|  |  |
| --- | --- |
|  | Zbornik radova Fakulteta tehničkih nauka, Novi Sad |

**UDK: 004.738.5:339**

**DOI:** [**https://doi.org/10.24867/10BE34Vukasinovic**](https://doi.org/10.24867/10BE34Vukasinovic)

**APLIKACIJA ZA REVIZIJU KNJIŽNOG FONDA BIBLIOTEKE**

**APPLICATION FOR LIBRARY COLLECTION REVISION**

Ivan Vukašinović, *Fakultet tehničkih nauka, Novi Sad*

**Oblast – RAČUNARSTVO I AUTOMATIKA**

**Kratak sadržaj –** *Cilj ovog rada je da olakša evidenciju stanja knjiga radnicima u bibliotekama. Umesto dosadašnjeg ručnog popisivanja knjiga, radnici biblioteke mogu upotrebom ove mobilne aplikacije da prođu kroz policu sa njigama i skeniraju kod na poleđini svake od njih. Aplikacija omogućava kreiranje sesije skeniranja koje se mogu kreirati na osnovu autora knjiga, reda na polici ili nekog drugog kriterijuma. Nakon kreiranja sesije otvra se skener koji evidentira svaki validan kod i čuva ga u memoriji telefona. Ukoliko radnik poželi da sinhronizuje stanje sa stanjem koje ima na serveru moguće je pritiskom na dugme za slanje na server.*

**Ključne reči:** *Skener, Biblioteka, aplikacija*

**Abstract** - *The aim of this paper is to make it easier for librarians to record the condition of books. Instead of manually listing books, library workers can use this mobile application to go through the shelf with them and scan the code on the back of each of them. The application allows you to create scan sessions that can be created based on the author of the books, the order on the shelf or some other criteria. After creating a session, a scanner opens that records each valid code and stores it in the phone's memory. If the worker wishes to synchronize the state with the state he has on the server, it is possible by pressing the send button for sending to the server.*

**Keywords:** *Scanner, Library, Application*

**1. UVOD**

Jedna od softverskih industrija sa najznačajnijim rastom u poslednje vreme je mobilna industrija [1].

U ovom radu biće predstavljeno rešenje mobilne aplika­cije koja omogućava evidenciju knjiga u biblioteci skeni­ranjem koda na svakoj knjizi. Aplikacija se izvršava na iOS i Android operativnim sistemima (OS). Upotre­bom jednog koda prikazuje se mogućnost zaobilaženja barijera u pisanju dve različite aplikacije, kako je najčešće do sada bio slučaj.

Umesto dosadašnjeg načina evidencije knjiga prilikom inventara, radnici bilioteka bi mogli upotrebom ove aplikacije uštedeti znatno vreme.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**NAPOMENA:**

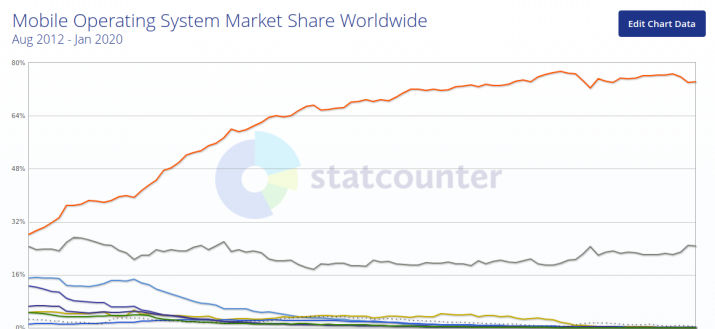
**Ovaj rad proistekao je iz master rada čiji mentor je bio dr Branko Milosavljević, red. prof.**

**2. HIBRIDNE MOBILNE APLIKACIJE**

Aplikacije koje treba pokrenuti na više platformi obično imaju različite izvorne kodove po platformi. Ovakav razvoj je kompleksan i troši znatnu količinu resursa jer se izvorni kod mora duplirati za različite platforme, što može povećati troškove projekta. Kao odgovor pojavili su se alati koji inženjerima omogućavaju da implementiraju jednu aplikaciju i „izvoze“ je na više platformi. Ova vrsta aplikacija poznata je kao cross-platform aplikacija i ima glavnu prednost u pojednostavljenju životnog ciklusa aplikacije centralizacijom razvoja u jednoj bazi koda [4].

