**ŽILINSKÁ UNIVERZITA V ŽILINE**

FAKULTA RIADENIA A INFORMATIKY

****

BAKALÁRSKA PRÁCA

Študijný program: informatika

Martin Kostelej

**Mobilná aplikácia Nočná lampa**

**Night light mobile application**

Vedúci práce: Ing. Boris Bučko, PhD.

Registračné číslo práce: 1223/2020

Ministerské číslo práce: 28360420211122

Žilina, 2021

**ŽILINSKÁ UNIVERZITA V ŽILINE**

FAKULTA RIADENIA A INFORMATIKY

****

BAKALÁRSKA PRÁCA

Martin Kostelej

**Mobilná aplikácia Nočná lampa**

**Night light mobile application**

Študijný odbor: informatika

Študijný program: informatika

Vedúci práce: Ing. Boris Bučko, PhD.

Stupeň kvalifikácie: bakalár(BC)

Dátum zadania práce: 31.10.2020

Dátum odovzdania práce: 6.5.2021

Žilina, 2021

**Zadanie témy**(sem vložím zadanie)

#### ČESTNÉ VYHLÁSENIE

Čestne prehlasujem, že som svoju prácu vypracoval samostatne s využitím dôsledne citovaných literárnych zdrojov a teoretických poznatkov od domácich, ako aj zahraničných autorov a vlastných skúseností a poznatkov nadobudnutých v priebehu štúdia. Obrázky pri ktorých nie je uvedený citačný zdroj, sú výsledkom môjho vlastného spracovania.

Žilina, Máj 2021

..................................

Meno Priezvisko

#### POĎAKOVANIE

Ďakujem Ing. Borisovi Bučkovi, PhD. za všetky rady, návrhy, postrehy a nápomocné pripomienky poskytované pri vypracovaní bakalárskej práce a za to že ma viedol správnym smerom.

#### Abstrakt

KOSTELEJ, Martin: Mobilná aplikácia Nočná lampa[bakalárska práca]. – Žilinská univerzita. Fakulta riadenia a informatiky; Katedra matematických metód a operačnej analýzy. – Vedúci práce: Ing. Boris Bučko, PhD. Stupeň odbornej kvalifikácie: bakalár. Študijný program: informatika. Žilina. 2021. 57 s.

Cieľom práce bolo implementovať mobilnú aplikáciu „Nočná lampa“, ktorá by sa využívala pre nočné osvetlenie miestnosti, či už pri bežných činnostiach alebo jemné osvetlenie popri spánku. Ako zdroj svetla by sa využíval displej inteligentného mobilného zariadenia. Podľa overenia už existujúcich podobných aplikácií, osobného testovania a preštudovania užívateľských recenzií sme navrhli, ako by mohla výsledná aplikácia vyzerať a aké funkcionality by ponúkala. Následne sme zvolili platformu, na ktorú implementujeme aplikáciu a na základe toho rozhodnutia sme vybrali vhodný programovací jazyk. Pri vývoji aplikácie sme prihliadali na pôsobenie svetla na ľudský organizmus počas spánku za pomoci zistených informácií z viacerých zdrojov. Na záver sme naše riešenie porovnali s ostatnými aplikáciami a našim pôvodným návrhom, plne otestovali celú funkcionalitu a zhodnotili sme že sa nám podarilo správne implementovať požadované funkcie.

**Kľúčové slová:** inteligentný mobilný telefón, iOS, mobilná aplikácia, nočná lampa, Swift

#### Abstract

KOSTELEJ, Martin: Night light mobile application. [bachelor thesis]. – University of Žilina. Faculty of Management Science and Informatics; xxxxxxxx. – Supervisor: Ing. Boris Bučko, PhD. Qualification level: bachelor. Study program: Informatics. Žilina. 2021. xx pages.

//TODO na záver preložiť abstrakt

**Keywords:** iOS, mobile application, night light, smartphone, Swift

Obsah

[Zoznam obrázkov 10](#_Toc65314842)

[Zoznam skratiek 11](#_Toc65314843)

[Úvod 12](#_Toc65314844)

[1 Analýza mobilných aplikácií a vplyvu svetla na organizmus 13](#_Toc65314845)

[1.1 Analýza podobných aplikácií 13](#_Toc65314846)

[1.1.1 Výber distribučných obchodov 13](#_Toc65314847)

[1.1.2 Pozitívne hodnotenia 14](#_Toc65314848)

[1.1.3 Negatívne hodnotenia 14](#_Toc65314849)

[1.1.4 Zhrnutie recenzií 14](#_Toc65314850)

[1.2 Pôsobenie svetla na ľudský organizmus počas spánku 14](#_Toc65314851)

[1.2.1 Vplyv svetla na organizmus 14](#_Toc65314852)

[1.2.2 Prispôsobovanie sa tela na zmeny intenzity svetla počas dňa 15](#_Toc65314853)

[1.2.3 Tvorba melatonínu 15](#_Toc65314854)

[1.2.4 Farba nočnej lampy 16](#_Toc65314855)

[2 Tvorba mobilných aplikácií 17](#_Toc65314856)

[2.1 Mobilné aplikácie a ich popularita 17](#_Toc65314857)

[2.2 Vývoj mobilných aplikácií 18](#_Toc65314858)

[2.2.1 Android 18](#_Toc65314859)

[2.2.2 iOS 20](#_Toc65314860)

[2.3 Swift 24](#_Toc65314861)

[2.3.1 Vznik a všeobecný prehľad 24](#_Toc65314862)

[2.3.2 Kompatibilita s Objective-C 25](#_Toc65314863)

[2.3.3 Bezpečnosť 26](#_Toc65314864)

[2.3.4 Syntax a jednoduché príklady 26](#_Toc65314865)

[2.3.5 Verzie 28](#_Toc65314866)

[2.4 Xcode 29](#_Toc65314867)

[3 Cieľ práce 32](#_Toc65314868)

[4 Návrh a implementácia 34](#_Toc65314869)

[4.1 Návrh funkcií a vzhľadu 34](#_Toc65314870)

[4.1.1 Platforma, programovací jazyk a IDE 34](#_Toc65314871)

[4.1.2 Výber funkcií 34](#_Toc65314872)

[4.1.3 Ovládanie 35](#_Toc65314873)

[4.1.4 Ikona 36](#_Toc65314874)

[4.1.5 Úvodná obrazovka 37](#_Toc65314875)

[4.1.6 Používateľské rozhranie 38](#_Toc65314876)

[4.2 Architektúra a diagram 39](#_Toc65314877)

[4.3 Výsledná implementácia 41](#_Toc65314878)

[4.3.1 Používateľské rozhranie 42](#_Toc65314879)

[4.3.2 Ukladanie používateľských údajov 44](#_Toc65314880)

[4.3.3 Výber farby 46](#_Toc65314881)

[4.3.4 Časovač 47](#_Toc65314882)

[4.3.5 Prechod medzi farbami 48](#_Toc65314883)

[4.3.6 Testovanie 48](#_Toc65314884)

[5 Výsledky práce 50](#_Toc65314885)

[5.1 Ukážka funkčnej aplikácie 50](#_Toc65314886)

[5.2 Porovnanie s existujúcimi aplikáciami 52](#_Toc65314887)

[Záver 54](#_Toc65314888)

[Zoznam použitej literatúry 55](#_Toc65314889)

# Zoznam obrázkov

Obrázok 1 Podiel trhu mobilných operačných systémov

Obrázok 2 Počet aktívnych používateľov inteligentných mobilov

Obrázok 3 iPhone OS 1

Obrázok 4 iOS 14

Obrázok 5 Obľúbenosť jazykov/technológií v roku 2015

Obrázok 6 Obľúbenosť jazykov/technológií v roku 2020

Obrázok 7 Príklad Swift kódu: základy

Obrázok 8 Príklad Swift kódu: cykly

Obrázok 9 Príklad Swift kódu: podmienky

Obrázok 10 Príklad zobrazenia návrhu užívateľského rozhrania v Xcode

Obrázok 11 Ikona aplikácie na domovskej obrazovke

Obrázok 12 Úvodná obrazovka

Obrázok 13 Návrh používateľského rozhrania

Obrázok 14 Vývojový diagram

Obrázok 15 Storyboard návrh pre používateľské rozhranie

Obrázok 16 Ukážka aplikácie: farba a prechod medzi farbami

Obrázok 17 Ukážka aplikácie: orientácia na šírku

Obrázok 18 Paleta farieb ColorPicker

Obrázok 19 Ukážka aplikácie: časovač

# Zoznam skratiek

ABI Application Binary Interface

API Application Programming Interface

BASIC Beginners All-Purpose Symbolic Instruction Code

IDE Integrated Development Environment

iOS iPhone operating system

MVC Model View Controller

SDK Software Development Kit

# Úvod

Mobil používa v dnešnej dobe skoro každý. Človek na ňom trávi značnú časť dňa, pretože množstvo aplikácií ktoré sú dostupné, sú zamerané na viaceré oblasti zo života. Mobil využívame na komunikáciu(textovú, hlasovú alebo aj video hovory) s priateľmi, kolegami z práce, manažment času, písanie poznámok, nastavovanie budíku, zábavu formou hier, sociálnych sietí... Dalo by sa povedať že možnosti sú prakticky neobmedzené. Obmedzuje nás len kreativita a schopnosti vývojárov.

Tému mobilnej aplikácie som si zvolil kvôli tomu, že aktuálne možnosti pre vývoj sú veľmi rozsiahle, dopyt po kvalitných aplikáciách stále narastá a daná oblasť ma zaujíma. Počet potenciálnych užívateľov mobilných aplikácií je enormný, keďže veľké množstvo obyvateľov má vlastný inteligentný telefón, kde si vyberajú aplikácie vhodné pre ich potreby. Konkrétnejšie sa jedná o prácu alebo voľný čas. My sme sa zamerali na voľnočasové aktivity používateľov a rozhodli sme sa vytvoriť vlastnú aplikáciu, ktorá sa môže používať ako nočná lampa. Ako zdroj svetla bude využívať displej a umožní nastavenie osvetlenia v izbe podľa potreby, či už sa jedná o farbu, intenzitu alebo nastavenie času po ktorom sa vypne.

Pre vývoj mobilných aplikácií je veľké množstvo technológií, takže ako vývojári sme mali viacero možností kade sa uberať. Zvolili sme si platformu iOS od firmy Apple, keďže s ich zariadeniami už máme skúsenosti. Na implementáciu sme využívali programovací jazyk Swift.

Pred samotnou implementáciou bol kľúčový vhodný návrh, ako by mala aplikácia vyzerať a fungovať. Z tohoto dôvodu bol dôležitý prieskum trhu s už existujúcimi podobnými aplikáciami, či už na platformu iOS alebo aj na konkurenčné operačné systémy.

Po následnom zistení ako pôsobí svetlo na ľudský organizmus počas spánku, sme mohli prejsť na finalizáciu návrhu a samotnú implementáciu. Počas programovania bolo potrebné aj veľké množstvo testovania a potrebných zmien na základe výsledných dojmov z aplikácie.

# Analýza mobilných aplikácií a vplyvu svetla na organizmus

## Analýza podobných aplikácií

### Výber distribučných obchodov

Prvým krokom pri tvorbe bolo preskúmanie trhu. Preto bolo dôležité vyhľadať už existujúce podobné aplikácie. Keďže sme zameraný všeobecne na vývoj mobilných aplikácií, kontrolovali sme nielen obchody pre iOS platformu ale aj konkurenčné operačné systémy. Nižšie máme uvedený graf, ktorý porovnáva celosvetový podiel trhu mobilných operačných systémov za rok 2020.



Obrázok 1 Podiel trhu mobilných operačných systémov

Zdroj: (1)

Ako vidíme na obrázku vyššie, tak najvyužívanejšie operačné systémy sú Android a iOS, kde má Android jasnú prevahu. Malý podiel na trhu tvoria ešte KaiOS, Samsung a Windows phone. Podobné aplikácie budeme teda hľadať na obchodoch pre tieto platformy.

Po dôslednom prehľadávaní sme zistili, že nočné lampy sa nachádzajú len na obchodoch pre Android(Google Play), iOS(App Store) a Windows phone(Windows phone store). Viacero obľúbených aplikácií malo dokonca multiplatformovú podporu.

### Preskúmané aplikácie

Pri prieskumu trhu sme sa zamerali hlavne na také aplikácie, ktoré majú dostatočný počet používateľských recenzií. Prihliadali sme na spokojnosť užívateľov a následne sme aplikácie vyskúšali na mobilnom zariadení. Testovaná funkčnosť bola z nasledovných aplikácií pre nočné lampy: Best Night Light, Night Light – Reading Lamp, Night Light Lite Nightlight, Mood Light, Night Light – Relax Sleep. Ďalej sme sa inšpirovali aj funkciami aplikácie Yeelight, ktorá síce nevyužíva na osvetlenie displej ale priamo pripojené inteligentné svetlo ale nastavovanie funkcionalít má podobné.

### Pozitívne hodnotenia

Prieskum nám ukázal, že ľudia majú zväčša radi jednoduché a priamočiare ovládanie bez zbytočných funkcií, ktoré nevyužijú. Obľúbené boli najmä aplikácie, ktoré využívali jednofarebné pozadie na celú obrazovku, ktorého farbu si mohol užívateľ jednoducho nastaviť pomocou palety. Často pridávaná funkcionalita bol taktiež časovač, po uplynutí ktorého sa nočná lampa vypla, keďže bola mnohokrát využívaná pri zaspávaní. Niektoré aplikácie ponúkali aj hudbu alebo šumenie na pozadí na lepšie zaspávanie, napríklad pre deti.

### Negatívne hodnotenia

Najčastejšie problémy v recenziách boli ohľadom neprívetivého ovládania, kde sa ľudia sťažovali napríklad na ťažkopádne nastavovanie farby svetla alebo jeho intenzity. Ďalšie negatíva boli vyskytovanie sa reklám(pokiaľ nezaplatili za prémiovú verziu), vysokej spotreby batérie alebo menších chýb.

