Sustav za skeniranje i praćenje prodaje ulaznica

Glavota, Fran

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

ELEKTROTEHNIČKI FAKULTET

Sveučilišni studij

Sustav za skeniranje i praćenje prodaje ulaznica

Završni rad

Fran Glavota

Osijek, 2016.

1. SADRŽAJ

1. UVOD

Apple je jedna od najvećih tvrtki na svijetu. Davno su probili granicu od 1 milijardu prodanih iOS uređaja i tako se svrstali u vodeće tehnološke kompanije. Mnogi se slažu da je Apple izgradio svoj veliki brend na mobilnom uređaju zvanom iPhone koji već godinama prednjači inovacijama u mobilnom svijetu. iOS kao platforma ima svojih dobrih i loših strana, međutim ono što svakog korisnika oduševljava je njezina - jednostavnost. Izradom ovog završnog rada i prateće mobilne aplikacije obuhvaćen je proces pojednostavljivanja očitavanja i praćenja prodaje ulaznica za razna kulturna, sportska i društvena događanja. Instancirana je baza podataka koja je pohranjena na poslužitelj, aplikacijsko programsko sučelje zaduženo za komunikaciju aplikacije s bazom podataka te na kraju sama aplikacija u programskom jeziku *Swift.* Rad obuhvaća kratku povijest i princip funkcioniranja bar kodova, uvod u iOS platformu, izradu same mobilne aplikacije te pripadajuće baze podataka.

Poglavlje 2 prikazuje povijest bar kodova, vrste te način funkcioniranja istih. Posebna pažnja posvećena je dvodiomenzionalnom (2D) kodu, odnosno QR (engl. *Quick Respnse*) kodu te linearnom bar kodu. U poglavlju 3 prolazi se kroz osnovnu arhitekturu platforme iOS, načina na koji ona funkcionira i od kojih se slojeva sastoji. Isto tako navedeni su najbitniji programski okviri (engl. *framework*) i tehnologije koje omogućuju izradu aplikacija za iOS uređaje. Poglavlje 4 donosi konkretne primjere korištenja navedenih razvojnih okvira i tehnologija. Također, obuhvaća cjelokupan proces izrade mobilne aplikacije koja ima bazu na poslužitelju, objašnjava proces kreiranja baze podataka, korisničkog sučelja te povezivanja aplikacije s bazom na poslužitelju kao i način na koji oni međusobno komuniciraju. Na kraju je razjašnjena i implementacija QR i bar kod čitača. U poglavlju 5 prikazani su rezultati i uspješnost izrađene aplikacije. Slikama se prikazuju rezultati skeniranja različitih vrsta bar i QR kodova. Provjerava se radi li aplikacija i sa ranije poništenim kodovima te brzina kojom uređaj skenira zadani kod.

1.1. Zadatak završnog rada

Potrebno je ostvariti aplikaciju za čitanje QR i linearnog bar koda s ulaznica za razna sportska, kulturna i društvena događanja. Također, treba kreirati bazu na poslužitelju koja će na temelju skeniranih podataka omogućiti izradu izvješća o očitanim i poništenim ulaznicama. Razvijeni sustav treba prikladno ispitati.

2. KRATKI PRIKAZ BAR KODOVA

Bar kod (engl. *Barcode*) predstavlja način označavanja raznih proizvoda, lokacija, usluga, cijena i slično. Koristi se u identifikaciji u postupku prolaska kroz određeni prostor [1]. Najpoznatiji primjeri korištenja bar kodova su: prilikom kupovine u trgovini, ulaska na koncert ili utakmicu, u zračnim lukama, za praćenje prtljage, putovnice itd. Praktični su iz razloga što istovremeno pružaju brzinu i sigurnost kontrole proizvoda.

Tehnički gledano, bar kod predstavlja numeričke, alfanumeričke ili *ASCII* znakove, pretvorene u smisleni niz tamnih linija ili modula i praznih (svijetlih) međuprostora. Njihovim skeniranjem pomoću čitača dobiva se identifikator objekta za pristup bazi podataka u kojoj se nalaze detaljnije informacije o proizvodu. Postoji više vrsta bar kodova, a u ovom radu koriste se linijski te dvodimenzinalni bar kod.

2.1. Linijski bar kod

Bar kod ili *UPC* kod se sastoji od okomitih nizova linija i praznina između njih. Svaka simbolika koristi različite početne i krajnje znakove bar koda. Također, može sadržavati i kontrolnu znamenku za provjeru ispravnosti očitanja kao i interpretacijsku liniju kako bi se mogao očitati kod i bez bar kod čitača. Prema [1], postoji više tipova linijskih bar kodova:

- numerički: *Code 11, EAN-13, EAN-8, PostNet, UPC-A, UPC-E* itd.

- alfanumerički: *Code 128, Code 39, Code 93*

Slika 2.1 prikazuje primjer UPC-A bar koda.

Slika 2.1 Primjer UPC-A bar koda

2.2. Dvodimenzionalni bar kod

Dvodimenzionalni bar kod je nastao nadogradnjom početnog linearnog bar koda. Njegova glavna prednost je što može pohraniti znatno više podataka od linearnog bar koda. Sadržaj ovakvog koda predstavljen je pomoću tamnih i svijetlih kvadratića. Dok je kod linearnog bar koda bitan samo vodoravni raspored i debljina linija, kod dvodimenzionalnog bar koda je bitan i vodoravan i okomit

raspored elemenata. Kao i kod linearnih, postoji više tipova dvodimenzionalnih bar kodova. Neki od njih su: *PDF417, DataMatrix, Maxicode, QR Code*. Slika 2.2 prikazuje primjer QR koda.

Slika 2.2 Primjer QR koda

U ovom radu će biti prikazano više različitih vrsta bar i QR kodova generiranih putem interneta te postupak skeniranja istih.

3. PREGLED iOS PLATFORME

iOS je operacijski sustav koji pokreće uređaje poput iPad-a, iPhone-a i iPod touch-a. Operacijski sustav upravlja sklopovljem (engl. *hardware*) uređaja i pruža tehnologiju potrebnu za implementiranje izvornih (engl. *native*) aplikacija. Također, operacijski sustav se isporučuje s različitim sustavskim aplikacijama kao što su Telefon (engl. *Phone),* Poruke (engl. *Messages)* i Safari web preglednik, a koje pružaju standardne sustavske usluge korisniku. Programski razvojni paket iOS (engl. *Software Development Kit, SDK*) sadrži alate i sučelja potrebna za ravoj, instalaciju, pokretanje i testiranje izvornih aplikacija koje se pojavljuju na iOS početnom zaslonu. Izvorne aplikacije su izrađene upotrebom iOS sustavskog programskog okvira i Objective-C ili u novije vrijeme, programskog jezika Swift. Za razliku od web aplikacija, izvorne aplikacije su instalirane fizički na uređaju i uvijek su dostupne korisniku, čak i kada je uređaj u zrakoplovnom načinu rada. Sama aplikacija i njeni podaci su sinkronizirani s korisnikovim računalom kroz iTunes [2].

