat ji, přepnout se do této aktivity, a až poté pouze v rámci jejího životního cyklu používat její metody pro zpracování obrazu.

Tím je nevhodně zaručeno, že výpočetní logika zpracování obrazu musí probíhat v aktuálně instancované aktivitě, spuštěné na popředí. Aplikace však používá pro všechny obrazovky společný layout, kterému není tato aktivita přizpůsobena, a jelikož Java nepodporuje vícenásobnou dědičnost, která by zajistila kompatibilitu i s univerzálním layoutem, není ani možné dát tuto logiku do aktivity s plnohodnotným UI.

Aktivita, která by dědila od `AbstractOpenCV`a zároveň chtěla zobrazovat UI v aplikaci, by proto musela duplikovat kód z `AbstractUniversalActivity`. Další možností je vytvořit aktivitu bez UI, která se instancuje, přepne na popředí, poskytne API pro výpočetní operace zpracování obrazu a po jejich provedení se zničí. V aplikaci byl zvolen tento způsob a demonstruje jej aktivita `TaskResolveActivity`. Nepoužívá žádný layout a po své inicializaci ihned spouští aktivitu fotoaparátu, jehož data (fotky) později zpracovává. Tato aktivita se projevuje pouze jako probliknutí černé obrazovky před spuštěním fotoaparátu. Aktivity se po svém spuštění ukládají na zásobník. Pokud chce uživatel po ohodonocení své řešení opravit, musí být tato aktivita při návratu do zadání odstraněna ze zásobníku, aby uživatel nebyl přesměrován na černou obrazovku.

`AbstractOpenCV` neobsahuje implementace metod pro zpracování obrazu, ale pouze jejich definice. Tyto metody jsou nativní. Jejich implementace je napsána v C++ a vyčleněna do samostatného modulu `cpp`. Analýze tohoto modulu se věnuje sekce 2.7

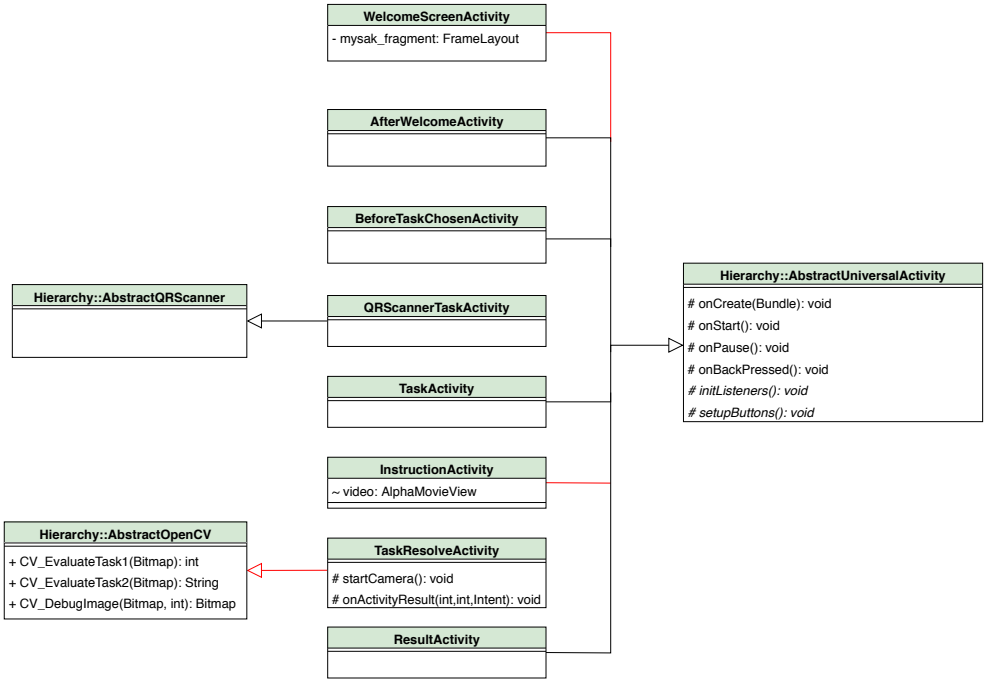
Třída `AbstractOpenCV` by tedy neměla být aktivitou (viz 2.6). Mělo by se jednat o obyčejnou třídu, která by se inicializovala při spuštění aplikace (načetla OpenCV knihovnu) a poskytovala metody zpracování obrazu.

Univerzální aktivita a chyby v dědičnosti

`AbstractUniversalActivity` (dále univerzální aktivita) je abstraktní třída uživatelského rozhraní, která poskytuje základní implementaci potřebnou pro používání univerzálního layoutu všem třídám, které od ní dědí. Třída obsahuje dva FrameLayouty, do kterých se dynamicky nahrávají fragmenty. Od této třídy dědí aktivity `WelcomeScreenActivity` a `InstructionActivity`, které univerzální layout se dvěma fragmenty nepoužívají. Obsahují tak metody pro instancování a obsluhu fragmentů, které však reálně neobsahují. Od univerzální aktivity zdědily i definice tlačítek pro dolní lištu a nápovědu. Tlačítko nápovědy uvnitř nápovědy nedává smysl.

Jedinou zděděnou metodou, kterou `InstructionActivity` používá, je funkce pro přehrávání hlasu myšáka a ve svém layoutu definuje místo fragmentů rovnou přehrávač videa. Pokud by se v této třídě zavolala jakákoliv jiná metoda zděděná od univerzální aktivity, došlo by pravděpodobně k pádu celé aplikace. Podobně je tomu u `WelcomeScreenActivity`, která ve svém layoutu definuje navíc další fragment. Společně se zděděnými atributy má v sobě definovány tři

FrameLayouty, z nichž dva reálně neobsahuje. Obsahuje i definici dolní lišty a tlačítek, přestože se v obrazovce této aktivity dolní lišta nevyskytuje. Při snaze použít v těchto třídách zděděné metody od AbstractUniversalActivity dojde u většiny z nich k pádu aplikace.



Obrázek 2.7: Diagram tříd aktivit odvozených z hierarchie. Červené vazby představují chyby v dědičnosti

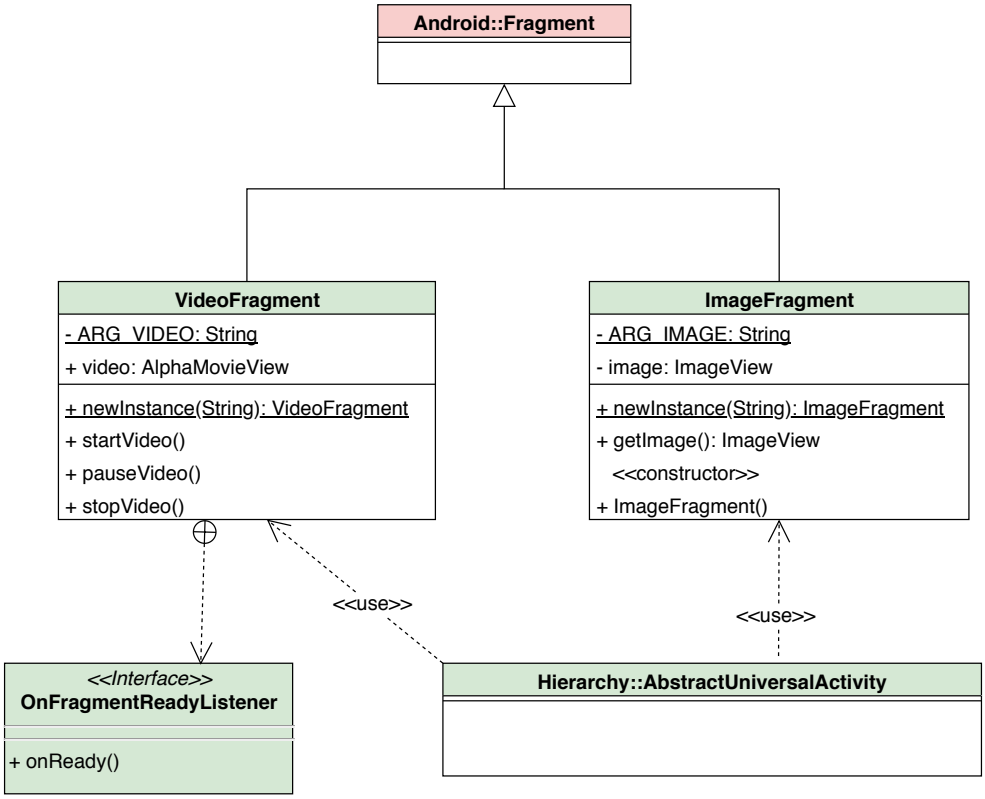
Fragments

je balíček, ve kterém jsou definovány fragmenty. Ty jsou používány aktivitami, které dědí od univerzální aktivity. Aplikace používá dva typy fragmentů:

1. ImageFragment

2. VideoFragment

Layout obou typů fragmentů představuje pouze obrázek nebo video, které svými rozměry kopíruje rodičovský layout fragmentu.
Ukázkové použití `ImageFragmentu` je v možné vidět v `AfterWelcomeActivity`, kde je do fragmentu dynamicky vkládán obrázek pracovních listů. Pomocí fragmentu `VideoFragment` jsou pak na každé obrazovce přehrávány animace myšáka.



Obrázek 2.8: Diagram tříd fragmentů

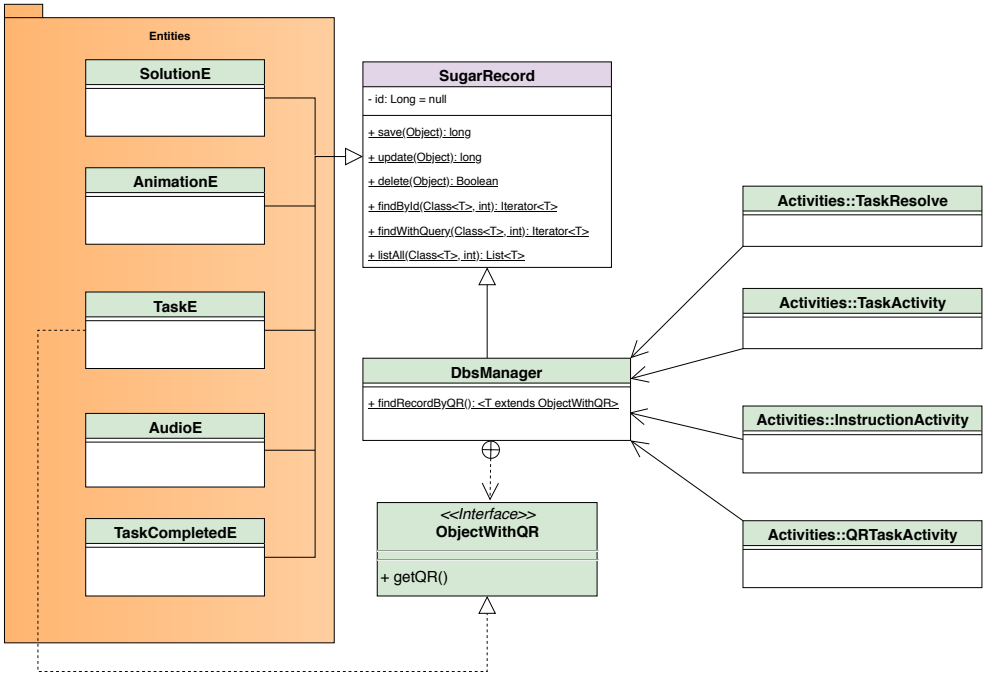
2.4.2 Datová vrstva

Balíček Data

V tomto balíčku jsou umístěny entitní třídy, které využívá SugarORM pro reprezentaci tabulek v databázi. Dále obsahuje třídy, které slouží k verzování a nahrávání databáze do zařízení.

Entitní třídy v diagramu 2.9 odpovídají reprezentaci tabulek Databázového modelu 2.4. Obsahují stejnojmené atributy, gettery, settery a bezparametrický konstruktor. Jejich popis není ničím zajímavý, a je proto vynechán

DatabaseHelper je třída, která slouží k verzování databáze a k jejímu nahrání do zařízení po spuštění aplikace. Databáze se po každém spuštění aplikace přepíše tovární verzí, která je zabalena v apk aplikace. Aplikace ve své výsledné podobě do databáze nic neukládá.Pokud by se však do databáze ukládala vypracovaná řešení, byla by po restartu aplikace ztracena. Protože je třída dobře navržena, není nijak závislá na Sugar ORM a není složité odstranit nechtěné přepisování databáze, bude ponechána do nové verze aplikace.



Obrázek 2.9: Diagram tříd balíčku Data

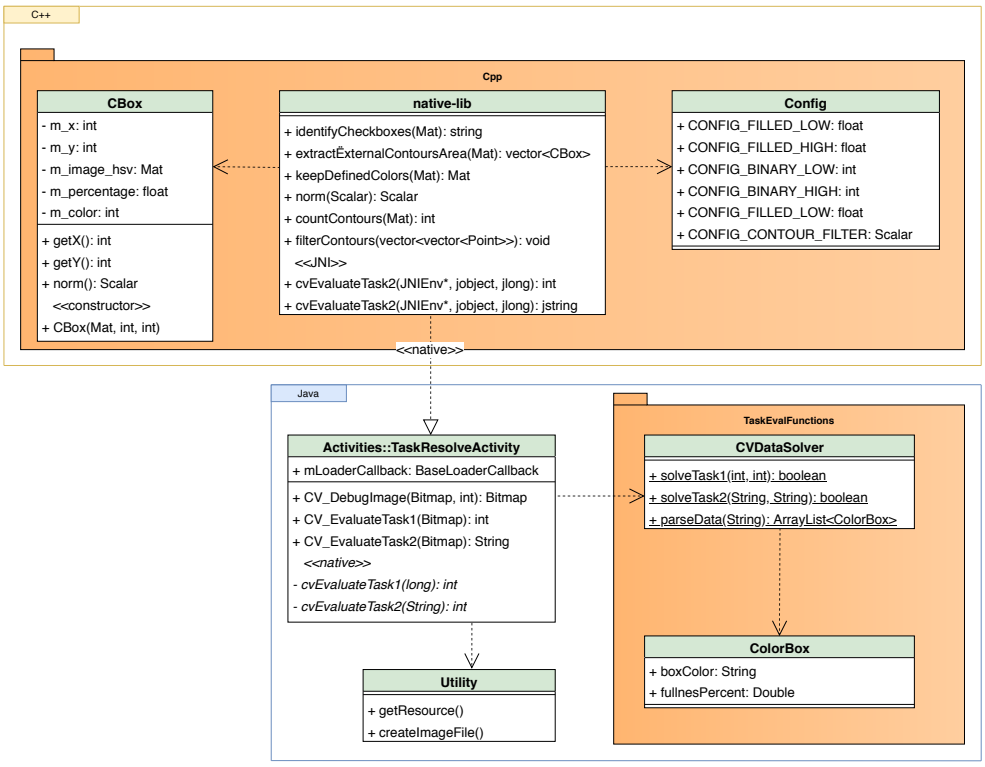
2.4.3 Vrstva aplikační logiky

Do vrstvy aplikační logiky spadají třídy, které se starají o ohodnocení úkolů. Nezpracovávají obraz, ale pouze porovnávají data poskytnutá od tříd v Cpp modulu 2.5 s referenčním řešením. Aplikační logiku představují balíčky `OpenCV`, `Support` a `Util`. Do vrstvy aplikačí logiky spadá s ohledem na její funkcionality i aktivita `AbstractOpenCV`. Jedná se však o aktivitu, což je prvek prezenční vrstvy a je proto dohledatelná v sekci 2.4.1.

OpenCV je složka obsahující pouze jednu aktivitu `DebugOpenCV`, která je potomkem `AbstractOpenCV` a sloužící k debugování úloh. Umožňuje otevřít fotoaparát, vyfotit snímek a zpracovat ho pomocí funkcí pro vyhodnocení úloh. Zpracovaný snímek zobrazí na obrazovku spolu s dodatečnými informacemi detekovanými funkcemi pro řešení úloh. Tato aktivita slouží k debugování úloh a je v rámci standardního průchodu aplikací nedosažitelná.

Support a Util Jsou balíčky s podpůrnými třídami sloužícími k přehrávání zvuku, animací, převádění formátů obrázků, vytváření pomocných souborů ve file systému, či parsování dat. Balíček Util navíc obsahuje třídu `CVDataSolver`, která obsahuje funkce pro hodnocení uživatelských řešení. Tyto funkce porovnávají data detekovaná knihovnou pro rozpoznání obrazu se vzorovými daty

uloženými v databázi. Zde probíhá již zmíněné hodnocení, které se omezuje pouze na dva stavy – úspěch a neúspěch. Neexistuje nic mezi tím a uživatel se nedozví, v čem udělal chybu. To může být zvláště pro dětského uživatele frustrující. Hlavním cílem není procvičovat logiku při řešení jednotlivých úloh, ale jemné tahy tužkou při kreslení. Bude-li mít uživatel možnost dovědět se, v čem udělal chybu, zvýší se tak jeho šance na úspěch. Při současném hodnocení nemá šanci poznat, v čem chybu udělal, což při opakovaném pokusu o odevzdání může vést k demotivaci. V nové verzi aplikace bude vhodné dát uživateli možnost jej odkrýt.



Obrázek 2.10: Diagram tříd aplikační logiky (komunikace mezi C++ a Java částí aplikace)

2.5 Dostupné úlohy

Aplikace nabízí k řešení dvě kategorie úloh. Pro danou kategorii může existovat více různých úloh, které mají společné hlasové zadání, nápovědu a stejný algoritmus pro jejich vyhodnocení. Konkrétní úlohu pak reprezentuje pracovní list. V levém horním rohu každého pracovního listu se vyskytuje QR kód, který jednoznačně identifikuje zadání úlohy. Kód je pro každou úlohu unikátní a slouží jako její identifikátor.

Dostupné kategorie úloh nejsou nijak pojmenovány a v databázi se označují následovně:

Category1 První kategorie se zabývá kreslením různého počtu uzavřených tvarů. Obsahuje pouze jednu úlohu, ve které má uživatel za úkol nakreslit na pracovní list dvě kolečka. Tato kategorie neobsahuje nápovědu.

Category2 Druhá kategorie obsahuje úlohy na doplňování barevných sekvencí čtverečků. Na pracovním listu se nachází řada střídajících se barev. Uživatel má za úkol dokreslit a vybarvit poslední čtvereček tak, aby se hodil do řady. Původní návrh obsahoval několik barevných řad pod sebou. Byl však náchylný na chyby v důsledku špatně vyfoceného řešení, a tak byl zredukován na jednu barevnou řadu na celém papíru.

2.6 Testování původní aplikace

Ve třetí závěrečné zprávě softwarového projektu [9] byly nastíněny situace, které mohou negativně ovlivnit chod aplikace. Jako problematické je zmíněno hodnocení vyfocených pracovních listů. Po konci předmětu BI-SP2 mělo dojít na začátku letního semestru k testování aplikace, které mělo proběhnout ve školce ČVUT a mělo ověřit, zda k problmatickým situacím dochází. Kvůli pandemii Covid-19 však neproběhlo a bylo odloženo na neurčito. Proto bylo v rámci této práce provedeno testování v omezené domácím prostředí na členech rodiny. Nikdo z nich nespadá do cílové skupiny uživatelů, avšak jak vyplývá z výsledků testování, pro potřeby této práce to nebylo nutné. S aplikací nikdy předtím do styku nepřišli a pouze věděli o její existenci. Tato sekce se zabývá shrnutím výsledků z tohoto testování a vyvozením závěrů z něj.

Focení řešení

Ve všech úlohách nezávisle na kategorii se projevovaly nedostatky způsobené focením vypracovaných listů. Focení samotné totiž klade velkou zodpovědnost na uživatele. Je potřeba, aby vyfocený snímek zabíral výhradně plochu papíru, nebyl rozmazaný a pracovní list na tomto snímku nebyl natočený. Z výsledků testování plyne existence mnoha scénářů, kdy může být po vyfocení správné řešení ohodnoceno jako chybné a naopak. Jedná se o následující situace.

Snímek obsahuje elementy prostředí Tato situace může nastat, pokud byl papír vyfocen z příliš velké dálky. Na snímku mohou být nechtěně zachyceny elementy z prostředí, jako například část stolu, na kterém papír leží. Ta se může chybně interpretovat v obou kategoriích úloh. Je-li element dostatečně tmavý, první kategorie úloh jej může rozpoznat jako uzavřenou křivku. V druhé kategorii může být interpretován jako barevný box.

Snímek nepokrývá celou plochu pracovního listu K této situaci může dojít, když je řešení vyfoceno příliš zblízka. Takový snímek nemusí obsahovat celé řešení.

Snímek je rozmazaný Tato situace je obtížně řešitelná, jelikož se v ní projevuje i kvalita kamery zařízení. Pro uspokojivé rozpoznání obrazu je vhodné použít alespoň qHD rozlišení 540×960 pixelů. Rozmazaný snímek může vzniknout i při použití vysokého rozlišení, například pokud se uživateli třese ruka a stiskne spoušť v nevhodný okamžik.

Snímek byl pořízen ze špatného úhlu Tato situace má mnoho podob a výrazný vliv především na úlohy z druhé kategorie. Papír může být na snímku různě natočen v závislosti na pozici, ze které byla fotka pořízena. Více bude tato situace popsána v kapitole 2.7

Hodnocení a pracovní listy

Z testování vyplývá, že chyba není pouze v kvalitě snímku pracovního listu.

V první kategorii docházelo k nerozpoznání nakreslených koleček, pokud byly jejich linie příliš tenké. Místo toho byly jako uzavřené křivky detekovány pouze části obrazce. Výsledkem byla detekce mnohem většího množství křivek, než se jich skutečně nacházelo na listu. Pro dosažení správného výsledku bylo nutné, aby byla kolečka nakreslena tlustou čarou.

Ve druhé kategorii úloh docházelo k řadě chyb:

- Často bylo detekováno mnohem více barevných čtverečků, než se ve skutečnosti vyskytovalo na snímku.
- Žlutá barva, přestože byla téměř vždy správně rozponána, udávala příliš nízké hodnoty zaplněnosti čtverečku (jen 10%).
- Barevné čtverečky byly seřazeny v nesprávném pořadí.
- Zadání některých úloh bylo nejednoznačné a mohlo mít 2 různá logická řešení. Aplikace však uznávala pouze jedno z nich.

Obrazovky uživatelského rozhraní

Po několika průchodech aplikací věděl každý uživatel, jak má řešit úlohy a co má v konkrétním obrazovce dělat. Přesto se potvrdily hypotézy vznesené v analýze uživatelského rozhraní (viz 2.20.

- Na úvodní obrazovce, kde se myšák představí, čekali všichni testovaní uživatelé na instrukce a nevěděli, že mají kliknout na obrazovku.
- V poslední obrazovce se všichni uživatelé opakovaně pletli v tom, kterou šipku mají zmáčknout pro opakování pokusu, a kterou pro návrat do menu. Uživatelé se opakovaně vraceli do menu v domění, že šipka vpřed vede k návartu do aktuální úlohy.
- Téměř všichni uživatelé potřebovali na poprvé podrobněji vysvětlit ovládání aplikace a jednotlivé obrazovky.
- Testovaný uživatel číslo 3 si zezačátku pletl načítání zadání s focením řešení a snažil se při načíání QR kódu zabírat svůj výsledek.
- U cyklicky opakujících se animací focení čekali uživatelé několik jejích cyklů, než poznali, že se nic nového nedozví a napadlo je pokračovat.

Uživatelé neměli závažnější potíže s nalezením a vyplněním správného řešení. Aplikace ho však téměř vždy chybně vyhodnotila jako špatné. Celková úspěšnost uživatelů při řešení úloh byla vysoká, avšak úspěšnost ohodnocení jejich řešení aplikací byla nízká (3 %). Navíc ke korektním hodnocením, která aplikace provedla, došlo pouze omylem a lze říci, že se její schopnost poznat správné řešení blíží nule.

S ohledem na všechny tyto skutečnosti je tedy lepší, že původní aplikace nepřišla do ruky předškolním dětem, které by z hodnocení myšáka byly pouze zklamané a nešťastné. U uživatelů, kteří byli testování se projevoval značný úpadek motivace a zájmu řešit další úlohy, poté co byla jejich správná řešení opakovaně hodnocena jako chybná. Všichni testovaní uživatelé byli přesvědčeni ve správnost svého řešení. Předškolák by však chybovost aplikace nemusel poznat a byl by pouze zmaten, kdyby bylo jeho správné řešení hodnoceno jako špatné.

Hlavním cílem nové verze aplikace bude odstranit velkou zátěž kladenou na uživatele při focení vypracovaného pracovního listu. Ta bude přesunuta na samostatný modul, který bude pracovní list rozpoznávat a načítat sám. Na konci třetí iterace softwarového projektu započal návrh takového modulu. Podrobněji je rozebrán v sekci 2.7.3.

2.7 Zpracování obrazu

Jak již bylo zmíněno, aplikace používá pro zpracování řešení úloh knihovnu OpenCV. V aktivitě `AbstractOpenCV` jsou definovány předpisy nativních funkcí pro zpracování úloh podle kategorií. Jejich implementace se nachází v balíčku `cpp`, který obsahuje všechny třídy modulu pro zpracování obrazu. Těmi se zabývá i tato sekce analýzy. Modul obsahuje tyto třídy:

native-lib Obsahuje implementaci všech metod pro zpracování obrazu

Cbox Reprezentuje jeden barevný čtvereček v úloze druhé kategorie. Každý čtvereček je reprezentován souřadnicemi, které odpovídají pozici jeho levého horního rohu v obrázku, ve kterém byl detekován. Dále ukládá informace jako je převažující barva a její procentuální zastoupení. Detekce předchozích informací je prováděna v konstruktoru.

config Hlavičkový soubor s konstantami a globálními proměnnými. Většina z nich je použita pro rozpoznání sekvencí barevných čtverečků.

Třídy mají slabou dokumentaci a v algoritmech jsou často používána magická čísla, u kterých nelze odvodit jejich účel. Komunikace s java částí aplikace je znázorněna v diagramu 2.10

Modul aplikaci limituje kvůli své implementaci v jazyce C++. Kvůli němu musí být zpracovaná data serializována do JSONu a předávána Java jako řetězec. Zpracování obrazu v C++ je rychlejší, avšak čas, který je získán rychlostí zpracování obrazu se ztrácí na serializaci. Při serializaci velkých dat pak tento čas převáží rychlost zpracování. I reprezentace dat samotných je komplikovaná, neboť je nutné držet definice některých tříd v aplikaci dvakrát. Příkladem mohou být třídy CBox a ColorBox, viz diagram 2.10. V podstatě reprezentují to samé, ale jedna je napsána v C++ a druhá slouží jako DTO v Javě. Jako řešení se nabízí provádět i ohodnocení uživatelského řešení v C++. Ohodnocené řešení se však ukládá do databáze a vytvoření ohodnocené instance CompletedTask by tak muselo být též prováděno v modulu Cpp a následně serializováno do Java části aplikace. Současné řešení navíc znemožňuje komunikovat v průběhu vyhodnocení s uživatelem. Nové úlohy by se klidně v budoucnu mohly vyhodnocovat postupně, zobrazovat uživateli jeho chyby a vysvětlovat mu řešení v průběhu vyhodnocení. Samotné zpracování obrazu v Javě je stejně na pozadí prováděno v C++. Java rozhraní OpenCV knihovny obsahuje pouze wrappery, které na pozadí volají stejnojmenné nativní funkce původního C++ API knihovny. Použití OpenCV v Javě tedy vytěžuje procesor pouze větším počtem nativních volání. Ta jsou ve většině případů zanedbatelná [10] (str. 126). V nové verzi aplikace by bylo vhodné použít následující postup:

1. Zpracování obrazu stejně jako hodnocení úloh přepsat do Javy (Kotlinu).
2. Ukáže-li se nějaká sekce jako kritická na výkon, bude přepsána do C++.

2.7.1 Algoritmy pro zpracování obrazu

Algoritmy se váží na úlohy definované v sekci 2.5.

První kategorie spočívá v detekci počtu uzavřených křivek vyskytujících se na vstupním obrázku. K tomu slouží funkce countContours definovaná v modulu `native-lib`. Princip rozpoznání uzavřených křivek spočívá v nalezení kontur, neboli obrysů. *„Obrysy lze definovat jako křivky spojující všechny spojité body, které mají stejnou barvu nebo intenzitu“* [11]. K detekci kontur slouží funkce findContours z knihovny OpenCV. Kontury se dají dále použít k detekci tvarů. Detekce tvarů však v úlohách první kategorie neprobíhá.

Druhá kategorie spojuje úlohy, jejichž cílem je dokreslit a vybarvit chybějící barevné čtverečky v řadě. Algoritmus kromě detekce uzavřených křivek rozlišuje i barvy čtverečků. Detekce barev probíhá pomocí barevného modelu HSV, který je popsán v kapitole 2.7.2.

Reprezentace HSV modelu v OpenCV je odlišná od standardu používaného v grafických aplikacích. Odstín barvy je udáván v polovičním rozmezí od 0 do 180 stupňů, sytost a světlost barvy je místo v procentech určena hodnotou od 0 do 255, kdy 255 reprezentuje maximální hodnotu 100 %. Ve třídě config jsou definovány rozsahy jednotlivých barev pomocí standardního modelu HSV a při použití jsou normalizovány, aby odpovídaly formátu, který používá OpenCV knihovna. Rozsahy jsou definovány rozdělením barevného spektra na 6 barev:

Barva	Dolni hranice	Horni hranice
Red	(330, 0.40, 0.25)	(30, 1, 1)
Yellow	(30, 0.40, 0.25)	(90, 1, 1)
Green	(90, 0.40, 0.25)	(150, 1, 1)
Cyan	(150, 0.40, 0.25)	(210, 1, 1)
Blue	(210, 0.40, 0.25)	(270, 1, 1)
Magenta	(270, 0.40, 0.25)	(330, 1, 1)

Tabulka 2.1: HSV rozsahy barev definované v třídě config.h

V tabulce lze zpozorovat, že se rozsahy barev překrývají ve svých hraničních hodnotách. Mějme například barevný čtvereček z druhé kategorie úloh s HSV hodnotami (30, 0.7, 0.7), který odpovídá oranžové barvě. Dle definovaných rozsahů mohou být jeho pixely interpretovány jako červené i jako žluté zároveň.

Tato chyba se nijak neprojeví při detekci barev, protože jsou jednotlivé barvy detekovány postupně. Pro každou barvu jsou separovány detekované pixely do samostatné černobílé bitové masky. Pixel, který spadá do rozhraní, má v masce hodnotu 1 a ostatní pixely 0. Tyto masky jsou vytvořeny pro každou barvu definovanou v konfiguraci a používají se k odfiltrování všech ostatních barev z pozadí. Jejich bitovým součtem (or) vznikne maska, která

obsahuje na všech pozicích pixelů barevných čtverečků hodnotu 1. Pixel detekovaný dvakrát se tedy neprojeví na výsledku, jelikož bit na jeho místě bude místo jednou, sečten dvakrát s jedničkou a výsledná hodnota v bitové masce bude v obou případech 1.

Chyba je však nežádoucí při určování převažující barvy čtverečku. Pixely této hraniční oranžové barvy budou započteny dvakrát – jednou jako červené a jednou jako žluté. Výsledné procentuální zastoupení barev ve čtverečku pak bude v součtu dávat přes 100 %.V nové verzi aplikace bude vhodné opravit rozsahy a přizpůsobit je hodnotám barev, které jsou použity při tisku pracovních listů. Bylo by též vhodné rozšířit paletu detekovatelných barev.

Algoritmy druhé kategorie úloh trpí na určení správného pořadí čtverečků. Jejich detekce probíhá v náhodném pořadí, ale pro správné ohodnocení úlohy musí být seřazeny vzestupně zleva doprava, shora dolů podle svých souřadnic do jednorozměrného pole. Řazení probíhá primárně podle y-ové souřadnice a sekundárně podle x-ové. Pokud bude snímek mírně natočen, budou čtverečky seřazeny v nesprávném pořadí. To je jedna z hlavních příčin neúspěchu při testování.

Navíc samotná implementace jejich řazení je chybná. Při něm je porovnáván součet x-ových a y-ových souřadnic čtverečků. Primárního řazení pomocí y-ových souřadnic je dosaženo jejich přenásobováním magickou konstantou. Při takové implementaci je nemožné používat úlohy, ve kterých je více řad barevných čtverečků (viz 2.1).

