



# MOBILNÍ APLIKACE PRO DETEKCI BAREV A VIZUÁLNÍ NÁVRH BAREVNÝCH SCHÉMAT

Šárka Prokopová

Bakalářská práce  
Fakulta informačních technologií  
České vysoké učení technické v Praze  
Katedra softwarového inženýrství  
Studijní program: Informatika  
Specializace: Softwarové inženýrství  
Vedoucí: doc. Ing. Marek Suchánek, Ph.D. et Ph.D.  
2. ledna 2026



## Zadání bakalářské práce

<b>Název:</b>	Mobilní aplikace pro detekci barev a vizuální návrh barevných schémat
<b>Student:</b>	Šárka Prokopová
<b>Vedoucí:</b>	Ing. Marek Suchánek, Ph.D. et Ph.D.
<b>Studijní program:</b>	Informatika
<b>Obor / specializace:</b>	Počítačová grafika 2021
<b>Katedra:</b>	Katedra softwarového inženýrství
<b>Platnost zadání:</b>	do konce letního semestru 2026/2027

### Pokyny pro vypracování

Se stále rostoucí oblíbeností a důležitostí vizuálního obsahu potřebují designéři, umělci a tvůrci obsahu nástroj, který jim umožní snadno a rychle identifikovat barvy a vytvářet soudržná barevná schémata. Cílem této práce je vyvinout multiplatformní mobilní aplikaci, která snadno umožní detektovat barvy ve fotografiích a nabízet vhodné barevné kombinace. Navíc taková aplikace může být užitečná i pro jedince s barvoslepotí nebo zrakovým postižením, kteří mohou mít potíže s rozlišováním barev. V neposlední řadě se v rámci takové aplikace dá zohlednit i použití metod rozšířené reality (AR).

- Analyzujte a popište barevné modely spolu s teorií a vnímáním barev lidmi, metody tvorby barevných palet, známé sady barev (např. Pantone či pojmenované barvy HTML) a možnosti extrakce informací o barvách z fotografií.
- Prozkoumejte také možnosti využití AR pro práci s barevami v rámci práce s mobilními zařízeními a fotografiemi.
- Proveďte rešerši existujících řešení, které se zabývají extrakcí barev z fotografií za účelem tvorby barevných palet, hledáním shodných barev ze známých barevných sad či využívají AR pro obdobnou práci s barevami.
- Sestavte požadavky na vlastní řešení ve formě multiplatformní mobilní aplikace podporující Android a iOS (v aktuálních verzích).
- Navrhněte architekturu aplikace v souladu s požadavky a zvolenými technologiemi. Volbu technologií zdůvodněte v kontextu požadavků. Dále navrhněte také uživatelské rozhraní tak, aby ovládání bylo intuitivní a uživatelsky přívětivé. Při návrhu zohledněte



možný budoucí rozvoj (jiná logika tvorby palet, nové sady barev atd.), popište použité návrhové vzory.

- Implementujte aplikaci dle návrhu, při vývoji aplikujte osvědčené postupy doporučené pro zvolené technologie (dokumentace, CI/CD, konvence stylu kódu, statická analýza apod.).
- Výslednou aplikaci řádně otestujte pomocí uživatelského testování. Pro aplikační logiku připravte také automatické (jednotkové) testy.
- Zhodnotte přínosy aplikace, porovnejte ji s konkurenčními řešeními a navrhněte další možný rozvoj.



České vysoké učení technické v Praze

Fakulta informačních technologií

© 2025 Šárka Prokopová. Všechna práva vyhrazena.

*Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení, je nezbytný souhlas autora.*

Odkaz na tuto práci: Prokopová Šárka. *Mobilní aplikace pro detekci barev a vizuální návrh barevných schémat*. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2025.

*Chtěla bych poděkovat především svému vedoucímu, Ing. Marku Sucháňkovi, Ph.D. et Ph.D., za vřelý přístup, odborné vedení bakalářské práce, trpělivost a cenné rady při komplikacích objevených v celém procesu vytváření této práce. Zároveň mé velké díky patří sestře a přátelům za mentální podporu v průběhu studia a mému partnerovi za rady a pomoc v nejvytíženějších chvílích.*

## Prohlášení

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

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

V Praze dne 2. ledna 2026

## **Abstrakt**

Tato bakalářská práce se věnuje vývoji multiplatformní mobilní aplikace pro detekci barev a vizuální návrh barevných schémat s dodržením tradičních postupů softwarového inženýrství. Součástí práce je analýza teorie barev, vnímání barev spolu s představením známých barevných sad a popsáním tvorby barevných palet. Provedena je také rešerše existujících řešení a využití metod rozšířené reality.

Na základě analýzy jsou sestaveny požadavky aplikace, dle nichž se realizoval návrh. Ten je doprovázen implementací aplikace a jejím testováním. Konec práce se věnuje zhodnocení výsledků a popisu možného rozvoje aplikace.

Výstupem této práce je funkční mobilní aplikace umožňující rozpoznávat barvy a na jejich základě vytvářet barevné palety.

**Klíčová slova** teorie barev, vnímání barev, mobilní aplikace, AR, extrakce barev, barevné palety, detekce barev

## **Abstract**

This bachelor thesis focuses on the development of a multiplatform mobile application for colour detection and visual design of colour schemes following traditional software engineering principles. The thesis includes an analysis of colour theory, colour perception along with an introduction of recognized colour sets and a description of the creation of colour palettes. A survey of existing solutions and the use of augmented reality methods is also presented.

On the basis of the analysis, the application requirements are established, according to which the design is developed. This is followed by the implementation of the application and its testing. The end of the thesis is dedicated to the evaluation of the results and description of the possible development of the application.

The output of this work is a functional mobile application that enables the recognition of colours and to create colour palettes based on the colours.

**Keywords** colour theory, perception of colours, mobile application, AR, colour extraction, colour palettes, colour detection

# Obsah

<b>Úvod</b>	<b>1</b>
<b>1 Cíl práce</b>	<b>1</b>
<b>2 Analýza</b>	<b>2</b>
2.1 Teorie barev . . . . .	2
2.1.1 Barva . . . . .	2
2.1.1.1 Isaac Newton . . . . .	3
2.1.1.2 Johann Wolfgang von Goethe . . . . .	3
2.1.1.3 Young-Helmholtzova teorie . . . . .	4
2.1.1.4 Vlastnosti barev . . . . .	5
2.1.1.5 Moderní definice . . . . .	5
2.1.2 Oko a barevné vidění . . . . .	6
2.1.2.1 Vliv kontextu na vnímání barev . . . . .	7
2.1.3 Vliv barev v běžném světě . . . . .	9
2.1.4 Poruchy barevného vidění . . . . .	10
2.1.4.1 Vrozené poruchy barevného vidění . . . . .	10
2.1.4.2 Získané poruchy barevného vidění . . . . .	11
2.1.4.3 Vyšetření barvocitu . . . . .	12
2.1.5 Teorie barev . . . . .	12
2.1.5.1 Kategorizace barev . . . . .	13
2.1.5.2 Harmonie barev . . . . .	13
2.1.5.3 Barevný kontext . . . . .	14
2.2 Barevné systémy a tvorba barevných palet . . . . .	14
2.2.1 Míchání barev . . . . .	15
2.2.2 Základní barevné systémy . . . . .	15
2.2.2.1 RGB . . . . .	16
2.2.2.2 CMYK . . . . .	16
2.2.2.3 HSV . . . . .	16
2.2.2.4 Pantone . . . . .	16
2.3 Rozšířená realita a její využití . . . . .	17
2.3.1 Virtuální realita vs rozšířená realita . . . . .	17
2.3.2 Princip fungování . . . . .	17
2.3.3 Možnosti využití . . . . .	18
2.3.4 Druhy rozšířené reality . . . . .	18
2.4 Rešerše existujících řešení . . . . .	19

2.4.1	Color Blind Pal . . . . .	19
2.4.2	Color Identifier: Color Picker . . . . .	19
2.4.3	Paleto . . . . .	21
2.4.4	Color Name Recognizer . . . . .	21
2.4.5	Shrnutí . . . . .	21
2.5	Požadavky . . . . .	23
2.5.1	Funkční požadavky . . . . .	23
2.5.2	Nefunkční požadavky . . . . .	24
2.6	Model případů užití . . . . .	25
2.6.1	Seznam případů užití . . . . .	25
2.6.2	Diagram případů užití . . . . .	26
2.7	Stavový diagram . . . . .	26
2.7.1	Stav . . . . .	26
2.7.2	Přechod stavů . . . . .	28
2.7.3	Událost . . . . .	28
<b>3</b>	<b>Návrh</b>	<b>30</b>
3.1	Technologie . . . . .	30
3.1.1	Nativní vs multiplatformní vývoj . . . . .	30
3.1.2	Vývojové prostředí . . . . .	31
3.1.3	Výběr frameworku . . . . .	31
3.1.4	Shrnutí výběru frameworku . . . . .	32
3.1.5	Expo . . . . .	32
3.2	Architektura a architektonické vzory . . . . .	32
3.2.1	MVC . . . . .	33
3.2.2	MVVM . . . . .	33
3.2.3	Component-based . . . . .	33
3.2.4	Výhody a nevýhody zmíněných architektonických vzorů	34
3.2.5	Shrnutí a výběr architektury a architektonického vzoru	35
3.3	Uživatelské rozhraní . . . . .	36
3.3.1	První spuštění . . . . .	36
3.3.2	Camera . . . . .	37
3.3.3	Library . . . . .	37
3.3.4	Detail . . . . .	38
3.3.5	Help . . . . .	39
3.3.6	Shrnutí návrhu uživatelského rozhraní . . . . .	39
<b>4</b>	<b>Implementace</b>	<b>40</b>
4.1	Continuous Integration . . . . .	40
4.2	GitHub . . . . .	40
4.3	Statická analýza . . . . .	41
4.3.1	ESLint . . . . .	41
4.3.2	Dokumentace . . . . .	42
4.4	Ukázky implementace . . . . .	42

4.4.1	App . . . . .	42
4.4.2	RootStack a HomeScreen . . . . .	43
4.4.3	CameraScreen . . . . .	43
4.4.4	SceneAR . . . . .	46
4.4.5	ColorData . . . . .	47
<b>5</b>	<b>Testování</b>	<b>51</b>
5.1	Unit testování . . . . .	51
5.1.1	Osvědčené postupy pro unit testování . . . . .	52
5.1.2	Ukázka . . . . .	53
5.2	Testování uživatelského rozhraní . . . . .	54
5.2.1	Výběr uživatelů . . . . .	54
5.2.2	Testovací scénáře . . . . .	54
5.2.3	Výsledky uživatelského testování . . . . .	56
<b>6</b>	<b>Vyhodnocení a další rozvoj</b>	<b>57</b>
6.1	Výsledky práce . . . . .	57
6.1.1	Shrnutí výsledků . . . . .	57
6.1.2	Splnění požadavků . . . . .	58
6.1.3	Porovnání s existujícími aplikacemi . . . . .	58
6.2	Možný rozvoj aplikace . . . . .	59
6.2.1	Rozvoj na základě výsledků testování . . . . .	59
6.2.2	Rozvoj na základě požadavků . . . . .	59
<b>Závěr</b>		<b>60</b>
<b>A</b>	<b>Ukázky scénářů uživatelského testování</b>	<b>61</b>
A.1	Detekování barvy v AR . . . . .	61
A.2	Uložení barvy . . . . .	61
A.3	Uložení fotografie . . . . .	62
A.4	Smažání 3 barev . . . . .	62
A.5	Zobrazení detailu právě detekované barvy . . . . .	62
<b>Obsah příloh</b>		<b>69</b>

## Seznam obrázků

2.1	Sir Isaac Newton experimentující s hranoletem. Rytina dle obrázku od J.A. Houston, ca. 1870. Se svolením The Granger Collection, New York [5] . . . . .	3
2.2	Spektrální absorpcní křivky pigmentů krátkých (S), středních (M) a dlouhých (L) vlnových délek v lidských čípkových a tyčinkových (R) buňkách. [8] . . . . .	5
2.3	Struktura tyčinek a čípků v oku. [12] . . . . .	6
2.4	Paměťová barva jahod rozpoznatelná i tehdy, když je jejich fotografie zmanipulována tak, že červená barva je šedá. [14] . . . . .	7
2.5	Příklady barevných stínů a vlivu pozadí na vnímání barvy objektu. [16] . . . . .	8
2.6	Vnímání spektra viditelného světla u zdravých osob a osob s poruchami barvocitu, včetně informace o procentuálním výskytu poruch barvocitu v populaci. [22] . . . . .	11
2.7	Příklad tabulky pro test barvocitu. [23] . . . . .	12
2.8	Paleta primárních, sekundárních a tericálních barev. [25] . . . . .	13
2.9	Ukázka komplementární a analogických barev. [26] . . . . .	14
2.10	Ukázka aditivních a substraktivních barev. [28] . . . . .	15
2.11	Pantone vzorník. [29] . . . . .	17
2.12	Ukázky aplikace Color Blind Pal. [32] . . . . .	20
2.13	Ukázky aplikace Color Identifier. [33] . . . . .	20
2.14	Ukázky aplikace Paleo[34] a Color Name Recognizer[35]. . . . .	22
2.15	Diagram případů užití dle [36]. . . . .	27
2.16	Stavový diagram aplikace ColorLensAR, dle [38]. . . . .	29
3.1	Ukázka komunikace modelu MVVM dle [46]. . . . .	34
3.2	Wireframe obrazovky s nápovědou při prvním spuštění a hlavního menu. . . . .	36
3.3	Ukázka obrazovek při spuštění Camera. . . . .	37
3.4	Ukázka obrazovek v knihovně barev. . . . .	38
3.5	Ukázka obrazovky vyobrazující detail o barvě. . . . .	38
3.6	Ukázka obrazovky při spuštění nápovědy. . . . .	39
4.1	Ukázka implementace úvodní obrazovky HomeScreen a knihovny Library. . . . .	44
4.2	Ukázka implementace ukládání fotografie. . . . .	45

4.3	Ukázka implementace obrazovky s kamerou v AR módu, výběru z galerie a potvrzení uložení obrázku. . . . .	47
4.4	Ukázka implementace obrazovky s detailem barvy, kódy a paletami. . . . .	50
5.1	Ukázka unit testů. . . . .	55

## **Seznam zkratek**

AR	Augmented Reality
CD	Continuous Deployment
CI	Continuous Integration
CMYK	Cyan, Magenta, Yellow, Key
GPS	Global Positioning System
HEX	Hexadecimal colour code
IDE	Integrated Development Environment
MoSCoW	Must have, Should have, Could have, Won't have
MVC	Model-View-Controller
MVVM	Model-View-ViewModel
RGB	Red, Green, Blue
UI	User Interface
UX	User Experience
VR	Virtuální Realita

# Úvod

Barvy a vnímání barev je i přes její důkladné zkoumání v historii stále velmi subjektivní záležitostí. Přesto se v průběhu času některým teoretikům, vědcům a filozofům podařilo stanovit určitá pravidla a vlastnosti barev, které se ve společnosti hojně využívají. Ať už se jedná o předpokládaný umělecký průmysl, v tomto smyslu nejen malba či kresba, využití v psychologii, digitálním odvětví, marketingu, či dokonce politice, barvy jsou důležitým aspektem, díky kterému můžeme v lidech probouzet emoce, prohlubovat naše myšlenky, v určitých případech je využívat také jako nástroj manipulace, nebo naopak motivace.

Barvy působí v nervovém systému. Ne každý však má možnost barvy vidět. Je tedy otázkou, jak se takový člověk může přizpůsobit dnešní společnosti a zda je možné od společnosti takovému člověku porozumět.

Hlavním cílem této bakalářské práce je proto vyvinout multiplatformní aplikaci, která s pomocí rozšířené reality umožní rozeznávat barvy objektů v reálném světě a vytvářet další barevné palety. Díky této aplikaci mohou lidé s poruchou vnímání barev rozlišovat barvy a lépe chápát návaznost na další odstíny. Zároveň je aplikace pomocníkem, díky kterému mohou umělci, tvůrci obsahu a designéři rychle detekovat barvy a vytvářet soudržná schémata.

Již existující řešení se soustředí většinou pouze na jediný aspekt, buď pouhé rozeznávání barev na fotografii či tvorbu palet na základě stanovené barvy. Pro uživatele právě sjednocení těchto funkcionalit umožňuje správné porozumění barvám a jejich vnímání. Práce se zabývá analýzou, návrhem a implementací aplikace podle tradičních postupů softwarového inženýrství.

## Kapitola 1

# Cíl práce

Hlavním cílem této práce je vývoj multiplatformní aplikace pro detekci barev a tvorbu barevných schémat, a to dle tradičních postupů softwarového inženýrství. Práce se tedy soustředí na analýzu, sestavení požadavků, návrh, implementaci a následné testování.

Prvním bodem je analýza teorie barev, vnímání barev a popsání známých barevných sad a tvorby palet. Současně se analýza zabývá extrakcí informací z fotografií, a nakonec také využitím AR pro práci s mobilním zařízením a fotografiemi.

Další část se budě věnovat rozbořem existujících řešení, které detekují barvy z fotografií či tvoří barevná schémata na základě stanovených barev.

Dále proběhne sestavení požadavků a případů užití výsledné aplikace. Bude následovat návrh aplikace, tvorba architektury a příprava konceptu uživatelského rozhraní. Dle tohoto návrhu proběhne implementace aplikace a následné testování.

Nakonec proběhne vyhodnocení výsledné aplikace spolu s dílčími cíli této práce, bude prodiskutován taktéž přínos a případný možný rozvoj aplikace.

## Kapitola 2

# Analýza

*První kapitola je věnována analýze, jenž je potřebná pro získání potřebných informací a popsání požadavků softwarového produktu. Díky analýze se můžeme v následující kapitole věnovat návrhu.*

*Následující stránky jsou věnovány analýze teorie barev a vnímání barev lidmi. Následně je věnována pozornost známým barevným sadám a tvorbě barevných palet, extrakci informací z fotografie a využití AR k této extrakci. Prostor je věnován také existujícím aplikacím s podobnými funkcionalitami, díky kterým si upřesníme a definujeme funkční a nefunkční požadavky. Na závěr specifikujeme požadavky pomocí modelu případu užití.*

## 2.1 Teorie barev

Teorie barev by se dala popsat jako soubor postupů a pravidel zahrnující používání primárních pigmentových barev, vytváření barevné harmonie, míchání barev a jejich aplikaci. Abychom se však mohli věnovat samotným teoriím a určit postupy, se kterými budeme k této práci přistupovat, je třeba začít s úplnými základy a definovat si, co vlastně je barva a jak je vnímána lidským okem.

### 2.1.1 Barva

Vnímání barev je velmi subjektivní a již od dob Aristotela nám jejich zkoumání nepřineslo jednotný názor, který by přesně definoval barvu. Ten založil své poznatky na pozorování slunečního světla, které při odrazu či průchodu objektem snižuje svou intenzitu nebo je ztmaveno. Vnímal tak barvu jako mísení, míchání, superpozici či juxtapozici černé a bílé. [1]

Tento názor se však později setkal s kritikou a byl nahrazen novými poznatky od dalších teoretiků. Dodnes je však velmi dobrým základem pro další zkoumání vlastností barev.

### 2.1.1.1 Isaac Newton

S významným názorem přišel v 18. století významný fyzik, matematik, astronom a alchymista, Isaac Newton, ve své knize *Optika*. Kniha dokumentuje Newtonovy pokusy s lomem světla skrze hranol. Během svého pozorování objevil, že se jediný paprsek světla rozkládá do více barev v podlouhlém tvaru. V té době se jednalo o překvapující objev. Dle přijatých zákonů lomu by mělo totiž rozptýlené světlo nabírat kruhovitých obrazců [2]. Po dalších pozorováních vznесl Newton názor, že barva je viditelná část elektromagnetického spektra. Díky tomu můžeme rozehnávat různé barvy, ve svém spektru identifikoval také konkrétní odstíny. Nejprve červenou, žlutou, zelenou, modrou a fialovou, později také oranžovou a indigo [3]. Počet barev, tedy číslo 7, v tomto případě hraje významnou roli. Prvočíslo s mystickými významy udává také počet tónů na stupnici. A dle Newtona právě tóny a sluch souvisel s barevným viděním.

