Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.4:621.395
DOI: https://doi.org/10.24867/01BE30Cizmar

PRIMENA IBEACON PROTOKOLA NA IOS PLATFORMI

APPLICATION OF IBEACON PROTOCOL ON IOS PLATFORM

Dorian Čizmar, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *U radu je opisan način rada bikona kao i primena i implementacija iBeacon protokola na iOS platformi. Takođe su opisani problemi kod razvoja aplikacija koje koriste bikone. Postoji implementirano rešenje u Swift programskom jeziku.*

Ključne reči: *Bikon, iBeacon protokol, iOS platforma*

Abstract – *The thesis describes working of beacons and application and implementation of iBeacon protocol on iOS platform. Also there is description of problems that can be faced during development of application that use beacons. There is an implemented software solution in Swift programming language.*

Keywords: *Beacon, iBeacon protocol, iOS platform*

1. UVOD

Mobilni telefoni su se pojavili i aktivno se koristili još krajem prošlog veka. Vremenom je tehnologija napredovala i došlo je do enormnih izmena kod mobilnih telefona, te se oni telefoni i telefoni koji se danas koriste jako razlikuju.

Današnji tzv. pametni telefoni se sastoje od procesora, memorije, operativnog sistema. Sve to omogućuje razvoj aplikacija na pametnim telefonima. Aplikacije korisniku omogućavaju interakciju sa drugim sistemima i uređajima. Jedni od tih uređaja su bikoni. Bikoni su mali uređaji koji emituju radio signal koji može biti detektovan od strane pametnog telefona.

Ovaj rad prati softversko rešenje u vidu iOS aplikacije za inventarisanje robe. Aplikacija korisniku omogućuje da pravi listu stavki za inventar i kasnije da proverava da li su sve stavke iz liste prisutne.

2. iOS

iOS je operativni sistem za mobilne i tablične uređaje, razvijen od strane *Apple*-a. Ovaj operativni sistem može da se pokreće isključivo na *iPhone* i *iPad* uređajima.

2.1 Slojevi iOS-a

Postoji 4 sloja iOS operativnog sistema [1]. Prvi sloj pod nazivom *CoreOS* je najniži sloj i odnosi se na servise vezane za hardver, npr. rad sa fajl sistemom, *bluetooth* servisi, servisi za pristup *Keychain*-u itd [1]. *Core Services* je drugi sloj i nudi servise za mrežu, lokaciju, lokalnu bazu podataka itd [1]. Treći sloj pod nazivom *Media Core* predstavlja sloj za obradu slike, zvuka i video sadržaja [1]. Najviši sloj iOS operativnog sistema jeste *CocoaTouch* i zadužen je za korisnički interfejs. Tu se nalaze dugmadi, tekstualna polja, razne kontrole itd [1].

2.2 Radno okruženje

Radno okruženje koje se uglavnom koristi za programiranje iOS aplikacija jeste Xcode. Xcode je razvijen od strane *Apple*-a.

2.3 Jezici

Postoje dva jezika za razvoj iOS aplikacija. To su Objective-C i Swift.

2.4 Swift programski jezik

Swift je najnoviji programski jezik za razvoj *iOS*, *macOS*, *watchOS* i *tvOS* aplikacija. Nudi primarne tipove kao što su *Int*, *Double*, *Float*, *Bool* i *String* [2]. Takođe postoje tri tipa kolekcija a to su *Array*, *Set* i *Dictionary* [2].

Varijable se mogu koristiti na dva načina. Kada definišemo varijablu sa ključnom reči *var* tada se ona može menjati bilo kada. Ukoliko definišemo varijablu sa *let* to znači da vrednost te varijable više ne možemo da menjamo [2].

2.5 Storyboard

XCode razvojno okruženje nudi jednostavan način generisanja grafičkog korisničkog interfejsa, preko *storyboard*-a [3]. Pomoću *storyboard*-a se može definisati korisnički interfejs za sve ekrane u aplikaciji i sve veze između njih. Postoji skup predefinisanih komponenti koje se prevlače na ekran za koji pravimo korisnički interfejs.

2.6 Softverski obrazac MVC

U današnje vreme postoje različiti pristupi kako organizovati kod i dodeliti uloge određenim komponentama sistema. Tako postoje različite softverske arhitekture za razvoj *iOS* aplikacija a to su *MVC*, *MVVM*, *MVI*, *MVP*, *VIPER* i druge. Ne postoji striktno propisana softverska arhitektura koja se mora koristiti ali *Apple* preporučuje korišćenje *MVC* softverske arhitekture. *Model* predstavlja sloj koji treba da opisuje entitete i sadrži poslovnu logiku aplikacije. *View* sloj služi za prikazivanje podataka, dok *Controller* treba da poveže prethodna dva modula.