2.7.2 Barevný model HSV

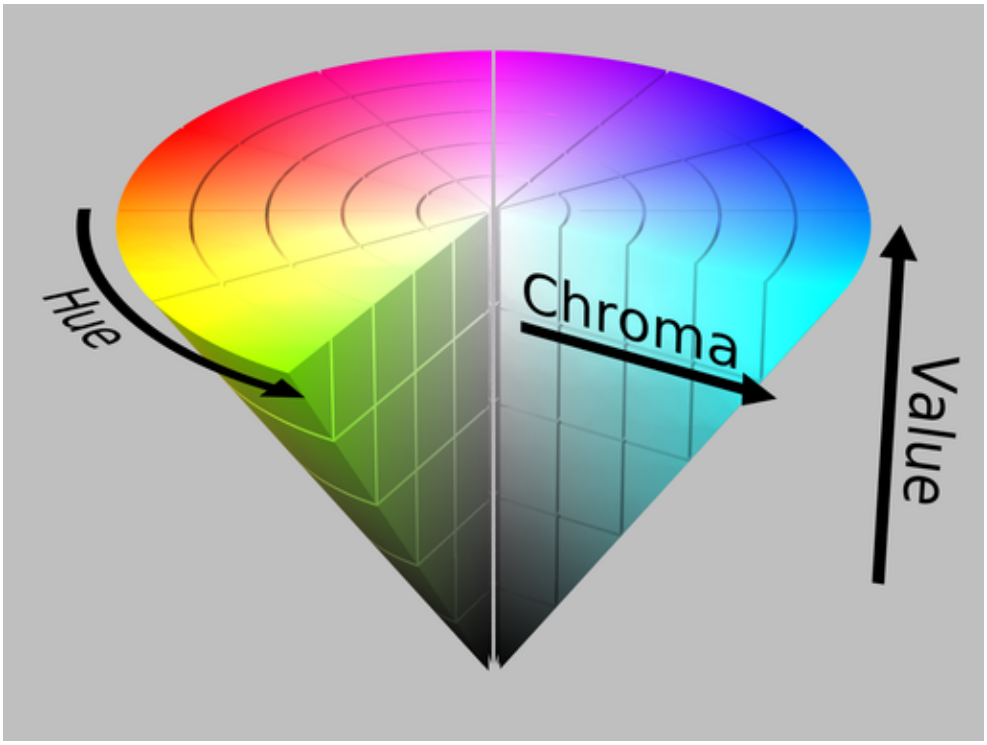
Jedná se o alternativu k modelům RGB a CMYK, které jsou založeny na míchání základních barev. Narozdíl od nich definuje HSV barvy pomocí tří svých složek. Princip modelu HSV je založen na podobném principu, jako vnímání barev lidským okem [12]. Má tyto složky:

Hue neboli odstín barvy, či barevný tón. Ve většině aplikací je udáván ve stupních od 0 do 360.

Saturation Je sytost barvy udávaná v procentech. Procentuální hodnota odpovídá vzdálenosti barvy od neutrální šedé a tedy míru zastoupení barevného odstínu. Barva se sytostí 0 reprezentuje pouze odstíny šedé.

Value je světlost barvy udávaná v procentech. Čím menší je procentuální zastoupení, tím tmavší je výsledná barva. Barva se světlostí 0 reprezentuje pouze černou barvu.

Barevný model HSV je hojně využíván knihovnami pro rozpoznávání obrazu, protože od sebe odděluje barvevný odstín a intenzitu světla. Samotný odstín je definován jednou hodnotou a je tak snadné definovat rozptyl pro konkrétní barvu.



Obrázek 2.11: Barevný model HSV. Obrázek byl převzat z práce [1]

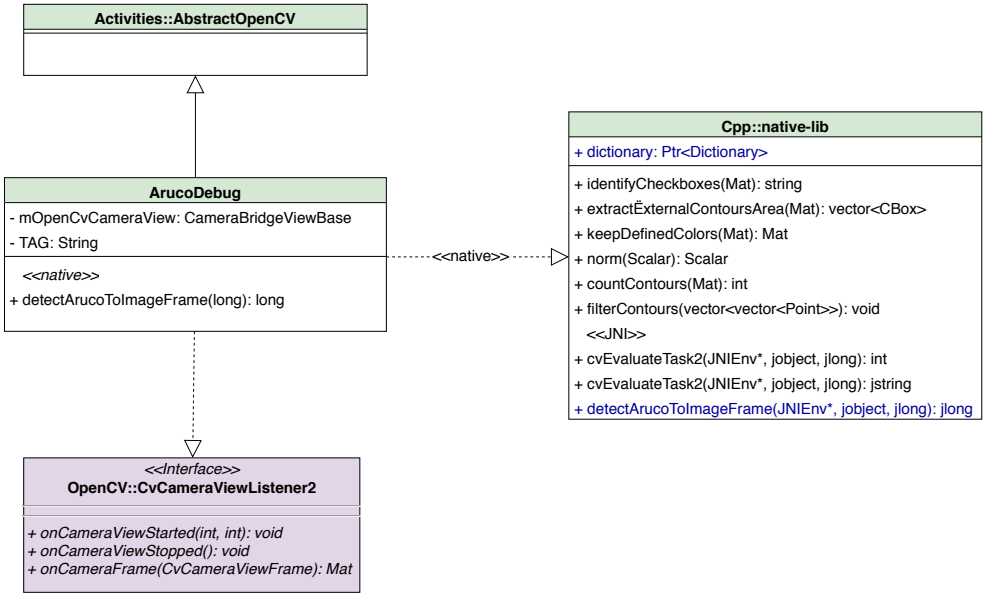
2.7.3 Modul pro načítání pracovních listů

Již v rámci původního softwarového projektu jsem v jeho finální fázi začal pracovat na modulu, který by odstranil nežádoucí negativní vlivy, které vyplynuly z testování, neboť výskyt těchto vlivů byl předpokládán. Klíčové mělo být přidání pozičních značek do každého rohu pracovního listu, pomocí kterých by byl list načítán automaticky. Pro potřeby projektu byly zvoleny planární poziční značky ArUco, protože OpenCV knihovna poskytuje modul, který umožňuje jejich snadnou a efektivní detekci. V repozitáři projektu je možné dohledat práci na tomto modulu ve větvi debug/ArucoDebug [13]. Ve větvi byly provedeny tyto změny:

ArucoDebug byla nově přidaná aktivita, která měla za úkol číst snímky z kamery. Do každého snímku byla zakreslena ohraničení pozičních značek. K tomu byla definována nativní metoda implementovaná v C++.

native-lib byla rozšířena o implementaci zmíněné nativní metody. Ta uměla na vstupním snímku detekovat a zvýraznit poziční značky.

Návrh tohoto modulu reprezentuje diagram 2.12.



Obrázek 2.12: Diagram tříd rozpracovaného modulu

Detekované značky nebyly nijak dál zpracovávány. Modul je uměl pouze najít a zakreslit do snímku, který mu byl předán na vstupu. Načítání listu pomocí těchto pozičních značek nebylo nijak navrženo ani implementováno a bude jedním z hlavních cílů této bakalářské práce.

Jedním z důvodů nedokončení tohoto modulu byly potíže s přidáním modulu ArUco na detekci značek do OpenCV knihovny. OpenCV nabízí některé své moduly jako extra. Ty nejsou součástí standardní distribuce knihovny, která je již sestavená a volně dostupná ke stažení. Jedním z těchto modulů je právě ArUco. Aby bylo možné přidat extra moduly do OpenCV, je nutné:

1. Vytvořit si kopii obou repozitářů – knihovny i extra modulů
2. Pomocí cmake nalinkovat tyto extra moduly ke knihovně
3. Vytvořit vlastní distribuci (build) knihovny
4. Nahradit tento build za stávající verzi OpenCV v aplikaci

Vytvoření vlastní distribuce knihovny zabralo nejvíce času a do konce softwarového projektu nebyla vytvořena taková, která by fungovala univerzálně na jakékoliv stanici. Navíc je pro správně fungování knihovny nutno zajistit kompatibilitu pro různé procesorové architektury. Aby bylo možné OpenCV knihovnu spustit na převážné většině tabletů a telefonů s OS Android, je nutné vytvořit sestavení pro 32bitovou a 64bitovou architekturu procesorů ARM. Pro zajištění plné kompatibility je nutné brát v potaz i tablety a mobilní telefony, které používají procesory s architekturou x86, jako je třeba

Asus ZenFone 2. V repozitáři aplikace se vyskytuje pouze sestavení pro procesory ARM s 32-bitovou architekturou, které navíc obsahuje chybné linkování. Do některých hlaviček OpenCV knihovny byly nalinkovány závislosti na soubory z lokální stanice, na které sestavení probíhalo. Ukázka chybného linkování je vidět na obrázku 2.13. Pochází z hlavičkových souborů OpenCV knihovny z větve ArucoDebug z repozitáře projektu [13]. Na ostatních zařízeních nebylo možné projekt s touto distribucí OpenCV zkompilovat.

```
#ifndef __OPENCV_ARUCO_HPP__
#define __OPENCV_ARUCO_HPP__

#include <opencv2/core.hpp>
#include "../../../../../../../../Android/Sdk/ndk/20.0.5594570/toolchains/llvm/prebuilt/linux-x86_64/sysroot/usr/include/c++/v1/vector"
#include "opencv2/aruco/dictionary.hpp"

/**
 * @defgroup aruco ArUco Marker Detection
```

Obrázek 2.13: Ukázka špatně nalinkované OpenCV knihovny

V repozitáři extra modulů je uveden návod na kompilaci knihovny, který však nepopisuje celou řadu krajních situací a chyb při kompilaci, které mohou nastat například nesprávným pořadím linkování souborů, či špatným nastavením některých proměnných při konfiguraci. Pro člověka, který nemá v této oblasti zkušenosti, je postup neintuitivní a složitý, zvláště pokud nastanou zmíněné chyby při kompilaci. Návody na internetu jsou často zastaralé a odkazují na použití neaktuálních nástrojů a knihoven, kvůli kterým se může vyskytnout mnoho obtíží při kompilaci. Proto je součástí této práce i manuál 6.4.1 na kompilaci a vytvoření vlastní distribuce knihovny s možnstí přidání extra modulů jako je ArUco.

Jelikož před volbou modulu pro rozpoznání pozičních značek neproběhl žádný průzkum frameworků, které by umožňovaly načítat poziční značky. Stane se tak v této bakalářské práci, aby byla zaručena volba nejvhodnější technologie. Analýze knihoven a frameworků pro detekci pozičních značek se věnuje sekce 3.2.

2.8 Funkční a nefunkční požadavky

Poslední sekce analýzy původního stavu se věnuje shrnutí původních funkčních a nefunkčních požadavků, které byly stanoveny na počátku softwarového projektu v rámci předmětu BI-SP1. Tyto požadavky jsou součástí příloh a zároveň jsou dostupné na adrese `https://tinyurl.com/MysakPozadavkySP`

Požadavky jsou převzaty ze softwarového projektu a doplněny o sloupec stavu indikující, které z nich byly splněny. Tyto požadavky jsou pouze informační a poskytnou náhled na to, kolik z původních navržených požadavků na systém bylo za obě iterace splněno. Některé požadavky jsou označeny jako částečně splněné. Takové byly v převážné míře splněny, ale s výhradami. Například požadavek na vyhledávání úkolů v databázi podle různých vlastností (například obtížnost) byl splněn pouze částečně, neboť úkoly lze vyhledávat, ale pouze podle jejich QR kódu. Následující tabulka 2.2 shrnuje, kolik požadavků bylo procentuálně splněno, kolik pouze částečně, a kolik nebylo splněno.

Stav	Počet požadavků	Zastoupení
Splněno	13	38 %
Nesplněno	18	53 %
Částečně splněno	3	9 %

Tabulka 2.2: Závěrečné zhodnocení funkčních a nefunkčních požadavků za celý průběh softwarového projektu

Mnoho nesplněných požadavků se týká uživatelských profilů, které byly z části implementovány, ale nakonec vyřazeny. Některé nesplněné požadavky jsou zajímavé. Přestože jsou mimo rozsah této bakalářské práce a nebudou v jejím rámci naplněny, je vhodné zmínit je a zvážit pro budoucí rozšíření aplikace. Za zmínku stojí tyto:

Měření za běhu Měření času a zobrazení úspěnosti neposlouží pouze uživateli. Tato data budou cená pro vývojáře při návrhu nových pracovních listů a vyřazování těch příliš obtížných.

Úložiště V požadavcích je uvedena kromě lokální databáze i existence databáze vzdálené. Synchronizace se vzdálenou databází by značně odlehčila aplikaci. V databázi zařízení by byla definována pouze kategorie úloh a tabulky s ní související. Zadání a referenční řešení konkrétní úlohy by bylo stahováno ze serveru. Na server by se zpět posílala ohodnocená řešení, která by bylo možné zobrazit v prohlížeči. Následně by z nich mohla být zpracovávána statistika globální úspěšnosti, čímž by mohlo být identifikováno, které úlohy jsou příliš lehké, a které příliš složité.

V budoucnu, až bude Myšák 2 plně funkčním prototypem se sadou funkčních otestovaných úloh, bude vhodné zvážit i to, zda nepřesunou část výpočetní logiky a dat aplikace na vzdálený server. Na něj by byla umístěna veškerá data, tedy úlohy, jejich kategorie, nápovědy i vzorová řešení. Server by kromě úložiště i poskytoval API pro komunikaci a hodnotil úlohy. Z aplikace Myšák 2 na lokálním zařízení by se stal tenký klient, který by pouze zobrazoval a odesílal požadavky na server. Rizikem by mohl být objem dat, který by byl posílán, a s tím spojené vytížení připojení. Toto řešení by navíc znemožnilo používat aplikaci offline. Za zvážení by stály i všechny situace, kdy dětskému uživateli nepůjde internet a jak na tuto skutečnost bude reagovat.

Herní část V herní části aplikace je uveden požadavek na příběh. Myšák je v aplikaci strohý a pouze uděluje pokyny. Poutavý příběh a rozšířené herní prostředí doplněné například o další postavy a dialogy by aplikaci udělaly poutavější. Příběh a celkový návrh kompletního herního prostředí by byly v budoucích verzích aplikace žádoucí.

Zpětná vazba Z této sekce je nejdůležitějším požadavkem zpětná vazba. Myšák by měl mít možnost zobrazit i textovou variantu zadání. Po vyhodnocení odevzdané úlohy by se uživateli mělo zobrazit správné řešení opět například s možností zobrazit textový výklad. Textové zadání a ohodnocení by mohlo případně pomoci rodičům a učitelům, kdyby dětem pomáhali s řešením. Děti by dále mohly být ve hře motivovány zobrazením herního postupu, či získáváním odměn za splněné úlohy.

Pro novou verzi aplikace budou požadavky upraveny.

Kapitola 3

Analýza technologií pro novou verzi aplikace

Z analýzy původního stavu projektu vyplývá, že používá nevhodné technologie, které je nutné nahradit nebo jejich náhradu alespoň zvážit, nenabízí-li se v současné době vhodnější technologie. Kritické sekce aplikace jsou:

Načítání pracovních listů V aplikaci musí být kompletně přepracován systém načítání pracovních listů. Základní myšlenka automatického načítání pomocí pozičních značek bude převzata ze softwarového projektu (viz 2.7.3). Je nutné analyzovat vhodný systém značek a vhodnou knihovnu, která je bude detekovat.

Technologie pro GUI Aplikace používá k zobrazení hlavní postavy a jejích animací nevhodné a neaktuální technologie (viz 2.2.2). Lepším řešením je zobrazovat a animovat postavu myšáka přímo ve 3D scéně. K tomu je potřeba zvolit vhodnou knihovnu, framework, či herní engine. Tato volba může významně ovlivnit návrh nové architektury.

Databáze a ORM Aplikace používá pro objektově-relační mapování nevhodný framework (viz 2.3.2), a proto budou analyzovány konkurenční technologie za účelem jeho nahrazení.

Při analýze všech knihoven a frameworků budou sledovány některé společné kategorie požadovaných vlastností. Na základě úspěšnosti v těchto vlastnostech bude za každou kategorii přiděleno bodové hodnocení od 0 (nejhorší) do 10 (nejlepší). Z hodnocení bude zhotoven vážený bodový průměr, který bude klíčový při závěrečném výběru technologie.

Pro všechny frameworky budou společné tyto kategorie:

Podpora programovacích jazyků Sledována bude podpora různých jazyků, přičemž preferovaným bude Kotlin (viz 3.1). Android umožňuje nativně vyvíjet části aplikace i v C++. Native development kit (NDK) pro Android je ale omezený a nezajišťuje plnou podporu všech funkcí standardního Android SDK.

Intuitivnost Čím komplexnější a složitější technologie, tím nižší číslo ji bude v této kategorii přiřazeno. Na novou aplikaci Myšák 2 by měli být schopni navázat další vývojáři. Tato návaznost bude o to snazší, čím intuitivnější použitá technologie.

Kvalita dokumentace Kvalita dokumentace hraje důležitou roli při konfiguraci a podpoře při řešení obtíží. Dopad na vývojáře má stejný jako předchozí kategorie.

Velikost uživatelské komunity Větší uživatelská komunita zajistí lepší podporu při řešení obtíží.

Podpora platforem Hodnocena bude podpora napříč platformami. Důraz bude kladen na OS Android.

Každá sekce může obsahovat další specifické požadavky týkající se konkrétní technologie. Na konci každé sekce analýzy budou sledovaným vlastnostem přiřazeny váhové koeficienty a spočten vážený průměr, který bude reprezentovat výsledné hodnocení technologie. Na základě tohoto průměru bude vybrána technologie pro novou verzi aplikace.

Do nové verze byl jako programovací jazyk zvolen Kotlin, který nahradil původní jazyk Java. Důvody této volby popisuje následující sekce.

3.1 Programovací jazyk Kotlin

Kotlin je staticky typovaný jazyk, který je překládán na bytecode a interpretován pomocí Java Virtual Machine (dále JVM). Podle způsobu kompilace rozlišuje několik verzí. Jednou z nich je i Kotlin/Native, která umožňuje provádět cross-platformní kompilaci na platformy Android a iOS [14]. Díky své široké podpoře je vhodným jazykem pro multiplatformní projekty.

Jeho největší výhodou je téměř plná kompatibilita s jazykem Java [15]. Jazyk je navíc navržen tak, aby bylo možné používat jak třídy Javy v Kotlin projektu, tak naopak. Android Studio nabízí možnost konvertovat třídy napsané v Javě do Kotlinu, což může být využito při refaktoringu současné verze aplikace.

Je zde však několik omezení, které mohou ovlivnit vývoj aplikace. Ty lze dohledat v dokumentaci jazyka Kotlin [16].
Mezi mnoho výhod Kotlinu, kterými Java nedisponuje patří například:

Rozšiřující funkce V Kotlinu je možné k již existujícím třídam vytvářet takzvané extension funkce. Může se jednat i o třídy externích knihoven, které mohou postrádat některé funkce, jež by vývojář mohl požadovat.

Přetěžování operátorů Oproti Javě je v Kotlinu podobně jako v C++ možné přetěžovat operátory. Tato funkcionalita bude dobře uplatnitelná hlavně při případném přepisování algoritmů pro zpracování obrazu z C++ do Kotlinu.

Data class Datové třídy, které umožňují snadné kopírování usnadní návrh datové vrstvy. Tyto třídy jsou vhodné pro objektově-relačním mapování.

Funkce vyšších řádů Kotlin pro kolekce poskytuje takzvané Higher-Order Functions. To jsou funkce, které berou jako vstupní parametry jiné funkce. Tyto funkce používají ve své vnitřní implementaci funkcionální přístup a pro kolekce je jich ve standardním API definováno mnoho.

Null safety Kotlin se snaží eliminovat negativní dopady plynoucí z používání null referencí. Kotlin typy dělí na ty, které mohou, a ty které nemohou obsahovat null referenci (nullable types).

Singleton pattern Kotlin umožňuje deklaraci singleton tříd jako konstrukci jazyka. K definici Singleton třídy slouží klíčové slovo `object`.

Kotlin disponuje mnoha dalšími výhodami a vyspělými technologiemi (Delegace, Korutiny). Výše uvedené jsou ty, které budou pro vývoj nové verze nejpodstatnější. Kotlin je od roku 2019 primárním programovacím jazykem pro platformu Android, a proto v něm bude nová verze aplikace napsána.

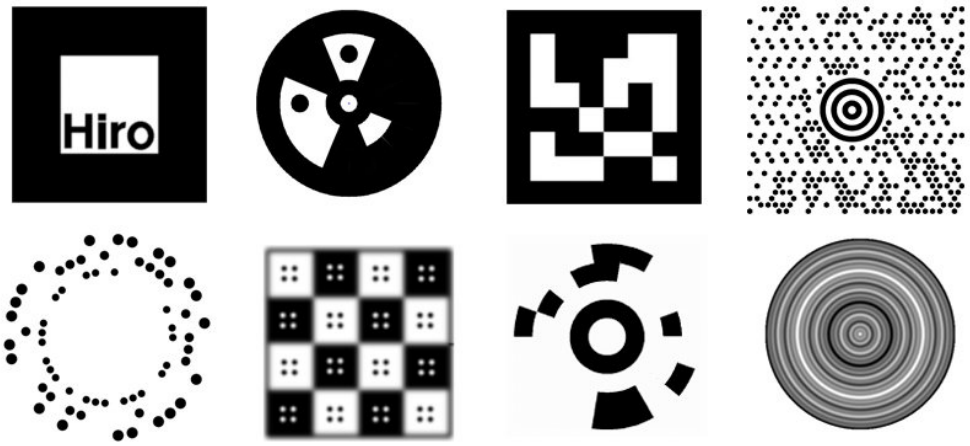
3.2 Analýza technologií pro detekci pozičních značek

V sekci návrhu modulu pro načítání pracovních listů 2.7.3 byla při jeho návrhu zvolena technologie vycházející z ArUco knihovny pro načítání čtverocvých pozičních značek. Pozičních značek je několik druhů a liší se mnoha parametry. Mohou být kruhové a dokonce i nespojité, složené z několika částí. Knihovny, pro jejich detekci mají většinou definován svůj vlastní druh značek. Některé umí detekovat i více různých druhů. Proto bude u knihoven pro jejich detekci kromě společných vlastností ze sekce 3 sledována i následující vlastnost:

Podpora načítání různých druhů značek Široká podpora různých systémů pozičních značek umožní experimentovat s nimi a nahrazovat je, pokud bude detekce neuspokojivá.

Typy pozičních značek

Před analýzou knihoven je potřeba vymezit a definovat poziční značky samotné. *„Mají jedinečné vzory a různé geometrické tvary, ale všechny mohou být automaticky detekovány pomocí kamer. Pozičních značky se používají v oblastech, jako jsou aplikace využívající rozšířenou realitu, medicína, astronomie a robotické úlohy. Řada aplikací určuje kritéria, která charakterizují kvalitativní vlastnosti značky a zahrnují takové hodnotící standardy, jako je odolnost vůči okluzi (viz 3.2), vzdálenost ke značce, falešně pozitivní a falešně negativní míry, citlivost na osvětlení a další.“* [2] Oblast uvnitř poziční značky slouží k uchování nějaké informace, například číselného identifikátoru. Analýza před konkrétními frameworky nejprve shrne různé systémy pozičních značek, které budou brány v potaz při volbě konkrétního frameworku.



Obrázek 3.1: Ukázka různých druhů pozičních značek, převzatá z práce [2]

Obrázek 3.1 znázorňuje různé druhy pozičních značek. Zleva doprava v první řadě se jedná o ARToolKit, Circular Data Matrix, ARToolKit Plus a Maxicode. Druhý řádek též z leva do prava reprezentuje RuneTag, BlurTag, Cantag a Fourier Tag. Poziční značky by měly nejlépe splňovat následující vlastnosti:

Rozlišitelnost Při rozpoznání značky musí být možné zjistit kromě jejích souřadnic i identifikátor. Podle pozice konkrétní značky na snímku bude určena orientace pracovního listu vůči kameře.

Nenápadnost Značky by neměly přitahovat pozornost uživatele a zbytečně jej rozptylovat.

Velikost Značky by neměly zabírat příliš místa na pracovním listu.

Volba pozičních značek bude záviset především na použitém frameworku pro detekci obrazu. Účel této kapitoly je hlavně seznámení s nimi. Pokud žádná konkurenční technologie nenabídne lehčí a intuitivnější API pro detekci pozičních značek, bude použit takový systém, který bude dobře podporovaný knihovnou OpenCV (tedy modul AruCo).

Okluze

Okluze je stav, při kterém z jistého úhlu pohledu dochází k zakrytí viditelné části pozorovaného objektu. Tento stav ovlivňuje detekci pozičních značek. Okluzi dělíme do dvou kategorií [17]:

Vlastní okluze K té dochází, když je z jistého pohledu jedna část objektu uzavřena jinou svou částí.

Meziobjektová okluze Při ní se dva sledované objekty navzájem zakrývají.

Za okluzi může být považováno i nechtěné přeškrtnutí,či zasáhnutí do poziční značky. Odolnost proti okluzi je žádoucí, ale v rámci aplikace by bylo vhodnější, spoléhat na ústní poučení uživatele o významu značek na pracovním listu. Uživatel se tak vyhne, jejich zneplatnění. Dokreslování do pozičních značek je nežádoucí. U značek Runtag a Cantag na obrázku 3.1 by mohlo dojít k tomu, že uživatel do značky zasáhne i nevědomě, jelikož nejsou spojitě ohraničeny. Takové značky jsou pro aplikaci Myšák 2 nevhodné, a bude proto nutné zvážit pouze systémy používající uzavřené, spojitě ohraničené poziční značky. Nejvhodnějšími kandidáty jsou systémy se značkami tvaru čtverce a kruhu jako je například Circular Data Matrix, Aruco nebo ARToolkit.

Při tvorbě pracovních listů bude důležité brát okluzi v potaz a ponechat okolo pozičních značek dostatek bílého místa.

3.2.1 ArUco knihovna

ArUco je open source knihovna, která slouží k detekci čtvercových pozičních značek [18, 19]. Pomocí těchto značek umí například rozpoznat pozici kamery, či kameru kalibrovat. Knihovna je napsána v C++, pro které poskytuje své api. Při použití v Androidu by musely být pro propojení knihovny s Javou vytvořeny nativní funkce stejně, jako tomu bylo v projektu Myšak2. Knihovna obsahuje některé závislosti na OpenCV.

Kromě vlastního systému stejnojmenných značek podporuje i rozpoznání cizích systémů, jako je ArTag, ARToolkit+, Chilltags a AprilTag. Nově knihovna podporuje i fraktální poziční značky [20].

K reprezenatci značek slouží v knihovně třída Marker a k detekci třída MarkerDetector. Různé druhy značek jsou reprezentovány svými slovníky. Pro použití stačí vytvořit instanci třídy MarkerDetector a předat ji slovník značek, které mají být detekovány. Rozpoznání značek je intuitivní, a může být provedeno pomocí jediného řádku kódu. Knihovna navíc umožňuje detekované značky do obrázku zakreslit.

Knihovna disponuje dokumentací v podobě jednoho dokumentu dostupného z [20] a dvou článků na webu researchgate [18, 19]. V nich jsou popsány principy, na kterých funguje rozpoznání značek. Popis principů detekce a tříd knihovny je podrobný a v dokumentaci lze nalézt několik příkladů použití. Knihovně však chybí manuál, který by popisoval proces její kompilace a přidání do projektu.

Uživatelská komunita knihovny je velká. Data v této bakalářské práci vychází ze statistik webu sourceforge.net, na kterém je uložen repozitář. Odkaz k repozitáři se nachází na hlavní stránce knihovny [20]. Celkový počet stažení ke dni 15. 4. 2020 se blíží číslu 127 000. Komunita knihovny je velká, avšak v sekci bugs (sourceforge) je mnoho ticketů – některé i z roku 2013 – a na většinu z nich nikdo doposud neodpověděl. Dá se tedy předpokládat, že při potížích je uživatel knihovny odkázán pouze na dokumentaci a sebe sama.

AruCo je cross-platformní knihovna a lze ji použít na platformách Windows, Android a Mac OS.

Kategorie	Body
Podpora jazyků	4
Intuitivnost	8
Dokumentace	7
Komunita	5
Podpora platforem	7
Systémy pozičních značek	8

Tabulka 3.1: Bodový zisk knihovny ArUco

3.2.2 ArUco modul pro OpenCV

Jak již bylo nastíněno v sekci 2.7.3, OpenCV knihovna možnost přidání extra modulu Aruco, který vychází ze stejnojmenné knihovny uvedené v předchozí kapitole 3.2.1. Jeho přidání do OpenCV v podobě externího modulu [21] usnadňuje práci s linkováním závislostí. Modul se v některých ohledech od původní knihovny liší.

Výhodou použití tohoto modulu je zisk podpory mnoha jazyků. OpenCV nabízí API pro jazyky C, C++, Python, Android, Java a ostatní jazyky založené na JVM, tedy i Scala, Kotlin nebo Closure. Knihovna je napsána v C++ a API pro ostatní jazyky je generováno právě z původní C++ implementace. Každá třída rozhraní obsahuje pouze adresu nativního objektu a definice nativních funkcí, které odkazují na svou původní implementaci v C++. Jazyky založené na JVM používají Java rozhraní OpenCV knihovny, které je založené na Java Native Interface (dále jen JNI).

Api modulu postrádá oproti ArUco knihovně, ze které vychází. Chybí v něm například třídy MarkerDetector a Marker, který v knihovně reprezentuje poziční značku. Značky je též možné detekovat pomocí jediného řádku, avšak funkce z modulu nevrací přímo pole detekovaných značek. Místo něj vrací dvě pole jejích atributů atributů jako své výstupní parametry. Tento způsob je těžkopádný, nepraktický a pravděpodobně byl zvolen kvůli zachování zpětné kompatibility s jazykem C. Kromě absence třídy pro poziční značky však modul podporuje většinu funkcí jako ArUco knihovna.

Dokumentace k modulu je dobrá. Jako ostatní extra moduly má vlastní sekci v oficiální dokumentaci OpenCV knihovny. V ní lze dohledat popisy jednotlivých tříd a funkcí, mnoho návodů a často kladené otázky týkající se detekce značek.

Velikost komunity samotného modulu lze jen těžko odhadnout, jelikož je součástí OpenCV a není u něj vedena žádná statistika použití. Modul je součástí repozitáře společného pro extra moduly. Není tedy vhodné odhadovat dle počtu jeho hvězd (hvězdy určují popularitu repozitáře – analogie lajku na sociálních sítích). Na fórech vývojáři nerozlišují ArUco knihovnu od ArUco modulu pro OpenCV, tudíž ani z nich není možné velikost komunity alespoň odhadnout. Oproti ArUco knihovně však modul disponuje lepší zpětnou vazbou kontributorů modulu. V issue trackeru modulu lze dohledat mnoho aktivních vláken s aktivní zpětnou vazbou [21].

Podpora platforem vychází z podpory OpenCV samotné. Ta je crossplatformní a podporuje jak desktop (Windows, Linux, Android, MacOS), tak mobilní zařízení (Android, iOS). Ze systémů pozičních značek umí modul rozpoznat systémy Aruco, AprilTag a ARTag.

Jak již bylo uvedeno v sekci 2.7, nevýhodou modulu je jeho nesnadné přidání do OpenCV knihovny.

Kategorie	Body
Podpora jazyků	9
Intuitivnost	7
Dokumentace	8
Komunita	6
Podpora platforem	9
Systémy pozičních značek	6

Tabulka 3.2: Bodový zisk ArUco modulu pro OpenCV

3.2.3 ARToolKit

Jedná se o původní knihovnu pro detekci stejnojmenných pozičních značek. [22] ARToolkit byl původně open-source projekt, který vznikl roku 2001. Ve vlastnictví společnosti ARToolworks byl veden jako open-source do roku 2015, kdy byl prodán a stal se tak komerčním softwarem. Jednalo se o jedno z prvních SDK pro podporu rozšířené reality v mobilních aplikacích. Zájem o projekt po prodeji klesl a tak byla roku 2017 vytvořena open-source pokračující alternativa v podobě projektu ARToolKitX [23]. Jelikož je původní projekt neaktuální (poslední commit na stránkách repozitáře projektu je z roku 2015 před prodejem sw) a uživatelská komunita přešla k ARToolKitX, bude se analýza zabývat touto verzí knihovny.

Ta je napsána v C++ a poskytuje API pro C++, Javu, Javascript a C#. Dokumentace knihovny je zmatená a nedostatečná. Například ARToolkit pro C# je separován do samostatného projektu, ke kterému kromě popisu neexistuje žádná dokumentace. Lépe na tom nejsou ani verze pro C++ a Javu. Obsahují pouze definice tříd a většina je jich bez detailnějšího popisu.

Api pro detekci značek je podobné jako u ArUco knihovny. Vše co může být knihovnou detekováno, dědí od třídy ARTrackable. K detekci slouží třída ARTrackerSquare, která nabízí podobné API jako MarkerDetector z knihovny ArUco.

Uživatelská komunita knihovny ARToolKit byla v minuosti široká a největší popularitu měla v roce 2010. Od počátku projektu má kinohnva k datu 4. 1. 2020 přes 776 000 stažení na sourceforge. Přestože je toto číslo několikanásobně vyšší než u ArUco, je nutno podotknout, že od roku 2015 zájem o knihovnu značně klesl. Komutia navazujícího projektu ARToolKitX je výrazně menší.

