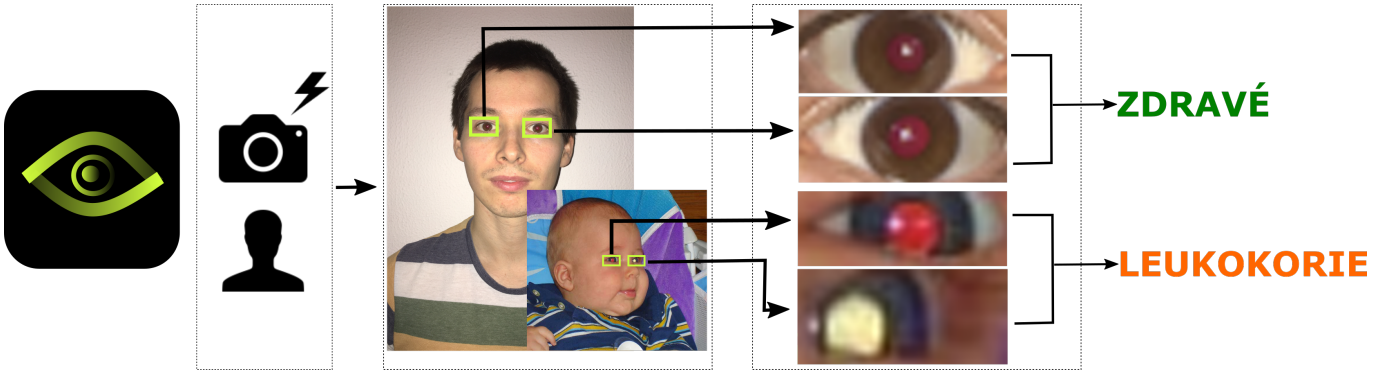
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2019
http://excel.fit.vutbr.cz

# Mobilní aplikace pro rozpoznání leukokorie ze snímku lidského obličeje

Pavel Hřebíček*




**Abstrakt**

Cílem této práce je návrh a implementace multiplatformní multijazyčné mobilní aplikace pro rozpoznání leukokorie ze snímku lidského obličeje pro platformy iOS a Android. Leukokorie je bělavý svit zornice, který se při použití blesku může na fotografii objevit. Včasnou detekcí tohoto symptomu lze zachránit zrak člověka. Samotná aplikace umožňuje analyzovat fotografii uživatele a detekovat přítomnost leukokorie. Cílem aplikace je tedy analýza očí člověka, od čehož je také odvozen název mobilní aplikace – Eye Check. K vytvoření multiplatformní aplikace byl použit framework React Native. Pro detekci obličeje a práci s fotografií byly použity knihovny OpenCV a Dlib. Komunikace mezi klientem a serverem je řešena pomocí architektury REST. Výsledkem je mobilní aplikace, která při detekci leukokorie uživatele upozorní, že by měl navštívit svého lékaře.

**Klíčová slova:** Mobilní aplikace — Eye Check — Leukokorie — Zdravé oči — iOS — Android — React Native — OpenCV — Dlib — REST

**Přiložené materiály:** Demonstrační video

*xhrebi04@stud.fit.vutbr.cz, *Fakulta informačních technologií, Vysoké učení technické v Brně*

## 1. Úvod

Tato práce si klade za cíl navrhnout a implementovat mobilní aplikaci pro detekci leukokorie ze snímku lidského obličeje. Leukokorie se může projevit na fotografii, která byla pořízena za zhoršených světelných podmínek. Místo klasického tzv. *efektu červených očí* se může projevit bělavý svit, který znamená, že oko člověka není zdravé a je nutná okamžitá návštěva lékaře. Včasnou diagnostikou lze předcházet očním onemocněním a zachránit tak zrak člověka. Právě tento fakt mě vedl k vytvoření této práce.

Na světovém trhu s mobilními telefony v dnešní době převládají zařízení s operačními systémy *iOS* a *Android*[1]. Tato skutečnost, cíl této práce poskytnout aplikaci co nejvíce uživatelům a existence moderních frameworků pro tvorbu multiplatformních mobilních aplikací mě přiměla k tomu, že jsem se rozhodl vytvořit **multiplatformní** mobilní aplikaci dostupnou právě pro operační systémy *iOS* a *Android*. Jelikož jeden z hlavních cílů této práce je poskytnout aplikaci co

[1] http://gs.statcounter.com/os-market-share/mobile/worldwide

nejširšímu spektru uživatelů, rozhodl jsem se vytvořit aplikaci **multijazyčnou** – konkrétně si uživatel může v současné době vybrat mezi češtinou a angličtinou.