I přesto se ve své době nesetkal vždy s pozitivním ohlasem. Mnozí umělci stále věřili, že barva je něco mnohem komplexnějšího. Barvá má průhlednost, může být leská i matná, disponuje texturou i odstínem. Všechny tyto složky jsou důležitými aspekty, které nám dohromady vytváří svět tak, jak jej vidíme. [4]



■ **Obrázek 2.1** Sir Isaac Newton experimentující s hranolem. Rytina dle obrázku od J.A. Houston, ca. 1870. Se svolením The Granger Collection, New York [5]

### 2.1.1.2 Johann Wolfgang von Goethe

Jedním z těch, kteří barvu chápali jako mnohem abstraktnější pojem, byl německý spisovatel Johann Wolfgang von Goethe. Důležitým dílem se stala kniha *Teorie barev*, kde Goethe zpochybňuje Newtonovy názory. V jeho spisu popisuje barvu ne jako vědecký údaj, ale jako subjektivní zážitek, přírodní vjem

projevující se kontrastem, mísením, zvětšením či dělením. Součástí díla je takéž první systematická studie o fyziologických účincích barev.

Goethe rozdělil barvu do tří tříd podle toho, jak se projevují. První jsou psychologické barvy, které závisí na oku a reakci orgánů. Dalšími jsou fyzické barvy, ty jsou produkovány materiálními médii, avšak sami o sobě barvu mít nemusí. Jejich vnímání je určené okem, a to prostřednictvím vnějších vjemů či odrazů. Tyto barvy jsou pomíjivé a nelze je zadržet na dlouhou dobu. Třetí třídou jsou chemické barvy, které patří daným látkám a jsou trvalé po libovolně dlouhou dobu. [1]

Goetheovo definování barev je založeno spíše na jeho zkušenosti a ve své knize se zaměřuje na vnímání psychologické a emocionální. Ačkoliv z lidského pohledu jeho tvrzení nejsou nekorektní, postrádají fyzikální základ a nesoustředí se na vědecké vlastnosti.

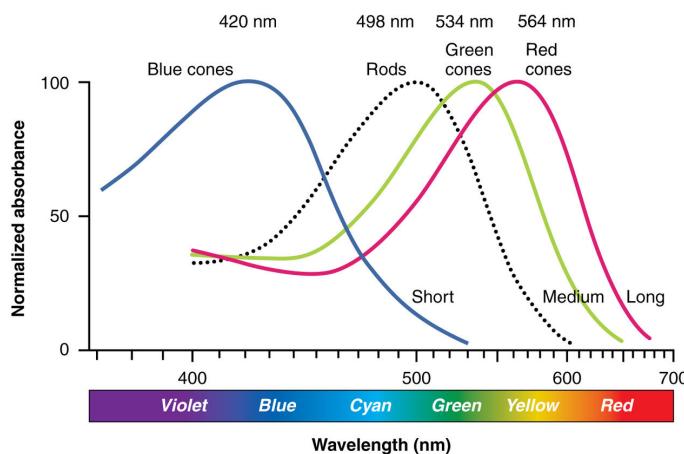
#### 2.1.1.3 Young-Helmholtzova teorie

Jen o pár desítek let později vznikla nová, dodnes v praxi využívaná teorie. Základ ji dodal anglický vědec Thomas Young v roce 1802, který přišel s *trichromatickou teorií barevného vidění*. Nespokojil se totiž s Newtnovým názorem o velkém množství častic v oku a vytvořil předpoklad, že existují tři fotosenzitivní receptory reagující na základní barvy, tedy červenou, modrou a zelenou. Drážděním receptorů pak vznikají další barevné kombinace a spojením všech tří vznikne dojem bílé barvy. Jejich absence pak vytváří dojem černé.

Myšlenku následně rozvedl německý fyziolog Hermann Ludwig Ferdinand von Helmholtz. Dle jeho poznatků nejsou receptory drážděny pouze jednou barvou, ale všemi třemi s různou intenzitou. Tak mohou vznikat všechny možné barevné odstíny a drážděním všech tří barev ve stejném poměru vytváří vjem bílé. Té lze ale dosáhnout také smícháním dvou komplementárních barev (např. modré s oranžovou).

Tuto teorii podpořili i další osobnosti, jedním z nich je Ragnar Granit. V roce 1964 Paul K. Brown a George Wald prokázali existenci tří fotopigmentů sítnice citlivé na různé vlnové délky, později Ragnar Granit, Haldan Keffer Hartline a George Wald dokázali existenci těchto fotoreceptorů spektrofotometrickým vyšetřením absorpce světla. Za tento objev získali v roce 1967 Nobelovu cenu. [6]

Tato teorie je využívána dodnes, dle některých zdrojů známá pod názvem *Young-Helmholtz-Maxwellova teorie*. Skotský fyzik, James Maxwell, se totiž v době Helmholtzova zkoumání věnoval stejnému tématu a rozšiřoval tuto teorii o další pozorování. Potvrzel existenci tří typů receptorů a popsal barvoslepou jako poruchu těchto receptorů. Zároveň definoval světlo jako elektromagnetické vlnění a popsal souvislosti vlnových délek s barvou světla. [7]



■ **Obrázek 2.2** Spektrální absorpční křivky pigmentů krátkých (S), středních (M) a dlouhých (L) vlnových délek v lidských čípkových a tyčinkových buňkách. [8]

#### 2.1.1.4 Vlastnosti barev

Pro pochopení, co je barva, je třeba znát i vlastnosti barev, které popisují jejich vzhled a charakterizují je. Definují se nejčastěji numericky vzhledem k faktu, že lidské oko jej objektivně vyhodnotit nedokáže. Základními atributy barev jsou odstín, sytost a jas [9].

**Odstín** Odstín je vyznačován názvem konkrétní barvy. Liší se v závislosti na vlnové délce a je rozhodující pro výslednou podobu barvy. Další atributy vzhledem k jejich podstatě totiž fungují pro diskriminaci barev spíše jako doplňkové.

**Sytost** Sytost určuje množství bílé v konkrétní barvě. Čím větší množství bílé barvy obsahuje, tím menší je sytost. Udávána je nejčastěji v procentech, kde 0 % značí černobílou a 100 % plnou sytost bez černého či bílého pigmentu.

**Jas** Jas určuje světelnou intenzitu barvy, tedy množství světla vycházející z dané barvy. Více světelných paprsků vytváří světlejší vjem barvy. Stupnice pro měření jasu je od 0 do 100, kde 0 značí absolutní černou. Lidské oko je údajně schopno rozlišit až 300 stupňů jasu barev v různých odstínech.

#### 2.1.1.5 Moderní definice

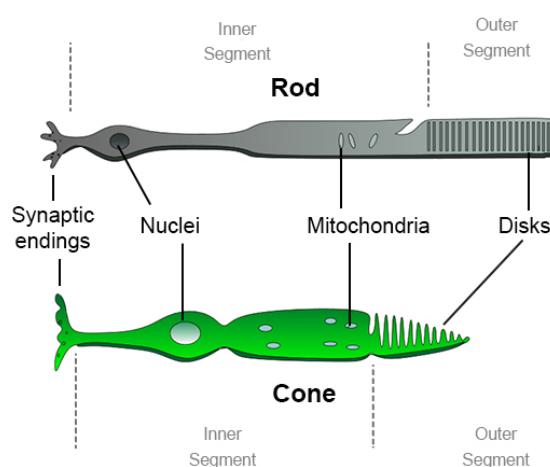
Jak tedy definovat barvu na základě veškerých zmíněných poznatků? Obecně lze říci, že barva je aspekt způsobený kvalitou světla odrázejícího se či pohlceného daným objektem. Abychom viděli barvu, je třeba mít světlo. Lidské oko zvládne vidět pouze barvy, které se odráží nebo reflekují. Samotné vnímání však zůstává stále velmi subjektivní [10].

## 2.1.2 Oko a barevné vidění

Oko je smyslový orgán reagující na světlo, který nám zajišťuje zrak. Díky očím můžeme vnímat naše okolí, vidět a rozpoznávat barvy v závislosti na jejich vlnové délce. Oko je složené z rohovky, duhovky, čočky, sklivce, zrakového nervu a stínice. Právě sítnice je zodpovědná za barevný vjem. Jedná se o průhlednou blanku, jenž je rozdělena linií ora serrata. Přední část neobsahuje žádné smyslové či nervové elementy, v zadní části se nachází světločivé buňky – tyčinky a čípky. Těch je v sítnici přibližně 130 milionů. [11]

**Tyčinky** Tyčinky jsou mnohem početnějším typem fotoreceptoru než čípky. Jsou taktéž mnohem citlivější, při optimálních podmírkách je drázdí i jednotlivé fotony. Tyčinky zajišťují černobilé vidění a umožňují nám adaptovat se na tmu. Převládají v periferním vidění, které je díky tomu citlivější na světlo. Jsou složeny ze zevního a vnitřního segmentu, které spojuje zúžená část. Pod tímto zúžením se nachází bazální tělíska s nahromaděnými mitochondriemi v jeho okolí, jejichž existence je důsledkem zvýšené produkce energie, jenž je důležitá pro proces vidění.

**Čípky** Čípky jsou strukturované podobně jako tyčinky. Zevní segment je však kratší, silnější v kónickém tvaru. Tvar se může lišit i dle jejich lokality na sítnici, v centrální jamce, místě nejsotřejšího vidění, mohou být i delší než tyčinky. Nejvíce jich pak lze nalézt v žluté skvrně. Čípky, stejně jako tyčinky, obsahují zrakový pigment. Narození od tyčinek osabující pouze jeden typ pigmentu, přesněji purpur, mají však rovnou tři typy tohoto pigmentu. Délí se dle vlnových délek na krátké S (short), středně dlouhé M (medium) a dlouhé L (long). Jsou citlivé na modré, zelené a červené světlo.



■ Obrázek 2.3 Struktura tyčinek a čípků v oku. [12]

Při vstupu světla do oka prochází postupně nejprve rohovkou, přední oční komorou s tekutinou, následně prostorem se sklivcem až do sítnice. Fotoreceptory pohlcují světlo spolu s velkým množstvím enzymů a molekul přeměňující světelnou energii na kinetickou energii. Tento chemický proces vytváří nervový vzruch a následně se šíří až do mozku. Samotný mechanismus vnímání barev dodnes není úplně jasný, v dnešní době je však nejrozšířenější dříve zmíněná Young-Helmholtzova teorie. Spolu s Heringovou teorií, dle které využíváme dva protikladné páry pro vnímání barev – červenou a zelenou s žlutou a modrou, tvoří Dvoustupňovou teorii kombinující oba názory, která je dnes akceptována vědci. [13]

### 2.1.2.1 Vliv kontextu na vnímání barev

Již víme, že oko obsahuje velké množství fotoreceptorů umožňující nám vidět v barvách. Ne vždy se na něj však lze spolehnout a v rámci vidění jsme limitováni různými aspekty, dle nichž se mohou některé věci jevit rozdílně. Kontext zde hraje důležitou roli.

Při segregaci objektů je důležité nejen to, co vidíme, ale také vyhledávání v paměti mezi tím, co již známe. Barva slouží k identifikaci objektů a stavu objektu. Banán rozeznáme díky jeho žluté barvě, stejně tak zvládneme díky barvě rozeznat citron od limetky. Barva, kterou si pamatujeme díky zkušenosti s konkrétním objektem, se nazývá *paměťová barva*. Barevně neutrální jsou pak objekty bez této paměťové barvy, příkladem mohou být například auta, u kterých si jedinou barvu nelze spojit.

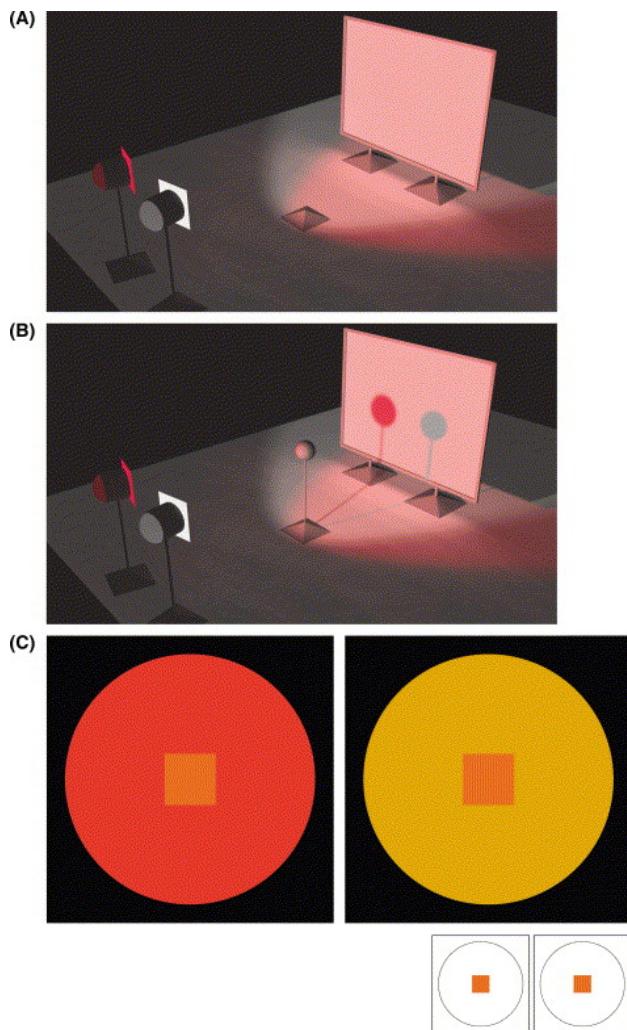
Kromě rozpoznávání objektů ovlivňuje paměťová barva i naše vnímání. Tento vliv se nazývá *Efekt paměťové barvy*. Dle tohoto efektu se nám mohou objekty jevit zbarvené do určitého odstínu i přesto, že jsou vyobrazeny pomocí stupnice šedi. Černobílý banán se nám tak stále může jevit jemně nažloutlý. Naopak pro dosažení černobílého vjemu je třeba úprava barvy směrem k opačnému odstínu. Například namodralý obrázek banánu se tak bude jevit spíše černobílým.



■ **Obrázek 2.4** Paměťová barva jahod rozpoznatelná i tehdy, když je jejich fotografie zmanipulována tak, že červená barva je šedá. [14]

Vliv paměťové barvy se ukázal nejsilnějším u modrých a žlutých odstínů. Jedním z důvodů se jeví proměnlivost denního světla. Právě žlutá s modrou odpovídají přirozenému dennímu světlu, což může způsobit nejistotu od pozorovatele [15].

Důležitým efektem kontextu na vnímání barev je také kontrast. Fenomén barevných stínů popisuje nasvícení bílé obrazovky dvěma světly. První světlo dlouhé vlnové délky vypadá červeně, druhé světlo bíle. Dle očekávání se obrazovka bude jevit růžově. Při posazení objektu do cesty mezi světla a obrazovku pak vzniknou dva stíny. První stín se zablokovaným zdrojem bílého světla se jeví sytě červeně, druhý stín se zablokovaným červeným světlem však působí nazelenale.



**Obrázek 2.5** Příklady barevných stínů a vlivu pozadí na vnímání barvy objektu. [16]

O tento jev se zajímal francouzský chemik Chevreul, který následně vydal spis popisující pravidla kontextu na vnímání barvy. Dle jeho poznatků je vnímaná barva objektu vychýlena směrem k opačné barvě pozadí na Newtonově barevném kruhu. Tento efekt byl hojně využíván v umění i průmyslu. [16]

I věk hraje roli v kontextu vnímání barev. S vyšším věkem je člověk náchylnější k poruchám vidění a hůře rozeznává barvy. Nejen vyšší věk je však důvodem zhoršeného barevného vidění. Studie ukazují, že u novorozenců dochází k postupnému vývinu čípkových kanálů, tedy zpočátku rozeznávají pouze velmi syté a dostatečně velké objekty s určitým odstínem. I v pozdějším věku se však vnímání stále vyvíjí a saturačního prahu na úrovni dospělého je dosaženo pravděpodobně až v pozdní adolescenci. [17]

### 2.1.3 Vliv barev v běžném světě

Vnímání barev není jen prostředkem pro rozeznávání objektů a vyhodnocování stavů daných objektů. Díky barevným vjemům můžeme prožívat emoce, ovlivňovat naše nálady a aktuální stav či upevňovat sociální nastavení. Díky vhodné volbě barevných prvků lze manipulovat člověkem ke koupi produktu či k volbě politické strany [18], nebo naopak vytvářet příjemná prostředí v interiérech, navrhovat vizuálně lichotící kusy oblečení a mnoho dalšího.

#### Test zelené barvy

Existují například studie, které sledovaly účinek zelené barvy jako primitivní rys přírodních prostředí na výsledky cvičení a změnu nálad. Z pozorování vyplynulo, že oproti červenému či achromatickému filtru zeleň snižuje narušení nálady spolu s hodnocením vnímání obtížnosti cvičení. Naopak pocit hněvu byl silnější u červeného filtru. [19]

#### Vliv barev na chování spotřebitele

Další studií zkoumající vliv barev v běžném světě se věnuje ovlivňování barev na chování spotřebitele. Zde je kladen důraz na rozlišování barev primárně dle zkušenosti. Tedy na rozdíly mezi teplými a studenými tóny, zářivými a tmavými odstíny a jejich asociaci s náladou. Studené tóny v lidech probouzí větší klid, tmavé odstíny jsou přirovnávány k negativnímu kontextu, smutku a nudě, naopak zářivé odstíny jsou vnímány jako hravé, radostné a plné naděje.

Záleží však také na původu spotřebitele, věku i vzdělání. Američtí konzumenti jsou přitahování k modré a červené, naopak občané Íránu a Kuwaitu preferují modrou se zelenou. Občané ve vyšším věku jsou méně otevřeni experimentování s barevami a u vzdělanějších osob probíhá výběr barev mnohem složitějším způsobem. Zakomponování teorie barev v konzumní společnosti je tedy náročný úkol, kterým se zabývají mnozí experti. I přes komplikovanost úkolu jsou totiž výsledky velmi efektivní. Design balení a výběr barev totiž

mají nejen signifikantní vliv na samotnou koupi produktu, ale také vytvoření asociace s produktem, díky které si lze zákazníka udržet. [20]

## 2.1.4 Poruchy barevného vidění

Ačkoliv je pro většinu z nás zcela přirozené vidět barvy, i přes jejich subjektivní vnímání, v plné kráse, barevné vidění není samozřejmostí pro každého. Barvoslepství je porucha, při které dochází k omezení rozeznávání barevných tónů, a to buď částečně, či dokonce úplně. Následující kapitola je převzata a upravena na základě bakalářské práce Kláry Holíšové (2007), která se komplexně věnuje problematice barevného vidění. [21]

### 2.1.4.1 Vrozené poruchy barevného vidění

Při úplné barvosleposti vnímá postižený svět jako černobílý film. V některých případech je schopnost rozeznávat barvy zcela omezena, v některých případech pouze oslabena. Budeme se řídit příponami, kde *-anomálie* značí oslabení a *-anopie* úplnou barvoslepství. Předpona *prot-* pak označuje vadu pro vnímání červené barvy, *deuter-* pro vnímání zelené a *trit-* pro vnímání modré. Osoby se správným vnímáním všech tří barev jsou označování *trichromati*, ti, kteří disponují poruchou barevného vnímání, se nazývají *dichromati* a *monochromati*.

#### Anomálie

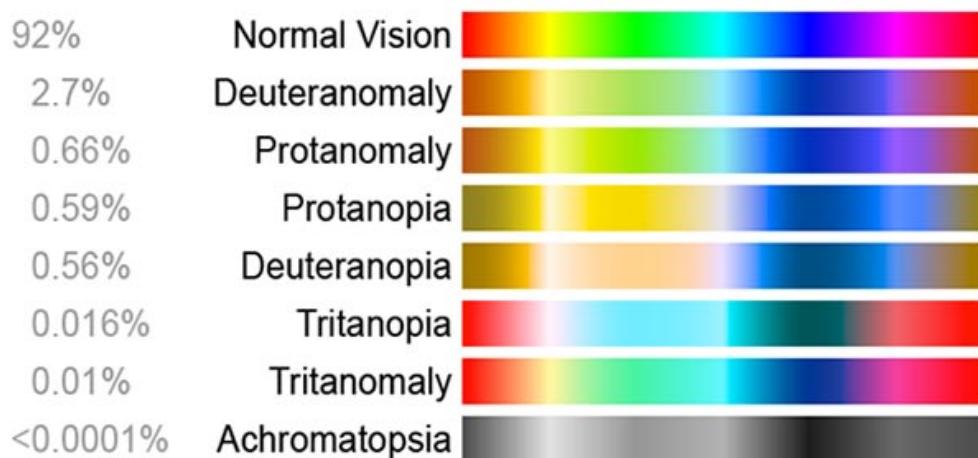
Rozlišujeme tři typy anomalií. Protanomálie způsobuje nedostatečné množství červených čípků, postižený špatně rozlišuje mezi červenou a zelenou. Touto anomalií trpí až 1 % mužů. Nejrozšířenějším typem anomálie je deuteranomálie, která se týká také zhoršeného rozeznávání červené a zelené z důsledku oslabení zelených čípků. Postihuje až 5 % mužů. Tritanomálie je vzácnější porucha, při které dochází k zhoršenému rozeznávání žluté a modré barvy. Není vázána na pohlaví, tedy postihuje ve stejném míře muže i ženy.