Podpora platforem ARToolKit je široká a měla by se týkat desktopu (Windows, Linux, MacOS) i mobilních zařízení (iOS a OS Android). Ze 78% všech stažení knihovny se však jednalo o desktopovou verzi pro Windows a jen zhruba 1000 ze všech stažení knihovny bylo pro OS Android. Na základě toho

lze předpokládat omezenou uživatelskou základnu, která by používala ARToolkit v mobilních aplikacích. V dokumentaci existují manuály na kompilaci a linkování pro MacOS, Windows a Linux. Nikde však pro Android.

Z dokumentace lze vyvodit, že knihovna podporuje pouze stejnojmenné poziční značky. Knihovna nedisponuje dostatečnou podporou pro Android. V porovnání s oběma verzemi ArUco, které navíc podporují detekci značek systému ARToolKit, je méně vhodná pro použití do nové verze aplikace.

Kategorie	Body
Podpora jazyků	7
Intuitivnost	6
Dokumentace	2
Komunita	5
Podpora platforem	6
Systémy pozičních značek	5

Tabulka 3.3: Bodový zisk knihovny ARToolKit

3.2.4 Chilitags

„Chilitags je softwarová knihovna pro platformu C++, která slouží k detekci a identifikaci 2D referenčních značek.“ [24]. Chilitags byl původně soukromý projekt vyvíjený v Chililab, který byl později uvolněn širší veřejnosti. Jedná se o open-source projekt licencovaný pod LGPL3. Chilitags značky lze rozpoznat i na obrázích s nízkým rozlišením (jsou rozpoznány značky až 12 pixelů úzké). Jsou však velmi citlivé na okluze [24].

Chili tags obsahuje API pro C++, JavaScript a JNI pro Android (mapované podobně jako ArUco modul pro OpenCV).

Detekce značek je též podobná předchozím knihovnám. Detekce probíhá pomocí třídy Chilitags a proces detekce lze, stejně jako u předchozích knihoven, provést pomocí jediného řádku. Značku reprezentuje třída TagCornerMap, ze které je možno získat souřadnice rohů poziční značky a její id. Zvýraznění značek v obrázku, ve kterém byly detekovány není tak snadné jako u předchozích modulů. Funkcionalitu, kterou v ArUco knihovně poskytuje funkce drawMarkers je nutno naprogramovat ručně.

Dokumentace knihovny sestává z jejího popisu v GitHub repozitáři. Je stručná a některé odkazy, jako například odkaz na předgenerované sety detekovatelných značek, vedou na neexistující adresy. V dokumentaci lze najít pouze několuk vzorových příkladů, a tedy ji lze považovat za slabší.

Komunita knihovny je skromná. Na platformě GitHub se počet stažení repozitáře nedá přímo dohledat, avšak repozitář samotný disponuje k datu 16. 4. 2020 počtem pouze 100 hvězd, z čehož lze usoudit, že ani odhadovaný počet stažení nelze srovnávat s předchozími knihovnami.

Knihovna se profiluje jako multiplatformní pro desktop (Windows, Linux, Mac OS) i Android zařízení. Kvůli závislosti na OpenCV musí být knihovna správně nalinkována a zkompilována. Pokud není na stanici distribuce OpenCV knihovny, musí být nejprve zkompilována OpenCV knihovna. Cesta k tomuto sestavení je poskytnuta při kompilaci Chilitags knihovny.

Na hlavní stránce GitHub projektu, který slouží jako jediná dokumentace nelze dohledat podporu různých systémů pozičních značek. Na GitHubu existuje odkaz na vzorový vygenerovaný set detekovatelných značek, který však vede na prázdnou stránku. Knihovna umí detekovat pouze svůj stejnojmenný systém značek.

Kategorie	Body
Podpora jazyků	7
Intuitivnost	6
Dokumentace	3
Komunita	2
Podpora platforem	7
Systémy pozičních značek	5

Tabulka 3.4: Bodový zisk knihovny Chilitags

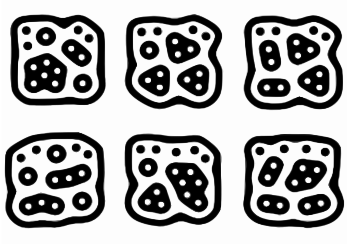
3.2.5 ReacTIVision

„ReacTIVision je open source, cross-platformní computer vision framework pro rychlou a robustní detekci pozičních značek připojených k fyzickým objektům, jakož i pro multi-touch finger tracking. Byl navržen hlavně jako sada nástrojů pro rychlý vývoj hmatatelných uživatelských rozhraní založených na tabulkách (TUI) a interaktivních povrchů s podporou multi-touch.“ [25]

Framework disponoval nemalou komunitou. Ke dni 15.4.2020 se počet stažení na sourceforge blíží 188000. Nejpopulárnější byl v roce 2008. Od roku 2011 zájem o framework strmě klesá. Méně rozsáhlé frameworky jako je tento, které slouužily pro počítačové vidění a rozšířenou realitu, byly pravděpodobně zastíněny rozsáhlejšími knihovnami a frameworky.

Knihovna poskytuje API pro C++, C#, Javu a PureData. Je schopná rozpoznat předdefinovanou sadu vlastních pozičních značek, určit jejich polohu, rotaci a identifikátor.

Podporuje pouze desktopové platformy Windows, Mac OS a Linux. Absencí podpory pro OS Android se tak framework stává nerelevantním pro aplikaci Myšák 2 a bude vyřazen z výsledného hodnocení.



Obrázek 3.2: Ukázka pozičních značek ReacTIVision

3.2.6 AR Core

Jedná se o open source framework společnosti Google, který poskytuje SDK pro vytváření mobilních aplikací, které používají rozšířenou realitu. Pomocí různých rozhraní umožňuje AR Core mobilnímu telefonu vnímat prostředí reálného světa kolem sebe, zpracovávat v něm různé informace a navíc je do něj informace doplňovat [26]. K určení polohy a orientace kamery není potřeba detekovat poziční značky, jako ve všech předchozích knihovnách. ARCore umí sám rozpoznat, metrické vlastnosti prostoru, který detekuje kamerou. Pomocí několika řádek kódu lze v prostoru rozpoznat rovnou plochu a umístit na ni objekty kliknutím na obrazovku.

Knihovna disponuje vynikající dokumentací, ve které lze nalézt vše od popisu rozhraní jednotlivých tříd, po množství návodů na konkrétní funkcionality, které knihovna nabízí.

V zadání této bakalářské práce je AR Core doporučenou technologií k podrobení analýzy pro detekci pozičních značek, jelikož se v současné době jedná o populární platformu pro tvorbu aplikací s rozšířenou realitou (ke které poziční značky často slouží). Framework je však mnohem dále, než veškeré dosavadní frameworky, které byly podrobeny této analýze. Čtvercové poziční značky framework nepotřebuje, jelikož dokáže z kamery sofistikovanou analýzou snímků vyčíst potřebné informace i bez nich. Po knihovně na detekci poziních značek vhodné pro tuto aplikaci je požadováno, aby byla schopna spolehlivě a rychle detekovat malé značky, z jejichž pozice bude schopna vyjmout oblast zájmu ze snímku a tu předat funkcím na ohodnocení řešení. Přestože poziční značky všech předchozích knihoven slouží především pro aplikace rozšířené reality tak, aby na ně mohly být zobrazovány nějaké objekty, v aplikaci Myšák 2 slouží zcela k jiným účelům. Hlavním cílem aplikace není doplňovat dodatečné informace do rozšířené reality, ale pouze analyzovat obraz.

Computer Vision vs Augmented Reality

V této části vymezím oblasti, kterými jsou rozšířená realita a počítačové vidění. Přestože mají mnoho společných principů, jejich hlavní zaměření je odlišné a lze říci, že rozšířená realita využívá počítačové vidění. Strojové vidění se zabývá analýzou, rozpoznáním obrazu. Snaží se chápat reálný svět z pohledu stroje. Rozšířená realita jde o něco dále, neboť do tohoto reálného světa navíc doplňuje počítačem vytvořené objekty. Rozšířená realita může být vnímána například skrze fotoaparát mobilního telefonu. Na tyto vědní disciplíny se zaměřují výše zmiňované frameworky. OpenCV je knihovna pro počítačové vidění a AR Core pro rozšířenou realitu. Přestože AR Core používá techniky počítačového vidění, jeho API je navrženo pro úpravu zobrazené scény reálného světa. V oblasti rozpoznání obrazu je high-level a není určeno pro vytváření algoritmů analýzy obrazu. K tomu naopak slouží OpenCV knihovna, která jde v této oblasti do hloubky. Knihovna AR Core nabízí kvalitní API pro aplikace s rozšířenou realitou, a přestože nebude použita v aplikaci Myšák 2 pro detekci pozičních značek, mohla by být v budoucnu použita pro oživení uživatelského rozhraní, a to především interakce s hlavní postavou myšáka. Ten by se například při otevření kamery zobrazil na pracovním listu a komunikoval s uživatelem.

Augmentace obrázků

Na první pohled může AR Core působit, že je schopný rozpoznat obrázky a tedy by pro něj neměl být problém detekovat poziční značky. Framework umí rozpoznat 2D obrázky, které mu jsou poskytnuty v referenčním setu. V praxi to znamená, že z databáze předdefinovaných podobných obrázků, je scéně rozpoznána odpovídající 2D plocha. Na ní pak lze například zobrazit a přehrát video. Uživatele frameworku by tak mohlo napadnout poskytnout frameworku sety pro poziční značky a detekovat je tak na scéně. Rozlišení obrázku však musí být alespoň 300x300 pixelů a obrázek musí pro správnou detekci zabírat alespoň 25% scény, což je v aplikaci Myšák 2 nežádoucí.

Podpora

Framework nabízí SDK pro Android, iOS a například i speciální rozšíření do herních enginů Unity a Unreal. Pro spuštění na Android zařízení je nutné disponovat alespoň operačním systémem Anroid 7.0 Nougat (SDK 24). Starší zařízení s SDK nižším aplikace s tímto frameworkem nespustí. Ani operační systém však nezaručí podporu, jelikož je technologie v současné době výpočetně náročná a klade velké nároky na fotoaparát. AR Core tedy podporuje jen určitá zařízení, která tyto nároky splňují. Seznam těchto zařízení se nachází na stránkách frameworku. Zařízení, která nejsou v seznamu uvedena, aplikace s tímto frameworkem nespustí. AR Core je mimo dosavadní scope projektu a nebude zařazen do hodnocení.

3.3 Analýza frameworků a enginů pro GUI

Aplikace Myšák 2 má ve své současné podobě komplikované řešení animací. Do nové verze aplikace je nutné zbavit se přehrávání animací v podobě videa. V souborech aplikace se vyskytuje i kreslený návrh myšáka, který byl původně animován sekvencí snímků a později nahrazen právě videem. Řešení pomocí sekvence snímků s technologiemi z Android SDK bylo komplikované, jelikož nejsou uzpůsobeny pro herní vývoj, ale spíše pro jednoduché efekty uživatelského rozhraní. Bude proto vhodné použít herní engine, či framework, které jsou pro to přizpůsobeny. Nabízí se dvě možnosti:

1. Zahození dosavadních animací, použití původního 2D kresleného návrhu a všechny animace vytvořit v nové technologii od začátku.

2. Vytvoření 3D scény, na kterou bude nahrán model myšáka, ze kterého byly vytvářeny videa animací. Ten by měl mít tyto animace uložené ve svém modelu a jejich spouštění by mělo probíhat v kódu.

Tvorba animací není součástí této bakalářské práce. Dosavadní animace v aplikaci jsou velmi užitečné i pro novou verzi. Trpěly pouze technickým řešením jejich zobrazení, a proto bude do nové verze aplikace zvoleno druhé řešení, kdy budou ponechány. Uživatelské rozhraní bude navrženo tak, aby bylo možné v navazujících pracích snadno vyměnit 3D scénu za 2D, kdyby bylo rozhodnuto původní uživatelské rozhraní zahodit.

Pro novou metodu zobrazování animací je nutné zvolit vhodnou technologii, která poskytne kvalitní API pro 3D, potažmo 2D grafiku. V potaz musí být brána i podpora pro OS Android a knihovnu OpenCV. Nutná je i podpora pro čtení dat z kamery zařízení. Pro novou verzi aplikace se nyní nabízí 2 možnosti. První je ponechat současný stav aplikace a provést refaktoring s použitím technologií, které poskytuje Android SDK, či technologií, které lze s Android SDK snadno integrovat. Druhá možnost je s ohledem na zvolený engine či framework vytvořit aplikaci od začátku. Výsledná volba vyplyne z této sekce analýzy.

Lze dohledat mnoho článků, které se zabývají porovnáním frameworků a herních enginů [27]. Aplikace Myšák 2 má však nestandardní požadavky a nevyžaduje žádné pokročilejší funkce herního vývoje jako například umělou inteligenci nebo pokročilou herní fyziku. Myšák se bude pouze zobrazovat, pohybovat v jednoduchém herním prostředí a komunikovat s uživatelem.

Kromě společných vlastností ze sekce 3 budou sledovány i následující vlastnosti:

Integrace s OpenCV Musí být možné integrovat OpenCV knihovnu s konkrétním frameworkem.

Komunikace s kamerou Framework musí umožňovat komunikaci s kamerou zařízení. Například otevřít ji v nové aktivitě a uložit snímek z ní do souboru.

Integrace s SQLite databází framework musí umožňovat připojení SQLite databáze, do které budou ukládána odevzdaná uživatelská řešení.

Podpora 3D framework musí umožňovat zobrazit 3D objekty, manipulovat s nimi a přehrávat jejich animace.

Podpora jazyků v této sekci analýzy nebude striktně zaměřena na Kotlin, pokud vybraný framework či engine poskytne dostatečnou podporu všech potřebných technologií. Bude-li vybrána technologie, která pro vykreslování používá jiný jazyk, bude Kotlin použit pouze pro funkce platformě závislé na OS Android. Kotlin zůstává preferovaným jazykem díky svým benefitům definovaným v sekci 3.1.

3.3.1 Android SDK s OpenGL ES

OpenGL je cross-platformní knihovna pro vývoj 2D a 3D grafických aplikací [28]. Poskytuje low level API s nízkou mírou abstrakce a hodně práce tak zbývá na uživatele. Při použití této knihovny je nutno komunikovat přímo s GPU. Mnoho herních enginů z této knihovny vychází nebo ji používá na pozadí a poskytuje nad ní určitou míru abstrakce.

Tato technologie je součástí SDK pro platformu Android [29] v podobě knihovny OpenGL ES [30]. Jedná se o odlehčenou verzi OpenGL knihovny pro vestavěná zařízení. Hlavními komponentami rozhraní knihovny je GLSurfaceView, do které jsou vykreslovány objekty a Renderer, který provádí vykreslování. Knihovna poskytuje api, které je možné používat v Javě či Kotlinu i Native Development Kit pro C++. Je tedy možné nakombinovat uživatelské rozhraní napsané v Kotlinu s 3D scénou, která je vykreslována pomocí C++.

OpenGL ES disponuje kvalitní dokumentací s řadou návodů, které popisují zakreslení jednoduchých objektů do GLSurfaceView. Avšak i vykreslení triviálních objektů je značně komplikované. Pro složitější operace jako je vykreslování a animace 3D objektů je potřeba knihovnu OpenGL ovládat alespoň na pokročilé úrovni.

Při použití OpenGL ES by byl dopad na aplikaci Myšák 2 minimální se zachovanou dosavadní kompatibilitou kamery, SQLite databáze a OpenCV knihovny, neboť bude nadále využito pouze Android SDK. Knihovna je populární, avšak místo knihovny samotné používá většina vývojářů enginy a frameworky, které jsou na ní vybudovány. Knihovna je pro aplikaci Myšák 2 nevhodná především kvůli své složitosti.

Kategorie	Body
Podpora jazyků	5
Intuitivnost	2
Dokumentace	8
Komunita	?
Podpora platforem	5
Podpora kamery	9
Integrace OpenCV	9
Integrace s SQLite	8
Podpora 3D	5

Tabulka 3.5: Bodový zisk OpenGL ES

3.3.2 Cocos 2d-x

Cocos 2d-x představuje open source framework učený pro herní vývoj. Je crossplatformní a umožňuje provádět deploy na iOS, Windows, Android a Mac s hlavním zaměřením na mobilní platformy. Je založen na OpenGL 2.0, OpenGL ES 2.0 a napsán v C++. Přestože název působí, že se jedná o engine zaměřený na 2D grafiku, podporuje i 3D [31].

Cocos 2d-x je jednou z verzí rodiny frameworků Cocos 2d [32]. Existují i další verze, které se mohou lišit v dostupných funkcionalitách. Analýza se bude zabývat původní nejrozšířenější verzí frameworku, tedy Cocos 2d-x.

Pro vytváření herního obsahu pro framework existuje oficiální editor CocosCreator. *„Jedná se o script oriented, entity-compontent, data driven nástroj pro vývoj her zaměřený na tvorbu obsahu. Jde tedy o high-level vývojové prostředí, které odprostí jeho uživatele od nutnosti psaní velkého množství kódu. Přichází se snadným pracovním postupem výroby obsahu a výkonnou sadou vývojářských nástrojů pro logiku hry a vysoce výkonnou tvorbu her.“* [33]

Knihovna má kvalitní dokumentaci i širokou komunitu. Ke dni 21. 4. 2020 disponuje GitHub repozitář knihovny počtem 14 300 hvězd. Kromě existence všech předchozích verzí lze pro Cocos 2d-x dohledat mnoho dalších nástrojů. Některé z nich jsou zastaralé a není jasné k čemu přesně který editor slouží, či který starší editor nahrazuje. Vymezím proto pojmy, týkající se frameworku.

Cocos 2d-x Je samotný frameworku, na kterém je postavena konkrétní hra. Je nutné jej stáhnout sestavit (vytvořit build). Z tohoto sestavení vytvořit projekt a ten nalinkovat do libovolného vývojového prostředí (dále IDE), ve kterém bude hra vyvíjena.

Cocos Creator Je editor, ve kterém lze vytvořit kompletní hra založená na javascriptové verzi enginu Cocos 2d. Lze v něm vytvořit celou hru od animací, přes herní fyziku, umělou inteligenci a uživatelské rozhraní po kompletní herní prostředí. Nevýhodou Cocos Creatoru je absence podpory pro platformu Linux. Toto vývojové prostředí běží pouze na Windows a Mac OS. Editor lze i použít jen k vytváření herního obsahu, který lze vyexportovat a použít v libovolném IDE s Cocos 2d-x.

Cocos Studio Nástroj pro vytváření obsahu do hry. Umožňoval podobné funkce jako Cocos Creator. Nyní je označen jako zastaralý a byl nahrazen Cocos Creatorem.

Cocos IDE Bylo IDE určené pro platformu Cocos 2d. Podporovalo programovací jazyky Lua a Javascript. Bylo označeno jako zastaralé a místo něj je možné psát kód v libovolném IDE (například v Android Studiu).

Framework podporuje SQLite databáze a nabízí pro ně snadno ovladatelné api. Na oficiálním fóru frameworku existuje několik návodů pro připojení této databáze.

Integrace s kamerou je v Cocos 2d-x komplikovaná. Neexistuje žádné rozhraní ani návod, který by popisoval, jak otevřít kameru přímo z jádra enginu. Lze předpokládat, že taková funkce je mimo rozsah projektu Cocos 2d a tedy je na samotném vývojáři, aby rozhraní pro komunikaci s kamerou napsal sám. Je tedy potřeba z C++ kódu tříd definovaných v balíčku `Classes` zavolat Java kód aktivity, ve které je aplikace spuštěna. V této aktivitě následně definovat metodu, která otevře kameru v nové aktivitě. Nesprávné nativní volání pro spuštění otevření nové aktivity však může vést k narušení životní cyklus aktivity, ve které běží herní smyčka. To může vést k pádu celé aplikace. Muselo by být zajištěno nenarušení životního cyklu aktivit. Podporu kamery zařízení Android frameworkem Cocos lze považovat za koplikovanou, bez dostatečné podpory a dokumentace.

Integrace OpenCV knihovny s frameworkem není nijak zdokumentována. Provedení integrace je složité, ale možné, což bylo ověřeno v rámci této bakalářské práce. Je nutné nalinkovat zkompilované distribuce do projektu a nastavit cesty k hlavičkovým souborům knihovny.

3D objekty lze ve frameworku reprezentovat třídou Sprite3D , která podporuje načítání tří formátů souborů:

obj standardní 3D formát, který podporuje řada softwarů pro 3D modelování. Nepodporuje animace.

c3t Neboli Cocos 3D text je textový soubor reprezentující 3D objekt ve formátu JSON. Formát c3t lze získat z formátu fbx pomocí utility, která je dostupná na stránkách frameworku. Může být snadno čten a lze v něm provádět úpravy. Jeho načítání a zpracovávání je však pro engine náročnější, a proto se doporučuje nepoužívat tyto objekty v relase verzi aplikace. Formát je díky čitelnosti vhodný k debugování.

c3d Neboli Cocos 3D binary je binární soubor, který lze též získat konvertováním z fbx pomocí fbx-conv. Oproti c3t nemůže být čten a upravován, ale má menší velikost a jeho načtení do aplikace je rychlejší. Při vývoji je tento typ souboru nepraktický, ale vhodný do release verze aplikace.

Cocos 2d-x je kvalitní a rozšířený framework pro vývoj 2D i 3D aplikací, avšak jeho kompatibilita s kamerou zařízení není dostatečně prozkoumána a zdokumentována, což bude klíčové při závěrečném porovnání s ostatními frameworky.

Kategorie	Body
Podpora jazyků	8
Intuitivnost	6
Dokumentace	8
Komunita	8
Podpora platforem	8
Podpora kamery	4
Integrace OpenCV	7
Integrace s SQLite	7
Podpora 3D	6

Tabulka 3.6: Bodový zisk Cocos 2d-x

3.3.3 LWJGL

LWJGL je zkratka od Lightweight Java Game Library. Je to knihovna, která poskytuje přístup k nativním API. Příklad takového API může být OpenGL, OpenAL či OpenCL [34]. Tato knihovna je tedy wrapper, který umožňuje komunikovat s nativními knihovnami. Nejedná se o framewrok, ani o engine. Práce s knihovnou je low-level a spíše než k hernímu vývoji samotnému slouží k poskytnutí API jiným enginům a frameworkům.

3.3.4 jMonkey

Jedná se o open-source herní engine pro platformu Java zaměřený na 3D grafiku. Podporuje platformy Android, iOS a Desktop (všechny platformy, s nainstalovaným JVM). Engine je založený na LWJGL (viz 3.3.3) a na pozadí používá OpenGL [35]. Pro herní vývoj poskytuje kromě enginu samotného i SDK s vlastním vývojovým prostředím, které vychází z NetBeans. Nabízí tak 2 možností vývoje aplikací:

1. Použít SDK s mnoho předdefinovanými funkcemi a knihovnami usnadňujícími herní vývoj. SDK usnadňuje sdílení herních komponent napříč projekty. Umožňuje například sdílet assety mezi projekty. SDK umí vygenerovat projekt s potřebnými třídami pro deploy na různé platformy.

2. Použít pouze herní engine a vytvořit v libovolném IDE vlastní projekt s jMonkey.

Obě možnosti naráží na své nedostatky. Pokud se člověk pomocí první možnosti pokusí založit projekt pro Android pomocí postupu z oficiální dokumentace, vygeneruje se adesářová struktura demo aplikace, která však obsahuje neplatné importy. Není správně nalinkována závislost na Android SDK a importy navíc odkazují na již nepodporované verze knihoven. IDE postavené na NetBeans navíc působí při vývoji pro Android jako přítěž, neboť slabě podporuje programovací jazyk Kotlin. Pro přidání podpory pro Kotlin je potřeba použít plugin (viz [36]), který byl naposled aktualizován roku 2017.

Druhá možnost je volnější, avšak k vytvoření projektu pro Android Studio chybí jakákoliv dokumentace a podpora. V oficiální dokumentaci lze nalézt pouze výčet závislostí pro Gradle, dále je uživatel enginu odkázán sám na sebe. Dokumentace samotná má své kladné i záporné stránky. Oficiální wiki dostupná ze stránek frameworku je nepřehledná a stručná. Slouží pouze jako rozcestrník, ve kterém se nachází odkazy na JavaDoc a série tutoriálů.

Z hlavní stránky enginu vede odkaz na JavaDoc enginu. U mnoha tříd neobsahuje ani jednu větu k jejich popisu. Místo dokumentace slouží spíše jako výčet tříd enginu, ze kterého lze těžko porozumět jejich funkci. Oficiální dokumentace a série video tutoriálů zabírají širokou škálu oblastí, které začínající uživatel enginu ocení a ve značné míře vynahradí nedostatky JavaDocs.

Oproti ostatním enginům a frameworkům má tento slabší uživatelskou základnu. GitHub repozitář disponuje ke dni 22. 4. 2020 počtem 2300 hvězd, což je několikanásobně méně než u ostatních frameworků. To se podepisuje i na podpoře. Engine má vlastní fórum pro uživatelskou komunitu. Při snaze dohledat články týkající se například podpory čtení dat z kamery či načítání 3D modelů vedou odkazy s řešením na neexistující stránky.

Podpora pro kameru je díky jazyku Java snadno dosažitelná. Lze předpokládat, že po opravení závislostí ve vytvořeném demo projektu by bylo možné využívat všechny prvky Android SDK, a tedy i čtení dat z kamery pomocí

otevření nové aktivity. Jelikož je vývojové prostředí pro jMonkey založené na NetBeans a podporuje stejné funkce pro import knihoven, integrace OpenCV do projektu je tak stejná jako do IDE NetBeans [37].

Výhodou jMonkey oproti Cocos 2d-x a LibGDX je přímá podpora načítání fbx souborů. Modely jsou automaticky zkonvertovány do vlastního formátu `j3o`, který engine používá. K přehrávání animací slouží třída AnimController.

Engine nenabízí žádné API pro komunikaci s databází. Komunikace by musela být nakonfigurována odděleně pro každou platformu. Každá platforma by vlastním způsobem spravovala a připojovala databázi. Taková konfigurace není nemožná, avšak je složitější, než například u Cocos 2d-x.

Engine je zaměřený na 3D. Jeho podpora pro 2D grafiku je slabá, jelikož pro ní engine není určen.

Kategorie	Body
Podpora jazyků	5
Intuitivnost	5
Dokumentace	4
Komunita	5
Podpora platforem	7
Podpora kamery	6
Integrace OpenCV	6
Integrace s SQLite	5
Podpora 3D	7

Tabulka 3.7: Bodový zisk jMonkey

Přes snazší integraci kamery a OpenCV knihovny je engine spíše nevhodný pro použití do projektu kvůli slabé podpoře 2D grafiky, horší dokumentaci, menší uživatelské základně, slabému IDE a neaktuální podpoře platformy Android.

3.3.5 Unity

Unity je closed source herní engine napsaný v C++. Nabízí API pro C# a na pozadí může používat OpenGL nebo DirectX. Je vyvíjen společností Unity Technologies a nabízí kromě svého jádra mnoho software a utilit pro vytváření herního obsahu. Podporuje 2D i 3D grafiku [38].

Engine je široce podporován, což dokazují statistiky z oficiálních stránek. Disponuje kvalitní dokumentací a velkou uživatelskou komunitou. Na fóru lze nalézet řadu návodů jak číst data z kamery zařízení Android či iOS. Pro integraci OpenCV do Unity projektu existuje též manuál [39]. Engine podporuje i připojení SQLite databáze [40].

Engine nabízí vlastní editor, který umožňuje přímý import fbx modelů.

Unity má mnoho verzí pro soukromé i komerční použití. Pro firmy jsou k dispozici placené business verze Plus, Pro a Enterprise. Pro soukromé použití jsou k dispozici dvě free verze Student a Personal, které však mají podmínky, které jejich uživatel musí splnit. Pokud dojde k porušení těchto podmínek, ztrácí nárok na použití free verze. Pro verzi Student musí být uživatel zapsán v akreditovaném studijním programu a splnit minimální věkové podmínky pro sběr osobních údajů. Hranice je nastavena na 13 let pro USA a 16 let pro Evropu. Student musí být navíc připojen do GitHub Student Pack. Současnou nevýhodou studentského účtu je jeho registrace. Unity má měsíční lhůtu na ověření nároků na licenci, což v době psaní této bakalářské práce znemožňuje její použití. Verze Personal je též zdarma, pokud uživatel nepřesáhne svým ziskem z aplikace 100000 dolarů za rok. Poté na free licenci ztrácí nárok.

Unity je rozsáhlý engine, který vyžaduje vyvinutí počátečního úsilí, kvůli orientaci v dostupných editorech a nástrojích. Projekty jsou vyvíjeny v nástroji Unity Editor, který pro pouhé pochopení základů vyžaduje nemalé množství času k ovládnutí technologie.

Kategorie	Body
Podpora jazyků	5
Intuitivnost	4
Dokumentace	9
Komunita	9
Podpora platforem	9
Podpora kamery	8
Integrace OpenCV	7
Integrace s SQLite	8
Podpora 3D	9

Tabulka 3.8: Bodový zisk Unity

3.3.6 LibGDX

Libgdx je open-source framework určený pro tvorbu her v Javě. Podporuje 2D i 3D grafiku, ale oproti předchozím technologiím postrádá vlastní vývojové prostředí. Neumožňuje tak například zobrazit náhled 3D scény, kterou by bylo možné upravovat pomocí drag&drop jako v Unity. Framework má díky programovacímu jazyku Java mnoho společného s enginem jMonkey. Budou proto v této části analýzy často porovnávány. LibGDX je oproti jMonkey univerzálnější ve volbě vývojového prostředí. Vytvoření projektu je též intuitivnější než v jMonkey. Stejně jako jMonkey používá pro komunikaci s OpenGL knihovnu LWJGL 3.3.3.

LibGDX je cross-platformní a podporuje deploy na Android, iOS, Desktop, BlackBerry. Podporuje i spouštění aplikace v prohlížeč pomocíi WebGL. Na pozadí používá knihovny OpenGL ES a LWJGL. Knihovna je převážně napsána v Javě, avšak části kritické na výkon jsou v C++ [41].

Framework má poměrně širokou komunitu a s počtem 16 800 hvězd v GitHub repozitáři předčí jMonkey i Cocos 2d-x. Oficiální dokumentace popisuje pouze základní prvky frameworku a nastavení vývojového prostředí. Komunita udržuje i vlastní stránky s návody a ukázky kódu z her, které mohou být vhodné pro začátečníky [42]. Nejlepší dokumentací pro LibGDX je wiki v repozitáři knihovny. V ní lze nalézt vše potřebné pro vývoj aplikace.

Stejně jako Cocos 2d-x podporuje framework herní fyziku Box2D [43]. Pro 3D grafiku je podporován framework herní fyziky Bullet Physics. V oficiálním repozitáři se je mnoho dalších utilit a nástrojů pro usnadnění vývoje.

Založení projektu je intuitivnější, než v jMonkey enginu. LibGDX poskytuje na oficiálních stránkách nástroj, který vygeneruje multiplatformní projekt postavený na systému gradle. Tento nástroj nabízí i možnost vygenerovat projekt v Kotlinu, který je kromě ztráty podpory pro platformu WebGL plně podporován. Podpora pro deploy do prohlížeče je však pro projekt Myšák 2 irelevantní. Vygenerovaný projekt s podporou Kotlinu je rozdělen na 4 moduly:

core Obsahuje veškerou logiku aplikace společnou pro všechny platformy. Na modul core mají závislosti všechny ostatní moduly. V této části se nachází hlavní třída, která spouští jádro aplikace. Její instanci ostatní moduly používají ke spuštění aplikace. V této části by se měl nacházet veškerý kód, který nezávisí na použité platformě.

android Modul, který obsahuje logiku pro spuštění aplikace v OS Android. Obsahuje i složku assets, která je společná pro všechny projekt. Moduly desktop a ios mají v této složce nastaven svůj pracovní adresář. Přestože ve složce assets je uložen herní obsah, který je platformně nezávislý, je tato složka v modulu android a ne core. Toto řešení bylo zvoleno, kvůli snazší lokaci herního obsahu pro sestavování pro OS Android.

desktop Modul, který obsahuje třídu pro spuštění aplikace na desktop zařízeních. Java běží nad JVM, tudíž je možné provádět deploy na všechny desktopové platformy (Windows, Mac OS i Linux).

ios Modul, který obsahuje logiku pro spuštění aplikace v iOS pomocí RoboVM. RoboVM je kompilátor, který překládá Java bytekód tak, aby mohl být interpretován na zařízeních iOS. Vývoj RoboVM byl však pozastaven a již 4 roky se neaktualizuje. V současnosti je možné provádět deploy pomocí RoboVM s nutností očekávat možné potíže s kompatibilitou, nebo použít nové řešení pomocí externích knihoven [44].