**1.1. Razvoj operativnih sistema mobilnih uređaja**

Tržište pametnih telefona je u ekspanziji u poslednjih 10 godina. Globalna isporuka pametnih telefona je porasla sa 173.5 miliona jedinica u 2009. godini na 1404.9 miliona jedinica u 2018. Tokom ovog perioda bilo je različitih mobilnih operativnih sistema kao što stu Android, Blackberry, iOS, Symbian i Windows. Danas na tržištu dominiraju dva konkurenta, Android i iOS [2]. Kompanije koje razvijaju mobilne aplikacije imaju problem na koji operativni sistem je najbolje fokusirati se. Čak iako danas vlada tržište Android i iOS operatvnih sistema, može još neko lako preuzeti tržište. Ovo su nesigurne okolnosti koje kompanije moraju uzeti u obzir pri izboru strategije [3]. Na slici 1. prikazano je tržište zastupljenosti opera­tivnih sistema mobilnih telefona.



Slika 1. *Učešće na tržištu mobilnih operativnih sistema širom sveta od avgusta 2012. do januara 2020. [3]*

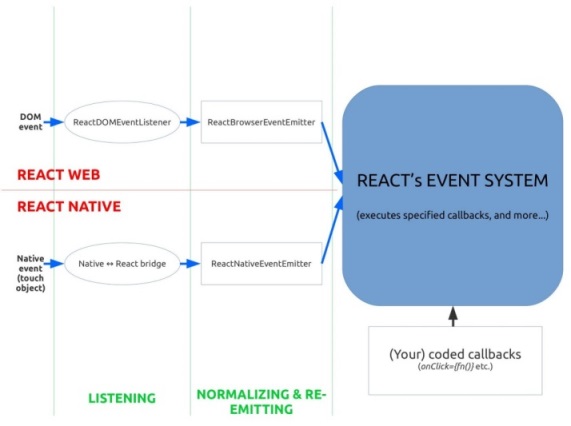
**2. REACT NATIVE**

React Native je framework (radni okvir) otvorenog koda koji je Facebook objavio 2015. godine. Godine 2018. imao je drugi po redu najveći broj saradnika u odnosu na bilo koji drugi repozitorijum na GitHub-u. Trenutno podržava Android i iOS.

Programski jezik koji se koristi za razvoj ove tehnologije je JavaScript i prikazuje korisnički interfejs pretvaranjem u matične komponente dajući aplikacijama razvijenim u React Native izvorni izgled i osećaj. UI je izgrađen sa komponentama platforme poput View, Text i Image koje se direktno preslikavaju u izvorne blokove određene platforme.

**2.1. Reakcija na događaje**

React i React Native koriste isti mehanizam kada je u pitanju reakcija na događaje (*event handling*). Koncep­tualno predstavlja veoma intuitivan proces. Svrha je presresti određene događaje (dodir ekrana, klik) i akti­virati određeni *callback koji je kreiran za tu svrhu [5].*



Slika 2. *Koraci reagovanja na događaje [5]*

Mehanizam koji React naglašava je harmonizacija: *cross-browser* za React i *cross-platform* za React Native. Sistem reagovanja na događaje je isti. Oba sistema (DOM i Native) reagovanja na događaje koriste isti kod [5].

Za React Native događaji se primaju preko mosta koji povezuje native kod sa React-om. Ukratko, kad god se stvori prikaz, React takođe prenosi svoj ID broj na native deo, kako bi mogao da primi sve događaje koji se odnose na taj element.

**2.2. Pristup senzorima**

Jedan od benefita rada u React Native-u je to što dopušta pozivanje matičnih funkcija uređaja uz pomoć JavaScript-a. Problem koji se javlja kada želimo da pristupim određenim senzorima upotrebom ove tehnologije je taj što su obično svi pristupi usko vezani za određenu platformu (iOS ili Android).