Důležitou součástí celé této práce bylo vyhledání odborníka z oboru očního lékařství. Podařilo se mi kontaktovat MUDr. Barboru Žajdlíkovou, která pracuje ve Fakultní nemocnici v Brně na dětském oddělení a specializuje se přímo na oční lékařství. Díky této spolupráci jsem se dozvěděl veškeré informace o leukokorii, které vedly ke zdokonalení celého díla.

## 2. Leukokorie

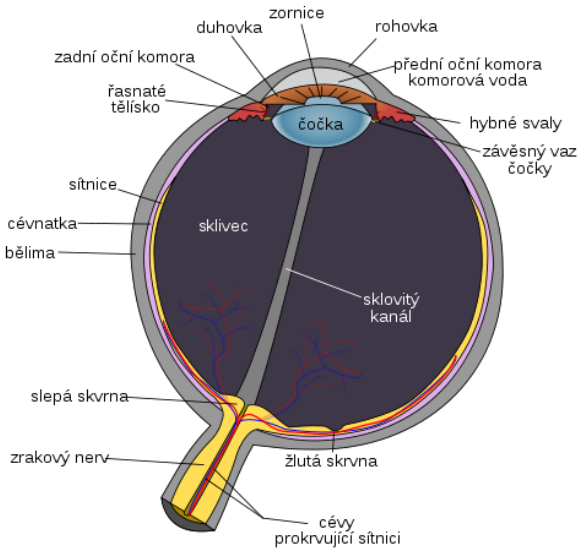
Leukokorie, nebo-li bělavý svit zornice, je vidět na obrázku 1. Představuje jeden z nejzávažnějších příznaků nitrooční patologie. Leukokorie znamená pouze symptom, nikoliv diagnózu. Mezi nejčastější příčiny leukokorie patří kongenitální katarakta, retinoblastom, retinopatie nedonošených (V. stádium), perzistující hyperplastický primární sklivec (PHPV). Mezi méně časté příčiny patří infekční onemocnění – toxoplazmóza a toxokaróza, kolobomy sítnice a cévnatky, Coatsova choroba [1].

**Obrázek 1.** Bělavý svit zornice, který představuje symptom leukokorie. Převzato z [2].

Na obrázku 2 lze vidět schéma lidského oka. Pokud je fotografie pořízena ve tmě nebo za zhoršených světelných podmínek a je použit blesk, dochází k tzv. *efektu červených očí*. Tento jev vzniká tak, že paprsek světla projde zornicí a odrazí se od sítnice, která je prokrvená. Existuje-li však nějaká překážka mezi sítnicí a předním segmentem, vzniká bělavý svit zornice nebo-li leukokorie [3].

## 3. Existující řešení

Mezi již dostupnou mobilní aplikaci pro detekci leukokorie patří *CRADLE White Eye Detector*, kterou vytvořili profesoři Bryan Shaw a Greg Hamerly z univerzity




**Obrázek 2.** Schéma lidského oka. Převzato z [4].

Baylor[2]. Mobilní aplikace je dostupná jak pro operační systém iOS[3], tak Android[4].

Oproti již existující mobilní aplikaci jsem se v této práci soustředil především na návrh a vytvoření uživatelsky přívětivějšího a graficky modernějšího uživatelského rozhraní. Další prioritou této práce bylo poskytnout aplikaci v českém jazyce pro české uživatele, a dostat tak leukokorii do většího povědomí lidí. V neposlední řadě jsem se snažil o přesnější detekci případné leukokorie a dosáhnutí co nejmenšího počtu falešně pozitivních (*false positive*) výsledků.

## 4. Návrh mobilní aplikace

### 4.1 Návrh uživatelského rozhraní

Při návrhu uživatelského rozhraní pro mobilní aplikaci Eye Check jsem se řídil již ověřeným a odzkoušeným procesem návrhu [5]. Vytvořil jsem tedy postupně *Sketch*, *Wireframe* a *Mockupy*.

*Sketch* je v podstatě ruční kresba na papír, která poskytuje tzv. *Low-Fidelity* reprezentaci aplikace [5]. Jde o velice rychlý a snadný způsob, jak lze vizualizovat první návrh samotné aplikace [6]. *Sketch* jsem tedy vytvářel pomocí tužky a papíru. Při samotném návrhu jsem se snažil o minimalizaci zbytečných tlačítek a možností, které uživateli budou zobrazeny.