3.1. Arhitektura iOS tehnologije

Na najvišoj razini, iOS se ponaša kao medijator između sklopovlja i same aplikacije. Aplikacije ne komuniciraju direktno sa sklopovljem, nego kroz skup definiranih sustavskih sučelja. Ta sučelja omogućavaju olakšano pisanje aplikacija koje konzistentno rade na uređajima raznih sklopovskih mogućnosti.Implementaciju iOS tehnologija može se gledati kao skup slojeva prikazanih na slici 3.1. Niži slojevi sadrže osnovne usluge i tehnologije, a viši slojevi pružaju sofisticiranije usluge i tehnologije.

Slika 3.1 Slojevi iOS platforme

3.1.1. Sloj *Cocoa Touch*

Sloj *Cocoa Touch* sadrži ključne programske okvire za izradu iOS aplikacije. Ti programski okviri definiraju izgled aplikacije te pružaju osnovnu strukturu aplikacije i podršku za tehnologije kao što su višezadaćnost (engl. *multitasking*), unos dodirom (engl. *touch-based input*), *push* notifikacije i puno usluga visoke razine složenosti (engl. *high-level*). Pri dizajniranju aplikacije potrebno je istražiti odgovaraju li tehnologije u ovom sloju potrebama aplikacije.

Neke od najvažnijih tehnologija koje pruža *Cocoa Touch* sloj nazivaju se engleskim nazivima:

- *App Extensions* – kod koji omogućuje neku funkcionalnost aplikaciji.
- *Handoff* – proširuje korisničko iskustvo na više uređaja, može se započeti neka aktivnost na jednom uređaju te je potom nastaviti na drugom.
- *Document Picker* – svojstvo koje korisniku daje pristup dokumentima izvan aplikacije, mehanizam za dijeljenje dokumenata između aplikacija.
- *AirDrop* – omogućuje korisnicima dijeljenje fotografija, dokumenata, URL-ova s uređajima u blizini
- *TextKit* – skup klasa za upravljanje tekstom i tipografijom
- *Auto Layout* – pomaže napraviti dinamičko sučelje s vrlo malo koda, definira kako prikazati elemente u korisničkom sučelju

3.1.2. Multimedijski sloj (engl. *Media layer*)

Multimedijski sloj sadrži grafičke, audio i video tehnologije koje se koriste za implementaciju multimedijalnog iskustva u aplikaciju. Sljedeće tehnologije u ovom sloju olakšavaju izradu aplikacije koja će izgledati i zvučati dobro.

- Grafičke tehnologije: *UIKit graphics, Core Graphics, Core Animation, Core Image, OpenGL, Metal, TextKit* and *Core Text, Image I/O, Photos Library*

- Audio tehnologije: *Media Playe*r, *AV Foundation, OpenAL, Core Audio*

- Video tehnologije: *AVKit, AV Foundation, Core Media*

3.1.3. Sloj osnovnih usluga (engl. *Core Services layer*)

Sloj *Core Services* sadrži osnovne usluge za aplikaciju. Najvažnije od njih su *Core Foundation* i *Foundation* koje definiraju osnovne tipove koje koriste sve aplikacije. Također, ovaj sloj sadrži

individualne tehnologije koje podupiru tehnologije poput lociranja, iCloud-a, društvenih mreža i umrežavanja. Mogućnosti visoke razine ovog sloja su sljedeće:

iCloud prostor za pohranu dopušta aplikaciji da skladišti korisničke podatke na središnju lokaciju. Tako korisnici mogu pristupiti podacima sa svih računala i iOS uređaja te uređivati dokumente s bilo kojeg uređaja.

Objektni blokovi (engl. *Block objects*) su konstruktori koji se mogu inkorporirati u C i Objective-C kod. U iOS-u, ovi blokovi se obično koriste:

- kao zamjena za delegate i delegatne metode
- kao zamjena za povratne funkcije
- za implementaciju rukovatelja za *one-time* operacije
- za izvršavanje zadatka na svim objektima u zbirci

Ovdje valja napraviti digresiju i objasniti što su delegati budući da u kasnijem procesu izrade aplikacije imaju važnu ulogu. Dakle, delegiranje je obrazac koji omogućuje klasi da povjeri (delegira) pojedine odgovornosti instanci različitog tipa, odnosno delegatu [3].

Podatkovna zaštita (engl. *Data protection*) dopušta aplikacijama koje rade s osjetljivim korisničkim podacima da spremaju podatke s kojima rade u kodiranom formatu. Na taj način, kada je uređaj zaključan dokument je nedostupan i aplikaciji i potencijalnom napadaču.

Podrška dijeljenju dokumenata (engl. *File-Sharing support*) dopušta aplikaciji da podatke učini dostupnima i na iTunes-u.

Glavni središnji otpremitelj (engl. *Grand Central Dispatch*) je tehnologija koja omogućuje upravljanje izvršavanjem zadataka aplikacije.

3.1.4. Sloj jezgre operacijskog sustava (engl. *Core OS*)

Sloj *Core OS* sloj sadrži osnovne opcije na kojima su izgrađene druge tehnologije. Iako se ove tehnologije ne koriste direktno u aplikaciji, razni programski okviri ih koriste.

***Akceleracijski programski okvir (engl. Accelerate Framework*)** sadrži sučelja za izvođenje obrade digitalnog signala, linearne algebre, kalkulacija za obradu slika.

Osnovni *Bluetooth* programski okvir (engl. *Core Bluetooth Framework*) omogućuje programerima komunikaciju s *Bluetooth* dodacima male potrošnje.

Programski okvir za vanjske dodatke (engl. *External Accessory Framework*) pruža podršku za komunikaciju sa sklopovskim dodacima koji su priključeni na iOS uređaje.