Često je potrebno doći na nivo ispod u apstrakciji kako bismo pristupili određenim funkcijama senzora.

Ukoliko je potrebno koristiti žiroskop (Gyroscope) senzor, nailazi se na problem kod podešavanja za obe platforme.

Međutim, već postoje biblioteke koje rešavaju ovaj problem i pružaju API koji je jednostavniji i intuitivniji. Jedna od takvih biblioteka je *react-native-sensors* [6].

Ova biblioteka pruža RxJS bazirani interfejs za akcelerometar, žiroskop i magnetometar.

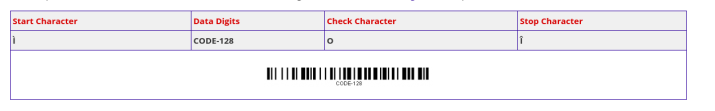
**3. IMPLEMENTACIJA**

Aplikacija je izrađena u React Native tehnologiji sa Expo podrškom. Expo je doveo u velikoj meri do ranijeg završetka aplikacije pruživši podršku za korišćenje kamere, vibracije i besplatno objavljivanje aplikacije na svojim serverima tako da je moguće bilo testirati *release* verziju u više navrata.

**3.1. Code 128**

Code 128 predstavlja jedan od standarda kodiranja teksta, brojeva, brojne funkcije i čitav skup ASCII karaktera. Obično se koristi za nekoliko primena. Naziva se još i ISBT-128, GS1-128, UCC-128, EAN-128 i USS Code 128 [7]. On sadrži 106 različitih štampanih obrazaca barkodova. Svaki odštampani bar kod može imati jedno od tri različita značenja, u zavisnosti od toga koji se od skupova znakova koristi, uz dostupnost tri različita početna znaka Code 128 za programiranje početnog skupa znakova. Funkcije su takođe obezbeđene u simbolici barkoda za prebacivanje između skupova znakova i kodiranje aplikacionih identifikatora. Barkod Code 128 može biti složen za upotrebu zbog različitih skupova znakova, što je osnovni razlog što IDAutomation pruža automatsku funkciju code 128, „Code128()“ u nekoliko alata za font, komponente i aplikacije [7].

Kompletan code 128 je sačinjen od početnog znaka, cifara podataka, modula 103 kontrolne cifre i znaka zaustavljanja što se vidi na slici 3.



Slika 3. *primer barkoda code 128*

**3.2. Skener**

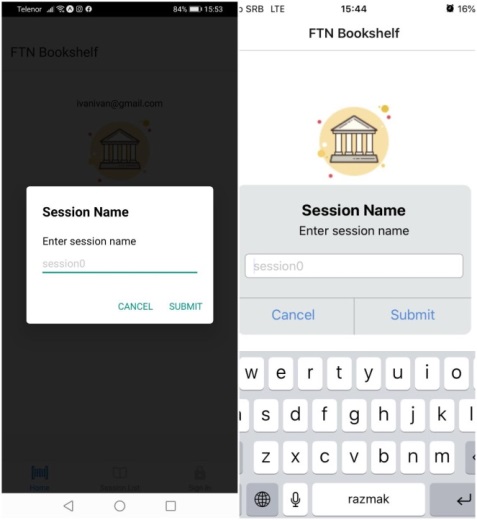
Izborom dugmeta za skeniranje prikazuje se modal u kojem se može izabrati naziv sesije skeniranja ili se može ostaviti prazno nakon čega aplikacija bira naziv na osnovu trenutne liste u memoriji (Slika 4).

U ovom trenutku je moguće i otkazati skeniranje pritiskom na dugme Cancel.

Odabirom dugmeta „Submit“ prelazi se na ekran sa ske­nerom. Skener je implementiran upotrebom Expo paketa expo-barcode-scanner koji je u znatnoj meri olakšao implementaciju. Ovaj paket podržava i druge vrste kodo­va kao što su qr, rss14, aztec, code 39,..