*Wireframe* je ekvivalent ke kostře nebo jednoduchému návrhu webové nebo mobilní aplikace. Mezi

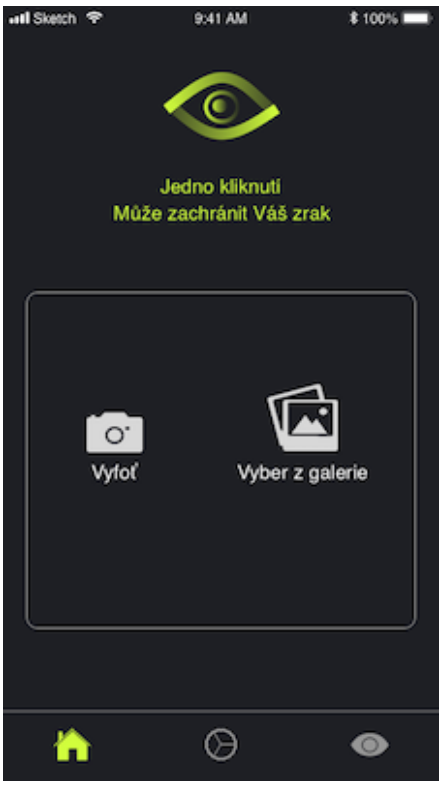
[2] http://cs.baylor.edu/~hamerly/leuko/

[3] https://itunes.apple.com/us/app/white-eye-detector/id904042354?mt=8

[4] https://play.google.com/store/apps/details?id=net.leuko.leuko_android

hlavní důvody, proč se wireframe vytváří, patří zobrazení vztahů mezi jednotlivými obrazovkami aplikace [5]. Při tomto návrhu jsem se soustředil především na minimalizaci počtu obrazovek. Samotný wireframe jsem vytvořil pomocí webové aplikace *draw.io*[5].

*Mockupy* představují velmi blízkou vizualizaci reality, se statickou reprezentací funkcionality. Při vytváření mockupů jsem již použil barvy, grafiku a fonty, které jsou zastoupené ve výsledné aplikaci. Při samotném návrhu jsem se soustředil na moderní designové zpracování aplikace. Mockup hlavní obrazovky mobilní aplikace Eye Check lze vidět na obrázku 3. Samotný návrh jsem vytvořil pomocí nástroje *Sketch*[6].




**Obrázek 3.** Mockup hlavní obrazovky mobilní aplikace Eye Check, který byl vytvořen při návrhu uživatelského rozhraní.

## 4.2 Testování uživatelského rozhraní

Cílem testování bylo zjistit, jak se uživatelům s aplikací pracuje a získat jejich zpětnou vazbu. Testování proběhlo s 11 uživateli a zúčastnili se ho uživatelé patřící do věkové skupiny 20-55 let. Mezi testovanými uživateli byli zastoupeni velmi zdatní jedinci, kteří využívají mobilní aplikace každý den, ale i uživatelé, kteří aplikace používají jen velmi zřídka.

Testování uživatelského rozhraní bylo prováděno pomocí mockupů. Pro samotné testování jsem vytvořil sadu úkolů, které měl každý uživatel splnit, a které jsou vypsány v tabulce 1. Úkoly byly zaměřené především na ověření, zda-li se uživatel v aplikaci snadno orientuje a dokáže jednoduše analyzovat fotografii na případný výskyt leukokorie. Při plnění úkolů jsem si na papír zaznamenával poznatky z průběhu testování (co uživateli šlo, kde se uživatel naopak pozastavil). Po konci testování jsem každému uživateli položil otázky ohledně právě skončeného testování, které jsou sepsány v tabulce 2. Zajímaly mě především dojmy a pocity z testování a ze samotné aplikace.

Všechny poznatky, připomínky a nápady na vylepšení byly poznamenány, vyhodnoceny a následně zapracovány.

**Tabulka 1.** Úkoly, které byly kladeny uživatelům v rámci testování uživatelského rozhraní.

| # | Úkol |
|---|---|
| 1. | Pořiďte fotografii člověka v mobilní aplikaci. |
| 2. | Vyberte fotografii člověka z galerie zařízení. |
| 3. | Fotografii analyzujte na výskyt leukokorie. |
| 4. | Změňte jazyk v aplikaci na angličtinu. |
| 5. | Nalezněte v aplikaci informace o leukokorii. |
| 6. | Nalezněte o jakou verzi aplikace se jedná. |