LibGDX má podobný systém pro práci s 3D objekty jako Cocos 2d-x. Objekt je reprezentován pomocí třídy Model. Model může mít několik různých instancí, které se mohou od původního lišit například jinou texturou či deformací. Pro modely jsou podporávny 3 typů formátů souborů:

obj Standardní 3D formát, který nepodporuje animace

g3dj Je textový soubor reprezentující 3D objekt ve formátu JSON. Formát lze získat z `fbx` pomocí utilit frameworku. Může být snadno čten a lze v něm provádět úpravy. Stejně jako v Cocos 2d-x se doporučuje nepoužívat textový formát modelu v relase verzi aplikace, ale pouze k debugování.

g3d Je binární soubor, který lze též získat konverzí z fbxNemůže být čten ani upravován, ale je snáze zpracovatelný a použíívá se v release verzi aplikace.

Ke správě animací slouží třída AnimationController a její použití je podobné stejnojmené třídě v jMonkey enginu.

LibGDX obsahuje v dokumentaci návod na komunikaci s platformě specifickou kamerou. Lze předpokládat její plnou podporu. Oproti jMonkey nabízí lepší podporu pro vývoj v libovolném IDE, tedy i Android Studio. OpenCV knihovna tak může být do nového projektu nalinkována stejným způsobem jako do stávajícího projektu aplikace Myšák 2. Podobně jako u jMonkey, neposkytuje LibGDX žádné API pro připojení a operace s databází. Připojení pro SQLite databázi by muselo být nakonfigurováno podle platformy, na které je aplikace spuštěna.

Kategorie	Body
Podpora jazyků	7
Intuitivnost	6
Dokumentace	9
Komunita	9
Podpora platforem	9
Podpora kamery	8
Integrace OpenCV	8
Integrace s SQLite	5
Podpora 3D	7

Tabulka 3.9: Bodový zisk LibGDX

Framework získává mnoho bodů za podporu jazyků díky preferovanému Kotlinu.

3.4 Analýza ORM frameworků

Jedním z cílů nové verze aplikace je vybrat nový ORM framework pro SQLite databázi. Současný framework použitý v aplikaci Myšák 2 má několik nedostatků, kvůli kterým je vhodné udělat průzkum konkurenčních technologií a zvážit jeho nahrazení.Při analýze budou kromě společných vlastností sledována i následující:

Podpora vazeb Důraz bude kladen na vazby One to Many a Many to Many.

Scope projektu nevyžaduje robustní řešení pro operaci s databází. V databázi se z převážné většiny ukládají pouze cesty k souborům, či proměnné primitivních datových typů. ORM má za úkol usnadnit následující operace:

1. Načíst zadání úkolu
2. Uložit vyhodnocené řešení

Operace nad databází jsou omezené a nenáročné. Úkoly se načítají po jednom a při jejich načtení bude s největší pravděpodobností uplatněn eager loading. Nebylo by proto vhodné vyměnit SugarORM za příliš robustní a složitý framework, neboť aplikace má na databázi nízké nároky. Při analýze velikosti uživatelské komunity frameworků bude jako metrika sloužit statistika používání knihoven zařízeními na trhu z `https://tinyurl.com/ORMStats`.

3.4.1 Sugar ORM

Sugar ORM je framework určený výhradně pro zařízení s operačním systémem Android [7]. Jedná se o framework použitý v původní verzi aplikace. Pro správnou konfiguraci frameworku po spuštění aplikace je nutné mu předat název databáze (očekává se její umístění ve složce assets/databases), a adresářovou strukturu, ve které jsou uloženy entitní třídy. Každá entitní třída získá základní předdefinované CRUD operace.

Problematická je jeho podpora vazeb. Vazby `One to Many` jsou řešeny nestandardně a podpora vazeb `Many to Many` zcela chybí. Je tedy na uživateli, aby si mapování těchto vazeb naprogramoval sám. Framework dobře podporuje Kotlin a jeho datové třídy. Mezi jeho slabiny patří oficiální dokumentace (viz [7]). Obtížně se v ní zjišťuje, jaké vazby framework podporuje a jak je konfigurovat.

I přes své nedostatky a fakt, že se jedná o velmi tenkou knihovnu, je poměrně rozšířen. V GitHub repozitáři projektu má 2 600 hvězd a podle statistik je framework použit v 0,27 % všech aplikací na Google Play [45]. Umožňuje i vygenerovat databázi z entitních tříd. Již zmíněné relace `Many to Many` však vytváří jako textová pole, nikoliv jako cizí klíče. Aplikaci, která používá Sugar ORM je tedy vhodnější vydávat s předgenerovanou databází a transakce těchto vazeb naprogramovat pomocí SQL.

Kategorie	Body
Vazby	5
Podpora jazyků	7
Podpora platforem	5
Intuitivnost	9
Dokumentace	2
Komunita	6

Tabulka 3.10: Bodový zisk Sugar ORM

3.4.2 GreenDao

Jedná se o open source framework určený pro OS Android. Hlavní výhodou je jeho rychlost a malá velikost (méně než 100KB) [46]. Malá velikost je pozitivní aspekt, neboť aplikace Myšák 2 díky staticky linkované knihovně OpenCV zabírá na zařízení mnoho místa.

GreenDao není tak intuitivní jako předchozí framework. Po uživateli požaduje vytvořit entitní třídy, jejich DAO a třídu Helper pro připojení a verzování databáze. Tato třída navíc obsahuje reference na všechny DAO entitních tříd. Tyto třídy jsou abstraktní a jejich implementace je generována po sestavení aplikace.

Framework má dobrou dokumentaci a početnou komunitu. Podle statistik je použit v 0,74 % všech aplikací na Google Play [45]. Na stránkách GitHub repozitáře disponuje počten 12 000 hvězd. Disponuje však i několika nevýhodami. První z nich je nedostatečná podpora Kotlinu. Entitní třídy, DAO i Helper musí být napsány v Javě, neboť generovaná implementace je v jazyce Java. Z tříd napsaných v Kotlinu framework implementace těcht tříd vygenerovat neumí. Framework vynucuje do aplikace psané v Kotlinu přidávat mnoho tříd Javy. Přestože jsou spolu jazyky kompatibilní, z důvodu přehlednosti by bylo vhodné vyhnout se této kombinaci.

Kategorie	Body
Vazby	7
Podpora jazyků	3
Podpora platforem	5
Intuitivnost	6
Dokumentace	8
Komunita	8

Tabulka 3.11: Bodový zisk GreenDao

3.4.3 Room

Stejně jako předchozí frameworky poskytuje abstrakční vrstvu nad SQLite databázemi [47]. Knihovna je určena hlavně pro vytvoření robustnějšího přístupu do databáze při zachování všech výhod, které SQLite nabízí. Knihovna je součástí balíčku Android Jetpack. To je sada knihoven, nástrojů a návodů, které mají developerům usnadnit vývoj moderních aplikací pro OS Android. Z toho plyne, že se jedná o framework čistě pro Android zařízení. Jako nástroj z oficiální stránky pro vývojáře Android aplikací poskytuje vynikající dokumentaci, ve které je popsán každý aspekt knihovny, mnohdy i s vysvětlením voleb, které vývojáři provedli při implementaci jednotlivých komponent. Knihovna je použita v 62 % všech aplikací na Google Play [45], z čehož plyne velká uživatelská základna.

Knihovna vyžaduje stejně jako předchozí frameworky definice entitních tříd a DAO, pomocí kterých se načítají a ukládají objekty do databáze. Podporuje jazyky Java i Kotlin.

Knihovna má několik omezení, která by při jejím použití v aplikaci Myšák 2 značně zesložitila vývoj. Hlavním omezením jsou vazby, jelikož Room zakazuje přímé objektové reference mezi entitními třídami. Důvody k tomu, jsou popsány v dokumentaci [48]. Toto omezení bylo zvoleno proto, aby nedocházelo k zmrazení hlavního vlákna, na kterém běží UI, při velkých a častých transakcích mezi zařízením a databází. Toto chování může nastat především u aplikací, které komunikují se vzdálenou databází. Pro vytvoření vazby mezi entitami musí být vytvořena další entitní třída, která definuje sloučenou tabulku pro tuto vazbu. Pokud by entita měla vazeb více, musela by být postupně pro každou z nich vymodelována definující entitní třída. Pokud by navíc uživatel chtěl implementovat eager loading do určité hloubky, počet nutných entitiních tříd by vzrostl násobně. To způsobuje nepřehlednost zvláště u entit s několika vazbami na ostatní tabulky. Přestože databázový model bude upraven, stále bude stěžejní tabulkou v databázi zadání úkolu, které bude mít mnoho vazeb na související tabulky, a které budou načítány metodou eager loading. V aplikaci by s knihovnou Room vznikla spousta zbytečných entitních tříd definující dílčí vazby, a tedy spousta generického kódu.

Kategorie	Body
Vazby	2
Podpora jazyků	6
Podpora platforem	5
Intuitivnost	4
Dokumentace	9
Komunita	9

Tabulka 3.12: Bodový zisk Room

3.4.4 OrmLite

Předchozí frameworky se týkají výhradně SQLite databází pro OS Android. OrmLite je framework se širokým záběrem na podporované databáze. Kromě Android databází podporuje i JDBC připojení pro Postgres, MySQL, Derby, H2, HSQLDB a mnoho dalších. Může být použit kdekoliv nad JVM, tedy i v programovacím jazyce Kotlin. Oproti jiným robustnějším frameworkům používajícím JDBC, jako je Hibernate nebo iBatis, je mnohem lehčí, jednoušší a obsahuje méně závislostí [49]. OrmLite pro svou práci využívá entitní třídy a DAO. Podporuje všechny typy požadovaných vazeb, přičemž Many to Many vazby musí být dekomponovány a operace nad nimi naprogramovány pomocí třídy `QueryBuilder`, který framework poskytuje. Dokumentace je podrobná s odkazy na vzorové příklady. Jedním z nich je například ukázková konfigurace `Many To Many` vazeb. Jedná se též o populární framework. Podle statistik je použit v 1,24 % všech aplikací na Google Play [45].

Kategorie	Body
Vazby	6
Podpora jazyků	8
Podpora platforem	9
Intuitivnost	6
Dokumentace	9
Komunita	4

Tabulka 3.13: Bodový zisk

Framework disponuje oproti ostatním širokou podporou databází a je vhodným kandidátem i pro multiplatformní projekty.

3.4.5 SQLDelight

V porovnání s ostatními frameworky se jedná se o opačné pojetí objektově-relačního mapování. Zatímco předchozí frameworky pomocí kódu objektově orientovaného jazyka generují SQL kód, framework SQLDelight používá opačnou strategii. Uživatel nejprve vytvoří create scripty pro své tabulky, definuje pomocí dotazů metody, které chce nad databází provádět. Framework pak z těchto souborů vytvoří databázové schéma a vygeneruje API pro operace nad databází. Vygenerované metody jsou typově zabezpečené. Při kompilaci se také kontroluje databázové schéma, dotazy a migrace
databáze [50]. SQLDelight navíc nabízí plugin, který do IDE přidá další funkce, jako je automatické doplňování SQL dotazů.

Oproti předchozím frameworkům je tento platformě nazávislý. Má několik verzí driverů pro OS Android, iOS, Windows a JVM. Jedná se o mladší technologii, čemuž odpovídá i menší uživatelská komunita. Dokumentace k frameworku je dohledatelná pouze přes odkaz z hlavního GitHub repozitáře. Vztahy mezi entitami a operace mezi nimi musí uživatel frameworku definovat sám. U složitějších vazeb musí uživatel definovat i vlastní adaptér, kterým se překládají sloupce tabulek na atributy tříd. Intuitivnost knihovny je nízká a navíc je vyžadována pokročilá znalost jazyka SQL.

Kategorie	Body
Vazby	5
Podpora jazyků	8
Podpora platforem	9
Intuitivnost	3
Dokumentace	7
Komunita	6

Tabulka 3.14: Bodový zisk SQLDelight

3.4.6 Realm

Nejedná se přímo o framework pro objektově-relační mapování, ale o zcela odlišný databázový systém pro mobilní aplikace. Realm je objektově orientovaná databáze, která používá ve svém jádře C++ a slouží jako alternativa k SQLite. Důraz je kladen hlavně na výkon. Databáze by měla podle oficiálních stránek produktu být až 10x rychlejší než SQLite [51]. Výhodou Realmu je jeho vlastní způsob ukládání dat, díky kterému může být snadno používan nezávisle na různých platformách. Má dobrou dokumentaci a podporuje mnoho programovacích jazyků – kromě Javy a Kotlinu také Swift, JavaScript, Objective C a další. Díky této široké podpoře se jedná se o poměrně rozšířenou technologii. Podle statistik je použit v 2,1 % všech aplikací na Google Play [45]. Vazby se definují podobně jako u ORM frameworků pro SQLite.

Realm má svá omezení, ze kterých nejvýraznější je absence podpory autoinkrementovaných ID. Přestože se prezentuje jako platformě nezávislá technologie, je určen primárně pro mobilní aplikace a desktopové architektury jej nepodporují. Seznam všech omezení je dohledatelný na oficiálních stránkách.

Databáze v aplikaci je primitivní a z převážné většiny pouze ukládá cesty k souborům. Výměna relační databáze za objektovou by byla pro aplikaci méně vhodná.

Kategorie	Body
Vazby	7
Podpora jazyků	8
Podpora platforem	5
Intuitivnost	4
Dokumentace	8
Komunita	8

Tabulka 3.15: Bodový zisk Realm

3.5 Výsledná volba technologií

Nejprve je nutné zvolit framework, či engine pro GUI, neboť jeho volba může značně ovlivnit výběr dalších technologií. Sledovaným vlastnostem budou v každé z následujících sekcí přiřazeny váhové koeficienty pomocí kterých bude spočten vážený bodový průměr pro každou technologii.

3.5.1 GUI

Klíčové oblasti při výběru technologie jsou podpora kamery a integrace OpenCV. Bez nich nemá technologie pro projekt žádný smysl, a budou proto hodnoceny váhovým koeficient 4. Další klíčovou oblastí je podpora platforem, ze kterých je nejdůležitější Android, na který aplikace cílí. Deploy na zařízení s OS Android podporují všechny technologie. Získané hodnocení v této sekci tak závisí pouze na podpoře dalších platforem, a proto získává váhový koeficient 2. Podpora 3D grafiky a SQLite databáze jsou klíčové faktory pro volbu technologie, a proto získávají koeficient 3. Velikost uživatelské základny, dokumentace a intuitivnost jsou důležité pro budoucí pokračování projektu, a proto získávají koeficient 3. Podpora jazyků je méně důležitým faktorem, neboť důraz je kladen pouze na podporu prefereovaného jazyku Kotlin. Získává váhu 2.

Technologie	Body
OpenGL ES	6.7
Cocos 2d-x	6.7
jMonkey	5.6
Unity	7.6
LibGDX	7.6

Tabulka 3.16: Výsledný bodový zisk frameworků a enginů pro GUI

V aplikaci Myšák 2 je místo rozsáhlého enginu vhodnější použít open-source framework menšího rozsahu. Ten vývojáři ponechá volné ruce a zároveň mu poskytne intuitivní vývojové podmínky. Projekt je nestandardní svou nutností podpory kamery a integrace OpenCV knihovny. Nejvhodnější pro nové jádro jsou frameworky Cocos 2d-x a LibGDX. Mají mnoho společných rysů a liší se především volbou programovacího jazyka, která má vliv na kompatibilitu s kamerou zařízení. Obrazovky v LibGDX jsou plně svázány s životním cyklem aktivity a poskytují podobné rozhraní jako aktivity v Android SDK. Kamera je knihovnou LibGDX podporována a obsahuje ve své dokumentaci návod na její konfiguraci. Proto byl do nové verze aplikace zvolen framework LibGDX.

3.5.2 Poziční značky

Všechny frameworky a knihovny jsou založeny na podobném základu. V principu se detekce značek provádí pomocí jediné metody, která vrací pole detekovaných značek. Kvůli volbě frameworku LibGDX je velmi vhodná podpora jazyku Kotlin, popřípadně C++, které se dá s Kotlinem pomocí JNI propojit. Jazyky budou hodnoceny koeficientem 3. Komunita, intuitivnost a dokumentace budou hodnoceny koeficientem 2, neboť se jedná o důležité aspekty pro navázání na projekt. Do hodnocení jsou zahrnuty pouze knihovny a frameworky, které podporují OS Android. Kategorie podpory platforem informuje o adaptabilitě frameworku na multiplatformní projekt. Nová verze aplikace se díky LibGDX, který byl zvolen jako technologie pro GUI aplikace, multiplatformním projektem stane. Podpora více platforem je žádoucí, a bude hodnocena koeficientem 2. Každá knihovna umí detekovat svůj vlastní systém pozičních značek. Do výsledného hodnocení jsou zahrnuty knihovny podporující čtvercové poziční značky. Podpora více systému značek je výhodou, avšak nehraje důležitou roli. Proto bude tato kategorie hodnocena koeficientem 1.

Technologie	Body
ArUco knihovna	6.2
ArUco modul	7.8
ARToolKit	5.3
Chillitags	5.2

Tabulka 3.17: Výsledný bodový zisk frameworků a knihoven detekce pozičních značek

Nejvhodnější technologií je i po analýze stále ArUco modul pro OpenCV knihovnu, a byl proto použit do nové verze aplikace. Jelikož je jeho přidání do OpenCV problematické, je součástí této práce i návod (viz 6.4.1).

3.5.3 ORM

Největší nevýhody ORM použitého v aplikaci jsou slabá podpora vazeb a špatná dokumentace. Proto získají váhový koeficient 3. Podpora platforem je důležitý aspekt, kvůli zvolené technologií LibGDX. Je vhodné, aby mohlo být pro všechny platformy použito jedno ORM. Podpora platforem má váhový koeficient 2. Velikost uživatelské komunity a intuitivnost jsou důležité při řešení obtíží. To je vhodné kvůli budoucí návaznosti na projekt, a proto získává též koeficient 2. Všechna ORM podporují alespoň jazyk Java a body ztrácejí, pouze pokud z nějakého důvodu špatně podporují Kotlin. Tato vlastnost získává koeficient 1.

Technologie	Body
SugarORM	5.2
GreenDao	6.6
Room	5.8
OrmLite	7.0
SQLDelight	6.15
Realm	7.0

Tabulka 3.18: Výsledný bodový zisk ORM frameworků

Nejvyššího hodnocení dosáhly frameworky OrmLite a Realm. Do nové verze aplikace bylo vybráno OrmLite, kvůli jeho snadnému použití se současnou SQLite databází. Jedná se o multiplatformní ORM, které lze používat jak pro Android databáze, tak pomocí JDBC. Realm není přímo ORM, přestože usnadňuje operace nad databází. Jedná se o objektovou databázi, která není, s ohledem na databázový model, pro projekt příliš vhodná. V databázi se ukládají u většiny tabulek pouze cesty k souborům, a proto je vhodné použít raději relační databázi a ORM.

Kapitola 4

Návrh nové architektury

Tato sekce se zabývá návrhem nové architektury aplikace Myšák 2, jehož cílem je odstranit nedostatky předchozí verze. V nové verzi aplikace byly provedeny následující změny:

- Výměna knihovny AlphaMovieView za framework LibGDX.
- Výměna současné verze OpenCV knihovny za vlastní distribuci s ArUco modulem. OpenCV knihovna se zbaví úplné závislosti na C++ a kód zpracování obrazu bude psán v Kotlinu (viz sekce 2.7).
- Výměna SugarORM za multiplatformní OrmLite.

Změny mají významný dopad na návrh nové architektury aplikace. Základní rozdělení architektury do modulů bude definováno způsobem použití frameworku LibGDX. Nabízí se 2 možnosti postupu:

LibGDX uvnitř View ve Fragmentu Pro platformu Android existuje možnost inicializovat framework LibGDX ve View uvnitř layoutu obrazovky s použitím Android SDK. Tato možnost by ponechala stávající UI aplikace. Ve fragmentu, který vykresluje myšáka na obrazovku by místo přehrávání videa knihovnou AlphaMovieView bylo použito API frameworku LibGDX a vykreslovale by se přímo 3D postava myšáka.

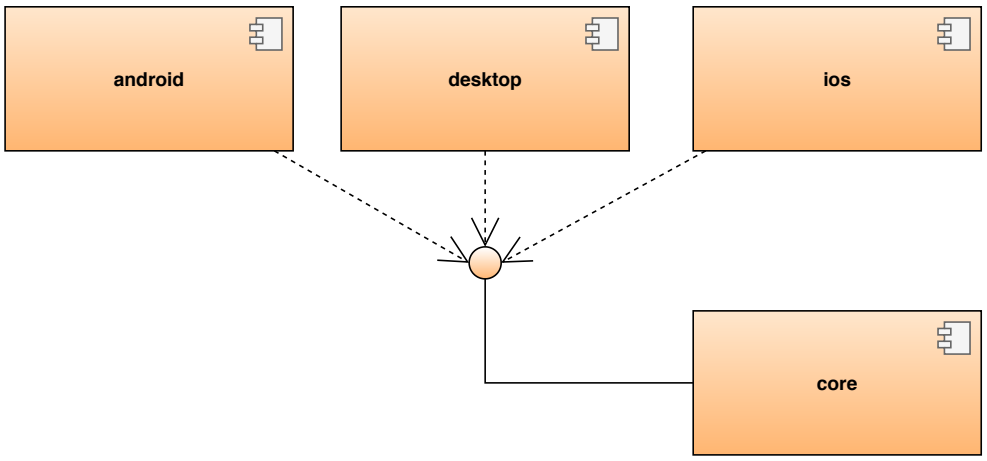
Tato cesta není v současné době vhodná kvůli přechodu Android knihoven na namespace AndroidX. V současnosti Google požaduje, aby měly aplikace nastaveno cílily na SDK verze 28, které vynucuje použití AndroidX knihoven. Kvůli nim obsahují třídy LibGDX pro Android jako je AndroidFragmentApplication neplatné importy na staré verze knihoven, kvůli kterým nelze tyto třídy zkompilovat. V současné době ke dni 10. 4. 2020 odstranilo LibGDX z dokumentace podporu týkající se tohoto způsobu použití frameworku, a proto nebude zvolen jako výchozí pro novou verzi aplikace.

Založení multiplaformního projektu Framework poskytuje na oficiálních stránkách utilitu, která vygeneruje multiplatformní projekt založený na systému Gradle se všemi potřebnými závislostmi. Vygenerovaný projekt podporuje deploy na Android, Desktop, iOS, pomocí WebGL do prohlížeče a je možné jej nahrát do libovolného vývojového prostředí (dále IDE).

S ohledem na potíže s kompatibilitou bude zvolena druhá možnost. Dosavadní architektura aplikace bude zahozena a bude vytvořena zcela nová založená na LibGDX. Aplikace Myšák 2 cílí pouze na Android zařízení. Nicméně bude vhodné přidat do projektu i moduly platforem iOS a Desktop, když tuto cestu LibGDX podporuje. V projektu budou jejich moduly pouze vytvořeny. Návrh a implementace platformě závislých částí pro desktop a iOS budou ponechány do případných budoucích prací.

Vygenerovaný projekt obsahuje 4 moduly:

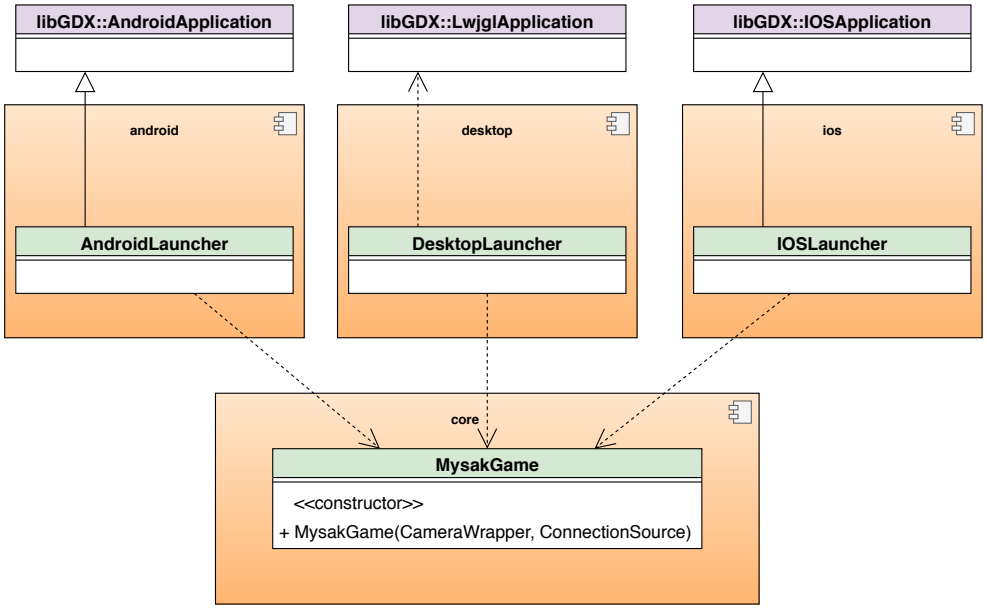
1. core
2. android
3. desktop
4. ios



Obrázek 4.1: Diagram modulů a jejich závislostí

Diagram 4.1 zobrazuje vzájemné závislosti modulů. Moduly řešící spouštění pro konkrétní platformu mají závislot na modulu core, který jim poskytuje své api. Hlavní třídou modulu core, která je poskytnuta navenek ostatním modulům je třída MysakGame, která je potomkem třídy Game (dále herní třída). Její instance je vytvářena v modulu konkrétní platformy pokaždé pro jinou konfiguraci. Modul core nemá závislosti na žádné platformě.

Následující diagram 4.2 demonstruje spuštění aplikace na konkrétní platformě. Každý `Launcher` vytváří při inicializaci instanci třídy `MysakGame` a poskytne ji konfiguraci pro běh na konkrétní platformě.



Obrázek 4.2: Diagram architektury, reprezentující spouštění aplikace na různých platformách

Cílem návrhu je umístit co nejvíce zodpovědnosti do modulu core tak, aby mohl být jeho kód využit na všech platformách nezávisle. V modulech pro konkrétní platformy se v novém projektu nacházejí pouze spouštěcí třídy, které správně nastaví nakonfigurují instanci třídy MysakGame. Některé události nelze přesunout do jádra aplikace a musí být vyřešeny na konkrétní platformě. Jedná se především o čtení dat z kamery a přístup k databázi.

4.1 Nové požadavky

Pro novou verzi aplikace byly aktualizovány funkční a nefunkční požadavky na aplikaci. Byly odstraněny požadavky týkající se uživatelských účtů. Doplněny byly nové požadavky ohledně pracovních listů a zpětné vazby. Přepracovány byly i požadavky na herní část a grafické prostředí. Tato bakalářská práce se zaměří pouze na nutné požadavky. Ostatní budou přenechány navazujícím pracím. Nové požadavky jsou součástí příloh této práce. Lze je též dohledat na adrese `https://tinyurl.com/MysakPozadavkyNew`.

4.2 Jádro aplikace

Tato část návrhu se zaměřuje na modul `core`, tedy třídy společné pro všechny platformy. Jádro aplikace bude využívat třívrstvou architekturu a bude podle ní rozděleno do následujících balíčků:

view bude obsahovat třídy uživatelského rozhraní, obrazovky, jejich stavy a kontroléry.

logic bude balíček obsahující data pro zpracování obrazu a veškerou business logiku aplikace

data bude obsahovat třídy datové vrstvy, tedy třídy provádějící operace nad databází.

Každé vrstvě bude věnována sekce návrhu.

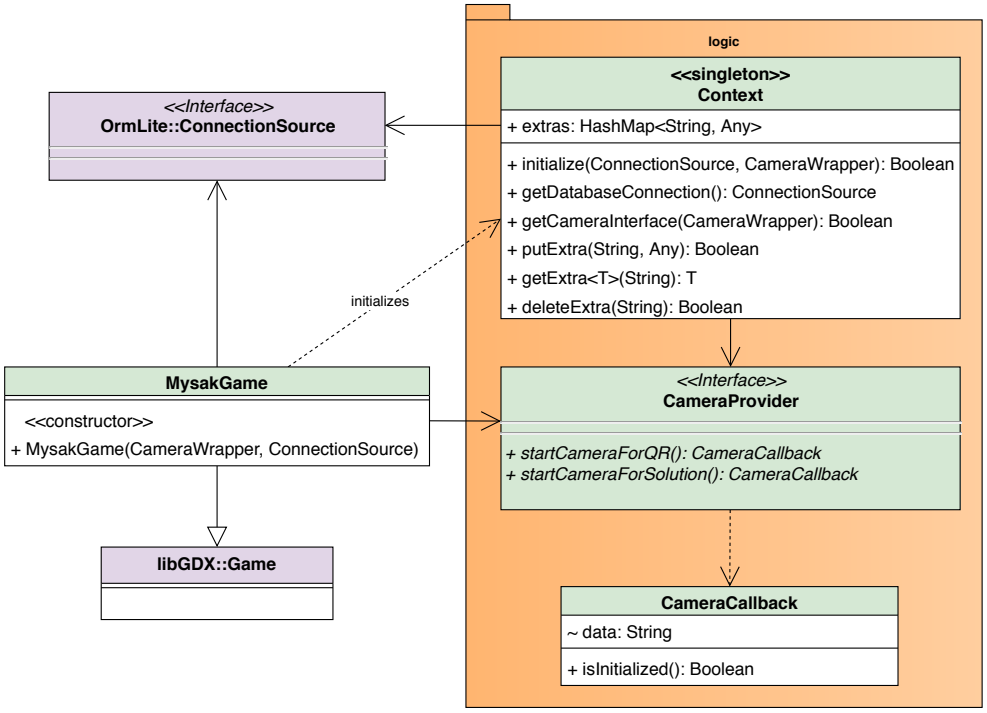
4.2.1 Logic

Tato sekce se bude zabývat byznys logikou aplikace. Bude se týkat především tříd pro zpracování obrazu, tedy hodnocení úkolů. Bude se týkat i základních tříd jádra, které se konfigurují těsně po spuštění. Do sekce logic spadají i třídy pro komunikaci s kamerou zařízení. Ta je však závislá na konkrétní platformě a netýká se výhradně jádra. Předávání dat mezi jádrem a třídami platformy Android se věnuje sekce 4.3. Tato sekce se nebude zabývat ani návrhem modulu pro načítání pracovních listů, přestože do balíčku logic spadá. Jemu je určena samostatná kapitola 5.

Hlavní třídou jádra je MysakGame. Tato třída řídí celý chod aplikace. V aplikaci Myšák 2 bude kromě toho poskytovat i databázové připojení a API pro komunikaci s kamerou na zařízení v podobě tříd `CameraWrapper` a `ConnectionSource`. Třída bude inicializovat `Context` a uloží do něj instance těchto dvou tříd. `Context` bude sloužit k poskytování rozhraní pro otevření kamery a připojení do databáze globálně kdekoliv v aplikaci. Bude obsahovat i přechodné úložiště některých instancí objektů, které je vhodné napříč aplikací sdílet. Může jít například o cesty vedoucí k dočasně vytvořeným souborům. Návrh třídy `Context` je inspirován třídou `Intent` z Android SDK. Třídy a jejich vzájemné závislosti znázorňuje diagram 4.3.