3. BIKONI

Bikon je mali *Bluetooth Low Energy* uređaj koji emituje radio signal. Taj signal može da bude detektovan od strane mobilnih uređaja.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Stevan Gostojić, vanr. prof.

3.1 Hardverska strana i fizika bikona

Bikon je podeljen u 3 celine: *ARM* računar, Bluetooth modul i baterija [4]. *ARM* računar predstavlja centralnu procesorsku jedinicu (eng. *Central Processing Unit - CPU*) koja pokreće softver niskog nivoa programiran tako da brine o ponašanju bikona. *CPU* modul takođe sadrži i antenu koja emituje radio signal [4]. *Bluetooth* modul je zadužen za prenos podataka *bluetooth* tehnologijom. Bikoni uglavnom koriste tzv. pametni *Bluetooth* standard (eng. *Smart Bluetooth standard*). Maksimalna veličina jednog paketa prilikom prenosa podataka je 257 bajta [4]. Ova količina jeste dovoljna za slanje podataka vezanih za sam bikon. Bikon ne šalje signal konstantno, već to radi tako što šalje signal u intervalima. Princip rada bikona je jednostavan. Centralna procesorska jedinica zna koje podatke treba da pošalje na osnovu softvera koji pokreće. Antenom se šalje *Bluetooth* paket sa podacima iz ARM računara, a električna energija se snabdeva baterijom.

3.2 Vrste tehnologija bikona

Tehnologija bikona podrazumeva skup mogućnosti bikona koji mogu da se konfigurišu pre upotrebe bikona. Postoji *Apple*-ova tehnologija pod nazivom *iBeacon* i *Google*-ova koja se zove *Eddystone*.

3.3 Konfiguracija bikona

Konfiguracija bikona predstavlja podešavanje određenih parametara koji utiču na način slanja signala.

3.3.1 Konfiguracija *iBeacons*

Parametri za konfigurisanje ove tehnologije su [6]:

1. UUID – Univerzalni jedinični identifikator
2. Major – Broj od 1 do 65535
3. Minor – Broj od 1 do 65535
4. Vremenski interval – interval u kom se šalje signal
5. Jačina slanja signala (u decibelima)

3.3.2 Konfiguracija *Eddystone*

Kod konfiguracije *Eddystone* tehnologije mogu se konfigurisati 3 različita paketa [7]:

1. Eddystone-UID – definiše se *Namespace* (statički identifikator dužine 10 bajta) i *Instance* (6-bajtni tekst)
2. Eddystone-URL – definiše se URL koji će se slati uz ovaj paket
3. Eddystone-TLM – Ovaj paket se ne šalje često, a kad se šalje onda se šalje uz UID ili URL pakete, sa dodatnim informacijama o bikonu (temperatura, zdravlje baterije, verzija itd.)

3.4 Načini skeniranja

Postoje dve vrste skeniranja bikona a to su *monitoring* i *ranging* [5]. Kod monitoringa se aplikacija obaveštava kada je uređaj ušao u region koji se skenira odn. primio signal od bikona. Informacije o bikonu koji je poslao signal se ne šalju kod ove vrste skeniranja. Primera radi, kada se ulazi u sportsku prodavnicu, može se koristiti ovaj režim skeniranja da se obavesti korisnik o potencijalnim popustima na određene artikle [5]. U ovom slučaju, aplikaciji nije bitno koji tačno bikon šalje signal, već u kom se regionu nalazi uređaj. *Ranging* skeniranje pruža više detalja o bikonu koji šalje signal. Tako se mogu poslati sledeće informacije [5]:

1. UUID, minor i major
2. Tačnost udaljenosti u metrima
3. Relativnu udaljenost (enumeracija sa vrednostima: blisko, daleko, neposredna blizina i nepoznato)
4. Jačina signala (u decibelima)

4. SPECIFIKACIJA ZAHTEVA

Uz ovaj rad je implementirana iOS aplikacija čija je namena da omogući inventarisanje pokretne imovine. Ideja je da svaki element koji treba da bude inventarisan poseduje svoj bikon i da na taj način aplikacija dobija informacije o elementu koji treba da se nađe u inventaru.

4.1 Funkcionalni zahtevi

Ovi zahtevi su prikazani pomoću dijagrama slučajeva korišćenja (eng. *Use-case diagram*) (slika 1).