**Tabulka 2.** Otázky, které byly kladeny uživatelům po ukončení testování uživatelského rozhraní.

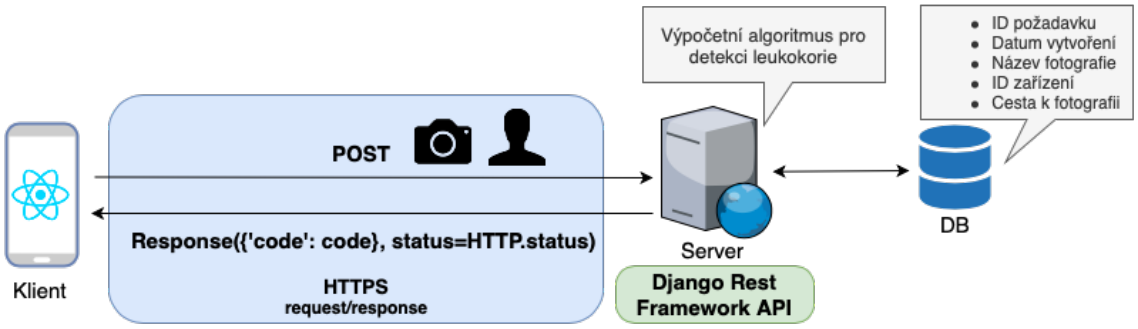
| # | Otázka |
|---|---|
| 1. | Přišla Vám orientace v aplikaci intuitivní? |
| 2. | Dělalo Vám splnění nějakého úkolu problém? |
| 3. | Co se Vám nelíbilo (a proč) a co byste změnil/a? |

## 4.3 Návrh architektury mobilní aplikace

Pro vytvoření algoritmu na detekci leukokorie jsem si vybral programovací jazyk *Python*. Mezi hlavní důvody, proč jsem si vybral právě tento jazyk patří fakt, že se jedná o jeden z nejpopulárnějších jazyků v oblasti zpracování obrazu. Mezi další důležitý faktor patří, že jazyk *Python* obsahuje rozhraní pro velmi populární knihovny pro zpracování obrazu – *OpenCV*[7] a *Dlib*[8]. Mobilní aplikace je implementována ve frameworku *React Native*. Důvody, proč jsem si vybral právě tento framework jsou uvedeny v sekci 5.

Pro komunikaci jsem tedy potřeboval vybrat architekturu, která odděluje implementaci a nabízené služby. Přesně tuto podmínku naprosto splňuje architektura `REST`. Mezi hlavní výhody architektury `REST` patří separace klienta a serveru. Realizace klienta a serveru může být provedena nezávisle, aniž by

[5] https://www.draw.io
[6] https://www.sketchapp.com
[7] https://opencv.org
[8] http://dlib.net




**Obrázek 4.** Architektura mobilní aplikace Eye Check.

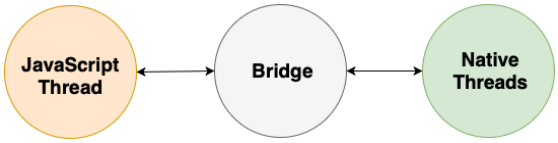
jeden o druhém věděl[9]. To znamená, že jak klient, tak server mohou být implementovány v jiném programovacím jazyku, ale mohou spolu komunikovat přes formát, který oba podporují – např. `JSON`. Právě tohoto jsem využil v mé práci, kdy klient je implementovaný v JavaScriptu, server v Pythonu a přes architekturu `REST` spolu komunikují. Tato komunikaci a celá architektura mobilní aplikace je znázorněna na obrázku 4.

# 5. Implementace mobilní aplikace

## 5.1 React Native

Pro vytvoření mobilní aplikace Eye Check jsem použil framework *React Native*[10]. *React Native* je framework pro vytváření mobilních aplikací za pomocí programovacího jazyka *JavaScript* a javascriptové knihovny *React* udržovaný společností *Facebook*. *React Native* pracuje na velmi podobném principu jako *React* s tím rozdílem, že k zobrazení aplikace nepoužívá *HTML*, ale poskytuje velmi podobné stavební bloky – komponenty [7]. *React Native* používá stejné základní stavební bloky uživatelského rozhraní jako běžné aplikace pro *iOS* a *Android*. Místo použití nativních programovacích jazyků jako *Swift* nebo *Java* lze však tyto stavební bloky dát dohromady pomocí *Javascriptu* a *Reactu*.