U nastavku je prikazan izvorni kod koji predstavlja kom­ponentu za skeniranje barkoda. BarCodeScanner kompo­nenta prima više parametara, među njima su callback funkcija onBarCodeScanned koja se aktivira kada se uspešno skenira kod i tipovi kodova koji su podržani kroz polje barCodeTypes.

Takođe u ovom fajlu se nalazi i SnackBar komponenta čija je uloga da se na kratko prikaže poruka ukoliko je kod uspešno skeniran ili poruka greške ukoliko knjiga već postoji u memoriji (Slika 6).



Slika 4. *Modal za unos naziva sesije*

Navedene poruke prati vibracija, pa tako ukoliko dođe do uspešnog skeniranja duža vibracija obavesti korisnika, dok ukoliko je došlo do greške, više kratkih vibracija označi da se desila greška.

*<View*

*style={{*

*flex: 1,*

*flexDirection: 'column',*

*justifyContent: 'flex-end'*

*}}*

*>*

*<Button title="Finish" onPress={() => finishScannig()} />*

*<BarCodeScanner*

*onBarCodeScanned={scanned ? undefined : handleBarCodeScanned}*

*barCodeTypes={[BarCodeScanner.Constants.BarCodeType.code128]}*

*style={{ ...StyleSheet.absoluteFillObject, bottom: 30 }}>*

*position="top"*

*/>*

*</View>*

Listing 1. *Prikaz izvornog koda skenera*

**4.3. Lista skeniranih sesija i knjiga**

Završetak skeniranja se obavlja pritiskom na dugme „FINISH“. Nakon toga pritiskom na drugi tab na dnu „Session List“ prebacuje se na ekran sa listom sesija. Lista sesija prikazana je na slici 5.

Na ovom ekranu vidimo prikaz svih sesija i dobijamo informaciju koliko knjiga ima u sesiji, naziv sesije, kada je kreirana i da li je sinhronizovana (poslata na server).

Na slici 5 prikazano je kako izgleda lista kod koje ni jedna sesija nije poslata na server.

Krug sa leve strane na svakoj sesiji obojen je sivom bojom, a sa desne strane vidimo ikonicu koja služi za slanje sesije na server.



Slika 5. *Lista sesija*

A screenshot of a cell phone

Description automatically generated

Slika 6. *Skeniranje koda*

**5. ZAKLJUČAK**

Postoji mnogo mogućnosti za proširenje ovog rešenja. Ukoliko bi se proširo entitet korisnika, dodali novi entiteti i povezali sa bazom podataka stvarne biblioteke, sistem bi se mogao prilagoditi da podržava više mogućih scenarija.

Rešenje može u velikoj meri da olakša posao radnicima biblioteka, s obzirom da ne bi bilo potreban prolazak i upisivanje svake knjige kao što je bio slučaj do sad prilikom inventara.

**6. LITERATURA**

[1] A Students’ Perspective of Native and Cross-Platform Approaches for Mobile Application Development

[2] Holst, A. (2019) Global smartphone shipments forecast from 2010 to 2022 (https://www.statista.com/statistics/263441/global-smartphone-shipmentsforecast/ [2020-02-05 14:12])

[3] Statcounter (2020) Mobile Operating System Market Share Worldwide Aug 2012– Jan 2020

[4] Bernardes, T.F., Miyake, M.Y.: Cross-platform mobile development approaches: a systematic review. IEEE Lat. Am. Trans. 14(4), 1892–1898 (2016)

[5] The React and React Native Event System Explained: A Harmonious Coexistence

[6] react-native-sensors (<https://github.com/react-native-sensors/react-native-sensors>)

[7] Code 128/GS1-128 Barcode FAQ & Tutorial (<https://www.barcodefaq.com/1d/code-128/>)

**Kratka biografija:**

|  |  |
| --- | --- |
|  | **Ivan Vukašinović** rođen je u Prištini 1995. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Primenjene računarske nauke odbranio je 2018.god.  kontakt: ivanvukasino@gmail.com |