### Zhrnutie recenzií

Po zhodnotení všetkých pozitívnych a negatívnych hodnotení a vlastných nápadoch a návrhoch sme si mohli zvoliť, aké funkcionality by bolo vhodné zahrnúť do našej implementácie nočnej lampy. Následne sme mohli navrhnúť približný výzor používateľského rozhrania. Bližší pohľad na návrh sa nachádza v kapitole 4.1 Návrh funkcií a vzhľadu.

## Pôsobenie svetla na ľudský organizmus počas spánku

### Vplyv svetla na organizmus

V dnešnej dobe, kedy nie je problém s nedostatkom osvetlenia vďaka umelým svetlám dostupným doma, v robote, obchodoch, či na uliciach vo forme pouličného osvetlenia, môže byť náš pravidelný cyklus striedania dňa a noci rozhodený. Svetlo má počas dňa a hlavne počas spánku na ľudský organizmus veľký vplyv. Ako je spomínané v článku (2) svetlo môže ovplyvňovať ako sme schopní fungovať cez deň po fyzickej ale aj psychickej stránke. Na základe toho môžu mať dopad aj na výkony v zamestnaní alebo v škole.

Faktorov ktoré negatívne pôsobia na synchronizáciu denného rytmu je viacero. Môže ísť o dlhé pozeranie televízie alebo prácu na počítači, zaspávanie pri zapnutej televízii, časté používanie mobilných zariadení pred spánkom alebo silné osvetlenie spálne. Tieto problémy sú spôsobené kvôli takzvanému modrému svetlu v elektronických zariadeniach.

### Prispôsobovanie sa tela na zmeny intenzity svetla počas dňa

Naše telo aj jeho orgány sú prispôsobené tomu, aby sa počas dňa menili. Tento efekt je nazývaný cirkadiánny rytmus. Zo všeobecného hľadiska ide o rytmus, počas ktorého telo rozpoznáva kedy má byť aktívne a naopak kedy je čas na odpočinok. Podľa článku (3) pri bližšom pozorovaní tela zistíme, že „prebiehajú aj zmeny telesnej teploty, tlaku, srdečnej frekvencie, menia sa aktivity tráviaceho traktu, vylučovania, hladiny hormónov v tele atď.“.

Podľa Maierovej (3) vplýva svetlo na organizmus troma spôsobmi:

* Keď na telo pôsobí intenzívne svetlo, tak sa znižuje ospalosť a navodzuje dobrú náladu. Pomocou intenzívneho svetla sa taktiež stabilizuje cirkadiánny rytmus.
* Zmena intenzity svetla počas svitania alebo súmraku vedie k tomu, že sa naše telo snaží prispôsobiť danej zmene na základe predlžovania alebo skracovania dňa. Intenzívne svetlo večer spôsobuje, že naše telo bude pripravené na spánok neskôr. Naopak svetlo ráno podporuje prirodzené vstávanie.
* Pokiaľ svetlo zasahuje do fázy kedy chceme spať, tak negatívne ovplyvňuje regeneráciu orgánov, rast a imunitný systém. Svetlo počas spánku naše telo vníma tak, že je už deň a nastávajú určité procesy, ktoré zabezpečujú dennú aktivitu. Z toho dôvodu sa počas spánku nemusí naše telo zregenerovať a nebude dostatočne oddýchnuté.

Z týchto troch zistení vyplýva, že by sme mali pristupovať k osvetleniu miestnosti počas noci veľmi zodpovedne. Pokiaľ sa cítime po spánku stále unavení, mali by sme uvažovať o obmedzení používania svetla pred večerom, aby si naše telo zvyklo na to, kedy si približne chceme ísť ľahnúť(t. j. stabilizujeme náš cirkadiánny rytmus). Ak máme určité osvetlenia v noci v spálni aktívne, tak by bolo potrebné pouvažovať nad znížením jeho intenzity.

### Tvorba melatonínu

Čo vlastne spôsobuje zmena dĺžky osvetlenia cez deň? Ako sa spomína v článku (4) „v priebehu jednotlivých cyklov sa pod vplyvom zmeny dĺžky osvetlenia mení hladina rôznych hormónov(napríklad serotonínu, melatonínu a kortizolu). Tvorba hormónu melatonín je dôležitá aj z pohľadu správneho fungovania cirkadiánneho rytmu.“. Konkrétne je za riadením tvorby melatonínu podľa Chochlíkovej (5) druh fotoreceptoru v oku, ktorý reaguje na modré svetlo. Hladina melatonínu v krvi sa najviac zvyšuje po zotmení. V práci sa ďalej uvádza, že dostatok melatonínu spôsobuje skvalitnenie spánku, dobré sny, ovplyvňuje priebeh starnutia, vystupovanie človeka, má vplyv taktiež na krvný tlak, produkciu pohlavných hormónov a imunitu človeka. Na opačnú stránku ak má človek nedostatok melatonínu, môžu nastať problémy so spánkom a v článku (3) sa dokonca uvádza že „môže dochádzať k oslabeniu imunitného systému a schopnosti regenerácie organizmu. Dlhodobé zníženie môže ohroziť zdravie človeka“. Nedostatok tohto hormónu môže byť spôsobený negatívnymi faktormi, ktoré boli spomínané vyššie v kapitole 1.2.1Vplyv svetla na organizmus. Pokiaľ je ale zažatá nočná lampa počas spánku našou prioritou, alebo ak sa potrebujeme v noci kvôli niečomu zobudiť, je podľa článku (5) vhodnejšie používať svetlo oranžovej alebo červenej farby.

### Farba nočnej lampy

Farba lampy hrá pri výbere vhodného osvetlenia veľkú rolu. Potrebné je odlišovať, na akú časť dňa je svetlo vyžadované. Ako sa spomína v článku (6), tak najhoršie čo môžeme urobiť uprostred noci, keď sa zobudíme a potrebujeme ísť do kúpeľne, je zažať veľké ostré biele svetlo. Také svetlo môže spôsobovať problémy s opätovným zaspaním alebo znížením kvality spánku. V článku sa ďalej hovorí, že pokiaľ chceme v noci niekam ísť, tak je najvhodnejšie mať po ruke svetlo v červenom, oranžovom alebo žltom odtieni. Ďalším riešením je umiestniť nočné lampičky v spomínaných farbách na chodbu, poprípade do kúpeľne. Farby s vysokou zložkou červenej farby majú frekvenciu, ktorá až tak neovplyvňuje náš mozog. Ak sa naopak chceme prebrať, tak je najvhodnejšie svetlo s modrými alebo zelenými odtieňmi, keďže mozgu navodzuje pocit denného osvetlenia.

# Tvorba mobilných aplikácií

## Mobilné aplikácie a ich popularita

Vývoju mobilných aplikácií je dnes zameraná veľká pozornosť. Dôvodom je obrovské množstvo používateľov inteligentných mobilov, z čoho vyplýva aj veľký potenciálny počet používateľov vytvorenej aplikácie. V grafe nižšie je zobrazený počet aktívnych používateľov inteligentných mobilov v rokoch 2017 až 2019. Informácie o počte sú získané

z článku (7).



Obrázok 2 Počet aktívnych používateľov inteligentných mobilov

V grafe vyššie vidíme, že každý rok počet používateľov rástol a v roku 2019 dosiahol až 3.2 miliardy. Keď toto číslo porovnáme s celkovým počtom obyvateľov pre rok 2019, ktorý bol podľa stránky (8) približne 7.71 miliardy, tak zistíme, že až 41.5 % obyvateľstva používa inteligentný mobil. Dopyt po aplikáciách je teda veľký ale zároveň s ním rastie aj ponuka. V dnešnej dobe je počet vytváraných mobilných aplikácií veľmi vysoký a z toho dôvodu vládne veľká konkurencia v kvalite dostupných služieb. Pokiaľ chceme vytvoriť úspešnú a obľúbenú aplikáciu, musíme byť v niečom lepší ako ostatní. Najčastejšie sa jedná o kvalitu implementácie, príjemný dizajn alebo inovatívne nápady. Podľa internetového článku (9) bolo v mesiaci jún 2020 na Google Play Store(distribučný Android obchod) dostupných 2.96 miliónov aplikácií. Pre Apple App Store (distribučný iOS obchod) bol v mesiaci júl 2020 počet dokonca až 4.37 milióna. Pri takom množstve aplikácií je potrebné pri vytváraní vlastného nápadu najskôr overiť, či už niekto niečo podobné vymyslel a ak áno, tak musíme byť v niečom inovatívny aby sme si získali zákazníkov.

Dôležitosť a obľúbenosť mobilných aplikácií je vidno aj z hľadiska veľkej prevahy v čase používania, oproti času strávenému prehliadaním webových stránok. Podľa prieskumu v článku (10) bol v roku 2016 priemerný čas strávený používaním internetového prehliadača na mobilnom zariadení 26 minút za deň. V porovnaní s tým bol priemerný čas využívania mobilných aplikácií až 2 hodiny a 11 minút. Z toho vyplýva, že približne 83 percent času využívania mobilného zariadenia sú používané aplikácie. Z tohoto dôvodu si aj viacero úspešných firiem vytvorí okrem ich vlastnej internetovej stránky aj mobilnú aplikáciu. Časté dôvody uprednostňovania verzie mobilnej aplikácie nejakej služby oproti jej webovej stránke sú rýchlejšia odozva, krajšie používateľské rozhranie alebo jednoduchšie ovládanie.

## Vývoj mobilných aplikácií

Pokiaľ sa chceme dať na vývoj mobilných aplikácií, prvá otázka zrejme bude, na aký operačný systém by sme chceli našu aplikáciu vytvoriť. Pokiaľ sa zameriavame iba na jednu platformu, tak vyberáme z nástrojov ktoré podporujú vývoj pre zvolený operačný systém. Najčastejšie využívaná mobilná platforma je Android a druhé miesto patrí iOS ako bolo spomínané v kapitole 1.1.1Výber distribučných obchodov . Preto sa bližšie zameriame na možnosti vo vývoji pre tieto dva operačné systémy. Ak potrebujeme vytvoriť aplikáciu na viacero platforiem, tak sú dostupné rozličné nástroje, ktoré podporujú tento vývoj, napríklad Xamarin, React Native, Flutter...

### Android

Operačný systém Android je veľmi obľúbený hlavne kvôli veľkému množstvu podporovaných značiek mobilných telefónov. Používatelia majú z toho dôvodu veľa možností na výber. Môžu sa rozhodovať na základe ceny, veľkosti a kvality displeja alebo výkonu zariadenia. Cenové zastúpenie zariadení sú od úplne najlacnejších, kde si môžeme jedno zariadenie kúpiť kľudne do 100 EUR, až po najdrahšie, kde cena presahuje aj 1000 EUR.

Kľúčové rozhodnutie na začiatku vývoja je výber programovacieho jazyka. Na tvorbu aplikácií pre platformu Android je ich dostupný hojný počet. Výber jazyka prispôsobujeme našim schopnostiam a typu aplikácie, ktorú chceme vytvoriť. Medzi najobľúbenejšie jazyky patria:

* **Java:** zrejme najobľúbenejší pre tvorbu Android aplikácií. Spolu s Kotlinom patrí medzi oficiálne jazyky pre Android vývoj. Tieto jazyky majú najväčšiu podporu od Google. Java bola vytvorená v roku 1995. (11)
* **Kotlin:** ďalší oficiálny podporovaný jazyk pre Android vývoj. Kotlin aj Java sú veľmi podobné a obe bežia na Java Virtual Machine. Častokrát sa stáva, že programátori preferujú viac Kotlin oproti Jave, pretože je prístupnejší pre začiatočníkov a kód je prehľadnejší. Ak už máme určité základy z Javy, tak prechod na Kotlin by nám nemal robiť problém.(11)
* **BASIC:** vhodný programovací jazyk pre začiatočníkov, kvôli jeho jednoduchosti. BASIC síce nie je oficiálne podporovaný jazyk na vývoj Android aplikácií ale dajú sa s ním vytvoriť menšie a jednoduchšie programy vhodné na začiatky. Pokiaľ ale chceme vyvíjať rozsiahlejšie a zložitejšie programy, tak by sme mali využiť inú alternatívu.(12)
* **C++:** tento jazyk je vhodný pre skúsenejších programátorov. Jeho veľká výhoda je, že ponúka značnú kontrolu nad prácou v aplikácií, napríklad pri práci s alokáciou pamäte. Vhodný na komplikovanejšie aplikácie, napríklad 3D hry.(11)
* **C#:** pokiaľ potrebujeme výkon C++ ale nechceme taký komplexný jazyk, tak by sme mali siahnuť po C#. Tento jazyk kombinuje vlastnosti Javy a C++ a je skvelou možnosťou pre vývoj Android aplikácií. C# patrí k novším jazykom, keďže bol vyvinutý v roku 2000 spoločnosťou Microsoft. Je vhodný aj na vývoj zložitejších aplikácií alebo hier.(12)
* **LUA:** open-source programovací jazyk používaný na tvorbu menších a jednoduchých hier. Použil sa napríklad na vývoj veľmi známej mobilnej hry „Angry Birds“. Keďže LUA nie je oficiálne podporovaný jazyk na Android vývoj, tak využíva Android SDK, ktorý je súčasťou Corona Game Engine.(12)
* **PhoneGap:** používa rovnaký kód, aký sa bežne využíva na tvorbu webových stránok(HTML, CSS, JavaSript). Aplikácia je v mobile zobrazená pomocou widgetu, ktorý sa používa na zobrazovanie webových stránok vnútri aplikácií.(11)

Po zvolení vhodného programovacieho jazyka pre našu budúcu aplikáciu postupujeme k výberu IDE v ktorom budeme písať kód. Na výber je veľké množstvo IDE a každý si môže vybrať podľa jeho preferencií. Na vývoj Android aplikácií je najčastejšie využívané Android Studio, ktoré má oficiálnu podporu od Google(podpora pre programovací jazyk Java alebo Kotlin ale umožňuje aj vývoj v C alebo C++, kde využíva Android Native Development Kit). Ďalšie obľúbené alternatívy sú IntelliJ IDEA, Microsoft Visual Studio, Eclipse...

Počas vývoja je veľa krát potrebné vidieť, ako naša aplikácia vyzerá na niektorých zariadeniach, aby sme mohli prispôsobiť vzhľad na rôzne veľkosti displeja a doladili chyby pri zobrazovaní. Jedna možnosť je používať priamo fyzické zariadenia, cez ktoré budeme aplikáciu spúšťať. Pri tejto variante testovania prichádzame ku problému, pri ktorom by sme potrebovali veľa typov mobilných zariadení pre správne otestovanie. Druhá možnosť je využívanie emulátorov. Z jednej stránky je to jednoduchšie, lebo emulátory podporujú veľa typov mobilných zariadení ale z druhej stránky narážame na problém, že nie všetko sa dá otestovať na simulovanom systéme. Často sa pristupuje ku kombinácii obidvoch možností, kde sa niektoré časti testujú na emulátore a iné sa skúšajú už konkrétne na fyzických zariadeniach. Viaceré IDE ponúkajú vstavané emulátory alebo umožňujú ich dodatočnú inštaláciu ako napríklad Android Studio, ktoré ma na výber širokú škálu značiek a typov Androidových mobilných zariadení.

Pokiaľ chceme naše vytvorené aplikácie aj publikovať cez distribučný obchod Google Play Store, budeme si musieť založiť vývojársky účet. Pre publikáciu našich výtvorov je nevyhnutný jednorazový poplatok 25 dolárov. Po vytvorení účtu a zaplatení poplatku nie sme obmedzení počtom publikovaných aplikácií. Za publikovanie aplikácie, ktorá je zadarmo sa neúčtujú žiadne ďalšie poplatky, avšak ak chceme do obchodu umiestniť platenú aplikáciu, tak si Google berie 30 percentnú províziu zo ziskov. (13)

### iOS

Operačný systém iOS od spoločnosti Apple sa teší veľkej obľube kvôli jeho svižnosti, jednoduchosti, spoľahlivosti a skvelým prepojením systémov Apple zariadení(iPhone, iPad, MacBook, Apple Watch...). Tento operačný systém je exkluzívny pre Apple mobilné zariadenia, tzv. iPhone. Systém iOS síce neponúka takú širokú škálu podporovaných zariadení ako Android, pretože systém Android nie je exkluzívny len pre jednu značku mobilných zariadení, ale vynahrádza to jeho nespornou kvalitou. Ceny za iPhony nie sú také variabilné ako Android zariadenia a bývajú väčšinou dosť vysoké. Je to spôsobené tým, že novo vydané modely iPhone majú vždy konkurovať najdrahším mobilom od konkurenčných značiek. Momentálne sa posledné modely vydané minulý rok pohybujú na Slovenskom trhu od priblžne 450 EUR(iPhone SE 2020, 64GB) až po najvýkonnejší model za 1600 EUR(iPhone 12 Pro Max, 512GB).

Prvá verzia operačného systému iOS bola predstavená v roku 2007 spolu s prvým modelom iPhone. Pri vydaní prvého iPhonu nebol jeho operačný systém ešte pomenovaný. Neskôr mu dali meno iPhone OS a po nejakom čase názov skrátili na iOS. V takej podobe ho poznáme dodnes. (14) Nižšie na obrázku vidíme ako systém iOS 1 vyzeral.

Obrázok, na ktorom je elektronika

Automaticky generovaný popis

Obrázok 3 iPhone OS 1

Zdroj: (15)

Časom sa spolu s ďalšími modelmi iPhonov vyvíjali a zlepšovali aj nové verzie systému iOS. Postupom rokov sa prepracovali cez veľa verzií a minulý rok vyšla už 14 verzia pre tento obľúbený mobilný operačný systém. Konkrétne bola vydaná 17. septembra 2020. (16) Na obrázku nižšie môžeme vidieť najnovší systém iOS 14.



Obrázok 4 iOS 14

Zdroj: (17)

Pokiaľ chceme začať vytvárať naše vlastné iOS aplikácie, budeme si musieť zvoliť programovací jazyk pre implementáciu a vhodný IDE v ktorom budeme písať kód, testovať a kde bude štruktúra našej aplikácie. Ďalej sme na rozdiel od tvorby Android aplikácií limitovaný aj výberom počítaču(konkrétne jeho operačným systémom). Pre vývoj iOS aplikácií je potrebný počítač s operačným systémom macOS. Pokiaľ ale nemáme možnosť zohnať počítač s týmto operačným systémom(častý problém je všeobecne vysoká cena Apple hardvéru), tak máme viacero možností. Jedným z riešení je využitie „cross-platformových“ nástrojov pre vývoj. Druhá alternatíva je využitie cloudových služieb na ktorých môžeme spúšťať aplikácie určené pre macOS. Pri tejto možnosti budeme musieť platiť mesačne zvolenej službe. Takúto ponuku môžeme nájsť napríklad na macincloud.com kde je možnosť platiť mesačne 20 dolárov alebo od počtu hodín kde je cena 1 dolár za hodinu. Ďalšia možnosť je si požičať od niekoho počítač s operačným systémom macOS. (18)

Pre vývoj aplikácií na operačný systém iOS sa vo veľkej väčšine prípadov využívajú len dva hlavné programovacie jazyky, Swift alebo Objective-C. Samozrejme sú na výber aj viaceré jazyky ale pri nich bude potrebné riešiť rozličné komplikácie. Jazyky Swift a Objective-C majú oficiálnu podporu.

* **Swift:** predstavený v roku 2014, takže patrí medzi nové a moderné programovacie jazyky. Každým rokom sa jeho obľúbenosť stále zvyšovala. V roku 2018 získal veľkú popularitu a stal sa hlavným programovacím jazykom pre vývoj Apple aplikácií(nielen pre systém iOS ale aj macOS, iPadOS, watchOS a tvOS). Medzi jeho výhody patrí hlavne jednoduchá syntax, obzvlášť keď sa porovnáva s Objective-C, prehľadný kód a podpora pre viaceré zariadenia. Hlavné nevýhody sú že nepodporuje staré verzie systému iOS, horšia možnosť práce s nástrojmi tretích strán, krátky čas pre získanie dôvery od vývojárov, keďže je na trhu stále pomerne krátko. Bližšie je programovací jazyk Swift popísaný v ďalšej kapitole 2.3Swift. (19)
* **Objective-C:** vytvorený v roku 1984. Predtým ako bol predstavený spoločnosťou Apple programovací jazyk Swift, bol Objective-C považovaný ako hlavný jazyk pre vývoj aplikácií pre iOS. Objective-C je založený na jazyku C, čiže má aj podobnú syntax. Ako už názov napovedá, tak je na rozdiel od klasického C objektovo orientovaný. Jedna z jeho silných stránok je kompatibilita s C a C++ knižnicami. Po príchode jazyka Swift sa dostáva Objective-C do úzadia a Swift je čím ďalej, tým viac preferovanou voľbou pre vývoj aplikácií pre systém iOS. (19)

Keď už máme zvolený programovací jazyk pre našu aplikáciu, tak vyberáme najvhodnejší IDE podľa našich potrieb. Na výber je viacero typov ale tak isto ako aj pri vývoji na Android(preferované Android Studio), majú niektoré väčšiu podporu od Apple. Najčastejšie využívané IDE sú:

* **Xcode:** vytvorený spoločnosťou Apple v roku 2003 pre operačný systém macOS. Momentálne je najčastejšou voľbou vývojárov pri vytváraní aplikácií pre Apple zariadenia(iOS, macOS, iPadOS, watchOS a tvOS). Aktuálna stabilná verzia je 12.4. Obľúbenosť si získava kvôli jeho prehľadnému rozhraniu, kvalitnému a rýchlemu inteligentnému dopĺňaniu kódu, možnostiam debugovania a testovania... Ďalej sa Xcode pýši emulátorom konkrétnych Apple zariadení na ktorých chceme aplikácie testovať. Samozrejme je aj možnosť spustenia aplikácie na vlastnom fyzickom zariadení po pripojení k počítaču. Xcode ponúka aj vytváranie základov používateľského rozhrania aplikácie. Funguje na princípe jednoduchého potiahnutia elementu do aplikácie, napríklad tlačidlá, tabuľky, popisy... Tieto elementy si môžeme následne prispôsobiť podľa seba pomocou kódu. Bližšie je Xcode popísaný v kapitole 2.4Xcode. (20)
* **CodeRunner:** navrhnutý tak, aby podporoval najčastejšie využívané programovacie jazyky. Má podporu až pre 25 jazykov, napríklad C++, C#, Swift, Objective-C, Python, Java... Pre veľa z nich ponúka kompletné inteligentné dopĺňanie kódu a taktiež debugovanie pomocou breakpointov. (21)
* **AppCode:** vytvorený spoločnosťou jetbrains, ktorý je známy viacerými IDE (CLion, PhpStorm, IntelliJ IDEA, PyCharm...). Podporuje viacero programovacích jazykov, napríklad Swift, Objective-C, C/C++... AppCode ponúka efektívnu a prehľadnú navigáciu v štruktúre projektu, inteligentné dopĺňanie kódu a tiež kvalitnú analýzu kódu kde nás IDE priamo upozorní pomocou chyby alebo varovania, kde nám navrhne riešenie problému. (22)

Pre publikáciu našich vytvorených aplikácií na distribučný obchod App Store od spoločnosti Apple si budeme potrebovať vytvoriť vývojársky účet. Na rozdiel od vývojárskeho účtu pre Android nebude platba jednorazová ale budeme si musieť za konto platiť každý rok. Takýto účet môžeme mať za cenu 99 dolárov za jeden rok.(13) Všetky aplikácie, ktoré chceme publikovať na App Store sú pred uverejnením ešte poriadne pretestované Apple testermi, ktorí overujú, či je aplikácia bezpečná a vhodná pre zverejnenie. Pri vlastnom testovaní sme limitovaní na 100 ľudí pre inštalovanie konkrétnej aplikácie na fyzické zariadenie. Po aktivácií vývojárskeho účtu máme viaceré benefity. Jeden z nich je už spomenuté testovanie našej aplikácie Apple testermi, ktorí nás upozornia na chyby. Ďalšia výhoda je prístup k beta verziám operačného systému, aby boli naše aplikácie pripravené na plánované aktualizácie. Ďalej máme prístup k viacerým nástrojom, ktoré nám pomôžu pri vývoji aplikácií. (23)

Ak publikujeme platenú aplikáciu, tak našu províziu tvorí 70 percent zo ziskov. Ak sme súčasťou „Small business“ programu tak nám ide z predaja až 85 percent. (23)

## Swift

### Vznik a všeobecný prehľad

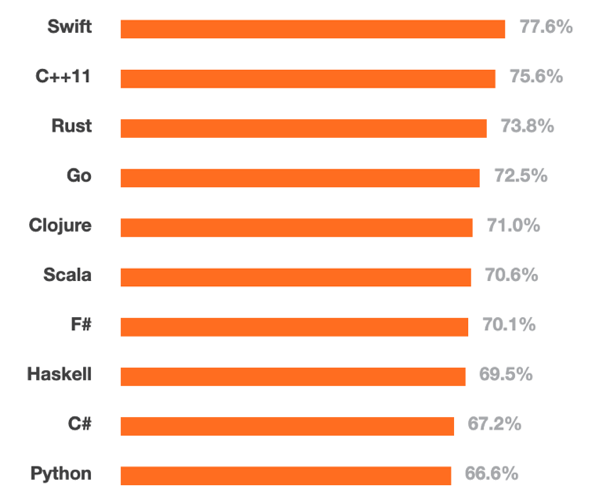
Vývoj nového programovacieho jazyku Swift začal v roku 2010 v spoločnosti Apple. Cieľom bolo vytvoriť vlastný jazyk, ktorý by mohol konkurovať Objective-C vo vývoji aplikácií na iOS. Hlavná myšlienka bola, aby bol výkonný a jednoduchý na naučenie. Prvá verzia Swiftu bola predstavená v roku 2014. (24) Bola súčasťou IDE Xcode verzie 6. Po roku sa Apple rozhodlo prejsť so Swiftom na open-source a kód zverjnilo 3. decembra 2015. (18)

Po predstavení jeho popularita stále rástla, až sa napokon stal najrýchlejšie rastúcim jazykom v histórii. Podarilo sa vytvoriť jazyk, ktorý podporuje tvorbu vlastnej aplikácie veľmi jednoducho, intuitívne a rýchlo. Tak isto je veľmi prívetivý k začiatočníkom a je vhodný ako prvý programovací jazyk. (25)

Ďalšou výhodou Swiftu je syntax, ktorá je stručná a zároveň jednoduchá na pochopenie. Dokonca nemá ani povinnú bodkočiarku za príkazmi. Pamäť je spravovaná automaticky pomocou deterministického počítania referencií. Vďaka tomu má minimálne využitie pamäte a nie je potrebná správa „garbage“ kolekcie. Swift disponuje veľkou svižnosťou naprogramovaných aplikácií a taktiež vysokou bezpečnosťou. (26) V článku (24) sa hovorí že je 2.6 krát rýchlejší ako Objective-C a až 8.4 krát rýchlejší ako Python.

Apple pridal do Swiftu zaujímavé prostredie „playground“, kde môžeme experimentovať s kódom a zisťovať ako niektoré časti fungujú. Toto je veľmi vhodné pre užívateľov, ktorí s jazykom Swift začínajú a potrebujú ho najprv poriadne pochopiť bez toho, aby mali znalosti o všetkých častiach projektu reálnej aplikácie. (18)

Podľa každoročných prieskumov známej stránky stackoverflow sa zistilo, že v roku 2015 bol Swift najobľúbenejším programovacím jazykom. V prieskume sa pýtali vývojárov, s akými programovacími jazykmi alebo technológiami pracovali v posledných rokoch a chceli by sa im venovať aj v budúcnosti. Až 77.6 percent spomenulo Swift. (27) Pre porovnanie je minuloročný prieskum kde sa Swift umiestnil na 10. mieste. Tento jazyk bol v roku 2020 zvolený 59.5 percentami opýtaných vývojárov. Z čoho vyplýva že si udržuje stálu popularitu medzi vývojármi. (28) Výsledky obidvoch rokov z prieskumov sú zobrazené na obrázkoch nižšie.



Obrázok 5 Obľúbenosť jazykov/technológií v roku 2015

Zdroj: (27)



Obrázok 6 Obľúbenosť jazykov/technológií v roku 2020

Zdroj: (28)

### Kompatibilita s Objective-C

Apple síce neoznámilo ukončenie podpory Objective-C ale jazyk Swift ho rýchlo predbieha v popularite. Keďže bol Objective-C predchodcom vo vývoji mobilných aplikácií, tak je priamo vo vývojovom prostredí Xcode zabudovaná jeho spätná kompatibilita. To znamená, že v projektoch vytváraných v jazyku Swift je možné používať knižnice z Objective-C. Pokiaľ chceme využívať tieto funkcie vo Swift projektoch, je potrebné definovať, že sa využíva knižnica, ktorá je z Objective-C. Kompatibilita dokonca umožňuje aby boli niektoré časti kódu písané v jednom jazyku a ostatné časti v druhom. (18)

### Bezpečnosť

Swift veľmi dbá na bezpečnostnú stránku pri programovaní. Snaží sa vyhýbať všetkým častiam, kde by mohla vzniknúť chyba, napríklad musia byť všetky premenné pred použitím inicializované, pri poliach a integeroch kontroluje pretečenie a pamäť je spravovaná automaticky. (26)

Ďalším bezpečnostným prvkom je, že defaultne nemôžu žiadne Swift objekty nadobúdať hodnotu nil(označenie pre premennú, ktorá nemá priradenú hodnotu; podobný variant z iných jazykov je známy ako null). Kompilátor jazyku Swift zistí pokiaľ chceme použiť nil objekt a zabráni tomu pomocou chyby pri kompilovaní. Týmto sa predchádza padnutiam aplikácie počas behu a kód je oveľa bezpečnejší. Avšak nastávajú situácie kedy je nil hodnota očakávaná a v poriadku. Je to možné vtedy, keď si explicitne označíme že daná premenná môže nadobúdať hodnotu nil a ako programátor s tým počítame. Robí sa to pomocou otázniku pri premennej. Pokiaľ povolíme niekde hodnotu nil, tak musíme pri práci s danou premennou počítať aj s touto hodnotou a vyhnúť sa chybám počas behu pomocou overovania cez podmienky. (26)

### Syntax a jednoduché príklady

Ako sa už spomínalo v predchádzajúcej kapitole, tak syntax jazyku Swift je veľmi jednoduchá na pochopenie a pokiaľ už máme nejaké skúsenosti s programovaním tak aj intuitívna. V tejto kapitole si vysvetlíme veľmi stručne základnú syntax jazyku Swift.

Pri definovaní sa používa kľúčové slovo *var* pre premenné a *let* pre konštanty. Dátové typy sa nemusia(môžu pokiaľ to je potrebné) explicitne definovať. Typ bude zvolený automaticky na základe priradenej hodnoty. Výpis na konzolu sa robí pomocou funkcie print(). Pre importovanie knižníc(napríklad z Objective-C) sa využíva kľúčové slovo *import* za ktoré sa píše názov potrebnej knižnice. Komentáre sa vyznačujú pomocou *//Moj komentar* pre jednoriadkové a */\* Moj komentar \*/* pre možnosť mať viacriadkový komentár. Bodkočiarky sú na konci príkazu nepovinné. Musia sa uvádzať len za podmienky, že chceme uviezť viac príkazov v jednom riadku za sebou. (18)

Pokiaľ chceme medzi sebou sčítavať čísla ktoré nie sú rovnakého typu, napríklad integer a double, tak je potrebné vykonať typovú konverziu. Konverzia sa vykonáva pomocou názvu typu na ktorý chceme zmeniť premennú a do zátvoriek dáme hodnotu, napríklad *Double(premenna).*

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 7 Príklad Swift kódu: základy

Na vetvenie programu sa využívajú podmienky a cykly. Syntax Swiftu je podobná ako väčšina ostatných programovacích jazykov. Cykly sú dvoch typov a to *while* a *for*. *While* sa vykonáva pokiaľ platí daná podmienka. Pokiaľ chceme aby sa cyklus vykonal minimálne jedenkrát tak požijeme *repeat while*. Druhý typ cyklu je *for*. V ňom sa používa syntax *for i in 0...4* alebo *for prvok in pole* keď chceme prejsť cez všetky prvky poľa. (18) Jednoduché príklady cyklov sú na príklade nižšie.

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 8 Príklad Swift kódu: cykly

Podmienky sú v základe dvoch typov a to *if-else* a *switch*. V podmienke typu *if* sa overuje či platí daná podmienka. Ak platí tak sa kód vykonáva. Za *if* podmienku môžeme pridať aj časť *else*, ktorá bude vykonaná ak podmienka nie je splnená. Vetvenie pomocou príkazu *switch* sa vykonáva tak, že sa za *switch* dá názov premennej a následne sa vyberie *case* vetva na základe nadobúdanej hodnoty. Na rozdiel od niektorých iných jazykov nie je potrebné zadávať za koncom *case* vetvy príkaz *break,* pretože po vykonaní vetvy automaticky vyskakuje preč zo *switch* bloku. (18) Nižšie sú zobrazené jednoduché príklady pre podmienky.

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 9 Príklad Swift kódu: podmienky

Funkcie sú vo Swifte deklarované pomocou kľúčového slova *func* za ktoré sa píše názov funkcie. Do zátvorky sa následne vložia parametre oddeľované čiarkou. Pokiaľ má funkcia vracať hodnotu, tak sa za zátvorky zadáva operátor *->* spolu s typom aký bude funkcia vracať. (18)

Príklad: *func dajVek(meno: String) -> Int*

### Verzie

* **1.0:** prvá verzia Swiftu vychádza v roku 2014 po štvorročnom vývoji, ktorý začal v strede roku 2010. 17.júla 2014 sa uvádza prvý commit do Swift Github repozitára. 2. júna bol jazyk oznámený a vývojári dostávali predbežnú verziu Swiftu. Oficiálne vychádza Swift 15. septembra na vývojovom prostredí Xcode 6. (24)
* **2.0:** verzia 2.0 bola oficiálne vydaná 15. septembra 2015. Kľúčová novinka bol prechod na open-source, kedy zverejnili git repozitár pre každého. Ďalej bol predstavený nový spôsob spracovávania chýb, kde využívajú príkazy *try, throw* a *catch* známe aj z iných jazykov. Novinkou je aj zameranie sa na využívanie starších verzií operačných systémov. Kompilátor upozorňuje chybou ak využívame API, ktoré je príliš nové na zvolenú verziu operačného systému. Swift 2 funguje s Apple SDK ešte lepšie kvôli generickým systémom a schopnosti SDK upozorniť API, ktoré nemôže vrátiť *nil* hodnotu. (29)
* **3.0:** tretia verzia bola vydaná 13.septembra 2016. Bolo to prvé veľké vydanie novej verzie od uvedenia Swiftu do open-source. Vylepšila sa prehľadnosť jazyka tým, že sa zmenili spôsoby ako sa volajú niektoré funkcie(využívajú sa značky pre parametre). Príklad zmeny: *cars.indexof(“Tesla“)* sa zmenilo na *cars.index(of: “Tesla“)*. Ďalej nastali väčšie zmeny v Linuxovej verzii Swiftu. Pridaný bol aj „Swift Package Manager“, ktorý mal uľahčiť spravovanie závislostí. (24)
* **4.0:** vydaný 29.marca 2018. Jedna z noviniek je protokol „Codable“, ktorý umôžňuje jednoduchú serializáciu dát v štruktúrach do externých formátov ako napríklad JSON. Zmenilo sa fungovanie dátového typu String, ktoré podporuje jednoduchšiu a rýchlejšiu prácu s ním. Taktiež bola pozmenená syntax pre prácu s viacriadkovými Stringami(používajú sa tri úvodzovky pre začiatok aj koniec *“““viacriadkový string“““*). (24)
* **5.0:** vydaný 25.marca 2019. V štandardných knižniciach nastávali viaceré zmeny, ktoré zvýšili výkon. Najväčšia zmena tejto verzie bola pridanie tzv. ABI stability a binárnej kompatibility. Tie zaručujú, že štandardné Swift knižnice sú súčasťou každého operačného systému Apple(macOS, iOS, watchOS, tvOS). Vďaka tomu budú aplikácie jednoduchšie na „buildovanie“ a budú zaberať menšie miesto, lebo nebude potrebné pridávanie daných knižníc. (24) Prechod z verzie 4.0, 4.1 a 4.2 je zabezpečený spätnou kompatibilitou. Pokiaľ chceme migrovať zo skoršej verzie, tak využívame migrátor kódu, ktorý je zahrnutý v Xcode verzii 10.2. Ten dokáže automaticky rozpoznať viaceré miesta, kde je potrebná zmena kódu. (30)

Zaujímavá zmena prišla vo verzii 5.3, v ktorej bola okrem iného expandovaná platformová podpora. Jazyk Swift je podporovaný od tejto verzie aj na operačnom systéme Windows. Pridané boli aj viaceré Linuxové distribúcie(Ubuntu, CentOS 8, Amazon Linux 2). (24)

## Xcode

Pokiaľ hľadáme vhodné IDE na tvorbu aplikácií pre operačné systémy od spoločnosti Apple(macOS, iOS, watchOS, tvOS, iPadOS), tak ideálnou voľbou je Xcode. Je to IDE vyvíjané samotnou spoločnosťou Apple a je určené na operačný systém macOS. Prvá verzia Xcode bola vydaná 28. septembra 2003. Aktuálne je najnovšia stabilná verzia 12.4, ktorá vyšla 26. januára 2021. (31)

Xcode prostredie ponúka prehľadné vytváranie návrhov našej aplikácie, počas ktorého vidíme nielen náš návrh užívateľského prostredia na konkrétnom modely zariadenia ale pomocou rozdeleného okna máme prístup súčasne aj ku kódu aplikácie vďaka asistenčnému editoru a na ľavej strane máme zobrazenú štruktúru projektu. (32) Príklad návrhu užívateľského rozhrania je na obrázku nižšie.



Obrázok 10 Príklad zobrazenia návrhu užívateľského rozhrania v Xcode

Vytváranie samotného návrhu je zabezpečené pomocou vstavaného „Interface buildera“, ktorý ponúka tvorenie konceptu pre užívateľské rozhranie dokonca bez toho, aby bol potrebný akýkoľvek riadok kódu. Aplikácia sa v podstate skladá z jedného alebo viacerých typov pohľadov(views), medzi ktorými sa pohybuje podľa požiadaviek používateľa. Do obrazoviek vkladáme potrebné tlačidlá, tabuľky, popisy, obrázky... Následne ich upravujeme podľa našej predstavy, ako by mali konkrétne obrazovky aplikácie vyzerať. Môžeme napríklad meniť veľkosť tlačidiel, typ, farbu, font písma, upravovať rozloženie objektov na obrazovke, prispôsobovať zmeny podľa veľkosti obrazovky zariadenia... Následne prepojíme návrh s našim kódom, kde implementujeme už konkrétne správanie sa aplikácie. Prepojenie elementu s kódom funguje veľmi jednoducho pomocou potiahnutia daného prvku do kódu. Vďaka asistenčnému editoru môžeme mať otvorený naraz grafický dizajn aj pracovať na kóde. (33)

Xcode ponúka pre spúšťanie aplikácie aj simulátor zariadení, kde je dostupné veľké množstvo podporovaných modelov. Na výber je väčšina doteraz vydaných iPhonov, iPadov a dokonca aj inteligentných hodiniek. Avšak ak potrebujeme simulovať staršie zariadenie, tak bude nevyhnutné stiahnuť potrebný simulátor, keďže defaultne je nainštalovaných v Xcode pár posledných vydaných modelov. Pri vývoji aplikácií na Apple Watch je možné párovanie zvolených hodiniek s vybraným simulátorom iPhone zariadenia. Okrem simulátorov ponúka pripojenie vlastného zariadenia, na ktorom budeme aplikáciu spúšťať. Ak chceme našu vytvorenú aplikáciu publikovať je vždy lepšie ju otestovať aj na fyzickom zariadení, pretože nie všetko sa dá vyskúšať v simulátore. Pre správne pripojenie nášho zariadenia budeme potrebovať vyplniť naše Apple ID k informáciám o účte a priradiť projekt k tímu. (34)

# Cieľ práce

Hlavný cieľ tejto práce je vytvorenie našej vlastnej plne funkčnej mobilnej aplikácie „Nočná lampa“ na zvolenú platformu iOS. Tvorba pozostáva z viacerých krokov:

* Prieskum podobných aplikácií
* Analýza vplyvu svetla na spánok človeka
* Zvolenie základných funkcií
* Návrh dizajnu
* Zvolenie architektúry
* Implementácia navrhnutého riešenia
* Testovanie a oprava chýb
* Zhodnotenie aplikácie

Pri tvorbe budeme postupovať na základe predchádzajúceho prieskumu trhu s podobnými aplikáciami. Ako inšpiráciu pri tvorbe berieme pozitívne hodnotené aplikácie dopĺňané vlastnými nápadmi a návrhmi pre vhodné funkcie. Pri návrhu sa snažíme vyvarovať chybám, na ktoré upozorňovali používatelia v negatívnych hodnoteniach ostatných aplikácií. Taktiež sa snažíme prispôsobovať funkcie a ich spôsob fungovania zisteniam, ku ktorým sme prišli pri analýze pôsobenia nočného svetla na kvalitu spánku.

Keď už máme zvolené funkcie, tak postupujeme k návrhu užívateľského rozhrania. Určujeme ako bude vyzerať naša aplikácia, ako sa bude zobrazovať svetlo a spôsob ako bude navrhnuté vyberanie možností aplikácie. Po zvolení výzoru navrhneme ikonu, poprípade obrazovku načítavania pri spúšťaní.

V ďalšom kroku sa snažíme uvažovať nad tým, ako by sme mohli dané funkcie spolu s dizajnom implementovať t. j. navrhujeme vhodné triedy, atribúty a metódy. Pri návrhu tried berieme do úvahy aj zvolenú architektúru. Pokiaľ to je potrebné, tak vytvárame aj vývojový diagram pre nadobúdané stavy aplikácie.

Ďalej je na rade už samotná implementácia programu. Postupujeme podľa navrhnutého diagramu a poznámok ktoré sme si poznačili k vyžadovaným funkciám. Implementácia sa často krát prekrýva s testovaním, kedy naprogramujeme určitú časť aplikácie a potom testujeme funkčnosť. Pokiaľ je všetko v poriadku, tak pokračujeme na ďalšiu funkcionalitu, ktorú po implementovaní opäť testujeme. Ak už máme hotové všetky časti aplikácie, tak postupujeme k celkovému testovaniu všetkých funkcií. Pri testovaní nie sme obmedzení len na jedno konkrétne zariadenie ale aplikáciu skúšame na viacerých iPhone modeloch. Následne overujeme či je aplikácia správne zobrazovaná na všetkých veľkostiach displejov mobilných zariadení. Ak nachádzame pri testovaní nejakú chybu tak prechádzame späť k implementácií kde nájdený problém odstránime a vraciame sa opäť k testovaniu.

Po vykonaní všetkých potrebných testov a opravení chýb určujeme, či sme dosiahli stanovené ciele a zhodnotíme kvalitu spracovania v porovnaní s našim pôvodným návrhom a s podobnými aplikáciami ktoré sú dostupné na trhu.

# Návrh a implementácia

## Návrh funkcií a vzhľadu

### Platforma, programovací jazyk a IDE

Na začiatku bolo dôležité zvolenie platformy, na ktorú chceme našu mobilnú aplikáciu „Nočná lampa“ vyvíjať. Po dôslednom preskúmaní mobilných operačných systémov, ich obľúbenosti a možnostiach pri vývoji, sme zvolili platformu iOS. Na vývoj pre tento systém sa najčastejšie vyberajú Objective-C alebo Swift. Preto sme sa rozhodovali len medzi týmito dvoma programovacími jazykmi. Napokon sme zvolili jazyk Swift, ktorý postupne nahradzuje v obľúbenosti kedysi často využívaný Objective-C. Keďže všetko potrebné na vývoj aplikácií pre iOS je zahrnuté v IDE Xcode a zároveň ponúka príjemné a pohodlné prostredie pre prácu, tak sme vybrali práve toto IDE vyvíjané a podporované spoločnosťou Apple.

### Výber funkcií

Po vyskúšaní viacerých aplikácií podobného štýlu na vlastnom zariadení a preštudovaní používateľských recenzií, sme mohli vybrať vhodné funkcie, ktoré sa nám páčili alebo boli obľúbené u ďalších užívateľoch. Pri voľbe funkcií prihliadame aj na vlastné nápady a postrehy, čo by mohlo byť v našej aplikácií užitočné. Zvolené funkcionality:

* **Svetlo na celý displej**: pre dosiahnutie ideálneho osvetlenia je vhodné využívať celú obrazovku len pre zvolené svetlo. To znamená, že po zvolení a nastavení osvetlenia by nemala žiadna časť obrazovky obsahovať iné rušivé elementy, okrem vybraného svetla. Kvôli tomu je najvhodnejšie implementovať menu s nastaveniami buď ako samotnú obrazovku, aby sa oddelila od hlavnej funkcie aplikácie alebo vložiť nastavenia do nejakého druhu plávajúceho menu na obrazovke, ktoré zmizne po výbere. Niektoré aplikácie využívali namiesto celej obrazovky pre svetlo napríklad len obrázok lampy, ktorej sa dala nastavovať intenzita svetla. Pri takejto voľbe narážame na problém, že samotné svetlo sa nachádza len na určitej časti displeja a nie sme schopní dosiahnuť také výsledky, ako keď využívame kompletne celú obrazovku.
* **Výber farby:** zvolenie farby lampy považujeme ako jednu z najdôležitejších funkcií v našej aplikácií. Je veľmi užitočné, keď si používateľ môže nastaviť farbu podľa jeho preferencií, nálady alebo typu aktivity ku ktorej potrebuje svetlo. Volíme iné svetlo pokiaľ chceme čítať knihu ako keď potrebujeme jemné osvetlenie pri pozeraní filmu. V samotných recenziách boli pozitívne ohlasy, pokiaľ bola v aplikácií táto možnosť. Ďalším z dôvodov pre zahrnutie funkcie zvolenia farby svetla bolo, že má výrazný vplyv na kvalitu spánku, ako bolo bližšie spomínané v kapitole 1.2 Pôsobenie svetla na ľudský organizmus počas spánku. Na výber by sme chceli dať z celej farebnej škály farieb.
* **Nastavenie intenzity:** väčšina inteligentných mobilov síce častokrát má nastavenie jasu rýchlo dostupné ale príjemnejšie a pohodlnejšie pre používateľov bude, keď bude táto možnosť zahrnutá priamo v aplikácii, keďže intenzita je kľúčová pri nastavení vhodného osvetlenia. Niekedy potrebujeme jasnejšie a ostrejšie svetlo a inokedy nám postačí aj nižší jas lampy.
* **Časovač:** vďaka používateľským recenziám sme zistili, že nočné lampy sú často krát využívané počas spánku, pretože niektorým ľuďom nie je príjemne zaspávať pri úplnej tme. Z toho dôvodu potrebujú jemné osvetlenie izby, častokrát využívané najmä pre malé deti. Pri takomto využití našej aplikácie bude vhodné, aby neostala svietiť lampa celú noc ale aby sa po určitom čase vypla. Výrazne sa tým ušetrí batéria mobilného zariadenia.
* **Prechod medzi farbami:** jedna zo zaujímavých funkcií pri nočných lampách bola zmena farieb po určitom čase. V našej aplikácii by sme chceli, aby si používateľ mohol zvoliť, či chce mať svetlo len jednej farby alebo aby sa medzi sebou viaceré striedali. Ak si vyberie prechod viacerých farieb, tak bude mať možnosť si navoliť štyri farby, ktoré sa budú časom meniť. Po výbere farieb by si mohol nastaviť rýchlosť akou bude prechádzať aplikácia medzi zvolenými farbami.
* **Ukladanie nastavení:** keď si používateľ nastaví nejakú farbu pre jednofarebné osvetlenie alebo aj pre prechod medzi farbami, tak sa tieto údaje ukladajú, aby po vypnutí a následnom opätovnom spustení zostali zachované predchádzajúce nastavenia. Týmto dosiahneme, že pri každom spustení nie je potrebné ručne prestavovať, ak má užívateľ obľúbenú farbu, ktorú využíva stále.

### Ovládanie

Viacero negatívnych hodnotení ktoré sa pri aplikáciách vyskytovali, boli ohľadom nepríjemného a komplikovaného ovládania, napríklad pri výbere farieb svetla. Ďalším problémom bola aj celková neprehľadnosť aplikácie a neintuitívne ovládanie. Z týchto dôvodov by sme chceli spraviť aplikáciu tak, aby bola používateľsky jednoduchá a aby bolo hneď od začiatku používania jasné, ako sa čo ovláda a nastavuje.

Aplikácia bude mať len jednu obrazovku, na ktorej bude zvolená farba zobrazená na celý displej. Pokiaľ bude chcieť používateľ niečo zmeniť, bude stačiť aby ťukol na hocijaké miesto na obrazovke a objaví sa plávajúce menu, v ktorom bude mať na výber z ponúkaných funkcií. Na pravej strane bude po zobrazení menu tzv. „slider“, pomocou ktorého si používateľ nastaví intenzitu osvetlenia.

Ďalej bude v menu na výber medzi farbou, časovačom a prechodom medzi farbami. Keď bude zvolená možnosť pre farbu tak sa v plávajúcom menu zobrazí „slider“ alebo paleta farieb, ktorými si budeme môcť zvoliť farbu, akú pre osvetlenie chceme. Ak v menu používateľ zvolí nastavenie časovača, tak sa zobrazí možnosť výberu času po akom sa vypne displej. Na výber budú minúty a hodiny, pričom minimálna dĺžka bude jedna minúta a maximálna 23 hodín a 59 minút. Pokiaľ bude zvolená možnosť nastavenia prechodu medzi farbami, tak sa zobrazia štyri ukazovatele aktuálne zvolených farieb medzi ktorými sa bude prechádzať. Ak bude chcieť používateľ zmeniť nejakú z nich tak stačí vybrať daný ukazovateľ a pomocou rovnakého spôsobu ako pri výbere jednofarebného osvetlenia si zvolí požadovanú farbu. Následne nastaví rýchlosť akou sa budú vybrané farby medzi sebou meniť. Pre lepšiu predstavu ako bude ovládanie fungovať slúži predbežný návrh používateľského prostredia na obrázku 13 Návrh používateľského rozhrania.

### Ikona

Každá aplikácia potrebuje ikonu, ktorá by upútala potenciálnych používateľov v distribuovanom obchode App store. Apple odporúča viaceré pravidlá pri vytváraní ikony. Jedným z nich je zameranie sa na jednoduchosť. Je dôležité nájsť kľúčový element, ktorý vystihuje našu aplikáciu a vyjadriť ho nie príliš zložitým spôsobom, aby sa aj na menších displejoch dal jednoducho rozpoznať. Podstatná je aj unikátnosť, z dôvodu rozlíšenia od ostatných aplikácií. Pozadie ikony by malo byť nepriehľadné, najčastejšie jednofarebné kvôli jednoduchosti. Slová by sa mali používať iba ak to je nevyhnutné. Väčšinou je to nepotrebné, keďže názov aplikácie sa nachádza priamo pod ikonou. Pri návrhu treba myslieť aj na to, aké pozadie domovskej obrazovky môže mať používateľ. (35)

Pri vytváraní ikony pre našu aplikáciu som sa snažil dodržiavať vyššie spomenuté pravidlá. Hlavný element ktorý vystihuje aplikáciu je samotná nočná lampa. Preto sme skúšali viaceré návrhy pre model vhodnej lampy a viaceré farebné kombinácie. Keď sme využívali moc veľa farieb, prvotný dojem nebol až taký dobrý a na základe toho sme zvolili jednoduchú čiernobielu kombináciu(čierne pozadie na ktorom je biely obrys nočnej lampy). Nižšie je zobrazená naša vytvorená ikona na domovskej obrazovke(Night light).

Obrázok, na ktorom je text, elektronika, zobraziť

Automaticky generovaný popis

Obrázok 11 Ikona aplikácie na domovskej obrazovke

Xcode potrebuje viaceré rozmery pre ikonu aplikácie, kvôli rôznym veľkostiam displejov zariadení. Využívajú sa rozdielne rozmery pre zobrazenie ikony na domovskej obrazovke ako na zobrazenie pri upozorneniach aplikácie, v nastaveniach alebo vo vyhľadávaní užívateľských aplikácií v mobile. Do nášho projektu potrebujeme všetky vyžadované veľkosti, ak chceme aby sa ikony zobrazovali.

### Úvodná obrazovka

Úvodná stránka je zobrazená okamžite po spustení a následne sa po krátkom čase dostávame dovnútra aplikácie. Každá aplikácia na iOS má povinnú úvodnú obrazovku. Jej vytváranie je zabezpečené pomocou rovnakej funkcionality Xcode, pomocou ktorej tvoríme aj používateľské rozhranie, tzv. storyboard. Príklad práce v storyboarde je na obrázku 10 Príklad zobrazenia návrhu užívateľského rozhrania v Xcode. Apple odporúča aby bola úvodná obrazovka podobného štýlu ako rozhranie aplikácie. Tu nastáva problém, keďže naša aplikácia zobrazí hneď po spustení zvolenú farbu osvetlenia a tá môže byť zakaždým iná. Preto bude zrejme najlepší variant zvoliť čierne pozadie na ktorom bude biele logo nočnej lampy, ktoré sme použili aj pre ikonu. Čierna farba pozadia bola zvolená kvôli tomu, aby sa nezobrazilo ostré svetlo, ak chce používateľ využiť aplikáciu pre jemné osvetlenie, napríklad ak sa v noci zobudil a potrebuje ísť do kúpeľne. (36)



Obrázok 12 Úvodná obrazovka

### Používateľské rozhranie

Pred samotnou implementáciou aplikácie nočnej lampy je vhodné spraviť predbežný návrh používateľského rozhrania, aby sme si špecifikovali čo má ako fungovať, aké sú požiadavky od programátora a aby bolo jednoduchšie si vizuálne predstaviť požadované ovládanie.

Pre vytvorenie dizajnu používateľského rozhrania pre našu aplikáciu sme zvolili program Sketch, ktorý je často využívaný nielen jednočlennými vývojármi ale aj veľkými tímami v nemalých spoločnostiach. Využíva sa pre dizajn aplikácií, hier alebo webových stránok. Pomocou tohto programu môžeme jednoducho vizualizovať naše predstavy o tom, ako by mali jednotlivé stránky vyzerať a ovládať sa. Sketch ponúka stiahnutie viacerých knižníc, v ktorých nájdeme jednotlivé elementy podľa toho na akú platformu chceme vytvoriť návrh(iOS, macOS, watchOS alebo tvOS). Pre naše potreby sme si stiahli iOS knižnicu. Na obrázku nižšie máme zobrazený náš predbežný návrh, kde sú zobrazené štyri obrazovky kvôli lepšej predstave.



Obrázok 13 Návrh používateľského rozhrania

* Na prvej obrazovke je zobrazenie aplikácie po tom ako skryjeme nastavenia pomocou ťuknutia na hocijakú časť obrazovky. Celú obrazovku v tom prípade pokrýva farba ktorú sme si ako používateľ zvolili.
* Druhá obrazovka zobrazuje plávajúce menu nastavení, v ktorom je aktuálne zvolená možnosť pre nastavovanie farby. Farbu si zvolíme pomocou „slideru“. Ďalej si môžeme nastaviť jas pomocou „slideru“, ktorý je na pravej strane obrazovky.
* Na tretej obrazovke je zvolené nastavenie časovača, po uplynutí ktorého sa vypne displej zariadenia.
* Štvrtá obrazovka zobrazuje nastavenie prechodu medzi farbami a zvolenie rýchlosti prechodu medzi nimi.

## Architektúra a diagram

V každej aplikácii je odporúčané vhodne zvoliť architektúru, podľa ktorej budeme mať usporiadané jednotlivé časti a ako budú medzi sebou fungovať. Pre tieto účely sme zvolili známu architektúru MVC(Model View Controller), ktorá bola predstavená už v roku 1970 v programovacom jazyku Smalltalk. Táto architektúra oddeľuje od seba logiku aplikácie a spôsob zobrazenia. Pozostáva z troch komponentov, ktoré medzi sebou spolupracujú (18):

* **Model:** predstavuje dáta aplikácie, poprípade kód pomocou ktorého ich získava, napríklad načítanie údajov z databázy alebo z uložených súborov. Umožňuje pracovať s dátami a postupne ich aktualizovať. V tejto časti sa vôbec nerieši, ako sa dané dáta v aplikácii zobrazia.
* **View:** časť ktorá špecifikuje ako budú zobrazované údaje. Slúži na to, aby určila ako bude vyzerať aplikácia, poprípade jej časti. Môže poskytovať viac pohľadov pre jeden vytvorený model.
* **Controller:** má v sebe najväčšiu časť aplikačnej logiky. Zabezpečuje spoluprácu medzi používateľom, modelom a pohľadom(view). Dokáže spracovať vstupy od používateľa, pristúpi k dátam modelu a zobrazí konkrétny pohľad užívateľovi. Má na starosti celé riadenie aplikácie, čiže čo a ako sa má vykonávať.

V aplikácii „Nočná lampa“ bude časť model predstavovať dáta ohľadom farieb pri statickom svetle(zobrazená jedna farba) alebo prechodom medzi farbami, údaje o časovači, rýchlosti prechodu medzi farbami a jase.

Časť pohľadu bude potrebná len jedna, keďže v aplikácii budeme pracovať len so zobrazením farby na celú obrazovku a všetky nastavenia budú zastúpené na tom istom pohľade.

Controller bude tvoriť najväčšiu časť našej aplikácie. Potrebný bude taktiež iba jeden controller, pretože pracujeme iba s jedným pohľadom.

Kvôli bližšiemu upresneniu nášho návrhu o tom, ako by sme chceli aby aplikácia fungovala a v akých stavoch sa nachádzala, je vhodné vytvoriť adekvátny vývojový diagram. Ten nám prehľadne znázorní, aké procesy sa budú počas behu vykonávať a zobrazí rozhodnutia, na základe ktorých sa bude prechádzať k rôznym častiam.

Vývojový diagram má pri tvorbe projektu viaceré pozitíva. Programátor ho môže vytvoriť aby si určil, ako budú jednotlivé kroky vyzerať. Z tých je schopný odvodiť približný postup pri implementácii. Diagramy ale neslúžia len pre vývojárov. Pokiaľ sa projekt tvorí pre väčšiu firmu, kde sa neustále kontroluje štádium v akom sa pri vývoji nachádzame alebo niekomu chceme predstaviť náš projekt, tak sa pri prezentácii aktuálnych výsledkov môže použiť aj diagram pre bližšie uvedenie zákazníkov do fungovania jednotlivých častí. Nižšie je zobrazený náš vývojový diagram behu aplikácie.



Obrázok 14 Vývojový diagram

## Výsledná implementácia

Po dôkladnom premýšľaní o spôsobe implementácie a vytvorení potrebných návrhov, či už používateľského rozhrania, všeobecného fungovania alebo výberu funkcií, pristupujeme ku hlavnému bodu tvorby aplikácie, t. j. samotná implementácia.

Postupne sme prechádzali jednotlivé body pri vývoji. Najprv sme sa venovali tvorbe používateľského rozhrania podľa nami vytvoreného návrhu z kapitoly 4.1.6 Používateľské rozhranie. Potom sme postúpili k implementovaniu konkrétnych navrhnutých funkcií a prepojili sme to s našim rozhraním. Tieto činnosti sa častokrát prelínali, keďže bolo potrebné viac krát niektoré funkcie meniť a upravovať, poprípade prispôsobovať rozhranie podľa spôsobu fungovania funkcie. Pri vývoji sme samozrejme narážali aj na viacero chýb, ktoré sa objavovali pri testovaní. Pre priebežné testovanie sme využívali viaceré simulátory zariadení a po väčších zmenách aj fyzické iOS zariadenie.

### Používateľské rozhranie

Na vytvorenie používateľského rozhrania môžeme využiť Xcode funkcionalitu storyboard, ktorá bola spomínaná v predchádzajúcich kapitolách. Vytvárali sme v nej taktiež úvodnú obrazovku, ktorá sa zobrazuje pri spúšťaní aplikácie.

Storyboard sa odporúča používať pre menšie aplikácie, ktoré nevyužívajú veľmi veľa pohľadov(views) a nie sú rozsiahle. Je to z dôvodu orientácie, pretože pri veľkom počte vložených pohľadov, medzi ktorými aplikácia prechádza sa stáva storyboard neprehľadný. Aplikácia „Nočná lampa“ bude mať podľa nášho vytvoreného návrhu len jeden pohľad. Storyboard je preto vhodnou voľbou pre vytváranie používateľského rozhrania. Ten ponúka približné grafické zobrazenie, ako bude obrazovka vyzerať po zobrazení pohľadu. Samotný storyboard tvorí časť „view“ z MVC architektúry našej aplikácie.

Pri tvorbe pohľadu postupujeme tak, že najprv vkladáme požadované elementy(obrázky, tlačidlá, popisy...) na obrazovku. Následne ich usporiadame podľa našich predstáv a vykonáme potrebné úpravy. Možnosti úpravy sú priamo v grafickom prostredí storyboardu pomerne široké, kde môžeme napríklad pre tlačidlo nastaviť font, veľkosť fontu, farbu textu, farbu pozadia, vložiť obrázok pre tlačidlo... Možnosti ktoré sa nenachádzajú v storyboarde sa programujú neskôr po prepojení elementu s triedou.

Na obrázku nižšie je zobrazené naše používateľské rozhranie, v ktorom sú znázornené nami pridané elementy pre pohľad aplikácie. Aj keď využívame iba jeden pohľad, tak pre prehľadnosť sme rozdelil elementy na tri obrázky, keďže sa budú skrývať podľa zvolenej možnosti v plávajúcom menu(Farba, Časovač, Prechod farieb). Postupovali sme podľa nášho návrhu, kde bude na celú obrazovku zobrazená zvolená farba a po ťuknutí sa objaví plávajúce menu spolu so „sliderom“ pre nastavenie jasu. Výška menu nie je zobrazená presne, pretože sa bude dynamicky meniť podľa voľby v menu. Táto možnosť sa doprogramuje až v kóde. Bližší popis jednotlivých rozložení je pod obrázkom.

Obrázok, na ktorom je text

Automaticky generovaný popisObrázok, na ktorom je text

Automaticky generovaný popisObrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 15 Storyboard návrh pre používateľské rozhranie

Prvý obrázok zľava zobrazuje usporiadanie elementov pre zvolenú možnosť „Farba“. Na pravej strane obrazovky je pridaný „slider“, zatiaľ v horizontálnej polohe, pretože otočenie prvkov sa dá vykonávať len v kóde. V plávajúcom menu bude zobrazený popis pre naposledy použité farby, pod ktorým bude 5 tlačidiel reprezentujúcich ich. Pokiaľ bude mať ale používateľ v minulosti zvolených menej ako 5 farieb, tak sa nepoužité tlačidlá skryjú. Tlačidlá sa ešte budú upravovať v kóde, ako napríklad zaoblenie rohov. Naposledy zvolené farby sú pridané z dôvodu rýchlej možnosti voľby z predchádzajúcich výberov. Ak chce používateľ zvoliť inú farbu, tak použije tlačidlo pre otvorenie palety farieb. Po výbere sa nastaví pozadie na jeho voľbu a pridá sa na prvú pozíciu pre naposledy zvolené farby. Bližší popis pre implementáciu výberu farby nájdeme v kapitole 4.3.3 Výber farby.

Stredný obrázok reprezentuje výzor plávajúceho menu po zvolení možnosti „Časovač“. Ten obsahuje len tri elementy: tlačidlo pre spustenie/zastavenie časovača, popis zobrazujúci ostávajúci čas a tzv. „date picker“, v ktorom sa nastavia hodiny a minúty pre dĺžku trvania. Popis ostávajúceho času a „date picker“ sa budú v zobrazení striedať, t. j. vždy bude viditeľný len jeden, podľa toho či je časovač spustený alebo nie. Ak nie je, tak sa zobrazuje „date picker“. Ak je naopak časovač spustený, tak sa skryje a namiesto neho sa zobrazuje koľko času ostáva pred zhasnutím displeja.

Usporiadanie menu po zvolení poslednej možnosti „Prechod farieb“ je zobrazené na obrázku najviac vpravo. Na pravej strane menu je vložené tlačidlo pre spustenie/zastavenie prechodu medzi farbami. Vedľa neho sú zobrazené štyri tlačidlá, ktoré reprezentujú zvolené farby, medzi ktorými sa bude prechádzať. Nad nimi sú vložené obrázky šípok, ktoré sa v kóde otočia aby ukazovali na priradenú farbu. Po kliknutí na jednotlivé tlačidlo farby sa skryjú všetky ostatné šípky a bude viditeľná iba tá, ktorá ukazuje na vybrané tlačidlo. Tieto obrázky sú tam pridané kvôli tomu, aby bolo pre používateľa jasné, akú farbu môže momentálne nastavovať. Pre zmenu farby sa využíva rovnaký princíp ako v prvej možnosti menu a to pomocou palety farieb. Toto tlačidlo sa nachádza aj s popisom pod tlačidlami farieb prechodu. Posledný vkladaný element v menu je „slider“, pomocou ktorého sa nastavuje ako rýchlo medzi sebou budú vybrané farby prechádzať. Priradený má samozrejme aj popis kvôli lepšej orientácii v menu.

Po pridaní všetkých elementov a ich prepojení s ViewController triedou(tvorí „controller“ časť v MVC architektúre), kde ich môžeme ovládať pomocou kódu, ešte musíme nastaviť obmedzenia. Vďaka ním zaistíme, aby sa prvky zobrazovali na obrazovke správne pre všetky veľkosti zariadení. Nastavujú sa obmedzenia pre šírku, výšku elementu alebo vzdialenosti od ostatných prvkov, napríklad vzdialenosť od okraju obrazovky, alebo od nejakého tlačidla. Taktiež umožňuje zarovnanie elementu na stred, buď vertikálne alebo horizontálne. Pomocou kombinácií všetkých týchto možnosti sa snažíme dosiahnuť najlepšie všeobecné zobrazenie pre obrazovky mobilných zariadení. Taktiež musíme prispôsobiť usporiadanie prvkov pre zmenu polohy zariadenia(na výšku alebo šírku).

### Ukladanie používateľských údajov

Pre pohodlné používanie „Nočnej lampy“ je vhodné, aby sa zvolené používateľské údaje ukladali a následne boli schopné načítať. Je to dôležité kvôli tomu, aby si aplikácia pamätala napríklad akú farbu si užívateľ zvolil naposledy. To znamená, že po ukončení a opätovnom spustení, by aplikácia mala byť schopná dané údaje o naposledy zvolenej farbe načítať a zobraziť ju.

Ukladanie dát sme vyriešili pomocou vstavanej triedy UserDefaults. Táto trieda podporuje prácu s lokálnou používateľskou databázou. Tá sa odporúča používať iba na malé dáta, ktoré musia byť načítané po spustení aplikácie, napríklad používateľské nastavenia. Pre naše účely to úplne postačí. Táto databáza funguje na princípe, kde priradíme dátam kľúč a potom môžeme pomocou neho k dátam pristupovať.

V implementácii sme si vytvorili štruktúru UserData, kde sme si definovali všetky premenné, ktoré musíme ukladať. Táto štruktúra tvorí „model“ časť z architektúry MVC. Pri inicializácii štruktúry UserData sa vždy najprv pokúšame prečítať konkrétne dáta z databázy podľa priradeného kľúču. Ak pre daný kľúč neexistujú žiadne uložené dáta(stáva sa vždy pri prvom spustení aplikácie), tak priraďuje premenným defaultne hodnoty(podľa našich definovaných konštánt v programe). Tieto údaje potom ukladá do UserDefaults databázy. Počas behu aplikácie sa pri vykonaní každej zmeny niektorej z týchto premenných opäť ukladá nová zvolená hodnota. Potrebné bolo uloženie týchto údajov:

* **Aktuálne zvolená farba:** využíva sa okamžite po spustení aplikácie, kedy sa nastavuje pozadie na túto farbu. Jej hodnota sa mení vždy po výbere buď z nedávno zvolených farieb alebo po voľbe pomocou palety farieb. Pri ukladaní pomocou UserDefaults sme narazili na problém, že neumožňovala ukladanie typu UIColor(typ používaný pre definovanie farby). Preto bolo potrebné pred uložením prekódovať premennú na dáta. Po nasledovnom načítaní sa musela opäť použiť spätná konverzia na typ UIColor. Pre ukladanie a načítavanie typu UIColor sme si vytvorili všeobecné rozšírenie triedy UserDefaults, ktoré sme využívali nielen pre aktuálne zvolenú farbu ale aj pre prácu s naposledy zvolenými farbami. (37)
* **Naposledy zvolené farby:** hodnoty sa priraďujú tlačidlám pre naposledy zvolené farby, ktorými sa môže meniť farba pozadia. Zmena dát a ich opätovné ukladanie nastáva pri výbere novej farby z palety. Do databázy sa ukladá celé pole. Na začiatku má veľkosť 1, kedy má priradenú len defaultnu hodnotu. Postupne sa rozširuje po výberoch nových farieb. Maximálne môže nadobúdať veľkosť 5, pretože v používateľskom rozhraní sa zobrazuje len 5 naposledy zvolených farieb. Okrem samotného poľa sa ukladá aj celočíselná hodnota indexu, kam sa má v poli uložiť ďalšia zvolená hodnota. Pomocou tohto indexu sa premazáva najstaršie zvolená farba. Hodnota indexu sa mení pri každom zápise do poľa. Práca s poľom farieb je bližšie popísaná v ďalšej kapitole.
* **Hodnota časovača:** ukladané pomocou poľa dvoch celočíselných hodnôt, kde nultý index reprezentuje hodiny a index 1 uchováva údaje o minútach. Pokiaľ nie sú dáta pri čítaní nájdené, priraďuje sa časovaču defaultna hodnota 1 hodina a 30 minút. Opätovné ukladanie dát nastáva zakaždým pri zmene hodnoty minút alebo hodín v časovači.
* **Rýchlosť prechodu:** určuje ako rýchlo sa budú medzi sebou striedať farby. Čím nižšie číslo, tým dlhšie bude trvať čas pred zmenou. Hodnota je reprezentovaná číslom typu Float. Po načítaní je priradená „slideru“, ktorý určuje rýchlosť prechodu. Ukladanie dát nastáva pri vykonaní pohybu „slideru“. Môže nadobúdať hodnoty od 0.0 do 1.0.
* **Zvolené farby pre prechod medzi farbami:** spôsob ukladania a načítania je rovnaký ako pre naposledy zvolené farby. V tomto prípade sa veľkosť poľa nikdy nemení. Má trvalú veľkosť 4, pretože aplikácia ponúka prechod medzi 4 farbami. Z tohto dôvodu nie je potrebné uchovávať si hodnotu indexu, kam sa vkladá ďalšia hodnota. Pokiaľ v databáze neexistujú uložené údaje, tak sa vytvára nové pole kde sa vložia 4 defaultne hodnoty. Ukladanie nastáva pri zmenách farieb pre konkrétne zvolené tlačidlo(podľa neho vieme pristúpiť na správny index v poli).

### Výber farby

Jedna z najdôležitejších funkcii je vyberanie požadovanej farby pre osvetlenie. Po spustení aplikácie sa zobrazí samotná zvolená farba na celú obrazovku. Pre implementáciu rôznych nastavení pre používateľa sme postupovali podľa nášho vytvoreného návrhu z kapitoly 4.1.6 Používateľské rozhranie. Oproti návrhu sme sa rozhodli pridať aj históriu zvolených farieb, ku ktorým sa používateľ bude môcť vrátiť po kliknutí na ich tlačidlo.

V aplikácii sme vytvorili plávajúce menu, ktoré sa zobrazí po ťuknutí na hociktorú časť obrazovky a jeho obsah sa bude meniť podľa zvolenej možnosti. Pre jednotlivé voľby sme si vytvorili konštanty pre výšku plávajúceho menu, keďže pre každú je rozdielne množstvo zobrazovaných elementov. Aplikácia kontroluje zmeny vo výbere možností. Po jej vykonaní sa overí, ktorá možnosť bola zvolená a podľa toho sa priradí správna výška menu, zobrazia sa jej elementy a ostatné sa skryjú.

Po zvolení voľby „Farba“ sa zobrazia v menu tlačidlá s naposledy použitými farbami. Tieto údaje sa načítavajú z lokálnej databázy. Hodnoty farieb sú uložené v poli s maximálnou veľkosťou 5. Farby sa musia nejakým spôsobom prepisovať, pokiaľ bolo v minulosti zvolených už viac ako 5 hodnôt. To je zabezpečené pomocou ďalšej premennej, ktorá uchováva index, na ktorý sa bude v poli nabudúce zapisovať. Index funguje na princípe stáleho zvyšovania o jedna a následného vykonania modulo 5 pri každom zápise do poľa. Ak je veľkosť poľa pri zápise menšia ako 5, tak sa vloží hodnota farby vždy na koniec. Ak sme naopak dosiahli maximálnu kapacitu, tak vkladáme farbu na index podľa vytvorenej premennej, napríklad ak má premenná index hodnotu 3 tak sa v poli vloží farba na index 3 a následne sa premenná zvýši o jedna.

Pri výpise naposledy zvolených farieb z poľa taktiež využívame premennú index, ktorá ukazuje kam sa nabudúce v poli zapisuje. Používa sa kvôli tomu, že ak sme dosiahli kapacitu 5, tak tento index reprezentuje najstaršiu zvolenú farbu. Pre výpis sa postupuje od najstaršej po najnovšiu. Ak je v histórii zvolených menej ako 5 farieb tak sa po výpise skryjú nepoužité tlačidlá.

Pokiaľ chce používateľ zvoliť inú farbu ako z naposledy zvolených, tak stláča tlačidlo pod nimi, ktoré otvára paletu farieb. V pôvodnom návrhu bolo vyberanie farby len pomocou jedného „slidera“. Po implementovaní takej možnosti sme si uvedomili, že nie sme schopní nastaviť dostatočnú škálu farieb. Kvôli tomu sme zvolili inú možnosť. Swift ponúka od verzie iOS 14.0 tzv. ColorPicker, ktorý zobrazí paletu farieb. V nej si môže používateľ vybrať z veľkého množstva ponúkaných farieb v prehľadne vytvorenom používateľskom rozhraní. Výzor použitej palety je zobrazený na Obrázok 18 Paleta farieb ColorPicker.

Po samotnom výbere farby je potrebné ešte vykonanie zmeny status baru(ukazovateľ času, batérie a ostatných ikoniek na vrchu obrazovky). Ten môže nadobúdať čiernu alebo bielu farbu. Kontrola zmeny status baru nastáva vždy po zmene farby. Funkcia overí jas zvolenej farby a podľa toho prispôsobuje status bar pre tmavý alebo svetlý variant.

### Časovač

Po výbere druhej možnosti v menu sa zobrazí nastavenie časovaču. V ňom máme aktívne vždy len dva elementy. Jeden je tlačidlo pomocou ktorého je používateľ schopný spustiť zvolený časovač a druhý sa strieda medzi výpisom aktuálne zostávajúceho času a tzv. „date pickeru“ ktorý umožňuje nastavovať hodiny a minúty. Druhý element sa vyberá podľa toho, či je časovač spustený alebo nie. Použitý „date picker“ sa vo swifte používa na určenie konkrétneho dátumu alebo času. V našom prípade sme ho prispôsobili, aby ponúkal výber hodín a minút.

Po stlačení tlačidla pre spustenie alebo vypnutie sa overí, či je časovač spustený. Pokiaľ nie je, tak ho aktivuje podľa priradených hodnôt pre hodiny a minúty. Po jeho spustení sa pristupuje každú sekundu do funkcie, ktorá odpočíta jednu sekundu, prepíše zostávajúci čas prepočítaný na hodiny, minúty a sekundy a overí či časovač nevypršal. Ak vypršal, tak ruší časovač pomocou ktorého sme pristupovali každú sekundu do funkcie. Po uplynutí tohto času sme chceli aby sa displej vypol. Avšak Swift neponúka funkciu pomocou ktorej by sme mohli natvrdo užívateľovi vypnúť displej. Preto sme po uplynutí času len aplikáciu vypli, nastavili jas na minimum aby nespôsoboval rušiace svetlo a displej sa sám vypne po uplynutí všeobecného nastavenia času po nečinnosti používateľa.

Pokiaľ časovač aktuálne beží a stlačí sa tlačidlo pre zapnutie/vypnutie, tak prerušuje časovač, skryje ukazovateľ zostávajúceho času a namiesto neho opäť zobrazí „date picker“ na výber hodín a minút.

### Prechod medzi farbami

Posledná funkcionalita, ktorú sme implementovali bolo striedanie farieb po uplynutom čase. V menu sa po tejto voľbe zobrazia štyri tlačidlá reprezentujúce farby, medzi ktorými sa bude prechádzať. Pri každej z nich je priradená šípka, ktorá ukazuje na farbu ktorá sa môže aktuálne meniť(vždy je viditeľná len jedna šípka). Hodnoty farieb sú uchovávané v poli s nemennou veľkosťou 4.

Zmena farby funguje rovnako ako v prvej voľbe menu. Vyberá sa pomocou palety farieb. Zvolená hodnota sa priradí do indexu poľa podľa toho, na ktoré tlačidlo ukazuje aktuálne šípka.

Po stlačení tlačidla pre zapínanie/vypínanie prechodu farieb sa overuje, či je prechod aktívny. Pokiaľ nie je, tak preberá hodnotu rýchlosti prechodu, ktorá je nastaviteľná pre používateľa pomocou „slideru“. Podľa zvolenej rýchlosti sa určí ako často bude vstupovať do funkcie, ktorá ovláda prelínanie medzi farbami. Pôvodne sme implementovali okamžitý prechod(farba sa hneď zmenila na ďalšiu), ktorý pôsobil veľmi rušivo. Preto sme upravili kód tak, aby po uplynutí času postupne pozadie tmavlo a potom sa preplo na ďalšiu farbu, ktorá sa časom rozjasní. Takýto prechod je plynulý a príjemnejší. Farby sa menia cyklicky pri prechode hodnotami poľa.

Ak používateľ zmení rýchlosť prechodu počas toho ako je aktívny, tak sa ruší pôvodný časovač pre vstupovanie do funkcie, vytvára sa nový s upravenou rýchlosťou a začína opäť od indexu 0. Ak sa stlačí tlačidlo pre spustenie/vypnutie prechodu medzi farbami a prechod je aktívny, tak sa taktiež ruší časovač a pozadie sa nastaví na hodnotu pôvodne zvolenej farby statického pozadia.

Počas prechodu medzi farbami sa musí riešiť aj zvolenie tmavého alebo svetlého status baru, keďže môžu byť zvolené rôzne tmavé farby. Overenie pre voľbu správneho status baru nastáva po tom, ako nastal úplný prechod na danú farbu, t. j. po skončení stmavenia.

### Testovanie

Aplikácia bola počas implementácie neustále testovaná po pridaní nových funkcií, aby sa nedostatky odhalili čo najskôr. Pre testovanie sa využívali simulátory zariadení rôznych veľkostí displeja. Použité boli modely iPhone SE(druhá generácia), iPhone 11, iPhone 12 mini a iPhone 12 Pro Max kvôli ich rôznorodým veľkostiam. Mohli sme teda vyskúšať, či sa elementy zobrazujú správne na všetkých modeloch. Taktiež sa skúšala správnosť zobrazenia na výšku aj na šírku, napríklad „slider“ pre nastavenie jasu displeja sa po otočení na šírku pretočí na vodorovnú polohu, aby bol celý viditeľný.

Aj keď sú simulátory kvalitne spracované, nie je to skutočné zariadenie. Kvôli tomu je vhodné po vyskúšaní aplikácie cez simulátor, otestovať ju aj na fyzickom zariadení, pokiaľ máme k nemu prístup. Niektoré funkcie sa dokonca nedajú vyskúšať na simulátore alebo to nie je také jednoduché, napríklad sa nezobrazuje zmena jasu displeja. Pre testovanie na fyzickom zariadení sme použili model iPhone SE druhej generácie. Všetky záverečné testovania prebehli v poriadku a aplikácia je plne funkčná so správnym zobrazovaním na všetkých veľkostiach displejov.

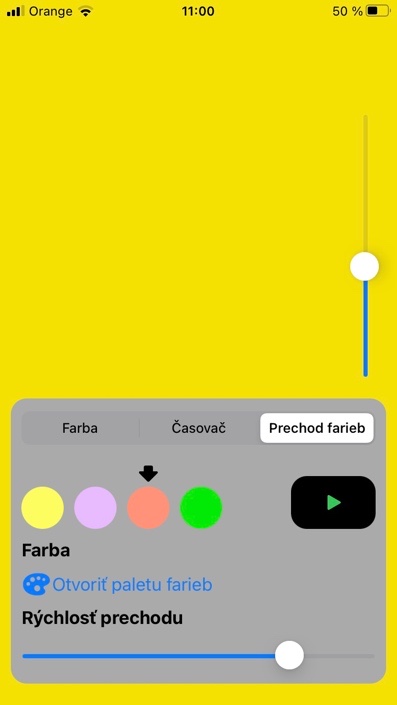
# Výsledky práce

Po vykonaní implementácie v jazyku Swift vo vývojovom prostredí Xcode a precíznom testovaní sme dokončili našu vlastnú aplikáciu „Nočná lampa“. Pri vývoji sme postupovali podľa našich vytvorených návrhov, diagramov a stanovených postupov, popri ktorých sme občas pristupovali aj k prispôsobovaným zmenám, kvôli dosiahnutiu lepšej funkčnosti alebo pohodlnosti. Výsledná aplikácia je svižná, jednoduchá na použitie a plní všetky požadované funkcionality pre nočnú lampu.

## Ukážka funkčnej aplikácie

Výsledné ukážky funkcionalít našej vytvorenej aplikácie pochádzajú s fyzického zariadenia iPhone SE druhej generácie. Na obrázkoch nižšie je zobrazené ako vyzerá naša implementácia aplikácie.

Obrázok, na ktorom je námestie

Automaticky generovaný popis

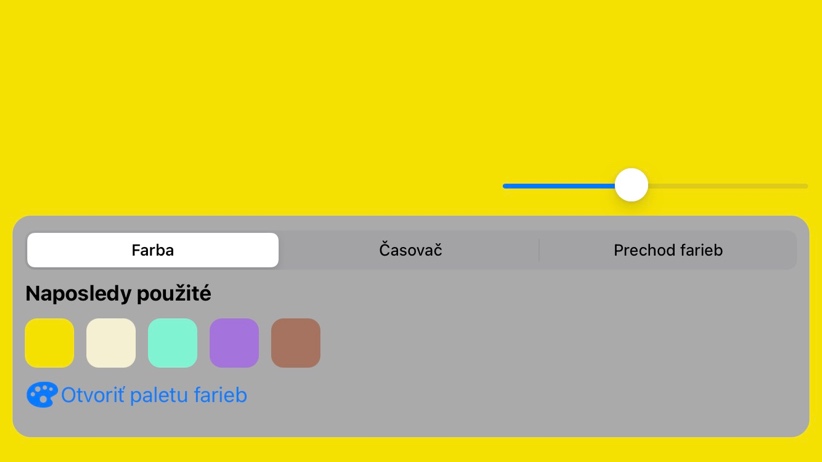
Obrázok 16 Ukážka aplikácie: farba a prechod medzi farbami

Obrázok 16 ukazuje tri rôzne zobrazenia aplikácie. Na prvej z nich je plávajúce menu schované po ťuknutí na obrazovku. V takom momente celú plochu obrazovky zaberá zvolená farba.