Hlavní výhodou *React Native* oproti konkurentním hybridním frameworkům (*Phonegap*[11], *Ionic*[12], *Cordova*[13]) zaměřených na vývoj multiplatformních mobilních aplikací spočívá ve skutečnosti, že *React Native* nepoužívá tzv. *WebView* – mobilní prohlížeč, ale používá tzv. *bridge* [8]. *Bridge* je srdcem celé *React Native* architektury. Jedná se o koncept, který umožňuje obousměrnou a asynchronní komunikaci mezi javascriptovým vláknem a vlákny nativními [8], jak je zobrazeno na obrázku 5. Javascriptová strana posílá asynchronní zprávy ve formátu `JSON` popisující akci, kterou má nativní část vykonat. *Bridge* tedy zastává roli zprostředkovatele zpráv a ovládá asynchronní příkazy mezi javascriptovou a nativní stranou.



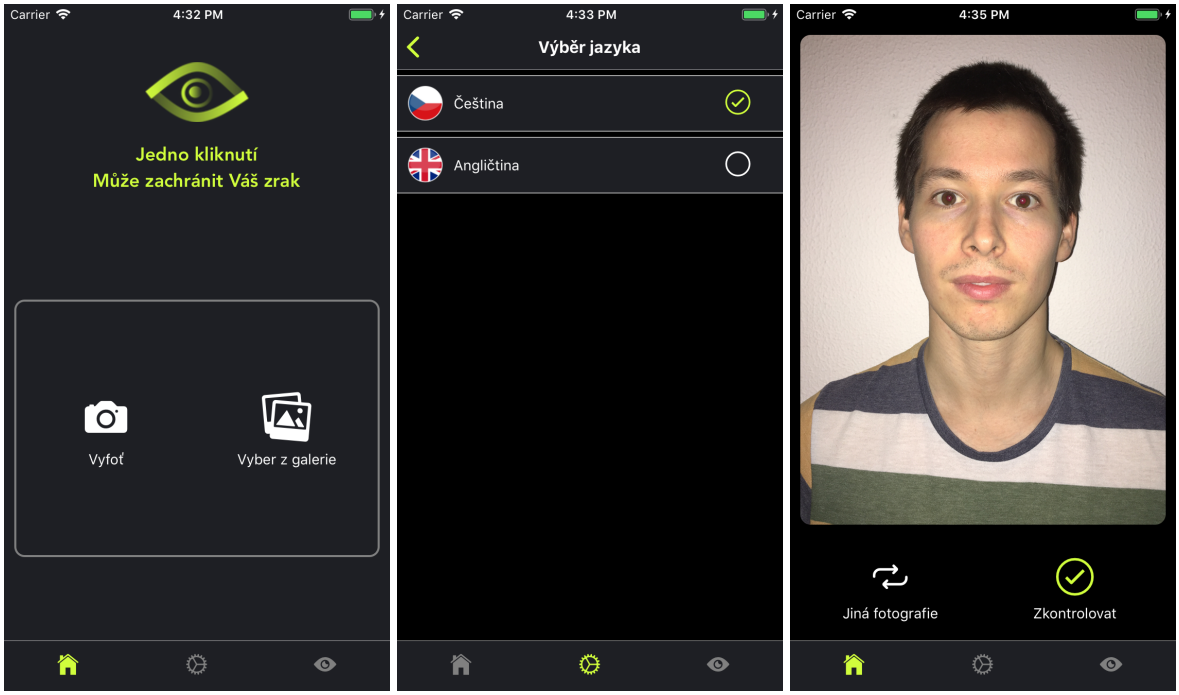

**Obrázek 5.** Koncept architektury *React Native*, kdy *JS* vlákno komunikuje s nativními vlákny pomocí *bridge*.

Mezi další důvody, proč jsem k implementaci použil právě framework *React Native* oproti hybridním frameworkům založených na *Web View* patří jeho výkon, který umožňuje dosáhnout až 60 fps a tzv. *Hot Reload*. Díky funkci *Hot Reload* lze vyvíjet aplikaci velmi rychle, jelikož namísto opětovné kompilace lze aplikaci okamžitě znovu načíst při zachování současného stavu. Funkce *Hot Reload* je velmi užitečná a sám jsem ji mnohokrát využil především při vytváření uživatelského rozhraní, kdy namísto opětovné kompilace lze pozorovat změny v reálném čase a ušetřit tak mnoho času. V neposlední řadě je třeba zmínit, že v případě potřeby optimalizovat část aplikace napsané v *React Native* lze vytvořit nativní kód napsaný v programovacím jazyku *Swift*, *Objective-C* nebo *Java* a tento kód potom využít v samotné aplikaci. Tento přístup je velmi užitečný, když existují platformně specifické komponenty – jako např. kamera a lze tak naplno využívat platformně specifické API. Na obrázku 6 jsou vidět některé obrazovky výsledné aplikace Eye Check.

