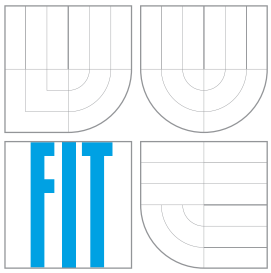
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MOBILNÍ APLIKACE PRO ANOTACE OBRÁZKŮ

MOBILE APPLICATION FOR IMAGE ANNOTATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PAVEL HŘEBÍČEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN BREJCHA
SUPERVISOR

BRNO 2016

Zadání bakalářské práce/18834/2015/xhrebi04

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Hřebíček Pavel**
Obor: Informační technologie
Téma: **Mobilní aplikace pro anotace obrázků**
Mobile Application for Image Annotation
Kategorie: Uživatelská rozhraní

Pokyny:

1. Proveďte rešerši existujících aplikací pro kreslení a anotaci obrázků.
2. Seznamte se s technologiemi, které budete potřebovat pro vývoj - např. framework QT.
3. Navrhněte architekturu a uživatelské rozhraní aplikace.
4. Implementujte mobilní aplikaci umožňující uživateli snadno nahrát a anotovat obrázky. Zaměřte se na co nejjednodušší ovladatelnost, optimalizujte čas potřebný k přepínání mezi jednotlivými obrázky.
5. Proveďte uživatelské testování.
6. Diskutujte možnosti dalšího vývoje.

Literatura:

- UNGER, Russ; CHANDLER, Carolyn. *A Project Guide to UX Design: For user experience designers in the field or in the making*. New Riders, 2012.
- RUMBAUGH, James; JACOBSON, Ivar; BOOCH, Grady. *Unified Modeling Language Reference Manual, The*. Pearson Higher Education, 2004.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3, rozpracování bodu 4.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Brejcha Jan, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce se zabývá tvorbou multiplatformní mobilní aplikace pro anotaci obrázků. Teoretická část popisuje specifikaci aplikace, jak z pohledu designu, tak z pohledu funkcionality. V praktické části je rozebrána samotná implementace aplikace od použitých technologií, přes architekturu aplikace, až po implementaci samotných nástrojů a funkcí. Pro praktickou část této práce byl použit framework Qt, který je založen na jazyku C++. Dále bylo potřeba seznámit se s problematikou UX Designu a Usability testingu.

Abstract

This thesis deals with a topic of creation of cross-platform mobile application for image annotation. The theoretical part of my thesis describes the specification of the application both in terms of design and functionality. The practical part discusses the actual implementation of the application from the technologies used through the application architecture to implementation of the tools and functions themselves. For the practical part of this thesis was used the Qt framework, which is based on the C++ language. It was also necessary to become familiar with the issues of UX Design and Usability testing.

Klíčová slova

Mobilní aplikace, anotace, anotace obrázků, anotační nástroj, uživatelské testování, UX Design, Qt, C++.

Keywords

Mobile application, annotation, annotation of images, annotation tool, usability testing, UX Design, Qt, C++.

Citace

HŘEBÍČEK, Pavel. *Mobilní aplikace pro anotace obrázků.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Brejcha Jan.

Mobilní aplikace pro anotace obrázků

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jana Brejchy. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Pavel Hřebíček
16. května 2016

Poděkování

Zde bych rád poděkoval svému vedoucímu práce panu Ing. Janu Brejchovi za odborné vedení, čas, ochotu a cenné rady při tvorbě této práce. Dále bych chtěl poděkovat Ing. Bronislavu Přibylovi a Ing. Kamilu Behúňovi, kteří se podíleli na testování aplikace a dali mi zpětnou vazbu pro vylepšení této práce.

Obsah

Kapitola 1

Úvod

V dnešní době se aplikace stávají neodmyslitelnou součástí každého chytrého mobilního telefonu, či tabletu. Lidé potřebují mít svá data neustále pod kontrolou, chtějí interagovat se svým okolím a vnímat situace, které se kolem nich dějí v reálném čase. Exisuje mnoho druhů aplikací, z nichž každá má jiný účel použití a cílí na odlišnou věkovou skupinu lidí. Stále častěji se do popředí dostávají aplikace, které slouží k samotné práci nebo výzkumu. Zařízení jako mobilní telefon, či tablet, se do popředí dostávají hlavně kvůli jejich přenositelnosti. Nesmíme ale opomenout další důležité faktory. Mnoho aplikací má na těchto zařízeních přirozenější ovládání pro samotného člověka. Příkladem může být malování. Když chceme malovat na počítači nebo notebooku, musíme k tomu použít myš – nejedná-li se o dotykový displej. Pokud malujeme na mobilním zařízení nebo tabletu, dotýkáme se plochy zařízení, jako bychom měli v ruce štětec a malovali na samotný papír.

Naším cílem je vytvořit multiplatformní mobilní aplikaci pro anotaci obrázků. Slovo anotace má mnoho významů a použití. V této práci je anotace použita ve smyslu označení segmentů nebo objektů na obrázku. Chceme, aby aplikace umožnila uživateli nahrát, anotovat a následně uložit obrázky. Obrázky chceme uložit jak v originální podobě, tak v podobě bitmapy. Aplikace cílí na uživatele, kteří se zabývají geolokalizací v přírodě nebo na uživatele, kteří chtějí označit nějaký objekt nebo segment. Uživatelské rozhraní a použitelnost aplikace byla otestována pomocí metody Usability testing.

Kapitola 2 shrnuje přehled vyhledaných informací o základních metodách a postupech, které byly v této práci použity. Dále byla provedena rešerše. V poslední části této kapitoly jsou porovnány aplikace, které jsou již na trhu, jak z pohledu designu, tak z pohledu funkcionality. Jsou zde zmíněny klady a zápory, které byly objeveny při pozorování těchto aplikací. Kapitola 3 se zabývá vymezením pojmů multiplatformní mobilní anotační aplikace, je zde definována cílová skupina, a jsou zde uvedeny požadavky na samotnou aplikaci. Kapitola 4 popisuje návrh a implementaci designu aplikace. Návrh popisuje vytvoření, otestování, vyhodnocení a následnou úpravu designu na základě zpětné vazby od testovaných uživatelů. Sekce implementace popisuje jakými nástroji byl design vytvořen. Poslední část se zabývá srovnáním, jak se povedlo návrh promítnout do výsledné aplikace. Kapitola 5 popisuje samotnou implementační část. Jsou zde uvedeny nástroje, které byly použity a programovací jazyky, které bylo potřeba nastudovat. Je zde ukázáno, jak vypadá celá architektura aplikace od základní logiky až po samotnou implementaci jednotlivých nástrojů. Kapitola 6 popisuje, jak probíhalo testování aplikace po jejím dokončení, jaké chyby se při testování našly, a jak byly odstraněny. Kapitola 7 shrnuje poznatky o celém díle. Dále se zaměřuje na to, v čem by mohla být aplikace v budoucnu rozšířena a vylepšena.

Kapitola 2

Rešerše

V této kapitole jsou stručně shrnuty základní metody a postupy, které jsou v této práci použity. Dále je zde uveden přehled výzkumné činnosti, který byl pro tyto metody a postupy publikován. U každého výzkumu jsou uvedeny výsledky a poznatky, které výzkum přinesl. V poslední části této kapitoly jsou zmíněny a porovnány již existující aplikace.

2.1 Související výzkum

Anotace

Tento pojem je poměrně široký a vyskytuje se v mnoha různých odvětvích. Samotné slovo anotace je latinského původu, a dalo by se charakterizovat jako připojování poznámek k textu, poznámka, vysvětlivka k textu, jak je zmíněno v sekci 3.1.

Morgan Ames a Mor Naaman se ve svém článku [1] zabývají otázkou, jakou má anotace motivaci pro lidi. Autoři tohoto článku píší, že uživatelé se anotování z počátku vyhýbali, protože neměli motivaci nic označovat, a to jak na stolních počítačích, tak na mobilních zařízeních. Změna podle autorů toho článku přišla až s nástupem služby Flickr. Flickr je služba pro sdílení fotografií. Díky tzv. „tagům“ – jedná se o formu anotace, se daly vyhledávat nejen uživatelé, ale i např. další fotografie. Autoři článku [16] tvrdí, že v dnešní době díky moderním digitálním technologiím vzniká velká spousta souborů, zejména digitálních fotografií. Díky tomu, že obrázků stále přibývá, je jejich nalezení stále složitější. Autoři tvrdí, že jednou z variant, jak tomuto zabránit, je uchovávat – nebo-li označit tyto obrázky pomocí tzv. „metadat“, které nám v jejich hledání pomohou. Tyto příklady využití anotací, které jsme uvedli výše, tvoří jen velmi malou část odvětví, kde se dá anotace využít. Anotace má mnohem širší využití. Anotovat lze dopravní značky na obrázku, mapu, horizont pohoří, či označit nějaký segment nebo objekt. Díky anotaci tak můžeme např. rozpoznat různé objekty – jak podle tvaru, barvy, významu, nebo rozlišit a označit polohu objektů. Když jsme ve škole na přednášce a podrtháváme si důležité části výkladu přednášejícího, ke kterým si děláme vlastní poznámky, jde taky o určitou formu anotace. Právě touto formou anotace se zabývají autoři článku [3], kteří představili aplikaci u-Annotate. Tato aplikace slouží pro psaní poznámek e-learningového obsahu. Umožňuje uživateli aktivně komentovat obsah, anotovaný obsah uložit, nebo ho exportovat pro sdílení s ostatními uživateli.

Tato práce je zaměřena na anotaci objektů nebo segmentů. Právě anotací reálných objektů se zabývali autoři článku [13]. Jako reálný objekt pro anotování si autoři tohoto článku vybrali automobilový motor. Na vyfocený snímek automobilového motoru si autoři „vygenerovali“ model motoru vytvořený programem AutoCAD. Tento model „přiložili“

na originální snímek. Ke každé anotované komponentě motoru přidali alfanumerický popisek samotné komponenty. Samostatný anotační soubor obsahoval právě tyto popisky a jejich umístění na obrázku, aby se rozpoznalo, kam se mají vygenerovat. Velmi podobný princip je použit i v této práci. Uživateli se uloží jak originální obrázek, tak anotovaná bitmapa – jedná se o „masku“, která určuje dané segmenty. Po „přiložení“ bitmapy na originální obrázek vznikne anotovaný obrázek.

Usability testing

Tento způsob testování byl v této práci použit jak pro otestování mockupů 4.2.2, tak pro otestování výsledné aplikace 6.

Usability testing se dá přeložit jako uživatelské testování. Jedná se o jednu z metod pro testování softwaru. Základy a principy tohoto způsobu testování byly položeny v knize [4] od Ericssona a Simona. Anne Kaikkonen ve svém článku [6] píše, že Usability testing je založen na hlasitém přemýšlení respondentů. Respondenti dostanou úkoly, u kterých při jejich vykonávání přemýšlí nahlas. To nám dává informace o tom, jak uživatel myslí, a upozorňuje nás to na funkce a procesy, které máme zlepšit. Na základě zpětné vazby od respondentů si dané problémy sepíšeme. Ke každému nalezenému problému si napíšeme, jak je problém závažný. Závažnost problému je většinou uvedena pomocí číselné stupnice, kde závažnější problémy představují vyšší čísla. Všechny nalezené problémy potom řešíme od těch nejzávažnějších, po ty méně závažné a důležité. Detailní popis této metody, který byl použit v této práci, se všemi získanými informacemi je popsán v podsekci 4.2.2.

Autoři článku [6] provedli výzkum, který se zabývá vlivem prostředí na uživatelské testování. Tento výzkum reagoval na vyjádření výzkumníků, kteří byli znepokojeni tím, že se testování provádí v klidném prostředí, a nesimuluje prostředí, kde jsou používány mobilní zařízení. Z tohoto důvodu rozdělili prostředí na tzv. **Laboratory** – „laboratoře“ a **Field** – „terén“. Do prostředí „laboratoří“ spadají například obývací pokoje nebo kanceláře. Jedná se o klidný prostor, kde se uživatel může na vykonávání úkolů plně soustředit. Prostředí „terén“ je zcela opačné. Patří sem například silnice, ulice, metro. Jedná se o hlučné prostředí, ve kterém lidé mobilní zařízení používají. Testovanou aplikací byla aplikace Mobile Wire, která se zabývá přenosem souborů mezi počítačem a „handsetem“. Testování probíhalo ve dvou skupinách po 20 respondentech. Respondentům bylo mezi 22 a 35 lety. Rozdíly ve zkušenostech s používaním mobilního telefonu byly minimální. Jedna skupina vykonávala uživatelské testování v „laboratořích“, druhá na „poli“, což představovalo město Helsinky, kde respondenti chodili, seděli nebo vykonávali činnost, kterou by normálně dělali. Testování vedli 4 moderátoři. Každý z těchto moderátorů měl s uživatelským testováním zkušenosti 5 až 13 let. Výsledky a poznatky tohoto výzkumu byly následující:

1. Pro testování mobilních aplikací je výhodnější klidné místo – „laboratoř“, protože je to časově výhodnější oproti testování v terénu. Testování v terénu vyžaduje dvojnásobek doby ve srovnání s „laboratoří“, tudíž zde můžeme vyzkoušet o polovinu méně testů za stejnou časovou jednotku.

2. Pro testování v terénu existuje mnoho detailů, které se mohou pokazit. Z tohoto důvodu je potřeba provést pilotní test před samotným testováním v terénu a ujistit se, že vše funguje správně.

3. Při testování v terénu musíme být připraveni, že věci nepůjdou podle plánu, a může dojít k neočekávané události mnohem častěji než v „laboratoři“.

Dongsong Zhang and Boonlit Adipat ve svém článku [17] tvrdí, že s rychlým pokrokem v oblasti mobilních technologií a aplikací se uživatelské testování stává stále důležitější součástí pro návrh, vývoj a nasazení úspěšných mobilních aplikací.

User Experience Design

User Experience (UX) Design má mnoho různých forem a definic od různých autorů a neexistuje žádná obecně platná definice. Spencer Lanoue ve svém článku [9] tvrdí, že UX Design je pojem, který má mnoho rozměrů a zahrnuje spoustu různých disciplín, jako je návrh interakcí, informační architekturu, vizuální design, použitelnost a interakce člověka s počítačem. Russ Unger a Carolyn Chandler ve své knize [15] píší, že UX Design je nedílnou součástí k tomu, aby byla aplikace úspěšná. Dále tvrdí, že rozsah tohoto pojmu je velký a stále se rozšiřuje. UX Design bere v úvahu obchodní cíle, potřeby uživatelů, kteří produkt používají, a jakákoliv omezení, která budou mít vliv na životaschopnost produktů.

2.2 Existující řešení

V následujících odstavcích jsou porovnány anotační aplikace a aplikace zaměřené na kreslení, které jsou již na trhu. Tyto aplikace jsou porovnány jak z pohledu designu, tak z pohledu funkcionality. Jsou zde zmíněny klady a zápory, které byly objeveny při pozorování těchto aplikací.

2.2.1 Anotační aplikace

V první fázi projektu jsem provedl průzkum trhu, při kterém jsem mapoval existující anotační aplikace. Tento průzkum mi poskytl pohled na to, v čem aplikace vynikají, a co je naopak nedostatečné. Průzkum probíhal tak, že jsem si našel vhodnou aplikaci, kterou jsem spustil a pozoroval ji jako celek, od rozložení prvků na obrazovce až po samotnou funkčnost. Mezi hlavní faktory, které jsem pozoroval, a podle kterých jsem aplikace hodnotil patří:

- Grafické rozhraní aplikace.
- Funkčnost aplikace.
- Rozložení prvků na obrazovce.
- Ovládání aplikace – intuitivnost.
- Zajímavé prvky – funkce v aplikaci.

