

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Jernej Janežič

**Razvoj iOS aplikacije za obdelavo in
naročanje fotografij**

DIPLOMSKO DELO
NA VISOKOŠOLSKEM ŠTUDIJU

Mentor: doc. dr. Peter Peer

Ljubljana, 2011

Št. naloge: 00012/2010

Datum: 01.10.2010



Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **JERNEJ JANEŽIČ**

Naslov: **RAZVOJ IOS APLIKACIJE ZA OBDELAVO IN NAROČANJE
FOTOGRAFIJ
DEVELOPMENT OF IOS APPLICATION FOR PHOTOGRAPHS
EDITING AND ORDERING**

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Predstavite razvojno platformo in okolje za iOS. Nato kot izhodišče za določanje funkcionalnosti in uporabniške izkušnje preučite aplikacijo ePhotoLab ter opišite razvoj njene mobilne različice za iOS. Opišite tako razvoj odjemalca kot tudi strežniškega dela.

Mentor:


doc. dr. Peter Peer

Dekan:


prof. dr. Nikolaj Zimic



Rezultati diplomskega dela so intelektualna lastnina Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavljane ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje Fakultete za računalništvo in informatiko ter mentorja.

Besedilo je oblikovano z urejevalnikom besedil $\text{\LaTeX}$.

Namesto te strani **vstavite** original izdane teme diplomskega dela s podpisom mentorja in dekana ter žigom fakultete, ki ga diplomant dvigne v študentskem referatu, preden odda izdelek v vezavo!

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani/-a Jernej Janežič,

z vpisno številko 63060118,

sem avtor/-ica diplomskega dela z naslovom:

Razvoj iOS aplikacije za obdelavo in naročanje fotografij

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom doc. dr. Peter Peer
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 28. 03. 2011

Podpis avtorja/-ice:

Zahvala

Zahvaljujem se mentorju doc. dr. Petru Peeru za vso podporo, pomoč in nasvete.

Prav tako bi se rad zahvalil svojemu dekletu, staršem, sorodnikom in prijateljem za vso podporo in pomoč pri študiju.

Vsem iskrena hvala.

Kazalo

Povzetek	1
Abstract	2
1 Uvod	3
2 Razvojna platforma	5
2.1 iPhone	5
2.2 iPod	6
2.3 iPad	6
2.4 iOS	7
2.5 Strežniški del	8
2.5.1 Apache	8
2.5.2 PHP	8
3 Razvojno okolje za iOS	9
3.1 Xcode	9
3.2 Objective C	10
3.3 Osnovno Xcode okno	11
3.4 Interface Builder	13
3.5 iPhone Simulator	15
3.6 Orodja za nadzor učinkovitosti aplikacije	15
4 Opis razvoja aplikacije	17
4.1 Razvoj iOS aplikacije	17
4.1.1 Dodajanje fotografij	18
4.1.2 Obdelava fotografij	20
4.1.2.1 Dodatna obdelava fotografij	24
4.1.2.2 Obdelava vseh fotografij	26
4.1.3 Naročanje fotografij	28

4.2	Razvoj strežniške aplikacije	31
4.2.1	Sprejemanje naročil	32
5	Analiza učinkovitosti	34
5.1	Možne izboljšave	37
6	Zaključek	38
	Literatura	39

Seznam uporabljenih kratic in simbolov

- MMS – Multimedia Messaging Service; multimedijski sporočilni sistem
- GPS – Global Positioning System; sistem globalnega določanja položaja
- iOS – Operating System; operacijski sistem
- SDK – Software Development Kit; zbirke knjižnic za razvoj aplikacij
- HTTP – Hypertext Transfer Protocol; primarni protokol za brskanje po spletu
- PHP – PHP Hypertext Preprocessor; jezik za razvoj dinamičnih spletnih strani
- ASP – Active Server Page; aktivno kreiranje spletnih strani
- ANSI – American National Standards Institute; Ameriški državni inštitut za standarde
- NIB – NeXT Interface Builder; datoteka izdelana v grafičnem vmesniku
- WLAN – Wireless Local Area Network; brezžično lokalno omrežje
- URL – Uniform Resource Locator; naslov spletnih strani v svetovnem spletu
- IDE – Integrated development environment; integrirano razvojno okolje

Povzetek

V diplomski nalogi je predstavljena iOS aplikacija, ki omogoča obdelavo in naročanje fotografij. Fotografije se lahko posnamejo z vgrajenim fotoaparatom ali pa prenesejo s spleta. Celotna aplikacija temelji na programskem jeziku Objective C. Aplikacija je bila izdelana v okolju Xcode.

V diplomski nalogi so najprej predstavljene posamezne naprave, ki omogočajo izvajanje te aplikacije ter operacijski sistemi, na katerih temeljijo. Vključen je tudi opis razvojnega okolja, v katerem je celotna aplikacija nastala.

Glavni del diplomske naloge opisuje izgradnjo aplikacije. Predstavlja predvsem sestavne dele posameznega pogleda aplikacije in vse funkcionalnosti, ki jih aplikacija omogoča. Nekaj besed je namenjenih strežniškemu delu, ki je pomemben faktor samega delovanja prototipnega izdelka. Na koncu je opisana analiza učinkovitosti aplikacije, ki opredeli zasedenost virov naprave.

Ključne besede:

iOS, aplikacija, fotografije, obdelava fotografij, naročanje fotografij, strežniška aplikacija

Abstract

The diploma thesis presents iOS application for editing and ordering photographs. The photographs can be captured with integrated camera or downloaded from the Internet. The entire application is based on programming language Objective C. Application was developed in Xcode environment.

The diploma starts with the description of devices that allow execution of the application and the operating systems that they are based on. Also a description of software development kit in which the application was designed is included.

The main part describes the development of the application. It mainly presents different views of the application and all functionalities that it allows. A part is devoted to server application which is a significant component of the prototype system. Finally, it describes an analysis of application efficiency, which reveals the usage of device resources.

Key words:

iOS, application, photographs, photographs editing, photographs ordering, server application

Poglavje 1

Uvod

Mnogo uporabnikov danes fotografij ne odnaša več v razvijanje osebno v poslovalnice, saj so tehnologije tako napredovale, da lahko fotografije pošljemo kar preko spleta. Prav tako se v današnjem času uporabljajo digitalne fotografije, ki jih lahko sami razvrščamo in urejamo. Spletne strani različnih fotografskih studiov ponujajo naročanje fotografij bodisi preko spletnega vmesnika ali programa na domačem računalniku. Vse kar je v slednjem primeru potrebno storiti je, da namestimo programsko opremo, v kateri lahko fotografije dodamo in odstranimo, spremenimo velikost, barvo, obliko itd., vnesemo podatke za naročilo in jih preko spleta pošljemo na strežnik fotografskega studia. Po nekaj dneh dobimo obvestilo, da so fotografije izdelane in da nas čakajo v izbrani poslovalnici.

Ker je na trgu vse več pametnih telefonov, ki vsebujejo tudi fotoaparate z ločljivostjo vse tja do 12 milijonov slikovnih elementov, uporabniki fotografirajo kar s telefonom. Težava se pojavi, ko bi določene fotografije radi naročili direktno s telefona, česar pa nam telefon ne omogoča. Vsi trenutno dostopni programi za obdelavo in naročanje fotografij v Sloveniji so namenjeni osebnim računalnikom. Potrebujemo torej aplikacijo za pametne telefone, ki bo to omogočala.

Naša zamisel je bila, da izdelamo aplikacijo, ki bi omogočala uporabnikom pošiljanje fotografij neposredno s telefona in bi zmogla v čim manj korakih poslati fotografije na spletni strežnik studia. Aplikacija naj bi omogočila izbiro poljubnih fotografij iz telefona, čeprav nekatere niso posnete s kamero telefona.

Pametni telefoni danes temeljijo predvsem na dveh operacijskih sistemih,

iOS in Android. Ker v okviru skupine GameTeam na Fakulteti za računalništvo in informatiko že dalj časa razvijamo predvsem za iOS, smo se tudi pri tej diplomski nalogi odločili zanj.

Kot izhodišče za določanje funkcionalnosti in uporabniške izkušnje smo vzeli program ePhotoLab [1], ki teče na osebнем računalniku.