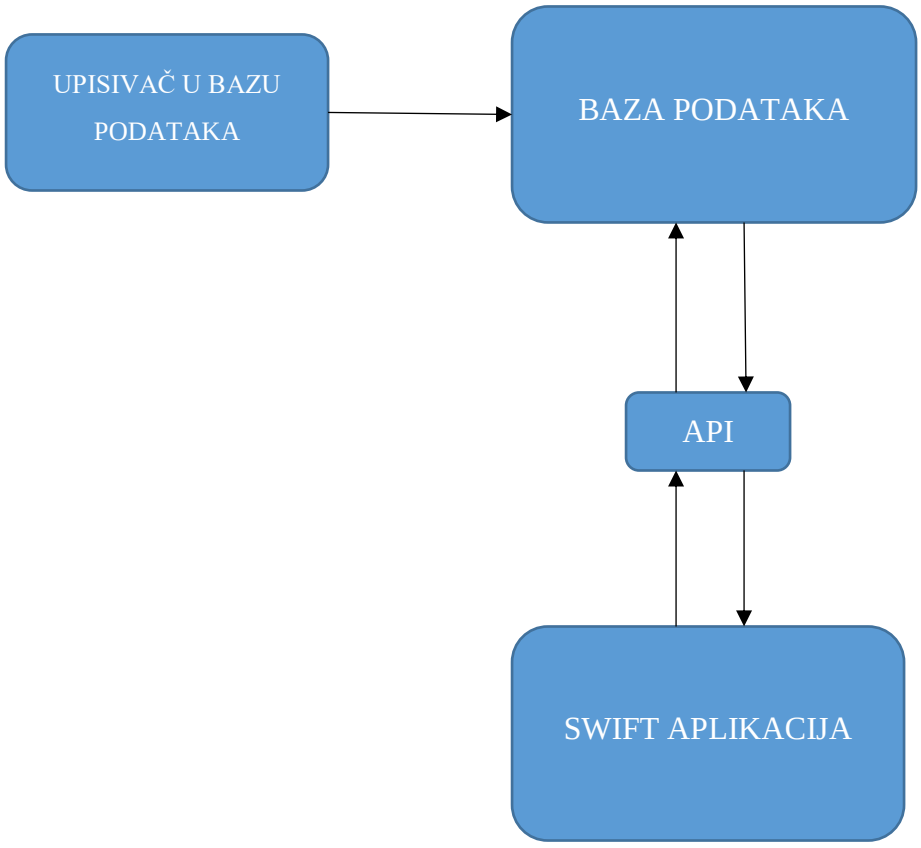
Općenite sigurnosne usluge (engl. *Generic Security Services*) pružaju standardni skup usluga povezanih sa sigurnošću iOS aplikacija.

Lokalni identifikacijski programski okvir (engl. *Local Authentication Framework*) dopušta uporabu senzora za prepoznavanje otiska prsta *(*engl. *Touch ID)* koji služi za identifikaciju korisnika.

Programski okvir za proširenje mreže (engl. *Network Extension Framework)* pruža podršku za konfiguraciju i kontroliranje *VPN tunela*.

4. IZRADA iOS MOBILNE APLIKACIJE

Izrada iOS mobilne aplikacije može se podijeliti u nekoliko faza. Broj faza i njihov redoslijed određeni su pojedinačno za svaku aplikaciju i nisu jednoznačno određeni. Ovaj projekt podijeljen je u četiri faze. Na početku je instancirana baza podataka na poslužitelju, a zatim isprogramirana skripta koja služi za automatsko upisivanje podataka u bazu. Zatim je kreirana skripta koja služi za komunikaciju između same aplikacije i baze na poslužitelju. Posljednja faza projekta je kreiranje *Swift* mobilne aplikacije. Ključne komponente projekta i smjerovi djelovanja/komunikacije prikazani dijagramom izgledaju kao na slici 4.1.



Slika 4.1 Komponente projekta i smjerovi komunikacije

4.1. Programski jezik PHP

PHP programski jezik je široko rasprostranjeni skriptni jezik otvorenog koda. Namijenjen je za *web* razvoj i može biti ugrađen u *HTML (*engl. *HyperText Markup Language*). *HTML* predstavlja prezentacijski jezik za izradu *web* stranica. *PHP* kod se izvršava na samom poslužitelju, generiranjem *HTML-a* ili nekog drugog strukturiranog oblika odgovora koji se onda šalje klijentu. Klijent prima samo rezultate nastale pokretanjem skripte, ali ne i kod koji se nalazi u njoj. Prednost programskog jezika *PHP* je jednostavnost korištenja, a s druge strane nudi napredna svojstva za naprednije programere.

4.2. Instanciranje baze podataka

Za ovaj tip aplikacije nužno je imati bazu podataka koja sadržava sve dostupne kodove. To može biti učinjeno na više načina, odnosno da je baza spremljena lokalno ili na nekom od poslužitelja. Budući da je u praktičnom smislu bolje imati bazu kojoj se može udaljeno pristupati i s više uređaja, baza je postavljena pomoću *OpenShift* besplatnog *hostinga*. Na poslužitelj je instaliran i besplatan alat *phpMyAdmin*, te je pomoću sučelja *phpMyAdmin* kreirana baza.

PhpMyAdmin pruža intuitivno *web* sučelje. Također, sadrži podršku za veliki broj *MySQL* svojstava kao što je pretraživanje i kreiranje baze podataka, tablica, polja i indeksa. Omogućuje preimenovanje, proširivanje baze, dodavanja stupaca i mnoge druge radnje. U navedenom alatu kreirana je baza imena „codeclaim“ koja se sastoji od 3 stupca. Prvi stupac „id“ predstavlja jedinstveni redni broj i sadrži brojeve, tj. varijable cjelobrojnog tipa. Označena je kao jedinstveni ključ prema kojem svaki upis u bazu postaje jedinstven. Drugi stupac „code“ je tipa VARCHAR veličine 255 u koju je spremljen jedinstveni kod u obliku polja alfanumeričkih znakova (engl. *string)*. Razlika između VARCHAR i običnog CHAR tipa podataka je u tome što VARCHAR pametnije koristi memoriju, tj. ukoliko je unešeni podatak manji od maksimalne dopuštene vrijednosti memorija se automatski smanjuje i time optimalno koristi. Treći stupac je posebno važan, jer može imati dvije vrijednost, a deklariran je kao BOOLEAN tip. Prema zadanom iznosi 0, ali nakon što se kod očita mijenja vrijednost u 1 te sprječava višekratnu upotrebu iste ulaznice.