Pro větší přehled jsem souhrn veškerých poznatků shrnul do tabulky 2.1.

Skitch

První aplikací, kterou jsem zkoumal, byla aplikace Skitch. Hlavním cílem bylo zjistit funkcionalitu nástroje a vyzkoušet si anotaci obrázků. Na obrázku 2.1 vlevo je vidět anotace části mapy v aplikaci Skitch, která byla vyfocena mobilním telefonem. Na tomto obrázku jsem se pokusil anotovat pozici a název budovy areálu Fakulty informačních technologií v Brně. Pomocí obdélníku, šipky a popisku jsem zcela zřetelně určil, kde se areál nachází. Aplikace Skitch poskytuje mnoho užitečných nástrojů a funkcí, mezi které patří:

- Možnost výběru obrázků z mobilního zařízení nebo pořízení aktuálního snímku.
- Možnost výběru z mnoha nástrojů – šipka, text, obdélník, pero, označení místa křížkem, rozmazání určitého segmentu, vložení emoticony na určité místo.
- Možnost výběru barvy pro kreslení.
- Možnost výběru tloušťky kreslícího nástroje.
- Funkce Undo, která umožní uživateli návrat o krok zpět v anotaci.
- Uživatel může smazat celou plochu.
- Možnost sdílení anotovaného obrázku na sociálních sítí.
- Možnost ořezat obrázek.
- Možnost rotace obrázku.

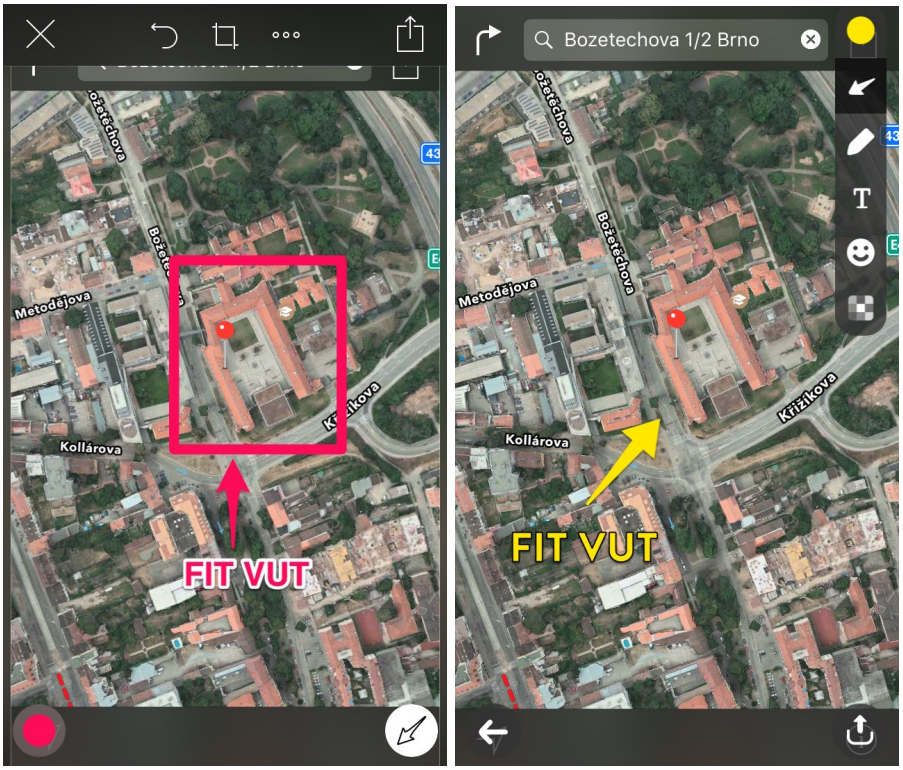
Grafické rozhraní působí velmi pěkně a uceleně. Ovládání je velmi intuitivní, prvky nejsou chaoticky rozházeny. Při anotaci jsem nezpozoroval žádné rušivé elementy. Aplikace není zaměřena na určitý obor, nebo skupinu lidí. Díky podpoře mnoha funkcí a nástrojů je cílena na širokou veřejnost. S mojí aplikací je zde spojitost ve výběru obrázků ze složky a nástrojů pro kreslení a anotování.

Annotate

Na obrázku 2.1 vpravo můžeme vidět, že jsem se i v této aplikaci pokusil anotovat část mapy, stejně jako tomu bylo v aplikaci Skitch 2.2.1. Tato aplikace nenabízí tolik nástrojů a možností jako aplikace Skitch 2.2.1. Například zde chybí anotační nástroj obdélník, takže jsem ani nemohl označit místo na mapě, jak je vidět na obrázku 2.1 vpravo. Dále zde oproti aplikaci Skitch 2.2.1 chybí:

- Možnost výběru tloušťky kreslícího nástroje.
- Sdílení anotovaných fotek.
- Možnost smazání celé plochy.
- Možnost ořezat obrázek.
- Možnost rotace obrázku.

Grafická stránka aplikace je velmi povedená a propracovaná do detailů. To stejné mohu říci o intuitivním ovládání a rozmístění prvků po obrazovce, kde jsem nezpozoroval žádné rušivé prvky. Při anotování je kreslení velmi plynulé, a pro uživatele velmi pohodlné. Stejně jako aplikace Skitch 2.2.1 je aplikace Annotate zaměřena na širší spektrum lidí, i přes menší počet nástrojů a možností. Spojitost s mojí aplikací zde vidím především ve výběru anotovaných obrázků ze složky mobilního zařízení a v implementaci nástrojů pro kreslení a anotování.



Obrázek 2.1: Srovnání aplikace Skitch a Annotate.

	+	-
Skitch	moderní grafika	chybí podpora gest
	intuitivní ovládání	
	rozložení prvků	
	mnoho nástrojů	
	možnost výběru barvy	
	možnost výběru tloušťky nástroje	
	funkce Undo	
	smazání celé plochy	
	ořezání a rotace obrázků	
	sdílení obrázků	
	plynulá anotace	
Annotate	moderní grafika	chybí podpora gest
	intuitivní ovládání	chybí výběr tloušťky nástroje
	rozložení prvků	málo anotačních nástrojů
	možnost výběru barvy	málo kreslících nástrojů
	funkce Undo	
	plynulá anotace	

Tabulka 2.1: Shrnutí nasbíraných poznatků o existujících anotačních aplikacích.

2.2.2 Aplikace zaměřené na kreslení

Anotační aplikace mají ve většině případů velmi úzkou souvislost s kreslením. Proto jsem si vyzkoušel funkčnost mobilních aplikací zaměřených na kreslení. Mezi hlavní faktory, které jsem pozoroval, a podle kterých jsem aplikace hodnotil patří:

- Grafické rozhraní aplikace.
- Funkčnost aplikace.
- Plynulost kreslení.
- Ovládání aplikace – intuitivnost.
- Zajímavé prvky – funkce v aplikaci.

Pro větší přehlednost je souhrn veškerých aplikací, kterými jsem se v průzkumu zabýval, uveden v tabulce 2.2. Dvě aplikace, které mě nejvíce zaujaly, jak po stránce designu a rozložení prvků, tak po stránce funkcionality, jsem porovnal v následujících odstavcích. Tyto aplikace splňovaly vlastnosti, které jsem chtěl přenést i do svojí aplikace. Mezi tyto vlastnosti patří hlavně pěkný, ucelený design a lehká ovladatelnost aplikace.

DoodleBuddy

Na obrázku 2.2 vlevo je zobrazena aplikace při jejím spuštění a následném namalováním čáry tahem štětce. Z tohoto obrázku lze vidět, že lišta pro výběr kreslících názorů je jasně viditelná a zřetelně ukazuje, který nástroj má uživatel nyní vybrán. Plocha na kreslení je velká a nejsou zde žádné zbytečné elementy, které by uživatele při malování rušily. Tahy štětcem jsou velmi plynulé, a ani po namalování několika čar na obrazovku se aplikace nijak neseká. Dalším velkým přínosem této aplikace je rozpoznávání gesta při zatřesení s mobilním zařízením. Toto gesto vyčistí celou obrazovku a uživatel může bez jakéhokoliv klikání znovu začít s malováním. Oproti aplikaci Sketches 2.2.2, o které se zmíním níže, je však design aplikace poněkud starší. Dalším negativním faktorem je, že aplikaci chybí nástroj guma.

Sketches

Tato aplikace mě zaujala ze všech zkoumaných aplikací nejvíce, a to ze dvou důvodů. Tím prvním je skvěle propracovaná grafická stránka aplikace, jak lze vidět na obrázku 2.2 vpravo. Dále je zde vidět propracovaný výběr kreslících nástrojů. Při výběru nástroje se vybraný nástroj zabarví a všechny ostatní se skryjí. Kreslící plocha je stejně jako u aplikace DoodleBuddy 2.2.2 velká a umožňuje uživateli vysoký komfort při malování. Druhý důvod je ten, že aplikace je současně velice jednoduše ovladatelná bez jakýchkoliv zbytečných tlačítek a možností navíc. Jedinou nevýhodou oproti aplikaci DoodleBuddy 2.2.2 je chybějící implementace rozpoznávání gest.

Obrázek 2.2: Srovnání aplikací DoodleBuddy a Sketches.

	+	-
DoodleBuddy	intuitivní	složitý výběr barvy
	plynulé tahy štětcem	starší grafika
	smazání plochy po zatřesení	chybí nástroj guma
Sketches	moderní grafika	
	moderní výběr štětců	
	plynulé tahy štětcem	
	rozvržení prvků	
	jednoduché ovládání	
ArtStudion	zpracování výběru barvy	chybí nástroj guma
	HSV i RGB mód	
Sketch Guru	nástroj guma	zastaralá grafika
	velké množství tvarů	neintuitivní výběr barvy
Finger Paint	jednoduché, ale moderní rozhraní	málo kreslících nástrojů
	jednoduchý výběr barvy	chybí nástroj guma
	velká plocha na malování	chybí návrat o krok zpět
	rozložení prvků	
Scribble	jednoduché rozhraní	málo kreslících nástrojů

Tabulka 2.2: Shrnutí nasbíraných poznatků o aplikacích zaměřených na kreslení.

Kapitola 3

Specifikace

3.1 Vymezení pojmů

Tato práce se zabývá vytvořením multiplatformní mobilní anotační aplikace. Pro lepší přehled čtenáře zde definujeme termíny, které jsou v práci použity.

- **Multiplatformní software** = programy, které mohou bežet na více platformách[1].
- **Mobilní** = schopný přemístění, pohyblivý, přenosný [7].
- **Anotace** = připojování poznámek k textu, poznámka, vysvětlivka k textu [12].
- **Aplikace** = programové vybavení, které umožňuje provádět nějakou užitečnou činnost[2].

3.2 Cílová skupina

- Výzkumníci v oboru geolokalizace v přírodě.
- Pracovníci nebo výzkumníci, kteří potřebují označit různé segmenty nebo objekty v obrázku.

3.3 Funkcionalita

Aplikace je primárně určena pro používání na tabletu. Můžeme ji však použít i na stolním počítači. Hlavním cílem této aplikace je snadné nahrání a anotování obrázků. Při vytvoření projektu uživatel zadá název projektu a vybere složku, ve které jsou obrázky k anotování. Uživatel si k samotnému anotování může vybrat z několika nástrojů – štětec, lomená čára, obdélník. Každý tah štětcem, lomená čára, obdélník mohou mít různou barvu a šířku. Aplikace podporuje další užitečné funkce, mezi které patří funkce *Undo* a *Redo*, které umožňují uživateli vrátit svůj poslední krok, nebo se posunout o jeden krok dopředu. Aplikace podporuje funkci, která smaže veškerý anotovaný obsah, aby uživatel mohl začít anotovat od začátku. Mezi další hlavní cíle této aplikace patří optimalizace času, který je potřebný pro přepínání mezi jednotlivými obrázky. Každý anotovaný obrázek se uloží do dvou souborů, kdy jeden představuje originální snímek a druhý anotovanou bitmapu. Uživatel může

[1] http://slovnik-cizich-slov.abz.cz/web.php/slovo/multiplatformni-software
[2] http://slovnik-cizich-slov.abz.cz/web.php/slovo/aplikace

rozdělaný projekt kdykoliv ukončit. K uloženému projektu se může kdykoliv v budoucnu vrátit nebo ho vymazat.

Pro lepší přehled čtenáře zde definujeme hlavní požadavky na aplikaci. Illa Neustadt a Jim Arlow ve své knize *UML 2 a unifikovaný proces vývoje aplikací* rozdělují požadavky na **funkční** a **nefunkční** [11]. Rozdělení na funkční a nefunkční požadavky patří mezi základní a nejjednodušší dělení. Funkční požadavky jsou požadavky určující, co by měl software dělat, nefunkční požadavky jsou požadavky vyjadřující nefunkční omezení systému [11]. U funkčních požadavků jsme zdůvodnili, proč je daný požadavek potřeba a definovali jeho prioritu pro implementaci. Prioritu 1 má požadavek s nejnižší prioritou. Prioritu 3 má požadavek s nejvyšší prioritou.

3.3.1 Funkční požadavky

- Založení nového projektu.
  K samotnému anotování potřebujeme získat nejprve informace – název projektu a složku s obrázky.
  Priorita: 3

- Implementace anotačních a kreslících nástrojů – štětec, lomená čára, obdélník.
  K samotnému anotování potřebujeme nástroje, kterými budeme anotovat.
  Priorita: 3

- Uživatel si bude moci vybrat barvu nástroje pro anotování a tloušťku čáry.
  Velmi často potřebujeme odlišit segmenty nebo objekty různou barvou, nebo šířkou.
  Priorita: 3

- Implementace uložení rozdělaného projektu a následného pokračování v projektu.
  V případě, kdy anotujeme větší počet obrázků, se nám funkce uložení celého projektu a následného pokračování může hodit.
  Priorita: 3

- Smazání projektu.
  Když můžeme projekt vytořit, musíme podporovat i opačnou operaci.
  Priorita: 3

- Implementace undo a redo.
  Při anotování se občas spleteme a potřebujeme se vrátit o krok zpět, nebo posunout o krok dopředu.
  Priorita: 2

- Možnost smazání celé pracovní plochy.
  Někdy se nám může stát, že chceme začít anotovat stejný obrázek od začátku.
  Priorita: 1

3.3.2 Nefunkční požadavky

- Aplikace musí bežet na více operačních systémech – multiplatformní.

- Optimalizace času, který je potřebný pro přepínání mezi jednotlivými obrázky.

- Minimalizovat počet tlačítek na obrazovce.

Illa Neustadt a Jim Arlow ve své knize [11] píší, že se požadavky zobrazují pomocí mechanismu případu užití. Mezi hlavní prvky diagramu případů užití patří: [8]

1. **Hranice systému.**
2. **Účastník (aktor).**
3. **Případ užití.**
4. **Interakce.**

Rumbaugh, Jacobson, Booch [14] definují případ užití jako jednotku funkčnosti vyjádřenou jako transakce mezi aktérem a subjektem. V příloze B jsou funkční požadavky ukázány v podobě diagramu případů užití.