V poglavju 2 bo beseda tekla o razvojni platformi. Predstavljene bodo naprave Apple, ki tečejo na operacijskem sistemu iOS in razvojna orodja na strežniški strani. Poglavje 3 je posvečeno razvojnemu okolju iOS, s katerimi je bil celoten projekt izdelan. Poglavje 4 opisuje posamezne module razvite aplikacije in njenih funkcionalnosti. Poglavje 5 podaja analizo učinkovitosti in možne izboljšave aplikacije. Zaključne misli s ponovitvijo bistvenih sklepov so podane v poglavju 6.

Poglavje 2

Razvojna platforma

2.1 iPhone

iPhone [2] je internetni multimedijski mobilni telefon podjetja Apple. Prvi iPhone je prišel na prodajne police 9. januarja 2007. Telefon ima virtualno tipkovnico in omogoča več sočasnih dotikov.

Ta pametni telefon vsebuje video kamero, fotoaparatus, omogoča pošiljanje besedilnih sporočil, elektronske pošte, multimedijskih sporočil in glasovne pošte. Uporabljamo ga lahko tudi kot prenosni predvajalnik in internetni odjemalec, ki omogoča dostop do elektronske pošte in brskanje po spletu. Dostop do internetnih vsebin omogoča brezžični modul ali pa podatkovna mobilna povezava.

Poleg že naštetih funkcij vsebuje tudi GPS, ki je odličen za prikaz trenutne lokacije, pospeškometer, preko katerega je omogočen nadzor aplikacij na hitrost premika telefona in kompas.

Do sedaj so bile izdane 4 različice telefona. Od modela do modela se specifikacije spreminjajo in izboljšujejo. Zadnji model, ki je bil izdelan, je iPhone 4.

iPhone telefoni uporabljajo operacijski sistem iOS.

Vsebine, ki jih prenesemo iz Applove trgovine ali iz drugih spletnih strani, se na iPhone prenašajo s pomočjo iTunes-a. iPhone povežemo z računalnikom in sprožimo sinhronizacijo.

2.2 iPod

iPod [3] je prenosni predvajalnik, ki so ga razvili v podjetju Apple. Prodajati so ga začeli 23. oktobra 2001.

Trenutno so na tržišču 4 različni modeli iPod-a. Najmanjši je iPod Shuffle, naslednji po velikosti je iPod Nano. Za zapis vsebine oba uporabljata pomnilnik flash. iPod Classic je edini predvajalnik, ki še uporablja notranji trdi disk. Najbolj inovativen in edini, ki se lahko primerja z iPhone-om je iPod Touch, ki ima podobne lastnosti kot iPhone, le da ne omogoča klicev. Na prodajne police je prišel 1. septembra 2010.

Operacijski sistem iOS uporabljajo iPod Shuffle, iPod Classic in iPod Touch.

Vsi iPod-i, z izjemo iPod Shuffla omogočajo predvajanje video vsebin, ki se na iPod naložijo preko računalniške aplikacije iTunes.

2.3 iPad

iPad [4] je tablični računalnik, ki deluje na podlagi operacijskega sistema iOS. Na tržišče je prišel 31. marca 2010.

iPad je bil izdan kot predvajalnik zvočnih in video vsebin. Omogoča branje elektronskih knjig, igranje iger in brskanje po spletu. Na iPad-u lahko poganjamo enake aplikacije kot na iPhone-u in iPod Touch-u. Tudi tukaj za interakcijo uporabljamo zaslon, občutljiv na večprstni dotik.

Za dostop do interneta se uporablja brezžični modul. Nekateri modeli imajo vgrajen tudi podatkovni mobilni modul. Za sinhronizacijo se uporablja program iTunes.

Od 17. marca 2011 se prodaja že druga različica iPad-a.



Slika 2.1: Naprave iPhone, iPad in iPod Touch

2.4 iOS

Slika 2.1 predstavlja vse tri bistvene naprave, ki poganjajo operacijski sistem iOS [5]. iOS je operacijski sistem, ki je bil prvotno razvit za telefone iPhone, razširili pa so ga tako, da ga lahko uporabljajo tudi predvajalniki iPod Touch in iPad. Njegovo jedro temelji na operacijskem sistemu Mac OS X, ki se uporablja na računalnikih Macintosh.

Uporabniški vmesnik iOS-a je bil izdelan posebej za večdotične zaslone. Odziv na vnos uporabnika je takojšen, kar zagotavlja dobro uporabniško izkušnjo. Podpira kopico naprav, tudi pospeškometer, ki na podlagi premikanja in trenjenja naprave omogoča dodatne funkcije in obračanje naprave v tri različne smeri.

iOS vsebuje 4 abstraktne plasti našete od najnižje do najvišje:

- Core OS plast,
- Core Service OS plast,
- multimedijsko plast,
- Cacao plast.

Operacijski sistem potrebuje približno 500 MB prostora.

Najnovejša verzija iOS-a je trenutno verzija 4.3, ki je izšla 9. marca 2011.

Pred iOS 4 je operacijski sistem omogočal izvajanje le ene aplikacije naenkrat, kar pa se je z iOS 4 spremenilo, saj ta sedaj omogoča, da se aplikacije izvajajo v ozadju in tako predstavlja večopravilni operacijski sistem.

Od 13. februarja 2008 lahko večopravilne aplikacije za iOS razvijajo tudi tretje osebe. Za razvoj se uporablja razvojni paket iOS SDK z okoljem Xcode in predvsem programskim jezikom Objective C.

2.5 Strežniški del

Za pravilno delovanje aplikacije je potreben tudi strežnik, na katerem se izvaja aplikacija za sprejem naročil. Na Apache strežniku se izvajajo PHP skripte, ki skrbijo, da se podatki pravilno shranijo in obdelajo.

2.5.1 Apache

Apache [6] je spletni strežnik, ki igra ključno vlogo pri širjenju spleta. Od aprila 1996 je Apache najbolj priljubljen HTTP strežnik. Od oktobra 2007 je bilo na strežniku Apache postavljenih približno 48% vseh spletnih strani.

Strežnik podpira več različnih operacijskih sistemov. Zadnji najbolj stabilen Apache strežnik je 2.2.17, ki je bil izdan 19. oktobra 2010.

2.5.2 PHP

Kratica PHP [7] zaznamuje odprtokodni skriptni jezik za kreiranje dinamičnih spletnih strani. Lahko ga primerjamo z Microsoftovim jezikom ASP, VBScript in JScript.

PHP omogoča programerjem razvoj zahtevnih spletnih strani brez potrebe po dolgotrajnem učenju.

Trenutno sta v uporabi dve večji različici, 5.3.x in 5.2.x. PHP podpira več različnih operacijskih sistemov.

Zadnja stabilna različica je 5.3.2, izdana 4. marca 2010.

Poglavje 3

Razvojno okolje za iOS

Z namestitvijo iOS SDK dobimo tudi razvojno okolje Xcode. V razvojnem okolju je zajetih veliko orodij, ki olajšujejo izdelavo aplikacij: orodja za nadzor porabe sistemskih virov naprave, orodje za gradnjo uporabniških vmesnikov, simulator, urejevalnik programske kode, razhroščevalnik, hitra pomoč, ki vsebuje primere uporabe programske kode, itd.

3.1 Xcode

Xcode [8] je brezplačna zbirka orodij za razvoj programske opreme na operacijskem sistemu Mac OS X, ki upravlja z vsemi datotekami v projektu. Xcode je tesno povezan s komponento za gradnjo uporabniških vmesnikov (angl. Interface Builder).

V Xcode paketu so vključeni prevajalniki, povezovalniki in pripomočki, ki so potrebni za izgradnjo knjižnic in aplikacij. Za brezhibno delovanje je potrebna popolna namestitev programa, čeprav želimo uporabljati le del paketa.

Xcode podpira različne programske jezike, kot so C, C++, Fortran, Java, AppleScript, Python in Ruby. Xcode uporablja tudi zmogljiv razhroščevalnik, s katerim lahko razvijalec razhroščuje svojo aplikacijo in tako preveri njeno pravilno delovanje.

Najnovejša različica Xcode 4 je bila izdana 14. oktobra 2010. Različico si lahko vsak uporabnik po uspešni registraciji prenese na svoj računalnik, saj le-ta ni vključena v sam operacijski sistem.

Najnovejša različica Xcode vsebuje izboljššan Interface Builder, ki omogoča izvajanje aplikacije le v enem oknu, kar pomeni, da se za to orodje ne odpre dodatno okno. Dodan je tudi pomočnik (angl. Assistant), ki poenostavlja razvoj in omogoča razvijalcu aplikacije lažji pregled povezav med posameznimi knjižnicami. Izboljšan je tudi črkovalnik (angl. Fix-it). Ta skrbi za napake, ki se pojavijo že med samim pisanjem programske kode. Najpomembnejše pa je, da je izdelan popolnoma nov razhroščevalnik (angl. Debugger), ki je sedaj trikrat hitrejši in dvakrat učinkovitejši [9].

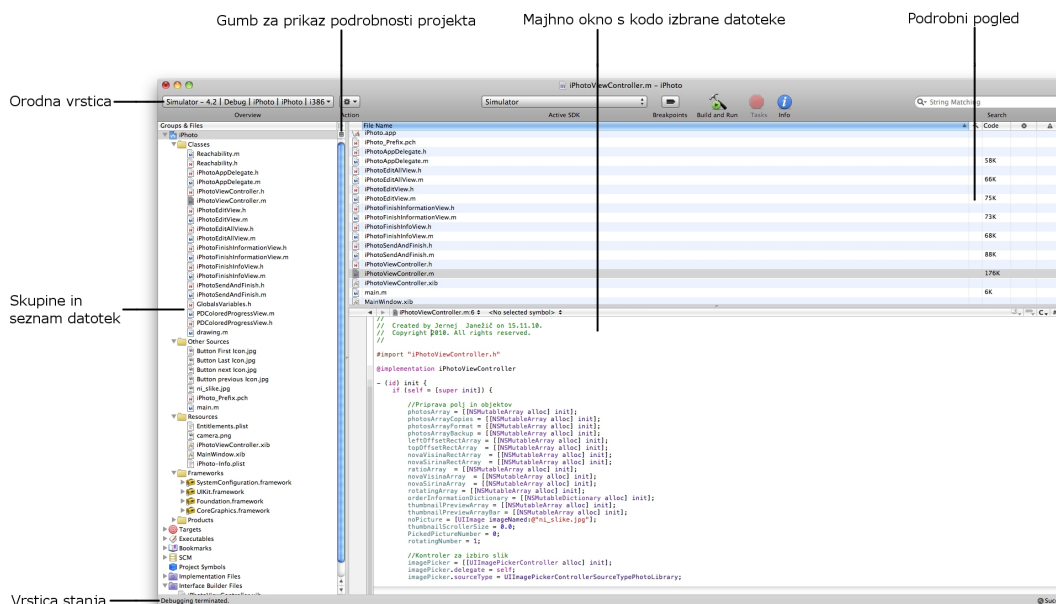
3.2 Objective C

Objective C [10] je objektno usmerjen programski jezik. Opredeljen je kot zmogljiv nabor razširitve standardnega jezika ANSI C. Objective C je zasnovan tako, da jeziku C omogoča popolno objektno programiranje na preprost in enostaven način.

Objective C sta v zgodnjih osemdesetih letih izdelala Brad Cox in Tom Love v svojem podjetju Productivity Products International (Stepstone), v katerem sta zaslužila dovolj, da sta razvila nekaj različic Objective C. Brad Cox je potem izdal glavni opis delovanja programskega jezika v knjigi Object-Oriented Programming, An Evolutionary Approach. Ko je Steve Jobs leta 1988 zapustil Apple Inc. in ustanovil podjetje NeXT, je kupil licenco Objective C podjetja Stepstone in izdelal lasten Objective C razhroščevalnik in knjižnice, na katerih je temeljil uporabniški vmesnik NeXTSTEP. Ko je leta 1996 Apple prevzel podjetje NeXT, je uporabil OpenStep v novem operacijskem sistemu Mac OS X. OpenStep je vseboval Objective C, NeXT Objective C razvojna orodja in okolje IDE [11]. Programski jezik temelji na programskem jeziku C.

Programski jezik se je pojavil leta 1986, zadnja stabilna različica je 2.1.

3.3 Osnovno Xcode okno



Slika 3.1: Primer osnovnega okna Xcode.

Osnovno okno (slika 3.1):

- **Skupine in sezname datotek**, ki vsebujejo oris vsebine projekta. Z organizacijo map in datotek se olajša iskanje in delo.

Projekt vsebuje več različnih podmap. Največkrat uporabljeni sta:

- **Classes**, v kateri se nahajajo vsi razredi naše aplikacije in
- **Resources**, v katerem se shranjujejo ostali viri, kot so npr. slike, ki se uporabljajo v projektu.

V projektu se nahajajo tudi datoteke NIB, ki se ustvarijo z orodjem za izdelavo grafičnega vmesnika aplikacije.

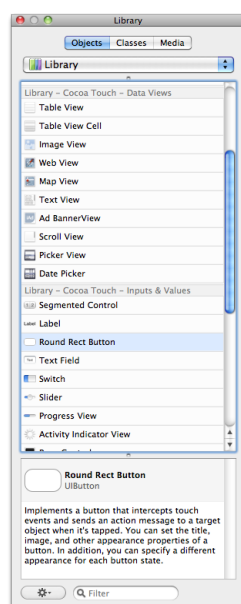
- **Gumb za prikaz podrobnosti projekta** omogoča razvijalcu prikaz podrobnosti projekta. Ob pritisku nanj se prikažejo ali skrijejo skupine in sezname datotek.

- **Podrobni pregled** prikaže vse datoteke, ki se nahajajo v posamezni skupini datotek. S pomočjo podrobnega pregleda lahko razvijalec brska po datotekah, jih razvršča po različnih skupinah ali išče s pomočjo polja za iskanje in tako hitreje najde svoj rezultat.
- **Vrstica stanja** nam prikazuje sporočila in stanje projekta. Xcode v tej vrstici prikazuje kazalnik napredka in različne napake ali opozorila, ki nastanejo med razvijanjem.
- **Orodna vrstica** omogoča najhitrejši dostop do funkcij Xcode programa. Sestavljena je iz več delov:
 - Predogled (angl. Overview) omogoča izbiro načina zagona projekta. Izbere se lahko simulator ali naprava, razhroščevanje ali izraza.
 - Akcija (angl. Action) omogoča izvedbo operacije na trenutno izbranem predmetu v projektnem oknu.
 - Izdelaj in zaženi (angl. Build and Run) omogoča, da Xcode projekt izdela in ga zažene na simulatorju ali napravi.
 - Prelomne točke (angl. Breakpoints) omogočajo razvijalcu nadzor na razhroščevanjem, saj se v teh točkah izvajanje aplikacije zaustavi, razvijalec pa dobi nadzor nad nadaljnjim izvajanjem.
 - Opravila (angl. Tasks) omogočajo, da razvijalec zaključi trenutno izvajanje.
 - Gumb za informacije (angl. Information) prikaže okno z informacijami za trenutno izbrano datoteko.
 - Iskalno polje (angl. Search field) je namenjeno lažjemu iskanju po programski kodi in datotekah.
- **Majhno okno s kodo izbrane datoteke** ponudi razvijalcu vpogled v kodo, v katerem jo lahko tudi ureja.

3.4 Interface Builder

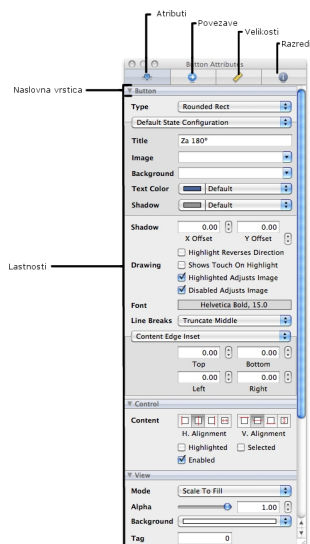
Interface Builder je Applov grafični urejevalnik za izdelavo grafičnega vmesnika (angl. Graphic User Interface) na napravi. Vsak projekt ima zbirko grafičnih vmesnikov. Največ teh objektov ustreza elementom na zaslonu naprave v obliki oken, pogledov, kontrolnikov in menijev.