4.2.1.1 Vyhodnocování úloh

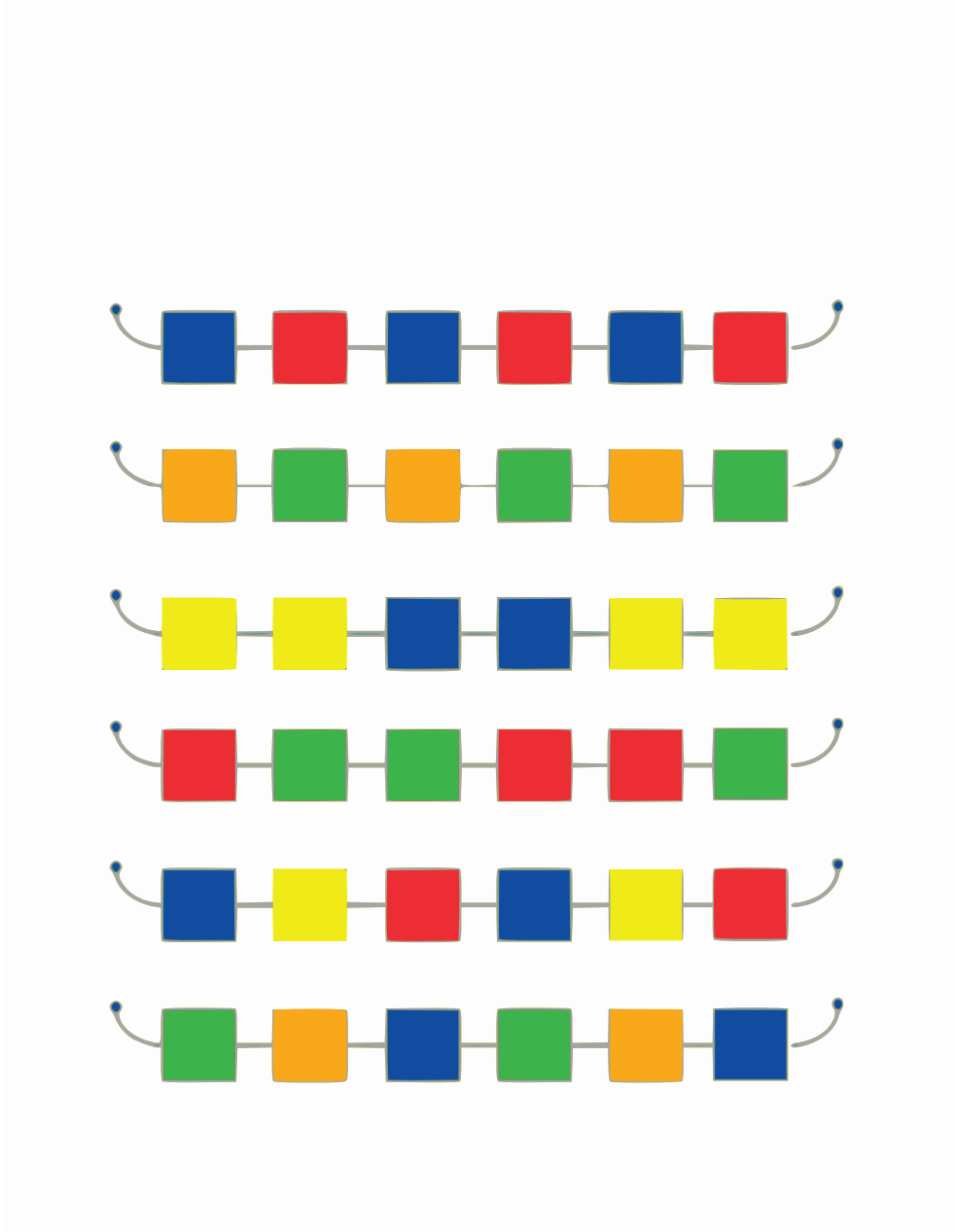
V úvodu této kapitoly se zmiňuji o odstranění závislosti na C++. Toto rozhodnutí vychází z vyhodnocení analýzy předchozího stavu aplikace 2.7. V nová verzi budou algoritmy pro zpracování obrazu knihovnou OpenCV napsány v Kotlinu. Při používání Java API OpenCV knihovny je nutné dbát zvýšené pozornosti při správě paměti. Podrobněji se tomuto problému věnuje sekce 4.4.



Obrázek 4.3: Hlavní třída a inicializace kontextu

Algoritmy zpracování obrazu se v nové verzi nebudou strukturálně lišit. Dojde pouze k jejich přepsání do Kotlinu a několika jejich úpravám, na základě poznatků z analýzy. Jejich úkolem je zpracovávat načtené pracovní listy. Cílem algoritmů není řešit natáčení obrazu, či provádění jeho korekcí. Korektní snímek budou očekávat již na svém vstupu v podobě jako na obrázku 4.4. Pro načítání pracovních listů bude vytvořen samostatný modul, kterému se věnuje sekce 5.

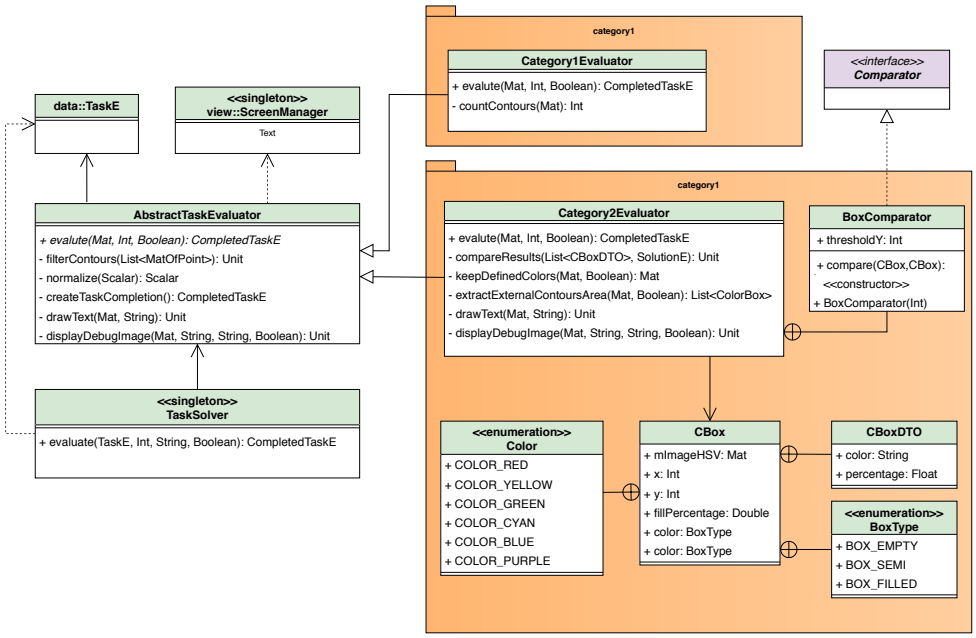
Nové upravené algoritmy vychází z přepisu jejich původní C++ verze. Jednotlivé metody budou rozděleny do tříd podle kategorií úloh, kterých se týkají. Návrh těchto tříd demonstruje diagram 4.5. Modul pro ohodnocení pracovních listů bude navenek dostupný třídou `TaskSolver`. Ta na základě poskytnuté úlohy poskytne správný `Evaluator`, který ji ohodnotí. Každá kategorie úloh bude mít v aplikaci vlastní balíček, ve kterém bude `Evaluator` pro danou kategorii a další pomocné třídy. Využití modulu v aplikaci bude triviální. Metodě evaluate třídy `TaskSolver` bude předán snímek pracovního listu a zadání úkolu. Metoda vrátí ohodnocené řešení.



Obrázek 4.4: Načtený pracovní list

Druhá kategorie úloh

V druhé kategorii úloh byly provedeny změny na základě nedostatků detekovaných v předchozí verzi aplikace. Ta měla velkou chybovost, kvůli chybnému řazení barevných čtverečků. Nový systém založený na komparátoru



Obrázek 4.5: Nový modul pro hodnocení úloh

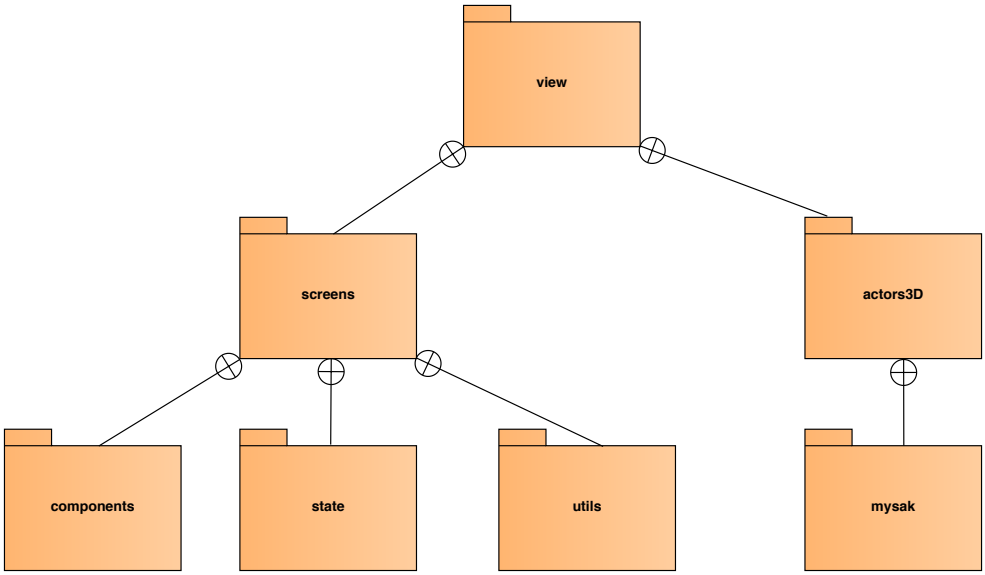
`BoxComparator` (viz diagram 4.5) řadí primárně podle y-ové a sekundárně podle x-ové souřadnice. Řazení je obohaceno o práh, který určuje výškovou toleranci. Určuje rozmezí výšky v pixelech, o kterou se dva barevné čtverečky mohou lišit, a je vypočítáván z rozměrů obrázku. Jeho účelem je odstranění chybovosti při zjištění pořadí čtverečků na pracovním listu.

V druhé kategorii byly upraveny HSV rozsahy jednotlivých barev tak, aby mezi sebou nekolidovaly. V nové verzi umí druhá kategorie úloh stále vyhodnotit pouze 6 barev:

- Červená
- Žlutá
- Zelená
- Tyrkysová
- Modrá
- Fialová

4.2.2 View

Tato sekce se zabývá návrhem nového technického řešení GUI. Vzhled obrazovek v aplikaci zůstane v nové verzi stejný a bude vycházet z wireframů softwarového projektu. Změněno bude jejich technologické řešení a vykreslování postavy myšáka. Jedním z nedostatků předchozího projektu bylo kromě způsobu animací i použití samostatné aktivity pro každou obrazovku. Přestože se většina obrazovek z wireframů téměř neliší ve svém vzhledu, byla při přechodu na další obrazovku instancována nová aktivita. Předchozí aktivita a její stav se uložily na zásobník a pokud nebyla volána nová aktivita pro čtení z kamery, byla vytvořena defakto kopie současné obrazovky pouze s jinou logiku uvnitř svých tlačítek a jinými animacemi. Tomu je potřeba se v nové verzi aplikace vyhnout.



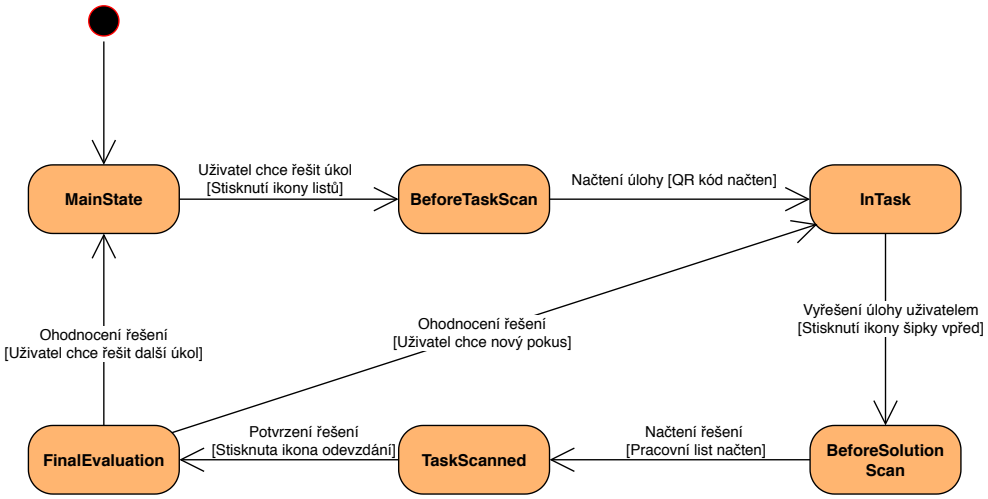
Obrázek 4.6: Rozdělení View do balíčků podle zodpovědností

View bude rozděleno do dílčích balíčků podle zodpovědnosti jednotlivých komponent. Balíček `screens` bude obsahovat vše týkající se obrazovek a balíček `actors3D` objekty, které budou vykreslovány na scénu.

LibGDX používá k reprezentaci obrazovky podobný systém jako aktivity v Android SDK. Obrazovku reprezentuje rozhraní `Screen`. Jakákoliv třída, která jej implementuje může být jako obrazovka zobrazena na display. Jednotlivé obrazovky jsou přepínány v herní třídě. Ta v sobě drží referenci na aktuální obrazovku, která je vykreslena na popředí. Herní třída neukládá obrazovky na backstack. Při výměně konkrétní obrazovky je předchozí instance zahozena, což nemusí být vždy chtěné. Některé funkce Android SDK používané v předchozí verzi aplikace tak bude nutné reimplementovat.

V obrazovce je definována `Stage`, která zodpovídá za nastavení pozice i vykreslování jednotlivých prvků. Vykreslování probíhá v obrazovce v metodě render. K vytvoření 3D scény v LibGDX slouží třídy `ModelBatch` a `Camera` (neplést s kamerou zařízení). Batch představuje 3D scénu, která obsahuje objekty a stará se o jejich vykreslování. Kamera definuje bod v prostoru této scény, ze kterého se na scénu uživatel dívá. Kamera i Stage si na displayi definují svou operační oblast. Stage může například zabírat pouze oblast navigační lišty a kamera oblast od lišty výše. Může tak být kombinováno 2D UI s 3D scénou. Součástí UI bude původní lišta a ikonka s nápovědou.

Obrazovky aplikace se téměř neliší. Většina z nich obsahuje navigační lištu, ikonku nápovědy, postavu myšáka a pozadí. Množství obrazovek se tak dá redukovat na jedinou, která se nachází v různém stavu. To co obrazovka zobrazuje je na tomto stavu závislé. Stavy obrazovky jsou uspořádány v sekvenci a mají dán přesný sled. Nelze mezi jednotlivými stavy měnit náhodně. Stavy hlavní obrazovky znázorňuje digram 4.7.



Obrázek 4.7: Stavy hlavní obrazovky

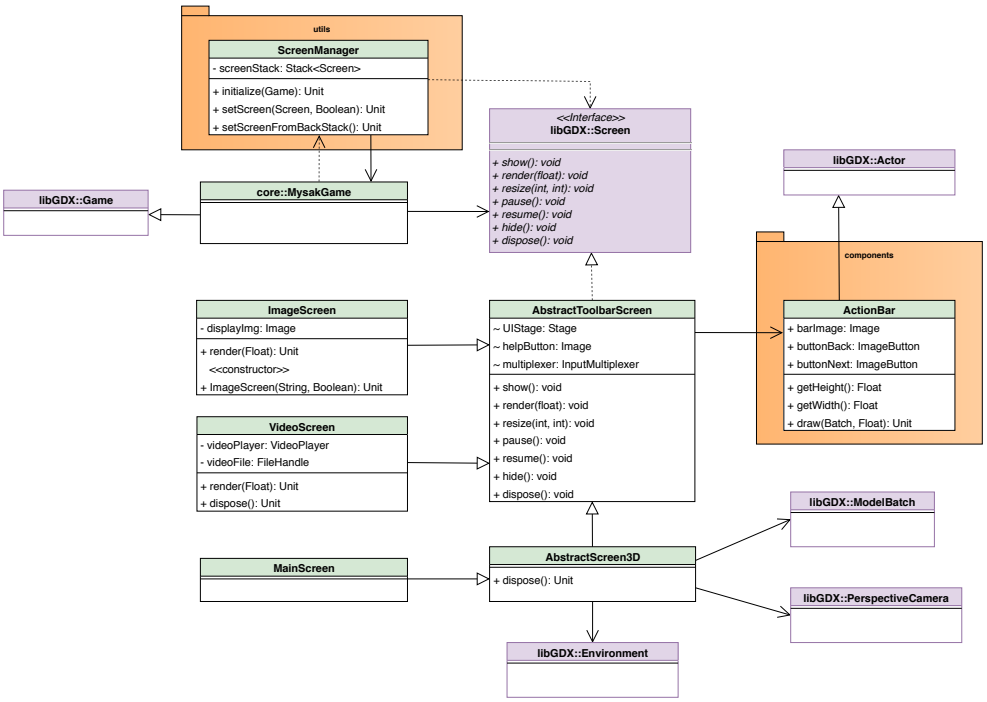
Kdyby byla při přechodu do dalšího stavu vytvořena nová obrazovka muselo by dojít k těmto událostem:

1. Smazání instanci staré obrazovky, což zahrnuje drahé operace. Její součástí je 3D scéna, na kterou se vykreslují modely. Instance celé scény všetně kamery a modelů by byly smazány.

2. Vytvoření instance nové obrazovky. To zahrnuje opět vytvoření původní 3D scény, kamery a instancování potřebných modelů. Jednalo by se o tutéž scénu, tytéž modely, jen v jiném stavu.

Mazání a vytváření 3D scény je drahá operace, která by mohla mít negativní dopad na výkon. Další možností je oddělit celé vykreslování do jiné třídy, která by byla obrazovkami sdílena. Oddělení vykreslování od obrazovky je komplikované, neboť je svázané s obrazovkou a musí probíhat v rámci jejího životního cyklu.

V nové verzi bude proto definována jedna obrazovka s několika stavy. K jejich vytváření, uchovávání a výměně bude sloužit `StateController`. Stavy a kontroler se budou nacházet v balíčku state. V balíčku components se budou nacházet jednotlivé prvky 2D uživatelského rozhraní a balíček utils bude obsahovat nástroje pro správu obrazovek (viz 4.6).

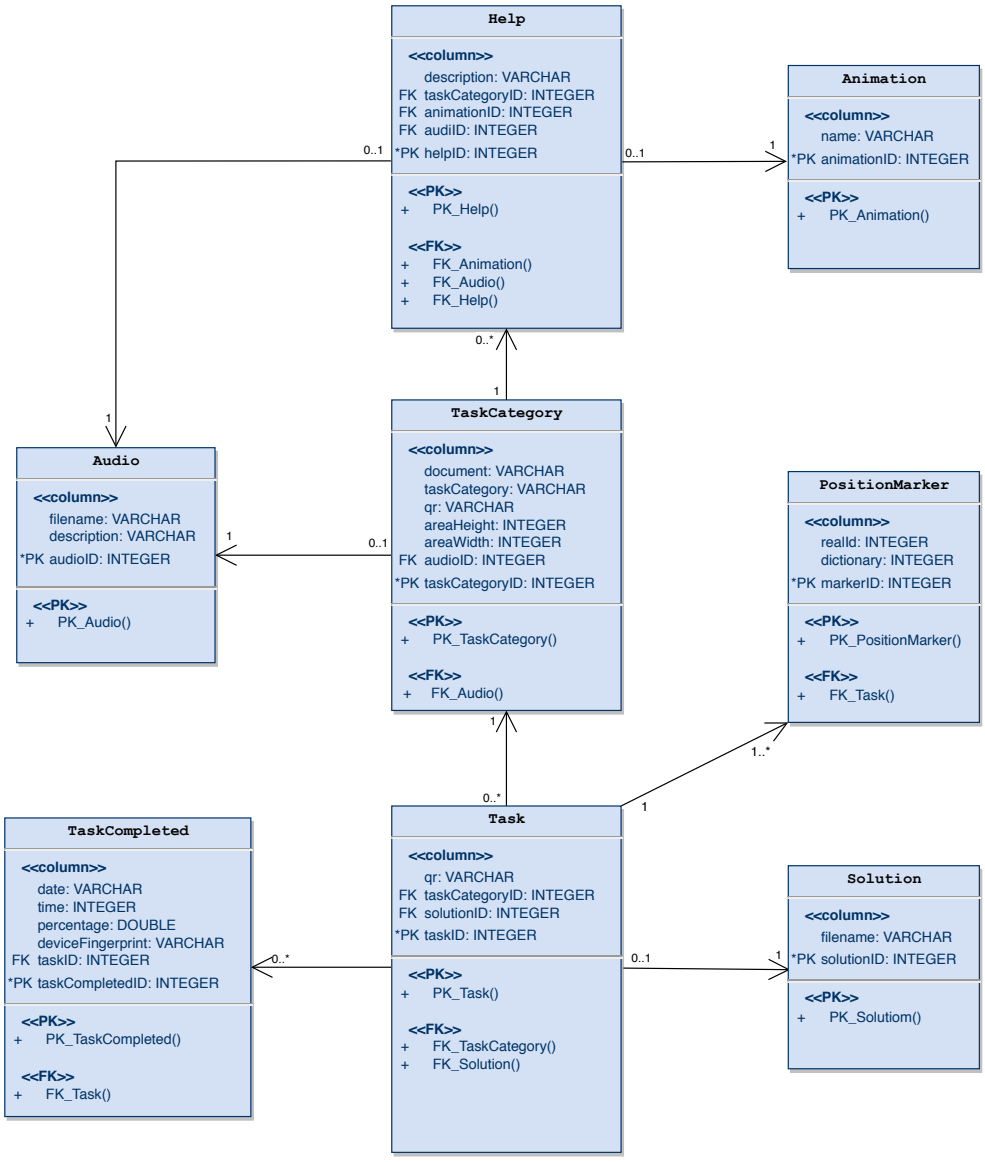


Obrázek 4.8: Návrh tříd balíčku screens

V balíčku actors3D budou definovány postavy, které se budou vykreslovat na obrazovku. V nové verzi aplikace půjde pouze o hlavní postavu myšáka, která bude nadále v aplikaci sloužit jako průvodce. Bude mít vlastní třídu, která bude sdružovat jeho model a kontroler pro přehrávání animací a mluvení.

4.2.3 Data

Datová vrstva aplikace bude v nové verzi rozšířena o nové tabulky. Návrh třídy pro operaci s databází je ovlivněn novou technologií ORM. Byl upraven i databázový model, který je nyní rozšířen o nové tabulky.



Obrázek 4.9: Nový databázový model

Za zmínku stojí nové tabulky:

Kategorie Sdružuje úlohy, které jsou v principu stejné a liší se pouze obměnou zadání. Úlohy z kategorie mají stejné hlasové zadání.

Nápověda Kategorie může mít několik různých nápověd, ale nemusí nutně obsahovat žádnou. Nápovědy by měly být univerzální a vysvětlit pouze princip řešení, místo řešení konkrétní úlohy. Nápověda má kromě animace i hlasový popis.

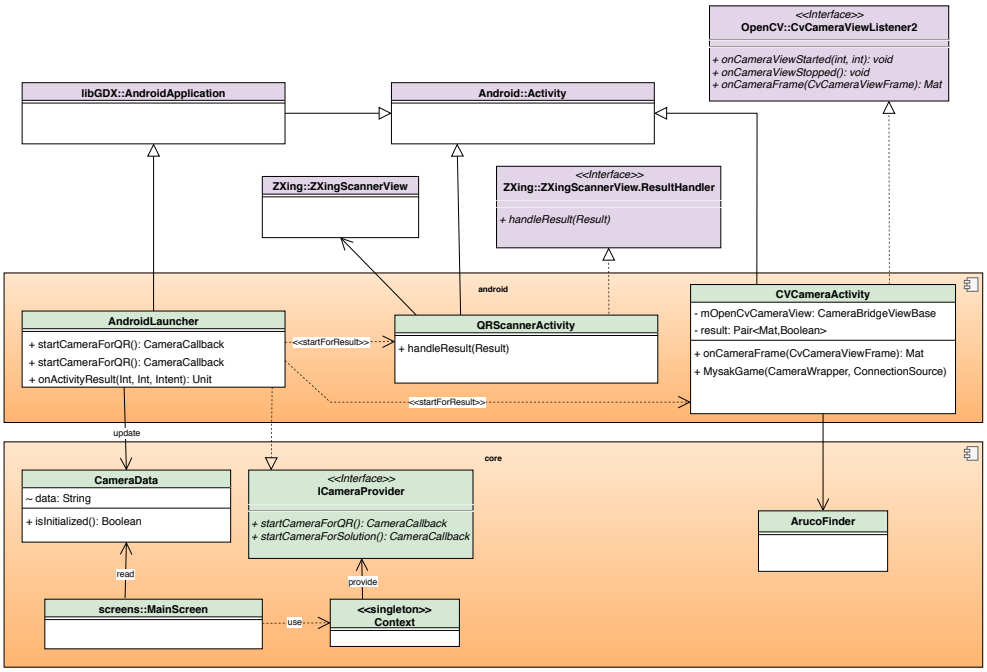
Poziční značka Neukládá obrázkovou reprezentaci značky, ale pouze její ID a ID slovníku, do kterého patří. Značka je svázána s úkolem, ke kterému je přiřazena a neměla by být použita u více různých úkolů. Každý úkol by měl mít přesně 4 značky. Použití značek je podrobněji popsáno v sekci 5.

Operace nad databází budou prováděny v hlavním vlákně. Databáze obsahuje z převžné části statická data, přístup do ní je synchronní a za běhu aplikace nejsou žádná data změněna (pouze jsou přidávány nové záznamy vypracovaných řešení). Poté co je načtena úloha, je ihned přistupováno i do záznamů jejích značek, kategorie a hlasového zadání. Proto bude pro ně bude uplatněno hladové načítání (eager loading).

Za zmínku v nové verzi stojí třída `DatabaseConfigUtil`, která slouží ke konfiguraci frameworku OrmLite pro platformu Android. Tato třída slouží k vytvoření cache paměti pro DAO na platformě Android. Do pomocného souboru ve složce res je zapsán konfigurační soubor, ve kterém jsou definovány závislosti mezi třídami. Tento soubor značně snižuje nároky kladené na zařízení a zrychluje přístup do databáze až desetkrát. DatabaseConfigUtil je samostatná utilita spustitelná v JVM. Spouští se na zařízení, které probíhá vývoj aplikace a generuje konfigurační soubor, který se nahrává na cílové zařízení. Používání konfiguračního souboru lze, za cenu možného zpomalení aplikace vyřadit.

4.3 Čtení dat z kamery

Čtení dat z kamery je v nové verzi aplikace komplikovanější kvůli použití frameworku LibGDX. Kvůli platformě specifickým vlastnostem kamer zařízení není vhodné umístit tuto zodpovědnost do modulu `core`. Je proto delegována na moduly jednotlivých platforem. Diagram 4.10 popisuje nově navržené třídy a jejich vzájemné vztahy. K získání dat z kamery zařízení slouží rozhraní ICameraProvider a třída CameraData. Jsou navrženy tak, aby bylo možné používat kameru na všech platformách. Nejomezující je při návrhu platforma Android. Ta potřebuje otevřít kameru v nové aktivitě. Otevření nové aktivity je asynchronní proces a je proto nutné nenarušit životní cyklus současné aktivity, ve které hra běží. Proto se v modulu android po zavolání metody startCameraForQR pouze vytvoří požadavek na otevření aktivity kamery. Ta se otevře později.

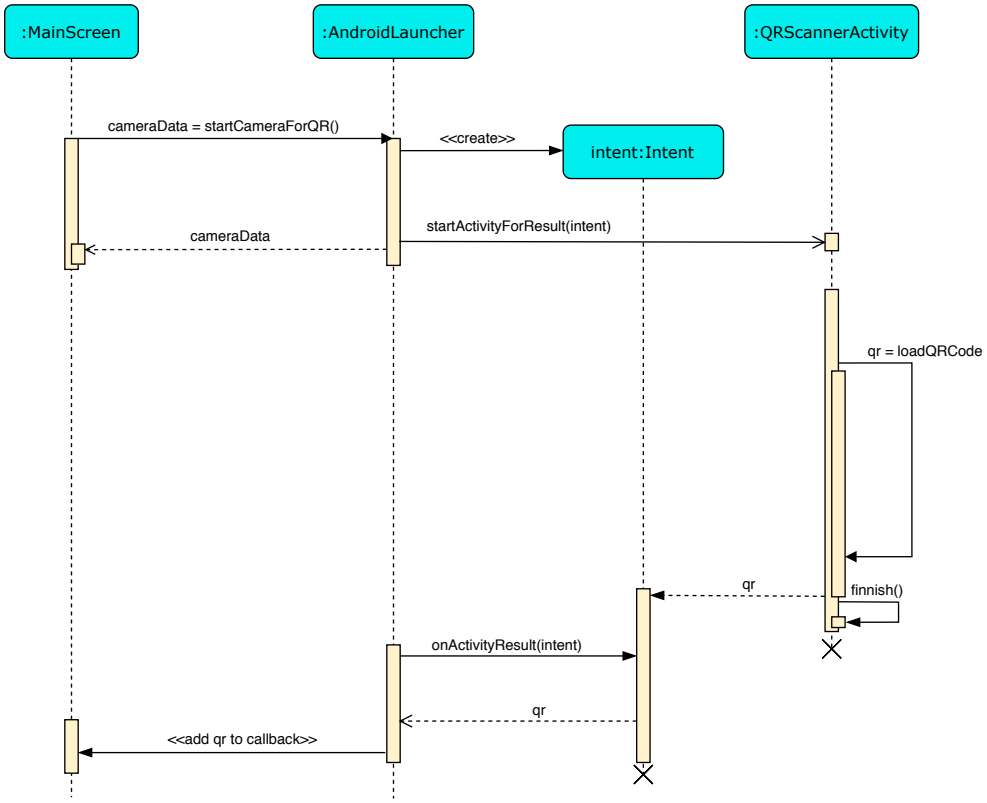


Obrázek 4.10: Přístup ke kameře a předávání dat do jádra

Proces čtení se skládá z těchto kroků:

1. Zavolání metody startCameraForQR kdekoliv z jádra aplikace.
2. Implementace této metody v modulu android vytvoří novou instanci aktivity kamery a zažádá o přepnutí do ní pomocí metody startActivityForResult.
3. Metoda vrátí prázdnou instanci třídy CameraCallback volajícímu.
4. Doběhnutí životního cyklu herní aktivity a její přesunutí na backstack.
5. Přepnutí do nové aktivity kamery.
6. Čtení dat z kamery.
7. Ukončení aktivity kamery, probuzení herní aktivity a vrácení dat z kamery do ní.
8. Úprava instance CameraData, která byla předána jádru. V modulu android na něj stále existuje reference.

Sekvenční diagram 4.11 znázorňuje tento proces na příkladu při čtení QR kódu.



Obrázek 4.11: Sekvenční diagram čtení QR kódu z kamery

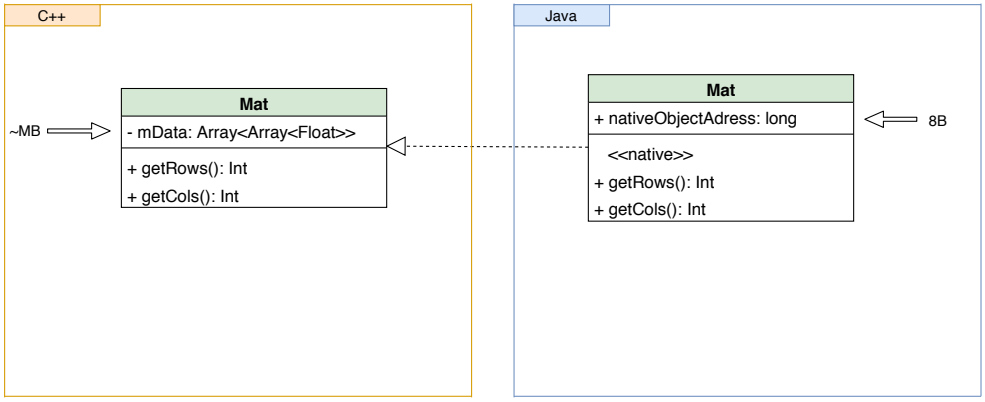
V jádru je aktuální obrazovka probuzena společně s obnovením herní aktivity. Obsahuje nyní data z kamery, která mohou být dále zpracována. Celý proces je asynchronní, avšak na data z kamery nebude přistoupeno dříve, než bude ukončen fotoaparát a obnovena aktivita launcheru. Ten je do té doby uspán na backstacku a nemůže s těmito daty nijak manipulovat. V jádře se po probuzení v aktuální obrazovce v metodě resume zkontrolují data vrácená z kamery, která se dále zpracují.

4.4 Správa paměti

Na základě testování 7.1, které se zabývalo porovnáním detekce značek v jazycích Kotlin a C++, budou v nové verzi aplikace přepsány algoritmy pro zpracování obrazu z C++ do Kotlinu. Výhody plynoucí z tohoto rozhodnutí jsou popsány v sekci 2.7.

Jazyky C++ a Kotlin se liší ve způsobu jakým je spravována paměť programu. Na desktopovém zařízení je Kotlin zkompilován na bytecode, který je interpretován pomocí Java Virtual Machine (dále JVM) a přenositelný napříč platformami, na kterých je tento virtuální stroj nainstalován. V Android zaří-

zení je Kotlin přeložen na speciální Dex bytecode, který je dále interpretován běhovým prostředím Android Runtime (dále ART). [52]. V obou případech podléhají spuštěné Java aplikace automatické správě paměti pomocí garbage collectoru (dále jen GC). Ten se v JVM i v ART liší, ale sdílí společný problém při správě nativní paměti. Java rozhraní OpenCV knihovny představuje pouze wrappery, které na pozadí stále používají původní C++ implementaci knihovny. Tato skutečnost může způsobit řadu komplikací právě při správě paměti, kterou tak nelze ponechat pouze na GC. Při použití Java API OpenCV knihovy je důležité uvolňovat instance objektů, které zabírají mnoho prostoru v paměti. Příkladem takových objektů jsou instance třídy Mat. Tato třída slouží nejčastěji k reprezentaci obrázků, které jsou v ní uchovány jako pole pixelů. Instance této třídy může reprezentovat velké množství dat v závislosti na rozlišení obrázku.