Kapitola 4

Design

Slovo design znamená vzhled, tvar [10]. Chceme dosáhnout toho, aby byl ucelený, přehledný a intuitivní. Hlavním požadavkem je minimalizovat počet tlačítek na obrazovce. Neměly by se zde vyskytovat prvky, které by uživatele rušily při anotování. Naším cílem je, aby byla aplikace co nejrychleji použitelná, aby čas, který v ní člověk stráví, byl co nejmenší, a zároveň co nejpříjemnější. Důležitým faktorem, který ovlivňuje design samotné aplikace je cílová skupina uživatelů. Jak jsme již zmínili v kapitole 3, naše aplikace je cílena pro výzkumníky v oboru geolokalizace, nebo pro pracovníky a výzkumníky, kteří potřebují označit různé segmenty nebo objekty v obrázku. Russ Unger a Carolyn Chandler ve své knize *A Project Guide to UX Design: For User Experience Designers in the Field or in the Making* popisují jako efektivní způsob testování designu tři základní způsoby: [15]

- Wireframy a Anotace.
- Prototyping.
- Mockupy.

Při vytváření designu můžeme postupně vytvářet wireframy, prototypy a následně mockupy. Není však podmínkou vytvářet všechny tři formy designu. Záleží na projektu a problémech, které chceme vyřešit. Dále záleží například na tom, jakou míru abstrakce požaduje potenciální zákazník[1]. Těchto faktorů, které ovlivňují výběr správné formy vytvoření designu je několik. Pro shrnutí jsem v tabulce 4.1 uvedl základní rysy a rozdíly.

	Přesnost	**Použití**	**Hlavní rysy**
Wireframe	Nízká	Dokumentace, rychlá komunikace	Bílo-černá reprezentace
Prototyp	Střední až vysoká	Uživatelské testování	Interaktivní
Mockup	Střední až vysoká	Sbírání zpětné vazby	Statická vizualizace

Tabulka 4.1: Ukázka hlavních rozdílů a příklady použití wireframů, prototypů a mockupů. Zdroj: http://designmodo.com/wp-content/uploads/2012/07/table.jpg

[1] http://mobidev.biz/blog/telling_the_difference_wireframes_prototypes_mockups

4.1 Mockupy

Slovo Mockup, nebo-li též Mock-Up, znamená v překladu něco jako model, model na testování nebo učení[2]. Jedná se o velmi blízkou vizualizaci reality, se statickou reprezentací funkcionality. Mockupy jsou velmi dobré nástroje pro zpětnou vazbu, protože testovaní uživatelé si aplikaci dokáží velmi dobře představit. Měli by mít pocit, jako by s aplikací skutečně interagovali. Mockupy již obsahují barvy, grafiku, grafy a fonty, které budou zastoupeny i ve výsledné aplikaci.

4.2 Návrh

Pro moji aplikaci jsem se rozhodl vytvořit mockupy. Hlavním důvodem výběru právě mockupů byl fakt, že jsem chtěl získat zpětnou vazbu od respondentů, kteří anotaci objektů v praxi používají, nebo mají s anotováním objektů již nějaké zkušenosti. Dále jsem jim chtěl nabídnout pohled na design, který bude co nejvíce odpovídat výsledné aplikaci. V dnešní době je mnoho nástrojů, díky kterým tvorba mockupu nezabere mnoho času a nemusíme se učit ani žádnou syntaxi nového jazyka.

4.2.1 Tvorba mockupů

V první fázi jsem vytvořil mockupy. Vytvořil jsem, jak verzi mockupů pro mobilní telefon 4.1, tak pro tablet 4.2. Důraz jsem kladl na lehké a intuitivní ovládání. Inspiroval jsem se aplikacemi, které jsou dostupné, a ze kterých jsem posbíral informace, viz. kapitola 2.2.

V další fázi jsem vytvořené mockupy otestoval na potenciálních zákaznících, kteří mi dali zpětnou vazbu. Všechny jejich poznatky jsem si poznamenal a provedl jsem vyhodnocení testování.

V poslední fázi jsem diskutoval názory testovaných uživatelů a upravoval mockupy.

4.2.2 Testování mockupů

Pro testování jsem použil tzv. **Usability testing**.

Usability testing

Russ Unger a Carolyn Chandler [15] tvrdí následující fakta. Usability testing je jedna z nejvíce používaných metod testování User Experience (UX) Designu, a zároveň se jedná o nejznámější způsob testování mezi UX designery. Mezi UX designery probíhá dlouhá diskuze, zda-li se jedná o **kvantitativní** nebo **kvalitativní** metodu. Obecně platí, že **kvantitativní** metoda potřebuje oproti **kvalitativní** metodě větší soubor dat a respondentů. Russ Unger a Carolyn Chandler píší, že je možný jak kvantitativní, tak kvalitativní přístup a každý z nich produkuje užitečné výsledky.

Zastánci **kvantitativní** metody tvrdí, že **kvantitativní** metoda:

- Přináší výsledky, které mohou být ověřeny statisticky.
- Snižuje pravděpodobnost individuální zaujatosti.
- Poskytuje vyšší stupeň jistoty, že výsledky jsou odrazem celé uživatelské základny.

[2] http://dictionary.reference.com/browse/mock-up

Obrázek 4.1: Ukázka prvotního mockupu pro mobilní telefon.
Zdroj obrázku: https://www.flickr.com/photos/tom_hall_nz/15613249585/
Autor: Tom Hall

Obrázek 4.2: Ukázka prvotního mockupu pro tablet.

- Nabízí jasné, numerické metody ověření nálezů – například, kolik uživatelů narazilo na stejný problém.

Zastánci **kvalitativní** metody tvrdí, že **kvalitativní** metoda:

- Buduje zkušenosti a empatii v návrháři, propaguje kreativní řešení zaměřené na uživatele.
- Je méně nákladná oproti kvantitativní, protože je potřeba méně uživatelů.
- Nálezy nejdou ověřit numericky, mohou být ověřeny návrhářem.

Samotný koncept Usability testingu je velmi jednoduchý.

1. Vytvoříme sadu úkolů.
2. Zadáme úkoly uživatelům, aby tyto úkoly vykonali.
3. Poznamenáme si, kde uspěli, a kde měli naopak problémy.

Usability testing definuje, že před samotným testováním bychom si měli odpovědět na čtyři základní otázky: [15]

1. Proč testuji?
    - Abych vytvořil intuitivní a dobře ovladatelnou aplikaci, získal zpětnou vazbu, a tím dosáhl tohoto cíle v co největší míře.
2. Kdo je testovaným?
    - Výzkumníci zabývající se geolokalizací.
    - Lidé, kteří mají s anotováním objektů nebo segmentů na mobilním zařízení nebo stolním počítači již nějaké zkušenosti.
3. Co testuji?
    - Rozložení prvků na obrazovce a použitelnost aplikace.
4. Jaké informace sbírám?
    - Jakékoliv informace, které mi pomohou aplikaci zlepšit v její funkčnosti, grafickém provedení, intuitivnosti, ovladatelnosti.

Po zodpovězení na tyto otázky jsem přistoupil k výběru testované skupiny lidí. Výběr lidí jsem udělal podle tzv. **screeneru** 4.2. Screener je typ dotazníku, který jsem použil na výběr vhodných kandidátů pro testování [15]. Vhodnými kandidáty jsou lidé, kteří splňují určitá kritéria. Lidé, kteří tato kritéria nesplňují, se pro testování nehodí. Tyto lidi označím v screeneru jako **TERMINATE**. Těmto lidem je nutno šetrně oznámit, že se pro testování nehodí. Je však dobrým zvykem, vzít si od těchto respondentů kontakt pro případné testování dalšího produktu, kde by naše kritéria splňovali. Screener pro moji anotační aplikaci jsem přehledně shrnul do tabulky 4.2. Spolu s pojmem screener je zde další důležitá otázka, která je velmi diskutovaná. Kolik uživatelů je pro testování dostačující? Russ Unger a Carolyn Chandler ve své knize [15] píší, že pro kvantitativní test je potřeba alespoň 20 účastníků pro každé testování. Pro kvalitativní test je ideální počet 5 až 8 účastníků. Obecně platí,

Otázka	Možnost odpovědi	Vhodnost
Do jakého věkového rozpětí patříte?	Pod 18 let	
	18-24 let	
	25-34let	
	35-44let	
	45-54let	
	54 a více let	
Ja často anotujete fotografie?	Nikdy jsem ještě neanotoval	TERMINATE
	Každý den	
	Jednou za týden	
	Párkrát za týden	
	Jednou za měsíc	
	Párkrát za měsíc	
	Jednou za rok	
	Párkrát za rok	
Pokud anotujete, jaký nástroj používáte?		
Na jakém zařízení anotujete?	Stolní PC	
	Notebook	
	Mobilní telefon	
	Tablet	

Tabulka 4.2: Screener pro anotační aplikaci.

že více jak jeden účastník je dostačující k tomu, aby byly odhaleny chyby, které náš návrh „skrývá“ [15].

Aby uživatel věděl, jaké akce se po něm při testování mockupu požadují, je nutné vytvořit úkoly pro testování, které bude uživatel postupně plnit. Tyto úkoly by měly mít rostoucí tendenci, co se týče jejich složitosti – začneme od jednodušších úkolů, které postupně stěžujeme. Pro testování jsem vytvořil sadu úkolů, kterou jsem shrnul do tabulky 4.3.

Číslo úkolu	Úkol
1	Založte nový projekt s názvem Example, vyberte složku Hory a začněte anotovat.
2	Vyberte kreslící nástroj lomená čára a namalujte úsečku.
3	Změňte barvu pera na zelenou a nakreslete další úsečku.
4	Vyberte kreslící nástroj obdélník a označte oblast.
5	Vraťte se o krok zpět.
6	Posuňte se o krok dopředu.
7	Přejděte na další obrázek.
8	Přerušte editaci.

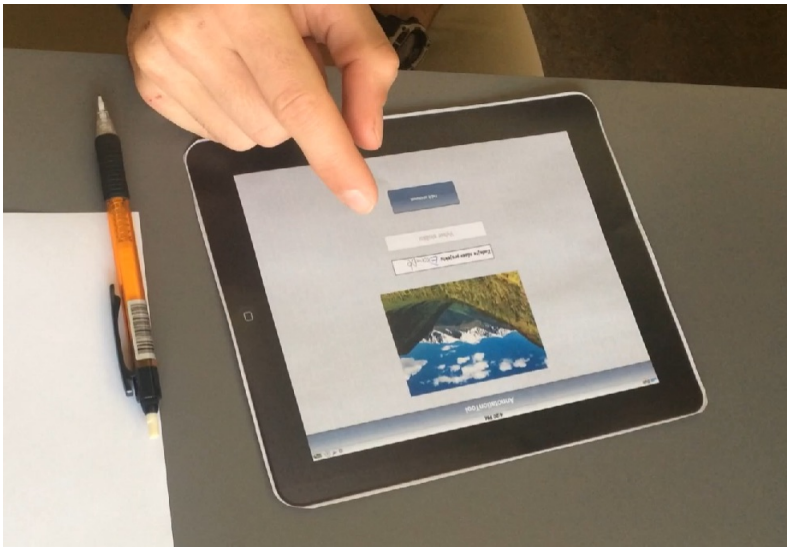
Tabulka 4.3: Úkoly pro testování.

Nastavení testování

Testování probíhalo v klidné a tiché místnosti, kde byl minimální hluk. Zúčastnili se ho 3 respondenti a bylo prováděno pomocí papírového mockupu. Na stůl jsem položil papírový mockup a vedle něj sadu úkolů 4.3, které se respondenti pokoušeli splnit. U testování byli přítomni ještě 2 moderátoři. Jeden moderátor měnil papírové mockupy podle chování uživatele, druhý celý proces testování nahrával pomocí mobilního telefonu, a to ze dvou hlavních důvodů:

1. Respondent nemusí čekat, až si něco poznamenáme. Testování je tak plynulejší a má větší a rychlejší spád.

2. Video si můžeme v budoucnu kdykoliv přehrát. Můžeme se podívat nejen na to, jak uživatel plnil úkoly, ale také, jak při tom myslel – tyto informace bychom si na papír poznamenat nestihli.



Obrázek 4.3: Obrázek pořízený při testování mockupu.

Na konci testování jsem se každého respondenta zeptal na pár otázek, jaké byly jejich pocity a dojmy z aplikace jako celku, co se jim líbilo, a co by změnili. V tabulce 4.4 jsou ukázány všechny závěrečné otázky, které jsem si připravil před samotným testováním. Odpovědi respondentů na tyto závěrečné otázky jsem shrnul do tabulek 4.5, 4.6, 4.7.

Číslo otázky	**Otázka**
1	Jak jste se cítil/a při práci s anotační aplikací?
2	Zdálo se Vám ovládání intuitivní?
3	Vyhovovalo Vám rozložení tlačítek na obrazovce?
4	Dělalo Vám splnění nějakého úkolu větší problém?
5	Co se Vám nelíbilo (a proč) a co byste změnil/a?

Tabulka 4.4: Závěrečné otázky po konci testování.

Číslo otázky	Odpověď
1	S aplikací se mi pracovalo poměrně intuitivně.
2	Ano.
3	Ano.
4	Ne.
5	Nelíbila se mi ikona anotačního nástroje pero.

Tabulka 4.5: Zodpovězené závěrečné otázky od respondenta 1.

Číslo otázky	Odpověď
1	Ovládání bylo jednoduché.
2	Ano, kromě posunu na další obrázek.
3	Ano.
4	Ano, úkol s posunutím se na další obrázek.
5	Změnil bych posunutí na další obrázek.

Tabulka 4.6: Zodpovězené závěrečné otázky od respondenta 2.

Číslo otázky	Odpověď
1	Cítil jsem se dobře.
2	Ano.
3	Ano.
4	Ne.
5	Změnil bych ikonu anotačního nástroje pero.

Tabulka 4.7: Zodpovězené závěrečné otázky od respondenta 3.

4.2.3 Vyhodnocení testování

V první fázi jsem provedl analýzu videa, které jsem natočil při testování. Udělal jsem si poznámky, s čím měli uživatelé problémy při plnění úkolů, a co by v aplikaci změnili. Ke každé poznámce jsem si sepsal čas videa, prioritu pro odstranění nedostatku a návrh možného řešení. Priorita 1 značí nejnižší prioritu, priorita 3 značí nejvyšší prioritu. Stručný výpis poznámek, který jsem si po přehrání videa vypsal u každého respondenta, je ukázán v tabulkách 4.8, 4.9, 4.10. Výhoda tohoto postupu spočívá v tom, že pokud se budeme chtít vrátit někdy v budoucnu k úpravě mockupu, nemusíme již znovu zhlédnout celé video, ale úplně nám postačí přehrát jen poznamenané úseky.

Čas	Poznámka	Priorita	Řešení/Doporučení
1:30	Tlačítko vyber složku je zašedlé.	3	Zvýraznit tlačítko.
1:58	Není uvedena složka projektu.	3	Uvést, jakou složku anotujeme.
4:35	Nelze vybírat z projektů.	3	Doimplementovat – QComboBox.
7:11	Špatná ikona nástroje pero.	3	Změnit ikonu pera.

Tabulka 4.8: Ukázka výpisu poznámek od respondenta 1.