#### Dichromati

Jak již bylo zmíněno, dichromati jsou osoby se schopností rozlišovat pouze odstíny vzniklé ze dvou základních barev. Jeden z čípků chybí. Protanopie značí poruchu vidění červené barvy, postižení zaměňují červenou za zelenou, špatně také rozeznávají rozdíly mezi modrou a zelenou. Deuteranopie je poruchou vidění zelené barvy. V tomto případě dochází k zaměňování zelené až modré za fialovou. Tritanopie je, stejně jako tritanomálie, nejvzácnější z těchto tří poruch. Jedná se o poruchu vidění modré barvy.

#### Monochromati

Monochromati disponují jediným typem čípků a nedokáží rozlišit barvy. Vidí v odstínech šedé, zvládnou rozlišit pouze intenzitu světla.



**Obrázek 2.6** Vnímání spektra viditelného světla u zdravých osob a osob s poruchami barvocitu, včetně informace o procentuálním výskytu poruch barvocitu v populaci. [22]

#### 2.1.4.2 Získané poruchy barevného vidění

Kromě vrozených poruch existují také poruchy získané. Ty jsou nezávislé na pohlaví, typicky bývají asymetrické mezi očima a méně stálé než choroby vrozené.

##### Katarakta

Katarakta, neboli šedý zákal, je choroba, při které má oční čočka sklon ke žloutnutí. Následkem žloutnutí absorbuje krátkovlnné světlo a zhoršuje se rozeznávání v modré oblasti spekra. Šedý zákal se rozvíjí v důsledku stárnutí, ale nejedná se o jediný faktor přispívající k jeho vzniku. Přispět může také například vystavení oka nepřiměřenému infračervenému záření, rozvinout se může také důsledkem traumatu či systémového onemocnění, jako je diabetes mellitus či galaktosemie.

##### Achromatopsie

Achromatopsie je vzácná porucha způsobená poškozením mozkových center, které zodpovídají za zpracování barevného vjemu v čtvrté vizuální oblasti temporálního laloku. Tato část je zodpovědná za zpracování barev a tvarů a je důležitá pro rozeznávání objektů. Vzniká důsledkem kyslíkových poruch, například při otravě oxidem uhelnatým či při mrtvici.

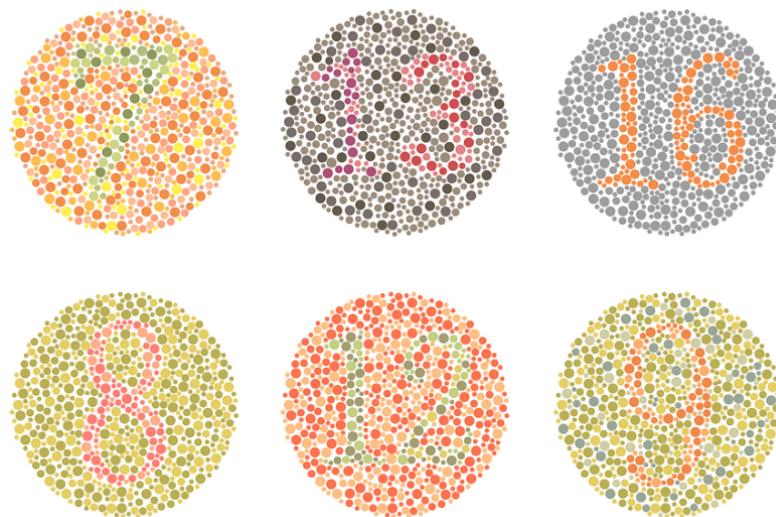
##### Chromatopsie

Důsledkem Chromatopsie je nepřirozené vnímání barev. Rozlišují se dle viděných barev na cyanopsii pro modrou, chloropsii pro zelenou, erythropsii pro

červenou a xantopsii pro žlutou barvu. Vzniká při dlouhodobém užívání drog, kataraktou, nebo například při oslnění světlem.

#### 2.1.4.3 Vyšetření barvocitu

Vyšetření barvocitu probíhá pomocí testů založených na principu správného rozližování odstínů. Pro testování je nutné mít správně osvětlené prostory a zdroj světla by měl být pod úhlem 45 stupňů. Vyšetření probíhá pro každé oko zvlášt. Rozšířené jsou pseudoizochromatické tabulky založené na principu splývání záměnných barev. Do tabulek jsou vepsány číslice, písmena či geometrické útvary z kruhových bodů, které se liší od ostatních bodů v tomto obrazci barevným tónem. Zdravý člověk by za správných podmínek ve vzdálosti přibližně 1 metr od tabulky měl rozeznat znak do 15 sekund. Normální stav povoluje jednu až dvě chyby. Při vážnějších chorobách jsou nutná další vyšetření a tento test pak slouží pouze orientačně. Není vhodný pro pacienty s poruchou modré složky. [21]



■ **Obrázek 2.7** Příklad tabulky pro test barvocitu. [23]

Dalším využívaným testem je *Farnsworth-Munsell 100-hue test* sloužící pro rozpoznání abnormalit i achromatopsie. V tomto testu pacient musí seřadit 85 knoflíků v různých barevných odstínech dle barev spektra. Obdobným testem využívajícím méně syté barvy je Lantonyho test. Záměny při přiřazování následně určí typ poruchy. [21]

#### 2.1.5 Teorie barev

Historie teorie barev jako souhrn pravidel má úzkou návaznost na kapitolu o definicích barev. Jména zmíněných osobností zabývajících se tím, jak popsat

barvu, totiž hrají roli i při hlubším zkoumání hierarchie barev, jejich míchání a použití. Aristoteles, jak již víme, předpokládal, že barva pochází pouze z bílé a černé. Jeho teorie popisuje vznik barev mísením těchto dvou primárních barev s přidaným vlivem přírodních živlů. Země je asociovaná s tmavšími barvami, jako je hnědá a černá, vzduch se světlými odstíny, oheň s načervenalými a zlatými odstíny, voda je přirovávána k modré a k transparentním barvám.

V historii najdeme mnoho dalších osobností, které se věnovaly teorii barev. Každý totiž vnímal důležitost různých aspektů jinak v závislosti na odvětví, ve kterém se pohyboval. Vznikaly proto teorie v umění, designu, ale například i ve vědě. Již zmínění Newton i Goethe vytvořili svou teorii barev, následovali je například také Tobias Mayer, Johannes Itten či Josef Albers [24]. Díky těmto jménům se postupně formovala dnes již běžně aplikovaná barevná teorie, která je popsána na následujících řádcích.

#### 2.1.5.1 Kategorizace barev

Pro kategorizaci barev je nutné definovat barevný kruh, který leží v úplném základě teorie. Představuje logicky uspořádanou posloupnost čistých odstínů. Varianty tohoto konceptu jsou stále předmětem četných diskusí, avšak v tomto případě se není nutné soustředit na jediný typ takového kruhu.

Nejprve rozlišujeme primární barvy. Mezi primární barvy se řadí červená, žlutá a modrá. Tyto tři pigmenty nelze smíchat ani vytvářet kombinací dalších barev. Veškeré další odstíny jsou tvořeny z těchto pigmentů.

Sekundární barvy jsou tvořeny z primárních barev, řadí se mezi ně zelená, oranžová a fialová.

Terciální barvy jsou další v pořadí a řadí se mezi ně odstíny smíchané ze sekundárních a primárních barev. [25]



■ **Obrázek 2.8** Paleta primárních, sekundárních a terciálních barev. [25]

#### 2.1.5.2 Harmonie barev

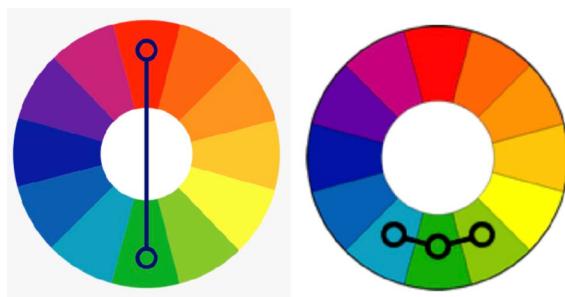
Harmonie je definována jako uspořádání částí budící v cílovém jedinci příjemný dojem. Z pohledu vizuální estetiky jde o nějaké uspořádání lahodící oku pozorovatele, jenž jej zapojuje a probouzí v něm vnitřní pocit řádu a rovnováhy. Existují základní teorie pro barevnou harmonii dosahující dostatečné rovnováhy pro vhodnou stimulaci, které fungují napříč různými kontexty. [25]

### Analogické barvy

Schémata s analogickými barvami jsou jakékoliv barvy, které jsou posazeny na barevném kruhu u sebe. V tomto schématu běžně převládá jedna z barev, do hromady však navozují pocit harmonie a klidu. Příkladem může být například kombinace zelené se žlutou, či žlutá s oranžovou.

### Komplementární barvy

V případě schémat tvořených z komplementárních barev vybíráme barvy lokalizované na barevném kruhu přímo proti sobě. Protikladné barvy vytváří maximální kontrast, jehož důsledkem je paradoxně také maximální stabilita. Jedná se například o fialovou a žlutou, či modrou s oranžovou.



■ **Obrázek 2.9** Ukázka komplementární a analogických barev. [26]

#### 2.1.5.3 Barevný kontext

V kapitole o vlivu kontextu barev jsme představili důležitost tohoto aspektu v rámci barevného vnímání. V rámci teorie barev je třeba přemýšlet nad tím, jak působí kontrast pozadí s objektem a kterého z daných výsledků chceme dosáhnout. Červený čtverec může působit zcela rozdílně na černém pozadí oproti bílému. Objekty se mohou jevit větší, nebo naopak menší, mohou být výraznější nebo naopak působit upozaděně. V rámci kontextu je proto důležité nejprve stanovit cíl, dle kterého jsou další souvislosti postupně budovány. [25]

## 2.2 Barevné systémy a tvorba barevných palet

V kapitole Barevné systémy a tvorba barevných palet si představíme základní používané palety, se kterými se lze setkat nejen v uměleckém odvětví, ale také v každodenním životě. Různé barevné prostory se hodí k rozdílným účelům. Jeden je vhodný pro zobrazení na monitoru počítače, další je lepší z hlediska výsledků tisku.

### 2.2.1 Míchání barev

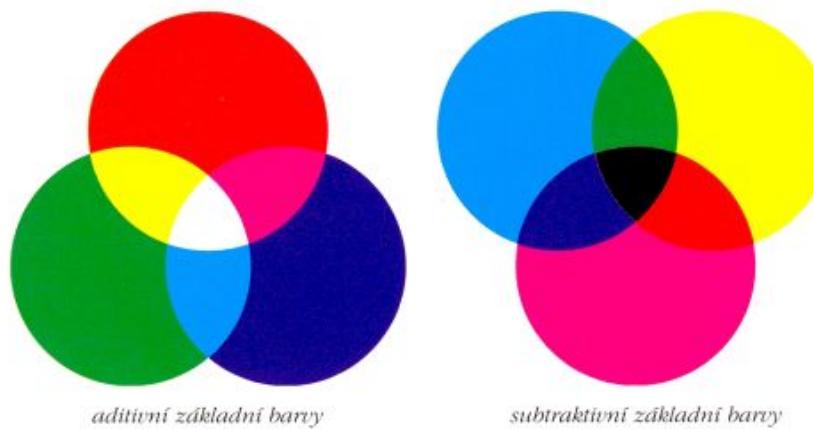
Existují dva základní přístupy pro získávání barev, a sice aditivní (součtové) a substraktivní (odčítací). Jejich primární rozdíl je zdroj světla, kdy v principu aditivním se jedná o míchání světla přímo ze zdroje, substraktivní princip naopak pracuje s odrazem světla od povrchu. [27]

#### Aditivní míchání

Při aditivním míchání se sčítají různě barevná světla s různou intenzitou. Primárními barvami jsou červená, modrá a zelená. Tímto principem lze získat veškeré barvy viditelné části spektra, při smíchání všech světel se stejnou intenzitou dosáhnete bílé barvy. Při jejich úplné absenci je výsledkem černá. [27]

#### Substraktivní míchání

Substraktivní princip vychází z práce malířů s barvami. Každý pigment pochlívá i odráží určitou část světla a při odrazu dochází k odečítání jednotlivých složek. Čím více tedy barevných pigmentů přidáme, tím tmavší výsledek získáme. Základem je bílá barva, ke které se nanášejí další. Vzájemným překrýváním primárních barev získáváme další nové barvy. Zde jsou primárními barvami azurová, purpurová a žlutá. [27]



■ **Obrázek 2.10** Ukázka aditivních a substraktivních barev. [28]

### 2.2.2 Základní barevné systémy

V této sekci si představíme základní systémy, kterými lze vyjádřit určité množství barev. Ačkoliv jich existuje více, představíme si pouze ty relevantní pro naši práci, a to RGB, CMYK a HSV. Poslední část se bude věnovat také systému Pantone.

### 2.2.2.1 RGB

RGB systém je jedním z nejznámějších systémů, které vychází z vnímání barev lidským okem. Jeho název vychází z názvů základních složek, červená (R), zelená (G) a modrá (B). Řadí se mezi aditivní systémy a je využit primárně pro obrazovky počítačů, smartphonů a dalších monitorů. Každá složka disponuje 256 stupni intenzity (0-255), tedy pro získání černé je R, G i B rovno 0, naopak pro bílou je nutné, aby R, G i B bylo rovno 255. Pro získání například šedé můžeme R, G i B nastavit na 100. [27]

Problém nastává při tisku. Barvy míchají rozdílným způsobem, a tedy při získání informace od počítače je třeba pracovat s odlišným systémem [29]. Ze systému RGB vychází další prostory vytvořené společnostmi pro lepší zpracování tiskárny. Adobe RGB vytváří většinu barev, které zvládne CMYK, avšak při použití RGB. Je tedy schopen získat přibližně 50 % všech viditelných barev. Dále existuje například ProPhoto RGB od společnosti Kodak s větším gamutem, či sRGB od firem Microsoft a Hewlett-Packard. sRGB je standardní paletou pro HTML a je hojně využíván v oblasti fotografií [27].

### 2.2.2.2 CMYK

CMYK využívá oproti RGB substraktivní metodu míchání a název vychází z barev nejčastěji využívaných tiskárny: azurová (Cyan), purpurová (Magenta), žlutá (Yellow) a černá (Značena jako "Key" doplňující kontrast). Dokáže však vyproduktovat pouze omezené množství barev, proto se také nabízí barvy do tiskáren s vlastními odstíny. Oproti monitoru není výsledek tak jasný a barevný. [29]

### 2.2.2.3 HSV

Model HSV je specifikován třemi hodnotami. Barevný tón (Hue) definovaný ve stupních, sytost (Saturation), která určuje odstín, a jas (Value) specifikující hodnotu bílého světla. Systém je využíván ke konkrétnějšímu popisu barev a velikou výhodou je jeho nezávislost na zařízení. Naopak nevýhodou je neplynulý barevný přechod. [27]

### 2.2.2.4 Pantone

Pro jednotnost barev firem a společností, ať už je tisknuté kdekoliv, slouží Pantone. Jedná se o standardizovaný vzorník barev, jenž vznikl v roce 1963. Obsahuje 1867 barev včetně metalických a reflexních odstínů. Vychází ze 16 barev, z nichž jsou přesně odměřeny poměry barev pro objektivní výsledek. [29]



■ Obrázek 2.11 Pantone vzorník. [29]

## 2.3 Rozšířená realita a její využití

Rozšířená realita ("AR") je definována jako integrace digitálních informací do prostředí uživatele v reálném čase. Technologie rozšířené reality překrývá obsah reálného světa a obohacuje jeho vnímání bez nahrazování reality samotné. [30]

### 2.3.1 Virtuální realita vs rozšířená realita

Virtuální realita i rozšířená realita jsou v digitálním světě již poměrně známé a jejich využití je na vzestupu. Oba procesy však fungují velmi odlišně, proto je třeba ujasnit si rozdíly mezi těmito výrazy a konkrétněji definovat jejich využití.

Systémy rozšířené reality, jak je již definování výše, začleňují digitální informace do reálného světa, ale samotné vnímání reality nijak nemění. Naopak virtuální realita ("VR") je popisována jako trojrozměrný svět generovaný počítačem. Simuluje tedy prostředí vnímané uživazelem a umožňuje s ním interagovat, a tedy simulovat jeden či více smyslů. Fúze obou principů se nazývá "mixed reality". [31]

### 2.3.2 Princip fungování

Zařízení využívající rozšířenou realitu nejprve přijímá videosignál ze zorného pole uživatele (např. kamery) a snímá okolní prostředí a fyzické objekty v tomto poli. Může to taktéž zahrnovat sběr dat z GPS, laserů či gyroskopů.

Software rozšířené reality zpracovává a skenuje toto přijímané prostředí. Snaží se identifikovat objekty a vlastnosti prostředí, které by bylo možné rozšířit. Tato identifikace může potenciálně využívat umělou inteligenci k rozpoznání objektů či různé senzory.

Ze softwaru pokračují informace zpět do zařízení, kde je generovaný obsah překrýván do zorného pole uživatele, a to ve správné perspektivě a orientaci. Pro interakci je možné využívat příkazy prostřednictvím fyzických gest, hlasu či dotykem obrazovky.

### 2.3.3 Možnosti využití

Dnes existuje mnoho možností, jak využívat rozšířenou realitu. V případě vzdělávání je možné zpřístupnit studentům lepší porozumění látky pomocí různých 3D modelů a simulací. To může například zahrnovat prozkoumávání historických budov, či dokonce celých, dnes již neexistujících měst.

Její využití však lze objevit i v herním průmyslu, kde se herní charakterysty a postavy objevují v reálném světě. Příkladem může být známá aplikace Pokemon GO, která umožňuje uživatelům dosáhnout ještě větší interakce se světem, čímž pomáhá ještě intenzivnějšímu prožitku ze hry.

Ve zdravotnictví existují AR systémy pro trénink, plánování operací pro preciznější výsledky a vzdělávání pacientů. Rozšířená realita tedy rozhodně neslouží pouze pro zábavní průmysl, může sloužit také jako podpora v oblastech, kde by vytvoření fyzického objektu znamenalo nereálné výdaje či čas na realizaci.

### 2.3.4 Druhy rozšířené reality

Rozlišujeme dva základní typy rozšířené reality. Rozlišují se dle druhů spouštěcích na "marker-based" a "marker-less".

#### Marker-based AR

"Marker-based AR" je realita založená na značkách, kterými jsou v tomto případě fyzické objekty vytvářející nějaký fyzický spouštěč. Může jít o QR kód, obrázek či jiná, předem definovaná značka. Při její detekci se spustí požadovaná akce rozšířené reality. Tento typ je dostupný bez omezení a je velmi flexibilní. Zároveň je méně nákladná.

#### Marker-less AR

"Marker-less AR" nevyžaduje narozdíl od předchozího typu žádný konkrétní spouštěč. Spoléhá na určené senzory, jakými může být GPS, akcelerometr či kamera, mapuje prostředí v reálném čase a snaží se mu porozumět. Pomocí různých algoritmů a strojového vidění AR sama určuje umístění digitálního obsahu, zajišťující tím dynamičtější a spontánnější zážitek. [30]

## 2.4 Rešerše existujících řešení

Analýza a rešerše již existujících aplikací detekující barvu z okolního prostředí a vytvářející odpovídající barevné palety je jedním z klíčových bodů pro popsaní silných a slabých stránek, které lze využít pro definování požadavků pro výslednou aplikaci. Výsledek z této rešerše bude využit k inspiraci z kladných aspektů využitých v konkurenčních aplikacích a k eliminaci nalezených nedostatků. V následujících podsekčích budou představeny čtyři vybrané aplikace s podobnými funkcionalitami, jejichž popis včetně diskutování slabých a silných stránek. Vyhledávání aplikací probíhalo na oficiální distribuční platformě AppStore s pomocí klíčových slov zaměřených na detekci barev s využitím i bez využití rozšířené reality a práci s paletami. Do výběru byly zařazeny aplikace, které splňovaly alespoň jednu z klíčových funkcionalit pro navrhovanou aplikaci. Veškeré níže zmíněné aplikace jsou dostupné v bezplatné verzi s možností premium placené verze.