V Interface Builder-ju vsak projekt vsebuje tudi okno s knjižnico (angl. Library window). Knjižnica (slika 3.2) vsebuje gradnike, ki se lahko dodajajo v projekt. Slika prikazuje Cocoa zbirko, ki vsebuje gradnik gumb. Za dodajanje gumba v projekt se objekt preprosto povleče na projekt.



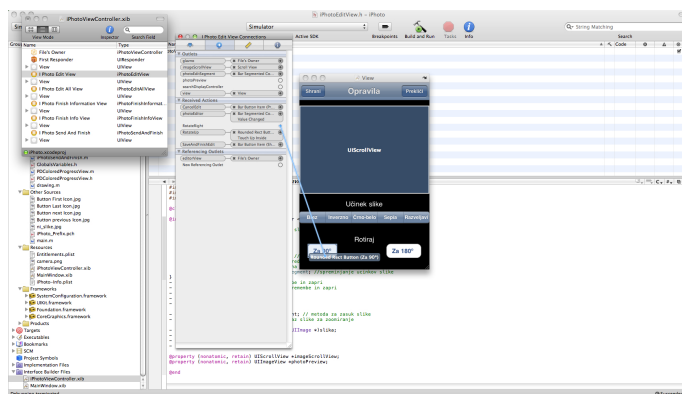
Slika 3.2: Knjižnica gradnikov.

Za spreminjanje lastnosti gradnikov se uporablja Inspector znotraj Interface Builder-ja (slika 3.3). Ta omogoča spreminjanje in prilagajanje gradnikov. Vsebuje več različnih zavihkov, v katerih spreminjamo posamezne lastnosti.



Slika 3.3: Primer okna Inspector za nastavitve lastnosti gradnika.

Za pravilno delovanje gradnikov znotraj projekta so potrebne povezave (slika 3.4). Posamezen gradnik je potrebno povezati s programsko kodo, ki potem izvede posamezno akcijo. Povezava se ustvari tako, da se označi ime razreda in objekt.



Slika 3.4: Povezovanje gradnikov in programske kode.

3.5 iPhone Simulator

Če razvijalec nima naprave, kjer bi preizkusil aplikacijo, je iPhone Simulator (slika 3.5) nepogrešljivo orodje. Pri izvajanju posnema napravo, tako da simulator uporablja vire razvojnega računalnika. iPhone Simulator je zelo priročen, saj prikaže dober približek dejanskega stanja aplikacije.

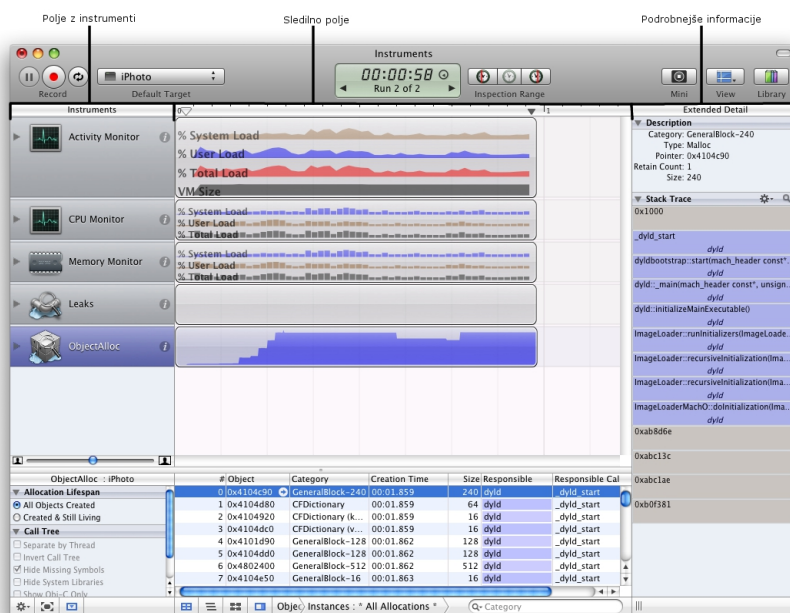


Slika 3.5: Simulator znotraj Xcode.

3.6 Orodja za nadzor učinkovitosti aplikacije

Z orodji za nadzor učinkovitosti aplikacije (slika 3.6) lahko pridobimo veliko pomembnih informacij o delovanju in obnašanju enega ali več procesov. Orodja so zasnovana tako, da zajamejo podatke o procesu za določen čas. Vsak instrument zajame podatke, kot so npr. dostop do podatkov, poraba notranjega spomina itd. in jih prikaže v obliki grafa. Orodja lahko konfiguriramo tako, da za določen proces zajamejo več različnih informacij.

Orodja omogočajo tudi izdelavo samostojnega instrumenta z uporabo instrumentalnega grafičnega vmesnika.



Slika 3.6: Okno Instruments za nadzor učinkovitosti aplikacije.

Trije glavni segmenti na sliki 3.6 so:

- Polje z instrumenti (angl. Instruments) zajema vse instrumente, ki jih želimo uporabiti. V polje lahko dodamo ali odstranimo različne instrumente.
- Sledilno polje (angl. Track) grafično prikaže zbrane podatke določenega instrumenta. Vsak instrument ima svojo vrstico, kamor izpisuje graf. Vse informacije so zgolj za branje in se jih ne da spreminjati.
- Podrobnejše informacije (angl. Extended Detail) prikazujejo informacije o izbranem stolpcu grafa.

Poglavje 4

Opis razvoja aplikacije

Na podlagi uporabniških zahtev se sestavi funkcionalna specifikacija, ki opisuje delovanje aplikacije z uporabniškega vidika. Funkcionalna specifikacija je kasneje tudi osnova za zaključek projekta.

Z vidika uporabnika mora biti aplikacija čim bolj enostavna, vendar kljub temu funkcionalna. V ta namen je bila potrebna izpopolnitev najpomembnejših funkcij, kot so npr. spreminjanje barvnih shem fotografije (črno-belo, sephia, inverzno), obračanje fotografije po širini ali dolžini itd.

Poleg vseh osnovnih funkcionalnosti pa smo hoteli implementirati tudi dodatne funkcije, ki bi čim bolj izkoristile funkcionalnosti naprave iPhone, iPod Touch in sedaj tudi iPad. Tako smo v načrt za implementacijo dodali večdotično interakcijo, ki omogoča določanje vsebine fotografije z dvema prstoma.

4.1 Razvoj iOS aplikacije

Razvoj iOS aplikacije se je pričel z izbiro fotografij iz zbirke fotografij (angl. Photo Library), ki se dodajajo v polje fotografij. Polje se ob zaključku izbire shrani in je pripravljeno za obdelavo. V urejevalnik fotografij so vključene različne funkcije, ki jih podpirajo računalniške aplikacije za delo s fotografijami in naročanje le-teh. Ko se vse fotografije uredijo, se odpre nov pogled, v katerega se vnesejo podatki, ki se skupaj s fotografijami pošljejo na strežnik fotografskega studia.

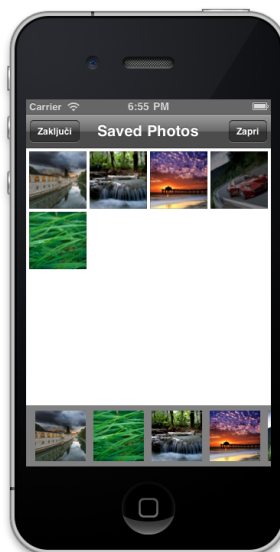
4.1.1 Dodajanje fotografij

Dodajanje fotografij je ena izmed prvih funkcij, ki jih aplikacija ponudi. Po zagonu aplikacije se vse funkcije razen te onemogočijo (slika 4.1).



Slika 4.1: Prvi pogled ob vstopu v aplikacijo.

Za dodajanje fotografij je najpomembnejši gumb z ikono , ki se nahaja v zgornjem levem kotu (slika 4.1). Ob pritisku na gumb se zamenja pogled (angl. View), ki prikaže zbirko fotografij (angl. Photo Library) (slika 4.2).



Slika 4.2: Zbirka fotografij.

Pogled (slika 4.2) vsebuje gumb Zaključ, ki ob izbiri shrani vse fotografije v polje in zapre pogled z zbirko fotografij. Fotografije se shranjujejo v polje, kar omogoča enostavnejše premikanje in iskanje. V ozadju se shranjuje tudi varnostna kopija izbranih fotografij, ki omogoča poznejšo razveljavitev. Vsako polje ima svojo funkcijo shranjevanja podatkov.

V polja se shranjujejo naslednji podatki:

- fotografije,
- varnostne kopije fotografij,
- izbrani formati fotografij,
- število kopij fotografij,
- širina okvirja, ki se izračuna na podlagi izbranega formata,
- višina okvirja, ki se izračuna na podlagi izbranega formata.

Gumb Zapri zapre pogled na zbirko fotografij in polj s podatki ne spreminja. Pod gumbi se nahaja polje fotografij, kjer so vse fotografije. Ob izbiri

posamezne fotografije, se le-ta pokaže pod poljem fotografij. Vse izbrane fotografije so prikazane kot sličice (angl. Thumbnails). Ker obstaja možnost izbire več fotografij, so le te prikazane na drsniku, ki ga lahko premikamo levo ali desno, kjer si izbrane fotografije tudi ogledamo.

4.1.2 Obdelava fotografij

Obdelava fotografij (slika 4.3) vsebuje največ funkcionalnosti. Ob zagonu aplikacije se ob prisotnosti brezžične internetne povezave podatki o cenah fotografij vedno samodejno posodobijo. V nasprotnem primeru nas aplikacija vpraša, če želimo posodobiti cenik. Vse cene se zapišejo v programski slovar (angl. Dictionary), ki je sestavljen iz ključa in vrednosti. Ključ je format fotografije, vrednost pa je cena posameznega formata.



Slika 4.3: Pogled za obdelavo fotografij.

Spodnja vrstica na sliki 4.3 prikazuje vsoto vseh fotografij. Ta vrstica se spremeni ob dodajanju fotografij, spreminjanju formata ali povečanju kopij fotografij.

Metoda za izračun cene fotografij izbranih formatov:

```
– (void) sestevVsehCen {
    float cena = 0;
    for (int i=0; i < [photosArray count]; i++){
        int x = [[photosArrayCopies objectAtIndex:i] intValue];
        NSString *z = [photosArrayFormat objectAtIndex:i];
        float y = [[photosDictionaryPrice objectForKey:z] floatValue];
        cena = cena + (x*y);
    }
    CountPrice = cena;
    photoCounterPrice.text = [NSString stringWithFormat:@"Cena: %.2f
        EUR", cena];
}
```

Metoda prebere iz polj format, ki se navezuje na fotografijo za katero je določena cena. Vse cene sešteje skupaj in izračuna skupno vsoto.

Izbirni gumbi imajo zapisane formate fotografij in so vezani na drsnik (angl. Slider). Ob različnih izbirah se izbrana vrednost zapiše v polje, na podlagi izbranega formata se izriše okvir in izračuna nova cena. Gumbi za premik na levi omogočajo pregled po različnih fotografijah. Vsak gumb ima funkcijo, ki mu omogoča povečanje ali zmanjšanje indeksa fotografije v polju. Premik omogoča tudi drsnik, s katerim se premikamo samo s potegom prsta. Ob pritisku gumba se izvede funkcija, ki povečuje oziroma zmanjšuje število indeksa in nato pokliče funkcijo drsnika, ki celotno operacijo izvede. Enak postopek se pojavi tudi pri količini fotografij (gumbi za premik na desni). Kadar spremenimo količino fotografij, vpliva sprememba tudi na spodnjo orodno vrstico, kjer sta zapisani število fotografij in cena.

Spreminjanje količine izvede naslednja metoda:

```
– (void) sestevVsehSlik {
    int vsota = 0;
    for (int i=0; i < [photosArrayCopies count]; i++) {
        int x = [[photosArrayCopies objectAtIndex:i] intValue];
        vsota = vsota+x;
    }
    PictureNumber = vsota;
    photoCounter.text = [NSString stringWithFormat:@"Slik: %i",
        vsota];
}
```

Merilnik kakovosti [12] fotografije je prikazan na levi strani takoj pod fotografijo. Kakovost fotografije se izračuna na podlagi velikosti fotografije in velikosti okvirja. Spodnja koda prikazuje izračun barve merilnika:

```

- (void) spremeniKvaliteto:(CGFloat)progressBar {
    UIColor *barva;
    if (progressBar > 1.0)
        progressBar = 1.0;
    if (progressBar >= 0.7) {
        barva = [UIColor greenColor];
    } else if (progressBar < 0.7 && progressBar >= 0.58){
        barva = [UIColor yellowColor];
    } else if (progressBar < 0.58) {
        barva = [UIColor redColor];
    }
    PDColoredProgressView *pView = [[PDColoredProgressView alloc]
        initWithProgressViewStyle:UIProgressViewStyleDefault];
    CGRect frame = kvaliteta.frame;
    pView.frame = frame;
    pView.progress = progressBar;
    [pView setTintColor:barva];
    [progressBarView addSubview:pView];
    [pView release];
}

```

Merilnik je sestavljen iz napredivalne vrstice (angl. Progress Bar). Vrstica se napolni na podlagi rezultata, ki se pretvori v decimalno število. Velikost se giblje od 0 do 1. Slabo kvaliteto fotografije za tisk signaliziramo z rdečo barvo, dobro kvaliteto z zeleno, srednjo pa z rumeno.

Obračanje okvirja se izvede tako, da obrne razmerje dolžine in širine fotografije. Najprej preveri celotno dolžino in širino fotografije. Iz teh dveh vrednosti izračuna razmerje, s katerim izračuna dolžino nove stranice in ponovno nariše okvir. Vse skupaj se zapiše v polje.

Predogled prikazuje fotografijo, ki jo uporabnik izbere. Prikazovalnik ima vgrajene tri funkcije. Prva omogoča, da se z dvema prstoma spremeni velikost okvirja. Prsta preprosto povlečemo narazen [13], če želimo okvir povečati in ju povlečemo skupaj če želimo okvir pomanjšati. Ob vsakem premiku se izračunajo nove vrednosti okvirja, ki se spreminjajo v poljih. Izračun se izvede v naslednji kodi:

```

- (UIImage*)modifiedRectWithImage:(UIImage*)uiImage :(NSInteger)
    levo :(NSInteger)gor{
    ...

    novaSirinaRect = novaSirinaRect + velikostRectSirina;
    if (novaSirinaRect >= novaSirina)
        novaSirinaRect = novaSirina;
    if (novaSirinaRect < 0)
        novaSirinaRect = 0;

    novaVisinaRect = novaVisinaRect + velikostRectVisina;
    if (novaVisinaRect >= novaVisina)
        novaVisinaRect = novaVisina;
    if (novaVisinaRect < 0)
        novaVisinaRect = 0;

    leftOffsetRect = leftOffsetRect + levo;
    if (leftOffsetRect <= 0)
        leftOffsetRect = 0;
    if ((leftOffsetRect + novaSirinaRect) >= uiImage.size.width)
        leftOffsetRect = uiImage.size.width - novaSirinaRect;

    topOffsetRect = topOffsetRect + gor;
    if (topOffsetRect <= 0)
        topOffsetRect = 0;
    if ((topOffsetRect + novaVisinaRect) >= uiImage.size.height)
        topOffsetRect = uiImage.size.height - novaVisinaRect;
    ...
}

```

Druga funkcionalnost omogoča premik okvirja v vse smeri. S tem določimo pozicijo okvirja za izrez fotografije. Ob premiku se izračunajo vrednosti odmika od levega in desnega roba in odmik od spodnjega in zgornjega roba. Te vrednosti se zapisuje v polja zato, da aplikacija na koncu izreže fotografijo takšne velikosti, kot želi uporabnik. Gumb za brisanje je namenjen brisanju trenutne prikazane fotografije. Ob pritisku se izvede koda, ki izbriše fotografijo iz polja fotografij in iz polja varnostnih kopij, izbriše pa tudi vse vrednosti, ki se navezujejo na to fotografijo. Vsaka fotografija je shranjena v polju z indeksom in ker so vse lastnosti shranjene z enakim indeksom tudi v drugih poljih, je brisanje podatkov enostavno. Če izbrišemo vse fotografije, se izpraznijo vsa polja in prikaže se fotografija z vsebino "Ni slike" (slika 4.1).

4.1.2.1 Dodatna obdelava fotografij

Tretja funkcija omogoča, da se ob dveh hitrih klikih oziroma dotikih (angl. Tap) spremeni pogled (slika 4.4), ki nam omogoča dodatno obdelavo. Pritisk na gumb z ikono  (uredi fotografijo), ki je ekvivalent dvema klikoma, nam prav tako spremeni pogled in prikaže nove funkcionalnosti, ki jih aplikacija ponuja.



Slika 4.4: Pogled za dodatno obdelavo fotografij.

Rotacijska gumba nam omogočata, da fotografijo obrnemo za toliko stopinj, kot je zapisano na gumbu.

Ob izbiri gumba za rotacijo [14], se izvede naslednja metoda:

```

- (void) rotating:(UIImageOrientation) orient {

    UIImage *changePicture = [[UIImage alloc] init];
    if (changedPicture == nil)
        changePicture = [photosArray objectAtIndex:locationPicture];
    else
        changePicture = changedPicture;

    CGImageRef imgRef = changePicture.CGImage;

    CGFloat width = CGImageGetWidth(imgRef);
    CGFloat height = CGImageGetHeight(imgRef);

    CGAffineTransform transform = CGAffineTransformIdentity;
    CGRect bounds = CGRectMake(0, 0, width, height);

    CGFloat scaleRatio = bounds.size.width / width;
    CGSize imageSize = CGSizeMake(CGImageGetWidth(imgRef),
        CGImageGetHeight(imgRef));
    CGFloat boundHeight;

    switch (orient) {
        ...
        case UIImageOrientationRight:
            boundHeight = bounds.size.height;
            bounds.size.height = bounds.size.width;
            bounds.size.width = boundHeight;
            transform = CGAffineTransformMakeTranslation(imageSize.height, 0.0);
            transform = CGAffineTransformRotate(transform, M_PI / 2.0);
            break;
    }
    ...
}

```

Koda prikazuje primer za rotacijo fotografije v desno.

S pomočjo učinkov lahko spremenimo barvno shemo fotografije v inverzno, črno-belo ali sephia. Vse spremembe lahko razveljavimo s pritiskom na gumb Razveljavi. Ob pritisku na ta gumb se izvede koda, ki prebere trenutno lokacijo fotografije iz polja fotografij, kjer so shranjene varnostne kopije oziroma originali. Ta pogled ima vključeno funkcijo povečave preko večdotične interakcije, ki omogoča ogled detajlov fotografije.

Pogled je sestavljen iz dveh komponent:

- UIImageView komponenta SDK omogoča, da vstavimo fotografijo, ki jo aplikacija prikaže na zaslonu naprave. Ta Pogled vsebuje spremenjeno fotografijo glede na izbrano funkcijo,
- UIScrollView komponenta SDK omogoča, da fotografije povečamo do štirikratne velikosti in ogled detajlov fotografije.

Naslovna vrstica pogleda ima dva gumba. Ob pritisku na gumb Shrani se shranijo vse spremembe, ki so bile narejene v tem pogledu. Gumb Shrani izvede kodo, ki spremenjeno fotografijo zamenja v polju fotografij s spremenjeno fotografijo. Vse narejene spremembe lahko tudi prekličemo s pritiskom na gumb Prekliči.

4.1.2.2 Obdelava vseh fotografij

Ker uporabniki pogosto želijo imeti vse fotografije enakega formata, je ta pogled namenjen tudi spreminjanju te nastavitve (slika 4.5). Format določimo tako, da pritisnemo na izbirni gumb z zapisanim formatom, ki se nahaja na drsniku.

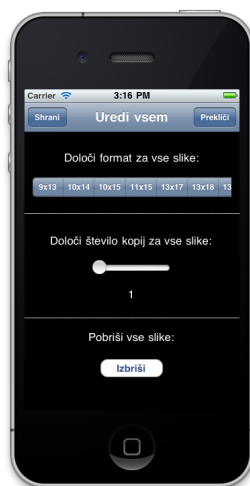
Izbrani format se zapiše v polje formatov s "for" zanko:

```
for (int i=0; i < [photosArrayFormat count]; i++) {
    [photosArrayFormat replaceObjectAtIndex:i withObject:[
        photosDictionarySegmentSize objectForKey:imageFormat]];
}
```

Drsnik za spreminjanje količine fotografij vpiše v polja za količino fotografij ustrezno spremenjeno število. Vse spremembe delujejo v vseh poljih.

Izbrano število kopij se zapiše v polja s "for" zanko:

```
for (int i=0; i < [photosArrayCopies count]; i++) {
    [photosArrayCopies replaceObjectAtIndex:i withObject:
        imageSliderCopiesValue];
}
```



Slika 4.5: Pogled za obdelavo vseh fotografij.

Gumb Izbriši izvede operacijo, ki izprazni vsa polja in zapre trenutni pogled.

Izbris vseh polj se izvede v naslednji kodi:

```
- (void) emptyArrays {  
    [photosArray removeAllObjects];  
    [photosArrayCopies removeAllObjects];  
    [photosArrayFormat removeAllObjects];  
    [photosArrayBackup removeAllObjects];  
    [thumbnailPreviewArray removeAllObjects];  
    [thumbnailPreviewArrayBar removeAllObjects];  
    for (UIView *subview in [thumbnailScroller subviews]) {  
        [subview removeFromSuperview];  
    }  
    thumbnailScrollerSize = 0.0;  
    [ratioArray removeAllObjects];  
    [leftOffsetRectArray removeAllObjects];  
    [topOffsetRectArray removeAllObjects];  
    [novaVisinaRectArray removeAllObjects];  
    [novaSirinaRectArray removeAllObjects];  
    [novaVisinaArray removeAllObjects];  
    [novaSirinaArray removeAllObjects];  
    [rotatingArray removeAllObjects];  
}
```

Uporabljena je metoda `removeAllObjects` (odstrani vse objekte), ki izprazni seznam naenkrat in ima enako vlogo, kot če bi se sprehodili skozi celotno polje z zanko "for" in posamezen zapis izbrisali.

Vse spremembe se uveljavijo šele, ko uporabnik pritisne na gumb Shrani. S pritiskom na gumb Prekliči spremembe zavržemo.

4.1.3 Naročanje fotografij

Za konec je potrebno fotografije poslati še na strežnik fotografskega studia. Naročanje fotografij je namenjeno zajemu vseh podatkov o uporabniku, studiu in poslovalnici za prevzem fotografij. Pogled na sliki 4.3 po pritisku na gumb Pošlji zahteva vnos podatkov o uporabniku.

Če uporabnik pritisne na gumb Prevzem preden vpiše osebne podatke, se pojavi obvestilo "Prosimo izpolnite vse podatke!" (slika 4.6).



Slika 4.6: Napaka ob neizpolnjenih poljih.

Opozorilo se prikaže s pomočjo komponente `UIAlertView`.



Slika 4.7: Pogled za vnos podatkov.

S pritiskom na gumb z ikono  (slika 4.7), se odpre izbirno okno, ki nam omogoča priklic predhodno vnesenih podatkov. Podatki se zapišejo v datoteko vedno, kadar uporabnik pravilno vnese podatke in pritisne na gumb Prevezem. Ob naslednjem pritisku na gumb , se v okenca vpišejo shranjeni podatki.



Slika 4.8: Pogled za izbiro poslovalnice za prevzem.

Po pravilnem vnosu se pojavi nov pogled (slika 4.8), kjer izberemo način prevzema poslanih fotografij. Izbiramo lahko med Osebnim prevzemom in Dostavo. Osebni prevzem ponuja več poslovalnic, zato so vse poslovalnice vnešene v komponento NSMutableArray, ki se navezuje na UIPickerView.

Spodnja koda prikazuje del vpisanih poslovalnic v NSMutableArray:

```
- (void) viewDidLoad {
    [super viewDidLoad];
    LocationOfPictures.hidden = LocationOfPicturesTitle.hidden =
        YES;

    osebniPrevzemArrayPicker = [[NSMutableArray alloc] init];
    [osebniPrevzemArrayPicker addObject:@"Avit Center, Maribor"];
    [osebniPrevzemArrayPicker addObject:@"Comtron d.o.o., Maribor"];
    ...
}
```

Po seznamu poslovalnic uporabnik brska tako, da s prstom povleče navzgor ali navzdol, da se kolo zavrti. Ko uporabnik izbere željeno poslovalnico, se pri izbiri gumba Dostava naročene fotografije dostavijo na naslov, podan nad kolesom.

Po izbiri uporabnik pritisne na gumb Zaključ, ki pokaže končni pogled naročila (slika 4.9).

Na vrhu se ponovno pojavi ikona za informacije ⓘ. Sedaj omogoča uporabniku, da svoje delo shrani v napravo in nadaljuje pozneje, ker morda trenutno nima na voljo brezžične internetne povezave. Aplikacija obreže vse fotografije glede na izbran okvir, ki jih je uporabnik določil. V tekstovno datoteko prav tako zapiše formate fotografij, količino fotografij in vse osebne podatke uporabnika.

Zadnji pogled (slika 4.9) prikaže vnešene osebne podatke. Na podlagi vseh izbir se izračuna končna cena naročenih izdelkov. Če se uporabnik strinja z vsemi podatki, pritisne gumb Pošlji in počaka, da se izpiše sporočilo "Fotografije so bile uspešno naročene".



Slika 4.9: Pogled za pregled vseh podatkov.

4.2 Razvoj strežniške aplikacije

Poslane fotografije in podatki o uporabniku se morajo shraniti na spletnem strežniku fotografskega studia. Ker želimo doseči čim boljšo odzivnost in hiter sprejem podatkov, je potrebno izdelati čim bolj enostavno, vendar funkcionalno strežniško aplikacijo.

Razvita strežniška aplikacija služi kot dokaz koncepta in ni bila bistvo te diplomske naloge.

4.2.1 Sprejemanje naročil

Za sprejemanje naročil je najenostavnejši implementacijski način izdelave spletnih skript. V našem primeru smo izbrali skriptni jezik PHP. Na strežniku se nahajata dve skripti, ki skrbita za pravilen sprejem podatkov.

Skripta `information.php`:

```
<?php

$podatki = $_POST[ 'podatki' ];
$naziv = $_POST[ 'naziv' ];
$naslov = $_POST[ 'naslov' ];
$telefon = $_POST[ 'telefon' ];
$mail = $_POST[ 'mail' ];
$prevzem = $_POST[ 'prevzem' ];
$lokacija = $_POST[ 'lokacija' ];
$index = $_POST[ 'index' ];
$cdrw = $_POST[ 'cdrw' ];
$top = $_POST[ 'top' ];
$sporocilo = $_POST[ 'message' ];

$information = "$podatki \r\n\r\n";
$name = "$naziv \r\n";
$address = "$naslov \r\n";
$phone = "$telefon \r\n";
$email = "$mail \r\n\r\n";
$taking = "$prevzem \r\n";
$location = "$lokacija \r\n\r\n";
$createIndex = "$index \r\n";
$createcdrw = "$cdrw \r\n\r\n";
$images = "Slike: \r\n";
$topInfo = "$top \r\n\r\n";
$message = "$sporocilo \r\n\r\n";

$File = "Podatki.txt";
$FileCreate = fopen($File, 'w') or die("can't open file");

fwrite($FileCreate, $information . $name . $address . $phone .
    $email . $taking . $location . $createIndex . $createcdrw .
    $images . $topInfo . $message);

Fclose($FileCreate);
?>
```

Skripta `information.php` sprejme podatke z zahtevo POST. Podatki se pošiljajo preko URL vrstice, kjer se posamezni podatki razlikujejo po imenu. PHP skripta potem izlušči ustrezen podatek. Vsak podatek se obdela in zapiše v datoteko.

Druga skripta, ki se izvaja na strežniku, je `upload.php`. Ta skripta skrbi za pravilen sprejem vseh fotografij. Fotografije se prav tako pošiljajo preko POST zahteve.

Skripta `upload.php` vsebuje naslednje ukaze:

```
<?php

$uploadDir = './';          //Upload v enak direktorij

$file = basename($_FILES['userfile']['name']);

$uploadFile = $file;

$newName = $uploadDir . $uploadFile;

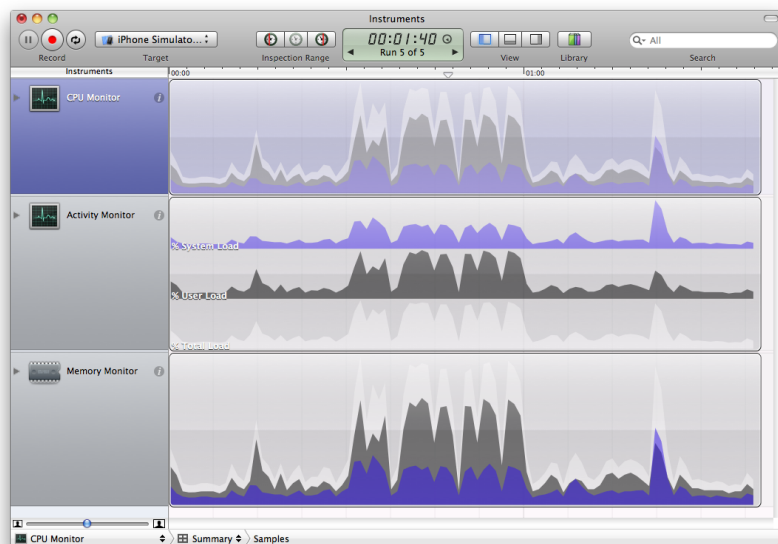
if (move_uploaded_file($_FILES['userfile']['tmp_name'], $newName))
{
    $postsize = ini_get('post_max_size');
    $canupload = ini_get('file_uploads');
    $tempdir = ini_get('upload_tmp_dir');
    $maxsize = ini_get('upload_max_filesize');
}
?>
```

Ko se podatki prejetih fotografij razberejo iz URL vrstice, se shranijo na spletni strežnik fotografskega studia.

Poglavje 5

Analiza učinkovitosti

Aplikacija ima zelo dober odzivni čas. Edina večja težava se pojavi, ko se uporabljajo fotografije večjih velikosti, saj mora naprava narediti veliko več izračunov in ti izračuni trajajo seveda dlje kot pri manjših fotografijah. Veliko večja poraba virov se opazi ob večjih spremembah, kot je npr. izris kvadrata okrog fotografije ali premik kvadrata po fotografiji. Poraba virov je lepo opazna na sliki 5.1.



Slika 5.1: Prikaz obremenitve virov.

Slika 5.1 prikazuje delovanje aplikacije v nekem določenem času. Na začetku merjenja porabe virov lahko opazimo, da obremenitev ni velika, saj meritev

ponazarja izbiro fotografij, premik po seznamu fotografij ipd. Pozneje se poraba poveča, izvajajo se zahtevnejše naloge, kot je npr. risanje in premikanje okvirja na fotografiji.

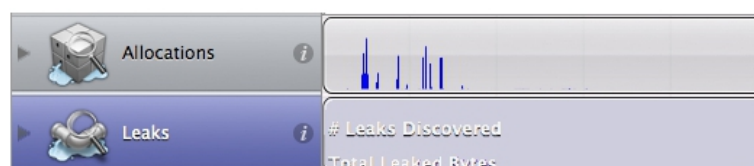
Iz grafov lahko opazimo, da je poraba procesorske in pomnilniške moči enakomerna.

Pri učinkovitosti delovanja aplikacije je potrebno spremljati tudi zasedenost pomnilnika, ki ni več v uporabi (angl. Memory Leaks), saj z nepravilnim sproščanjem zasedamo nepotreben delež pomnilnika. Objekt, ki je še vedno v pomnilniku, aplikacija pa ga ne uporablja več, se imenuje izguba (angl. Leak). Slika 5.2 prikazuje graf, kjer uporabljamo objekte razreda UIImage, vendar se ti ne sprostijo pravilno. Ker se objekti niso pravilno sprostili, narašča zasedenost pomnilnika, to pa povzroči nedelovanje ali počasnejše delovanje aplikacije.