Čas	Poznámka	Priorita	Řešení/Doporučení
3:00	Problémy s posunutím na další obrázek.	3	Implementovat tlačítka.
6:06	Fit úsečky na hranu horizontu.	1	Doporučení pro zlepšení.

Tabulka 4.9: Ukázka výpisu poznámek od respondenta 2.

Čas	Poznámka	Priorita	Řešení/Doporučení
0:44	Špatná ikona nástroje pero.	3	Změnit ikonu nástroje.
3:46	Problémy s posunutím na další slide.	3	Implementovat tlačítka.
5:38	Posouvat hrany obdélníku.	1	Doporučení pro zlepšení.

Tabulka 4.10: Ukázka výpisu poznámek od respondenta 3.

4.2.4 Úprava mockupů

V této fázi jsem si vzal poznámky od respondentů viz. tabulky 4.8, 4.9, 4.10. Začal jsem s poznámkami s prioritou 3. Zamyslel jsem se, jestli jsou nedostatky, které mi respondenti poskytli dobře zdůvodnitelné a pochopitelné. Všechny nedostatky s prioritou 3 mi přišly věcné a podstatné pro vylepšení aplikace. Tyto prvky jsem z mockupů odstranil a nahradil je novými. Na obrázku 4.2 můžeme vidět návrh mockupu pro tablet před testováním. Obrázek 4.4 nám ukazuje výsledný mockup pro tablet, který byl předělán na základě výsledku testování.

4.3 Implementace

V dnešní době je mnoho nástrojů pro tvorbu mockupů. Mezi ty nejznámější patří Balsamiq[3], Mockingbird[4], Mockup Builder[5], Mockflow[6], Pencil[7].

4.3.1 Použité technologie

Pro vytvoření mockupu jsem využil open-source software Pencil, který můžeme vidět na obrázku 4.5. Mezi hlavní důvody, proč jsem si vybral právě Pencil patří:

- Jde o multiplatformní software – OS X, Linux, Windows.
- Umožňuje export do různých formátů – PDF, PNG, Single Web Page, SVG, ODT.
- Umožňuje vytvořit akce na jednotlivá tlačítka a následně „propojit“ jednotlivé mockupy. Můžeme tedy mockup testovat přímo z pohledu respondenta ještě před samotným testováním.
- Umožňuje vytvářet diagramy pro větší přehled a orientaci.

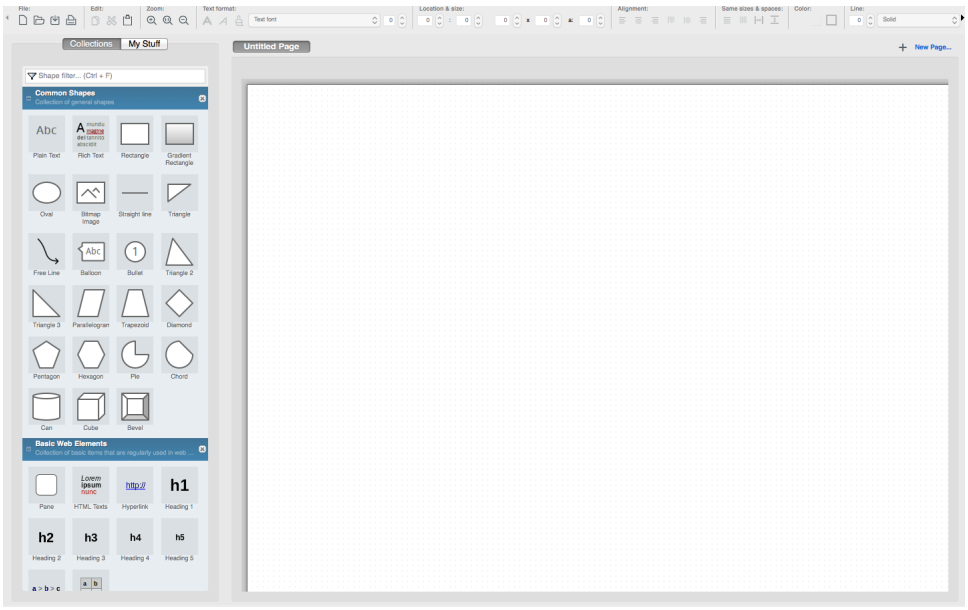
[3] https://balsamiq.com/products/mockups/
[4] https://gomockingbird.com/home
[5] http://mockupbuilder.com/
[6] https://www.mockflow.com/
[7] http://pencil.evolus.vn

Obrázek 4.4: Ukázka výsledného mockupu pro tablet.
Zdroj obrázku: https://www.flickr.com/photos/tom_hall_nz/15613249585/
Autor: Tom Hall

4.3.2 Převedení mockupu do reálné implementace

Mockupy byly vytvářeny pro operační systém iOS. Výsledný mockup aplikace pro tablet je ukázán na obrázku 4.4. V dnešní době existuje mnoho frameworků, které dokáží zachovat tzv. **look and feel**[8] na různých platformách. V této práci je především kladen důraz na přenositelnost aplikace pro různá zařízení, proto byl pro implementaci použit Qt Framework. Díky tomuto frameworku je aplikace přenositelná na mnoho zařízení, jak můžeme vidět v tabulce 5.1. Jak je ale patrné z obrázku 4.6, tak vzhled ikonek nástrojů, rozložení grafických prvků a funkcí bylo zachováno.

[8] http://www.dictionary.com/browse/look-and-feel



Obrázek 4.5: Ukázka softwaru Pencil pro vytváření mockupů.

Obrázek 4.6: Ukázka vzhledu výsledné aplikace na operačním systému iOS.
Zdroj obrázku: https://www.flickr.com/photos/tom_hall_nz/15613249585/
Autor: Tom Hall

Kapitola 5

Implementace

5.1 Použité technologie

Pro implementaci aplikace jsem si vybral **Qt Framework**. Mezi hlavní důvody, proč jsem si vybral Qt Framework, patří skutečnost, že podporuje velké množství platforem. Všechny podporované platformy jsem přehledně shrnul do tabulky 5.1. Tabulka je platná pro Qt Framework verzi 5.6 ke dni 26. 3. 2016. Mezi další výhody patří velmi přehledně zpracovaná dokumentace[1], vývojové prostředí Qt Creator[2] a Qt Designer[3]. Pro vývoj aplikace jsem použil **Qt Framework** verze 5.5 a vývojové prostředí **Qt Creator** verze 3.5.0.

Desktopové platformy	**Mobilní platformy**
Windows	Android
Linux/x11	iOS
OS X	WinRT

Tabulka 5.1: Podporované platformy frameworku Qt.
Zdroj: http://doc.qt.io/qt-5/supported-platforms.html

Qt Framework

Slouží pro vytváření multiplatformních programů a aplikací s grafickým uživatelským rozhraním. Jedná se o framework programovacího jazyka C++, ale existuje i pro jiné jazyky – např. Python (PyQt) nebo Java (Jambi) [5].

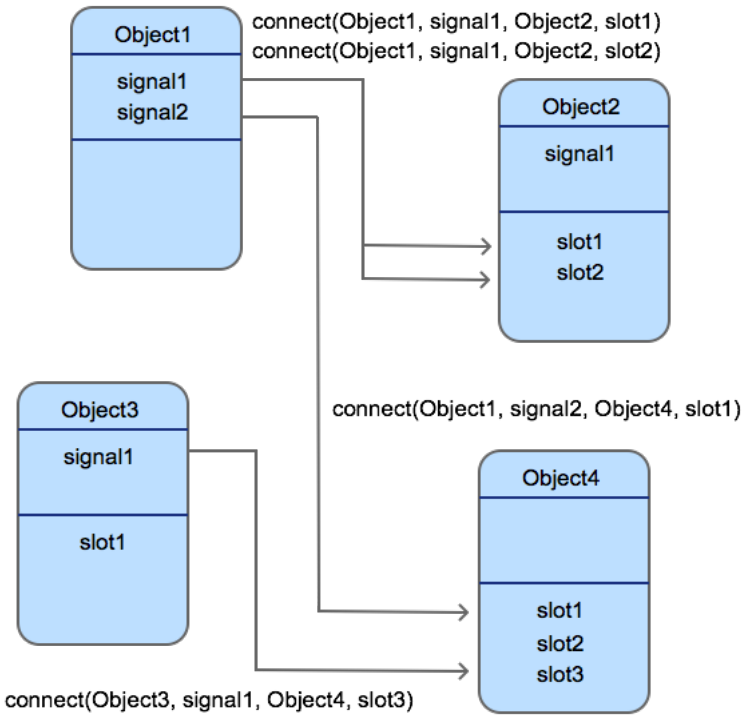
Mezi hlavní mechanismy Qt Frameworku patří tzv. **Signály a sloty**[4]. Jedná se o mechanismus komunikace mezi objekty, který je zobrazen na obrázku 5.1. Dříve se pro komunikaci mezi objekty používal tzv. **callback**. Tento přístup měl 2 základní nevýhody [5]:

1. Nebyl typově bezpečný – nebyla zde typová kontrola.
2. Volaná metoda musela znát ukazatel na metodu, ze které byla vyvolána.

[1] http://doc.qt.io/
[2] https://www.qt.io/ide/
[3] http://doc.qt.io/qt-4.8/designer-manual.html
[4] http://doc.qt.io/qt-4.8/signalsandslots.html

Tyto nevýhody s příchodem signálů a slotů vymizely. Qt Framework má mnoho předdefinovaných signálů. Například signál **clicked()**, který je zachycen, když uživatel klikne na požadované tlačítko. Výhodou ovšem je, že si můžeme nadefinovat vlastní signály, které budou spouštět metody při jejich zachycení. Ve své aplikaci jsem mechanismus signálů a slotů použil například pro zachycení události, že uživatel klikl na tlačítko, nebo pro zavírání přechozích dialogů.

Dalším důležitým prvkem tohoto frameworku je tzv. **widget**[5]. Jedná se o základní grafický objekt tohoto frameworku. Widget je tedy například tlačítko, label, checkbox. Pro rozmístění těchto widgetů se používá tzv. **layout**[6]. V mojí aplikaci používám k rozmístění widgetů převážně tzv. **GridLayout**[7]. Mluvíme o tzv. rozmístění do „mříže“. Rozděluje nám layout na řádky a sloupce, do kterých můžeme přidat widgety. Výhodou tohoto rozložení je, že nám drží prvky téměř identicky pro různá zařízení.



Obrázek 5.1: Ukázka komunikace mezi objekty v Qt Frameworku pomocí signálů a slotů. Zdroj: http://doc.qt.io/qt-4.8/signalsandslots.html

Qt Creator

Jedná se o vývojové prostředí pro programování v jazyce C++ a Qt Frameworku. Mezi hlavní výhody patří:

- Jedná se o multiplatformní software.

[5] http://doc.qt.io/qt-4.8/qwidget.html
[6] http://doc.qt.io/qt-4.8/layout.html
[7] http://doc.qt.io/qt-4.8/qgridlayout.html

- Podporuje správu verzí.
- Doplňování kódu.
- Zvýraznění syntaxe.
- Má k dispozici debugger.
- Můžeme zde rovnou spouštět aplikace pro různé operační systémy a zařízení.

5.2 Architektura aplikace

Při spuštení aplikace má uživatel na výběr ze tří možností – založit nový projekt, pokračovat v již rozpracovaném projektu nebo smazat projekt. Při zvolení možnosti založit nový projekt musí uživatel vyplnit název projektu a vybrat složku, ve které jsou nahrány obrázky pro anotování. Po zadání názvu projektu a výběru složky může uživatel přistoupit k samotné anotaci. Po ukončení anotace je uživateli oznámena informace, že anotování proběhlo úspěšně a jeho data byla uložena. Jakmile uživatel tuto informací potvrdí, je běh aplikace přesunut na začátek a celý tento cyklus se může opakovat. Uživatel může pokračovat v uložených projektech, nebo projekty vymazat. Slovní popis architektury aplikace jsem zobrazil pomocí flowchart diagramu, který je ukázán na obrázku 5.2. Aplikace je rozdělena do 5 hlavních oken tzv. **dialogů**[8]:

1. **Dialog**.
2. **NewProjectDialog**.
3. **StartAnnotatingDialog**.
4. **ImageCanvas**.
5. **FinishAnnotatingDialog**.

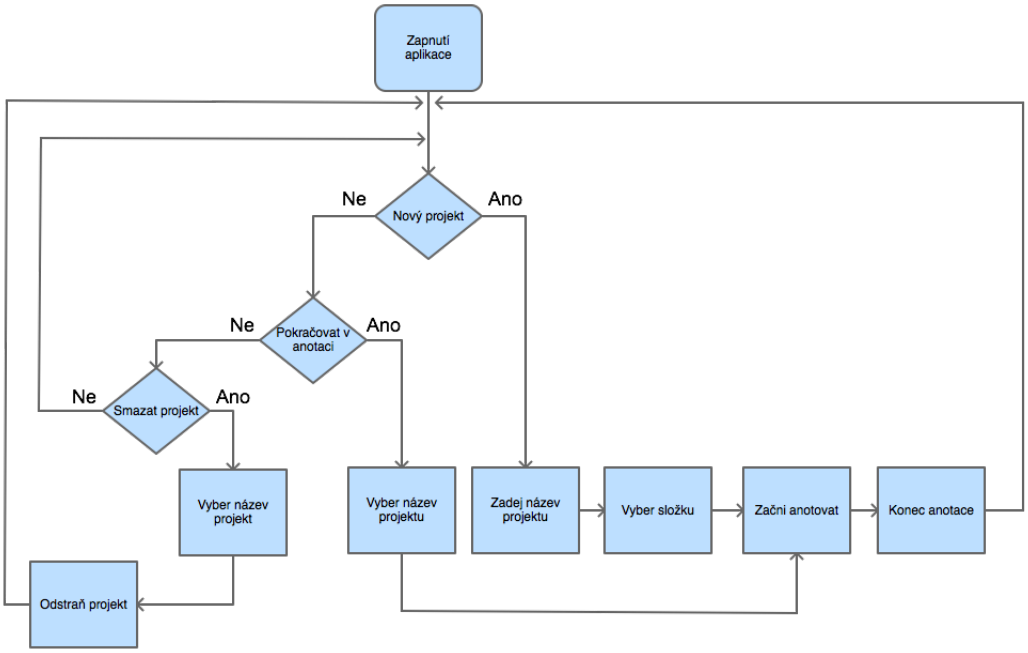
Dialogy mohou být **modální** a **nemodální** [2]. Modální dialogy vyžadují interakci uživatele. Nemodální dialogy nebrání uživateli v interakci s některým dalším oknem v aplikaci.

Vztahy mezi jednotlivými dialogy jsou přehledně shrnuty na obrázku 5.3. Jednotlivé dialogy z pohledu funkcionality jsou popsány níže. U každého dialogu jsou popsány hlavní widgety, ze kterých se dialog skládá. V příloze C je zobrazen třídní diagram. Vytvoření komponent v dialozích zajišťují metody, které se volají v konstruktorech jednotlivých tříd. Samotné komponenty se do QGridLayoutu přidávají také v konstruktorech jednotlivých tříd pomocí knihovní funkce *addWidget()*, kde se pomocí parametrů udává jejich umístění na obrazovce.