Obrázek 4.12: Mapování objektů z Java rozhraní OpenCV na jejich C++ implementaci

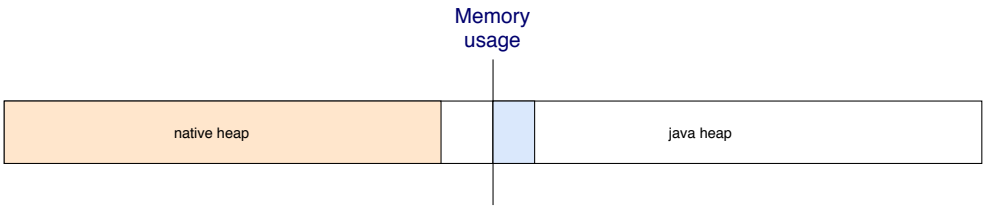
Většina tříd OpenCV knihovny má strukturu, kterou demonstruje diagram 4.12. Na něm je představena zjednodušená implementace třídy Mat v Java API OpenCV knihovny. Skutečná struktura této třídy je komplikovanější a diagram slouží pouze k nastínění problematiky. Pokud je v Android aplikaci použit nativní kód, je jeho paměť oddělena od paměti virtuálního stroje. Nativní část aplikace má na haldě vyhrazen svů prostor – nativní haldu. Tato nativní halda je oddělena od zbytku aplikace a jelikož není spravována virtuálním strojem, je nutné zde provádět manuální správu paměti. Za správu paměti OpenCV knihovny zde ručí C++ api. Třídy knihovny mají definovány vlastní destruktory, které se o uvolnění paměti starají.

Při vytvoření instance třídy Mat v Javě, je vytvořen i přidružený nativní objekt na nativní haldě, který je pomocí své adresy mapován na Java instanci [53] (3:56). Java instance udržuje pouze ukazatel o velikosti 8B na tento nativní objekt, zatímco skutečná data (pole pixelů v případě třídy Mat) se nachází na nativní haldě. Java API pro OpenCV je implementováno tak, že pokud dojde

k uvolnění Java instance garbage collectorem, bude i na jeho přidruženém nativním objektu zavolán destruktor, který jej uvolní z nativní haldy.

Uvolňování paměti pomocí GC je nedeterministické. Pořadí, ve kterém jsou objekty uvolňovány závisí na mnoha faktorech. V ART se alokované objekty dělí do generací podle jejich očekávané velikosti a životnosti. Každá z těchto generací má definovaný horní limit paměti, kterého může dosáhnout. Zaplnění paměti aktuální generace ovlivňuje frekvenci sběru paměti, ke kterému pak dochází častěji. [54].

Z těchto skutečností plynou některé krajní situace, které způsobí nežádoucí chování a pád aplikace.. Příkladem může být frekventované vytváření mnoha lokálních instancí velkých objektů. V aplikaci Myšák 2 se tato situace odehrává při čtení snímků z kamery. Při čtení je každou vteřinou vytvářeno několik snímků, a každý je uložen v instanci třídy Mat. Následkem je mnohem rychlejší zaplňování nativní části haldy, než části pro Javu. V té budou z pohledu virtuálního stroje čekat na uvolnění pouze osmi bajtové reference, které nezabírají téměř žádné místo. Virtuální stroj o existenci nativní haldy neví. Může se proto domnívat, že paměti je dostatek a GC bude spouštět méně frekventovaně. [55](12:50). Situaci znázorňuje obrázek 4.13.



Obrázek 4.13: Zaplnění nativní haldy. Garbage Collector neví o nativní části haldy

Operační systém Android byl původně navržen i pro zařízení s malou kapacitou paměti. Aktuálně spuštěným procesům přiřazuje priority, podle kterých se rozhoduje, který proces bude při nedostatku paměti zabit [56]. Výše popsaná situace může způsobit nekontrolovatelné zahlcení paměti spuštěnou aplikací, kvůli nedostatečnému uvolňování nativní části haldy. Ve chvíli, kdy začne operačnímu systému docházet paměť, sám aplikaci zabije [55] (4:58). Odhalit skutečnou příčinu situace může být obtížné, neboť nedojde k žádnému chybovému výpisu.

Java API OpenCV knihovny s těmito problémy počítá. V objektech, které při alokaci zabírají mnoho paměti, jsou proto v Java API definovány metody pro přímé volání destruktoru, které objekt uvolní z nativní haldy. Pro třídu Mat je to metoda `release`. Na každé vytvořené instanci je nutno zavolat tuto metodu v okamžiku, kdy již instance není potřeba. Tato skutečnost ovlivňuje návrh a implementaci nových algoritmů používajících Java API OpenCV knihovny.

Kapitola 5

Návrh modulu pro načítání pracovních listů

Tato kapitola se zabývá návrhem nového modulu pro načítání pracovních listů. Jeho cílem je odstranit nedostatky v hodnocení uživatelských řešení popsané v analýze v sekci 2.5. Základní ideou modulu je detekce pozičních značek, které budou umístěny na obrázek, a ze kterých bude následně určena poloha a orientace pracovního listu v prostoru.

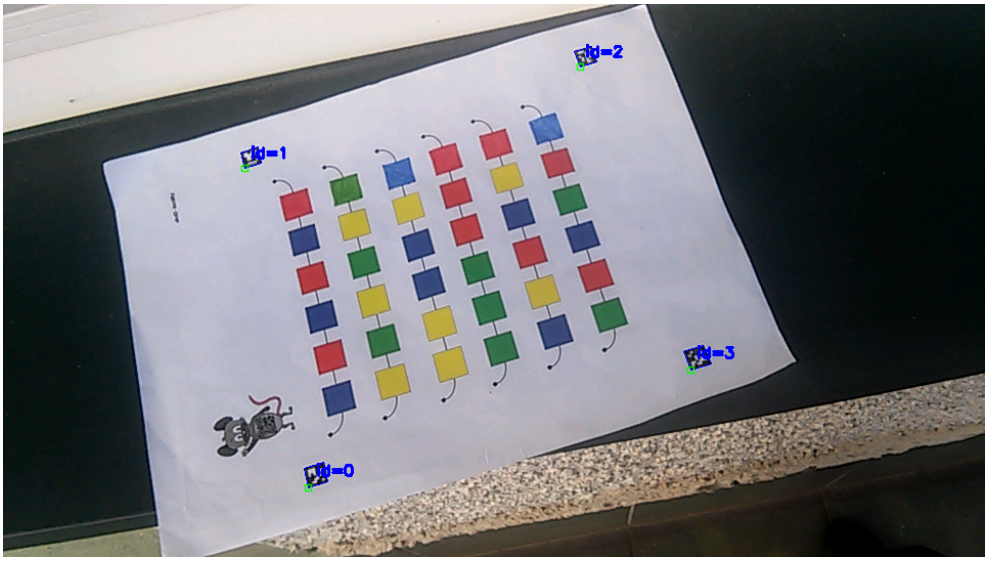
Je tedy potřeba najít způsob, jak ze snímku pomocí detekovaných značek načíst pracovní list. Ten nemusí být načítán celý, stačí pouze oblast, na kterou uživatel kreslí. Pokud by byla vymezena umístěním pozičních značek, bylo by možné ji ze snímku extrahovat pomocí geometrických transformací obrazu.

5.1 Geometrické transformace obrazu

Většina poznatků o transformacích obrazu v této sekci pochází z knihy Moderní Počítačová Grafika [12] a dokumentace OpenCV knihovny [57]. Cílem modulu je ze snímku kamery 5.1, na kterém budou detekovány značky, extrahovat cílovou oblast, do které je vyplněno uživatelské řešení 5.2.

Transformace pracují buď s body nebo se spojitou reprezentací objektu, tedy s rastrovou či vektorovou reprezentací obrázku [12](Kapitola 4.3). V případě aplikace Myšák 2 se jedná o rastrové snímky, které jsou čteny z kamery. S transformací rastrového obrazu přichází několik komplikací, neboť se jedná o diskrétní reprezentaci.

Geometrická transformace obrazu je funkce, která pixelu o celočíselných souřadnicích [i, j] ve vstupním obrazu přiřadí pixel o souřadnicích [x, y] ve výstupním obrazu. Souřadnice [x, y] nemusí být diskrétní a proto musí docházet k mapování. V nově transformovaném obraze mohou vznikat prázdná místa, nebo se může několik pixelů s různými souřadnicemi namapovat na



Obrázek 5.1: Snímek s načtenými pozičními značkami obdržený z kamery.

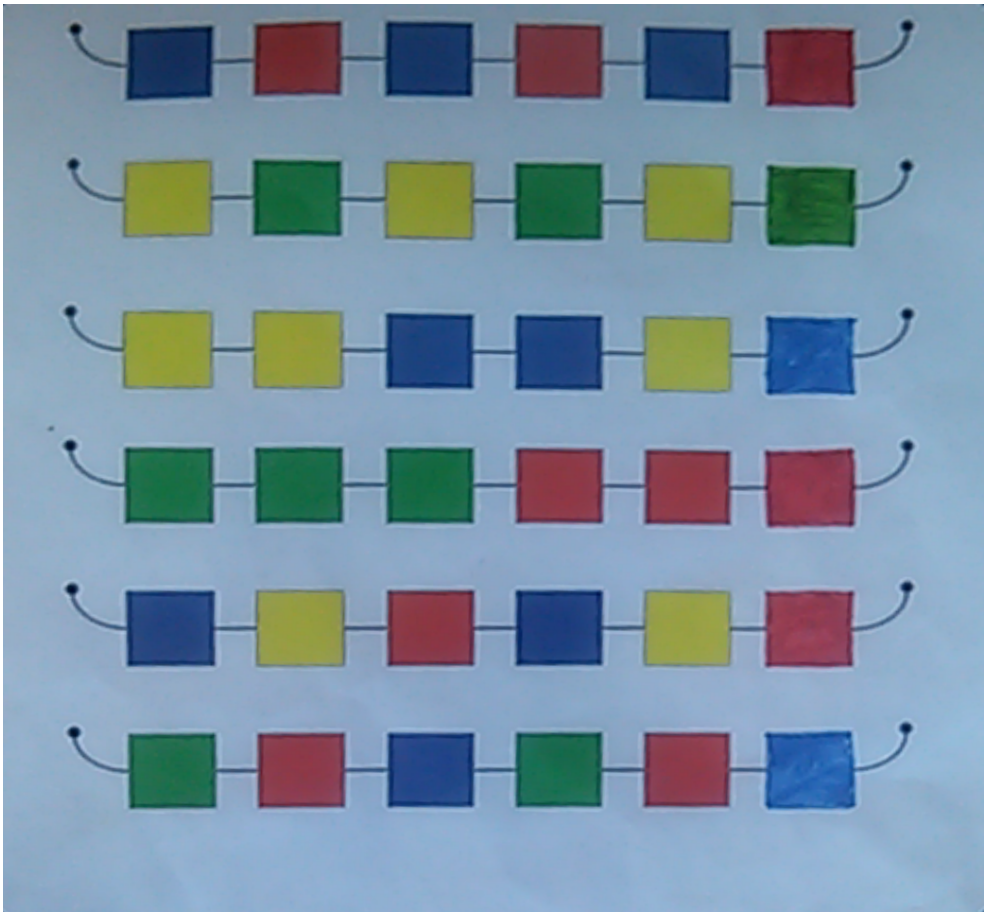
stejné místo [12](Kapitola 4.3).

Mapování určuje způsob přiřazení pixelu ze vstupního obrazu do výsledného transformovaného. Mapování může být rozděleno na dopředné a zpětné. Dopředné mapování zobrazuje každý pixel ze vstupního obrazu do cílové oblasti výstupního obrazu. Zpětné mapování prochází výsledný obraz a každému pixelu přiřazuje jeho vzor ve vstupním obrazu. Geometrické transformace nemusí při mapování zachovávat velikost obrazu, což podporuje vznik potenciálních prázdných míst. Tento problém se týká hlavně dopředného mapování, a proto se v praxi používá mnohem méně [12]. Geometrické transformace není třeba implementovat, neboť jich mnoho poskytuje OpenCV knihovna. Všechny tyto funkce používají zpětné mapování.

5.1.1 Převzorkování

Převzorkování (resampling) je posloupnost operací, kdy se nejprve signál rekonstruuje, poté se zpracuje a opět se vzorkuje zpět do diskrétní oblasti [12](Kapitola 4.3). Tato technika se používá hlavně kvůli výše zmíněné možnosti vzniku prázdných míst v obrázku (děr). Pokud funkce zvětší transformací obrázek dvakrát na výšku i na šířku s předpisem $T(u, v) = [2u, 2v]$, je potřeba dopočítat pixely, které se budou v novém obrázku nacházet na lichých pozicích. To znamená, že transformovaný obraz může být zkreslený, neostrý.

Poziční značky jsou však náchylné na příliš silnou rotaci, která by způsobila nečitelnost obrazu a nejsou v takovém případě detekovány. Tato skutečnost značně omezuje potíže s načtením příliš zkresleného snímku.



Obrázek 5.2: Snímek získaný transformací obrazu z 5.1.

5.1.2 Warping

Warping je transformace, kterou lze použít jak u diskrétního, tak u spojitého obrazu. Pomocí warpingu lze i měnit perspektivu snímku a jedná se tak o vhodnou transformaci pro použití v aplikaci. Cílem je změnit perspektivu snímku tak, aby poloha kamery při jeho focení odpovídala rovnoběžnému pohledu shora (viz 5.3).

Transformace obecně mohou být reprezentovány pomocí matic a pixely v obrázku jako vektory pomocí homogenních souřadnic. Geometrická transformace je tedy provedena pomocí násobení matic. Každý pixel o souřadnicích (x,y) je doplněn na homogenní souřadnice a vynásoben touto maticí zobrazení, čímž získá odpovídající souřadnice v cílovém obrázku.

OpenCV knihovna nabízí ve svém API funkce, které provádí různé druhy transformací. K provedení perspektivní transformace je k dispozici funkce `warpPerspective` [57]. Jediné, co je k provedení transformace potřeba je zdro-

jový obrázek a matice zobrazení. Ani ta nemusí být počítána manuálně, jelikož pro její získání z referenčních bodů nabízí OpenCV funkci `getPerspective`. K odvození matice zobrazení jsou potřeba dvě čtveřice bodů. Každá musí být vzájemně různá a tvořit čtyřúhelník. První čtveřice odpovídá konkrétním bodům ve zdrojovém obrázku a druhá jejich poloze v cílovém obrázku po provedení transformace.

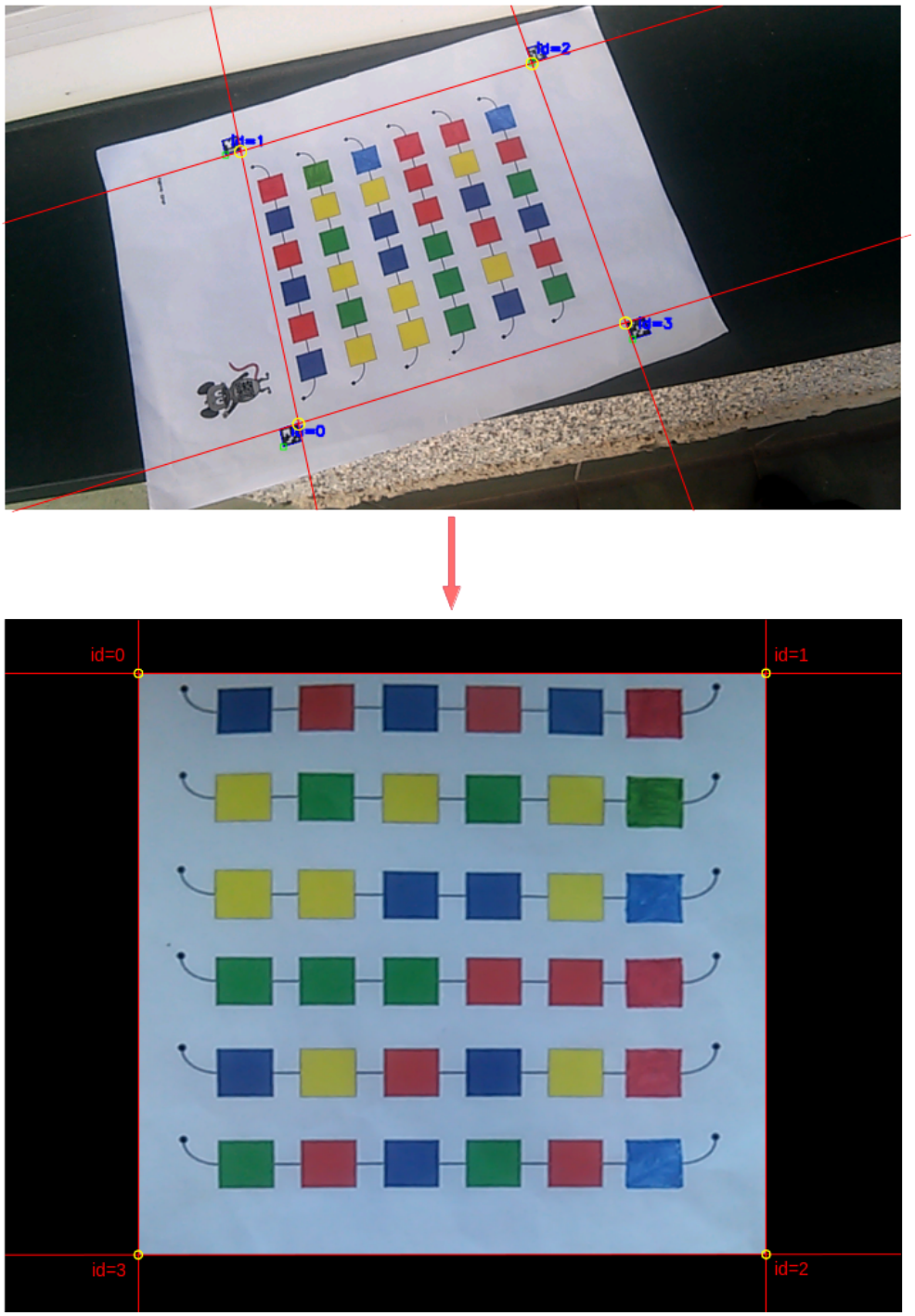
Úprava pracovního listu

Provedení transformace tedy závisí na mapování bodů z detekovaných značek na jejich nové umístění v transformovaném obrázku. Poziční značky ArUco modulu mají svou vlastní identitu, kterou uchovávají v podobě číselného identifikátoru (dále ID). Tím od sebe mohou být jednoznačně odlišeny. Umístěním čtyř značek na papír bude jejich detekcí docíleno zisku pozice čtyř požadovaných bodů pro určení perspektivní transformace. Značky budou svou vnitří oblastí tvořit obdelníkový tvar a budou dodržovat na každém listu z kategorie stejné umístění. Díky tomu bude možné dopočítat skutečné rozměry oblasti ze snímku kamery a vždy určit správnou oritentaci kamery vůči pracovnímu listu. Každý list bude mít pevně dané umístění konkrétních značek. Ty budou podle svého ID rozmístěny vzestupně takto:

1. levý horní roh (nejnižší ID)
2. pravý horní roh
3. pravý dolní roh
4. levý dolní roh (nejvyšší ID)

Polohy kamery tak bude nezávislá na orientaci listu ve výstupním snímku. Po detekci konkrétní značky bude podle jejího ID zjištěno, o který roh cílové oblasti se jedná. Ukázkovou situaci detekce a následné perspektivní transformace znázorňuje obrázek 5.3. Z něj vyplývá, že není potřeba, aby transformovaný snímek obsahoval celý pracovní list. Redundantní jsou veškeré informace, kromě oblasti uvnitř pozičních značek. Proto budou referenční pozice značek zvoleny jako rohové body výstupního obrázku. Ostatní pixely zobrazené mimo snímek se oříznou, což je na obrázku 5.3 znázorněno černou oblastí okolo výstupního snímku.

Zbývá pouze zvolit velikost výstupního obrázku, do kterého bude zakreslena transformovaná oblast mezi pozičními značkami. Nesmí být závislý na rozlišení snímku kamery, které může být téměř libovolné. Poměr výšky a šířky v něm nemusí odpovídat skutečnému poměru výšky a šířky zdrojového obrázku. Skutečnou výšku a šířku oblasti nelze ze vstupního snímku určit, neboť vzdálenosti podléhají perspektivnímu zkreslení. Do databáze proto bude ke kategorii úloh uložen skutečný poměr výšky a šířky cílové oblasti na pracovním listu tak, aby mohly být odvozeny správné rozměry obrázku.



Obrázek 5.3: Perspektivní transformace pomocí mapování souřadnic pozičních značek

5.2 Algoritmus načítání pracovního listu

Nyní jsou definovány potřebné transformace a lze přejít k návrhu algoritmu samotného. Ten bude založen na následujícím principu.

1. Ve smyčce budou čteny snímky z kamery, budou zpracovány a zobrazeny na výstup
2. Každý snímek bude podroben detekci značek.
3. Značky budou ve snímku barevně ohraničeny a upravený snímek se zobrazí na display.
4. Pokud budou detekovány 4 poziční značky, které patří k úloze, bude ze snímku načtena cílová oblast pracovního listu.
5. Ukončí se čtení z kamery a transformovaný snímek je vrácen jádru aplikace pro další zpracování.

Při detekci může nastat několik situací, které musí být vyřešeny.

Uživatel se pokusí načíst pracovní list jné, než aktuální úlohy V takové situaci bude upozorněn a pracovní list se nenačte. Značky se ohraničí pomocí jiné barvy.

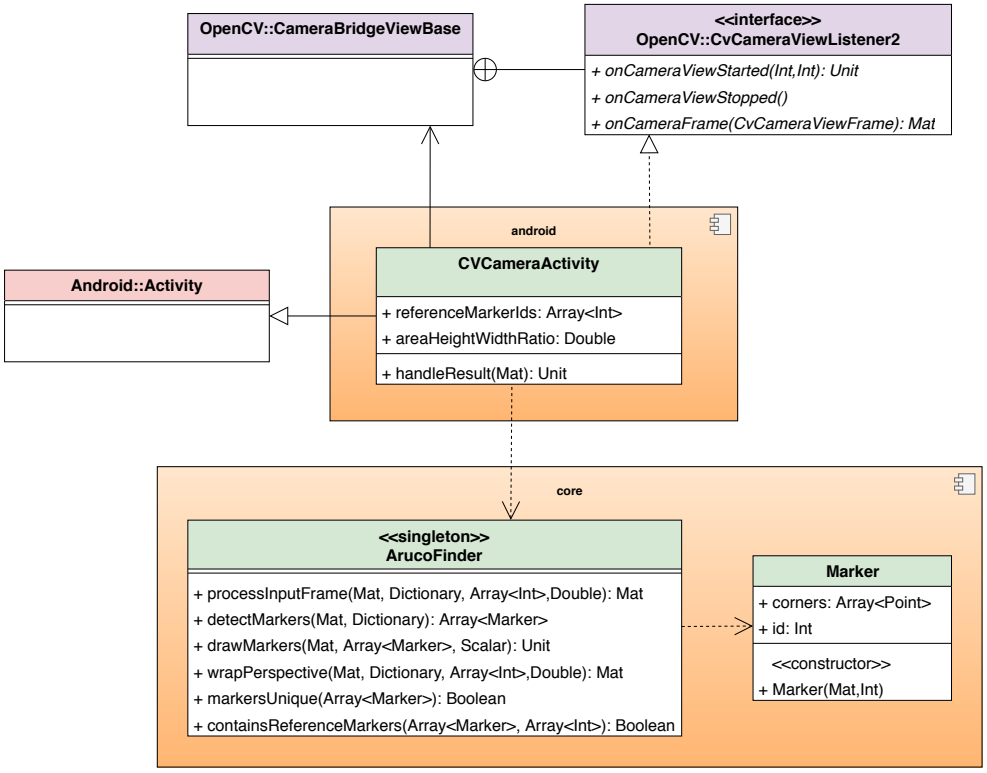
Na snímku je 5 a více značek K takové situaci by mohlo dojít, kdyby měl uživatel na stole připraveno více pracovních listů, například složených na sobě. Tato situace může mít 2 podoby:

- Detekované značky patří k jiným úlohám.
- Na snímku se vyskytují duplicitní značky.

V prvním případě budou značky ohraničeny jinou barvou a při provádění transformací obrazu ignorovány. Druhý případ je komplikovaný. Tato situace nastane, pokud bude mít uživatel připraveno více pracovních listů od stejné úlohy pro případ, že by jeho řešení bylo neúspěšné a nedalo se opravit v aktuálním listu. V tomto případě budou značky zvýrazněny speciální barvou a opět k detekci nedojde.

Některé mezní situace (především duplicitní značky) jsou snadno detekovatelné, avšak obtížně řešitelné v kódu. Lze je řešit informováním dětského uživatele. Při duplicitní detekci například vyzvat uživatele, aby ze záběru odstranil přebytečný pracovní list.

Diagram 5.4 znázorňuje nárh tříd modulu pro detekci značek. K manipulacím se snímkem z kamery slouží třída `ArucoFinder`. Ta bude poskytovat jak metody provádějící celý proces detekce a transformace, tak dílčí metody při ní použité jako veřejné.

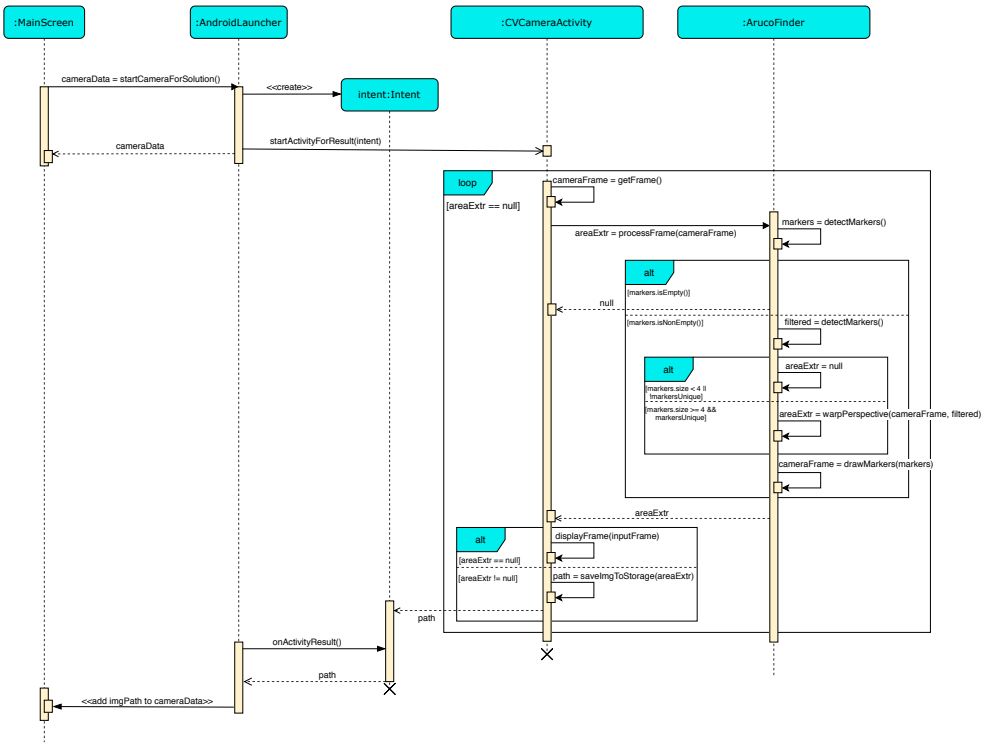


Obrázek 5.4: Návrh tříd pro detekci pozičních značek

Sekvenční diagram 5.5 shrnuje detekci pozičních značek, od požadavku z jádra po jeho vrácení. Jedná se o analogii čtení QR kódu. Tentokrát je však načítán pracovní list.

5.2.1 API OpenCV knihovny

Při analýze modulů a knihoven pro poziční značky byly v sekci 3.2.2 zmíněny nedostatky API OpenCV knihovny. Moduly kvůli zpětné kompatibilitě často používají ve funkcích výstupní parametry. Například funkce `detectMarkers` z ArUco modulu detekuje značky do dvou oddělených výstupních proměnných. První představuje pole identifikátorů značek a druhá dvourozměrné pole jejich souřadnic. V Java rozhraní knihovny se navíc pro reprezentaci těchto proměnných nepoužívá pole jako v C++, ale matice, do které se hodnoty ukládají do sloupce nebo řádku. V OpenCV API pro Javu je tato praxe běžná, přestože třída `Mat` je pro reprezentaci některých typů hodnot nevhodná. Lze dohledat řadu předpřipravených konvertorů, které transformují instance třídy Mat podle hodnot v nich uložených, například na pole bodů. Největší slabinou ArUco modulu je absence jednotné třídy, která by reprezentovala poziční značku. Bez zavedení jednotné třídy pro poziční značky by například jejich



Obrázek 5.5: Načítání pracovních listů v OS Android

řazení podle identifikátoru bylo velice obtížné, neboť jsou atributy pozičních značek uloženy ve dvou separátních strukturách. Řazení značek je nutné kvůli dodržování daného pořadí značek na pracovním listu.

V modulu pro detekci značek byly navrženy třídy a funkce, jejichž cílem je nedostatečné API zapouzdřit a navenek poskytnout kvalitnější. Poziční značka s jejím id a rohovými body bude reprezentována třídou ArucoMarker (viz 5.4).

Dalším znatelným nedostatkem je pevná orientace kamery na landscape mode při použití podpůrné třídy CameraBridgeViewBase. Tato třída poskytuje intuitivní rozhraní pro čtení dat z kamery. Poskytuje čtecí smyčku pro snímky z kamery, které vrací přímo ve formátu Mat, který OpenCV používá pro reprezentaci obrázků. Tyto snímky mohou být modifikovány a předány k zobrazení na display. Použití rozhraní pro komunikaci s kamerou, které knihovna poskytuje je přesto výhodné. Případná orientace kamery může být simulována v kódu.

Kapitola 6

Implementace

Kapitola se zabývá výslednou implementací vycházející z návrhu. Je shrnuto kolik požadavků bylo realizováno, a v čem se implementace liší od návrhu. Následně jsou uvedeny manuály na kompilaci OpenCV knihovny a její následné linkování do projektu v Android Studiu.

Nová verze aplikace se nachází v novém repozitáři mimo projekt původní aplikace ze softwarového projektu. Má velmi odlišnou architekturu, a kromě vzhledové stránky spolu projekty nemají mnoho společného.

6.1 Změny oproti návrhu

Většina navržených tříd a se v průběhu implementace nezměnila. Přesto se v implementaci oproti návrhu vyskytují odlišnosti.

Načítání pracovních listů

V modulu pro načítání pracovních listů se v metodě pro kompletní zpracování snímku z kamery zakreslí detekované značky přimo do vstupního obrázku, místo do vytvořené kopie. Důvody tohoto kroku vychází ze správy paměti (viz 4.4). Vytváření instancí třídy Mat v cyklu způsobuje leaky v paměti, pokud se jedná o lokální proměnnou tohoto cyklu. Jednou z možností je přidat do singeton třídy `ArucoFinder` pro detekci značek privátní atribut, který bude v každé iteraci cyklu při čtení snímků z kamery přepisován. Tím by však do ní byl zanesen nechtěný stav. Pro potřeby algoritmu není nutné zachovat původní obraz bez zakreslených pozic značek. Kopie by byla vytvořena zbytečně a původní instance by musela být při čtení uvolněna. Pokud bude při budoucím rozšíření aplikace potřeba zachovat při detekci původní obraz, bude muset být vytvořena jeho kopie před zavoláním metody.

Uživatelské rozhraní

Nové uživatelské rozhraní vychází z původních wireframů a vzhledově se téměř neliší od původní verze aplikace. Rozdílná je jeho vnitřní implementace. Existují nyní pouze dva druhy obrazovek:

MainScreen Hlavní obrazovka, ve které se odehrává celý chod aplikace. Má svůj kontrolér a stavy.

ImageScreen Slouží k zobrazení dodatečných obrazovek při lazení algoritmů pro zpracování obrazu.