4.2.1. Upisivač u bazu podataka

Nakon što je kreirana baza podataka na virtualnom poslužitelju, u nju je trebalo unijeti i podatke. Stoga je kreiran *importer.php* dokument koji služi za automatsko upisivanje kodova iz tekstualne datoteke u samu bazu kako ne bi bilo potrebno unositi podatke ručno. Programski kod 4.1 prikazuje postupak spajanja na poslužitelj te raspakiravanje tekstualnog dokumenta po redcima.

```
$con=
mysqli_connect("127.6.118.130:3306","ticketUser","yPL@ticket1234&J","code
claim")
     or die("Failed to connect to MySQL: " . mysqli_connect_error());
$txt_file = file_get_contents('RandomKodovi.txt');
$rows = explode("\n", $txt_file);
array_shift($rows);
```

Programski kod 4.1. Povezivanje na poslužitelj i raspakiravanje tekstualnog dokumenta

U prvom dijelu dokumenta nužno je povezati se na poslužitelj putem zadane IP adrese i pristupa, korisničkog imena, lozinke. Također potrebno se referencirati na ime baze kojoj pristupamo „codeclaim“. Naredba *file_get_contents* učitava tekstualnu datoteku u kojoj su sadržani kodovi. Funkcijom *explode* tekstualni dokument se dijeli na polje alfanumeričkih znakova. Oznakom „\n“ definiran je uzorak prepoznavanja novog alfanumeričkog znaka unutar dokumenta. Programski kod 4.2 prikazuje naredbu za upisivanje podataka u bazu po redcima.

```
$sql = "INSERT INTO tickets (code, claimed) VALUES ('$row_data[0]', 0)";
$result = mysqli_query($con,$sql);
```

Programski kod 4.2. Upisivanje u bazu podataka po redcima

Na kraju, svaki odijeljeni redak upisan je u bazu i predstavlja jedinstveni kod koji će biti sadržan na jednoj ulaznici.

Napravljena tablica u bazi prikazana je na slici 4.2.

	id	code	claimed
Uređivanje Kopiraj Izbriši	3	2916131026	0
Uređivanje Kopiraj Izbriši	4	5017815672	0
Uređivanje Kopiraj Izbriši	5	9550045343	0
Uređivanje Kopiraj Izbriši	6	4420332951	0
Uređivanje Kopiraj Izbriši	7	2632058373	0
Uređivanje Kopiraj Izbriši	8	7874041754	0
Uređivanje Kopiraj Izbriši	9	1844330807	0
Uređivanje Kopiraj Izbriši	10	2310649673	0
Uređivanje Kopiraj Izbriši	11	8078188607	0
Uređivanje Kopiraj Izbriši	12	4519842342	0
Uređivanje Kopiraj Izbriši	13	2108300414	0
Uređivanje Kopiraj Izbriši	14	7655796960	0
Uređivanje Kopiraj Izbriši	15	656398656	0
Uređivanje Kopiraj Izbriši	16	6804163921	0
Uređivanje Kopiraj Izbriši	17	469894172	0
Uređivanje Kopiraj Izbriši	18	6576183231	0
Uređivanje Kopiraj Izbriši	19	6190231405	0

Slika 4.2 Kodovi upisani u tablicu baze podataka

4.3. Aplikacijsko programsko sučelje (engl. *Application Programming Interface, API)*

Aplikacijsko programsko sučelje *(API)* je skup instrukcija/kodova i specifikacija koje mogu slijediti programi kako bi komunicirali međusobno. Dakle, API služi kao posrednik između različitih programskih rješenja i olakšava njihovu komunikaciju na sličan način na koji korisničko sučelje olakšava komunikaciju između ljudi i računala. U ovom konkretnom projektu treba kreirati *API* koji će omogućiti komunikaciju između iOS aplikacije napisane u *Swift* programskom jeziku i baze podataka koja se nalazi na virtualnom poslužitelju. Zamišljeno je da aplikacija šalje parametre QR ili bar koda u obliku usklađenog lokatora sadržaja (engl. *URL*) putem zahtjeva *GET*.

4.3.1. Metoda GET

*GET HTTP (*engl. *Hypertext Transfer Protocol)* zahtjev metoda je jedna od dvije najčešće korištene metode, *HTTP* predstavlja protokol za prijenos podataka na globalnoj mreži. Ona zahtjeva podatke od naznačenog izvora, u ovom slučaju baze na poslužitelju. Za testiranje provođenja ove metode postoji više različitih alata za slanje zahtjeva *GET*.

4.3.2. Kreiranje *API-a*

API koji omogućava komunikaciju između aplikacije i baze napisan je u obliku PHP skripte. Dakle, njegova glavna zadaća je slanje zahtjeva prema poslužitelju i definiranje odgovora od strane poslužitelja. Ovisno o tome izmjenjuju se podaci u bazi ili se ne događa ništa. Programski kod 4.3 prikazuje povezivanje na poslužitelj te promjenu vrijednosti u stupcu namijenjenu za označavanje poništenih ulaznica.

```
$con=
mysqli_connect("127.6.118.130:3306","ticketUser","yPL@ticket1234&J","code
claim")
     or die("Failed to connect to MySQL: " . mysqli_connect_error());

$code = $_GET['code'];

     $result = mysqli_query($con,"UPDATE tickets SET claimed = 1 WHERE
code = '$code'");
```

Programski kod 4.3. Povezivanje na poslužitelj i označavanje poništene ulaznice

Nakon spajanja na poslužitelj čita se varijabla `$code` koja je dobivena iz *URL-a* zahtjeva *GET* prema poslužitelju s parametrom koda. Zatim se definira varijabla `$result` koja definira rezultat upita koji se šalje bazi podataka. Programski kod 4.4. prikazuje vraćanje *HTTP* statusa ovisno o tome na koliko je redaka upit utjecao.

```
    if(mysqli_affected_rows($con) == 0)
        {
            http_response_code(204);
        }
```

```
else
    {
        http_response_code(200);
    }
```

Programski kod 4.4. Vraćanje *HTTP* statusa

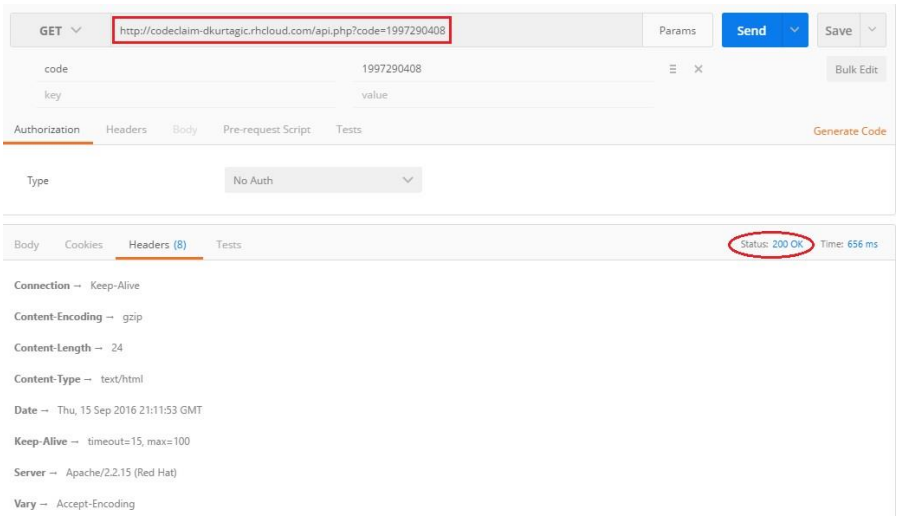
Funkcija `if` funkcija radi sljedeće. Ukoliko je broj izmijenjenih redaka jednak nuli tada poslužitelj kao odgovor vraća *HTTP status kod rezultata* 204, odnosno "Nema sadržaja". U suprotnom, upit je utjecao na neke od redova i poslužitelj kao odgovor vraća *HTTP status kod rezultata* 200, tj. "OK" što predstavlja standardni odgovor za uspješno poslan zahtjev *GET*. Dalje će aplikacija odlučivati ovisno o odgovoru poslužitelja što će prikazivati na zaslonu.