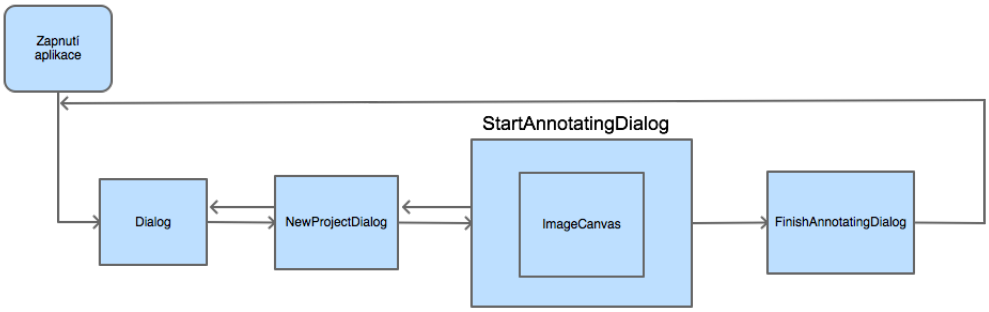
Dialog

Tento dialog je zobrazen při spuštění aplikace. Uživatel si zde může vybrat, jestli chce založit nový projekt, pokračovat v již rozdělaném projektu nebo smazat projekt. Dialog se skládá ze 3 hlavních widgetů, které můžeme spolu s grafickým provedením dialogu vidět na obrázku 5.4.

[8] http://doc.qt.io/qt-4.8/dialogs.html



Obrázek 5.2: Ukázka základní logiky aplikace.



Obrázek 5.3: Ukázka hlavních dialogů aplikace.

Metoda *createLogo()* vytvoří logo aplikace. Jedná se o Qlabel, na který je „nastaven“ obrázek loga aplikace. Všechny obrázky jsou v aplikaci uloženy v tzv. Resource Collection Files (**.qrc**[9]) souboru, který umožňuje vkládat obrázky pomocí relativní cesty.

Metoda *createComboBox()* vytvoří komponentu pro výběr z uložených projektů. Tento výběr slouží pro následné pokračování v projektu, nebo pro smazání vybraného projektu. Výběr z projektů je implementován pomocí komponenty QComboBox.

Metoda *createButtons()* vytvoří tlačítka pro vytvoření nového projektu, pokračování v anotaci z již uloženého projektu, smazání projektu. V této metodě jsou vytvořeny 3 QPushButtony – „Smazat projekt“, „Pokračovat v anotaci“, „Nový projekt“, které jsou přidány do QDialogButtonBoxu. Pomocí knihovní funkce *setOrientation()* můžeme nastavit, jestli chceme zobrazit tlačítka pod sebou – vertikálně nebo vedle sebe – horizontálně. Tlačítka „Smazat

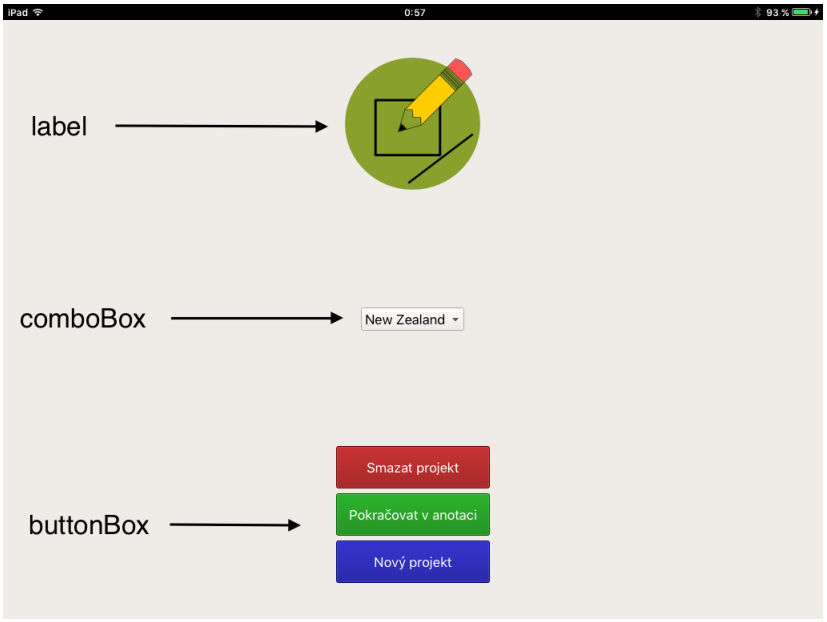
[9] http://doc.qt.io/qt-5/resources.html

projekt“, „Pokračovat v anotaci“, „Nový projekt“ mají nastavené vertikální zobrazení.

Pomocí signálů a slotů jsou zde řešeny události, které nastanou po samotném kliku uživatele na tato tlačítka. Například po kliknutí uživatele na tlačítko „Nový projekt“ dochází k zavolání slotu *newProjectStart()*, který otevře dialog pro vytvoření nového projektu. Ukázku kódu s „navázáním“ slotu na kliknutí na tlačítko můžeme vidět na obrázku 5.1.

```
connect(newProjectBtn,SIGNAL(clicked()), this, SLOT(
    newProjectStart()));
```

Ukázka kódu 5.1: Signály a sloty



Obrázek 5.4: Ukázka dialogu Dialog s widgety.

NewProjectDialog

Slouží k získání potřebných informací a dat pro samotnou anotaci. Uživatel zde zadává název projektu a vybere složku, ve které jsou uloženy obrázky, které chce anotovat. Aplikace podporuje obrázky s koncovkou **JPG** a **PNG**. NewProjectDialog se skládá ze 3 hlavních widgetů, které můžeme spolu s grafickým provedením dialogu vidět na obrázku 5.5.

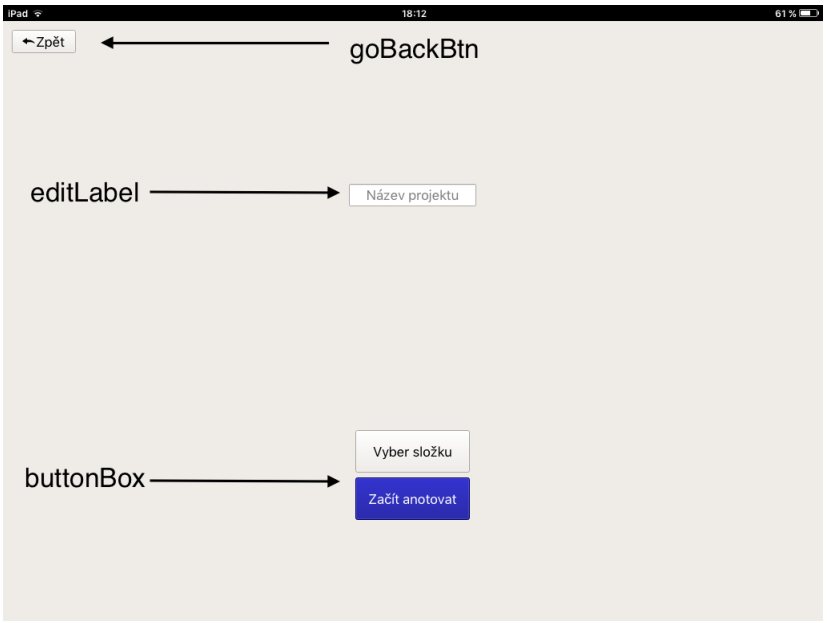
Metoda *createEditLabel()* vytvoří editační „pole“, do kterého uživatel napíše název projektu.

Metoda *checkProjectName()* kontroluje, zda uživatel zadal název projektu. Vyplnění „pole“ pro název projektu je nutnou součástí před samotnou anotací. V Případě nevyplnění je uživatel informován pomocí chybové hlášky. Editační „pole“ je vytvořeno pomocí komponenty QLineEdit.

Metoda *createButtons()* vytvoří tlačítko „Zpět“ pro návrat na úvodní dialog aplikace. Dále vytvoří tlačítka „Vyber složku“ a „Začít anotovat“. Tlačítko „Zpět“ je vytvořeno pomocí komponenty QPushButton. Uživatel se však může na úvodní dialog dostat i pomocí gesta swipe. Mechanismus swipe je implementován v metodách *mousePressEvent()* a *mouseReleaseEvent()*. V první fázi dochází k uložení bodu, kde uživatel začal gesto swipe. Jakmile

uživatel pustí prst, je tento bod porovnám s bodem, kde uživatel začal gesto swipe. V další fázi se zjišťuje, zda-li došlo k vertikálnímu nebo horizontálnímu posunu uživatelova prstu, a zda-li bylo gesto provedeno zleva doprava nebo zprava doleva. Dojde-li k horizontálnímu posunu prstu, který je veden zleva doprava, vykoná se požadovaná akce – přesun na úvodní obrazovku aplikace. Tlačítka „Vyber složku“ a „Začít anotovat“ jsou 2 QPushButtony, které jsou přidány do QDialogButtonBoxu. QDialogButtonBox je nastaven na vertikální zobrazení.

Výběr složky, ve které jsou uloženy obrázky pro anotaci je implementován pomocí QFileDialogu. Vyhledávání složky je zahájeno v dokumentové složce každého zařízení. Zde nastal problém, protože na každém zařízení je cesta k dokumentové složce jiná. Tento problém byl vyřešen pomocí tzv. **QStandardPaths**[10]. Při nastavení QStandardPaths::DesktopLocation je nám vrácena cesta do dokumentové složky pro všechna podporovaná zařízení, která byla vyjmenována v tabulce 5.1.



Obrázek 5.5: Ukázka dialogu NewProjectDialog s widgety.

StartAnnotatingDialog

Jedná se o nejrozsáhlejší dialog celé aplikace. Zajišťuje logiku celé anotace. Umožňuje uživateli přepínat mezi jednotlivými obrázky. Vybírat si jednotlivé nástroje. Uživatel zde vidí název projektu, který anotuje a pořadí právě anotovaného obrázku. StartAnnotatingDialog se skládá z 8 hlavních widgetů. Tyto widgety jsou popsány na obrázku 5.6. Dále můžeme na obrázku 5.6 vidět **StartAnnotatingDialog**, do kterého je „vložen“ **ImageCanvas** dialog.

Při vytvoření tohoto dialogu potřebujeme načíst obrázky ze složky, kterou si uživatel zvolil v předcházejícím dialogu. Vybrání obrázku ze složky zajišťuje metoda *findingImageFromFolder()*. Metoda potřebuje znát cestu ke složce. Jedná se o informaci, kterou uži-

[10] http://doc.qt.io/qt-5/qstandardpaths.html

vatel zadal v předcházejícím dialogu, proto je tato informace předávána jako parametr při vytvoření instance StartAnnotatingDialogu.

Metoda *createButtons()* vytvoří tlačítko „Zpět“ pro návrat na předcházející dialog. Dále jsou v této metodě vytvořena tlačítka pro posun obrázku dopředu nebo dozadu. Tlačítka jsou vytvořena pomocí komponenty QPushButton.

Metoda *createToolsButtonBox()* vytvoří tlačítka pro všechny anotační nástroje a funkce, viz. 5.3, 5.4. Jedná se o 7 QPushButtonů, které jsou přidány do QDialogButtonBoxu. QDialogButtonBox je nastaven na horizontální zobrazení. Po kliknutí na jedno z tlačítek štětec, lomená čára, obdélník je zavolán slot *selectTool()*, který uživateli ukáže, jaký nástroj používá. Toto znázornění je implementováno pomocí knihovní funkce *setStyleSheet()*. Pozadí používaného nástroje je zbarveno do modra.

Metoda *createLabels()* vytvoří číselnou informaci, která uživateli říká, na jakém obrázku se právě nachází. Dále potom vytvoří informaci, která uživatele informuje o názvu projektu. Stejně jako v případě cesty ke složce s obrázky pro anotaci se jedná se o informaci, kterou uživatel vyplnil v předcházejícím dialogu, proto se i tato informace předává jako parametr při vytvoření instance StartAnnotatingDialogu.

Metoda *createPenWidthSpinBox()* vytvoří editační „pole“ pro změnu šířky nástroje. Rozmezí hodnot je nastaveno od 1 do 50. Při spuštění dialogu je nastavena hodnota 5.

Slot *backImageAction()* se zavolá při kliku uživatele na tlačítko pro přechod na předcházející obrázek. Kontroluje se zde, na jakém obrázku se uživatel nachází. Jestliže se uživatel nachází na prvním obrázku, tak na tlačítko pro předchozí obrázek nelze kliknout. Pokud se uživatel může vrátit, je přehozen obrázek a sníženo počítadlo, které ukazuje, na kterém obrázku se nacházíme.

Slot *nextImageAction()* se zavolá při kliku uživatele na tlačítko pro přechod na další obrázek. Kontroluje se zde, na jakém obrázku se uživatel nachází. Jestliže se uživatel nachází na posledním, zobrazí se **FinishAnnotatingDialog** s informací, že již nezbývá žádný další obrázek a anotace je u konce. Pokud uživatel posune obrázek a není to obrázek poslední, je přehozen obrázek a zvýšeno počítadlo, které ukazuje, na kterém obrázku se nacházíme.

V samotném dialogu je umístěn další dialog – **ImageCanvas**. Vytvoření tohoto dialogu probíhá v metodě *createImageCanvas()*.

ImageCanvas

V tomto dialogu je realizováno samotné kreslení. Kreslení probíhá v metodě *paintEvent()*. Tato metoda je zavolána pokaždé, když dojde k překreslení obsahu. Překreslení probíhá po zavolání jedné z knihovní funkce *update()* nebo *repaint()*.

Dále jsou zde implementovány čtyři důležité metody, které zachytávají dotyk prstu na obrazovce:

1. *mousePressEvent()* – je zachycena, dotkne-li se uživatel prstem displeje zařízení.
2. *mouseMoveEvent()* – je zachycena, pohybuje-li uživatel s prstem po displeji zařízení.
3. *mouseReleaseEvent* – je zachycena, pustí-li uživatel prst z displeje zařízení.
4. *tapAndHold* – je zachycena, podrží-li uživatel prst delší dobu na displeji zařízení.

Obrázek 5.6: Ukázka hlavních dialogů aplikace s widgety.
Zdroj obrázku: https://www.flickr.com/photos/tom_hall_nz/15613249585/
Autor: Tom Hall

FinishAnnotatingDialog

Informuje uživatele o ukončení anotace a uložení požadovaných dat. Po potvrzení informace je zobrazena úvodní obrazovka celé aplikace a uživatel může anotační cyklus znovu opakovat. FinishAnnotatingDialog se skládá ze 2 hlavních widgetů, které můžeme spolu s grafickým provedením dialogu vidět na obrázku 5.7.