V rámci implementace byly provedeny i drobné změny uživatelského rozhraní. Pro ikona nápovědy, která je nyní na každé obrazovce byla zamýšlena novoá funkce. Po jejím stisknutí by sloužila jako hlasová nápověda kdekoliv v hlavní obrazovce. Měla sloužit uživateli, kterému měla kdykoliv vysvětlit prvky UI a jednotlivé kroky, které má v aktuálním okamžiku udělat. Pro nápovědu úkolu byla zvolena ikonka knihy, která bude dostupná pouze ve stavu zadání úkolu. Do nové verze byl použit původní dabing, který by měl být v budoucnu rozšířen. V současné době je aplikace kvůli uživatelskému rozhraní obtížně testovatelná. Důvodem je model postavy myšáka. V průběhu implementace se ukázalo se, že existují tři různé verze postavy, každá obsahuje jiné animace, a žádná z nich neobsahuje všechny potřebné. Do aplikace byla přidána starší verze modelu obsahující pouze mluvení, nic nedělání a dvě animace pro hodnocení úkolu. Například ukázkové načítání QR kódu myšákem je proto nekompatibilní s novou verzí aplikace. Model má navíc po vyexportování do formátu `g3d`, který používá LibGDX, špatně namapované textury. Při budoucí úpravě UI bude vhodné kromě jeho kompletního přepracování, zvážit vytvoření nového modelu hlavní postavy. V současné době není vhodné testovat bakalářskou práci jako celek. Závěrečné testování proto bude zaměřeno pouze na načítání pracovních listů a hodnocení úloh.

Nápověda k úkolu a přehrávání videa

V původní aplikaci byly nápovědy k úkolům přehrávány jako instruktážní videa, ve kterých bylo částečně ukázáno řešení. Nápovědy neobsahovaly postavu myšáka a v analýze bylo rozhodnuto ponechat je do nové verze aplikace. Knihovna LibGDX obsahuje doplňkový modul `VideoPlayer`, který umožňuje v jádru frameworku přehrávat videa různých formátů. Modul je však zastaralý a dlouho neaktualizovaný. Obsahuje nekompatibilní importy a používá mnoho zastaralých metod, kvůli kterým není možné jej použí v aplikaci. Jelikož se nápovědy týkají hlavně uživatelského rozhraní, nebyly v této práci implementovány. Tento problém bude zaznamenán a ponechán k řešení navazujícím pracím a projektům. Jednou z možností je implementovat přehrávání videa zvlášť, závisle na platformě. Další možností je nápovědy k úkolům přepracovat a neukládat je jako video.

Integrace OpenCV knihovny do projektu

OpenCV knihovna nemůže být do projektu nalinkována tak, jako v předchozí verzi aplikace. Knihovna nepočítá s jejím použitím v multiplatformním projektu, a proto není kód jádra knihovny oddělen od doplňkových tříd pro konkrétní platformu. To znamená, že distribuce OpenCV pro Android obsahuje i třídy specifické pro tuto platformu. Tyto třídy se vyskytují ve složce:

```
<open-cv>/sdk/java/src/org/opencv/android
```

Nový projekt aplikace je multiplatformní a zmíněný balíček `android` obsahuje závislost na Android SDK. OpenCV knihovna však musí být v projektu k dispozici jak modulu core, pro funkce zpracování obrazu a čtení pozičních značek, tak modulu android, kterému poskytuje speciální API pro čtení dat z kamery. Čtení dat z kamery je součástí balíčku `android`, na kterou nemá zbytek tříd OpenCV knihovny žádné závislosti. Nabízí se dvě možnosti jak knihovnu nalinkovat do projektu.

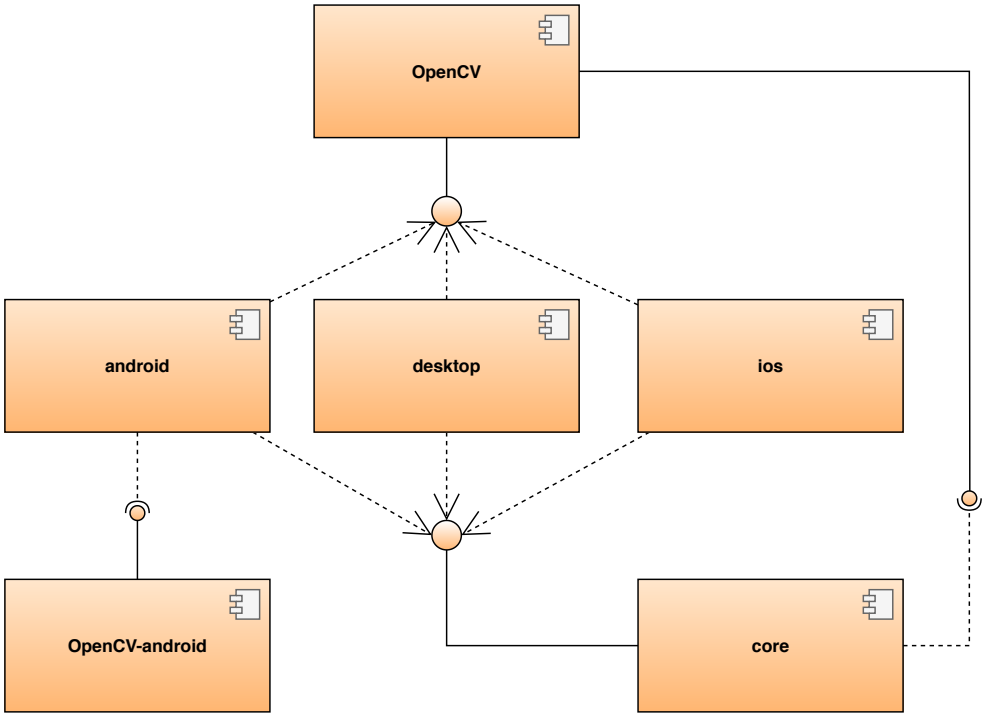
Úprava gradle konfigurace modulu core Do modulu core, místo přidání závislosti na OpenCV knihovnu, nalinkovat její třídy jako vlastní zdroje, ze kterých by byl vyloučen balíček android.

Rozdělení OpenCV knihovny Knihovna by byla rozdělena na dvě části – `core` a `android`. To je možné, neboť platformě závislé třídy pro Android jsou od zbytku knihovny plně separabilní. Tento systém používá i framework OrmLite, který je v projektu použit pro objektově-relační mapování. Skládá se z několika částí:

- ormlite-core
- ormlite-android
- ormlite-jdbc

Framework poskytuje v balíčku core společné třídy pro všechny platformy. Pro android a desktop poskytuje vlastní drivery pro připojení databáze, čehož využívá i nová verze aplikace.

V projektu byla zvolena druhá možnost kvůli přehlednosti. Modul core má závislost pouze na jádru OpenCV knihovny. Modul android závisí na též na jádru a navíc i na odděleném modulu OpenCV-android. Výsledné řešení znázorňuje diagram závislostí 6.1.



Obrázek 6.1: Diagram závislostí jednotlivých modulů na OpenCV knihovnu

6.2 Shrnutí a zhodnocení nových požadavků

Nová verze aplikace disponuje novým jádrem, které je postavené na frameworku LibGDX a je rozdělena na 3 vrstvy. Zásadní pozornost byla věnována zpracování obrazu a načítání pracovních listů. Aplikace používá SQLite databázi s novým multiplatformním ORM frameworkem. Operace nad databází jsou prováděny v modulu core a v modulech platforem probíhá pouze připojení databáze.

Aplikace obsahuje původní uživatelské rozhraní doplněné o některé nové funkce. V analýze byly uvedeny nedostatky původní verze. Na jeho grafickou úpravu, vytváření herního obsahu, či nového dabingu se tato práce nezaměřuje a pouze bylo změněno technické řešení původního UI. Tato práce neodstranila nedostatky týkající se vzhledu, či komunikace s uživatelem. Do budoucích verzí aplikace musí být uživatelské rozhraní přepracováno, aby byla aplikace pro dětského uživatele atraktivnější.

V analýze původní verze aplikace byly zdůrazněny některé původní funkční požadavky, které bude vhodné zvážit, až bude aplikace disponovat první stabilní verzí. V návrhu nové verze byly vytyčeny upravené funkční a nefunkční požadavky skládající se z původních požadavků sestavených v předmětu BI-SP1 doplněných o nové.

Nesplněné a částečně splněné požadavky

Všechny nutné požadavky byly splněny s výjimkou zobrazení řešení po konci aplikace. Většina nesplněných a částačně splněných požadavků se týká uživatelského rozhraní. Některé z nich zde budou zmíněny, neboť jsou na pomezí implementace a vzhledové stránky uživatelského rozhraní. Ostatní nesplněné požadavky jsou ty, které byly ponechány z původní verze a byly již popsány v analýze 2.8. Nové funkční požadavky jsou součástí příloh této práce. Lze je též zobrazit na adrese `https://tinyurl.com/MysakPozadavkyNew`.

Zobrazení řešení Vzorové řešení se nachází v souborech aplikace, avšak není uživateli nijak zobrazeno. Nebyl pro něj vytvořen dabing a pouhé zobrazení řešení bez komentáře by mohlo dětského uživatele mást. Do nové verze bylo ponecháno slovní ohodnocení úlohy jako správné či špatné bez vysvětlení postupu, stejně jako v původní verzi. Toto rozhodnutí bylo učiněno kvůli nedostatečným možnostem dosavadního dabingu. V navazujících pracích a projektech by bylo vhodné zaměřit se na vytvoření herního příběhu, od kterého by se dabing odvíjel, a podle tohoto příběhu jej kompletně přepracovat.

Textové zadání a hodnocení Cílem tohoto požadavku je pomoci rodičům a učitelům v případě, kdy si dětský uživatel neví rady ani po opakovaném vyslechnutí zadání a poskytnutých nápovědách. Nebudou muset vyslýchat zadání, ale zobrazí si jej v textové podobě, stejně tak nápovědy, které si dítě vzalo. Požadavek byl splněn pouze částečně. Textové ohodnocení úlohy k dispozici není a textové podoby zadání se zatím též nedají zobrazit, ale jsou již uloženy v databázi.

Nápovědy k úkolům Nápovědy k úkolům nebyly realizovány kvůli technickým omezením, která jsou popsána v sekci 6.1. Jedná se hlavně o špatnou kompatibilitu doplňkového modulu pro přehrávání videa. Nová verze obsahuje potřebné soubory nápovědy druhého úkolu, převzaté z původní verze aplikace.

Stav	Počet požadavků	Zastoupení
Splněno	20	56 %
Nesplněno	12	33 %
Částečně splněno	4	11 %

Tabulka 6.1: Závěrečné zhodnocení funkčních a nefunkčních požadavků nové verze aplikace

6.3 Správa aplikace

V nové aplikaci byly použity technologie, které usnadní její budoucí udržování a rozšiřování.

CI/CD

Chybné sestavení vlastní distribuce OpenCV knihovny, které proběhlo v rámci předmětu BI-SP2, vedlo v projektu ke komplikacím, jejichž odstranění zabralo mnoho času. OpenCV knihovna je v nové verzi aplikace, stejně jako v předchozí, linkována staticky. Má-li být průběžně aktualizována, je zde velká pravděpodobnost, že se bude opakovat podobná chyba jako ta, kterou znázorňuje obrázek 2.13, kvůlu špatně nastavené konfiguraci CMaku.

Pokud maintainer projektu provede aktualizaci OpenCV knihovny (stáhne novou verzi, sestaví ji a vymění ji za tu, která se nachází v projektu), může v ní nastat zmíněné nalinkování souborů z lokální stanice, které se při kompilaci projektu na ní neprojeví. Pokud by takto sestavenou knihovnu nahrál na verzovací systém a naklonovali si ji ostatní členové projektu, nebudou moci projekt zkompilovat. Této situaci lze zabránit pomocí průběžné integrace. Při každém novém commitu ve verzovacím systému bude projekt v odděleném prostředí sestaven a otestován.

V nové verzi aplikace bylo nastavena průběžná integrace a automatický deploy aplikace na Google Play. Pro průběžnou integraci byla použita technologie GitLab CI. Jeho konfigurace byla v novém projektu prováděna za pomoci manuálu z diplomové práce Ondřeje Slabého, kde je celý postup podrobně popsán v kapitole 4.2 [58]. Celý postup sestával z těchto kroků:

1. Vytvoření keystore pro podepisování aplikace
2. Přidání aplikace na Google Play
3. Vytvoření nového Docker image pro runner
4. Konfigurace pipeline
5. Vytvoření nového runneru, který je sdílen i s dalšími projekty

Firebase

Nová verze aplikace používá systém Firebase Crashlytics pro správu chyb. Zařízení, které aplikaci používá, odesílá při každém jejím pádu chybovou zprávu na server Firebase. Tato technologie usnadňuje sběr dat z interního testování aplikace.

6.4 Návody

Návody byly prováděny na platformě `x86_64 GNU/Linux 5.5.18-3-MANJARO` a kromě pár odlišností by měly být kompatibilní i s ostatními distribucemi Linuxu. Na platformách Windows a Mac OS se mohou lišit v použitých nástrojích (nastavení proměnných prostředí, kompilace s mingw, použití obdobných příkazů).

6.4.1 Kompilace OpenCV knihovny

Tato kapitola slouží jako manuál pro kompilaci OpenCV knihovny pro Android (vytvoření vlastní distribuce). Do vlastní distribuce je možné přikompilovat extra moduly z repozitáře opencv_contrib, které nejsou součástí oficiálního SDK. Před začátkem kompilace je nutné mít k dispozici následující nástroje:

- Android SDK – ve složce `<android-sdk>`
- Android Ndk verze 20b– ve složce `<android-ndk>`
- cmake-gui
- CMake
- GTK+2.x
- Knihovny ffmpeg nebo libav (libavcodec-dev, libavformat-dev, libswscale-de)
- Zhruba 5GB místa na disku pro jedno sestavení – vznikne velké množství souborů.

V tomto manuálu bude ke konfiguraci sestavení použit grafický nástroj cmake-gui. Sestavení knihovny proběhne pomocí CMake z příkazové řádky. Návod bude demonstrovat sestavení knihovny pro procesorovou architekturu `arm64-v8a`, jelikož bývá nejvíce problematické. Pro zaručení plné kompatibility na většině Android zařízení je potřeba vytvořit sestavení pro architektury `armeabi-v7a` a `arm64-v8a`. Pro kompatibilitu na Android zařízeních s procesorovou architekturou x86, jako je například Asus ZenFone2, je nutné knihovnu zkompilovat pro `x86` a `x86_64`. Architektury `arm64-v8a` a `x86_64` jsou 64-bitové a jsou zpětně kompatibilní s 32-bitovými verzemi knihoven. Aby však mohla být aplikace vydávána na Google Play, musí obsahovat i 64-bitové verze knihoven [59]. Pro zařízení plné kompatibility vlastní distribuce je nutné vytvořit sestavení pro `armeabi-v7a`, `arm64-v8a`, `x86` i `x86_64`.

6.4.1.1 Konfigurace CMake

Nejprve je nutné nakonfigurovat projekt OpenCV knihovny, nastavit potřebné proměnné a nalinkovat extra moduly, které budou přikompilovány. Ke konfiguraci je potřeba udělat následující kroky:

1. Na lokální stanici naklonovat z GitHubu OpenCV knihovny tyto repozitáře:
   - opencv – do složky `<opencv-source>`
   - opencv_contrib – do složky `<opencv-contrib>`
2. Ve složce `<opencv-source>/platforms` vytvořit složku `android_arm64`
3. Spustit program `cmake-gui`
4. -> Browse Source a jako zdrojovou složku zvolit `<opencv-source>`
5. -> Browse Build a jako cílovou složku zvolit `<opencv-source>/platforms/android_arm64`.
6. -> Configure -> Specify toolchain file for cross-compiling
7. Zvolit toolchain, který je součástí ndk, `<android-ndk>/platforms/android/android.toolchain.cmake` a kliknout na `Finnish`. Mnoho manuálů na internetu doporučuje použit ke konfiguraci toolchain, který je obsažen v repozitáři `<opencv-source>`. Je však zastaralý a nepodporuje ndk vyšší verze než 15c. Jeho použití vede k četným chybám při kompilaci a může způsobit i nalinkování lokálních cest ze stanice, na které kompilace probíhá (viz 2.13). Doporučené nastavení, otestované při kompilaci na Linuxu je s použitím aktuálního ndk a jeho vlastního toolchainu. V době psaní této práce se jedná o verzi ndk20b.
8. Po úspěšné konfiguraci se v logu zobrazí hláška `Configuration done`.
9. Dále je potřeba nastavit postupně následující proměnné pomocí Add Entry:
   - `ANDROID_SDK` – typ PATH a hodnota `<android-sdk>`
   - `ANDROID_NDK` – typ PATH a hodnota `<android-ndk>`
   - `ANDROID_ABI` – typ STRING a hodnota `arm64-v8a`
     Hodnota této proměnné závisí na cílové architektuře.
   - `ANDROID_PLATFORM` – typ STRING a hodnotu 21. Ta koresponduje s minSdkVersion v Android projektu. To je v době psaní této práce 21, což je minimální hodnota kterou doporučuji nastavit. Pokud není proměnná nastavena manuálně je tato hodnota nastavena na

nejnižší API level podporovaný NDK. Například pro verzi ndk20b je to 16. Důležité je tuto hodnotu změnit a nastavit alespoň verzi 17, neboť při nižším API dochází k četným chybám při kompilaci knihovny, kvůli zpětné kompatibilitě zastaralých technologií.

10. Pokud kompilace probíhá na stanici s 64-bitovou architekturou, je navíc nutné přidat proměnnou `ANDROID_NDK_HOST_X64` typu BOOL a nastavit hodnotu true (zaškrtnutím políčka Value).

11. Je potřeba zkontrolovat následující proměnné (viz 6.2):

    - `ANDROID_MIN_SDK_VERSION` – musí odpovídat `ANDROID_PLATFORM`
    - `BUILD_SHARED_LIBS` - Cílem je vytvořit statický build knihovny. Pole musí být odškrtnuté (false).
    - Pokud má být knihovna používána pouze v Javě, je vhodné pro urychlení kompilace nastavit následující proměnné na false:
      - `BUILD_EXAMPLES`
      - `BUILD_TESTS`
      - `BUILD_PERF_TESTS`

    Vypne se tak přibalení ukázkových projektů a testů pro C++.

12. -> `Configure`

13. Nyní je potřeba nalinkovat extra moduly z repozitáře `<opencv-contrib>`. Pod záložkou `OPENCV` se nachází proměnná `OPENCV_EXTRA_MODULES_PATH` Do té je potřeba přidat cestu `<opencv-contrib>/modules`, ve které se tyto moduly nachází.

14. -> `Configure`

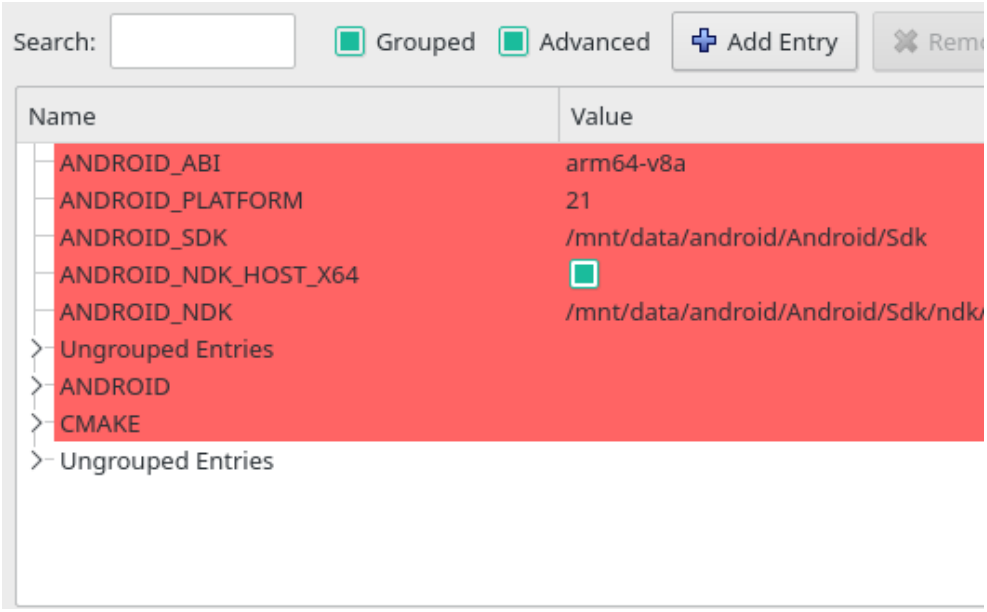
15. V sekci `BUILD` přibydou nové proměnné `BUILD_opecv_<jmeno-modulu>` Tyto proměnné říkají, který extra modul má být přikompilován do distribuce. Cílem tohoto manuálu je přidat pouze modul ArUco. Ostatní je tedy vhodné odškrnout, aby nebyly zbytečně přibaleny.

16. -> `Configure`

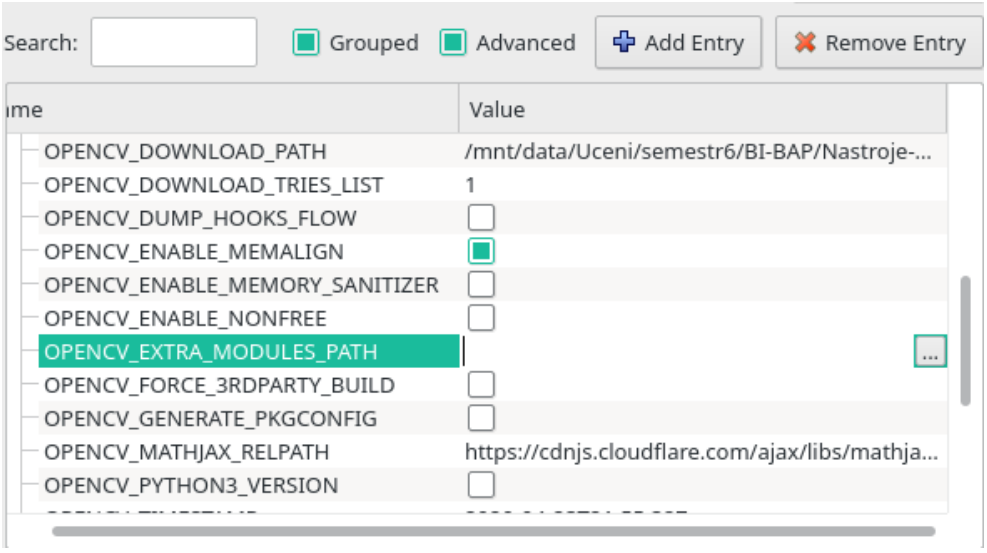
17. Je potřeba, aby v konfiguraci před generováním nebyly žádné proměnné zvýrazněny červeně. Takové by totiž nebyly aktualizovány. Pokud v konfiguraci jsou neaktuální proměnné je potřeba konfiguraci spustit znovu -> `Configure`

18. Posledním krokem je vygenerování makefile souborů pro kompilaci knihovny. -> `Generate`

Nyní je projekt připraven ke kompilaci.



Obrázek 6.2: Ukázková konfigurace kompatibility pro Android



Obrázek 6.3: Nastavení cesty k extra modulům OpenCV knihovny

6.4.1.2 Kompilace

Ve druhé fázi bude knihovna pomocí předchozí konfigrace sestavena. Před spuštěním kompilace je potřeba mít nastavené následující proměnné prostředí:

- `ANDROID_SDK_HOME` – jako `<android-sdk>`
- `ANDROID_NDK_HOME` – jako `<android-ndk>`

Postup kompilace je následující:

1. Nejprve je potřeba přejít do složky `<opencv-source>/platforms/android_arm64`
2. Z příkazového řádku spustit příkaz `make -j4` a počkat až kompilace proběhne. Přepínač umožňuje zapojit více jader, v tomto případě čtyři. Doba kompilace se v závislosti na výkonu stanice značně zkrátí. Bez toho může trvat i 30 minut.
3. Po úspěšné kompilaci je potřeba spustit příkaz `make install`
4. Ve složce `<opencv-source>/platforms/android_arm64/install/sdk` se nyní nachází SDK OpenCV knihovny podobné jako to, které lze stáhnout ze sourceforge. Oproti němu obsahuje pouze verzi knihovny pro `arm64-v8a`, jelikož byla zvolena v konfiguraci cmaku.
5. Doporučuji vytvořit build i pro ostatní architektury `armeabi-v7a`, `x86` a `x86_64`. Pro každou architekturu musí být zopakován celý proces od konfigurace (viz 6.4.1.1) znovu. Pro každou architekturu je nutné započít konfiguraci CMake v nové složce. Tím se předejde zbytečným chybám při linkování správných verzí knihoven.
   Nový build knihovny nesmí být prováděn v totožné složce s pouhou změny proměnné `ANDROID_ABI` v předchozí konfiguraci. CMake bude mít stále detekován procesor pro předchozí ABI a nalinkovány špatné knihovny. Silně proto doporučuji začít konfiguraci od začátku v nové složce.

Posledním krokem je vytvořit balíček obsahující verze knihovny pro všechny čtyři procesorové architektury. Ve složce `<opencv-source>/platforms` je potřeba vytvořit složku `android_sdk`. Do ní budou postupně zkopírována (bez nahrazování souboru, pouze přidání, těch které nejsou obsaženy) všechna SDK ze složek `install` ve vytvořených buildech knihovny.

Složka `<opencv-source>/plaftorms/android_sdk` by poté měla mít následující strukturu:

- složky etc a libcxx_helper
- soubor build.gradle
- složka `java` obsahující definice Java tříd knihovny.
- složka `native` se zkompilovanou knihovnou. V podsložce libs by se měly nacházet adresáře pro každou procesorovou architekturu:
  - `armeabi-v7a`
  - `arm64-v8a`
  - `x86`
  - `x86_64`

Nyní je k dispozici kompletní distribuci knihovny, kterou je možné nalinkovat do Android projektu.

6.4.1.3 Chyby při kompilaci

Při kompilaci může nastat řada problémů. Pokud se tak stane, je často vhodné začít ji od začátku. V některých případech to však nepomůže. Cílem této sekce je jmenovat některé z nich a popsat možné způsoby jejich odstranění.

Chybějící indexy ve statických knihovnách Při výskytu této chyby je pravděpodobné, že je použita nekompatibilní verze NDK. První způsob řešení je použít při konfiguraci jinou verzi a její toolchain. Tato chyba nastává při kompilaci pro architekturu `arm64-v8a`. Druhou možností je pomocí příkazu ranlib přidat indexy manuálně u souborů, které chybu způsobují. Ranlib přidává statickým knihovnám informace o jejich uložených object file souborech v podobě indexů, které jsou používány při kompilaci.

LLVM/Wrong format V průběhu kompilace statických knihoven může nastat chybová hláška:

```
Error adding symbols: File in wrong format
```

Nejčastěji má jednu ze 2 příčin:

- `cmake` se pokusil nalinkovat nekompatibilní verzi knihovny. Nejčastěji se stává, že se pokusí vzít knihovnu pro špatnou architekturu a místo `arm64-v8a` zkusí nalinkovat verzi knihovny pro `x86_64`

Příčinou může být opět použití nekompatibilní verze NDK. Řešením je opakování kompilace s jinou verzí. Pokud chyba přetrvává, je dalším možným řešením upravit konfiguraci a v cmake-gui zaškrtnout proměnnou BUILD_ZLIB. Tím dojde k vynucení použití verzí některých knihoven z repozitáře knihovny místo těch, které jsou uloženy v NDK. Kompilace musí být spuštěna od začátku.

Chybějící knihovny a soubory Při kompilaci pro 64-bitovou architekturu procesorů ARM může nastat řada chyb při kompilaci (nejčastěji právě u modulu v `perf_core`), kdy se cmake odvolává na neexistující knihovny či neplatné příkazy. Pokud tyto potíže nastanou, je potřeba doinstalovat do stanice `GNU Compiler aarch64`. To je balíček obsahující soubory, utility a doplňky sloužící ke cross-kompilaci na 64-bitovou architekturu procesorů ARM. Výčet těchto balíčkou je na obrázku 6.4.

Carotene Při kompilaci knihovny pro platformu x86 mohou nastat potíže s knihovnou `carotene`, která provádí různé optimalizace při sestavování aplikace. Řešením může být tyto optimalizace vypnout, za cenu pomalejšího sestavení. V cmake-gui k tomu slouží flag `WITH_CAROTENE`. V této situaci je potřeba začít kompilaci od začátku.

Obrázek 6.4: Knihovny pro cross-kompilaci na architekturu arm64-v8a (OS Linux)

6.4.2 Nalinkování OpenCV SDK do Android Studia

Pro tuto část musí mít uživatel k dispozici SDK OpenCV knihovny. Je nutné rozhodnout se, jestli bude používána Java nebo C++ verze OpenCV knihovny. V obou případech je první část návodu společná.
Budou použity tyto zástupné symboly:

- `<android-sdk>` – kořenový adresář Android SDK, stejný jako v předchozím návodu
- `<project-root>` – kořenový adresář projektu, do kterého budeme linkovat OpenCV knihovnu
- `<opencv-sdk>` – adresář `sdk` z android buildu (distribuce) OpenCV knihovny.

Postup přidání knihovny do projektu je následující:

1. Nejprve je potřeba v Android Studiu otveřít projekt, do kterého má být knihovna přidána. Nechť je jeho kořenový adresář `<project-root>`
2. `File -> New -> Import Module`
3. Jako source directory bude zvit složku `<opencv-sdk>/java` a pojmenovat ji OpenCV.
4. -> `Finnish`. Nyní je modul přidán do projektu, avšak s chybami, které je potřeba opravit ručně.
5. Je potřeba otevřít soubor `build.gradle` uvnitř nově přidaného OpenCV modulu.
6. Upravit následující řádek:

```
//apply plugin: 'com.android.application'
apply plugin: 'com.android.library'
```

7. Synchronizovat buildToolsVersion, compileSdkVersion, minSdkVersion a targetSdkVersion s hodnotami definovanými v `build.gradle` projektu
8. Modul je nyní importován do projektu.
9. Nyní je potřeba přidat na nově importovanou knihovny závislost. `File -> Project Structure -> Dependencies`. Zvolit modul `app` a v záložce Declared Dependencies je potřeba přidat modul OpenCV.
10. Zkopírovat nativní knihovny do projektu. V OpenCV SDK je možné nalézt složku `libs`, buď přímo ve složce `sdk`, nebo v podsložce `native`. Tu zkopírujeme do projektu do složky `app/src/main` a přejmenujeme ji na jniLibs.

11. Je potřeba zkontrolovat, že se v souboru `build.gradle modulu` ve složce `app` nachází následující struktura:

```
sourceSets {
   main {
      jniLibs.srcDirs = ['src/main/jniLibs']
   }
}
```

   Pokud ne, musí být přidána

Nyní je možné používat v aplikaci Java rozhraní OpenCV knihovny.

Android, OpenCV a C++

Pokud je cílem projektu používat knihovnu s C++, je kromě předchozích kroků nutné i vytvořit C++ subprojekt a správně jej nalinkovat. K tomu je potřeba provést následující kroky:

1. Otevřít projekt, do kterého má být knihovnu přidána. Nechť je jeho kořenový adresář `<project-root>`.

2. Z `<opencv-sdk>/native` zkopírovat složku libs do `<project-root>/app/src/main` a přejmenovat ji na `jniLibs`.

3. V `<project-root>/app/src/main/` vytvořit složku `jniIncludes`

4. Do této složky nakopírovat z `<opencv-sdk>/native/jni` složku `include`

5. Nyní se nabízí dva systémy pro sestavování C++ modulů:

   - CMake, který k sestavení používá soubory `CMakeLists.txt`
   - ndk-build, který používá soubory `Android.mk`

   Systémy se liší v jazyku, který je použit při sestavování. Systém ndk-build může být pro uživatele bez zkušeností složitější. CMake navíc poskytuje výhody, kdy mohou být C++ moduly projektu snadno přenositelné mezi platformami, tedy jejich sestavení může proběhnout v libovolném prostředí, a je možné celý modul nasdílet do jiného projektu bez potíží s kompatibilitou. Ndk-build používá pouze platforma Android a je nepřenositelný. Sekce se proto dále bude zabývat sestavením pomocí CMake, jelikož je univerzálnější.

6. V `<project-root>/app/src/main` založit složku s názvem cpp.

7. V `<project-root>/app/src/main` vytvořit soubor `CMakeLists.txt`

8. Do něj musí být přidána například tato konfigurace:

```
cmake_minimum_required(VERSION 3.4.1)

  #Nalinkování čhlaviek OpenCV knihovny.
  include_directories(
      ${CMAKE_CURRENT_SOURCE_DIR}/jniIncludes/include)
  add_library(lib_opencv SHARED IMPORTED )

  set_target_properties(lib_opencv PROPERTIES
      IMPORTED_LOCATION
      ${CMAKE_CURRENT_SOURCE_DIR}/jniLibs/${ANDROID_ABI}/libopencv_java4.so)

  add_library(nativelib SHARED <zdrojové kódy ěoddlené
      mezerou> )

  target_link_libraries(nativelib lib_opencv)
```

9. `File -> Link C++ Project with Gradle`. Jako build systém zvolit CMake a `Project path` nastavit na nově vytvořený `CMakeLists.txt`.