4.3.3. Aplikacija Postman

Postman aplikacija je namijenjena za izradu, testiranje i dokumentaciju aplikacijskog programskog sučelja koju koriste milijuni programera. Njezine ključne mogućnosti su sljedeće:

- sadrži povijest poslanih zahtjeva
- brzo kreiranje zahtjeva
- ugrađena identifikacijska pomoć
- robustan testni okvir
- prilagodljivost skripti

Primjer uobičajenog zahtjeva zahtjeva *GET* i dobivenog odgovora slijedi na slici 4.3.



Slika 4.3 Primjer zahtjeva GET

4.4. Integrirano razvojno okruženje Xcode

Xcode je Apple-*ovo* razvojno okruženje koje se koristi za izradu aplikacija za njihove proizvode. *Xcode* sadrži alate kojima se upravlja cjelokupnim procesom razvoja – od izrade, testiranja, optimiziranja i izdavanja na *Apple Store*.

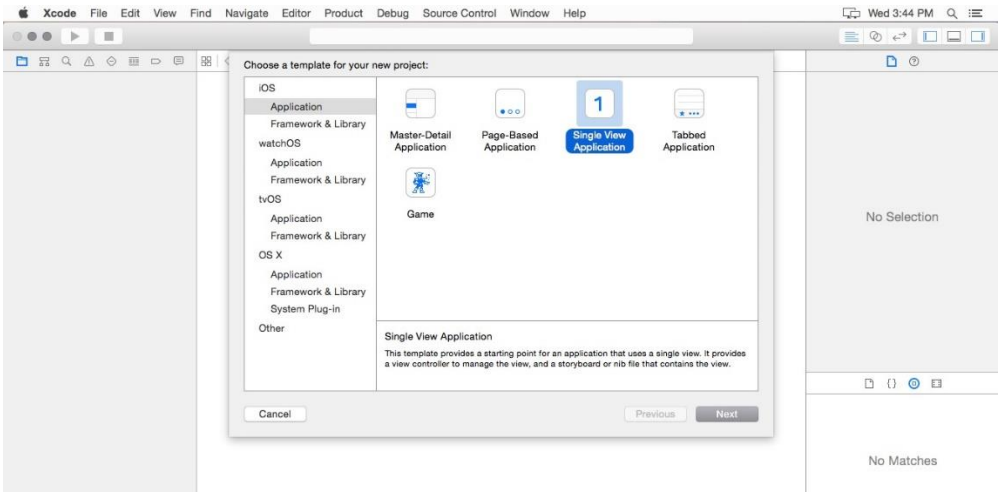
Xcode sučelje obuhvaća uređivanje koda, dizajn korisničkog sučelja i testiranje u jednom razvojnom sučelju. To sučelje se mijenja dok traje proces izrade aplikacije. Na primjer, odabere se dokument na jednom mjestu i prikladni uređivač se otvara na drugom mjestu. Odabere li se simbol ili korisničko sučelje njegova dokumentacija se pojavljuje u blizini.

Omogućuje fokusiranje na zadatak tako što prikazuje samo ono potrebno, kao što je samo izvorni kod ili samo izgled korisničkog sučelja. Također, može se raditi s kodom i korisničkim sučeljem jedno do drugog. Dalje, moguće je prilagoditi okruženje otvaranjem više prozora ili kartica u prozoru.

4.4.1. Pokretanje i kreiranje projekta

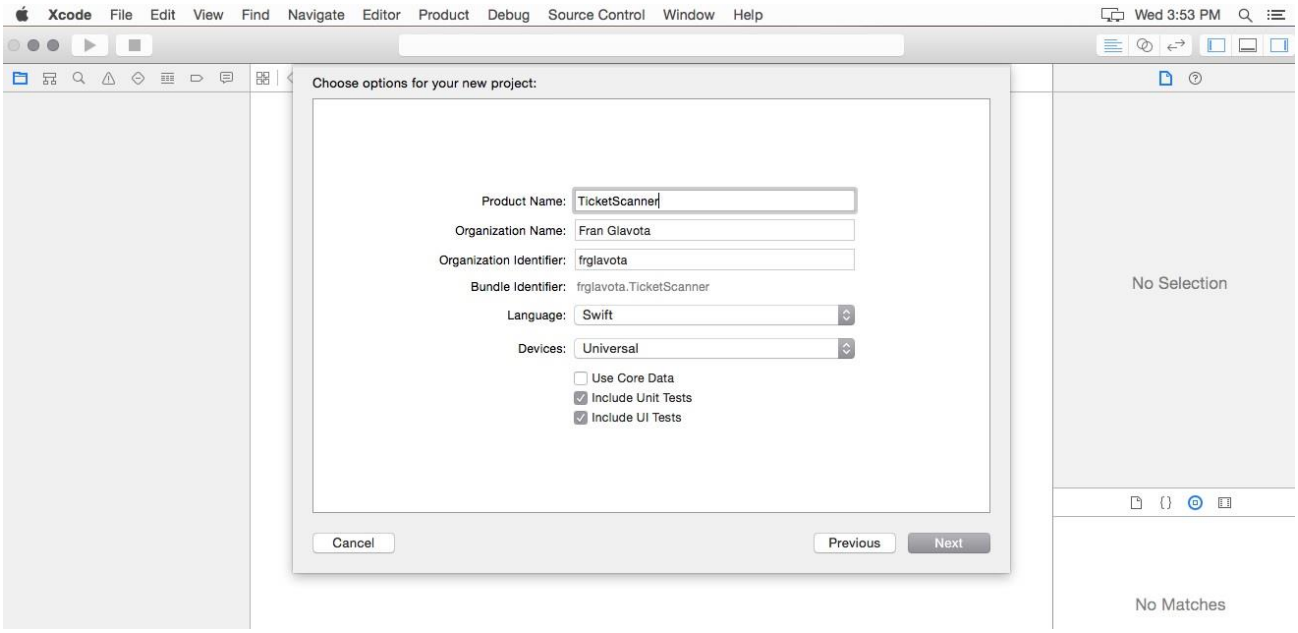
Nakon što se otvori *Xcode* okruženje projekt se kreira redoslijedom *File* → *New* → *Project*

U izborniku koji se otvori izabere se predložak za aplikaciju koja koristi jedan ekran (engl. *Single View Application)* kao dobru praksa pri započinjanju svakog projekta (Slika 4.4).