Stredná obrazovka ukazuje plávajúce menu po výbere možnosti „Farba“. Na výber je posledných 5 použitých farieb. Pod nimi je tlačidlo, ktorým sa otvára paleta farieb, kde si môže používateľ zvoliť ľubovoľnú farbu.

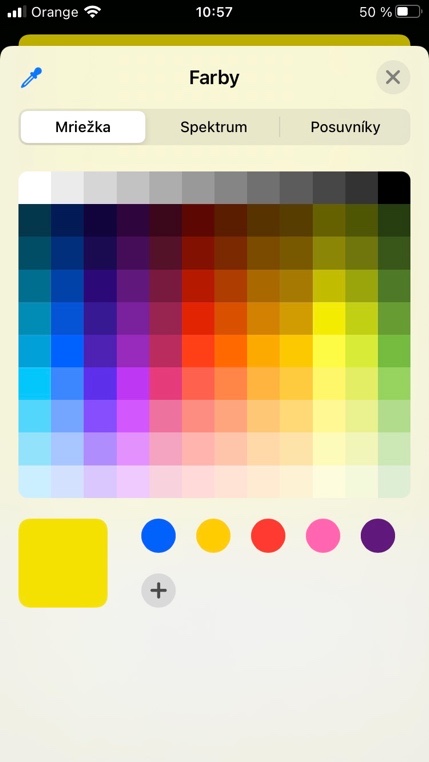
Obrazovka najviac vpravo reprezentuje menu po výbere možnosti „Prechod farieb“. Na ukážke je prechod vypnutý, čo vidíme podľa zeleného tlačidla štart(po spustení sa mení na červené tlačidlo stop). Zobrazené sú štyri tlačidlá s farbami medzi ktorými sa po spustení prechádza. Aktuálne je zvolené tretie tlačidlo, čo nám značí šípka. To znamená, že danú farbu môžeme aktuálne meniť cez otvorenie palety. Na spodku menu je zobrazený „slider“, pomocou ktorého určujeme rýchlosť prechodu.

Aplikácia podporuje aj zobrazenie na šírku. Na obrázku nižšie je zobrazené menu pre výber farby s orientáciou zariadenia na šírku.



Obrázok 17 Ukážka aplikácie: orientácia na šírku

Obrázok dole zobrazuje, ako vyzerá použitá paleta farieb(ColorPicker), ktorá sa využíva pre účel voľby farby. Používa sa pre výber statického osvetlenia a pre prechod medzi viacerými farbami.



Obrázok 18 Paleta farieb ColorPicker

Na obrázku nižšie je zobrazenie časovača, ktorý určuje po akom čase sa aplikácia vypne. Na jednej obrazovke je zobrazenie zapnutého časovača a na druhej vypnutého. Keď je vypnutý, tak je pre používateľa dostupná zmena hodnôt.



Obrázok 19 Ukážka aplikácie: časovač

## Porovnanie s existujúcimi aplikáciami

Pri návrhu a implementácii sme sa inšpirovali aj inými podobnými aplikáciami. Podľa nich sme chceli do našej implementácie zahrnúť tie funkcionality, ktoré boli obľúbené v recenziách a my sme ich považoval za dôležité. Ďalej sme sa pokúšali vyvarovať, všetkým problémom, ktoré sa používateľom nepáčili.

Častý problém v recenziách bol komplikovaný výber farby pre svetlo. Výber farby sme vyriešili pomocou palety, ktorá je podľa nášho názoru veľmi prehľadná a jednoduchá. Taktiež je výhodou, že v histórii zostáva posledných 5 zvolených farieb, aby sa k ním používateľ vedel rýchlo dostať.

Kvôli rýchlosti prístupu k nastaveniu jasu bola táto možnosť pridaná priamo do aplikácie, kde je dostupná nad plávajúcim menu. Dostupná je pri každej voľbe v menu a nastavuje sa pomocou „slidera“.

Jednoduchosť aplikácie sme dosiahli taktiež implementáciou za pomoci jediného pohľadu. Používateľ teda nemá problém pri zorientovaní v nastaveniach. Výber konkrétnych nastavení je jednoducho definovaný podľa samo popisných názvov(farba, časovač a prechod farieb).

Ďalším častým problémom bolo veľké množstvo zobrazovania reklám počas používania aplikácie, pokiaľ si používateľ nezaplatil členstvo. Tento problém sme nemali ako riešiť, keďže naša aplikácia nebude publikovaná v distribučnom obchode App Store, z dôvodu vyžadovanej ročnej platby 99 dolárov za vývojársky účet.

Posledný problém bol veľký výskyt chýb. Aj keď sa skoro nikdy nedá vyhnúť všetkým nedostatkom, tak väčšine chýb sme sa pokúšali vyvarovať pri uváženom naplánovaní implementácie a opakovanom testovaní po pridaní nových funkcionalít.

# Záver

Cieľom tejto práce bolo implementovať aplikáciu „Nočná lampa“ pre mobilné platformy. Počas tvorby sme prešli viacerými krokmi, ktoré nás viedli k jej úspešnému vývoju.

Pred tvorbou aplikácie je na začiatku najvhodnejšie preskúmať trh. Preto sme hľadali aplikácie s podobnou funkcionalitou, kontrolovali recenzie používateľov a nachádzali chyby, ktorým sme sa chceli počas implementácie vyvarovať. Taktiež sme sa inšpirovali vhodne vytvorenými funkciami nočnej lampy.

Ďalším krokom bolo spraviť prieskum vplyvu svetla na spánok človeka. Pomocou neho sme zistili, aké farby majú vplyv na kvalitu a dĺžku spánku. Vieme teda povedať, ktorú farbu svetla zvoliť tesne pred spánkom a ktorú keď chceme aby sa naše telo postupne prebralo.

Na základe poznatkov z podobných aplikácií, prieskumov ohľadom vplyvu svetla a našich návrhov sme si zvolili aké funkcie bude nadobúdať naša aplikácia. Ďalej sme si vytvorili návrh používateľského rozhrania, ktoré bude prehľadné a jednoduché. Po preskúmaní viacerých možností pre vývoj mobilných aplikácií sme si zvolili ako cieľový operačný systém iOS a použili sme programovací jazyk Swift. Implementácia prebiehala v IDE Xcode. Po návrhu rozhrania sme postúpili k tvorbe ikony a úvodnej obrazovky. Posledným krokom pred samotnou implementáciou bolo zvolenie architektúry a vytvorenie vývojového diagramu.

Hlavným bodom pri tvorbe bola samotná implementácia, pri ktorej sme postupovali na základe predošlých krokov a návrhov. Po úspešnom naprogramovaní aplikácie sme postupovali k testovaniu na viacerých mobilných zariadeniach a oprave chýb.

Po porovnaní našej implementácie s ostatnými podobnými aplikáciami a s našimi prvotne navrhnutými funkciami sme prišli k záveru, že sa nám podarilo vytvoriť funkčnú a používateľsky prívetivú mobilnú aplikáciu nočnej lampy.

# Zoznam použitej literatúry

1. *Mobile* *Operating System Market Share Worldwide.* [online].Dostupné na internete: <https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-202001-202012-bar>
2. Bečárová, Katarína. *Biologické LED osvetlenie s cirkadinánnou reguláciou.* [online](24.01.2019). Dostupné na internete: <https://freyaled.com/blog/biologicke-led-osvetlenie-s-cirkadiannou-regulaciou>
3. Maireová, Lenka. *Svetlo v budovách a jeho nevizuálne vnímanie.* [online] (11.01.2018). Dostupné na internete: <https://www.asb.sk/stavebnictvo/technicke-zariadenia-budov/osvetlenie-a-elektroinstalacie/svetlo-v-budovach-a-jeho-nevizualne-vnimanie>
4. Chochlíková, Lucia. *Vplyv osvetlenia na fungovanie biologických hodín.* [online] (16.05.2014) Dostupné na internete: <https://freyaled.com/blog/vplyv-osvetlenia-na-fungovanie-biologickych-hodin>
5. Chochlíková, Lucia. *Kvalita osvetlenia a jej vplyv na naše zdravie.* [online] (02.02.2014). Dostupné na internete: <https://freyaled.com/blog/vplyv-osvetlenia-na-fungovanie-biologickych-hodin>
6. Capretto, Lisa. *The night-light color that can actually help foster sleep*. [online] (12.11.2014). Dostupné na internete: <https://www.huffpost.com/entry/nightlight-color-better-sleep_n_6142098?guccounter=1>
7. Gu, Tianyi. *Newzoo’s Global Mobile Market Report*. [online] (17.09.2019). Dostupné na internete: <https://newzoo.com/insights/articles/newzoos-global-mobile-market-report-insights-into-the-worlds-3-2-billion-smartphone-users-the-devices-they-use-the-mobile-games-they-play/>
8. *Current World Population.* [online]. Dostupné na internete: <https://www.worldometers.info/world-population/>
9. *28 Mobile App Statistics To Know In 2020*. [online] (2020). Dostupné na internete: <https://mindsea.com/app-stats/>
10. *App time will make up nearly 20% of total media time this year*. [online] (11.04.2017). Dostupné na internete: <https://www.emarketer.com/Article/eMarketer-Unveils-New-Estimates-Mobile-App-Usage/1015611>
11. Sims, Gary. *I want to develop Android apps — What languages should I learn?* [online] (10.08.2019). Dostupné na internete: <https://www.androidauthority.com/develop-android-apps-languages-learn-391008/>
12. Scully, Ethan. *Best Android Programming Languages*. [online] (12.07.2020). Dostupné na internete: <https://careerkarma.com/blog/programming-languages-android/>
13. *How much does a Google & Apple Developer Account Cost?* [online]. Dostupné na internete: <https://www.appypie.com/faqs/how-much-does-a-googleapple-developer-account-cost>
14. Royal, Simon. *iPhone OS 1: The Beginning of an Era*. [online] (13.01.2017). Dostupné na internete: <https://lowendmac.com/2017/iphone-os-1-the-beginning-of-an-era/>
15. Zeppeda, Danny. *Revisiting Apple’s iPhone OS 1 ten years later.* [online] (22.04. 2017). Dostupné na internete: <https://www.technobuffalo.com/revisiting-apples-iphone-os-1-ten-years-later>
16. Costello, Sam. *The History of iOS, from Version 1.0 to 14.0.* [online] (18.09.2020). Dostupné na internete: <https://www.lifewire.com/ios-versions-4147730>
17. Illéš, Miroslav. *iOS 14 oficiálne: Kompletný výpis noviniek a inovácií.* [online] (17.09.2020). Dostupné na internete: <https://touchit.sk/ios-14-oficialne-kompletny-vypis-noviniek-a-inovacii/309086>
18. Lacko, Ľuboslav. *Vývoj aplikací pro iOS.* Brno: Computer Press. 2018. ISBN 978-80-251-4947-8, s. 11-14, 43-47, 59-62, 72-73.
19. Biggs, Josh. *Top 7 Programming Languages for iPhone App Development*. [online] (24.07.2020). Dostupné na internete: <https://www.meldium.com/top-7-programming-languages-for-iphone-app-development/>
20. *Introducing Xcode 12.* [online]. Dostupné na internete: <https://developer.apple.com/xcode/>
21. *CodeRunner 4.* [online]. Dostupné na internete: <https://coderunnerapp.com/>
22. *AppCode.* [online]. Dostupné na internete: <https://www.jetbrains.com/objc/>
23. *Membership Details.* [online]. Dostupné na internete: <https://developer.apple.com/programs/whats-included/>
24. Bohon, Cory. *Apple's Swift programming language: Cheat sheet*. [online] (25.09.2020). Dostupné na internete: <https://www.techrepublic.com/article/apples-swift-programming-language-the-smart-persons-guide/>
25. *Welcome to Swift.org*. [online]. Dostupné na internete: <https://swift.org/>
26. *Swift.* [online]. Dostupné na internete: <https://developer.apple.com/swift/>
27. *2015 Developer Survey.* [online]. Dostupné na internete: <https://insights.stackoverflow.com/survey/2015?_ga=2.249939777.638892807.1612531393-661867365.1612531393>
28. *2020 Developer Survey.* [online]. Dostupné na internete: <https://insights.stackoverflow.com/survey/2020#technology-most-loved-dreaded-and-wanted-languages>
29. *Swift 2.0*. [online]. (08.06.2015). Dostupné na internete: <https://developer.apple.com/swift/blog/?id=29>
30. Kremenek, Ted. *Swift 5 Released!* [online]. (25.03.2019). Dostupné na internete: <https://swift.org/blog/swift-5-released/>
31. *Xcode Releases.* [online]. Dostupné na internete: <https://xcodereleases.com/>
32. *Xcode, Tools you’ll love to use.* [online]. Dostupné na internete: <https://developer.apple.com/xcode/ide/>
33. *Xcode, Interface Builder Built-In.* [online]. Dostupné na internete: <https://developer.apple.com/xcode/interface-builder/>
34. *Running Your App in the Simulator or on a Device.* [online]. Dostupné na internete: <https://developer.apple.com/documentation/xcode/running_your_app_in_the_simulator_or_on_a_device>
35. *App Icon.* [online]. Dostupné na internete: <https://developer.apple.com/design/human-interface-guidelines/ios/icons-and-images/app-icon/>
36. *Launch Screen.* [online]. Dostupné na internete: <https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/launch-screen>
37. Desouky, Hassan El. *Save UIColor With UserDefaults in Swift 5.* [online] (22.07.2019). Dostupné na internete: <https://betterprogramming.pub/save-uicolor-with-userdefaults-in-swift-5-951ef1aa88e8>