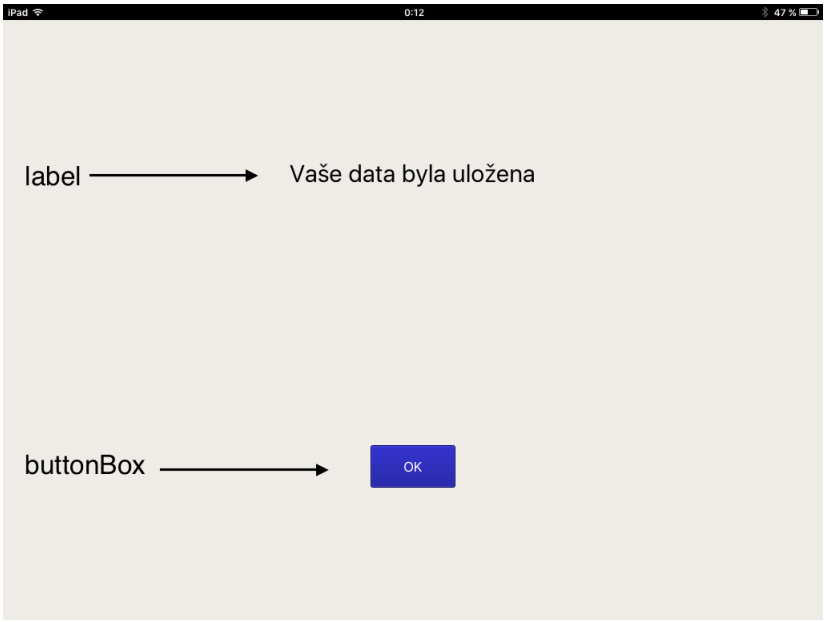
Metoda *createLabel()* vytvoří informační text pro uživatele. Metoda *createButton()* vytvoří tlačítko „OK“ pro návrat na úvodní dialog. Návrat na úvodní obrazovku je implementován pomocí signálů a slotů. Tento dialog „vyšle“ signál pomocí knihovní funkce *emit*. Signál je zachycen předchozím dialogem, který se ukončí a pošle signál svému rodiči, dokud nedojdeme na úvodní obrazovku.

5.3 Anotační nástroje

Za anotační nástroje považujeme – štětec, lomenou čáru, obdélník. Použití a implementace těchto nástrojů je popsáno v následujících odstavcích.

5.3.1 Nástroj štětec

Umožňuje uživateli malovat na obrázek, který je určený k anotaci. Každý tah štětcem může mít jinou barvu a být jinak široký. Štětec začne malovat, jakmile si uživatel vybere nástroj „štětec“ a přejede prstem po displeji zařízení. Ukončení malování je indikováno, když uživatel „zvedne“ prst z displeje zařízení. Na začátku malování – uživatel se dotkne displeje, je vytvořena instance třídy **Scribbling**. Při každém pohybu prstu na displeji jsou ukládany tyto informace:



Obrázek 5.7: Ukázka dialogu FinishAnnotatingDialog s hlavními widgety.

- Bod, na který uživatel klikl.
- Barva, kterou měl uživatel zvolenou při kliknutí.
- Šířku pera, kterou měl uživatel zvolenou při kliknutí.

Po ukončení použití nástroje „štětec“ – uživatel „zvedne“ prst z displeje zařízení, je instance přidána do statického listu **scribblingList**. Každý dokončený tah představuje jednu položku v tomto listu. Tento list nám slouží jak pro účely, kdy dojde k překreslení obsahu, tak pro funkci undo a redo, která je popsána níže 5.4.1. Malování probíhá v metodě *scribbling()*. Tahy štětcem představují několik úseček poskládaných za sebe. Z tohoto důvodu se v metodě *scribbling()* používá k vykreslení tahů štětce knihovní funkce *drawLine()*. Pseudokód metody *scribbling()* je ukázán na kódu 5.2.

```
if scribblingList is not empty
    for (int i = 0; i < scribblingList.count(); i++)
        for (int j = 0; j < (pointList.count() - 1); j++)
            Set painter pen
            Draw line
        endfor
    endfor
endfor
```

Ukázka kódu 5.2: Pseudokód metody scribbling

5.3.2 Nástroj lomená čára

Umožňuje uživateli kreslit lomené čáry, kde každá čára může mít jinou barvu a šířku. Lomená čára se začne kreslit, jakmile si uživatel vybere nástroj „lomená čára“ a dotkne

se prstem obrazovky alespoň dvakrát – musíme mít alespoň dva body pro vytvoření úsečky. Dále se již uživatel svým prstem dotýká obrazovky a čára se začíná prodlužovat a zalamovat. Jakmile chce ukončit jednu lomenou čáru a začít kreslit druhou, podrží svůj prst na obrazovce zařízení. Po chvilce je na obrazovce indikováno kolečko na konci posledního bodu lomené čáry, které informuje o konci lomené čáry.

Jakmile uživatel začne kreslit, vytvoří se instance třídy **DrawLine**. Při každém kliknutí uživatele se ukládají tyto informace:

- Bod, na který uživatel klikl.
- Barva, kterou měl uživatel zvolenou při kliknutí.
- Šířku pera, kterou měl uživatel zvolenou při kliknutí.

Po ukončení použití nástroje „lomená čára“ – uživatel podrží svůj prst na obrazovce delší dobu, je instance přidána do statického listu **drawLineList**. Každá dokončená lomená čára představuje jednu položku v tomto listu. Tento list nám slouží jak pro účely, kdy dojde k překreslení obsahu, tak pro funkci undo a redo, která je popsána níže 5.4.1. Samotné kreslení lomených čar probíhá v metodě *drawLine()* za použití stejnojmenné knihovní funkce. Pseudokód metody *drawLine()* je ukázán na kódu 5.3.

```
if drawLineList is not empty
    for (int i = 0; i < drawLineList.count(); i++)
        for (int j = 0; j < (pointList.count() - 1); j++)
            Set painter pen
            Draw line
        endfor
    endfor
endfor
```

Ukázka kódu 5.3: Pseudokód metody drawLine

5.3.3 Nástroj obdélník

Obdélník se začne kreslit, jakmile si uživatel vybere nástroj „obdélník“ a dotkne se prstem obrazovky. Dotyk prstu značí levý horní roh obdélníku. Jakmile se začne uživatel tahem prstu „pohybovat“ po displeji zařízení, začne se tvar obdélníku měnit – začne se měnit jeho výška a šířka. Obdélník je namalován, jakmile uživatel zvedne prst z displeje.

Jakmile se začne kreslit obdélník, vytvoří se instance třídy **DrawRectangle**. Při prvotním kliknutí uživatele se ukládají tyto informace:

- Bod, na který uživatel klikl, který definuje levý horní roh obdélníku.
- Barva, kterou měl uživatel zvolenou při kliknutí.
- Šířku pera, kterou měl uživatel zvolenou při kliknutí.

Při přesunu prstu uživatele po displeji dochází k překreslení obsahu. Po ukončení použití nástroje „obdélník“ – uživatel zvedne prst z displeje zařízení, je instance přidána do statického listu **drawRectangleList**. Každý dokončený obdélník představuje jednu položku v tomto listu. Tento list nám slouží jak pro účely, kdy dojde k překreslení obsahu, tak pro funkci

undo a redo, která je popsána níže 5.4.1. Samotné kreslení obdélníku probíhá v metodě *drawRectangle()* za použití knihovní funkce *drawRect()*. Pseudokód metody *drawRectangle()* je ukázán na kódu 5.4

```
if drawRectangleList is not empty
    for (int i = 0; i < drawRectangleList.size(); i++)
        Set painter pen
        Draw rectangle
    endfor
endfor
```

Ukázka kódu 5.4: Pseudokód metody drawRectangle

5.4 Funkce

Za funkce považujeme – undo, redo a smazání celé plochy. Použití a implementace těchto funkcí je popsáno v následujících odstavcích.

5.4.1 Undo a Redo

Funkce undo slouží pro návrat o jeden krok zpět. Funkce redo se posune o jeden krok dopředu. Implementace obou metod – *undo()* a *redo()* se nachází ve třídě **ImageCanvas**. Obě metody využívají tří statických listů, ve kterých jsou uloženy jednotlivé instance anotačních nástrojů.

- sribblingList - zde jsou uloženy instance pro jednotlivé tahy štětcem.
- drawLineList - zde jsou uloženy instance namalovaných lomených čar.
- drawRectangleList - zde jsou uloženy instance namalovaných obdélníků.

Vždy po tahu štětcem, nakreslení lomené čáry nebo obdélníku, se do proměnné usedTool, která je typu QList přidá nástroj, který byl použit. Když uživatel požaduje funkci undo, podívame se, jaký nástroj byl naposledy použit, projdeme statické listy a smažeme poslední instanci požadovaného nástroje. Před tím si však tuto instanci musíme uchovat a poznačit si, o jaký nástroj jde pro případné redo. Pro případné redo jsou zde zase tři statické proměnné – redoSribblingList, redoDrawLineList, redoDrawRectangleList, kde se instance uchovávají. Při požadavku na redo se projdou tyto listy, vezme se poslední instance, přidá se k aktuálnímu listu a dojde k překreslení obsahu.

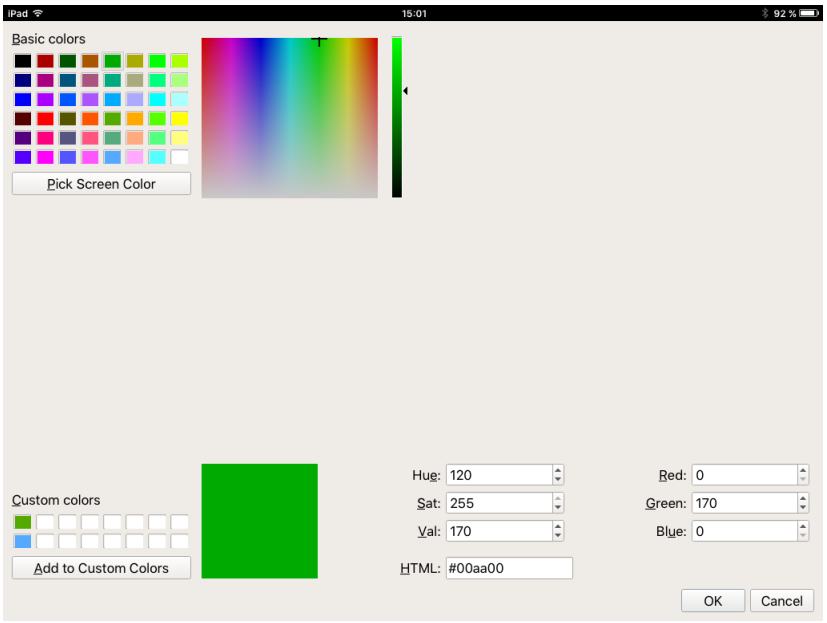
5.4.2 Smazání plochy

Při anotování se nám může stát, že chceme smazat všechny křivky, tvary, obrazce a začít anotovat obrázek od začátku. Tato funkce je zde implementována hlavně z důvodu, aby uživatel, který se chce vrátit na začátek, nemusel mnohokrát klikat na tlačítko undo. Stačí pouze jednou kliknout na tlačítko pro smazání plochy a uživatel může začít anotovat od začátku. Smazání plochy zajišťuje metoda *clearPaintingArea()*, která je implementována ve třídě **ImageCanvas**. Tato metoda smaže všechny listy, ve kterých jsou uloženy informace o všech doposud namalovaných tvarech a obrazcích. Poté se zavolá knihovní funkce *update()*, která celý obsah překreslí.

5.4.3 Výběr barvy

V mnoha případech potřebujeme anotované objekty rozlišit podle barvy, například kvůli lepšímu rozpoznání anotovaných objektů. Z tohoto důvodu je součástí aplikace dialog pro výběr barvy. Zobrazení a skrytí tohoto dialogu zajišťují metody *chooseColorAction()* a *hideColorDialog()*. Tento dialog je implementován pomocí komponenty *QColorDialog* a je ukázán na obrázku 5.8. Mezi velké výhody této komponenty patří:

- Uživatel si může vybrat ze základních barev.
- Uživatel si může vybrat barvu přejetím prstu po barevné paletě.
- Uživatel může zadat barvu v šestnáctkovém zápisu barvy pro HTML.
- Uživatel může zadat barvu ve formátu RGB nebo HSV.
- Zvolenou barvu si může uložit a použít ji pro další obrázky.



Obrázek 5.8: Ukázka dialogu pro výběr barvy.

5.5 Ukládání anotovaných obrázků

K uložení obrázku dochází, jakmile se uživatel posune na další obrázek. Ukládá se jak originální obrázek, tak jeho anotovaná bitmapa ve formátu PNG. Na obrázku 5.9 vlevo můžeme vidět uložený originální anotovaný obrázek, vpravo je potom jeho anotovaná bitmapa. Anotované obrázky se ukládají do dokumentové složky zařízení. Adresářová struktura uložených obrázků vypadá následovně: **Dokumenty/annotated_Images/názevProjektu**.

Obrázek 5.9: Ukázka uloženého obrázku jako originálu a bitmapy.
Zdroj obrázku: https://www.flickr.com/photos/tom_hall_nz/15613249585/
Autor: Tom Hall

5.6 Ukládání projektů

Je implementováno pomocí QTextStream[11]. Jedná se o třídu poskytující rozhraní pro jednoduché čtení a psaní textových souborů. Ukládání je prováděno pokaždé, když uživatel založí projekt nebo klikne na tlačítko Zpět v anotačním dialogu. Tento způsob ukládání byl implementován z důvodu, aby nebyl uživatel stále dotazován pomocí rozhodovacích dialogů, nemusel klikat na žádná tlačítka a mohl se soustředit pouze na anotování. Projekt se tedy ukládá vždy a uživatel v něm může kdykoliv pokračovat, nebo ho kdykoliv vymazat. Při vymazání projektu dochází k odstranění celé složky s anotovanými obrázky.

Při ukládání projektu si do textového souboru uložíme tři infromace, které nám postačí pro následné pokračování nebo smazání projektu. Mezi tyto tři informace patří:

1. Název projektu.
2. Cesta ke složce, kde jsou uloženy obrázky pro anotování.
3. Index obrázku, na kterém uživatel anotování ukončil.

Když uživatel v hlavním dialogu zmáčkne tlačítko *Pokračovat v anotaci*, dojde k vytvoření instance třídy **StartAnnotatingDialog**, které se předají právě tyto informace. Uložené textové soubory s těmito daty se ukládají do dokumentové složky každého zařízení, kde je vytvořen adresář **saved_projects**. Dokumentová složka byla zvolena z jednoho hlavního důvodu, kterým je přístup ke čtení a zápisu – např. v operačním systému iOS není každý adresář určen pro čtení a zápis.

5.7 Programová dokumentace

K této aplikaci byla vygenerována programová dokumentace pomocí nástroje **Doxygen**[12]. Programovou dokumentaci můžete nalézt na přiloženém CD – annotationAppDoc.zip, nebo na adrese http://www.stud.fit.vutbr.cz/~xhrebi04/index.html. Pro spuštění dokumentace z přiloženého CD stačí rozbalit archiv annotationAppDoc.zip a otevřít soubor index.html.

[11] http://doc.qt.io/qt-4.8/qtextstream.html
[12] http://www.stack.nl/~dimitri/doxygen/

Kapitola 6

Testování

Pro testování aplikace jsem použil tzv. **Usability testing**. Detailnější popis této metody, která je určená k testování, jsem uvedl výše 4.2.2. Jak jsem již zmínil, před samotným testováním bychom si měli zodpovědět na čtyři základní otázky: [15].

1. Proč testuji?

    - Abych otestoval funkčnost vytvořené aplikace a získal zpětnou vazbu, která mi poskytne informace pro vylepšení aplikace.

Odpovědi na zbylé 3 otázky jsou identické jako v případě testování mockupů 4.2.2. Jak jsem zmínil výše 4.2.2, tak po zodpovězení na tyto otázky následuje vytvoření tzv. **screeneru** [15]. Screener pro toto testování byl identický se screenerem, který mi sloužil pro testování mockupů 4.2. Znamená to tedy, že testování se zúčastnili respondenti, kteří již měli s anotováním objektů nebo segmentů na mobilním zařízení nebo stolním počítači nějaké zkušenosti. Pro testování jsem vytvořil sadu úkolů, kterou jsem pro lepší přehled čtenáře shrnul do tabulky 6.1.