Slika 4.4 Predložak za aplikaciju koja koristi jedan ekran

Potom se imenuje projekt, bira programski jezik *Swift* te klikne *Create* (Slika 4.5).



Slika 4.5 Kreiranje projekta

4.4.2. Postavljanje klase za upravljanje sučeljima (engl. *UIViewController)*

`UIViewController` je klasa koja omogućuje infrastrukturu za upravljanje sučeljima u aplikaciji. *View controller* je objekt koji upravlja nizom sučelja i kontrolira tok informacija između podatkovnog dijela aplikacije i sučelja koja prikazuju iste. Odgovorna je i za interakciju između sučelja i za koordiniranje obaveza s bilo kojim podatkovnim objektima. *View controlleri* također surađuju s drugim kontrolnim objektima i tako pomažu upravljanjem cjelokupnog sučelja.

Glavne zadaće *view controllera* su sljedeće:

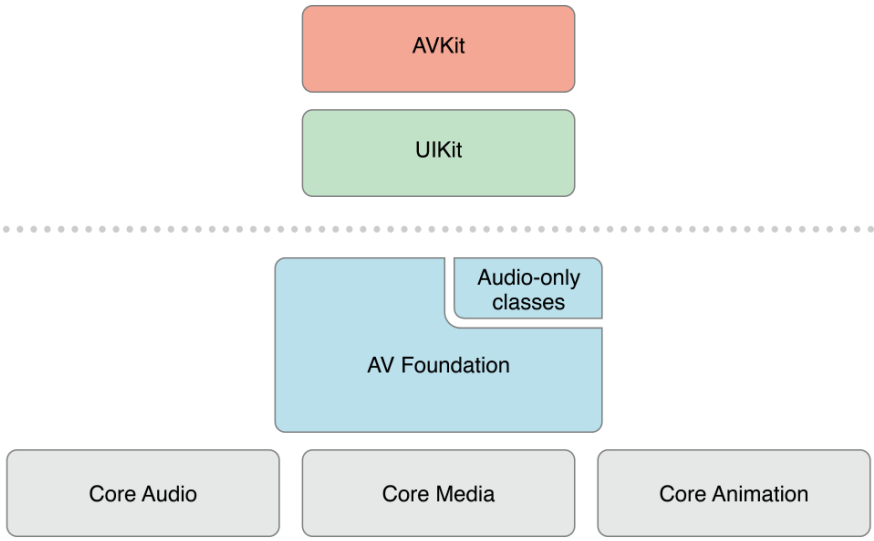
- ažuriranje sadržaja sučelja, obično kao odgovor na neku podatkovnu promjenu
- odgovor na interakciju korisnika sa sučeljima
- promjena veličine i upravljanje izgledom cjelokupnog sučelja

Ovu klasu u kod ubacujemo naredbom `import UIKit`.

4.4.3. Programski okvir AVFoundation

AVFoundation je jedan od nekoliko programskih okvira koji se mogu koristiti za reprodukciju i kreiranje vremenski zasnovanih audiovizualnih medija. On pruža sučelje koje se koristi za rad s vremenski - baziranim audiovizualnim podacima. Na primjer, može se koristiti za ispitivanje,

kreiranje te uređivanje medijskih datoteka. Također, mogu se dobiti ulazni signali od uređaja te manipulirati videom u stvarnomvremenu. Na slici 4.6 je prema [4] prikazana arhitektura ovog programskog okvira.



Slika 4.6 Arhitektura AVFoundation programskog okvira

4.5. Programski jezik Swift

Swift je snažan objektno-orijentirani programski jezik namijenjen za programiranje aplikacija za *macOS, iOS, watchOS i tvOS* [5]. Nastao je iz programskih jezika Objective-C i C jezika. Njegove odlike su sigurnost, fleksibilnost i brzina. Za izradu aplikacije unutar ovog završnog rada korištena je najnovija verzija *Swift 3.0*.

4.6. Klasa *ScannerViewController*

ScannerViewController je jedini *view controller* kojeg sadrži ova aplikacija i u njemu su sadržane sve naredbe i funkcionalnosti koje ona posjeduje.

Na početku svakog programa potrebno je navesti programske okvire koji će se koristiti te globalne varijable. Nakon toga može se preći na specifičnosti svakog pojedinog projekta. U slučaju ove aplikacije namijenjene skeniranju QR i bar kodova potrebno je najprije dohvatiti sklopovsku kameru kojom će se kasnije upravljati na željeni način. To čini sljedeći blok naredbi prikazanih u programskom kodu 4.5

```
do {
    videoInput = try AVCaptureDeviceInput(device: videoCaptureDevice)
} catch {
    return
}
```

Programski kod 4.5. Dohvaćanje sklopovske kamere

AVCaptureDeviceInput je pod klasa klase AVCaptureInput koju se koristi da bi se preuzelo podatke od AVCaptureDevice objekta. To je objekt koji predstavlja fizičko dohvaćanje uređaja i njime povezanih postavki. Ova klasa se koristi kada se želi nabrojati dostupne uređaje te ispitivati njihove mogućnosti.

Potom se *ScannerViewController* postavlja kao delegat za funkciju čitanja koda:

```
metadataOutput.skupMetadataObjectsDelegate(self, queue: DispatchQueue.main)
```

Delegat, odnosno ovaj *controller*, će biti zadužen za čekanje dok se kod očita te će dalje procesirati zadane radnje. Postavljanje delegata je bitno da bi se znao red izvođenja procesa i odgovornosti.

Kako bi se moglo očitati bilo što pomoću kamere potrebno je postaviti izlaz kamere na zaslon. Programski kod 4.6. prikazuje postavljanje izlaza kamere na zaslon uređaja.

```
    previewLayer = AVCaptureVideoPreviewLayer(session: captureSession);
    previewLayer.frame = view.layer.bounds;
    previewLayer.videoGravity = AVLayerVideoGravityResizeAspectFill;
    view.layer.addSublayer(previewLayer);

    captureSession.startRunning();
}
```

Programski kod 4.6. Postavljanje izlaza kamere na zaslon

Koristi se AVCaptureSession objekt koji koordinira tok podataka od audio/video ulaza do izlaza.