## 5.2 Klient-Server komunikace

Jakmile uživatel vybere ze svého zařízení fotografii, kterou chce analyzovat na výskyt leukokorie a zmáčkne tlačítko pro odeslání, dojde k poslání požadavku na

[9] https://www.codecademy.com/articles/what-is-rest
[10] https://facebook.github.io/react-native/
[11] https://phonegap.com
[12] https://ionicframework.com
[13] https://cordova.apache.org




**Obrázek 6.** Mobilní aplikace Eye Check.

server. Jak již bylo výše zmíněno, komunikace mezi klientem a serverem probíhá pomocí architektury `REST`.

Klient komunikuje se serverem pomocí metody `POST`. Jelikož klient posílá na server fotografii, je v `HTTP` hlavičce požadavku jako `Content-Type` nastavena hodnota *multipart/form-data*. K vytvoření samotného těla požadavku je použit *FormData* objekt.

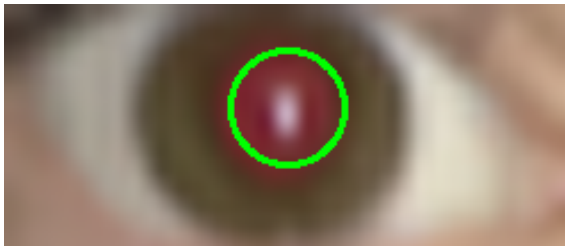
Jakmile přijde požadavek na server, dojde k uložení fotografie uživatele. Každá takto uložená fotografie je následně přidána do datové sady, která se postupně zvětšuje a pomáhá tak k postupnému vylepšování celého algoritmu. Zde lze vidět i jedna z dalších základních myšlenek aplikace. Klient si může otestovat svůj zrak a následně pomůže k vylepšení a přesnější detekci leukokorie pro budoucí uživatele. Server zpracuje požadavek a následně odpovídá klientovi pomocí formátu `JSON` jedním z následujících kódů:

- **NO_FACE_DETECT** – Na obrázku nebyl detekován žádný obličej. Je potřeba nahrát novou fotografii.
- **MORE_FACES_DETECT** – Na obrázku bylo detekováno více obličejů. Je potřeba nahrát novou fotografii.
- **NO_RED_EYES** – Na obrázku nebyl detekován tzv. *efekt červených očí*. Je potřeba nahrát novou fotografii.
- **NO_LECORIA** – Na obrázku nebyla detekována leukokorie.
- **LECORIA** – Na obrázku byla detekována leukokorie.

## 5.3 Algoritmus pro detekci leukokorie

K detekci obličeje a očí jsem použil knihovnu *Dlib*. Pro práci se samotnými fotografiemi jsem potom využil knihovnu *OpenCV*.

Na obrázku 7 lze vidět vidět výřez lidského oka, který je získán právě pomocí knihovny *Dlib* a tzv. *facial landmarks*. *Facial Landmarks* se používají k lokalizaci nejznámějších oblastí obličeje člověka jako jsou oči, obočí, nos, ústa a čelist. Knihovna *Dlib* již obsahuje předtrénovaný detektor pro detekci těchto oblastí ve formě 68 souřadnic [9]. Celý proces nalezení obalového obdélníku (bounding box) lidského oka je tak díky knihovně *Dlib* velmi přesný a snadný.



**Obrázek 7.** Bounding box lidského oka, který je získán z fotografie uživatele. Zelený kruh značí zornici nebo-li prostor, ve kterém se prohledává a porovná každý pixel s RGB prahem.

Po získání obalového obdélníku lidského oka dojde v prostoru zornice, který je na obrázku 7 vyznačen zeleným kruhem, k prohledávání a porovnávání pixelů s RGB prahem. Tento RGB práh reprezentuje různé odstíny červené barvy, která je očekávána u zdravých očí jedince. Hodnota prahu byla nastavena experimentálně podle dostupných testovacích dat.

Po analýze obou očí je podle výsledného skóre určen finální výsledek, který je uživateli zobrazen.

Souběžně s vylepšováním výše zmíněného algoritmu jsem hledal další možná a efektivnější řešení pro detekci leukokorie. Jednou z variant, která by mohla v budoucnu dosáhnout ještě lepších a přesnějších výsledků, je použití konvolučních neuronových sítí (CNN). Pro vytvoření testovací CNN jsem využil knihovnu *Keras*[14] běžící na backendu *TensorFlow*[15]. Pro natrénování CNN a dosahování lepších a přesnějších výsledků je však potřeba mnohem více dat (fotografií), než které mám doposud k dispozici. Pokud by se však v budoucnu podařilo nasbírat velké množství dat, mohlo by použití CNN nahradit stávající řešení, které je popsáno výše.