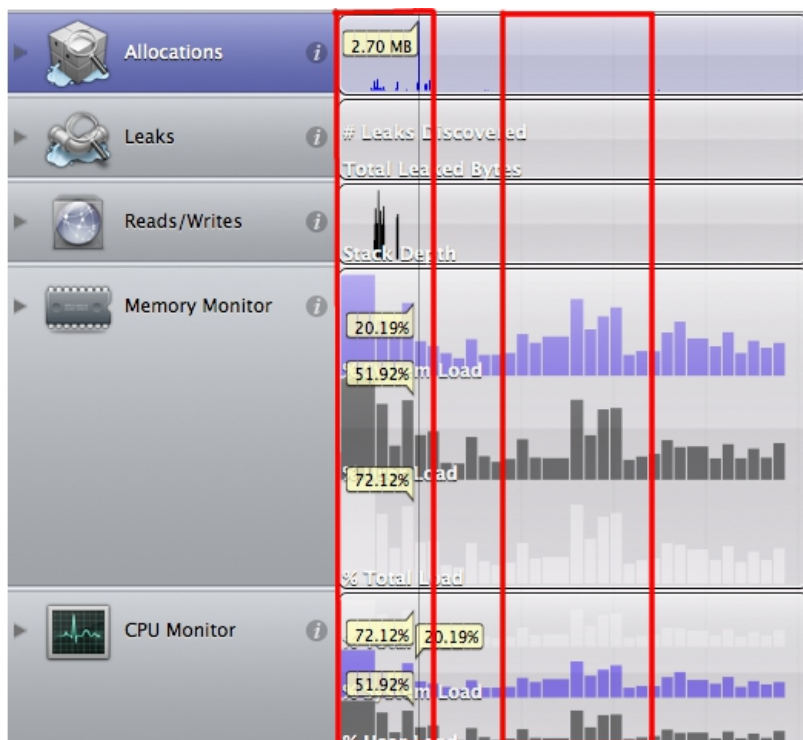
Slika 5.2: Prikaz napak pri sproščanju pomnilnika.

Na sliki 5.3 se zasedenost pomnilnika zmanjša, saj objekte UIImage pravilno sprostimo.



Slika 5.3: Prikaz pravilnega sproščanja pomnilnika.

Ker aplikacija uporablja več virov, prikazuje slika 5.4 odstotke porabe vsakega vira. Največja obremenitev nastane pri premikanju okvirja, ki je na sliki predstavljen z rdečima okvirjema.



Slika 5.4: Prikaz porabe virov.

Aplikacija v tem primeru uporablja 6,19 MB pomnilniškega prostora od tega za premik okvirja potrebujemo 2,70 MB, saj se za vsak premik izriše nova fotografija. Branje in pisanje se uporabljata samo ob izbiri in shranjevanju fotografij, zato je graf skoraj neaktiven. Poraba virov torej niha na podlagi operacije, ki jo izvaja aplikacija. Pri manj zahtevnih operacijah, kot so npr. izbira fotografij, premikanje po fotografijah, vnos podatkov, izbira poslovalnice je poraba virov nizka.

5.1 Možne izboljšave

Pri izboljšavi aplikacije bi izpostavili kompaktno pošiljanje fotografij. Najprimernejša izboljšava bi bila, če bi celotno naročilo poslali v stisnjeni obliki, v eni datoteki.

Pri izboljšavah je potrebno misliti na intenzivnost uporabe posameznih funkcionalnosti. Z optimizacijo aplikacije bi lahko izboljšali izrabo virov.

Smotrno bi bilo vključiti tudi funkcijo za odpravljanje rdečih oči, s čimer lahko očem povrnemo njihov naravni videz.

V aplikaciji prav tako manjka ročno nastavljanje kontrasta, svetlosti in gama vrednosti.

Uporabniki želijo na posameznih fotografijah zabeležiti tudi datum nastanka fotografije, zato bi se lahko vključila tudi ta funkcija.

Ker je uporabniški vmesnik zgrajen tako, da ga lahko uporabnik uporablja samo v pokončni smeri, bi bilo potrebno izdelati tudi uporabniški vmesnik za ležečo smer, vsaj za vnos besedila.

V uporabniški vmesnik bi lahko dodali vpogled v cenik fotografij.

Pri gumbih na sliki 4.7 ni jasno ponazorjeno, katero funkcijo ima posamezen gumb. Izboljšali bi gumb  tako, da bi takoj nakazal, katera funkcija se bo izvedla brez potrebe po dodatnem oknu. Tudi gumba za izdelavo indeksa slik in izdelavo zgoščenke bi bilo smiselno zamenjati z DA in NE. Ob vnosu poštne številke bi se lahko ime pošte izpisalo avtomatično.

Poglavje 6

Zaključek

Končni rezultat diplomske naloge je delujoča aplikacija za obdelavo in naročanje fotografij v fotografskem studiu.

Med drugim je bil moj cilj izdelati aplikacijo, ki bi omogočila pošiljanje fotografij v čim manj korakih. Če upoštevam, da uporabnik ne obdeluje fotografij in jih takoj po izbiri pošlje, mu je to omogočeno v štirih korakih.

Pri razvoju aplikacije sem imel na začetku težave z uporabniškim vmesnikom, saj gre za specifično mobilne naprave. Z nekaj truda in branja sem težavo odpravil. Nekaj težav se je pojavilo tudi pri samem programiranju, saj sem se moral naučiti, kako aplikacija Xcode deluje in kako se definirajo in uporabljajo posamezni gradniki aplikacije. Za učenje sem uporabil video predstavitve in primere iz knjig.

S programskim jezikom Objective-C in programom Xcode sem se naučil veliko novega, saj sem spoznal veliko novih gradnikov, načinov uporabe in na koncu dosegel cilj, ki sem si ga zadal. Razvoj aplikacije je že delno optimiziran, saj Xcode omogoča pregled dela z viri naprave.

Aplikacijo bomo poskušali tudi lansirati v slovenskem okolju, prvi stik je že vzpostavljen.

Literatura

- [1] ePhotoLab. Dostopno na:
<http://www.ineta.si/?action=epl> (zadnji obisk: 26. 3. 2011)
- [2] iPhone. Dostopno na:
<http://en.wikipedia.org/wiki/IPhone> (zadnji obisk: 26. 3. 2011)
- [3] iPod. Dostopno na:
<http://en.wikipedia.org/wiki/IPod> (zadnji obisk: 14. 3. 2011)
- [4] iPad. Dostopno na:
<http://en.wikipedia.org/wiki/IPad> (zadnji obisk: 26. 3. 2011)
- [5] iOS Apple. Dostopno na:
[http://en.wikipedia.org/wiki/IOS_\(Apple\)](http://en.wikipedia.org/wiki/IOS_(Apple)) (zadnji obisk: 23. 3. 2011)
- [6] Apache server. Dostopno na:
<http://httpd.apache.org/> (zadnji obisk 7. 3. 2011)
- [7] PHP. Dostopno na:
<http://en.wikipedia.org/wiki/PHP> (zadnji obisk: 25. 3. 2011)
- [8] XcodeWorkspace Guide. Dostopno na:
http://developer.apple.com/library/mac/documentation/DeveloperTools/Conceptual/-XcodeWorkspace/Xcode_Workspace.pdf (zadnji obisk: 24. 1. 2011)
- [9] Developer Tools: What's new in Xcode 4. Dostopno na:
<http://developer.apple.com/technologies/tools/whats-new.html#new-debugger> (zadnji obisk: 15. 3. 2011)
- [10] The Objective C Programming Language. Dostopno na:
<http://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/-ObjectiveC/ObjC.pdf> (zadnji obisk: 24. 2. 2011)

- [11] Objective C. Dostopno na:
<http://en.wikipedia.org/wiki/Objective-C> (zadnji obisk: 26. 3. 2011)
- [12] James Bucanek, *Beginning Xcode*, Wrox Press, 2006.
- [13] Fritz Anderson, *Step into Xcode Mac OS X Development*, Addison Wesley Professional, 2006.
- [14] Stephen G. Kochan *Programming in Objective-C 2.0*, Addison-Wesley, USA, 2008, str. 212-229, 262-312.