AVCaptureVideoPreviewLayer je pod klasa koja se koristi za prikaz onoga što kamera trenutno vidi u stvarnom vremenu.

Nakon toga, postavlja se delegat metoda čitanja koda s kamere na način prikazan programskim kodom 4.7.

```
func captureOutput(_ captureOutput: AVCaptureOutput!, didOutputMetadataObjects
metadataObjects: [Any]!, from connection: AVCaptureConnection!) {
```

Programski kod 4.7. Postavljanje delegat metode čitanja koda

Iz praktičnog razloga, nije potrebno da sustav detektira nove kodove ukoliko već postoji zahtjev prema poslužitelju za provjeru koda. To se sprječava blokom naredbi prikazanih u programskom kodu 4.8.

```
if codeCaptured == true{
    return
}
codeCaptured = true
```

Programski kod 4.8. Sprečavanje nepotrebnog detektiranja kodova

Na kraju slijedi postavljanje komunikacije aplikacije s poslužiteljom, slanje zahtjeva pomoću očitanih kodova te interpretacija primljenih odgovora. Za to je zadužena kreirana funkcija *foundCode* prema programskom kodu 4.9.

```
func foundCode(code: String) {
    print(code)
    let url = URL(string: "https://codeClaim-dkurtagic.rhcloud.com/api.php" + "?code=" +
code)!
```

Programski kod 4.9. Funkcija *foundCode*

Broj dobiven skeniranjem QR ili bar koda predstavlja parametar koji se šalje u *URL-u* prema poslužitelju putem *GET* zahtjeva.

Potrebno je definirati *HTTP* metodu, odnosno zahtjev koji se želi poslati.

```
request.httpMethod = "GET"
```

Dalje slijedi standardni poziv *HTTP* zahtjeva s ranije podešenim parametrima. Ovdje se definiraju akcije kao odgovori na primljene informacije od poslužitelja. Odgovori s poslužitelja ovise o prethodno definiranom *API-u*. U njemu su postavljeni odgovori na promjene u bazi podataka. Ukoliko je odgovor s poslužitelja 200 tada je kod uspješno očitan i ispisuje se prikladna poruka. U slučaju da je odgovor jednak 204 tada kod ne postoji u bazi ili je već prethodno poništen. Također, mora se uzeti u obzir i slučaj kada poslužitelj ne vrati nijednu od navedene dvije vrijednosti. To se može dogoditi ukoliko postoji greška u komunikaciji s poslužiteljom ili slično. Tada također treba ispisati prikladnu poruku na zaslonu. Programski kod 4.10. prikazuje standardni poziv *HTTP* zahtjeva.

```
let task = session.dataTask(with: request as URLRequest, completionHandler: { (data,
response, error)
-> Void in
        if let httpResponse = response as? HTTPURLResponse {
            if httpResponse.statusCode == 200
            {
                self.present(UIAlertController.init(title:nil, message: "Kod je uspješno
poništen", preferredStyle: UIAlertControllerStyle.alert) , animated: true, completion:{
                    DispatchQueue.main.asyncAfter(deadline: .now() + 3.0) {
                        self.dismiss(animated: true, completion: {
                            self.codeCaptured = false
                        })
                    }
                })
            }
            else if httpResponse.statusCode == 204
            {
                self.present(UIAlertController.init(title:nil, message: "Kod ne postoji u bazi ili je
prethodno poništen", preferredStyle: UIAlertControllerStyle.alert) , animated:true,
completion:{
                    DispatchQueue.main.asyncAfter(deadline: .now() + 3.0) {
                        self.dismiss(animated: true, completion:{
                            self.codeCaptured = false
```

Programski kod 4.10. Standardni poziv *HTTP* zahtjeva

Valja primijetiti da je na u svakom od slučajeva definirano i vrijeme od tri sekunde u kojem će poruka biti ispisana na zaslonu.

4.7. Sustav GIT

Pri kreiranju svakog složenijeg projekta važno je pratiti proces izrade i vođenja procesa. Iz tog razloga nastao je GIT, sustav za kontrolu i upravljanje izvornim kodom. Njegove glavne osobine su:

Distribuiranost – nije potrebno sve promjene koda spremati na središnje spremište *(master)* već se mogu pohraniti lokalno, a sinkronizirati s glavnim spremištem po potrebi

Usklađenost s protokolima – podržava razne internetske protokole kao što su: *HTTP, FTP, git, SSH, rsync*

Učinkovitost u radu s velikim projektima

Postoji više vrsta GIT klijenata, a u ovom projektu korišten je besplatni klijent *SourceTree.*

5. REZULTATI TESTIRANJA

U ovom poglavlju prikazani su rezultati, odnosno uspješnost kreirane aplikacije. Najbolji način za njihov prikaz su priložene slike zaslona na kojima se može vidjet uspješnost prepoznavanja kodova. Da bi se uopće moglo testirati radi li aplikacija onako kako bi trebala potrebno je generirati bar kodove koji sadrže numeričke znakove unešene u bazu. Na internetu postoje razni bar kod generatori [6] u kojima se može izabrati tip koda te unijeti znakove u onom obliku u kojem su unešeni u bazu podataka. Nakon što se izgenerirao zadani bar kod potrebno je pokrenuti aplikaciju na mobilnom uređaju. Budući da je aplikacija isprogramirana na način da se odmah po ulasku u nju pali kamera i prikazuje ono što ona vidi na zaslonu, u vrlo kratkom roku se mobitel može približiti željenom bar kodu i pokušati očitati ga. Ovisno o tome da li bar kod postoji u bazi podataka ili je već prethodno poništen, ispisuje se poruka na zaslonu mobilnog uređaja. Na slici 5.1 nalazi se primjer uspješno poništenog QR bar koda.

Uspješnost očitanosti ne ovisi samo o tome da li promatrani bar kod sadrži podatke unesene u bazu već i o količini tipova bar koda podržanih od strane same aplikacije. Na to se može utjecati dodavanjem što raznovrsnijih tipova bar koda pri samom pisanju *Swift* aplikacije. Tipovi podržani ovom aplikacijom su sljedeći: *QR, UPC-E, Code 39, Code 128, EAN-8, EAN-13. PDF417, Code 39Mod43, Aztec, ITF-14, Code 93, DataMatrix, Interleaved 2 of 5*.