### 5.4 Vyhodnocení úspěšnosti algoritmu

Jak již bylo zmíněno výše, k detekci obličeje a očí se používá knihovna *Dlib*. K tomu, aby správně detekovala lidské oko, však potřebuje fotografii, která zachycuje celý lidský obličej v tzv. *frontální pozici* – pozice, kde se člověk dívá přímo do objektivu fotoaparátu a nemá obličej natočen do strany.

Další podmínkou, kterou musí fotografie splňovat, aby se zařadila do testovací sady je přítomnost tzv. *efektu červených očí*. Kdyby tato podmínka nebyla požadována, bylo by velmi těžké určit, zda-li se jedná v zornici člověka o leukokorii nebo o pouhý odraz světla způsobený bleskem.

Najít fotografie, na kterých je vidět celý obličej člověka v tzv. *frontální pozici*, a kde se zároveň projeví tzv. *efekt červených očí*, nebylo jednoduché. Najít fotografie, které splňují výše uvedené požadavky, a kde se projevila leukokorie bylo ještě složitější. Z těchto důvodů je testovací sada, na které byl algoritmus testován, velmi malá. Výsledky a vyhodnocení úspěšnosti algoritmu na testovací datové sadě jsou uvedeny v tabulce 3.

**Tabulka 3.** Výsledky a vyhodnocení úspěšnosti algoritmu na testovací datové sadě.

| Třída | # Fotografií | Úspěšnost |
|---|---|---|
| Zdravé oči (obsahující EČO) | 105 | 99,05 % |
| Zdravé oči (bez EČO) | 208 | 97,12 % |
| Leukokorie | 14 | 100,00 % |
| Fotografie bez obličeje | 126 | 99,21 % |
| Fotografie s více obličeji | 108 | 97,22 % |
| Celkem | 561 | 98,52 % |

*Poznámka: EČO = efekt červených očí*

[14] https://keras.io
[15] https://www.tensorflow.org

### 5.5 Dostupnost

Jak již bylo zmíněno, jeden z hlavních cílů této práce je poskytnout mobilní aplikaci co nejširšímu spektru uživatelů. V aplikaci je tedy dostupná multijazyčnost a uživatel si může vybrat mezi češtinou a angličtinou. Do budoucna se plánuje rozšíření i o další jazyky (francouzština, němčina).

Pro operační systém *iOS* je mobilní aplikace dostupná pro verzi *iOS 9.0* a novější. Na zařízeních s operačním systémem *Android* je aplikace dostupná pro verzi *Android 4.1 (Jelly Bean)* a novější.

Mobilní aplikace Eye Check je dostupná pro veřejnost jak v *App Store*, tak na *Google Play*.

## 6. Testování mobilní aplikace

Testování mobilní aplikace Eye Check se uskutečnilo pro obě dostupné platformy – *iOS* a *Android*. Pro operační systém *iOS* byla výsledná aplikace nahrána na *App Store Connect*[16]. Uživatelé následně prováděli testy na svých zařízeních pomocí aplikace *TestFlight*[17]. Pro operační systém *Android* byla aplikace testována pomocí *Google Play Console*[18], kde byla aplikace nahrána do tzv. *alfa kanálu*, který je určen k testování. Testerům byl následně odeslán odkaz k nainstalování aplikace, po jehož rozkliknutí byla aplikace nainstalována do uživatelova mobilního zařízení.

Největší problém, který se při testování aplikace Eye Check objevil, souvisel s pořizováním a výběrem fotografie pro samotnou analýzu. Mnoho uživatelů pořizovalo fotografii přímo detailu oka nebo očí. Tento nalezený problém jsem se rozhodl vyřešit pomocí grafické vizualizace. Konkrétně jsem do obrazovky pro pořizování fotografie přidal geometrický útvar kruh. Uživatel se tedy snaží vmístit svůj obličej do tohoto kruhu, což má za výsledek, že fotografie je správně vyfocena ve smyslu detailu fotografie a připravena k analýze.

## 7. Rozšíření práce

Jako rozšíření této práce byla pro mobilní aplikaci vytvořena i responzivní multijazyčná webová aplikace[19], kterou jsem vytvořil ve frameworku *Nette*[20]. K docílení responzivního webu byl použit framework *Bootstrap*[21]. Webová aplikace obsahuje jak klientskou, tak administrační část.