Slika 1. *Dijagram slučajeva korišćenja*

Najbitniji slučajevi korišćenja su:

1. Prijava u aplikaciju
2. Odjava iz aplikacije
3. Dodavanje novog inventara
4. Odabir željenog inventara
5. Dodavanje stavke u inventar
6. Brisanje stavke iz inventara
7. Izmena stavke iz inventara
8. Početak inventarisanja
9. Završetak inventarisanja
10. Pregled stavki inventara na mapi
11. Generisanje izveštaja za inventar
12. Slanje izveštaja elektronskom poštom
13. Konfigurisanje korisničkog profila

4.2 Nefunkcionalni zahtevi

Neki od nefunkcionalnih zahteva su:

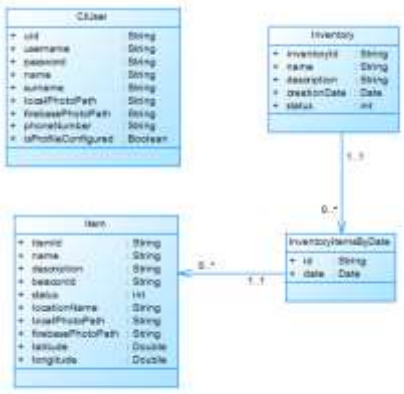
1. Instalacija aplikacije – instalacija treba da se vrši preko *XCode*-a.
2. *Offline mode* – Aplikacija treba da prikazuje postojeće inventare i bez Internet konekcije.

3. Softverski zahtevi – korisnik treba da uređaj minimalno sa iOS 11 operativnim sistemom.
4. Hardverski zahtevi – Korisnik treba da poseduje *iPhone* ili *iPad* uređaj i *Estimote* bikone.
5. Performanse – Korisnik ne sme da čeka duže od 5-6 sekundi na skeniranje bikona.
6. Sigurnost – Kredencijali korisnika treba da se čuvaju u *Keychain*-u.
7. Održivost – Aplikacija treba da je razvijena po nekoj od gore navedenih softverskih arhitektura.
8. Dodavanje korisnika u sistem – treba da se vrši isključivo na web portalu.

5. SPECIFIKACIJA DIZAJNA

Dizajn sistema je opisan sa dva tipa dijagrama. Jedni opisuju strukturu aplikacije (dijagram klasa i dijagram komponenti), dok drugi opisuje ponašanje aplikacije (dijagrami sekvenci).

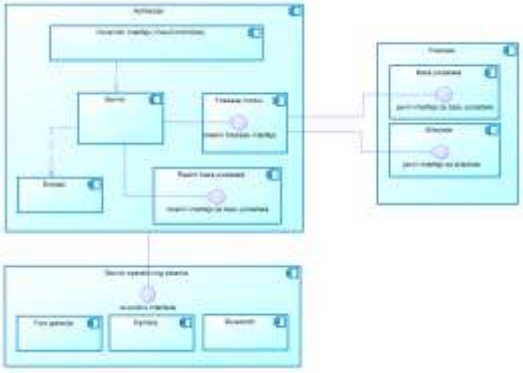
5.1 Dijagram klasa



Slika 2. *Dijagram klasa*

Dijagram klasa (slika 2.) sadrži 4 klase. Prva klasa jeste *CIUser* koja predstavlja model korisnika sa njegovim osnovnim podacima. Druga klasa modeluje inventar. To je klasa pod nazivom *Inventory*. Inventar može da se vrši više puta, te je neophodno modelovati datum inventara. Datum inventara je opisan klasom *InventoryItemsByDate*. Inventar za svaki datum može da ima listu stavki. Stavka je modelovana klasom *Item*.

5.2 Dijagram komponenti



Slika 3. *Dijagram komponenti*

Ovaj dijagram (slika 3) opisuje strukturu čitavog sistema. Postoje dve celine a to su server i mobilna aplikacija. Aplikacija koristi servise operativnog sistema kao što je bluetooth servis, kamera i foto galerija. Server je predstavljen *Google*-ovim servisom pod nazivom *Firebase*. Aplikacija se obraća *Firebase*-u preko *Firebase* modula. Taj modul parsira podatke i šalje ih servis modulu. Servis modul koristeći *Realm Database* modul pamti podatke lokalno i prikazuje ih na ekranu.

6. IMPLEMENTACIJA

U ovom odeljku je opisana implementacija bitnih delova aplikacije za inventarisanje nepokretne imovine.

6.1 Struktura projekta

Projekat je urađen po ugledu na MVC softversku arhitekturu te stoga postoje Model, View i Controller.U *Model*-u se nalazi opis entiteta, komunikacija i parsiranje zahteva i odgovora servera kao i sva poslovna logika aplikacije. U *View* sloju se prikazuju podaci pripremljeni u modelu, a ViewController spaja Model i View slojeve tako što uzme pripremljene podatke od modela i prezentuje ih u View sloju.

6.2 iBeacons u Swift programskom jeziku

Napravljena je bazna klasa *BeaconScanner* koja sadrži logiku za konfigurisanje regiona za skeniranje i generalne stvari koje su potrebne za skeniranje bikona. U toj klasi se instancira klasa koja brine o skeniranju bikona. To je klasa *ESTBeaconManager*.