Ukoliko se pokuša očitati bilo koji tip koda koji nije podržan u aplikaciji, bez obzira sadrži li podatke iz baze podataka, na ekranu se neće ispisati nikakva poruka. Na slici 5.2. nalazi se primjer neuspješno očitanog bar koda tipa Code 93.

Slika 5.1 Primjer uspješno očitanog QR koda

Slika 5.2 Primjer neuspješno očitanog bar koda tipa Code 93

Dakle, može se zaključiti da je cilj završnog rada ispunjen. Kodovi se uspješno skeniraju (100%), a aplikacija prepoznaje kodove koji su poništeni ranije. Na novijim iPhone modelima uređaja rezultat skeniranja je gotovo trenutan, dok je starijim modelima potrebno nešto više vremena (tri do četiri sekunde) da bi prepoznali kod i ispisali rezultat na zaslonu.

6. ZAKLJUČAK

U ovom radu predstavljen je sustav za skeniranje i praćenje prodaje ulaznica putem iOS mobilne aplikacije napisane u programskom jeziku *Swift*. Ona je popraćena *MySQL* bazom podataka koja se nalazi na poslužitelju. Za pisanje skripti koje služe za komunikaciju aplikacije s bazom te automatsko upisivanje u bazu korišten je programski jezik *PHP*. Baza podataka kreirana je pomoću sučelja *phpMyAdmin* i pohranjena pomoću *OpenShift hostinga*. Potom je napisana skripta *importer.php* u *PHP* programskom jeziku koja služi za automatsko upisivanje podataka iz tekstualnog dokumenta u bazu podataka. Njena zadaća bila je spojiti se na poslužitelj na kojem se nalazi baza podataka, raspakirati tekstualnu datoteku te zatim upisati red po red iz tekstualne datoteke u kolumne tablice koja je stvorena u bazi podataka. Nakon toga, u istom programskom jeziku, isprogramirano je aplikacijsko programsko sučelje (*API)* koje je zaduženo za komunikaciju aplikacije s bazom podataka na poslužitelju. Funkcionira na način da aplikacija šalje parametre QR ili bar koda u obliku *URL-a* putem zahtjeva *GET*. Uspješnost slanja zahtjeva ispitana je pomoću aplikacije *Postman*. Na kraju je kreirana *Swift* aplikacija koja se sastoji od jednog sučelja koje se pojavi odmah nakon pokretanja aplikacije. Prvo se fizički dohvaća kamera, kreira delegat za funkciju čitanja koda te potom postavlja izlaz kamere na zaslon. Zatim se postavlja delegat metoda čitanja koda s kamere i na kraju slijedi postavljanje komunikacije aplikacije s poslužiteljom, slanje zahtjeva pomoću očitanih kodova te interpretacija primljenih odgovora.

Dobiveno rješenje je zadovoljavajuće. Aplikacija se pokazala kao brza i pouzdana. Odgovor na očitavanje koda je gotovo trenutan i točan. Rješenje jednako dobro funkcionira pri očitavanju s različitih podloga kao što su zaslon računala, papirnata podloga, plastična naljepnica i slično.

Aplikacija bi se mogla poboljšati dodavanjem dodatnog sučelja koje bi grafički prikazivalo stanje u bazi podataka. Na taj način bi bilo slikovito prikazano koliko je karata očitano u određenom trenutku te koliko je karata još uvijek ne poništeno.

7. LITERATURA

[1] Bar kodovi i QR kodovi, http://web.studenti.math.pmf.unizg.hr/~dmiocev/Barkod.html, datum pristupa: 15.09.2016.

[2] IOS technology overview, https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html, datum pristupa: 16.06.2016.

[3] Protocols, https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/Protocols.html, datum pristupa: 15.09.2016.

[4] AVFoundation, https://developer.apple.com/library/content/documentation/AudioVideo/Conceptual/AVFoundationPG/Articles/00_Introduction.html, datum pristupa: 16.06.2016.

[5] D.H., Steinberg, A Swift Kickstart, Dim Sum Thinking Inc., 2014.

[6] Bar kod generator, http://www.barcode-generator.org/, datum pristupa: 15.09.2016.

8. SAŽETAK

U ovom radu predstavljeno je rješenje za skeniranje i praćenje prodaje ulaznica putem iOS mobilne aplikacije. Na početku se kreira *MySQL* baza podataka koja se nalazi na poslužitelju. Da bi joj se moglo pristupiti s više uređaja je pohranjena pomoću *OpenShift* besplatnog hostinga. Pomoću *phpMyAdmin* sučelja kreirana je tablica u bazi u kojoj će biti pohranjeni podaci. Potom je napisana skripta *importer.php* za automatsko upisivanje podataka iz tekstualnog dokumenta u bazu podataka. Nakon toga, isprogramirano je aplikacijsko programsko sučelje (*API)* koje je zaduženo za komunikaciju aplikacije s bazom podataka na poslužitelju. Na kraju je kreirana aplikacija *Swift* koja obuhvaća sve procese u cjelinu. Definira izgled sučelja, dohvaća kameru, detektira i skenira bar kod te potom ispisuje rezultate na zaslon.

Ključne riječi: aplikacijsko programsko sučelje, baza podataka, iOS, skeniranje koda, Swift.

Abstract

In this work solution for scanning and tracking ticket selling through iOS mobile application is presented. First, MySQL database is created which runs on server. It is stored through OpenShift free hosting so that it could be accessed from numerous devices. With the help of phpMyAdmin interface table is created in a database which will later store the data. Afterwards, script called importer.php was made, and it was used for automated importing data from text document to database. Later, Application Programming Interface (API) is created which serves for communication between application and database which runs on server. Lastly, Swift application which combines all the processes in whole is created. It defines design of interface, catching the camera, detecting and scanning barcode and in the end writes results on the screen.

Keywords: application programming interface, database, iOS, code scanning, Swift.

9. ŽIVOTOPIS

Fran Glavota rođen je 8. rujna 1994. u Osijeku. Osnovnu školu upisao je 2001. godine u Osijeku, a završio 2009. godine. Također, paralelno je završio i Glazbenu školu Franje Kuhača u Osijeku gdje je svirao gitaru. III. Gimnaziju u Osijeku pohađao je od 2009. do 2013. godine, a potom upisao sveučilišni Preddiplomski studij elektrotehnike na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija pri Sveučilištu Josipa Jurja Strossmayera u Osijeku. U toku studija proveo je jedan semestar na Lodz University of Technology u Poljskoj u sklopu Erasmus+ projekta.

_____

vlastoručni potpis

10. PRILOZI (na CD-u)

Prilog 1. Dokument „FRANGLAVOTA_ZR“ u .docx formatu.

Prilog 2. Programski kod „Ticket Scanner“ u .zip formatu.