Číslo úkolu	Úkol
1	Založte nový projekt s názvem New Zealand, vyberte složku New Zealand.
2	Začněte anotovat.
3	Vyberte kreslící nástroj lomená čára a namalujte úsečku.
4	Vyberte nástroj obdélník a označte libovolný segment vrcholku hor.
5	Změňte barvu na červenou.
6	Vyberte nástroj obdélník a označte další libovolný segment vrcholku hor.
7	Vraťte se o krok zpět.
8	Posuňte se o krok dopředu.
9	Smažte všechny anotované čáry - „vyčistěte plochu“.
10	Přejděte na další obrázek.
11	Vraťte se na úvodní obrazovku aplikace.
12	Smažte projekt New Zealand.

Tabulka 6.1: Úkoly pro testování aplikace.

6.1 Nastavení testování

Testování probíhalo v klidné a tiché počítačové místnosti, kde byl minimální hluk. Zúčastnili se ho 2 respondenti a bylo prováděno na tabletu s operačním systémem iOS 9.3.1, procesorem A7 a velikostí RAM paměti 1 GB. Na stůl jsem položil tablet a vedle něj sadu úkolů 6.1, které se respondenti pokoušeli splnit. Všechny poznámky jsem si po celou dobu testování zapisoval na papír. Na konci testování jsem se každého respondenta zeptal na pár otázek, jaké byly jejich pocity a dojmy z aplikace jako celku, co se jim líbilo, a co by změnili. Tyto otázky byly identické s otázkami, které jsem respondentům pokládal při testování mockupů 4.4. Odpovědi respondentů na tyto závěrečné otázky jsem shrnul do tabulek 6.2 a 6.3.

Číslo otázky	**Odpověď**
1	Cítil jsem se dobře. S aplikací se mi pracovalo pohodlně.
2	Ano.
3	Ano.
4	Ne.
5	Přidal bych informaci, že se nacházím v dialogu pro vytvoření projektu.

Tabulka 6.2: Zodpovězené závěrečné otázky od respondenta 1.

Číslo otázky	**Odpověď**
1	S aplikací jako celku se mi pracovalo dobře.
2	Ano.
3	Ano.
4	Ne.
5	Při smazání celé plochy bych se uživatele znovu dotázal.

Tabulka 6.3: Zodpovězené závěrečné otázky od respondenta 2.

6.2 Vyhodnocení testování

Analyzoval jsem poznámky, které jsem při testování zaznamenal a ke každé poznámce jsem doplnil prioritu pro odstranění tohoto „nedostatku“ a návrh možného řešení. Priorita 1 značí nejnižší prioritu, priorita 3 značí nejvyšší prioritu. Moje poznámky z testování jsem shrnul do tabulek 6.4 a 6.5.

Poznámka	**Priorita**	**Řešení/Doporučení**
Uživatel kreslí úsečku tahem.	-	Informace o chování uživatele.
Chybí informace, že se nacházím v NewProjectDialogu.	2	Doporučení pro zlepšení.

Tabulka 6.4: Ukázka výpisu poznámek od respondenta 1.

Poznámka	Priorita	Řešení/Doporučení
Před smazáním celého obsahu se uživatele znovu dotázat.	2	Doporučení pro zlepšení.

Tabulka 6.5: Ukázka výpisu poznámek od respondenta 2.

6.3 Úprava aplikace

Analyzoval jsem poznámky od respondentů, které jsem si sepsal, viz. tabulky 6.4, 6.5.

Respondent 1 při pokusu o splnění třetího úkolu 6.1 maloval úsečku pomocí tahu prstem po displeji tabletu. Tahem po displeji se uživateli žádná úsečka nevykreslila. Ihned po tomto neúspěšném pokusu se uživatel pokusil namalovat úsečku pomocí kliknutí na 2 body na displeji, po kterém již k namalování úsečky došlo. Po konci testování jsem se respondenta 1 zeptal, co by v aplikaci vylepšil nebo změnil. Respondent 1 mi doporučil zobrazit informaci o tom, že se nacházím v dialogu s novým projektem. Když anotujeme a chceme se vrátit zpět, nevíme, jestli jsme již na úvodní obrazovce celé aplikace nebo ne. Z toho důvodu byl do **NewProjectDialogu** přidán widget infoLabel, který zobrazuje informaci, v jakém dialogu se uživatel právě nachází. S touto informací uživatel hned ví, že je v dialogu, který se týká založení nového projektu a může pohodlně založit nový projekt.

Respondent 2 neměl během plnění zadaných úkolů 6.1 žádné problémy. Po konci testování jsem se respondenta 2 zeptal, co by v aplikaci vylepšil nebo změnil. Respondent 2 mi doporučil před smazáním celého obsahu se uživatele prvně dotázat, zda-li tuto akci chce opravdu provést. Na první pohled se může zdát, že dotazovací dialog bude zbytečně navíc a bude uživatele jen „obtěžovat“. Při podrobnějším zamyšlení dojdeme k opaku tohoto tvrzení, a to hned ze dvou důvodů. Tím prvním je fakt, že smazání celého obsahu je nevratná akce, a proto bychom se měli uživatele ještě jednou dotázat. Druhým důvodem je fakt, že aplikace je určena primárně k používání na tabletu. Každý člověk má jinak velké prsty a ikona smazání celého obsahu je umístěna hned vedle funkce *redo* a *výběr barvy*. Uživatel s většími prsty tak velmi lehce může omylem „kliknout“ na ikonu pro smazání celého obsahu, ikdyž požadoval akci úplně jinou. Z toho důvodu byl vytvořen **DecisiveDialog**, který se po zmáčknutí na tlačítko smazat plochu uživatele dotáže, jestli chce akci opravdu provést.

Obě doporučení, které mi respondenti poskytli po konci testování, a které jsem zmínil v předešlých odstavcích, jsou ve výsledné aplikaci opraveny.

6.4 Domácí testování

Jak jsem popsal v předcházejících odstavcích, po dokončení aplikace jsem provedl testování výsledné aplikace. Nezávazně na tomto testování jsem udělal ještě domácí testování. Cílem bylo otestovat aplikaci na co možná nejvíce různých operačních systémech a zařízeních. Hlavní faktor, který jsem pozoroval, byl výkon aplikace, jak z pohledu aplikace jako celku, tak hlavně z pohledu rychlosti vykreslení jednotlivých tvarů – obdélník, čára, tah štětcem.

Nastavení domácího testování

Testování bylo prováděno na notebooku, stolním počítači a tabletu. Na těchto zařízeních byly nainstalovány operační systémy OS X 10.11.4, Windows 7, Xubuntu 14.04 a iOS 9.3.1.

Testoval jsem, za jak dlouhý časový interval dojde k vykreslení 1000 obdélníků. Pro měření rychlosti vykreslení obdélníků jsem použil třídu **QTime**[1], která poskytuje funkce pro práci s časem. Výsledky tohoto testování včetně naměřených časových údajů pro jednotlivá zařízení a operační systémy jsem shrnul do tabulky 6.6.

Zařízení	**Operační systém**	**Procesor**	**RAM**	**Čas**
Notebook	OS X 10.11.4	i5 (1,4 GHz)	8 GB	19 ms
Stolní PC	Windows 7	i7 (2,8 GHz)	4 GB	109 ms
Stolní PC	Xubuntu 14.04	i7 (2,8 GHz)	4 GB	31 ms
Tablet	iOS 9.3.1	A5	0,5 GB	334 ms
Tablet	iOS 9.3.1	A7	1 GB	157 ms

Tabulka 6.6: Srovnání rychlosti vykreslení na různých systémech.

Vyhodnocení domácího testování

Tabulka 6.6 ukazuje, že vykreslení na tabletu s operačním systémem iOS 9.3.1, procesorem A5 a velikostí RAM paměti 0,5 GB bylo až 17,58x pomalejší, než vykreslení na stolním počítači s operačním systémem OS X 10.11.4, procesorem i5 a velikostí RAM paměti 8 GB. Přičina tohoto problému spočívá v neustálém překreslování anotovaných tvarů – tahy štětcem, lomená čára, obdélník na QPixmapu. Neustálé překreslování obsahu na QPixmapu spotřebovává mnoho CPU, a z tohoto důvodu není vykreslování na tabletu se slabším procesorem a RAM pamětí tak plynulé, jak bychom očekávali.

Testování ukázalo, že pro budoucí vývoj aplikace by mělo dojít k optimalizaci vykreslování, aby se zrychlilo kreslení i na zařízeních se slabším procesorem a menší RAM pamětí. Jednou z možností, jak toho v budoucnu dosáhnout, je změna widgetu, který slouží pro kreslení. Místo jednoho widgetu by došlo k vytvoření 2 widgetů, přičemž první widget by pouze zobrazoval obrázek, a na druhém by probíhalo kreslení. Můžeme si to představit jako dvě vrstvy, které by byly nad sebou. Vrchní průhledná vrstva by sloužila pro kreslení. Opticky by to vypadalo, jako že se kreslí na obrázek, ale kreslilo by se pouze na widget. Díky tomu by nedocházelo k opakovanému překreslení QPixmapy. Pro další urychlení a plynulost kreslení by se do budoucna mohlo kreslení implemetovat pomocí OpenGL.

[1] http://doc.qt.io/qt-4.8/qtime.html

Kapitola 7

Závěr

Cílem práce bylo navrhnout, vytvořit a otestovat multiplatformní mobilní anotační aplikaci. V první fázi byla provedena řešerše 2 základních metod a postupů, které byly pro tuto práci důležité. Dále byly porovnány jak existující anotační aplikace, tak aplikace, které se zaměřují na kreslení.

Pro návrh designu aplikace byly vytvořeny mockupy 4.1. Mockupy byly vytvořeny jak pro mobilní zařízení 4.1, tak pro tablet 4.2. Hlavním důvodem výběru mockupů byl fakt, že jsem chtěl získat informaci, zda dává návrh rozhraní smysl, abych zbytečně neimplementoval produkt, který bude nepoužitelný. Mockupy byly následně otestovány respondenty, a díky zpětné vazbě byly mockupy upraveny a předělány do podoby, která se blížila výsledné aplikaci. Testování probíhalo pomocí metody Usability testing 4.2.2.

Pro anotační aplikaci byly implementovány nástroje štětec 5.3.1, lomená čára 5.3.2 a obdélník 5.3.3. Každý tah štětcem, lomená čára nebo obdélník může mít jinou tloušťku a barvu 5.4.3. Dále byly implementovány funkce undo a redo 5.4.1, díky kterým se může uživatel vrátit o krok zpět, nebo posunout o krok dopředu v samotné anotaci. Dále byla implemetována funkce, která smaže veškerý anotovaný obsah a uživatel tak může začít pohodlně anotovat od začátku 5.4.2. Každý anotovaný obrázek je ukládán jak ve formě originálního snímku, tak ve formě anotované bitmapy ve formátu PNG 5.5. Dále bylo implementováno ukládání projektů 5.6. V každém uloženém projektu lze kdykoliv pokračovat, nebo ho smazat.

Po samotné implementaci bylo provedeno testování aplikace 6, které probíhalo metodou Usability testing. Nezávazně na tomto testování bylo provedeno ještě domácí testování, které se zabývalo výkonem aplikace na různých zařízeních 6.4. Při tomto testování bylo zjištěno, že aplikace a hlavně samotné kreslení není na všech zařízeních stejně plynulé. Z důvodu této skutečnosti byl proveden návrh možné implementace do budoucnosti, díky které by došlo k urychlení kreslení 6.4.

Dalším možným rozšířením do budoucna by mohlo být implementování funkce *guma* a možnost posunutí anotovaných obdélníků. Při využití funkce *guma* by uživatel kromě návratu o krok zpět, posunutí se o krok dopředu, nebo smazání celé plochy, mohl smazat i konkrétní části anotovaných tvarů. Implementace možnosti posunu jednotlivých obdélníků by sloužila k přesnější a pohodlnější anotaci.

Literatura

[1] Ames, M.; Naaman, M.: Why we tag: motivations for annotation in mobile and online media. 2007: s. 971–980.

[2] Beran, V.: Úvod do (grafických) uživatelských rozhraní [online]. https://www.fit.vutbr.cz/study/courses/ITU/private/lectures/Intro/itu-gui-cs.uvod.pdf, 2015, [cit. 2016-05-11].

[3] Chatti, M. A.; Sodhi, T.; Specht, M.; aj.: u-Annotate: An Application for User-Driven Freeform Digital Ink Annotation of E-Learning Content. červenec 2006: s. 1039–1043.

[4] Ericsson, K. A.; Simon, H. A.: *Protocol Analysis: Verbal Reports as Data*. Cambridge: The MIT Press, revised vydání, 1984, ISBN 978-0262550239.

[5] Jurzykowski, M.: ITU uživatelská rozhraní Qt [online]. https://www.fit.vutbr.cz/study/courses/ITU/private/lectures/Qt/itu-qt-cs.pdf, 2014, [cit. 2016-05-10].

[6] Kaikkonen, A.; Kekäläinen, A.; Cankar, M.; aj.: Usability testing of mobile applications: a comparison between laboratory and field testing. *Journal of Usability Studies*, ročník 1, č. 1, listopad 2005: s. 4–16.

[7] Klimeš, L.: *Slovník cizích slov*. Praha: Státní pedagogické nakladatelství, třetí vydání, 1985, ISBN 14-621-85.

[8] Křena, B.; Kočí, R.: Úvod do softwarového inženýrství [online]. https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IUS-IT/lectures/IUS3.pdf?cid=9410, 2015, [cit. 2016-05-11].

[9] Lanoue, S.: What is UX Design? [online]. https://www.usertesting.com/blog/2015/09/16/what-is-ux-design-15-user-experience-experts-weigh-in/, 2015, [cit. 2016-03-07].

[10] Linhart, J.: *Slovník cizích slov pro nové století*. Litvínov: Dialog, první vydání, 2003, ISBN 80-85843-61-7.

[11] Neustadt, I.; Arlow, J.: *UML 2 a unifikovaný proces vývoje aplikací*. Brno: Computer Press, druhé vydání, 2007, ISBN 978-80-251-1503-9.

[12] Petráčková, V.; Kraus, J.: *Akademický slovník cizích slov*. Praha: Academia, první vydání, 1998, ISBN 80-200-0607-9.

[13] Rose, E.; Breen, D.; Ahlers, K. H.; aj.: Annotating Real-World Objects Using Augmented Reality. červen 1995: s. 357–370.

[14] Rumbaugh, J.; Jacobson, I.; Booch, G.: *The Unified Modeling Language Reference Manual*. Boston: Pearson Education, druhé vydání, 2005, ISBN 0-321-24562-8.

[15] Unger, R.; Chandler, C.: *A Project Guide to UX Design For User Experience Designers in the Field or in the Making*. Berkeley: New Riders, druhé vydání, 2012, ISBN 0-321-81538-6.

[16] Wilhelm, A.; Takhteyev, Y.; Sarvas, R.; aj.: Photo annotation on a camera phone. 2004: s. 1403–1406.

[17] Zhang, D.; Adipat, B.: Challenges, Methodologies, and Issues in the Usability Testing of Mobile Applications. *International Journal of Human-Computer Interaction*, ročník 18, č. 3, 2005: s. 293–308.