6.3 Dodavanje stavke u inventar

Kada se dodaje stavka u inventar, korisnik unosi podatke preko View sloja aplikacije. Ti podaci se obrađuju u servis sloju a nakon toga se vrši dodavanje tih podataka na *Firebase*. Za dodavanje podataka na *Firebase* zadužena je klasa *FirebaseInventoryEngine* koja u sebi instancira *FirebaseDatabase* objekat koji je zadužen za komunikaciju sa serverom.

6.4 Algoritmi za skeniranje bikona

Algoritmi se nalaze u klasama *NearestBeaconScanner* i *MonitoringBeaconScanner*. Obe klase nasleđuju baznu klasu BeaconScanner i implementiraju logiku za različite potrebe skeniranja.

6.4.1 Struktura podataka za skladištenje skeniranih bikona

Struktura predstavlja modifikovan red (eng. *queue*), koji ima ograničenje za broj elemenata u redu. Ukoliko se doda novi element a red je pun, tada se iz reda izbacuje prvi dodati element.

6.4.2 Algoritam za skeniranje najbližeg bikona

Ovaj algoritam koristi *ranging* način skeniranja. Svake sekunde se dobija lista bikona i proverava se da li je bikon u neposrednoj blizini na 0,5m sa jačinom signala većom

od -75db. Ukoliko su zadovoljeni ti uslovi, bikon se dodaje u red. Ukoliko postoji više od 3 istih bikona u redu taj bikon se uzima kao skenirani bikon za stavku.

6.4.3 Algoritam za skeniranje bikona na osnovu liste identifikatora

Pre početka algoritma se definiše lista bikona koji treba da budu skenirani. Takođe se koristi *ranging* način skeniranja. Svake sekunde se dobija lista bikona. Algoritam gleda da li postoji neki bikon iz dobijene liste bikona u predefinisanoj listi bikona. Ukoliko postoji i ukoliko je udaljenost bikona između 0 i 1 tada se dodaje u red. Ukoliko postoji više od 3 istih bikona u redu može se smatrati da je bikon pronađen.

7. DEMONSTRACIJA

Na slici 4 je prikazan ekran sa listom stavki u jednom inventaru. Na tom ekranu se može vršiti dodavanje, izmena i brisanje stavke. Takođe klikom na „Start inventory“ se započinje novi inventar. Nakon završetka inventara moguće je generisati izveštaj u PDF formatu.

Slika 4. *Stavke u inventaru*

8. ZAKLJUČAK

U ovom radu je opisan princip rada bikona na iOS platformi. Dve najbitnije stvari kod razvoja aplikacije koja treba da radi sa bikonima su da se bikoni konfigurišu u skladu sa potrebama i da se osmisli odgovarajući algoritam za skeniranje bikona.

Aplikacija koja prati ovaj rad ne podržava rad više korisnika na jednom inventaru, što bi u stvarnosti trebalo da bude omogućeno, te se to može smatrati kao mana i polje gde ova aplikacija može da se unapredi. Takođe aplikacija je ograničena na rad sa *Estimote* bikonima, stoga se ovo takođe može smatrati kao mana i oblast za unapređenje.

9. LITERATURA

[1] Stanford kurs za *iOS 11* programiranje - https://docs.microsoft.com/en-us/xamarin/cross-platform/get-started/introduction-to-mobile-development (pristupljeno u septembru 2018.)

[2] Swift dokumentacija - https://docs.swift.org/swift-book/LanguageGuide/TheBasics.html (pristupljeno u septembru 2018.)

[3] Storyboard - https://developer.apple.com/library/archive/documentation/General/ Conceptual/Devpedia-CocoaApp/Storyboard.html (pristupljeno u septembru 2018.)

[4] Estimote blog o hardveru bikona - https://blog.estimote.com/post/106913675010/how-do-beacons-work-the-physics-of-beacon-tech (pristupljeno u septembru 2018.)

[5] Estimote blog o monitoringu i rangiranju - https://community.estimote.com/hc/en-us/articles/203356607-What-are-region-Monitoring-and-Ranging- (pristupljeno u septembru 2018.)

[6] Apple dokumentacija o iBeacon tehnologiji - https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf (pristupljeno u septembru 2018.)

[7] Google dokumentacija o Eddystone format - https://developers.google.com/beacons/eddystone (pristupljeno u septembru 2018.)

Kratka biografija:

Dorian Čizmar rođen je u Vrbasu 1993. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Računarstvo i automatika odbranio je 2016. god.

kontakt: 064/0456-945
mail:theory93rk@gmail.com