### 2.4.1 Color Blind Pal

Aplikace Color Blind Pal je rozšířenou aplikací umožňující osobám trpícím barvoslabostí rozpoznávat okolní barvy, zároveň však umožňuje osobám s normálním viděním zobrazit svět tak, jak jej vnímají pacienti s různými diagnostikami. Pro úpravu světa jsou využívány různé filtry, které se snaží přiblížit co nejvíce uvedené vadě v barevném vidění. Jedná se spíše o velmi přibližnou ilustraci, vzhledem k tomu, že každá vada je u každého velmi specifická. Samotné ovládání je velmi jednoduché a intuitivní, aplikace pracuje pouze s kamerou a barevným spektrem ve spodní části. Nevýhodou může být neustále přeskakující detektor barev, jelikož se kamera neustále snaží zaostřovat a adaptovat. Při detekci je zobrazen také název barvy, v případě jemných odstínů je však tato detekce omezená, což může být v některých případech klíčové. Neexistuje taktéž žádná možnost uložení barev.

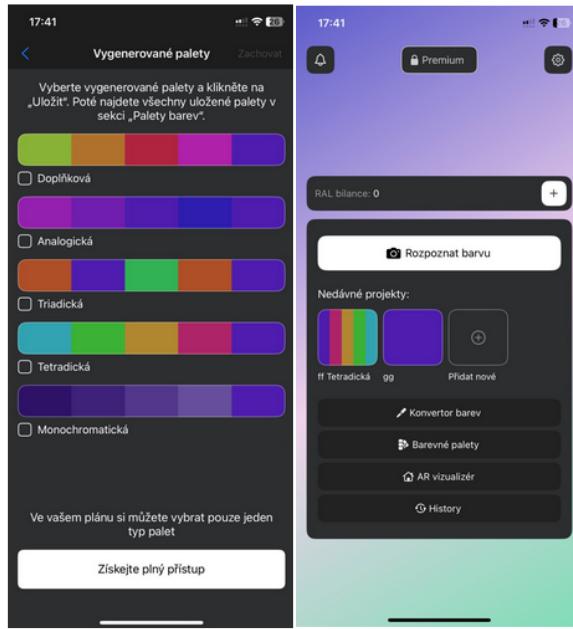
### 2.4.2 Color Identifier: Color Picker

Aplikace Color Identifier: Color Picker se věnuje rozpoznávání barev z kamery a tvorbě vlastních barevných palet. Umožňuje nejen ukládat detekované barevy, ale také z barevného spektra vybírat další odstíny, které si lze uložit do knihovny. Aplikace dále nabízí také konvertor barev dle HEX, RGB či CMYK kódu nebo pomocí výběru ze spektra. V nové verzi je uživatelům nabízen také režim pro barvoslepé nabízející zjednodušené barevné palety. Aplikace je velmi uživatelsky přívětivá, disponuje podporou více jazyků a přesně určuje detekované barvy. Ty jsou zobrazeny v boxu spolu s dalšími kódy. Po výběru barvy v knihovně lze vygenerovat 5 různých barevných palet v počtu až 5 barev, které si lze také uložit. Ačkoliv je aplikace velmi přehledná, obsahuje invazivní množství reklam a jen velmi omezené možnosti v bezplatné verzi, některé z



■ Obrázek 2.12 Ukázky aplikace Color Blind Pal. [32]

nich jsou pro běžné využití nadbytečné (Např. Generování pouze 1 barvy v paletě). Zároveň nelze v režimu výběru ve fotoaparátu zvolenou barvu při pohybu kamery zachovat.



■ Obrázek 2.13 Ukázky aplikace Color Identifier. [33]

### 2.4.3 **Paleto**

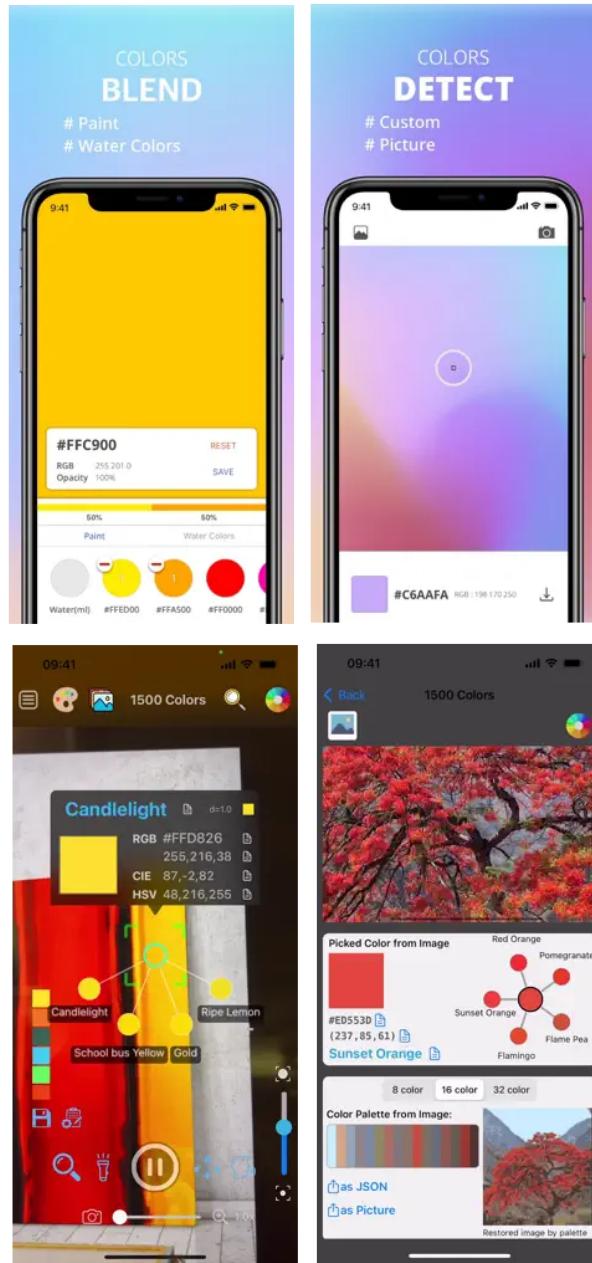
Paleto se orientuje primárně na tvorbu palet a míchání barev z konkrétních odstínů včetně volby poměru. Poskytuje možnost generování palety z fotografie či obrázku, hledání barvy v obášlé knihovně i ukládání vytvořených odstínů. V této aplikaci nelze detekovat konkrétní barvu, generování z fotoaparátu poskytuje pouze omezený počet nalezených barev. Aplikace je jinak minimalistická a neobsahuje nadbytečné funkcionality, které by komplikovaly její použití.

### 2.4.4 **Color Name Recognizer**

Mobilní aplikace Color Name Recognizer se také věnuje detekci barev z kamery. Nabízí box s názvem, ukázkou zvolené barvy a kódy RGB, CIE a HSV. Nástroj obsahuje voličem velikosti detekované plochy, přiblížením, možností zobrazení 1 až 3 podobných barev z palety a lupou pro konkrétnější výběr oblasti. Výhodou je také možnost pozastavení kamery pro získání detekované barvy. Detektor je ovšem neustále zapnutý, což má za následek stejnou problematiku, se kterou se potýká aplikace Color Blind Pal. Box neustále mění detekované informace a mění se také generované barvy na paletě. Dochází tak k přeskokům, které pro uživatele nemusí působit přívětivě.

### 2.4.5 **Shrnutí**

Uvedené aplikace se potýkají s podobnými nedostatkami i výhodami, zároveň ukazují, jak důležitá je volba vizuálních prvků. Uvedená rešerše poslouží jako podklad k analýze požadavků, kterým se věnuje následující sekce.



■ Obrázek 2.14 Ukázky aplikace Paleo[34] a Color Name Recognizer[35].

## 2.5 Požadavky

Po zhodnocení již existujících řešení, analýze a pro splnění další dílčí části zadání této práce je třeba zadefinovat si požadavky pro aplikaci, jenž je hlavním cílem této práce. Jejich specifikování je podstatné pro další fáze vývoje, zjištění důležitých prvků a funkcionalit a stanovení priorit, kterými se budeme ve vývoji řídit.

Každý požadavek vyžaduje uvedení informací umožňující nám je specifikovat pro lepší pochopení. V této práci využíváme následující údaje dle [36].

- **Název** uvádějící stručný popis požadavku.
- **Identifikátor** umožňující snazší odkazování na požadavek.
- **Popis** upřesňující detaily požadavku. Jedná se o nejdůležitější část.
- **Priorita** identifikující důležitost požadavku při vývoji. Specifikuje nám, bez kterých požadavků by aplikace nemohla existovat a které lze zrealizovat, ale nejsou pro tento běh významné. Pro rozdělení priorit požadavků využijeme MoSCoW metodu rozdělující prioritu do čtyř kategorií [37]:
  1. **Must have** definuje požadavky, které je nutné zahrnout ve finálním produkту.
  2. **Should have** popisuje význačné požadavky, které by měl produkt obsahovat, ale aplikace bez nich může fungovat.
  3. **Could have** požadavky jsou žádoucí, ale pouze v případě, že nevyžadují příliš velké úsilí či náklady.
  4. **Will not have** jsou požadavky, které je v zájmu zainteresovaných stran implementovat, ale ne v aktuální časové verzi.

Požadavky se dělí na **funkční** a **nefunkční** požadavky. **Funkční požadavky** jsou takové požadavky, které definují chování systému. **Nefunkční požadavky** určují omezení informačního systému a mají dopad na zvolení architektury a dodržování standardů. Níže jsou vypsány veškeré požadavky pro tuto práci i s popisem a určením priorit.

### 2.5.1 Funkční požadavky

**FP1 – Identifikace barev** Aplikace rozpozná barvy podle kamery. (*Must have*)

**FP2 – Tvorba barevných schémat** Na základě vybrané barvy se automaticky generují korespondující barevné palety. (*Must have*)

**FP3 – Ukládání a vyhledávání barev** V aplikaci je možné ukládat identifikované barvy a vyhledávat již uložené v knihovně barev. (*Must have*)

**FP4 – Odstranění barev z knihovny** Možnost mazat jednotlivé či více uložených barev. (*Should have*)

**FP5 – Návod** Asistent pomáhající k lepší orientaci v aplikaci. (*Should have*)

**FP6 – Vizualizace identifikované barvy** Extrahovaná barva je zobrazena na bodě detekce v kameře. (*Must have*)

**FP7 – Interakce v rozšířené realitě** Vytvoření virtuálního bodu za pomocí rozšířené reality. (*Must have*)

**FP8 – Rozšířená interakce v AR** Vytvoření více virtuálních bodů v AR pro porovnání barev. (*Could have*)

**FP9 – Extrakce z fotografie** Extrakce barvy z fotografie. (*Could have*)

**FP10 – Uložení vytvořeného snímku** Snímek z kamery lze uložit do zařízení. (*Could have*)

**FP11 – Míchání uložených barev** Aplikace vytváří další barvy z již uložených. (*Will not have*)

**FP12 – Tvorba dalších barevných schémat** Aplikace vytváří další typy barevných palet. (*Will not have*)

### 2.5.2 Nefunkční požadavky

**NP1 – Multiplatformní aplikace** Aplikace je vyvíjena pro iOS i Android platformy. (*Must have*)

**NP2 – Použití rozšířené reality** Využívají se komponenty z AR - interakce se světem, umístění objektů. (*Must have*)

**NP3 – Přístupnost** Aplikace má jednoduché UI a je přístupná všeobecnému publiku včetně osob s poruchami barevného vidění. (*Must have*)

**NP4 – Rozšiřitelnost** Aplikaci lze dále jednoduše rozšiřovat v rámci tvorby dalších palet či větší interakce s extrahovanými barvami. (*Should have*)

**NP5 – Výkon** Aplikace lokálně funguje plynule bez sekání a prodlev, detekce je rychlá bez významného zaznamenání při používání. (*Must have*)

**NP6 – Technická dostupnost** Aplikace je dostupná pro modely Apple iPhone 8 a vyšší a Android zařízení se systémem Android 7.0 a vyšší (s ARCore podporou). (*Should have*)

## 2.6 Model případů užití

Model případů užití prezentuje využití funkčních požadavků na konkrétních případech, čímž pomáhá vysvětlit požadované funkcionality a poskytuje pro ně detailnější popis, tedy případy užití. Model případů užití je tvořen seznamem aktérů a diagramem případů užití, které jsou níže popsány. [36]

### 2.6.1 Seznam případů užití

#### UC1. Detekce barvy z kamery

Po rozkliknutí "Camera" v hlavním menu se uživateli zobrazí kamera a v horní části nejprve dojde k inicializaci. Ve chvíli, kdy je kamera připravena, lze kliknout na požadovaný objekt, který vygeneruje objekt na místě stisku spolu s informačním boxem s názvem barvy, ukázkou a přesným kódem. Pro detekce jiné barvy stačí opět kliknout na místo, kde má barva být detekována.

#### UC2. Detekce barvy z obrázku

Po rozkliknutí "Camera" v hlavním menu je možné v levém spodním rohu kliknout na ikonu galerie. Po povolení přístupu může uživatel vybrat požadovanou fotografií či obrázek ze své galerie, který je načten na obrazovku. Uživatel může libovolně vybírat stiskem místo pro určení barvy, kde se následně vytvoří orámované kolečko. Vybraná barva spolu se jménem a kódem se objeví na informačním boxu uprostřed obrazovky.

#### UC3. Zobrazení detailu barvy

Pro zobrazení detailu barvy je možné zvolit dva postupy. První po výběru a detekci barvy z kamery či obrázku, a to kliknutím na obrázek palety ve spodní části aplikace. Druhým způsobem je výběr "Library" v hlavním menu a výběrem požadované barvy. Na stránce s detailem je zobrazena barva, její název, HEX, CMYK a HSV kód spolu s monochromatickou, analogickou a komplementární paletou.

#### UC4. Uložení barvy

Po výběru barvy z kamery či obrázku je možné uložit zvolenou barvu do knihovny pomocí stisku ikony uložení vedle ikony galerie. Barva se následně objeví v sekci "Library", ke které má uživatel neomezený přístup.

#### UC5. Prohlížení uložených barev

Uživatel může v sekci "Library" v hlavním menu získat přístup k celé knihovně

barev uložených z kamery či obrázku. Pomocí posuvníku lze procházet barvy seřazené od posledního uložení po nejstarší.

#### **UC6. Smazání uložených barev**

V sekci "Library" v hlavním menu získá uživatel přístup ke knihovně barev. Pro výběr barvy lze podržet stisk na požadovaném políčku, přičemž se ve vrchní části objeví tlačítka pro vymazání barvy či zrušení výběru. Uživatel takto může vybírat a mazat více barev najednou. Pro výběr více než jedné barvy stačí krátký stisk, při opětovném stisku je barva z výběru odebrána.

#### **UC7. Zobrazení nápovědy**

Nápověda použití aplikace je uživateli zobrazena při prvním spuštění aplikace. V případě, že je nápověda vyžadována v průběhu používání, lze ji najít v sekci "Help" v hlavním menu. Nápověda obsahuje plnohodnotný návod na používání aplikace včetně vizuálního doprovodu.

### **2.6.2 Diagram případů užití**

#### **2.7 Stavový diagram**

Stavový diagram patří mezi využívané nástroje objektového modelování. Slouží k vizualizaci objektů v rámci jeho životního cyklu, tedy stavů, kterými prochází, a událostí způsobujících změnu stavu [38]. Pomocí stavového diagramu lze přesně popsat logiku chování systému napříč všemi případy užití [39]. Komponentami diagramu jsou **Stav**, **Přechod stavu** a **Událost** [38].

#### **2.7.1 Stav**

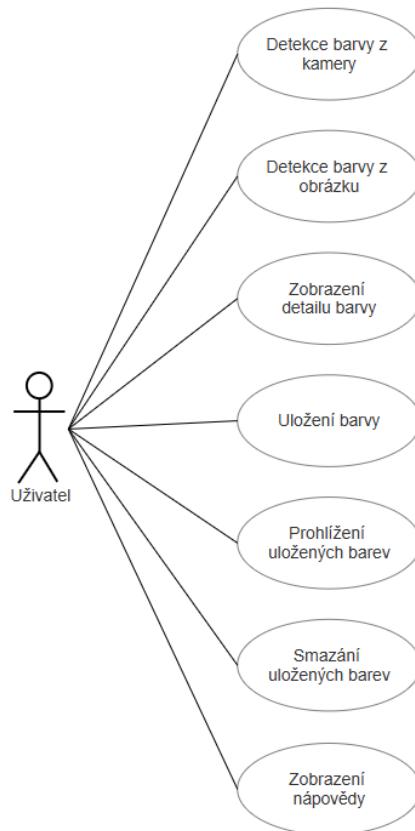
Stav popisuje stav v životním cyklu objektů či interakce, ve které se stav nachází. Je vyznačena čtvercem se zaoblenými rohy a popsána názvem stavu. Může být doplněna o další vnitřní stavy, které může stav nabývat. Příkladem stavu pro objekt Student je: přihlášený, přijatý, zapsaný, apod. Speciálním případem stavů je počáteční a koncový stav. [38]

Ve stavovém diagramu pro vyvíjenou aplikaci v této práci viditelným na 2.16, který popisuje nejdůležitější funkcionality aplikace, definujeme následující stavy:

**START** Počáteční stav popisující první spuštění aplikace.

**HELP** Zobrazení nápovědy pro používání aplikace při prvním spuštění aplikace.

**MAIN** Obrazovka hlavního menu.



■ **Obrázek 2.15** Diagram případů užití dle [36].

**AR\_CAMERA, PHOTO** Stavy, ve kterých může uživatel vybírat barvu, buď z fyzického prostoru pomocí AR nebo z obrázku z galerie.

**AR\_COLOR, PHOTO\_COLOR** Stav popisující moment detekování barvy a výpis informací o barvě v boxu, přesněji název a kód.

**GALLERY** Stav, při němž probíhá výběr fotografie či obrázku z galerie fotoaparátu.

**COLOR\_DETAIL** Obrazovka s detailními informacemi o zvolené barvě. Pozadí je určené dle zvolené barvy, na obrazovce se nachází název barvy, HSV, CMYK a HEX kód a monochromatická, komplementární a analogická paleta.

**LIBRARY** Obrazovka s knihovnou uložených barev, zobrazuje uložené barvy v orámovaném čtverci se zaoblenými rohy.

**MULTI\_SELECT** Stav, při kterém uživatel vybral v knihovně barev jednu či více barev.

## 2.7.2 Přechod stavů

Přechod stavů popisuje vztah mezi dvěma stavy, přesněji přechod z prvního stavu do druhého. Změna stavu proběhne při splnění podmínky a značí se šipkou z jednoho stavu do druhého a jeho popisem.

## 2.7.3 Událost

Událost se stane v určitém časovém okamžiku, nemá trvání, například po uplynutí času či splnění podmínky.

Ve stavovém diagramu pro vyvíjenou aplikaci v této práci viditelným na 2.16, který popisuje nejdůležitější funkcionality aplikace, definujeme následující události v přechodech stavů:

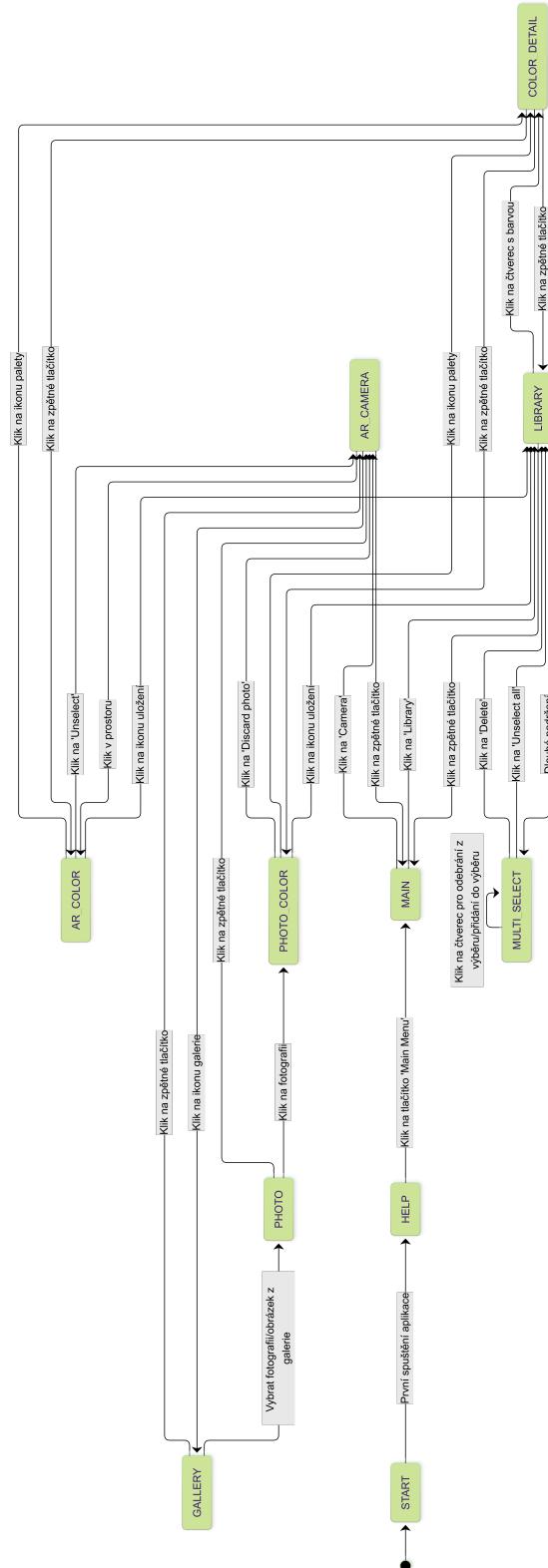
**První spuštění aplikace** Přechod z počátečního stavu do stavu START při prvním spuštění aplikace.

**Klik na {název prvku}** Přechod popisuje akci kliknutí na prvek v aplikaci. Prvkem může být tlačítko, ikona, text, zpětné tlačítko či čtverec s barvou v knihovně.

**Vybrat fotografiu/obrázek z galerie** Přechod z GALLERY do PHOTO po vybrání obrázku či fotografie z galerie fotoaparátu.

**Klik v prostoru** Kliknutí ve stavu AR\_CAMERA na místo v prostoru pro získání informace o barvě.

**Dlouhé podržení** Přechod z LIBRARY do MULTI\_SELECT dlouhým přidržením jedné z barev.



■ **Obrázek 2.16** Stavový diagram aplikace ColorLensAR, dle [38].

# Kapitola 3

## Návrh

*Chování a předpoklady aplikace byly jasně definovány v požadavcích shrnutých v analýze. Tyto poznatky budou dále využity při výběru technologií popsaných dále v této kapitole pro úspěšný vývoj aplikace. Kapitola se věnuje také návrhem architektury a uživatelského rozhraní.*

### 3.1 Technologie

Cílem této sekce je výběr technologií pro úspěšný vývoj aplikace. V první části budou přiblíženy rozdíly multiplatformního a nativního vývoje, následně dojde k porovnání vhodných frameworků a odůvodnění finální volby technologie pro implementaci aplikace.

#### 3.1.1 Nativní vs multiplatformní vývoj

Následující přehled shrnuje základní rozdíly mezi nativním a multiplatformním vývojem mobilních aplikací dle [40]. Nativní aplikace jsou vyvíjeny pro konkrétní platformu, typicky pro iOS od firmy Apple či Google Android. V závislosti na zvolené platformě je volen také specifický programovací jazyk. Pro iOS lze využít například Objective-C či Swift, pro Android je využívána Java či Kotlin.

Mezi výhody vývoje nativní aplikace patří vysoký výkon a plynulost způsobené optimalizací pro danou platformu. Zároveň je jednodušší přístup k nativním funkcím, jakými jsou například kamera či GPS, a díky přízpůsobení UX/UI konkrétní platformě stoupá spokojenosť uživatelů.

Nevýhody vývoje nativních aplikací jsou primárně vyšší náklady a delší vývoj vzhledem k nutnosti vývoje samostatných verzí pro platformy. Zároveň jsou vyšší nároky na údržbu a aktualizaci verzí.

Cíl multiplatformních aplikací je vývoj pro více platform najednou za využití jednoho frameworku. Cílem je hlavně rychlejší vytváření aplikací, významné snížení nákladů a umožňuje škálování. Využívají se moderní frameworky,

jako například React Native, které dosahují vysokého výkonu i přívětivého uživatelského zážitku napříč platformami.

Avšak i multiplatformní vývoj musí čelit různým problémům. Mezi nevýhody patří omezený přístup k některým nativním funkcím, možné problémy s kompatibilitou či nižší výkon u graficky náročných aplikací.

V této bakalářské práci je vyvíjena multiplatformní aplikace. Hlavními důvody jsou větší možnosti testování i rozšiřitelnosti a velikost prostoru pro design. Aplikace taktéž vyžaduje přístup pouze k těm nativním funkcím, které jsou jednoduše přístupné na obou platformách, což výrazně ulehčuje její vývoj.

### 3.1.2 Vývojové prostředí

Při vývoji multiplatformní aplikace, tedy aplikace dostupné pro systémy iOS i Android, případně také pro desktopové aplikace, je možné využít různá vývojová prostředí, taktéž neexistuje omezení pro operační systém. Je tedy možné aplikace vyvíjet na MacOS, Windows či Linux systémech. Příkladem může být Visual Studio Code, Microsoft Visual Studio, Eclipse či Xcode.

Přestože je vývoj multiplatformní aplikace obecně nezávislý na cílovém operačním systému, pro účely testování a ladění aplikace na zařízení se systémem iOS je nutné využít vývojové prostředí Xcode, jenž je vyžadováno platformou iOS. Ačkoliv samotný vývoj proběhne v prostředí Visual Studio Code kvůli jeho přívětivosti, Xcode sloužil jako klíčový prvek pro sestavení aplikace k jejímu úspěšnému spuštění.

Pro vývoj aplikace v Xcode je vyžadováno vlastnictví zařízení Mac s operačním systémem MacOS. Xcode je integrované vývojové prostředí, které lze bezplatně využívat a nabízí podporu velkého množství programovacích jazyků včetně C, C++, Python a další [41].

### 3.1.3 Výběr frameworku

K vývoji multiplatformních aplikací existuje několik frameworků nabízejících různé funkcionality, které usnadňují a zrychlují proces vývoje aplikací. Výběr frameworku může být ovlivněn programovacím jazykem, výběrem knihoven či možnostmi integrace. Mezi nejznámější a nejrozšířenější frameworky patří Flutter, React Native a Kotlin Multiplatform.

#### Flutter

Flutter je framework vhodný pro mobilní i desktopové a webové aplikace. Využívá programovací jazyk Dart vytvořený společností Google a podporuje využití AR v aplikacích. Výhodou frameworku Flutter je možnost zobrazení modifikace aplikace bez nutnosti rekompilace, podporuje také Material Design systém poskytující různé komponenty a nástroje následující osvědčené postupy pro tvorbu uživatelského rozhraní. [42]

### Kotlin Multiplatform

Kotlin Multiplatform je open-source technologie vytvořená JetBrains kombinující výhody nativního vývoje s možnostmi vývoje multiplatformních aplikací. Využívá programovací jazyk Kotlin a jeho největší výhodou je možnost využívání kódu napříč platformami, možnost psaní nativního kódu a snadná integrace do jakéhokoliv projektu [43]. Naproti tomu Kotlin Multiplatform nedisponuje podporou knihoven pro rozšířenou realitu.

### React Native

React Native je open-source UI framework vytvořen společností Meta Platforms. Využívá již rozšířený JavaScript/TypeScript a díky Fast Refresh funkcí mohou vývojáři prohlížet změny v aplikaci ihned po vytvoření změny v komponentách. Zároveň se React Native soustředí na uživatelské rozhraní, čímž podporuje responzivní prostředí, které lze snadno přizpůsobit. Díky rozsáhlé komunitě usnadňuje vývoj aplikací a poskytuje velké množství [42] knihoven včetně knihoven rozšířené reality.

#### 3.1.4 Shrnutí výběru frameworku

Pro vývoj této bakalářské práce je využit framework React Native, a to na základě několika faktorů. Prvním je využitý jazyk JavaScript/TypeScript, jenž je jednoduchý, responsivní a rozšířený, což usnadňuje celkový vývoj aplikace. Dalším důvodem je podpora knihoven rozšířené reality a rychlé projevení změn při psaní kódu.

#### 3.1.5 Expo

Pro zjednodušení celkového procesu vývoje je využit framework Expo, jenž je definován jako sada nástrojů a služeb postavených na React Native. Slouží k usnadnění pracovních postupů a poskytuje prostor vývojářům soustředit se na psaní samotného kódu bez nutnosti soustředit se na nastavení konfigurací a nativních nástrojů [44]. Expo slouží jako mezikrok mezi samotným vývojem v React Native a následným sestavením aplikace v Xcode.

## 3.2 Architektura a architektonické vzory

Pro návrh struktury systému a specifikaci interakcí mezi jednotlivými částmi je nutné zvolit správnou architekturu a architektonický vzor. Ta nám usnadní vývoj a škálovatelnost, zároveň určuje komplexitu našeho projektu. Architekturu spolu s architektonickým vzorem vybereme na základě požadavků a výběru technologií vhodných pro náš projekt.

Kapitola se zaměřuje na popis známých návrhových vzorů využívaných pro vývoj multiplatformních aplikací včetně jejich způsobu komunikace. Následně budou shrnuty jejich výhody a nevýhody. V závěru proběhne porovnání těchto

návrhových vzorů včetně určení nevhodnějšího výběru pro aplikaci v této práci.

### 3.2.1 MVC

MVC, celým názvem ”Model-View-Controller”, rozděluje aplikaci do tří částí, každá reprezentující rozdílnou problematiku aplikace. Její popis je vyobrazen níže dle [45].

**Model** Reprezentuje objekt nesoucí data. Může také v sobě uchovávat logiku pro aktualizaci Controller části při aktualizaci dat.

**View** View vizualizuje data, které model obsahuje.

**Controller** Působí pro obě části, tedy jak pro Model, tak pro View. Stará se o tok dat do objektu a aktualizuje zobrazení, tedy View část, pokaždé, když dojde ke změně dat. udržuje zároveň separaci mezi Model a View.

### 3.2.2 MVVM

Model MVVM obsahuje tři hlavní komponenty: Model, View a ViewModel. ViewModel se stará o izolaci View od Model a umožňuje vývoj komponenty Model bez závislosti na View. Na vysoké úrovni však funguje komunikace mezi těmito komponentami a vzájemně se do určité míry ovlivňují. Popis MVVM vychází z [46].

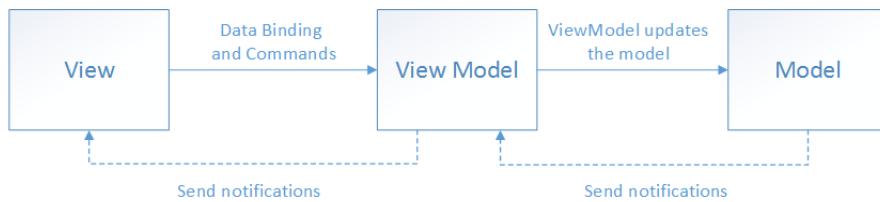
**View** Zodpovídá za definování struktury a vizuální obsah viditelný uživateli. Neobsahuje obchodní logiku, kromě specifických případů, a lze být reprezentováno šablonou dat definující rozhraní.

**ViewModel** Implementuje příkazy a vlastnosti zajišťující vazbu dat na model View a upozorňuje jej na změnu stavu. Tyto příkazy definují funkce v uživatelském rozhraní a model View určuje formu jejich zobrazení.

**Model** Model zapouzdruje data aplikace. Lze si jej představit jako reprezentaci doménového modelu aplikace zahrnující obchodní a ověřovací logiku zároveň. Je možné spojení s dalšími službami či úložiště zajišťující přístup k datům.

### 3.2.3 Component-based

Component-based je založená na opakování využitelných částech aplikace, kde každá komponenta má definovanou funkcionalitu, jenž je vložena do aplikace



■ **Obrázek 3.1** Ukázka komunikace modelu MVVM dle [46].

bez úpravy ostatních komponent. Dle zdroje [47] si lze komponentu představit jako kostku lega.

Při stavbě struktury z kostek lega, v tomto případě aplikace, lze vybírat z různých tvarů, velikostí a barev, které představují její komponenty. Každá kostka má přitom své určení, například dveře, různé konstrukční prvky či okna. Ty lze dle jejich vlastností propojit s ostatními bloky, jejich přidání či odebrání má přitom minimální dopad na strukturu. Vývoj aplikací je sice náročnější proces, avšak i v případě komplikovanějších programů je tato analogie popisující koncept Component-based vzoru relevantní.

### 3.2.4 Výhody a nevýhody zmíněných architektonických vzorů

Po stručném popsání jednotlivých návrhových vzorů je třeba před závěrečným výběrem shrnout si jejich výhody a nevýhody. Nejprve následuje popis **výhod** plynoucích z využívání jednotlivých vzorů.

**MVC** MVC umožňuje oddělení uživatelského rozhraní od dat. Tím usnadňuje změnu a další vývoj jednotlivých částí, aniž by zasahovaly jedna do druhé. Program je tak lépe rozčleněn, funkcionality se navzájem nepřekrývají a je jasně určená zodpovědnost každé z nich. [48]

**MVVM** Ačkoliv je MVVM velmi podobný MVC, nabízí určité výhody, které MVC ve svém modelu nezakomponovává. MVVM podporuje vazbu dat mezi View a ViewModel, podporuje vytváření více vztahů mezi nimi a usnadňuje Unit testování. Obchodní logika je navíc zcela oddělená od UI. [48]

**Component-based** Komponenty lze znova využívat bez nutnosti jejich změny napříč aplikací, ale také napříč různými projekty. Systémy mohou být jednoduše rozšiřovány a komponenty snadno modifikovány s minimálním zásahem do celku. Kód je zároveň díky přesnému specifikování každé komponenty organizovanější a umožňuje vytvářet čistou strukturu. [47]

Každý model však skýtá i určité nevýhody jeho využití.

**MVC** Nevýhodou MVC je zásah obchodní logiky do UI a těžší implementace testů. MVC je starší model, a tedy jeho používání může být v kontextu moderních uživatelských rozhraní obtížné. [49]

**MVVM** Vzor MVVM není vhodný pro projekty s jednoduchým UI vzhledem k jeho komplexnosti. Zároveň je náročné správně navrhnut ViewModel, což by mohlo vyústit k vysokému počtu bloků s duplicitním kódem. [50]

**Component-based** Vzhledem k vytváření oddělených komponent, Component-based přidává do systému vyšší komplexitu. Každá její komponenta musí být přesně definována, vyvinuta a spravována, což může vyústit k složitým vzájemným závislostem. Pokud jsou zároveň různé komponenty vyvíjeny různými týmy či s rozdílnými technologiemi, může jejich integrace představovat další překážku. [51]

### 3.2.5 Shrnutí a výběr architektury a architektonického vzoru

Po shrnutí výhod a nevýhod všech zmíněných vzorů jsme získali lepší obrázek o tom, jak každý z nich funguje a k jakému účelu slouží. Nyní proto můžeme vybrat vzor nevhodnější pro náš projekt. Z hlediska celkové architektury aplikace byla zvolena monolitická architektura, jelikož vyvážený projekt svým rozsahem ani očekávaným zatížením nevyžaduje použití distribuovaných architektur. Veškerá aplikační logika, vizualizační vrstva i práce s daty jsou realizovány přímo na straně klienta. Monolitický přístup umožňuje jednodušší návrh, implementaci i nasazení aplikace a zároveň snižuje režii spojenou se správou komunikace mezi jednotlivými částmi systému. Pro účely této práce tak představuje přehledné a efektivní řešení, které odpovídá charakteru vyvíjené aplikace.

Náš projekt nedosahuje takových rozměrů, aby bylo vhodné pro něj využít vzor MVVM. Zároveň jeho obousměrný tok dat může v některých případech způsobit neočekávané vedlejší efekty. Proto je třeba hledat jednodušší řešení a vyvarovat se těm, které by mohly způsobit nadbytečné komplikace.

MVC je sice vhodný pro projekty se stejnými rozměry, jako má tato práce, avšak vhodnější volbou je Component-based, a to z několika následujících důvodů. MVC často vyžaduje explicitní instrukce při změně modelu pro aktualizaci View, zároveň s růstem aplikace začíná být zajišťování závislostí a udržování oddělení částí mnohem náročnější. Component-based je založena na dynamičnosti a její velkou výhodou, jak již bylo zmíněno v předchozí podsekci, je právě možnost rozšířitelnosti s minimálním dopadem na celý systém. Dle [52] je React Native založený na komponentách a kromě základních a nativních komponent existují i další, vytvořené komunitou. Je proto vhodná pro náš projekt a můžeme říci, že pro zájmy této práce a v souladu se standard-

ními postupy v softwarovém inženýrství byla vybrána správná architektura i architektonický vzor pro vyvýjenou aplikaci a můžeme pokračovat k poslední části návrhu.

### 3.3 Uživatelské rozhraní

Poslední část návrhu je věnová návrhu uživatelského rozhraní. Řídí se 10 obecnými zásadami od Jakoba Nielsena pro návrh interakcí, které stanovují obecné principy založené na problémech, se kterými se uživatelé setkávají při používání digitálních systémů [53].

Následující podsekce popisují jednotlivé části aplikace doplněné o wireframes, které představují zjednodušený **přibližný** vzhled daného uživatelského rozhraní. Návrh poslouží jako předloha pro implementaci.

#### 3.3.1 První spuštění

Po prvním spuštění aplikace se uživatel nachází na úvodní obrazovce s návodou, která vysvětluje, jak aplikaci používat a jak se zorientovat. Je složená z textu s vizuálním doprovodem. Poté lze návod ukončit stisknutím spodního tlačítka, který uživatele dovede na hlavní obrazovku s menu, viz Obrázek 3.2.

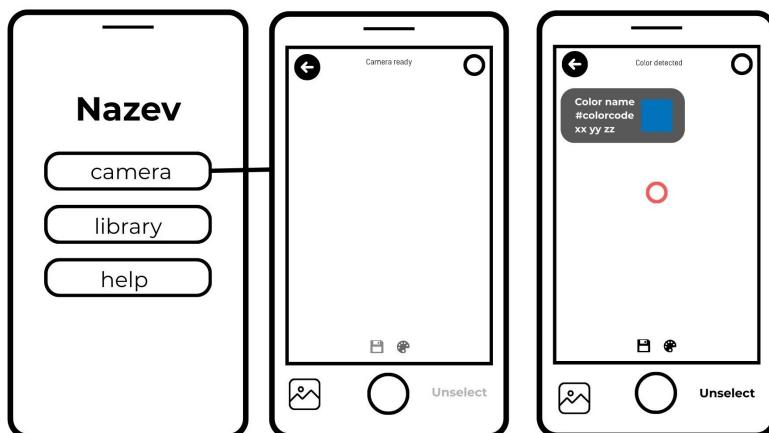


■ **Obrázek 3.2** Wireframe obrazovky s návodem při prvním spuštění a hlavního menu.

### 3.3.2 Camera

Z domovské stránky může uživatel pokračovat na stránku s kamerou z hlavního menu, které je záměrně ponecháno minimalistické. Tlačítka jsou velká a jasně označená textem navigace. V kameře je jasně označené, zda lze tlačítko použít, dle jeho průhlednosti. Vybledlá tlačítka nelze stisknout. Jsou popsané buď slovně dle akce, kterou uživatel může provést, nebo prezentovány pomocí ikon reprezentující akci napříč platformami. Vzhled je volen na základě osvědčených postupů a snaží se zachovat 5. pravidlo Nielsenova desatera dle [53], které popisuje konzistentnost a dodržování konvencí platformy i celého průmyslu. Příkladem je ikona galerie či zpětné tlačítko v levém horním rohu, které uživatele zavede vždy na předchozí stránku.

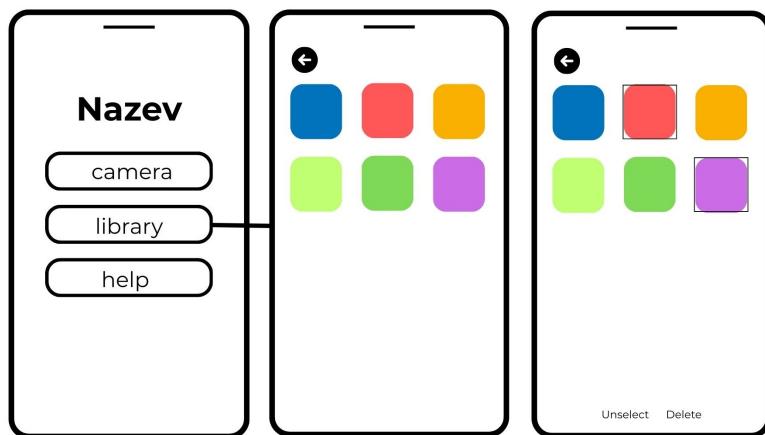
Při zvolení barvy je vyobrazen zvolený bod spolu s polopruhledným boxem obsahující čtverec s detekovanou barvou, název a kód barvy. Vzhled boxu je zachován, ať už v případě AR kamery či při detekci barvy z fotografie/obrázku z galerie. Pro udržení informovanosti uživatele dle doporučení číslo 1 v [53] je v horní části obrazovky připraven text s aktuálním statusem určující, zda je aktuálně detekována barva, zda je kamera připravena či probíhá načítání. Ukázku lze vidět na Obrázku 3.3.



■ **Obrázek 3.3** Ukázka obrazovek při spuštění Camera.

### 3.3.3 Library

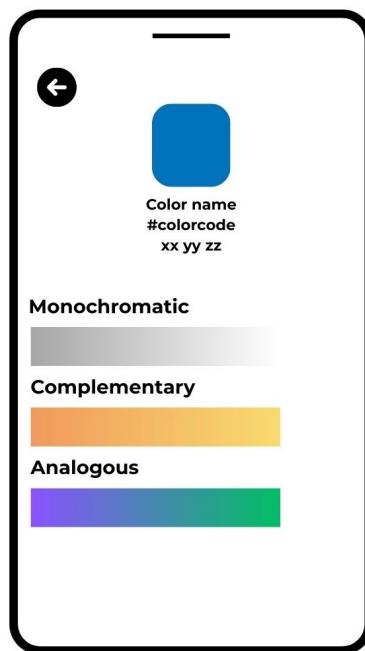
Z hlavní obrazovky je možné přistoupit na stránku s knihovnou uložených barev, jak je vidět na Obrázku 3.4. Knihovna vyobrazuje čtverce s definovaným tvarem, je zachováno minimalistické prostředí. Při výběru barev je zjevné ohraňování zvolených barev. Při odebrání výběru je ohraňování zrušeno. Ve spodní části se nachází dvě tlačítka s jasně popsanou akcí pomocí textu, tedy buď vymazání barev či odebrání označení u všech barev. Levé zpětné tlačítko je zachováno napříč celou aplikací jednotně.



■ **Obrázek 3.4** Ukázka obrazovek v knihovně barev.

### 3.3.4 Detail

Obrazovka detail je dostupná při kliknutí na barvu v knihovně či při kliknutí na ikonu palety v kameře po zvolení barvy. Detail vyobrazuje barvu spolu se jménem a dalšími kódy popisující barvu. Ve spodní části se nachází tři palety, které jsou jasně popsány názvem palety a ohraničené pro lepší odlišení od pozadí. (viz Obrázek 3.5)



■ **Obrázek 3.5** Ukázka obrazovky vyobrazující detail o barvě.

### 3.3.5 Help

Kromě prvního spuštění je možné vstoupit do obrazovky s návodem z hlavní obrazovky. Struktura je zachována pro dodržení konzistence, text je doprovázen vizuály pro vyšší atraktivitu a zřejmost informací. Spodní tlačítko je nahrazeno zpětným tlačítkem v levém horním rohu, stejně jako je tomu ve zbytku aplikace. Wireframe obrazovky je vyobrazen na Obrázku 3.6.



■ **Obrázek 3.6** Ukázka obrazovky při spuštění návodů.

### 3.3.6 Shrnutí návrhu uživatelského rozhraní

Uživatelské rozhraní aplikace zachovává konzistenci napříč všemi obrazovkami, má jednotný design, barvy i fonty. Pomocí známých a jasně rozpoznatelných ikon a textu jasně popisuje akce, které uživatel může vykonávat, aby zachoval využívané standardy a dle pravidla 6 v [53] bylo minimalizováno využití paměti. Návrh uživatelského rozhraní pomohl uzavřít kapitolu o návrhu aplikace. Ten zahrnuje také výběr architektury včetně popsání alternativ spolu s porovnáním a výběrem technologií, a to za dodržení běžných postupů softwarového inženýrství pro splnění bodů vyplývajících ze zadání této práce.

# Kapitola 4

## Implementace

Po úspěšném provedení analýzy, sestavení požadavků aplikace a vytvoření návrhu včetně výběru vhodných technologií je možné započít s implementační fází aplikace.

Následující kapitola se zabývá popisem procesu implementace aplikace dle stanovených požadavků s ukázkami použitého kódu.

### 4.1 Continuous Integration

Jedním z klíčových kroků při implementaci je zavedení *Continuous Integration* (CI). Jedná se o softwarový postup zahrnující sloučení kódu na sdíleném úložišti a automatické spouštění spolu s testováním. Vztahuje se nejčastěji k fázi sestavení nebo při integrační fázi. Hlavním cílem CI je rychlejší vyhledávání chyb a jejich snazší řešení, zlepšení kvality kódu a šetření času při ověřování a vydávání nových aktualizací. [54]

Pro zavedení CI do projektu a jeho verzování v průběhu celého vývoje bylo zvolené úložiště GitHub a proces jeho zavedení je popsán v následující podepskci.

### 4.2 GitHub

GitHub je webová platforma pro vývojáře k ukládání a spravování kódu. Umožňuje spolupráci na projektech a sdílení open-source projektů [55]. Pro vytvoření vlastních workflow pro CI/CD nyní GitHub nabízí **GitHub Actions** umožňující sestavení kódu přímo v repozitáři a spouštění testů. Pro veřejné repozitáře využívající *GitHub-hosted runners*, což je virtuální prostředí poskytnuté od GitHub, je tato funkcionality bez poplatku. Pro soukromé repozitáře je služba zpoplatněna po využití stanoveného počtu minut. Poslední možností je spouštět pipeline lokálně, pak je služba taktéž bez poplatku.

Následující popis fungování workflow v GitHub Actions vychází z [56].

Workflow je konfigurovatelný automatizovaný proces, při němž se spouští jeden či více procesů. Je definovaný souborem YAML uloženým v repozitáři a spustí se buď dle definovaného plánu, ručně či po spuštění akce v repozitáři. Workflows jsou definované v repozitáři v adresáři `.github/workflows`, v této práci přesněji v souboru `ci.yml`.

Event je specifická aktivita v repozitáři, která spustí provedení Workflow. Je zadefinována v části `on`, a to při `push` či `pull_request` pro vybrané větve.

Job je sada kroků (Steps), které se provedou na stejném runner. Buď se jedná o shell skript či akce (Actions), které budou provedeny. Kroky se provádějí ve stanoveném pořadí a jsou na sobě vzájemně závislé. V případě této práce se nejprve provádí testy, po kterých následuje sestavení aplikace pro různé platformy. Mohou být zároveň nepovinné, tedy pokud nedojde k jejich úspěšnému dokončení, lze i přesto pokračovat.

Action je předdefinovaná sada úloh či kód, který je opakovaně využitelný a provádí úkony v rámci pracovního postupu. Snižuje množství opakujícího se kódu, příkladem je načtení repozitáře, nastavení autentizace pro poskytovatele cloud služeb či nastavení správných toolchainů.

Runner je server, na kterém je spuštěna Workflow. GitHub nabízí Ubuntu Linux, Microsoft Windows i macOS runner a každý může provádět jeden Job v daný moment. Ke spuštění testů je využit `ubuntu-latest` a k sestavení iOS aplikace `macos-latest`.

Pro potřeby této bakalářské práce je využita CI, dalším krokem by mohl být *Continuous deployment* sloužící k automatizaci nasazení a publikování projektů, například přímo do obchodu s aplikacemi. Pro sdílení aplikace v AppStore je však vyžadována placená služba Apple Developer Program a aplikace musí projít schvalovacím procesem.

## 4.3 Statická analýza

Statická analýza kódu představuje důležitou součást procesu vývoje softwaru, jejímž cílem je odhalení potenciálních chyb, nekonzistence a porušení definovaných pravidel již během vývoje aplikace, a to bez nutnosti jejího spuštění. Umožňuje včasnou identifikaci problémů, které by se jinak projevily až v pozdějších fázích vývoje nebo při provozu aplikace, a přispívá tak ke zvýšení kvality, čitelnosti a dlouhodobé udržitelnosti zdrojového kódu. V této práci je statická analýza integrována do procesu *Continuous Integration (CI)*, kde je automaticky prováděna při testování aplikace. Pro kontrolu dodržování standardů a odhalování běžných chyb je využit nástroj ESLint.

### 4.3.1 ESLint

ESLint je open-source projekt pomáhající nalézt a opravit problémy v JavaScript kódu. S ESLint je možné analyzovat JavaScript kód psaný v prohlížeči i na serveru, v libovolném Frameworku i bez něj. Soustředí se na různé

typy problémů, od potenciálních chyb běhu, přes nedodržování osvědčených postupů, po problémy se stylem.

Pravidla (**Rules**) jsou základním stavebním kamenem ESLint. Každé pravidlo ověřuje, zda kód splňuje určitá očekávání a pokud tato očekávání nesplňuje, pomáhá s dalším postupem. Mohou také obsahovat další konfigurační možnosti specifické pro nějaké pravidlo. ESLint disponuje knihovnou stanovených pravidel, která analyzuje, lze však vytvářet vlastní pravidla či využívat pravidla vytvořená dalšími vývojáři. Při vyvíjení softwaru bylo využité základní nastavení ESLint bez přidaných pravidel a v CI pipeline je označené jako *optional*, jelikož hlavní prioritou CI je testování a sestavení aplikace. Avšak tímto způsobem umožnuje vylepšovat kvalitu kódu a automatizovat tento proces. [55]

### 4.3.2 Dokumentace

Nedílnou součástí kvalitního softwaru je spolu s automatizovaným testováním a statickou analýzou také dokumentace a přehledná struktura zdrojového kódu. Tyto aspekty významně přispívají k dlouhodobé udržitelnosti aplikace a usnadňují její další rozšiřování, pro samotného autora i případné spolupracovníky. Základní přehled o projektu poskytuje soubor README, který obsahuje shrnutí účelu aplikace a popis pro její sestavení a spuštění. Slouží jako vstupní bod pro nové uživatele i vývojáře.

Dalším důležitým prvkem jsou komentáře přímo ve zdrojovém kódu, které vysvětlují složitější části implementace, netriviální algoritmy či specifické chování aplikace. Komentáře nejsou využívány k popisu samozřejmých konstrukcí, ale zaměřují se na objasnění důvodů zvoleného řešení. Tyto formy dokumentace jsou v této práci provázány se statickou analýzou kódu, která prostřednictvím nástroje ESLint napomáhá udržovat konzistentní styl, čitelnost a dodržování definovaných pravidel napříč celým projektem.

## 4.4 Ukázky implementace

V následujících řádcích jsou prezentovány významné části implementace vyvíjené aplikace, jejich popis spolu s ukázkami zdrojových kódů.

Implementace se řídí architekturou stanovenou v návrhu a splňuje funkční i nefunkční požadavky uvedené při analýze. Sekce je tedy dle využité Component-based architektury rozdělená na popis jednotlivých komponent důležitých pro fungování celé aplikace.

### 4.4.1 App

Komponenta App představuje vstupní komponentu celé vyvíjené aplikace. Hlavním cílem je inicializace navigace a umožňuje obrazovkám mít přístup k navigaci. Je nutná pro fungování knihovny `@react-navigation`.

```
<NavigationContainer>
  <RootStack />
</NavigationContainer>
```

#### 4.4.2 RootStack a HomeScreen

RootStack definuje navigaci celé aplikace pomocí `createNativeStackNavigator`. Výchozí obrazovkou je **Home**, další obrazovky jsou Camera, Library, Help a ColorDetail. Navigace je vytvořena formou zásobníku `Stack.Navigator`, každá obrazovka má vlastní hlavičku. HomeScreen kontroluje, zda se jedná o první spuštění aplikace, či zda byla aplikace v minulosti již spuštěna, dle této podmínky přesměruje uživatele na Help či HomeScreen. (Viz Ukázka 4.4.2)

```
React.useEffect(() => {
  const checkFirstLaunch = async () => {
    const alreadyLaunched = await
      AsyncStorage.getItem('alreadyLaunched');
    if (alreadyLaunched === null) {
      await AsyncStorage.setItem('alreadyLaunched', 'true');
      navigation.replace('Help');
    }
  };
  checkFirstLaunch();
}, []);
```

*Ukázka implementace hooks při prvním spuštění.*

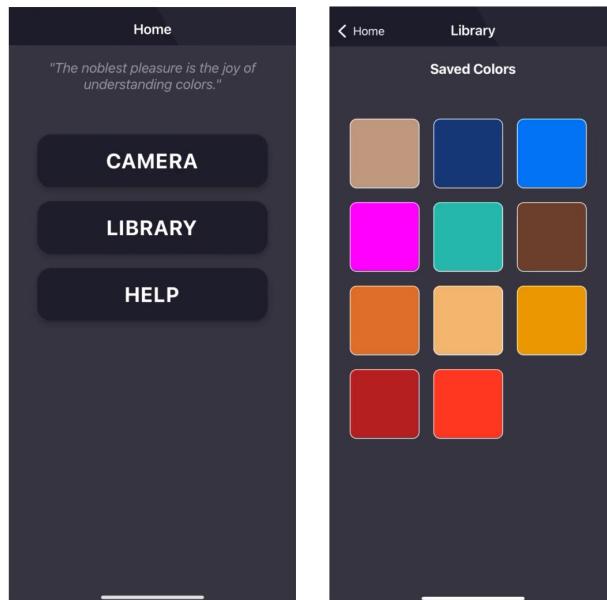
Obrázek 4.1 zobrazuje úvodní stránku HomeScreen se zobrazením navigace a vzhled obrazovky s hlavičkou header definovanou pro všechny obrazovky v RootStack.

#### 4.4.3 CameraScreen

Tato část zajišťuje výběr barev z prostoru a jejich vizualizaci v prostoru. Má na starost práci s kamerou, galerií a perzistentním ukládáním dat. CameraScreen je hlavní obrazovka kamery. Řídí zároveň komunikaci s AR scénou.

Funkce `handleTap` zjišťuje pozici kliknutí uživatele, následně se snaží z detektovat barvu pixelu, buď na základě vybrané fotografie z galerie nebo pomocí `CaptureRef`, která vytvoří dočasný snímek kamery pro získání pixelu. Díky knihovně `react-native-pixel-color` lze získat HEX kód barvy. Následně aktualizuje status. (Viz Ukázka 4.4.3)

```
const handleTap = async (e: any) => {
  const px = PixelRatio.get();
  const x = Math.round(e.nativeEvent.locationX * px);
```



■ **Obrázek 4.1** Ukázka implementace úvodní obrazovky HomeScreen a knihovny Library.

```
const y = Math.round(e.nativeEvent.locationY * px);
setTapPos({ x: e.nativeEvent.locationX, y:
    ↳ e.nativeEvent.locationY });

try {
    const tag = findNodeHandle(viewRef.current);
    if (!tag) throw new Error("View ref not found");
    const uri = capturedPhoto || (await captureRef(tag, {
        ↳ format: "png", quality: 1 }));
    const color = await PixelColor.getHex(uri, { x, y });
    await placeAtPointRef.current?.(x, y, color);

    setSelectedColor(color);
    setStatus("Color selected");
} catch (err) {
    console.warn("Pixel read error:", err);
    setStatus("Pixel read failed");
}
};
```

Ukázka implementace detekce barvy kliknutím.

CameraScreen se stará taktéž o vytvoření a uložení fotografie. V případě

snímání kamery funkce `takePhoto` vytvoří pomocí `captureRef` fotografi, pro odstranění UI na fotografi je využit časovač `setTimeout(r, 100)` a stav `uiVisible`.

K uložení fotografie slouží funkce `savePhoto` využívající nativní knihovnu `@react-native-camera-roll/camera-roll`, která zajišťuje přístup do knihovny, kam se fotografie ukládá. (Viz Ukázka 4.2)

```
await CameraRoll.saveAsset(capturedPhoto, {
    type: "photo",
    album: "ColorFinder",
});
```

■ **Obrázek 4.2** Ukázka implementace ukládání fotografie.

Poslední význačnou částí CameraScreen je možnost výběru fotografie z galerie. Tu zajišťuje funkce `pickFromGallery` s `launchImageLibrary` z knihovny `react-native-image-picker`. Pro načtení využívá časovač, během kterého se nastaví vybraná fotografie. (Viz Ukázka 4.4.3)

```
const pickFromGallery = async () => {
    try {
        const result = await launchImageLibrary({ mediaType:
            → "photo" });
        if (result.didCancel || !result.assets?.[0]?.uri) return;
        setCapturedPhoto(result.assets[0].uri);
        setStatus("Photo loaded");
        setSelectedColor("");
        setUiVisible(false);

        setTimeout(async () => {
            try {
                const tag = findNodeHandle(viewRef.current);
                if (!tag) return;
                const uri = await captureRef(tag, { format: "png",
                    → quality: 1 });
                setCapturedPhoto(uri);
            } catch (e) {
                console.warn("Snapshot after gallery load failed:",
                    → e);
            } finally {
                setUiVisible(true);
            }
        }, 500);
    } catch {
        Alert.alert("Error", "Failed to pick image");
    }
}
```

```
    }
};
```

*Ukázka implementace funkce výběru fotografie z kamery.*

#### 4.4.4 SceneAR

CameraScreen komunikuje se SceneAR pomocí ViroARSceneNavigator. SceneAR zajišťuje logiku rozšířené reality, tvorbu materiálů a umístění informačního boxu s detekovanou barvou do prostoru.

**Hooks** v React Native jsou funkce, které nám zpřístupní nějakou funkcionalitu z React [57]. Tyto funkcionality jsou využívány v celém projektu a velký význam mají právě v komponentě SceneAR. Využívá hook useEffect při registerPlaceAtPoint, kde je třeba najít bod umístění AR objektu do prostoru. V něm, jak je vidět v Ukázce 4.4.4 níže, využívá hit test performARHitTestWithPoint vysílající paprsek dle souřadnic bodu pixelu v 2D scéně. Pokud je nalezení úspěšné nastaví se pozice objektů. V opačném případě je reportována chyba. Objekty jsou vytvářeny s ViroMaterials.

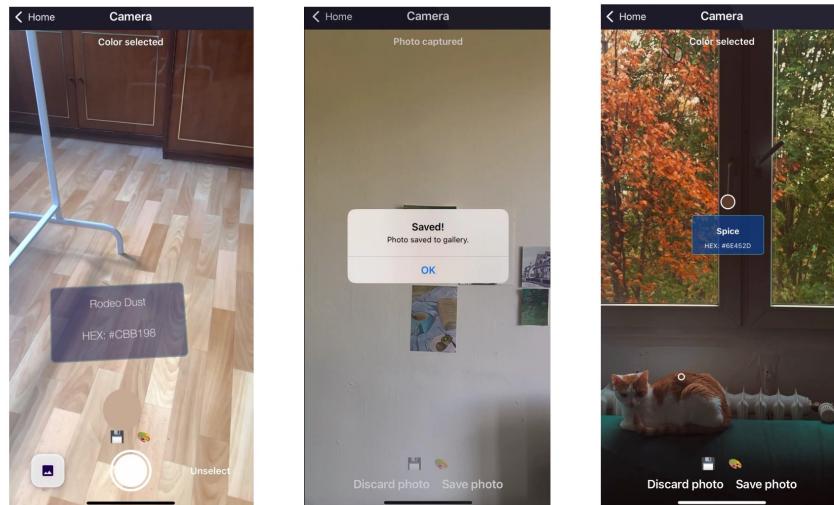
```
try {
  const results = await
    → arRef.current.performARHitTestWithPoint(x, y);
  const hit =
    results?.find((r: any) => r.type ===
      ← "ExistingPlaneUsingExtent") ||
    results?.find((r: any) => r.type === "ExistingPlane")
    ← ||
    results?.find((r: any) => String(r.type ||
      ← "").includes("Estimated")) ||
    results?.find((r: any) => r.type === "FeaturePoint");

  if (hit?.transform?.position) {
    setPlacedPos(hit.transform.position as [number,
      ← number, number]);
    setPlacedHEX(color);
    setPlacedName(colorData.colorName);

    → props.sceneNavigator?.viroAppProps?.setSelectedColor?(color);
    reportStatus("Placed");
    return true;
  }
  return false;
} catch {
  reportStatus("Hit-test failed");
```

```
    return false;
}
```

*Ukázka implementace hit testu.*



■ **Obrázek 4.3** Ukázka implementace obrazovky s kamerou v AR módu, výběru z galerie a potvrzení uložení obrázku.

#### 4.4.5 ColorData

Komponenta `useColorData` řídí logiku zobrazenou následně v detailu barvy. První část zajišťuje výpočty v převodech barevných systémů. Detail vyobrazuje HEX, CMYK a HSL, následující příklad vyobrazuje funkci s výpočtem převodu z HEX do CMYK. Nejprve dojde k separaci jednotlivých složek, následně proběhne převod. (Viz Ukázka 4.4.5)

```
const hexToCmyk = (hex: string) => {
  const { r, g, b } = hexToRgb(hex);
  const rNorm = r/255;
  const gNorm = g/255;
  const bNorm = b/255;
  const k = 1 - Math.max(rNorm, gNorm, bNorm);
  const c = k === 1 ? 0 : (1 - rNorm - k) / (1 - k);
  const m = k === 1 ? 0 : (1 - gNorm - k) / (1 - k);
  const y = k === 1 ? 0 : (1 - bNorm - k) / (1 - k);
```

```

return `C: ${Math.round(c*100)}%, M:
→ ${Math.round(m*100)}%, Y: ${Math.round(y*100)}%, K:
→ ${Math.round(k*100)}%`;
};

```

*Ukázka implementace funkce převodu kódů.*

Zároveň se díky funkci `getContrastTextColor`, která detekuje kontrast konkrétní barvy, mění ohraničení v paletách a text. Tak je postaráno o čitelnost textu.

Druhá část komponenty je určena pro tvorbu palet. Následující ukázka zobrazuje celý postup výpočtu monochromatické, analogické i komplementární palety. U monochromatické palety byl zvolen offset, díky kterému je zajištěno při jasných či naopak velmi tmavých odstínech posun zajišťující, že paleta nebude vyobrazovat pouze černé či pouze bílé odstíny. (Viz Ukázka 4.4.5)

```

OFFSET = 20

...
const generateMonochrome = (h: number, s: number, l: number)
→ => {
  const step = 10;
  const count = 3;
  const minL = l - count * step;
  const maxL = l + count * step;

  let shift = 0;

  if (minL < OFFSET) {
    shift = OFFSET - minL;
  }

  if (maxL > 100 - OFFSET) {
    shift = 100 - OFFSET - maxL;
  }

  const palette = [];
  for (let i = -count; i <= count; i++) {
    if (i === 0) continue;
    let newL = l + i * step + shift;
    newL = Math.min(100, Math.max(0, newL));
    palette.push(`hsl(${h}, ${s}%, ${newL}%)`);
  }
  return palette;
}

```

```
};

const generateComplementary = (h: number, s: number, l:
→  number) => {
  const step = 10;
  let compHue = (h + 180) % 360;
  const palette = [];
  for (let i = -3; i <= 3; i++) {
    if( i == 0 ) continue;
    let newCompHue = (compHue + i * step) % 360;
    palette.push(`hsl(${newCompHue}, ${s}%, ${l}%)`);
  }
  return palette;
};

const generateAnalogous = (h: number, s: number, l: number) =>
→  {
  const step = 20;
  const palette = [];
  let newHue = h;
  for (let i = -3; i <= 3; i++) {
    if( i == 0 ) continue;
    newHue = (h + i * step) % 360;
    palette.push(`hsl(${newHue}, ${s}%, ${l}%)`);
  }
  return palette;
};
```



■ **Obrázek 4.4** Ukázka implementace obrazovky s detailem barvy, kódy a paletami.

## Kapitola 5

# Testování

*Tato kapitola je věnována testovací fázi vývoje softwaru. Díky testům lze identifikovat nedostatky aplikace a snížit tak riziko jejího selhání. Nalezením chyb v raných fázích vývoje lze předejít jejich případnému šíření do dalších částí projektu.*

*Následující podsekce jsou věnovány dvěma typům testování aplikace. Prvním je unit testování, následuje testování uživatelského rozhraní. Každá sekce obsahuje stručný popis metody s následným popisem samotného testování.*

### 5.1 Unit testování

Unit testování je metoda při vyhodnocování softwaru věnující speciální pozornost komponentám, metodám či nejménším možným jednotkám kódu. Testování zahrnuje taktéž izolaci těchto jednotek, aby mohla být řádně ověřena jejich funkčnost předtím, než budou integrovány do systému. Tato sekce vychází z [58].

Unit testy urychlují vývojový proces díky možnosti jejich automatizace. Z dlouhodobého hlediska pomáhají unit testy šetřit náklady, jelikož v pozdějších fázích vývoje, kde náklady bývají značně vyšší, je třeba méně ladění. Důvodem je také uvažování o potenciálních problémech již v raných fázích a možnost jejich předcházení. Podporují také vyšší kvalitu kódu a celkově vytváří spolehlivé kódové základy.

Pro unit testování existuje 5 obecně uznávaných kroků, které se provádějí postupně.

#### 1. Identifikace jednotky

V této části probíhá výběr jednotky, která následně bude testována. Může se jednat o funkci, třídu, metodu či komponentu.

## 2. Výběr přístupu

V části výběru přístupu je rozhodováno o typu testování, které má být provedeno. Může se jednat o manuální testování či automatizované testování za využití některého z dostupných frameworků.

## 3. Vytvoření testovacího prostředí

K provedení samotného testování je třeba správný výběr prostředí. Je nutné, aby splňovalo veškeré podmínky pro provedení testů včetně testovacích dat a závislostí. Využívá se integrované vývojové prostředí (IDE) podporující unit testování.

## 4. Vytvoření a použití testovacích jednotek

Ve 4. části probíhá výběr testovacího frameworku a napsání testovacích případů, které budou využity. Kompilátor převede testy napsané ve zvoleném programovacím jazyce na spustitelný kód. Následně je třeba potvrdit výsledky testů.

## 5. Odstranění chyb a vyřešení problémů

Pokud některý z testovacích případů selže, je nutné odladit kód a zjistit příčinu problému. Následně pro potvrzení, že je chyba opravdu opravena, se spustí znova jednotkové testy.

Při unit testování je možné využít některý z dostupných frameworků, které usnadňují organizaci, report chyb a vyhodnocování testů. Pro Javascript je možné využít Jest či Mocha frameworky, Pytest je určen pro testy v jazyce Python, existuje však mnoho dalších.

### 5.1.1 Osvědčené postupy pro unit testování

V následujících řádcích jsou zmíněny vybrané osvědčené postupy při unit testování, které pomohou k lepším výsledkům a vyšší kvalitě testování.

#### Testování co největšího množství kódu

Je důležité otestovat a vyhodnotit co nejvíce kritických částí kódu. Ne vždy lze otestovat 100 % kódu, ale developer by měl vždy mířit na dostatečně vysoké množství, například 70 % až 80 %.

#### Využívání CI/CD pipeline

Použití *continuous integration/continuous deployment* je klíč k procesu testování díky automatizaci testovacích funkcí. Díky spouštění CI/CD pipeline jsou automatizované unit testy spouštěny vždy v případě provedení změn v kódu.

### Zvažování krajních případů

Okrajové případy reflektují extrémní případy používání softwaru, ke kterým dochází u jednotek či provozních parametrů. Příklady těchto chyb zahrnují přístup k poli mimo jeho rozsah, index překračující povolenou hodnotu a další. V takových případech je často nutná restrukturalizace kódu.

#### 5.1.2 Ukázka

Následuje ukázka unit testů vyvíjeného software pro uzavření sekce o unit testování. Následující unit test zjišťuje, zda je pro poskytnuté barvy správně proveden převod z RGB do CMYK.

```
describe('RGB to CMYK Conversion', () => {
  it('converts pure cyan #00FFFF to CMYK with C=100%', () => {
    const { result } = renderHook(() =>
      → useColorData('#00FFFF'));
    expect(result.current.cmykCode).toBe('C: 100%, M: 0%, Y:
      → 0%, K: 0%');
  });

  it('converts pure magenta #FF00FF to CMYK with M=100%', () => {
    const { result } = renderHook(() =>
      → useColorData('#FF00FF'));
    expect(result.current.cmykCode).toBe('C: 0%, M: 100%, Y:
      → 0%, K: 0%');
  });

  it('converts pure yellow #FFFF00 to CMYK with Y=100%', () => {
    const { result } = renderHook(() =>
      → useColorData('#FFFF00'));
    expect(result.current.cmykCode).toBe('C: 0%, M: 0%, Y:
      → 100%, K: 0%');
  });

  it('converts black #000000 to CMYK with K=100%', () => {
    const { result } = renderHook(() =>
      → useColorData('#000000'));
    expect(result.current.cmykCode).toBe('C: 0%, M: 0%, Y: 0%,
      → K: 100%');
  });

  it('converts white #FFFFFF to CMYK with all 0%', () => {
```

```
const { result } = renderHook(() =>
  → useColorData('#FFFFFF'));
expect(result.current.cmykCode).toBe('C: 0%, M: 0%, Y: 0%,
  → K: 0%');
});

it('formats CMYK string correctly', () => {
  const { result } = renderHook(() =>
    → useColorData('#FF5733'));
  expect(result.current.cmykCode).toMatch(/^C: \d+%, M:
    → \d+%, Y: \d+, K: \d+\%$/);
});
});
```

V druhé ukázce je proveden test správného zobrazení textu. Jelikož na obrazovce s detailním barvou barva pozadí odpovídá vybranému odstínu, je třeba uvažovat kontrast s textem. Proto na světlém pozadí text a ohraničení palet je černé, naopak při tmavém odstínu je text s ohraničením nastaven na bílou barvu.

## 5.2 Testování uživatelského rozhraní

Následující sekce se věnuje uživatelskému testování ověřující funkčnost UI/UX. Spočívá v interakci skutečných lidí s vytvořeným produktem a sledování jejich reakcí a chování. Testování je možné provádět několika způsoby včetně laboratorního testování pohybu očí. V každém případě je však nezbytným krokem pro zajištění příjemného a efektivního zážitku pro uživatele. [59]

### 5.2.1 Výběr uživatelů

Pro uživatelské testování byly vybrány 4 osoby ze sociálního okruhu autorky práce ve věku od 15 do 51 let. Tři osoby využívají operační systém iOS na denní bázi, jedna z nich naopak disponuje zařízením se systémem Android. Testování však proběhlo ve všech případech na zařízení se systémem iOS. Vzhledem k jednotnému prostředí aplikace na obou platformách jsou však rozdíly při používání aplikace minoritní. Jedna z osob je diagnostikována s poruchou barevného vidění, zbylé osoby bez potíží.

### 5.2.2 Testovací scénáře

Testování aplikace proběhlo na zařízeních iPhone XR a iPhone 11 Pro Max. Mobilní aplikace již byla předem nainstalovaná do zařízení, hlavním cílem testování bylo ověření funkčnosti a intuitivnosti uživatelského rozhraní a správné fungování rozšířené reality.

```
describe('Contrast Text Color', () => {
  it('uses dark text for light background (white)', () => {
    const route = {
      params: { color: '#FFFFFF' },
    };

    const { getByText } = render(<ColorDetail route={route}>
      </>);
    const title = getByText('Mocked Color Name');

    // Should have black text color (#000)
    expect(title.props.style).toEqual(
      expect.arrayContaining([
        expect.objectContaining({ color: '#000' })
      ])
    );
  });
});

it('uses light text for dark background (black)', () => {
  const route = {
    params: { color: '#000000' },
  };

  const { getByText } = render(<ColorDetail route={route}>
    </>);
  const title = getByText('Mocked Color Name');

  // Should have white text color (#fff)
  expect(title.props.style).toEqual(
    expect.arrayContaining([
      expect.objectContaining({ color: '#fff' })
    ])
  );
});
```

■ **Obrázek 5.1** Ukázka unit testů.

Jednotlivé testy proběhly pod vedení autorky práce, pokud byly potřebné prerekvizity, byly uživateli poskytnuty před samotným testováním. Po testování každý z testerů vypověděl zpětnou vazbu vztahující se k fungování aplikace pro následnou analýzu. Ukázky scénářů z uživatelského testování se nachází v příloze A. Zahrnují detekování barvy za pomocí rozšířen reality, mazání barev, vytváření a ukládání fotografie i nalezení cesty k detailu barvy.

### 5.2.3 Výsledky uživatelského testování

Všichni uživatelé splnili jednotlivé testovací scénáře bez návodů či jiného zásahu. Uživatelé ocenili přehlednou navigaci, jedna z osob zdůraznila užitečnost zpoloprůhledných tlačítek ve chvíli, kdy je není možné využívat. Pro osobu s poruchou barevného vidění bylo vše zřetelné, při testování využila porovnání dvou záměnných barev, čímž aplikace prokázala funkčnost při snaze o pochopení rozdílnosti odstínů. Ve funkčnosti aplikace uživatelé neidentifikovali zásadní problémy, v jednom případě proběhlo nepřesné umístění barvy v rozšířené realitě.

## Kapitola 6

# Vyhodnocení a další rozvoj

Závěrečná kapitola prezentuje výsledky této bakalářské práce spolu s případnými možnostmi rozvoje vyvíjené aplikace.

## 6.1 Výsledky práce

Následující sekce se věnuje celkovému zhodnocení výsledků jednotlivých fází vývoje po implementaci a testování. Zároveň poskytuje vyhodnocení plnění funkčních a nefunkčních požadavků a porovnání s existujícími aplikacemi.

### 6.1.1 Shrnutí výsledků

Analytická část poskytla důkladnější vhled do teorie barev spolu s popisem barevných systému. Část byla věnována také problematice barevného vidění a poruchám barevného vidění, následované rešení existujících aplikací včetně zaznamenání jejich výhod a nevýhod. Díky analytické části bylo možné sestavit funkční a nefunkční požadavky pro vyvíjenou aplikaci.

Vybraná *Component-based* architektura se osvědčila při vývoji a poskytla díky rozdelení do komponent lepší organizaci a přehlednost celého kódu. Díky volbě této architektury lze aplikaci v budoucnosti dále rozšiřovat a umožnila také snadné testování.

Volba VSCode spolu s Xcode pro spouštění aplikace se ukázalo jako vhodným kompromisem. VSCode poskytuje přehlednější prostředí a více možností přizpůsobení. Vzhledem k tomu, že bylo nutné při implementaci spouštět aplikaci na fyzickém telefonu, jelikož simulátor nepovoluje rozšířenou realitu, Xcode sloužil jako funkční prostředník při sestavení aplikace pro iPhone.

Implementační část sloužila k vývoji aplikace, aby následně mohla být řádně otestována a mohly být vyladěny její nedostatky.

Výsledkem bakalářské práce je aplikace pro platformu iOS i Android detekující barvy za využití rozšířené reality, celý proces vývoje tak lze považovat za úspěšný.

### 6.1.2 Splnění požadavků

Výsledná aplikace splňuje veškeré nefunkční požadavky stanovené v analýze. Jedná se o multiplatformní aplikaci pro Android i iOS, vhodná pro modely Apple iPhone 8 a Android 7.0 a vyšší, využívá rozšířenou realitu k detekci barev. Aplikace zároveň funguje plynule a díky jednoduchému UI a zvoleným UI prvky je přístupná všeobecnému publiku. Aplikaci je možné snadno rozšiřovat o další palety i možnosti interakce.

Veškeré funkční požadavky s prioritou *must have* a *should have* byly splněny. Taktéž došlo z většiny ke splnění požadavků označených *could have*. V souladu s časovým rozvržením nebyly implementovány požadavky s prioritou *will not have*.

### 6.1.3 Porovnání s existujícími aplikacemi

V poslední subsekci dojde k porovnání výsledné aplikace s již existujícími konkurenčními aplikacemi, které byly představeny v analýze. Jedná se o Color Blind Pal, Color Identifier: Color Picker a Color Name Recognizer.

#### Color Blind Pal

Aplikace umožňuje rozpoznávat barvy z kamery, tedy má podobnou úlohu, jako aplikace vyvinutá v této práci. Oproti Color Blind Pal je však implementována funkce rozšířené reality a díky možnosti volit bod na kamere stiskem se zaměnilo automatickému přeskakování mezi barvami na spektru, jako v Color Blind Pal, které působí chaoticky. Funkcí navíc je také možnost detekce barvy z fotografie a ukládání barev. Uživatel se k nim tak později může vrátit.

#### Color Identifier: Color Picker

Color Identifier: Color Picker disponuje velkou řadou stejných funkcionalit, jako vyvíjená aplikace. Dokáže detektovat barvu z kamery, ukládat si nové barvy a vytvářet barevné palety. Vyvíjená aplikace v této práci, stejně jako v předchozím porovnání, nabídne navíc AR režim, kterým ani jedna z aplikací nedisponuje. Zároveň je velkou výhodou aplikace vyvíjené v této práci její jednoduchost. Color Identifier: Color Picker obsahuje mnoho funkcionalit, avšak toto množství zhoršuje uživatelskou přívětivost a v aplikaci lze nalézt nadbytečné funkce. Jednoduchost a snaha soustředit se na nejdůležitější složky softwaru, které může uživatel využívat, umožňuje vytvořit přátelštější prostředí.

#### Color Name Recognizer

Color Name Recognizer umožňuje detekci barev z kamery i fotografií. Její výhodou je velká interakce s kamerou a možnost pozastavení kamery pro detekci barev. Tato funkcionalita byla implementována i v aplikaci v této práci. V režimu kamery se však aplikace Color Name Recognizer potýká se stejným

problémem, jako v prvním případě, a sice přeskakování barev a chaotičnost. Díky těmto dvěma příkladům byl při vývoji kladen důraz na větší konzistenci a stabilitu aplikace a bylo navrženo řešení pro lepší uživatelskou přívětivost.

## 6.2 Možný rozvoj aplikace

Tato sekce popisuje další možný rozvoj vytvořené mobilní aplikace v budoucnu. Je rozdělena do dvou podsekcí, první se týká možnosti rozvoje vyplývající z testování, druhá část vyplývá z požadavků.

### 6.2.1 Rozvoj na základě výsledků testování

Díky uživatelskému testování vyplynulo několik užitečných návrhů pro budoucí možný rozvoj aplikace. Jedním z návrhů byla větší interakce s kamerou, možnost zoomu či podpora širokoúhlého objektivu. Zároveň byl zmíněn návrh pro vytvoření filtrů, které by simulovaly poruchy barvného vidění pro lepší pochopení problematiky.

V rámci dalšího rozvoje by mohlo dojít také k lepší implementaci návodů, například interaktivním způsobem jako překryv obrazovky s šipkami, kam je třeba kliknout pro danou akci.

Tyto návrhy nemění fungování samotné aplikace, avšak jejich implementace by mohla vylepšit celkový zážitek z jejího používání.

### 6.2.2 Rozvoj na základě požadavků

Vhodným adeptem pro další rozvoj aplikace jsou neimplementované funkční požadavky. Jedná se o požadavky s prioritou *will not have* a jeden s prioritou *could have*. Jelikož se jedná o požadavky, které plynule navazují na již implementované funkcionality, bylo by vhodné, i díky jednoduchosti integrace, implementovat je jako jedny z prvních.

# Závěr

Cílem této bakalářské práce bylo vytvořit multiplatformní aplikace detekující barevy z kamery za pomocí rozšířené reality a vytváření odpovídajících barevných schémat. To zahrnuje provedení analýzy, vytvoření návrhu, implementace, testování aplikace a následné vyhodnocení výsledků s návrhem dalšího možného rozvoje aplikace.

Prvním krokem byla analýza teorie barev a analýza barevného vidění spolu s popisem poruch barevného vidění. Zahrnovala taktéž rešerši konkurenčních aplikací včetně vyhodnocení jejich výhod a nevýhod. Na základě získaných poznatků proběhlo sestavení funkčních a nefunkčních požadavků spolu s případy užití. Kapitola byla zakončena sestavením stavového diagramu.

V druhé kapitole proběhlo zvolení využívaných technologií a zvážení dalších možných řešení. Následovalo porovnání architektur a návrhových vzorů a došlo ke zvolení vhodné architektury vyvíjené aplikace tak, aby vyhověla požadavkům, tedy *Component-based architektura*. Poslední část se věnovala návrhu uživatelského rozhraní a vytvoření wireframů.

Třetí kapitola se již soustředila na samotnou implementaci aplikace, popisem kontinuální integrace, organizací a rozdělením do jednotlivých komponent dle funkcionalit. Pro implementaci bylo v návrhu zvoleno IDE VSCode, spolu s Xcode a frameworkem Expo pro sestavení a pro využití rozšířené reality se nevhodnější volbou stala Viro knihovna. Nejvýznačnější části jsou doplněny ukázkami kódu.

Navazující část o testování popisuje unit testování a osvědčené postupy při testování během vývoje. Součástí jsou také ukázky unit testů vyvážené aplikace. Následně proběhlo uživatelské testování s reálnými uživateli pro ověření funkčnosti a analýza výsledků testování. V závěru proběhlo zhodnocení výsledků práce, porovnání implementace s existujícími aplikacemi a na základě výsledků testování a vyhodnocení plnění funkčních požadavků byla kapitola zakončena možným rozvojem aplikace v budoucnu.

Tak byly veškeré dílčí cíle a hlavní cíl splněny, tedy došlo i k úspěšnému splnění zadání této bakalářské práce.

## Příloha A

# Ukázky scénářů uživatelského testování

## A.1 Detekování barvy v AR

**Prerekvizity:** Bez prerekvizit.

### Testovací scénář

1. Zapněte aplikaci a počkejte na načtení hlavní obrazovky či projděte návodu.
2. Klikněte na tlačítko Camera.
3. Počkejte na načtení prostředí a klikněte pro detekování barvy v prostoru.

## A.2 Uložení barvy

**Prerekvizity:** Bez prerekvizit.

### Testovací scénář

1. Zapněte aplikaci a počkejte na načtení hlavní obrazovky či projděte návodu.
2. Klikněte na tlačítko Camera.
3. Vyberte barvu kliknutím na požadovaný objekt.
4. Klikněte na ikonu uložení.

### A.3 Uložení fotografií

**Prerekviziity:** Bez prerekvizit.

**Testovací scénář**

1. Zapněte aplikaci a počkejte na načtení hlavní obrazovky či projděte nápo-vědu.
2. Klikněte na tlačítko Camera.
3. Stiskněte prostřední tlačítko pro vytvoření fotografie.
4. Klikněte na Save photo.

### A.4 Smazání 3 barev

**Prerekviziity:** Uživatel má uložených 3 a více barev v knihovně.

**Testovací scénář**

1. Zapněte aplikaci a počkejte na načtení hlavní obrazovky či projděte nápo-vědu.
2. Klikněte na tlačítko Library.
3. Dlouhým stiskem vyberte libovolnou barvu.
4. Krátkým stiskem vyberte dvě další barvy.
5. Klikněte na Delete.

### A.5 Zobrazení detailu právě detekované barvy

**Prerekviziity:** Bez prerekvizit.

**Testovací scénář**

1. Zapněte aplikaci a počkejte na načtení hlavní obrazovky či projděte nápo-vědu.
2. Klikněte na tlačítko Camera.
3. Počkejte na načtení prostředí a klikněte pro detekování barvy v prostoru.
4. Klikněte na ikonu palety pro zobrazení detailu barvy.

# Bibliografie

1. GOETHE, Johann Wolfgang von. *Theory of Colours*. Přel. EASTLAKE, Charles Lock. London: John Murray, 1840. With notes by Charles Lock Eastlake.
2. ADAMS, C. R. “in Experiments, where Sense is Judge” – Isaac Newton’s Tonometer and Colorimeter. *Journal of the Oughtred Society*. 2013, roč. 22, č. 1, s. 41–45. Spring issue; JOS Plus Supplement.
3. LIBRARIES, Smithsonian; ARCHIVES. *The Science of Color* [online]. Smithsonian Libraries a Archives, 2025. [cit. 2025-07-23]. Dostupné z: <https://library.si.edu/exhibition/color-in-a-new-light/science>. Accessed July 23, 2025.
4. GAGE, J.; GROVIER, K. *Colour in Art*. Thames a Hudson Limited, 2023. World of Art. ISBN 9780500778807. Dostupné také z: <https://books.google.cz/books?id=mujNEAAAQBAJ>.
5. LIBRARIES, Smithsonian; ARCHIVES. *The science of Color / Medium* [online]. 2025. [cit. 2025-07-28]. Dostupné z: <https://library.si.edu/exhibition/color-in-a-new-light/science>.
6. VACKOVÁ, Aneta. *Teorie barevného vidění* [online]. Brno, 2013 [cit. 2025-07-23]. Dostupné z: [https://is.muni.cz/th/u7qow/Teorie\\_barevneho\\_videni.pdf](https://is.muni.cz/th/u7qow/Teorie_barevneho_videni.pdf). Bakalářská práce. Masarykova univerzita, Lékařská fakulta. Vedoucí práce Ph.D. PETR VESELÝ DiS.
7. SENGUPTA, D.L.; SARKAR, T.K. Maxwell, Hertz, the Maxwellians and the early history of electromagnetic waves. In: *IEEE Antennas and Propagation Society International Symposium. 2001 Digest. Held in conjunction with: USNC/URSI National Radio Science Meeting (Cat. No.01CH37229)*. 2001, sv. 1, 14–17 vol.1. Dostupné z DOI: 10.1109/APS.2001.958782.
8. WIKIMEDIA COMMONS CONTRIBUTORS. *Cone response* [online]. 2008. [cit. 2025-07-28]. Dostupné z: <https://commons.wikimedia.org/wiki/File:Cone-response-en.png>.

9. JELEN, Vojtěch. *Otázka vlivu barev na spotřebitelské chování v kontextu jejich percepce a symboliky* [online]. Praha, 2023 [cit. 2025-07-23]. Dostupné z: <https://dspace.cuni.cz/bitstream/handle/20.500.11956/188964/130380558.pdf?sequence=1>. Bakalářská práce. Univerzita Karlova, Fakulta humanitních studií. Vedoucí práce Ph.D. PHDR. VÁCLAV HÁJEK.
10. PANTONE LLC. *What Is Color?* [online]. Pantone, 2025. [cit. 2025-07-23]. Dostupné z: [https://www.pantone.com/articles/color-fundamentals/what-is-color?srsltid=AfmB0opg6UBko2e-BNFPDss2Tu\\_9yYq8S7QT2e\\_GIcs\\_cMv0m\\_fwgiri](https://www.pantone.com/articles/color-fundamentals/what-is-color?srsltid=AfmB0opg6UBko2e-BNFPDss2Tu_9yYq8S7QT2e_GIcs_cMv0m_fwgiri).
11. HANULÍKOVÁ, Gabriela. *Zraková ostrost a citlivost vliv fyzikálních parametrů prostředí* [online]. 2013. [cit. 2025-07-28]. Dostupné z: [https://is.muni.cz/th/gpxge/Diplomova\\_prace.pdf](https://is.muni.cz/th/gpxge/Diplomova_prace.pdf). Diplomová práce. Masarykova univerzita, Lékařská fakulta. Vedoucí práce Ph.D. MGR. VLADAN BERNARD.
12. DR. BIOLOGY. *Rods and Cones of the Human Eye* [online]. 2010. [cit. 2025-07-28]. Dostupné z: <https://askabiologist.asu.edu/rods-and-cones>.
13. SOUKUP, Jan. *Vliv světelných podmínek na vnímání barev* [online]. 2020. [cit. 2025-07-28]. Dostupné z: <https://dspace.cvut.cz/bitstream/handle/10467/91222/FBMI-BP-2020-Soukup-Jan-prace.pdf>. Bakalářská práce. České vysoké učení technické v Praze, Fakulta biomedicínského inženýrství. Vedoucí práce Ph.D. MGR. JANA URZOVÁ.
14. KITAOKA, Akiyoshi. *Strawberries memory colour: English Wikipedia* [online]. Wikimedia Foundation, 2017. [cit. 2025-07-28]. Dostupné z: [https://en.wikipedia.org/wiki/Memory\\_color\\_effect#/media/File:Strawberries\\_memory\\_colour.jpg](https://en.wikipedia.org/wiki/Memory_color_effect#/media/File:Strawberries_memory_colour.jpg).
15. WITZEL, Christoph; GEGENFURTNER, Karl R. Color Perception: Objects, Constancy, and Categories. *Annual Review of Vision Science*. 2018, roč. 4, č. Volume 4, 2018, s. 475–499. ISSN 2374-4650. Dostupné z DOI: <https://doi.org/10.1146/annurev-vision-091517-034231>.
16. BEAU LOTTO, R; PURVES, Dale. The empirical basis of color perception. *Consciousness and Cognition*. 2002, roč. 11, č. 4, s. 609–629. ISSN 1053-8100. Dostupné z DOI: [https://doi.org/10.1016/S1053-8100\(02\)00014-4](https://doi.org/10.1016/S1053-8100(02)00014-4).
17. SKELTON, Alice E.; MAULE, John; FRANKLIN, Anna. Infant color perception: Insight into perceptual development. *Child Development Perspectives*. 2022, roč. 16, č. 2, s. 90–95. Dostupné z DOI: <https://doi.org/10.1111/cdep.12447>.

18. PORTLAND, Donna; STRONG, Lynne. The Power of Colour on Election Campaigns. *The Bugle News*. 2024. Dostupné také z: <https://thebuglenews.com.au/NewsStory/the-power-of-colour-on-election-campaigns/66e8e4e36bd0c1002e76628a>.
19. AKERS, A.; BARTON, J.; COSSEY, R.; GAINSFORD, P.; GRIFFIN, M.; MICKLEWRIGHT, D. Visual color perception in green exercise: positive effects on mood and perceived exertion. *Environmental Science & Technology*. 2012, roč. 46, č. 16, s. 8661–8666. Dostupné z DOI: 10.1021/es301685g.
20. REN, Long; CHEN, Yun. Influence of Color Perception on Consumer Behavior. In: NAH, Fiona Fui-Hoon; XIAO, Bo Sophia (ed.). *HCI in Business, Government, and Organizations*. Cham: Springer International Publishing, 2018, s. 413–421. ISBN 978-3-319-91716-0.
21. HOLIŠOVÁ, Klára. *Barevné vidění*. Brno, 2007. Dostupné také z: <https://is.muni.cz/th/n43x5/>.
22. LENSTORE. *Your Guide to Colour Blindness* [online]. 2025. [cit. 2025-07-31]. Dostupné z: <https://www.lenstore.co.uk/eyecare/your-guide-colour-blindness>.
23. CAMPAIGN MONITOR. *Color Blindness, Accessibility and the Vision-Impaired in Email Design* [online]. 2019. [cit. 2025-07-31]. Dostupné z: <https://www.campaignmonitor.com/blog/email-marketing/color-blindness-accessibility-and-the-vision-impaired-in-email-design/>.
24. BEST, Janet. *Colour Design - Theories and Applications (2nd Edition)*. 22.12 Late Modernism (c.1955-1985). Elsevier, 2017. ISBN 978-0-08-101270-3. Dostupné také z: <https://app.knovel.com/mlink/khtml/id:kt011G8LH7/colour-design-theories/late-modernism-c-1955>.
25. MORTON, Jill. *Basic Color Theory* [online]. 2025. [cit. 2025-07-31]. Dostupné z: <https://www.colormatters.com/color-and-design/basic-color-theory>.
26. HUANG, Yuheng. Consumer Preferences for Food Logos. *Journal of Student Research*. 2024, roč. 13. Dostupné z DOI: 10.47611/jsrhs.v13i1.6217.
27. MIKOLÁŠOVÁ, Zuzana. *Barevné systémy a jejich aplikace* [online]. 2014. [cit. 2025-07-28]. Dostupné z: <https://naos-be.zcu.cz/server/api/core/bitstreams/830d5224-cbd9-42c3-af0d-3ff9dd034966/content>. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky. Vedoucí práce Ph.D. MGR. VLADAN BERNARD.

28. MASARYKOVA UNIVERZITA, PŘÍRODOVĚDECKÁ FAKULTA, MINERALOGIE. *Aditivní barvy* [online]. Masarykova univerzita, 2025. [cit. 2025-12-15]. Dostupné z: [https://mineralogie.sci.muni.cz/kap\\_4\\_4\\_barva/barvy\\_aditivni.htm](https://mineralogie.sci.muni.cz/kap_4_4_barva/barvy_aditivni.htm).
29. PREMO S.R.O. *Jaké jsou rozdíly mezi RGB, CMYK a Pantone?* [online]. 2025. [cit. 2025-12-08]. Dostupné z: <https://www.premocz.eu/barvy-rgb-cmyk-a-pantone>.
30. HAYES, Molly; DOWNIE, Amanda. *What is Augmented Reality?* [online]. 2025. [cit. 2025-12-13]. Dostupné z: <https://www.ibm.com/think/topics/augmented-reality>.
31. GAOL, Ford Lumban; PRASOLOVA-FØRLAND, Ekaterina. Special section editorial: The frontiers of augmented and mixed reality in all levels of education. *Education and Information Technologies*. 2022. ISSN 1360-2357. Dostupné z DOI: 10.1007/s10639-021-10746-2.
32. FIORENTINI, Vincent. *Color Blind Pal* [online]. Apple App Store, 2025. [cit. 2025-12-15]. Dostupné z: <https://apps.apple.com/cz/app/color-blind-pal/id1037744228>.
33. METELKINA, Irina. *Color Identifier: Color Picker* [online]. Apple App Store, 2025. [cit. 2025-12-15]. Dostupné z: <https://apps.apple.com/cz/app/color-identifier-color-picker/id1641085395>.
34. APPYOUN. *Paleto: Mixing Colors* [online]. Apple App Store, 2025. [cit. 2025-12-15]. Dostupné z: <https://apps.apple.com/cz/app/paleto-mixing-colors/id1425573301>.
35. XIAO JIAN, Zhao. *Color Name Recognizer Camera* [online]. Apple App Store, 2025. [cit. 2025-12-15]. Dostupné z: <https://apps.apple.com/cz/app/color-name-recognizer-camera/id6473697819>.
36. MLEJNEK, Jan. *Analýza a sběr požadavků* [online]. 2025. [cit. 2025-12-15]. Dostupné z: [https://moodle-vyuka.cvut.cz/pluginfile.php/898714/mod\\_resource/content/11/03.prednaska.pdf](https://moodle-vyuka.cvut.cz/pluginfile.php/898714/mod_resource/content/11/03.prednaska.pdf).
37. AHMAD, Khadija Sania; AHMAD, Nazia; TAHIR, Hina; KHAN, Shaista. Fuzzy MoSCoW: A fuzzy based MoSCoW method for the prioritization of software requirements. In: *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*. 2017. Dostupné z DOI: 10.1109/ICICICT1.2017.8342602.
38. FRANĚK, Zdeněk. *UML Stavový diagram* [online]. Slezská univerzita v Opavě, Obchodně podnikatelská fakulta, 2021. [cit. 2025-12-17]. Dostupné z: [https://is.slu.cz/el/opf/zima2021/INMNPOMM/um/6D\\_Stavu.pdf](https://is.slu.cz/el/opf/zima2021/INMNPOMM/um/6D_Stavu.pdf).

39. FRANĚK, Zdeněk. *Objektové metody modelování: Přednáška 5* [online]. Slezská univerzita v Opavě, Obchodně podnikatelská fakulta, 2021. [cit. 2025-12-17]. Dostupné z: [https://is.slu.cz/el/opf/zima2022/INMNK0MM/2968337/Objektove\\_metody\\_modelovani\\_5.pdf](https://is.slu.cz/el/opf/zima2022/INMNK0MM/2968337/Objektove_metody_modelovani_5.pdf).
40. KACZOR, Michal. *Nativní versus multiplatformní mobilní aplikace: jak si správně vybrat?* [online]. Argo22 s.r.o., 2025. [cit. 2025-12-17]. Dostupné z: <https://argo22.com/blog/nativni-vs-multiplatformni-aplikace/>.
41. EKREN, Ekin K. *What Is Xcode and How to Use It?* [online]. Netguru, 2022. [cit. 2025-12-17]. Dostupné z: <https://www.netguru.com/blog/what-is-xcode-and-how-to-use-it>.
42. JETBRAINS. *The Six Most Popular Cross-Platform App Development Frameworks* [online]. Kotlin Documentation, 2025. [cit. 2025-12-18]. Dostupné z: <https://kotlinlang.org/docs/multiplatform/cross-platform-frameworks.html#react-native>.
43. PAVLU, Martin. *Kotlin Multiplatform je nyní ve stabilní verzi a je připraven pro nasazení na produkci* [online]. JetBrains, 2023. [cit. 2025-12-18]. Dostupné z: <https://blog.jetbrains.com/cs/kotlin/2023/11/kotlin-multiplatform-je-nyni-ve-stabilni-verzi-a-je-pripraven-pro-nasazeni-na-produkci/>.
44. SARWAR, Farrukh. *How to build React Native apps with Expo — A Developer’s Tale* [online]. Medium, 2025. [cit. 2025-12-18]. Dostupné z: <https://medium.com/@farrukh.sarwar/react-native-with-expo-a-developers-tale-7afcaa499bdb>.
45. TUTORIALSPOINT. *Design Patterns – MVC Pattern* [online]. 2025. [cit. 2025-12-20]. Dostupné z: [https://www.tutorialspoint.com/design\\_pattern/mvc\\_pattern.htm](https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm).
46. MICROSOFT. *Model–View–ViewModel (MVVM) – .NET MAUI Architecture guidance* [online]. Microsoft Learn, 2025. [cit. 2025-12-20]. Dostupné z: <https://learn.microsoft.com/cs-cz/dotnet/architecture/maui/mvvm>.
47. DICESARE, Maria. *What is Component-Based Architecture?* [online]. Mendix, 2025. [cit. 2025-12-20]. Dostupné z: <https://www.mendix.com/blog/what-is-component-based-architecture/>.
48. SHAKURO. *MVC vs MVVM: Framework Architecture* [online]. Shakuro, 2025. [cit. 2025-12-21]. Dostupné z: <https://shakuro.com/blog/mvc-vs-mvvm#What-is-MVVM-Framework-Architecture>.
49. GURU99. *MVC vs MVVM – rozdíl mezi nimi* [online]. Guru99, 2025. [cit. 2025-12-21]. Dostupné z: <https://www.guru99.com/cs/mvc-vs-mvvm.html>.

50. GOSSMAN, John. *Advantages and disadvantages of M-V-VM* [online]. Microsoft Learn Archive, 2006-03-04. [cit. 2025-12-21]. Dostupné z: <https://learn.microsoft.com/en-us/archive/blogs/johngossman/advantages-and-disadvantages-of-m-v-vm>.
51. SAKOVICH, Natallia. *What Is Component-Based Architecture?* [online]. SaM Solutions Blog, 2025. [cit. 2025-12-21]. Dostupné z: <https://sam-solutions.com/blog/what-is-component-based-architecture/>.
52. REACT NATIVE DOCUMENTATION. *Core Components and Native Components* [online]. React Native Docs, 2025. [cit. 2025-12-21]. Dostupné z: [https://reactnative.dev/docs/intro-react-native-components?utm\\_source=chatgpt.com](https://reactnative.dev/docs/intro-react-native-components?utm_source=chatgpt.com).
53. NIELSEN, Jakob. *10 Usability Heuristics for User Interface Design* [online]. Nielsen Norman Group, 2024-01-30. [cit. 2025-12-21]. Dostupné z: <https://www.nngroup.com/articles/ten-usability-heuristics/>.
54. AMAZON WEB SERVICES, INC. *What is Continuous Integration?* [online]. Amazon Web Services, 2025. [cit. 2025-12-23]. Dostupné z: <https://aws.amazon.com/devops/continuous-integration/>.
55. STAFF, Coursera. *What Is GitHub? Use Cases and FAQ* [online]. Coursera, 2025-09. [cit. 2025-12-27]. Dostupné z: <https://www.coursera.org/articles/what-is-github>.
56. GITHUB, INC. *Understanding GitHub Actions* [online]. GitHub Docs, 2025. [cit. 2025-12-28]. Dostupné z: <https://docs.github.com/en/actions/get-started/understanding-github-actions>.
57. HARTINGER, David. *Stavy v Reactu a hook useState()* [online]. ITnetwork, 2025. [cit. 2025-12-26]. Dostupné z: <https://www.itnetwork.cz/javascript/react/stavy-v-reactu-a-hook-usestate>.
58. POWELL, Phill; SMALLEY, Ian. *What is Unit Testing?* [online]. IBM, 2025. [cit. 2025-12-28]. Dostupné z: <https://www.ibm.com/think/topics/unit-testing>.
59. CONTENTSQUARE CONTENT TEAM. *Usability Testing: what it is, its benefits, and why it matters* [online]. Contentsquare, 2024. [cit. 2025-12-28]. Dostupné z: <https://contentsquare.com/guides/usability-testing/>.

# Obsah příloh

```
/  
└── readme.txt.....stručný popis obsahu média  
└── src  
    ├── impl .....zdrojové kódy implementace  
    └── thesis.....zdrojová forma práce ve formátu LATEX  
└── examples  
    └── video .....video pro ukázku funkčnosti aplikace  
└── text .....text práce  
    └── thesis.pdf .....text práce ve formátu PDF
```