[16] https://appstoreconnect.apple.com
[17] https://testflight.apple.com
[18] https://developer.android.com/distribute/console
[19] http://eyecheck.cz
[20] https://nette.org
[21] https://getbootstrap.com

Klientská část slouží jako prezentační materiál k mobilní aplikaci. Do budoucna se však plánuje rozšířit její účel o možnost nahrát fotografie přes webový prohlížeč a zvětšovat tak datovou sadu za účelem vylepšení algoritmu pro detekci leukokorie. Dalším možným vylepšením do budoucna je potom možnost detekovat přítomnost leukokorie přímo z webového prohlížeče.

Administrační část slouží především pro správu obsahu webové aplikace. V rozhraní lze přidávat, upravovat a mazat uživatele, kteří se mohou starat o obsah webové aplikace. Lze tady také dynamicky měnit multijazyčný obsah podle potřeby uživatele nebo sledovat statistiky o tom, kolik fotografií již bylo pomocí mobilní aplikace analyzováno.

## 8. Závěr

Cílem této práce bylo navrhnout a implementovat multiplatformní mobilní aplikaci pro detekci leukokorie ze snímku lidského obličeje pro platformy *iOS* a *Android*. Důležitou součástí celé této práce bylo vyhledání odborníka z oboru očního lékařství a následné konzultace ohledně symptomu leukokorie, které vedly ke zdokonalení celého díla.

Před samotným vývojem mobilní aplikace byl proveden návrh a testování uživatelského rozhraní. Byl vytvořen *sketch*, *wireframe* i *mockupy* mobilní aplikace. Testování uživatelského rozhraní proběhlo na mockupech. Testování bylo vyhodnoceno a veškeré změny byly zavedeny do mobilní aplikace. Dále byl vytvořen samotný návrh architektury mobilní aplikace. K implementaci samotné aplikace byl vybrán framework *React Native*.

Výsledkem je tedy multiplatformní multijazyčná mobilní aplikace pro platformy *iOS* a *Android*, která umožňuje uživateli nahrát fotografii a následně ji nechat analyzovat na případný výskyt leukokorie. V případě detekce tohoto symptomu je uživatel mobilní aplikací na tuto skutečnost upozorněn a je mu doporučena návštěva lékaře.

Jako rozšíření této práce byla pro mobilní aplikaci vytvořena i responzivní multijazyčná webová aplikace.

## Poděkování

Rád bych poděkoval svému vedoucímu práce panu profesoru Ing. Adamu Heroutovi, Ph.D. za odborné vedení, čas, ochotu a cenné rady při tvorbě této práce. Dále bych zde chtěl poděkovat MUDr. Barboře Žajdlíkové, za její čas, spolupráci a za poskytnutí odborné konzultace v oblasti očního lékařství, bez které by byla celá tato práce jen těžko realizovatelná. V neposlední řadě bych zde rád poděkoval své rodině a přítelkyni za jejich velkou a nenahraditelnou podporu a všem testerům, kteří se podíleli na testování aplikace a poskytli mi zpětnou vazbu.

## Literatura

[1] Renáta Schoffer, Inka Krejčířová, Rudolf Autrata. Diferenciální diagnostika leukokorie u dětí. online, 2017. https://www.pediatriepropraxi.cz/incpdfs/act-000269-0001_10_001.pdf.

[2] Alistair Scott. Your camera could save a child's life. online. http://alscotts.blogspot.com/2009/10/your-camera-can-save-childs-life-true.html.

[3] Barbora Žajdlíková. Osobní sdělení lékaře (lékaře Fakultní nemocnice Brno – Dětská nemocnice, Černopolní 9, Brno) dne 12. listopadu 2018.

[4] Magda Králová. Lidské oko. online. https://edu.techmania.cz/cs/encyklopedie/fyzika/svetlo/lidske-oko.

[5] Matt Warcholinski. What is the difference between wireframe, mockup and prototype? online. https://brainhub.eu/blog/difference-between-wireframe-mockup-prototype/.

[6] Emily Esposito. Low-fidelity vs. high-fidelity prototyping. online, 2018. https://www.invisionapp.com/inside-design/low-fi-vs-hi-fi-prototyping/.

[7] Alexis Mangin. What are the main differences between reactjs and react-native? online, 2016. https://medium.com/@alexmngn/from-reactjs-to-react-native-what-are-the-main-differences-between-both-d6e8e88ebf24.

[8] Marvin Frachet. Understanding the react native bridge concept. online, 2018. https://hackernoon.com/understanding-react-native-bridge-concept-e9526066ddb8.

[9] Adrian Rosebrock. Facial landmarks with dlib, opencv, and python. online, 2017. https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/.