Přílohy

Seznam příloh

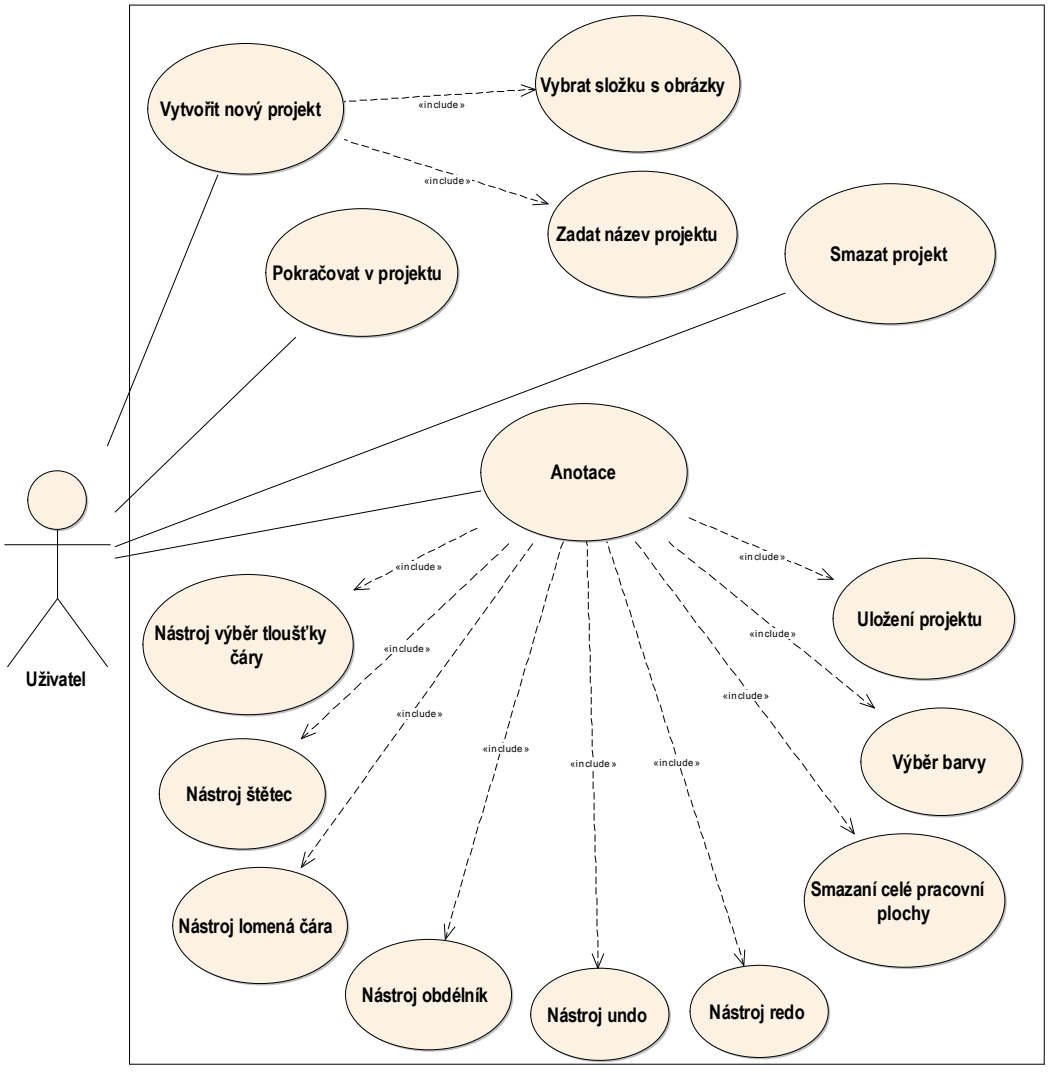
Příloha A

Obsah CD

- projekt.pdf – Technická zpráva ve formátu PDF.
- projektTisk.pdf – Technická zpráva ve formátu PDF určená pro tisk.
- projekt.zip – Zdrojové soubory technické zprávy.
- annotationApp.zip – Zdrojové soubory anotační aplikace (*.cpp, *.h).
- annotationAppDoc.zip – Programová dokumentace anotační aplikace.

Příloha B

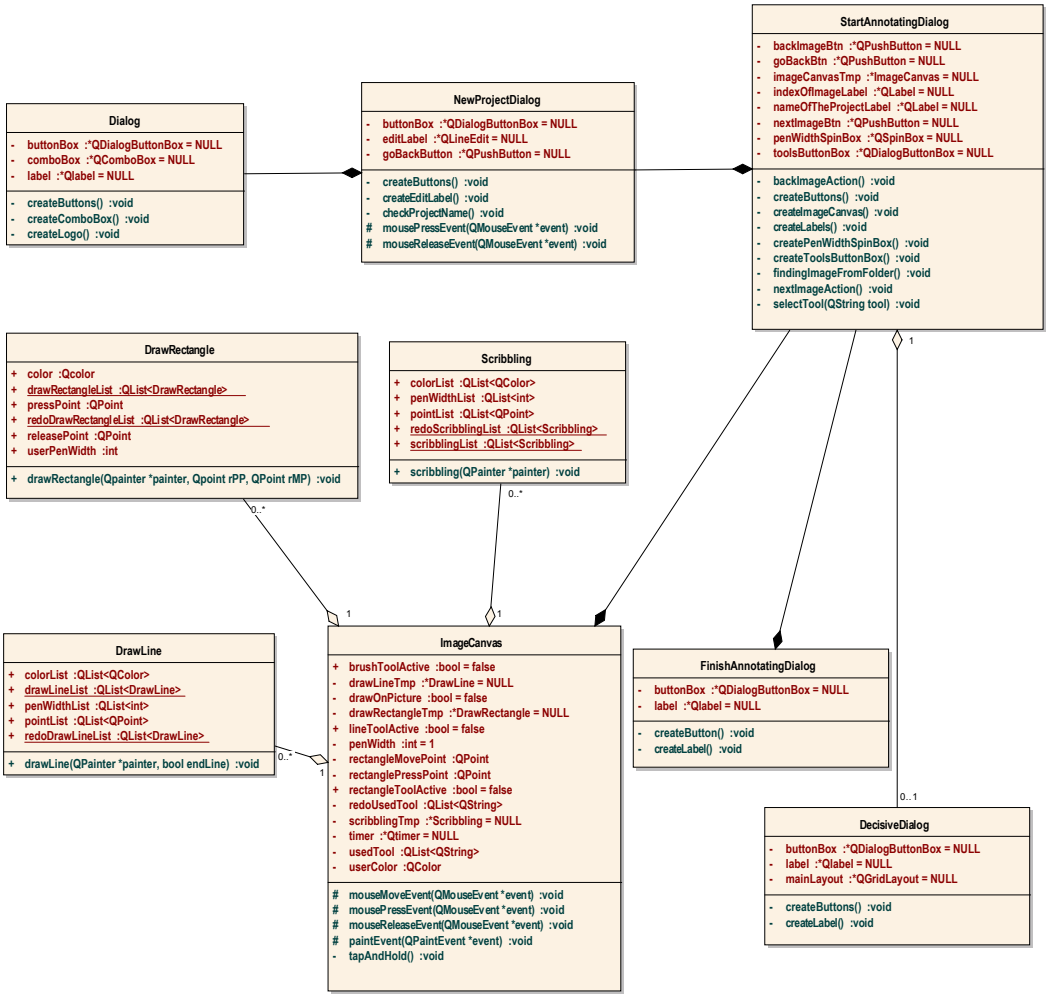
Diagram případů užití



Obrázek B.1: Diagram případů užití funkčních požadavků.

Příloha C

Třídní diagram



Obrázek C.1: Třídní diagram.

Příloha D

Instalace

Uvedený postup byl vyzkoušen pro operační systémy OS X 10.11.4, Xubuntu 14.04, Windows 7. Níže uvedený postup stáhne a nainstaluje Qt Framework spolu s vývojovým prostředím Qt Creator.

1. Přejděte na adresu http://www.qt.io/download/.
2. Na otázku *There are a few ways you can get started with Qt. To help us find the option for you, tell us what your application or device will be developed for...* zaklikněte **Open source distribution under a LGPL or GPL license**.
3. Na otázku *Are you prepared to make your application source code publicly available?* zaklikněte **Yes**.
4. Na otázku *Are you able to comply with the obligations of the LGPL (or GPL) and/or does your corporation allow open source usage?* zaklikněte **Yes**.
5. Klikněte na tlačítko **Get started**.
6. Klikněte na tlačítko **Download Now**.
7. Stažený soubor nainstalujte.
8. Po stažení a nainstalování máme k dispozici jak Qt Framework, tak vývojové prostředí Qt Creator.

D.1 Možné problémy

1. Pokud používáte počítač s operačním systémem Linux, stáhne se soubor s koncovkou *.run. Tento soubor spustíte následovně: *klikněte na něj pravým tlačítkem -> Properties (Vlastnosti) -> Permissions (Oprávnění) -> zaškrtněte volbu Allow executing file as program (Povolit tomuto souboru spuštění)* a začněte instalovat.

Příloha E

Postup spuštění

Uvedený postup předpokládá nainstalovaný Qt Framework a vývojové prostředí Qt Creator. Instalace je popsána v příloze D.

1. Rozbalte si archiv annotationApp.zip, který je přiložen na CD a přesuňte složku annotationApp na plochu vašeho počítače.

2. Otevřete nainstalované vývojové prostředí Qt Creator.

   - Klikněte na tlačítko otevřít projekt, nebo v levém horním menu *Soubor -> Otevřít soubor nebo projekt... -> Plocha -> annotationApp -> annotationApp.pro*. Otevřít projekt lze i za pomocí klávesových zkratek pro Windows a Linux *CTRL+O*, pro OS X *CMD+O*.
   - Následně budete vyzváni s dotazem, jaké „kity“ chcete nainstalovat, nebo-li pro jaké zařízení chcete projekt přeložit. Zaškrtněte Desktop, pokud máte operační systém OS X, tak zaškrtněte také kity pro iOS. Kity pro operační systém android můžete volitelně doinstalovat později. Klikněte na tlačítko *Nastavit projekt*. Po tomto kroku by se měla ve vývojovém prostředí Qt Creator vytvořit adresářová struktura ze souborů *.cpp a *.h.
   - V levém postranním panelu vývojového prostředí Qt Creator klikněte na *projekty* a **odznačte** možnost *Stínové sestavování* u záložky Desktop.
   - V levém postranním panelu vývojového prostředí Qt Creator klikněte na zelenou šipku spustit.
   - Pokud se aplikace spustí, tak proběhl překlad v pořádku. Aplikace je nyní zcela připravena pro otestování na stolním počítači. V opačném případě se prosím podívejte do sekce E.1, kde jsou vypsány možné problémy, které mohly při postupu vzniknout.

Jak jsem se zmínil v sekci 3, aplikaci je možné používat na stolním počítači, primárně je však určena pro používání na tabletech. Po provedení kroku 1 a 2, které jsou popsány výše, je aplikace připravena pro nasazení na mobilní zařízení – tablet. V následujících odstavcích bude detailněji popsán postup pro spuštění aplikace na tabletu s operačním systémem iOS.

Mobilní zařízení

Pro spuštění aplikace na tabletu s operačním systémem iOS je potřeba:

1. Počítač s nainstalovaným operačním systémem OS X.
2. Vývojové prostředí Xcode.
3. Software iTunes pro přenos složek s obrázky mezi počítačem a aplikací.
4. Apple ID.

Uvedený postup byl vyzkoušen na počítači s nainstalovaným operačním systémem OS X 10.11.4, vývojovým prostředím Xcode 7.3 a softwarem iTunes 12.3.3.17. Aplikace byla vyzkoušena pro tablet iPad Mini s operačním systémem iOS 9.3.1. Postup spuštění je následující:

1. Připojte tablet do počítače s operačním systémem OS X.
2. V levém postranním panelu vývojového prostředí Qt Creator změňte sadu na *iphoneos-clang*.
3. V levém postranním panelu vývojového prostředí Qt Creator klikněte na *projekty* a **odznačte** možnost *Stínové sestavování* u záložky *Iphoneos-clang*.
4. V levém postranním panelu vývojového prostředí Qt Creator klikněte na zelenou šipku spustit. Pokud vše proběhlo v pořádku, měli byste na svém iPadu vidět přidanou novou aplikaci s názvem *annotationApp*.
5. Na vašem počítači přejděte do projektové složky annotationApp a otevřete vygenerovaný soubor *annotationApp.xcodeproj* ve vývojovém prostředí Xcode.
6. Ve vývojovém prostředí Xcode klikněte na *Xcode -> Preferences -> Accounts*. Zde přidejte vaše Apple ID. Po přidání byste měli vidět, že Vám přibyl developerský účet.
7. V levém postraním panelu vývojového prostředí Xcode klikněte na soubor annotationApp. V sekci *General -> Identity -> Team* zadejte váš developerský účet, který jste si vytvořili v předešlém kroku.
8. Nyní musíte ve vašem iPadu nastavit, aby zařízení aplikaci důvěřovalo. V Tabletu klikněte na *Nastavení -> Obecné -> Správa zařízení -> Aplikace vývojáře -> Důvěřovat vývojáři ->Důvěřovat*.
9. Přejděte do vývojového prostředí Xcode. Vyberte vaše zařízení a spusťte aplikaci. Po provedení tohoto kroku by se Vám měla aplikace na vašem tabletu spustit. Pro samotné anotování však potřebuje nahrát složku s anotovanými obrázky do Vašeho zařízení. Tento postup je detailněji vysvětlen v následujících odstavcích.

Přenos složky s obrázky do zařízení

1. Na vašem počítači přejděte do projektové složky annotationApp a otevřete vygenerovaný soubor *Info.plist* ve vývojovém prostředí Xcode. V levém sloupci vyberte možnost *Application supports iTunes file sharing*. V pravém sloupci přehoďte hodnotu Value z **NO** na **YES**. Uložte soubor Info.plist.
2. Znovu spusťte aplikaci ve vývojovém prostředí Xcode.

3. Otevřete software iTunes, vyberte Vaše zařízení a klikněte na položku *Aplikace*. Ve spodní části otevřené nabídky by se měla ukázat možnost *Sdílení souborů* a pod ní název aplikace *annotationApp*.

4. Klikněte na možnost *Přidat...* a vyberte složku z vašeho počítače, ve které jsou obrázky k anotování.

5. Nyní je vše připraveno ke spuštění a otestování aplikace na tabletu.

E.1 Možné problémy

1. Překlad grafických aplikací vyžaduje OpenGL. Většina linuxových distribucí tuto komponentu v základu neinstaluje. Více informací lze nalézt na adrese `http://doc.qt.io/qt-5/linux.html`. Překladač hlásí převážně 2 chyby:

   - **g++: Command not found.**
   - **GL/gl.h: No such file or directory.**

   Pro vyřešení těchto problémů postupujte podle operačního systému, který máte na svém počítači.

   Debian/Ubuntu (apt-get)

   - sudo apt-get install build-essential libgl1-mesa-dev

   Fedora/RHEL/CentOS (yum)

   - sudo yum groupinstall
   - sudo yum install mesa-libGL-devel

   Po provedení těchto příkazů a nainstalování balíčků restartujte vývojové prostředí Qt Creator a zkuste aplikaci znovu spustit.