10. Do souboru `build.gradle` do sekce `android.defaultConfig` přidat toto nastavení:

```
externalNativeBuild {
   cmake {
      cppFlags "-frtti -fexceptions"
      abiFilters 'x86', 'x86_64', 'armeabi-v7a', 'arm64-v8a'
   }
}
```

Nyní je v projektu nalinkován C++ subprojekt sestavovaný pomocí systému CMake, ve kterém je možné psát algoritmy zpracování obrazu OpenCV knihovny v C++.

Kapitola 7

Testování

7.1 Porovnání rychlosti detekce

V knize Android Application Programming with OpenCV 3 autor v kapitole 6 tvrdí: *„Když Java/Python rozhraní OpenCV knihovny odešle funkční volání do její C++ implementace, nese sebou malou režiji. Pokud kód provádí těchto volání 1000 pro každý snímek, je potřeba tuto režiji zvážit“*[10]. V aplikaci Myšák 2 budou v novém modulu pro načítání pracovních listů čteny snímky z kamery. Každý z těchto snímků bude podroben detekci značek. Při použití Java rozhraní OpenCV knihovny to může oproti C++ znamenat zátěž navíc v podobě nativních volání. Toto testování se zabývá porovnáním detekce pozičních značek pomocí Java rozhraní OpenCV knihovny a C++ pomocí JNI.

7.1.1 Rozdíly v implementaci

V případě použití Java rozhraní je provedeno mnohem více nativních volání. V případě C++ je toto volání definována jedna nativní funkce, která odkazuje na svou C++ implementaci. Při čtení snímků z kamery je v případě C++ provedeno pouze jedno nativní volání a samotný algoritmus je napsán pomocí původních tříd implementace OpenCV knihovny. S použitím Java rozhraní je provedeno mnoho nativních volání, jelikož je celý algoritmus napsán pomocí Java wrapperů odkazujících na původní C++ implementaci.

Lze tak předpokládat, že detekce značek v C++ bude méně vytěžovat procesor a tedy bude rychlejší. Použití Kotlinu a Java rozhraní OpenCV knihovny však přináší výhody, které pramení z použití programovacího jazyka Kotlin. Navíc v tomto případě odpadá nutnost vytvářet C++ subprojekt a mapovat jej na implementaci v Javě (podobně jako je vytvořeno Java rozhraní OpenCV knihovny). Cílem tohoto testování je zjistit, zda dojde k očekávanému poklesu výkonu, a pokud ano, tak v jaké míře se projeví.

7.1.2 Měření

Metrikou testování pro porovnání 2 výše zmíněných možností bude průměrné FPS dosažené při detekci pozičních značek. V rámci tohoto měření byl vytvořen testovací demo projekt, který obsahuje implementaci obou způsobů detekce. Čtení z kamery je podobně jako v aplikaci Myšák 2 založeno na třídě `CameraBridgeViewBase`, která je součástí balíčku android, knihovny OpenCV. V průběhu čtení snímků z kamery bude do souboru v zařízení ukládáno průměrné zaznamenané FPS, ze kterého bude vyvozena výsledná statistika. Při testování byla použita zařízení `LG L90 d405n` a `Evolveo Strongphone G4`. Jedná se o starší zařízení z levnější cenové kategorie, a lze předpokládat dosažení výrazně lepších výsledků při použití novějšího hardware.
Test pro oba moduly sestával z následujících kroků:

1. Spuštění demo aplikace, která otevře kameru zařízení, čte a zpracovává snímky.
2. Namíření kamery na pracovní list s pozičními značkami.
3. Čtení snímků z kamery a průběžný záznam průměrného naměřeného FPS (každé 3 vteřiny) do logu po dobu 5 minut.
4. Ukončení čtení a vypnutí aplikace.
5. Zanesení výsledků do tabulky.

Pro srovnání byly naměřeny i hodnoty průměrného FPS při pouhém čtení snímků z kamery bez detekce značek. Data z testování je možné dohledat v přílohách a na adrese `https://tinyurl.com/MysakTestCV`

Shrnutí výsledků z testování zobrazuje tabulka 7.1. Výsledky ukazují průměrné FPS při použití různé implementace. Jsou zaokrouhleny na 2 desetinná místa. V případě tohoto testování, byla kamera po celou dobu čtení namířena přímo na pracovní list tak, aby zabírala všechny 4 značky, a tedy bylo prováděno co nejvíce možných operací. Při namíření fotoaparátu mimo pracovní list, lze očekávat mnohem menší pokles výkonu.

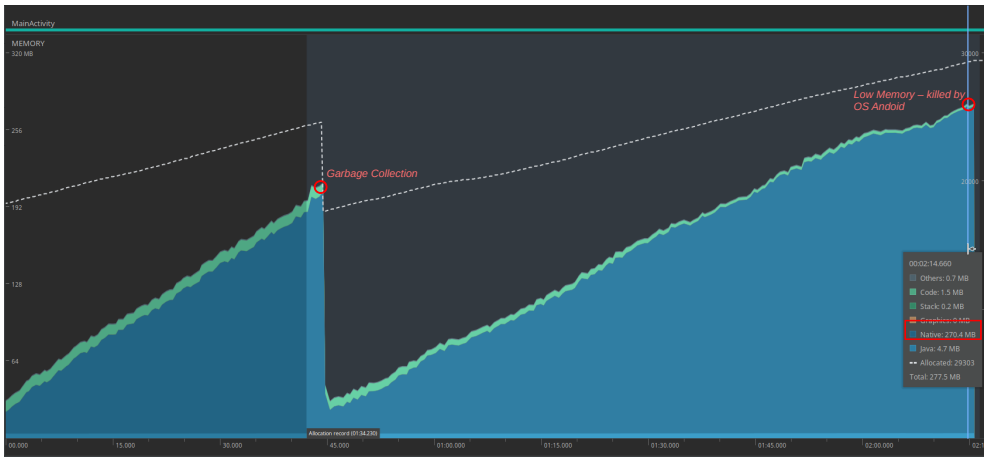
Zařízení	Rozlišení	Java	C++	Bez zpracování
Evolveo G4	800×600	9,96	10,02	17,92
LG L90	768×432	7,11	7,62	23,70

Tabulka 7.1: Vyhodnocení rychlosti čtení pozičních značek

Z výsledků vyplývá, že v Kotlinu sice dochází k poklesu výkonu, avšak rozdíl oproti C++ není příliš významný. Lze předpokládat, že vyšší počet jni volání v aplikaci nezpůsobí výrazný pokles výkonu.

7.1.3 Další poznatky z testování

Při implementaci jednoduchého prototypu určeného k tomuto testování došlo k několika potížím, které vedly k pádu aplikace a zkreslení výsledků při testování. Proto byla navíc průběžně měřena spotřeba paměti pomocí nástroje `Profiler`, který je součástí Android Studia. Při použití Java rozhraní OpenCV knihovny docházelo při čtení z kamery k zabití aplikace operačním systémem, kvůli spotřebě paměti. Docházelo k němu i přesto, že byly zváženy možné potíže se správou paměti, které byly probrány v sekci 4.4. Ukázku vývoje paměti v čase znázorňuje obrázek 7.1.



Obrázek 7.1: Vyčerpání paměti při detekci značek.

K vyčerpání paměti však docházelo i při použití C++ implementace, přestože byly všechny pomocné instance třídy Mat v obou případech uvolňovány. Je potřeba zdůraznit, že ani detekce v C++ se na Android zařízení neobejde bez Java rozhraní OpenCV knihovny, je-li pro čtení dat z kamery použita třída `CameraBridgeViewBase`. Ta je součástí Java rozhraní OpenCV pro Android. Problém nenastával v kódu algoritmu, ale již v Javě při získání snímku z kamery . Pomocí následující konstrukce byla nejprve vytvořena kopie vstupního snímku a poté byl zpracován nativní funkcí. Pro příklad je uveden pseudo kód čtění snímků z kamery:

```
override fun onCameraFrame
   (inputFrame: CameraBridgeViewBase.CvCameraViewFrame?): Mat {
   val tmp = Mat()
   inputFrame!!.rgba().copyTo(tmp)
    ...
    //process input frame
   ...
   return tmp
}
```

Přestože je vytvářena kopie vstupního snímku inputFrame, není potřeba jej manuálně uvolňovat, i když obsahuje ve své vnitřní struktuře instance třídy Mat. Toto uvolnění se odehrává ve vnitřní implementaci `CameraBridgeViewBase` po zobrazení snímku na obrazovku. Problematický je řádek:

```
val tmp = Mat()
```

Na pozadí se v C++ vytvoří nová instance a alokuje se pro ni paměť. Jedná se o instanci prázdné matice, avšak nějaký prostor v pamětí zabírá kvůli hlavičkám a vnitřním proměnným. Nárust paměti není tak velký jako při neuvolnění instance uchovávající data obrázku, ale kvůli zpracování několika snímků za vteřinu dojde po zhruba po 40 sekundách k zabití aplikace. Na obrázku 7.1 k němu došlo mnohem později, jelikož byl systémem v průběhu detekce spuštěn GarbageCollector. Tato situace se dá řešit přesunutím deklarace proměnných vně cyklu.

```
val tmp = Mat()
override fun onCameraFrame
    (inputFrame: CameraBridgeViewBase.CvCameraViewFrame?): Mat {
    inputFrame!!.rgba().copyTo(tmp)
    ...
    //process input frame
    ...
    return tmp
}
```

Všechny čtené snímky mají stejné rozměry, a proto dojde v každém cyklu pouze k překopírování dat bez vytváření nového paměťového místa.

7.2 Uživatelské testování

Výstupem této práce je prototyp aplikace Myšák 2, umožňující řešit dvě kategorie úloh a automaticky při tom načítat jejich pracovní listy. Tento prototyp byl v závěru této práce podroben akceptačnímu testování. Cílem tohoto testování bylo zjistit, jestli nový modul načítání značek pracuje správně, a jestli jeho přidání snížilo chybovost aplikace při řešení úloh. Materiály a dokumenty z tohoto testování jsou součástí příloh této bakalářské práce. Pro účely testování bylo vytvořeno celkem 5 úloh a jejich pracovních listů. Jeden pro první kategorii a zbylé čtyři pro kategorii druhou. Tyto nové úlohy jsou složitější než ty, které byly použity při testování původní verze aplikace. Ta používala pouze pracovní listy s jednou barevnou řadou čtverečků, a ani tu nebyla schopna správně ohodnotit. Nová verze obsahuje barevných řad několik. I to je potřeba zvážit při porovnání výsledků testování. Všechna data z testování je možné dohledat v přílohách, či na stránce `https://tinyurl.com/MysakTestUserNew`.

7.2.1 Uživatelé a podmínky

Předchozí testování bylo omezenější, neboť mu byli podrobeni pouze čtyři uživatelé. Testování nového prototypu se zůčastnilo celkem 10 uživatelů. Testování bylo opět provedeno v omezených podmínkách kvůli probíhající pandemii Covid-19 a do věkové skupiny uživatelů tak spadal pouze jediný. Přesto přineslo zajímavé poznatky a velmi cenné podněty pro budoucí zlepšení aplikace.

Uživatelé	Věk	Pohlaví	Povolání
Uživatel 1	neuvedeno	muž	programátor
Uživatel 2	6	muž	dítě
Uživatel 3	neuvedeno	muž	student
Uživatel 4	neuvedeno	žena	neuvedeno
Uživatel 5	neuvedeno	muž	programátor
Uživatel 6	neuvedeno	muž	programátor
Uživatel 7	43	žena	grafik
Uživatel 8	19	muž	student
Uživatel 9	69	žena	důchodce
Uživatel 10	72	muž	důchode

Testování bylo provedeno celkem na třech zařízeních. Uživatelé navíc kromě barevných fix používali i barevné pastelky a pastelky proggreso. Uživatelé používali vlastní barvičky, aby bylo simulováno domácí prostředí. Uživatelské rozhraní aplikace prototypu je nedostatečné a kvůli chybám kompatibility neobsahuje některé původní animace, které byly velmi důležité při odevzdání. Jedná se například o ukázkové čtení QR kódu a focení řešení. Modelu myšáka se navíc špatně mapují textury a není tak poznat, kdy mluví. Proto byl každý uživatel při průchodu aplikací v každém kroku, kromě načítání řešení, a vyhodnocení seznámem s tím, co má dělat. Přehled uživatelů, kteří se účastnili testování zobrazuje tabulka 7.2.1.

7.2.2 Načítání pracovních listů

Aplikace dokázala vždy správně načíst pracovní list. Vyskytlo se však několik potíží, jejichž příčinou je především nedostatečné uživatelské rozhraní, které uživatele neinformovalo o tom, co má dělat. Moderátor musel při testování často zasahovat a připomínat uživateli co má dělat. Někteří uživatelé však měli problém pochopit, že musí být při načítání úloh detekovány poziční značky. Nikdo z nich neznal jejich význam. Při prvním načítání proto museli být instruování o tom, že se musí v záběru vyskytovat všechny čtyři, aby mohl

být list načten. Aplikace měla na některých zařízeních potíže s načtením listu, pokud byl čten z příliš velkého úhlu a značky se téměř nevešly na snímek. Dětský uživatel má navíc zařízení v mnohem nižší poloze, než dospělý. Uživatel číslo 2 si musel při načítání pracovního listu stoupnout. S tím by měla aplikace počítat a myšák by měl dítě k tomuto kroku vyzvat.

Potíže se vyskytly i při čtení QR kódu. Ve třech případech aplikace nebyla schopna načítat QR kódy kvůli špatně spuštěnému fotoaparátu, který na čtení nebyl připraven. Tuto chybu způsobuje knihovna, která byla použita na čtení kódů. Někteří uživatelé se velmi ztráceli ve stavech, ve kterých se aplikace nacházela. Několikrát se stalo, že se uživatel snažil načíst řešení, a přitom byl ve stavu, kdy aplikace očekává načtení zadání, tedy QR kódu.

7.2.3 Výsledky a závěr z testování

Aplikace uživatele i v nové verzi hodnotila pouze způsobem povedlo/nepovedlo se. Uživatelské rozhraní nové verze je minimálně upraveno oproti původní verzi a kvůli časové tísni nebyla do aplikace přidána možnost zobrazení správného řešení. Výsledky z testování znázorňuje tabulka 7.2.3.

Uživatel		Úloha1	Úloha2.1	Úloha2.2	Úloha2.3	Úloha2.4	Celkem	Úspěšnost uživatelů
Počet odevzdání	*Uživatel 1*	0	0	0	0	3	3	-
	Uživatel 2	0	0	0	0	1	1	-
	Uživatel 3	0	0	0	0	1	1	-
	Uživatel 4	0	0	0	0	1	1	-
	Uživatel 5	0	0	0	0	1	1	-
	Uživatel 6	0	0	0	0	1	1	-
	Uživatel 7	2	2	6	1	1	12	-
	Uživatel 8	1	1	3	1	2	8	-
	Uživatel 9	1	1	1	1	1	5	-
	Uživatel 10	1	1	1	1	1	5	-
Počet správných řešení	*Uživatel 1*	0	0	0	0	2	2	67%
	Uživatel 2	0	0	0	0	1	1	100%
	Uživatel 3	0	0	0	0	1	1	100%
	Uživatel 4	0	0	0	0	1	1	100%
	Uživatel 5	0	0	0	0	1	1	100%
	Uživatel 6	0	0	0	0	1	1	100%
	Uživatel 7	2	2	6	1	1	12	100%
	Uživatel 8	1	1	3	1	2	8	100%
	Uživatel 9	1	1	1	1	1	5	100%
	Uživatel 10	1	1	1	1	1	5	100%
Počet úspěšných hodnocení aplikace	*Uživatel 1*	0	0	0	0	2	2	67%
	Uživatel 2	0	0	0	0	0	0	0%
	Uživatel 3	0	0	0	0	0	0	0%
	Uživatel 4	0	0	0	0	0	0	0%
	Uživatel 5	0	0	0	0	0	0	0%
	Uživatel 6	0	0	0	0	0	0	0%
	Uživatel 7	0	2	4	1	1	8	67%
	Uživatel 8	1	0	0	1	2	4	50%
	Uživatel 9	0	1	0	0	0	1	20%
	Uživatel 10	0	0	0	0	0	0	0%
Úspěšnost v nalezení řešení		100%	100%	100%	100%	92%	**97%**	
Úspěšnost aplikace při hodnocení		20%	60%	36%	50%	38%	**39%**	

Úspěšnost aplikace při hodnocení popisuje, jestli byla schopna správně hodnotit uživatelovo řešení. Tato úspěšnost se zvedla oproti původnímu testování z 3% na 39%. Tento výsledek je však stále není dostatečný. Modul odstranil některé příčiny neúspěšného hodnocení. Mnoho je jich však stále potřeba odstranit.

Příčiny neúspěchu

Světelné podmínky Hlavní přičinou neúspěchu byla kvalita osvětlení. Uživatelé, kteří vypracovávali úkoly při denním světle měli vyšší úspěšnost než ti, kteří úlohy řešili při umělém osvětlení. Umělé osvětlení způsobovalo významné potíže při detekci žluté barvy, která byla často zahozena jako šum.

Odstíny barev Oproti testování původní verze byly na pracovní listy navíc přidány i fialové čtverečky. Uživatelé často místo fialové použili tmavě modrou nebo růžovou. Odstín barvy nikdy neodpovídal barevným čtverečkům na pracovním listu. Uživatelé často použili hraniční odstín barev, který se špatně interpretoval.

Použité prostředky Při hodnocení záleželo velmi i na tom, jestli uživatel používá fixy nebo pastelky. Většina uživatelů používala pastelky, na které světelné podmínky doléhaly ještě významněji než na fixy. Někteří uživatelé navíc vybarvili řešení tak slabě, že algoritmus na pracovním listu téměř nic nerozpoznal

Návrh nových změn

V dalších verzích aplikace bude vhodné provést následující změny, který by opět měly zvýšit úspěšnost aplikace v hodnocení uživatelských řešení.

Světelné podmínky Je potřeba provést v načteném snímku automatickou korekci světla, či spustit při načítání blesk

Odstíny barev Je potřeba doladit správně hranice jednotlivých barev

Komunikace Aplikace musí s uživatelem lépe komunikovat, aby k jejímu používání nebyla nutna přítomnost poučené osoby

Animace Je nutné navrátit do aplikace názorné animace načítání QR kódu a pracovních listů. Někteří uživatelé si nepředstavili, co mají v daný okamžik udělat.

Závěr

Cílem této práce bylo analyzovat nedostatky aplikace Myšák 2, navrhnout a implementovat novou verzi aplikace. Tuto aplikaci podrobit testování a porovnat výsledky s předchozí verzí. Cíle práce byly splněny. Aplikace nyní disponuje novým jádrem postaveným na frameworku LibGDX. Byl zachován původní průchod aplikací a systém hodnocení úloh byl přepsán a rozšířen o funkce, které značně usnadní debugování a vývoj nových úloh. Aplikace díky novému systému načítání pracovních listů dosahuje mnohem menší chybovosti při ohodnocení uživatelských řešení. Uživatelské testování prokázalo zvýšení úspěšnosti při hodnoceni. K nasazení aplikace však bude potřeba provést ještě mnoho úprav. Nejhlavnější z nich se týkají uživatelského rozhraní a odstranění negativních aspektů osvětlení při hodnocení úkolů.Dále bude nutné předělat model myšáka tak, aby měl správně namapované textury. Vhodné by bylo též rozšířit jeho animace a vymodelovat mu vhodné prostředí, ve kterém by se pohyboval. Až budou odstraněny nedostatky vyhodnocování úloh a aplikace bude disponovat dostatečným uživatelským rozhraním, bude vhodná k nasazení a otevření pro uživatelské beta testování. Pro budoucí rozšiřování aplikace o nové úkoly bude vhodné vytvořit editor pracovních listů, který poskytne šablonu s vyznačenou sekcí kresby a osadí pracovní list pozičními značkami, aby byl vzhled listů sjednocen.

Tato práce ukázala, jak je aplikace Myšák 2 specifická a jak velký rozsah má. Specifická je především svou cílovou uživatelskou skupinou, díky které je kladena velká zodpovědnost na vývojáře. Při návrhu nových řešení bylo důležité vcítit se do role předškoláka a mít stále na mysli, co vše dovede, a co už je za hranicí jeho schopností. Zanedbání tohoto vcítění lze považovat za příčinu některých nedostatků předchozího projektu, na který tato práce navazovala.

Ve vývoji aplikace bych chtěl nadále pokračovat. Zaměřil bych se na algoritmy rozpoznání obrazu a design herního prostředí. Po nasazení první release verze bych aplikaci rád rozšířil i na platformu iOS.

Literatura

[1] Wong, M. Y.; Zhou, Y.; Xu, H.: Big Data in Fashion Industry: Color Cycle Mining from Runway Data. In *AMCIS*, 2016.

[2] Shabalina, K.; Sagitov, A.; Sabirova, L.; aj.: *ARTag, AprilTag and CALTag Fiducial Systems Comparison in a Presence of Partial Rotation: Manual and Automated Approaches*. 01 2020, ISBN 978-3-030-11291-2, s. 536–558, doi:10.1007/978-3-030-11292-9_27.

[3] Pekárková, S.; Baborová, J.: *Grafomotorické dovednosti*. Rozvoj dítěte, Pedagogicko-psychologická poradna STEP s.r.o., 2018, ISBN 978-80-907072-1-4. Dostupné z: `https://books.google.cz/books?id=byBUuwEACAAJ`

[4] Crombez, Emmanuel: Graphisme Maternelle. [online], [cit. 2020-05-02]. Dostupné z: `https://apps.apple.com/us/app/graphismematernelle/id413501532`

[5] Kříž, P.; Šulc, M.; Kop, M.; aj.: 2. Závěrečná zpráva softwarového projektu Myšák2. [online], [cit. 2020-05-02]. Dostupné z: `https://docs.google.com/document/d/1EewedEJu1O1BqwTHE6dez8g173SfO52KwJb-UrFBfOg`

[6] Lorenz, M.; Rudolph, J.-P.; Hesse, G.; aj.: Object-Relational Mapping Revisited - A Quantitative Study on the Impact of Database Technology on O/R Mapping Strategies. In *HICSS*, 2017.

[7] Narayan, Satya: Sugar ORM. Dostupné z: `https://satyan.github.io/sugar/`

[8] Bradski, G.: The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[9] Kříž, P.; Šulc, M.; Kop, M.; aj.: 3. Závěrečná zpráva softwarového projektu Myšák2. [online], [cit. 2020-05-02]. Dostupné z: https://docs.google.com/document/d/1CGXCFzBxEenaq6MeOKtzb6oa6iphAz7SsbDDEkPTdvo/edit?usp=sharing

[10] Howse, J.: *Android Application Programming with OpenCV 3*. Community experience distilled, Packt Publishing, 2015, ISBN 9781785285387. Dostupné z: https://books.google.cz/books/about/Android_Application_Programming_with_Ope.html?id=hb8FCgAAQBAJ&redir_esc=y

[11] OpenCV: *The OpenCV Reference Manual*. čtvrté vydání, 2019.

[12] Žára, J.; Beneš, B.; Sochor, J.; aj.: *Moderní počítačová grafika*. Computer Press, druhé vydání, 2004, ISBN 9788025104545. Dostupné z: https://books.google.cz/books?id=USQnAAAACAAJ

[13] Kříž, P.; Šulc, M.; Kop, M.; aj.: Výuková aplikace pro rozvoj grafomotoriky Myšák2. Dostupné z: https://gitlab.fit.cvut.cz/mysak/mysak_2

[14] Jetbrains, s.r.o.: Jetbrains: Kotlin/Native. [online], [cit. 2020-05-06]. Dostupné z: https://kotlinlang.org/docs/reference/native-overview.html

[15] Jetbrains, s.r.o.: Jetbrains: Kotlin programming language. [online], [cit. 2020-05-06]. Dostupné z: https://kotlinlang.org/

[16] Jetbrains, s.r.o.: Jetbrains: Kotlin and Java comparison. [online], [cit. 2020-05-06]. Dostupné z: https://kotlinlang.org/docs/reference/comparison-to-java.html

[17] Chandel, H.; Vatta, S.: Occlusion Detection and Handling: A Review. *International Journal of Computer Applications*, ročník 120, 06 2015: s. 33–38, doi:10.5120/21264-3857.

[18] Garrido-Jurado, S.; Muñoz-Salinas, R.; Madrid-Cuevas, F.; aj.: Generation of fiducial marker dictionaries using Mixed Integer Linear Programming. *Pattern Recognition*, ročník 51, 10 2015, doi:10.1016/j.patcog.2015.09.023.

[19] Romero-Ramirez, F.; Muñoz-Salinas, R.; Medina-Carnicer, R.: Speeded Up Detection of Squared Fiducial Markers. *Image and Vision Computing*, ročník 76, 06 2018, doi:10.1016/j.imavis.2018.05.004.

[20] Garrido-Jurado, Sergio and Muñoz-Salinas, Rafael and Madrid-Cuevas, Francisco and Medina-Carnicer: ArUco library. [online], [cit. 2020-05-03]. Dostupné z: https://www.uco.es/investiga/grupos/ava/node/26

[21] OpenCV: Extra modules for OpenCV library. [online], [cit. 2020-05-03]. Dostupné z: https://github.com/opencv/opencv_contrib

[22] Lamb, Philip and Looser, Julian and Grasset, Raphael and Pintaric, Thomas and Woessner, Uwe and Piekarski, Wayne and Seichter, Hartmut: ARToolKit: Software library for building AR applications. [online], [cit. 2020-05-03]. Dostupné z: http://www.hitl.washington.edu/artoolkit/

[23] Realmax, Inc.: ARToolKitX. [online], [cit. 2020-05-03]. Dostupné z: http://www.artoolkitx.org/

[24] CHILI Lab: Chillitags: Fiducial Markers for Augmented Reality And Robotics. [online], [cit. 2020-05-03]. Dostupné z: https://www.epfl.ch/labs/chili/dissemination/software/

[25] Kaltenbrunner, Martin and Bencina, Ross: ReacTIVision: Toolkit for tangible multi-touch surfaces. [online], [cit. 2020-05-03]. Dostupné z: http://reactivision.sourceforge.net/

[26] Google, Inc.: AR Core. [online], [cit. 2020-05-03]. Dostupné z: https://developers.google.com/ar/discover

[27] Stefanova, Cristina: The Best 15 Mobile Game Engines / Development Platforms and Tools in 2020. [online], [cit. 2020-05-04]. Dostupné z: https://thetool.io/2018/mobile-game-development-platforms

[28] Khronos Group, Inc.: OpenGL. [online], [cit. 2020-05-04]. Dostupné z: https://www.khronos.org/opengl/

[29] Google, Inc.: OpenGL in Android applications. [online], [cit. 2020-05-04]. Dostupné z: https://developer.android.com/guide/topics/graphics/opengl

[30] Khronos Group, Inc.: OpenGL ES (Embedded System). [online], [cit. 2020-05-04]. Dostupné z: https://www.khronos.org/opengles/

[31] Chukong Technologies: Cocos 2d-x. [online], [cit. 2020-05-04]. Dostupné z: https://www.cocos.com/en/products#Cocos2d-x

[32] YAJI Software: Cocos 2d. [online], [cit. 2020-05-04]. Dostupné z: http://cocos2d.org/

[33] YAJI Software: Cocos Creator. [online], [cit. 2020-05-04]. Dostupné z: https://www.cocos.com/en/creator

[34] Tsakpinis, Ioannis and Burjack, Kai and Linhart, Leon and Matzon, Brian and Ludwig, Daniel and Harsha Chilakapati, Sri: LWJGL. [online], [cit. 2020-05-04]. Dostupné z: https://www.lwjgl.org/

[35] Hansen, Normen and Speed, Paul and Bouquet, Rémy and Vainer, Kirill and Seinturier, Julien and others: jMonkey game engine. [online], [cit. 2020-05-04]. Dostupné z: https://jmonkeyengine.org

[36] Baratynskiy, Alexander: Kotlin plugin for NetBeans IDE. [online], [cit. 2020-05-04]. Dostupné z: http://plugins.netbeans.org/plugin/68590/kotlin

[37] Code Project: *How to Use OpenCV with Java under NetBeans IDE [online]*. [cit. 2020-05-03]. Dostupné z: https://www.codeproject.com/Tips/717283/How-to-Use-OpenCV-with-Java-under-NetBeans-IDE

[38] Unity Technologies: Unity. [online], [cit. 2020-05-04]. Dostupné z: https://unity.com/our-company

[39] Unity Technologies: *Using C++ OpenCV within Unity [online]*. [cit. 2020-05-03]. Dostupné z: https://www.codeproject.com/Tips/717283/How-to-Use-OpenCV-with-Java-under-NetBeans-IDE

[40] Medium: *SQLite and Unity: How to do it right [online]*. [cit. 2020-05-03]. Dostupné z: https://www.codeproject.com/Tips/717283/How-to-Use-OpenCV-with-Java-under-NetBeans-IDE

[41] Zechner, Mario and Sweet, Nathan and Shapcott, Justin and others : LibGDX: Game development framework. [online], [cit. 2020-05-04]. Dostupné z: https://libgdx.badlogicgames.com/

[42] LibGDX community: LibGDX: Tutorials and Codesamples. [online], [cit. 2020-05-04]. Dostupné z: https://libgdx.info/

[43] Catto, Erin: Box2D: A 2D Physics Engine for Games. [online], [cit. 2020-05-04]. Dostupné z: https://box2d.org/

[44] Zechner, Mario: LibGDX: RoboVM alternatives for iOS deployment. *LibGDX - wordpress [online]*, duben 2016, [cit. 2020-05-04]. Dostupné z: https://www.badlogicgames.com/wordpress/?p=3925

[45] Maurer, Uwe and Vogelzang, Mathijs : Android: Database libraries market share. [online], [cit. 2020-12-05]. Dostupné z: https://www.appbrain.com/stats/libraries/tag/database/database-libraries

[46] Junginger, Markus: GreenDAO: Android ORM for SQLite database. [online], [cit. 2020-05-04]. Dostupné z: https://greenrobot.org/greendao/

[47] Google, Inc.: Room: Persistence library for SQLite database. [online], [cit. 2020-05-04]. Dostupné z: https://developer.android.com/topic/libraries/architecture/room

[48] Google, Inc.: Mapping relations in Room database. [online], [cit. 2020-05-04]. Dostupné z: https://developer.android.com/training/data-storage/room/referencing-data#understand-no-object-references

[49] Watson, Gray: OrmLite: Lightweight ORM Java Package. [online], [cit. 2020-05-04]. Dostupné z: https://ormlite.com/

[50] Cash App: SQLDelight: Generate typesafe APIs from SQL statements. [online], [cit. 2020-05-04]. Dostupné z: https://cashapp.github.io/sqldelight/

[51] Stigsen, Alexander and Christiansen, Bjarne: Realm: Object-oriented alternative to SQLite database. [online], [cit. 2020-05-04]. Dostupné z: https://realm.io/products/realm-database/

[52] Google, Inc.: ART: Application runtime envirnoment. [online], [cit. 2020-05-03]. Dostupné z: https://developers.google.com/ar/discover

[53] Google, Inc.: Google I/O (2017): How to Manage Native C++ Memory in Android. Dostupné z: https://www.youtube.com/watch?v=7_caITSjk1k

[54] Google, Inc.: Android: Garbage Collection Performance. [online], [cit. 2020-05-06]. Dostupné z: https://docs.oracle.com/javase/9/gctuning/factors-affecting-garbage-collection-performance.htm#JSGCT-GUID-5508674B-F32D-4B02-9002-D0D8C7CDDC75

[55] Google, Inc.: Google I/O (2018): Understanding Android memory usage. Dostupné z: https://www.youtube.com/watch?v=7_caITSjk1k

[56] Google, Inc.: Android: Processes and Application Lifecycle. [online], [cit. 2020-05-06]. Dostupné z: https://developer.android.com/guide/components/activities/process-lifecycle

[57] OpenCV: OpenCV: Geometric Image Transformations. [online], [cit. 2020-05-12]. Dostupné z: https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html

[58] Slabý, O.: *Věnná města českých královen – Jádro mobilního klienta*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 05 2020.

[59] Google, Inc.: Android Developers: Support 64-bit architectures. [online], [cit. 2020-05-10]. Dostupné z: https://developer.android.com/distribute/best-practices/develop/64-bit

Příloha A

Seznam použitých zkratek

UI User Interface

GPU Graphical Processing Unit

ORM Object-Relational Mapping

SDK Software Development Kit

API Application Program Interface

IDE Integrated Development Environment

JVM Java Virtual Machine

ART Android Runtime

ABI Application Binary Interface

GC Garbage Collector

FPS Frames per second

JNI Java Native Interface

CRUD Create Read Update Delete

BI-SP1/2 předmět Softwarový týmový projekt 1 a 2 (rok absolvování 2019/2020)

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS