StudentKit - aplikacija za studente grada Zagreba temeljena na Android operacijskom sustavu

Genzić, Vedran

SVEUČILIŠTE U ZAGREBU

FAKULTET PROMETNIH ZNANOSTI

Vedran Genzić

STUDENTKIT – APLIKACIJA ZA STUDENTE GRADA ZAGREBA TEMELJENA NA ANDROID OPERACIJSKOM SUSTAVU

ZAVRŠNI RAD

Zagreb, 2020.

Sveučilište u Zagrebu

Fakultet prometnih znanosti

ZAVRŠNI RAD

STUDENTKIT – APLIKACIJA ZA STUDENTE GRADA ZAGREBA TEMELJENA NA ANDROID OPERACIJSKOM SUSTAVU

STUDENTKIT – ANDROID OPERATING SYSTEM BASED APPLICATION MADE FOR ZAGREB STUDENTS

Mentor: prof. dr. sc. Dragan Peraković

Student: Vedran Genzić

JMBAG: 0135246355

Zagreb, rujan 2020.

SAŽETAK

Glavna tema ovog rada je opis izrade mobilne aplikacije koja studentima grada Zagreba olakšava svakodnevno izvršavanje obaveza na fakultetu, korištenjem React Native radnog okvira za razvoj aplikacija koji koristi JavaScript programski jezik unutar tekstualnog uređivača Visual Studio. Ostatak rada opis je razvoja Android OS-a, njegove arhitekture te tehnologija koje se koriste za razvoj mobilnih aplikacija.

Ključne riječi: Android, React Native, mobilna aplikacija

SUMMARY

The main topic of this paper is a description of creating a mobile application that makes it easier for students of the city of Zagreb to perform their daily duties at the faculty, using the React Native framework for application development that uses JavaScript programming language within the text editor Visual Studio. The rest of the paper is a description of the development of the Android OS, its architecture and the technologies used to develop mobile applications.

Key words: Android, React Native, mobile application

SADRŽAJ

1. UVOD

S godinama razvoj pametnih telefona (eng. *smartphone*) je sve veći i većina stvari koje su se prije mogle obaviti samo na računalu potpuno je prilagođena pametnom telefonu. Samim rastom i razvojem povećava se broj aplikacija koje olakšavaju ljudski život. Današnje aplikacije omogućuju da se u par klikova dobije informacija ili obavi neki zadatak za koji je prije trebalo puno više vremena.

Kako bi se aplikacije prilagodile korisniku i njegovom uređaju moraju biti razvijene na stabilnom operacijskom sustavu kao što je Android operacijski sustav (eng. *Operating system* - OS) ili iOS uz korištenje kvalitetnih i stabilnih tehnologija.

Glavni cilj i zamisao ovog rada je pojašnjenje tehnologija i sustava za izradu aplikacija mobilnih uređaja te sama izrada aplikacije “StudentKit” koja bi svakom studentu, trenutno na području grada Zagreba, a uz nadogradnje u budućnosti i na širem području olakšao studiranje uz pomoć funkcionalnosti koje će biti detaljnije objašnjene u radu.

Ovaj se rad sastoji od pet cjelina u kojima se pobliže objašnjavaju korištene tehnologije i sama izrada aplikacije. Radi se o sljedećem cjelinama:

1. Uvod,

2. Značajke Android operacijskih sustava,

3.Popularni radni okviri za razvoj mobilnih aplikacija,

4. Postupak razvoja aplikacije StudentKit,

5. Zaključak.

Drugim poglavljem pobliže je opisan i objašnjen Android OS. Uz sami početak razvoja sustava opisuje se i arhitektura Android OS-a te se navode sve njegove verzije od trenutka nastajanja pa sve do danas. Također, Android OS uspoređuje se i s ostalim operacijskim sustavima na tržištu, a ponajviše s najvećim konkurentom na tržištu Apple-ovim iOS-om.

U trećem poglavlju predstavljeni su popularni radni okviri za razvoj mobilnih aplikacija i njihove vrste s obzirom na tip aplikacije za koje se koriste. Isto tako biti će uspoređen razvoj izvornih s razvojem hibridnih aplikacija.

Četvrtim poglavljem objašnjene su tehnologije React Native i Expo koje su korištene za izradu aplikacije “StudentKit”. Također, unutar istog poglavlja opisuju se svi dijelovi

aplikacije sa izgledom, funkcionalnostima i kodom koji je potreban za izvođenje istih. Na samom kraju objašnjen je sustav za pohranu podataka odnosno baza podataka „AsyncStorage“.

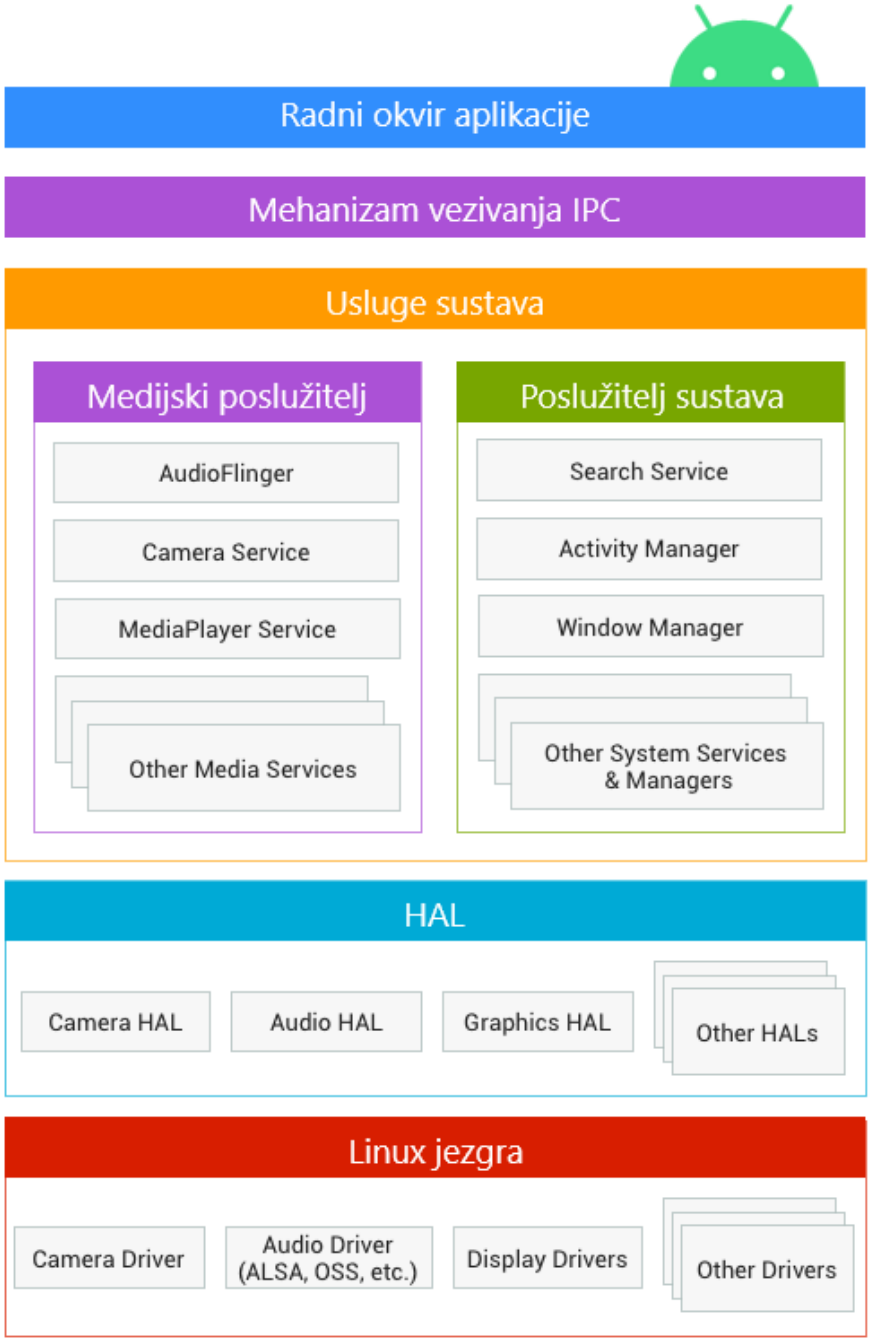
2. ZNAČAJKE ANDROID OPERACIJSKIH SUSTAVA

Jedan od osnivača Android OS-a Andy Rubin u govoru koji je održao u Tokiju 2013. godine otkrio je da je Android OS izvorno namijenjen poboljšanju operacijskih sustava digitalnih fotoaparata, [1]. Tvrtka je 2004. investitorima prikazala kako će se Android instaliran na kameri bežično povezati s računalom. To bi se računalo tada povezalo u Android *Datacenter* gdje bi vlasnici fotoaparata mogli svoje fotografije pohraniti na poslužitelj u “oblaku“.

Nekoliko mjeseci nakon govora zbog stalnog pada tržišta digitalnih fotoaparata, tvrtka Android svoj OS počinje koristiti za mobilne uređaje, a 2005. godine tvrtka je kupljena od strane američke tvrtke Google za otprilike pedeset milijuna američkih dolara. Rubin i ostali osnivači ostaju u tvrtki kako bi nastavili s razvojem OS-a.

2.1 Arhitektura Android OS-a

Android OS je projekt otvorenog koda odnosno koda koji je dostupan široj javnosti za uporabu u bilo koje svrhe ili promjene izvornog stanja. S takvim prilagodljivim izvornim kodom i dokumentacijom koja je dostupna svima može se prenijeti na gotovo svaki uređaj, [2]. Android OS je skup softverskih komponenti čija arhitektura sadrži pet slojeva koji su prikazani na slici 1 sa različitim zadaćama koje će biti objašnjene u nastavku.



Slika 1. Arhitektura sustava, [3]

- **Radni okvir aplikacije –** Radni okvir aplikacije najčešće koriste programeri kod razvoja aplikacija. Sadrži velik broj aplikacijskih programskih sučelja (eng. *Application programming interface* - API) od kojih većina omogućava direktno povezivanje na sloj apstrakcije sklopovlja.

- **Mehanizam vezivanja IPC** (eng. *Inter-Process Communication*) - Mehanizam vezivanja IPC omogućava radnom okviru aplikacije da kod idućeg sloja pozove usluge sustava. To omogućava API-jima radnih okvira da stupaju u interakciju s uslugama sustava.

- **Usluge sustava -** Usluge sustava su modularne i fokusirane komponente kao što su Window Manager, Usluga pretraživanja ili Upravitelj obavijesti. Funkcionalnost izložena API-jima radnog okivra aplikacije komunicira sa uslugama sustava za pristup temeljnom hardveru. Android uključuje dvije grupe usluga: sustava (kao što su Window Manager i upravitelj obavijesti) i medija (usluge uključene u reprodukciju i snimanje medija).

- **Sloj apstrakcije sklopovlja** (eng. *Hardware abstraction layer* - HAL) – HAL definira standardno sučelje koje dobavljači hardvera mogu implementirati što omogućava Android OS-u da ne mari za implementaciju nižih upravljačkih programa. Upotreba HAL-a omogućava implementiranje funkcionalnosti bez utjecaja ili izmjene sustava više razine. Implementacije HAL-a spakirane su u module i Android OS ih učitava u vrijeme kada je to potrebno.

- **Linux jezgra –** Razvoj upravljačkih programa za mobilne uređaje sličan je razvoju uobičajenog upravljačkog programa za Linux OS. Android OS koristi verziju Linux jezgre s nekoliko posebnih dodataka kao što su *Low Memory Killer* (sustav upravljanja memorijom koji je agresivniji u očuvanju memorije), *wake locks* (*PowerManager* usluga sustava), *Mehanizam vezivanja IPC* i ostale važne značajke za mobilnu ugrađenu platformu. Ovi dodaci primarno se odnose na funkcionalnost sustava i ne utječu na razvoj pokretačkih programa. Možete koristiti bilo koju verziju kernela sve dok podržava potrebne značajke (poput mehanizma za vezivanje), [3].

.

2.2 Verzije Android OS-a

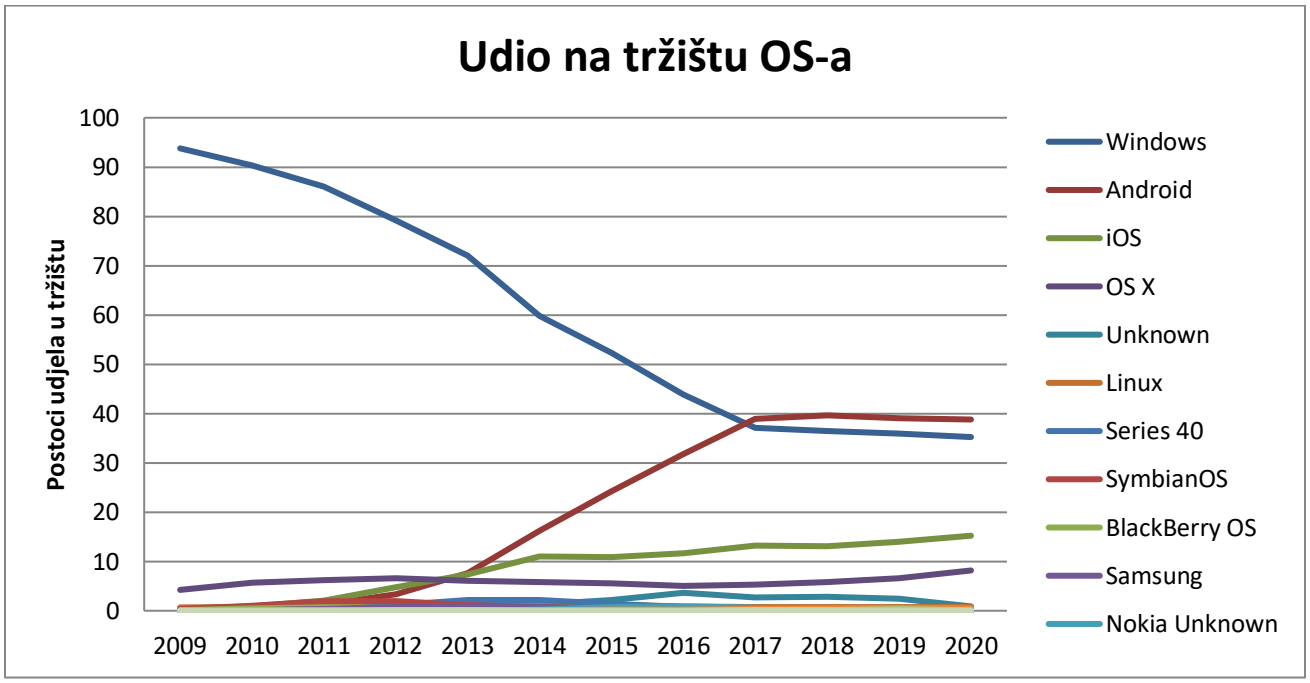
Google je beta verziju Android OS-a izbacio 2007. godine, a u rujnu 2008. najavljen je prvi Android pametni telefon T-Mobile G1 u drugim dijelovima svijeta poznat i kao HTC Dream. Prodaja istog započela je u Sjedinjenim Američkim Državama u listopadu te godine.Voditelj projekta razvitka Android OS-a Ryan Gibson osmislio je imenovanje verzija sustava uz pomoć naziva slastičarskih proizvoda. Počevši od Android OS 1.5 verzije pod nazivom Cupcake. Takav tip imenovanja verzija zadržao se do 2019, [4]. Verzije i datumi izlaska verzija nakon prvotne Android OS 1.0 verzije prikazane su u tablici 1:

Tablica 1. Android OS verzije, [5]

Naziv	Broj verzije	Datum izlaska verzije
Cupcake	1.5	27.4.2009.
Donut	1.6	15.9.2009.
Eclair	2.0 – 2.1	26.10.2009.
Froyo	2.2 – 2.2.3	20.5.2010.
Gingerbread	2.3 – 2.3.7	6.12.2010.
Honeycomb	3.0 – 3.2.6	22.2.2011.
Ice Cream Sandwich	4.0 – 4.0.4	18.10.2011.
Jelly Bean	4.1 – 4.3.1	9.7.2012.
KitKat	4.4 – 4.4.4	31.10.2013.
Lollipop	5.0 – 5.1.1	12.11.2014.
Marshmallow	6.0 – 6.0.1	5.10.2015.
Oreo	8.0 – 8.1	21.8.2017
Pie	9	6.8.2018.
Android 10	10	3.9.2019.

2.3 Usporedba Android OS-a i ostalih operacijskih sustava

Android OS tijekom godina postiže najveći udio na tržištu operacijskih sustava, ne samo unutar svijeta mobilnih uređaja, nego je uz operacijski sustav koji koriste stolna računala pod nazivom Windows najpopularniji na svijetu. Grafom 1 prikazan je rast Android OS-a unazad zadnjih 11 godina i njegovo najnovije stanje u odnosu na ostale OS.



Graf 1. Udio na tržištu OS-a – postotak udjela OS-a svake godine, [6]

U svijetu OS-a za mobilne uređaje trenutno jedini "pravi" konkurent Android OS-u je OS američke tvrtke Apple koja razvija svoj OS pod nazivom iOS. Za razliku od Android OS-a iOS se može koristiti isključivo na mobilnim uređajima tvrtke Apple, a to je i jedna od glavnih činjenica koje se odnose na veći broj korisnika Android OS-a. Ostale ključne razlike između njih prikazane su tabilicom 2.

Tablica 2. Usporedba Android OS-a i iOS-a, [7]

	Android OS	iOS
Prilagodljivost	Moguća promjena većine stvari na uređaju	Ograničen osim u slučaju *jailbreaka*
Tip koda	Otvoreni kod	Zatvoreni, ali neke komponente sadrže kod otvorenog tipa
Prijenos podataka	Lakše od iOS-a. Fotografije se mogu prenijeti putem USB-a bez aplikacija	Medijske datoteke mogu se prenijeti pomoću iTunes desktop aplikacije. Fotografije se mogu prenijeti putem USB-a bez aplikacija
Widget-i	Da, osim na zaključanom zaslonu	Ne, osim u centru za obavijesti
Pretraga interneta	Google Chrome (dostupni su i drugi preglednici). Bilo koja aplikacija preglednika može se postaviti kao zadana.	Safari (Dostupni su i ostali preglednici, ali se ne mogu postaviti kao zadani)
Usluga *web* karti	Google Maps	Apple Maps, zadano od iOS-a vezije 6 (Google Maps dostupne su i putem zasebnog preuzimanja aplikacije, ali ne kao zadane, prethodno korištene u iOS-u 5 i starijim verzijama)
Dostupni jezici	100+ jezika	34 jezika
Video poziv	Google Duo i druge aplikacije treće strane	FaceTime (samo Apple uređaji) i druge aplikacije treće strane
Virtualni asistent	Google Assistant	Siri
Dostupno na	Velik broj telefona i tableta. Glavni proizvođači poput Samsung, Oppo, OnePlus, Vivo, Honor i Xiaomi. Uređaji Android One čisti su Android. Pixel liniju uređaja izrađuje Google, koristeći gotovo čistu verziju Androida	iPod Touch, iPhone, iPad, Apple TV (druga i treća generacija)

Trgovina aplikacija, pristupačnost i sučelje	Google Play Store - 2,000,000+ aplikacija. Druge trgovine aplikacija kao Amazon i Aptoide također distribuiraju Androidove aplikacije. Aplikacije koje sadrže virus su rijetke, ali postoje.	Apple App Store – 1,000,000+ aplikacija. Aplikacija koja sadrži virus je rijetka ili nepostojeća.
Alternative trgovine aplikacije	Nekoliko alternativnih trgovina aplikacija, osim službene trgovine Google Play. (npr. Aptoide, Galaxy Apps)	Apple blokira trgovine aplikacija treće strane. Moguće je preuzeti aplikacije iz drugih trgovina samo ako je uređaj *jaibreak*-an.
Vijek trajanja baterije	Mnogi, ali ne i svi proizvođači Android mobilnih uređaja svoje uređaje opremljuju velikim baterijama s duljim vijekom trajanja.	Apple baterije obično nisu tako velike kao najveće Android baterije. Apple uspijeva održati pristojan vijek trajanja baterije putem hardverske / softverske optimizacije.
Organizator datoteka	Da. (Običan organizator za datoteke uključen na Android uređajima)	Ograničena i manja korisnost.
Fotografije i Video sigurnosna kopija	Dostupne su aplikacije za automatsko sigurnosno kopiranje fotografija i videozapisa. Google Photos omogućuje neograničen broj sigurnosnih kopija fotografija komprimirane kvalitete. OneDrive, Amazon Photos i Dropbox su druge alternative.	Za do 5 GB fotografija i videozapisa može se automatski izraditi sigurnosna kopija s iCloud-om ili više uz plaćeni iCloud prostor za pohranu. Svi ostali dobavljači kao što su Google, Amazon, Dropbox, Flickr i Microsoft imaju aplikacije za automatsko sigurnosno kopiranje za iOS i Android.
Cloud usluge	Izvorna integracija sa Google Drive skladištem.15 GB besplatno, 2 USD/mjesečno za 100 GB, 1 TB za 10 USD. Aplikacije dostupne za Amazon Photos, OneDrive i Dropbox.	Izvorna integracija sa iCloud-om. 5 GB besplatno, 50 GB za 1 USD/mjesečno, 200 GB za 3 USD/mjesečno, 1 TB za 10 USD/mjesečno. Aplikacije dostupne za Google Drive, Google Photos, Amazon Photos, OneDrive i Dropbox.
Sučelje	Zaslon na dodir	Zaslon na dodir

Biometrija Autentifikacija	Otisak prsta i / ili provjera autentičnosti lica. Dostupnost ovisi o hardveru proizvođača.	Otisak prsta ili provjera autentičnosti lica. Otisak prsta dostupan na iPhone-u (5s i noviji) i iPadu (Air 2 i noviji), ali ne i na iPhoneu X i novijim uređajima. Provjera autentičnosti lica dostupna na iPhone-u X i novijim, zamjenjujući otisak prsta
OS	Linux	OS X, UNIX
Utor za slušalice	Neki trenutni Android pametni telefoni imaju, a mnogi ne.	Na iPhoneu 7 i novijima tzv. *lightning* utor do 3,5 mm više ne dolazi s mobilnim uređajima nakon iPhone XS
Sigurnost	Mjesečna sigurnosna ažuriranja. Android zakrpe najbrže su dostupne korisnicima Pixel uređaja. Proizvođači obično zaostaju u izbacivanju ovih ažuriranja Tako da u bilo kojem trenutku velika većina Android uređaja pokreće zastarjeli softver OS.	Povremene sigurnosne nadopune. Sigurnosne prijetnje su rijetke, jer je iOS *proprietary* sustav, a preuzimanje aplikacija iz App Store-a je komplicirano.
Pozivi i poruke	Google Messages, aplikacije treće strane kao Facebook Messenger, WhatsApp, Google Duo, Discord i Skype rade na Android OS-u i na iOS-u	iMessage, FaceTime (samo s ostalim Apple uređajima), aplikacije treće strane kao Google Hangouts, Facebook Messenger, WhatsApp, Google Duo, Discord i Skype rade na Android OS-u i na iOS-u

3. POPULARNI RADNI OKVIRI ZA RAZVOJ MOBILNIH APLIKACIJA

Mobilne aplikacije su ključan faktor mobilnih uređaja i samim time stalan razvoj novijih i boljih je neizbježan. Kako bi se aplikacije mogle razvijati za to su potrebni razni radni okviri čiji broj s godinama raste.

U ovom poglavlju biti će opisane kategorije radnih okvira za razvoj mobilnih aplikacija, trenutno najkorišteniji i najbolji radni okviri te razlika kod razvoja aplikacija pomoću njih i razvoja pomoću integriranog razvojnog okruženja *(*eng. *Integrated Development Environment - IDE)* pod nazivom Android Studio*.*

3.1 Vrste radnih okvira

Kod razvitka mobilnih aplikacija postoje različiti radni okviri koji se mogu podijeliti u tri kategorije:

1. Izvorni radni okviri za razvoj specifičan za platformu
2. Radni okviri za razvoj mobilnih web aplikacija
3. Radni okviri za razvoj hibridnih aplikacija

3.1.1 Izvorni radni okviri za razvoj specifičan za platformu

Koriste se za razvoj aplikacija koje se nalaze na uređaju, a njima se pristupa putem ikona na zaslonu uređaja. Izvorne aplikacije instaliraju se putem trgovine aplikacija poput Google Play u slučaju Android OS-a ili App Store kod iOS-a. Razvijene su posebno za jednu platformu i mogu u potpunosti iskoristiti sve značajke uređaja, odnsono mogu koristiti kameru, GPS (eng. *Global Positioning System*), akcelerometar, kompas, popis kontakata itd. Mogu im se ugraditi i geste, bilo standardne geste operacijskog sustava ili nove geste definirane aplikacijom, [8].

Tehnologije koje se koriste kod ovakvih radnih okvira:

- **Objective-C** - Objective-C primarni je programski jezik koji se koristi prilikom pisanja softvera za OS X i iOS. To je superset programskog jezika C te pruža objektno orijentirane mogućnosti i dinamično vrijeme izvođenja. Objective-C ima značajke programskog jezika C i na to dodaje sintaksu za definiranje klasa i metoda. Također dodaje podršku na razini

jezika za upravljanje objektnim grafikonima i literalima objekata, istovremeno pružajući dinamično tipkanje i uvezivanje, odgađajući brojne odgovornosti do izvršavanja, [9].

- **Swift** - Swift je relativno novi radni okvir koji je Apple uveo i koji je postao alternativa za izgradnju izvornih aplikacija za iOS, [10].

- **Java** - Java je programski jezik koji se ne upotrebljava prvenstveno za Android mobilne aplikacije, već i u druge svrhe. Veliki dio razvoja za radne površine i weba zasnovan je na Javi. Međutim, aplikacije na Javi zahtijevaju više memorije i rade sporije u odnosu na druge okvire, [11].

- **Kotlin** - Kotlin se može koristiti za bilo koju vrstu razvoja, bilo da se radi o strani poslužitelja, strani klijenta ili Android OS-u. Kotlin se koristi za mobilne aplikacije i aplikacije na strani poslužitelja, na strani klijenta s JavaScriptom ili JavaFX-om te za znanost o podacima, [12].

3.1.2 Radni okviri za razvoj mobilnih web aplikacija

Radni okviri za mobilne web aplikacije su druga kategorija radnih okvira i ona se odnosi na razvoj aplikacija koje nisu stvarne aplikacije nego su to zaista web stranice koje na prvu izgledaju kao izvorne aplikacije, ali ne rade kao takve. Njih vodi preglednik i obično se pišu u HTML-u (eng. *Hypertext Markup Language*). Korisnici prvo pristupaju njima kao što bi pristupili bilo kojoj web stranici te se navigiraju do posebnog URL-a (eng. *Universal Resource Locator*), a potom imaju mogućnost instalirati aplikaciju na početni zaslon stvaranjem oznake na toj stranici, [8].

Za razvoj ovih aplikacija koriste se tehnologije kao što su:

- **JavaScript** – JavaScript objektni je skriptni jezik jednostavan za korištenje, dizajniran za stvaranje web aplikacija koje povezuju objekte i resurse na strani klijenta kao i na strani poslužitelja, [13].

- **CSS** (eng. *Cascading Style Sheets*) – CSS stilski je jezik koji se koristi za opisivanje odnosno predstavljanje dokumenta napisanog u HTML-u. CSS opisuje kako se elementi trebaju prikazati na zaslonu, papiru te u govoru ili drugim medijima, [14].

- **HTML** – HTML definira značenje i strukturu web sadržaja, [15].

3.1.3 Radni okviri za razvoj hibridnih aplikacija

Hibridne aplikacije kombiniraju značajke izvornog i mobilnog radnog okvira za web aplikacije. Kao i izvorne aplikacije, one se nalaze u trgovini aplikacija i mogu koristiti brojne dostupne značajke uređaja. Kao i web aplikacije oslanjaju se na prikazivanje HTML-a u pregledniku s naznakom da je preglednik ugrađen u aplikaciju. Tvrtke grade hibridne aplikacije oko postojećih web stranica i na taj način pokušavaju se ostvariti prisutnost u trgovini aplikacija, a da ne troše značajne napore na razvoj drugačije aplikacije. Hibridne su aplikacije također popularne jer omogućuju razvoj na više platformi i na taj način značajno smanjuju troškove razvoja, [8].

Najpopularnije tehnologije koje se koriste kod ovakvih radnih okvira:

- **React Native** - React Native omogućuje korištenje radnog okvira React (koji se koristi za izradu frontenda na web-u) pri izradi korisničkih sučelja za Android OS i iOS, [16].

- **Xamarin** - Xamarin je radni okvir koji omogućuje izradu hibridnih aplikacija korištenjem programskog jezika, [17].

- **Ionic** - Ionic radni okvir je alat za izgradnju aplikacija za mobilne uređaje na više platformi. Omogućuje korištenje raznih radnih okvira za izradu korisničkih sučelja poput Reacta, Vuea, Angulara te korištenje običnog JavaScripta, [18].

3.2 Usporedba u razvoju hibridnih i izvornih aplikacija

U prošlom potpoglavlju spomenuta je glavna razlika između hibridnih i izvornih aplikacija, a to je mogućnost rada hibridnih aplikacija neovisno o platformi, dok kod izvornih aplikacija to nije slučaj. Uz glavnu razliku postoje prednosti kod svake vrste razvoja koje se nalaze u tablici 3.

Tablica 3. Prednosti u razvoju

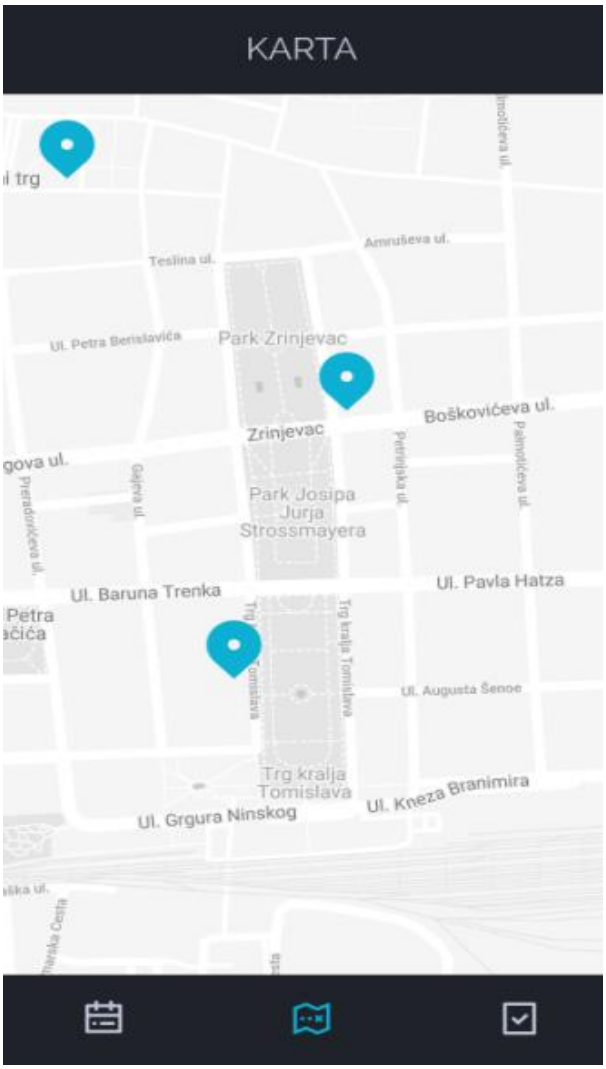
Razvoj izvornih aplikacija	Razvoj hibridnih aplikacija
Prednosti	
Brzina i stabilnost	Kraće vrijeme razvoja
Vijek trajanja životnog ciklusa	Veći broj korisnika
Skalabilnost	Usklađenost pri ažuriranju
Nema ograničenja glede arhitekture i funkcionalnosti	70 - 90% koda može se ponovo iskoristiti
Sučelje napravljeno u potpunosti u skladu s platformom	Veća održavljivost

Izvorna mobilna aplikacija mora ispuniti zahtjeve određenog operativnog sustava pomoću SDK-a (eng. *Software Development Kit*), korištenih tehnologija kao i memorije, kamere, senzora i drugih programa instaliranih na uređaju dok su hibridne aplikacije kompatibilne s više operativnih sustava te se stoga može pokretati na bilo kojem pametnom telefonu, tabletu, računalu, pametnom satu ili povezanom televizoru, [19].

4. POSTUPAK RAZVOJA APLIKACIJE STUDENTKIT

Mobilna aplikacija “StudentKit” je skup alata čija je namjena olakšavanje odrade svakodnevnih zadataka studentima na fakultetu kao što su praćenje dolaznosti na predavanja, bilježenje zadataka koje trebaju obaviti te pronalazak najbližih menzi. Ovom aplikacijom može se služiti svaki student na području grada Zagreba, a uz dodatne nadogradnje moguće je proširenje tog područja.

Početni zaslon vidljiv je na slici 2. Ova aplikacija izrađena je u “Visual Studio Code” tekstualnom uređivaču korištenjem radnog okvir “React Native” koji je spomenut u prethodnom poglavlju. Uz to korišten je skup alata i usluga “Expo” koji je izgrađen oko React Native-a. Detaljni opis svega navedenog biti će proveden u nastavku.



Slika 2. Početni zaslon aplikacije

4.1 React Native

React Native je, kao što je u prethodnom poglavlju rečeno, radni okvir za izradu hibridnih aplikacija temeljen na “React” radnom okviru koji se koristi za izradu frontend dijela web stranica koji koristi programski jezik JavaScript.

4.1.1 Komponente temeljem funkcija i klasa

Pomoću React Native-a mogu se izrađivati komponente koristeći klase ili funkcije. Izvorno su komponente klase bile jedine komponente koje su mogle imati stanje, no od uvođenja React Hooks API-ja komponentama se može dodati stanje i više, [16].

Objekt stanja služi za pohranu vrijednosti svojstava koje pripadaju određenoj komponenti, a promjenom tog stanja komponenta se ponovno prikazuje, [20].

Hooks su funkcije koje omogućuju da se spoje značajke stanja i životnog ciklusa iz komponenti funkcija, ne rade unutar klase te se uz njih koristi React bez klasa. React Hooks API omogućuje izvlačenje logike iz komponente tako da se može samostalno testirati i ponovno upotrijebiti te omogućuje ponovnu upotrebu logike bez promjene hijerarhije komponente. To olakšava dijeljenje *Hooks-a* među komponentama. React Hooks koristi više značajki React Native-a bez klasa. Komponente su uvijek bile bliže funkcijama, a React Hooks obuhvaća funkcije, ali bez žrtvovanja praktičnosti, [21].

```
import React, { useState } from 'react';

function Example() {
  // Declare a new state variable, which we'll call "count"
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

Slika 3. „useState“ *hook*, [21]

Na slici 3 nalazi se primjer i funkcioniranje *Hook*-a pod nazivom „useState“. *Hook* se poziva unutar funkcionalne komponente kako bi mu se dodalo neko lokalno stanje. React će zadržati to stanje između prikazivanja komponenti. *Hook* „useState“ vraća trenutnu vrijednost stanja i funkciju koja omogućuje ažuriranje tog stanja, [21].

4.1.2 Glavne komponente React Native-a

React Native ima velik broj glavnih komponenti koje se koriste za sve, od kontrole formi do pokazatelja aktivnosti. Glavne komponente prikazane su u tablici 4.

Tablica 4. Glavne komponente React Native-a, [16]

React Native UI komponenta	Android	Web	Opis
`<View>`	`<ViewGroup>`	`<div>`	Kontejner koji podržava izgled s flexboxom, stilom, upravljanjem dodirom i kontrolama pristupačnosti
`<Text>`	`<TextView>`	`<p>`	Prikazuje, stilove i tekst, pa čak i rukovanja dodirima događaja
`<Image>`	`<ImageView>`	`<img>`	Prikaz različitih tipova slika
`<ScrollView>`	`<ScrollView>`	`<div>`	Običan kontejner koji omogućuje scroll i može sadržavati više komponenti
`<TextInput>`	`<EditText>`	`<input type="text">`	Omogućuje korisniku upis teksta

4.2 Expo

Expo je skup alata i usluga izgrađenih oko React Native-a. Služi za olakšavanje razvoja, izgradnju i implementaciju iOS-a, Android OS-a i web aplikacija iz iste JavaScript baze podataka.

Aplikacije se grade uz upravljani tijek rada pomoću *expo-cli*-a, Expo klijenta na mobilnom uređaju i različitih usluga: push notifikacije, usluge izrade i OTA (eng. *over-the-air*). Expo olakšava izradu za programera, zbog čega se naziva upravljanim tijekom rada. Programeri koji koriste upravljani tijek rada ne koriste Android Studio, nego samo pišu JavaScript kôd i upravljaju konfiguracijom za stvari poput ikone aplikacije i splash screen-a putem *app.json* datoteke. Expo SDK otkriva sveobuhvatniji skup API-ja koji omogućuju pristup mogućnostima uređaja poput kamere, biometrijske provjere autentičnosti, datoteka, itd.

4.2.1 Razvoj u Expo-u

Expo-cli se koristi za posluživanje aplikacija kroz tunel uslugu, to znači da nije potrebno posjedovati uređaj spojen na računalo ili čak biti u istoj prostoriji ili državi. Zbog tog je moguće udaljeno uklanjanje pogrešaka JavaScripta i sve stvari koje se mogu raditi s React Native-om. Negativno je to što je korištenje tunela sporije od korištenja lokalnog servera.

4.2.2 Ograničenja

Expo uz sve mogućnosti sadrži i ograničenja koja su dostupna u tablici 5.

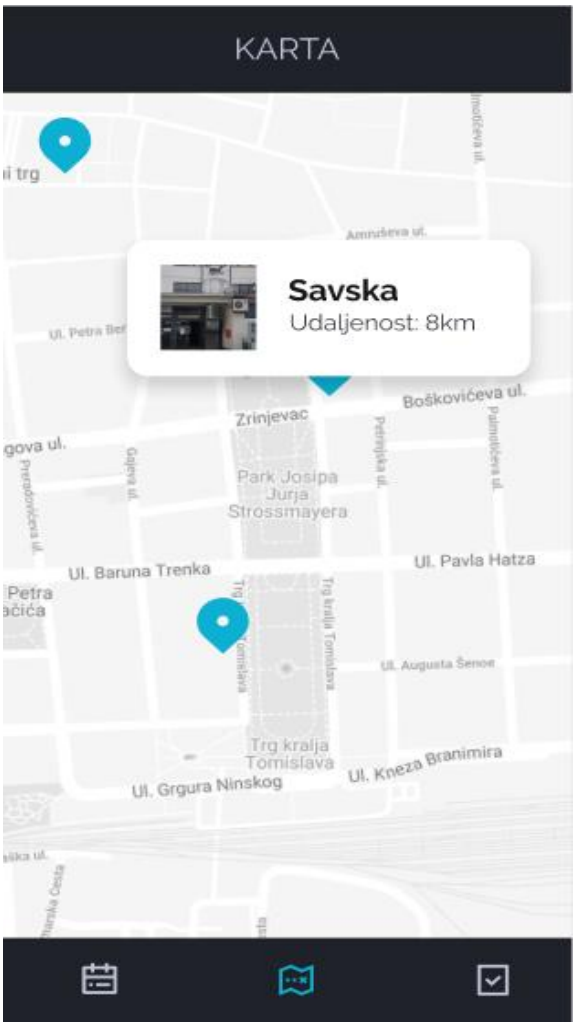
Tablica 5. Ograničenja Expo-a

Ograničenja Expo-a
Nedostupnost svih iOS i Android API-ja
SDK ne podržava sve vrste izvršenja pozadinskog koda
Izvorne biblioteke koje se koriste za *proprietary* usluge nisu uključene u SDK
Jedina podržana usluga push notifikacije je Expo usuga za notifikacije
Minimalne podržane verzije OS-a su Android 5+ i iOS 10+
Ažuriranja (JavaScript i sredstva) za ažuriranja i sastavljanja OTA su ograničena na veličinu datoteke

4.3 Izrada aplikacije StudentKit

StudentKit aplikacija sastoji se od tri dijela odnosno zaslona. Prvi odnosno početni zaslon koji je prikazan na početku četvrtog poglavlja odnosi se na kartu koja služi za prikaz i pronalazak najbližih menzi u gradu Zagrebu.

Glavna zamisao i funkcionalost ovog zaslona je da se pritiskom na marker prikazuju se informacije o određenoj menzi odnosno naziv, pripadajuća slika i udaljenost menze od trenutne lokacije korisnika što je vidljivo na slici 6. Dodirom na balon u kojem se nalaze sve informacije o menzi moguće je otvoriti aplikaciju „Google Maps“ koja će prikazati put i usmjeriti korisnika do željene menze.



Slika 6. Izgled balona nad markerom

Ovaj dio aplikacije sastoji se od dvije komponente. Prva komponenta pod nazivom “MapScreen” služi za dohvat lokacija menzi i prikaz istih na karti.

```
import React, { Component } from 'react';
import * as Permissions from 'expo-permissions';
import * as Location from 'expo-location';
import Map from '../components/Map';
import data from '../RestaurantData';

export default class MapScreen extends Component {
  state = {
    location: null,
    errorMessage: null,
    restaurants: []
  };

  getRestaurants = () => {
    let restaurants = data;
    this.setState({ restaurants });
  };

  getLocationAsync = async () => {
    let { status } = await Permissions.askAsync(Permissions.LOCATION);
    if (status !== 'granted') {
      this.setState({
        errorMessage: 'Permission to access location was denied'
      });
    }
    let location = await Location.getCurrentPositionAsync({});
    this.setState({ location });
    this.getRestaurants();
  };

  componentDidMount = () => {
    this.getLocationAsync();
};


  render() {
    const { location, restaurants } = this.state;
    return (
      <Map location={location} places={restaurants}
      />
    );
  }
}
```

Slika 4. Kod komponente “MapScreen”

Kao što je vidljivo na slici 4 “MapScreen” komponenta sadrži više funkcija, prva funkcija pod nazivom “getRestaurants” služi za dohvat podataka o svim menzama koje se nalaze unutar datoteke “RestaurantData” i sve dohvaćene podatke sprema u varijablu „restaurants“. Glavna funkcija komponente „getLocationAsync“ čija je zadaća dobivanje dopuštenja o korištenju lokacije uređaja od strane korisnika te dohvaćanje iste odnosno trenutne lokacije korisnika. U trenutku dohvata lokacije poziva se i prethodna funkcija “getLocationAsync” u trenutku pokretanja aplikacije.

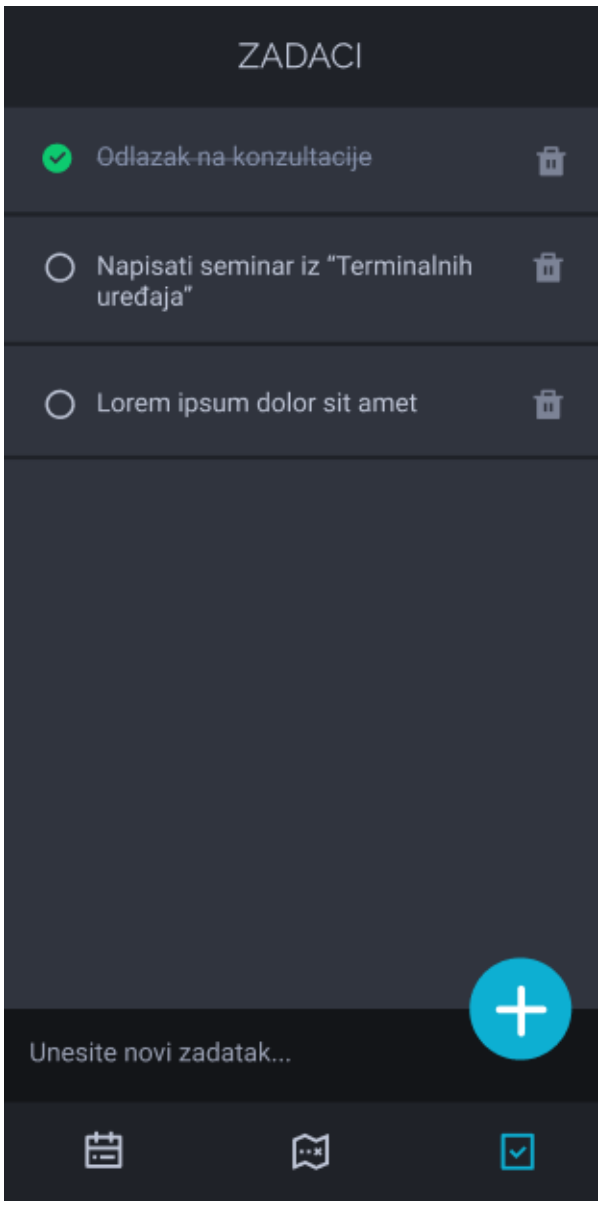
Nakon izvršenih funkcija komponente “MapScreen” ona vraća komponentu “Map” sa dobivenim vrijednostima funkcija koje će se koristiti unutar te komponente. Komponenta je učitana unutar “MapScreen” komponente i ona omogućava prikaz markera na karti koje se odnose na lokacije menzi i samog korisnika.

```
renderMarkers() {

    return this.props.places.map((place, i) => (
        <Marker
            key={i}
            title={place.name}
            coordinate={place.coords}
            onCalloutPress={() => this.handleGetDirections(i)}
        >
            <Callout tooltip>
                <View>
                    <View style={styles.bubble}>
                        <View style={styles.restaurantInfo}>
                            <Text style={styles.text}>
                                <Text>{place.name}</Text>
                                <Text>Udaljenost: {(this.getDistance(place))} km</Text>
                            </Text>
                            <MaterialCommunityIcons name="bullseye-arrow" size={40} color="dodgerblue" />
                        </View>
                        <View style={styles.calloutImg}>
                        <Text>
                            <Image
                                source={require("../assets/restoran-savska.png")}
                                resizeMode='cover'
                                style={styles.calloutImg}/>
                        </Text>
                        </View>

                    </View>
                </View>
            </Callout>
        </Marker>

    ));
}

render() {
    const { location } = this.props;
    const region = {
        latitude: get(location, 'coords.latitude', null),
        longitude: get(location, 'coords.longitude', null),
        ...deltas
    };
    if (!region.latitude || !region.longitude) {
        return (
            <View style={{ backgroundColor: '#191B1F', flex: 1, flexDirection: 'row', justifyContent: "center" }}>
                <ActivityIndicator size="large" color="#D2691E" />
            </View>
        );
    }
```

Slika 5. Glavni dio koda “Map” komponente

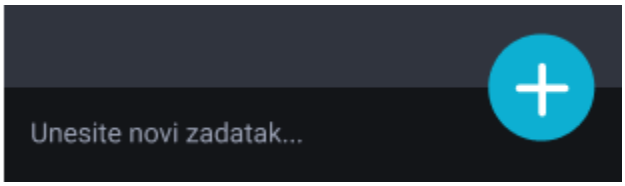
Slikom 5 prikazan je glavni dio koda komponente “Map” koja sadrži funkciju “renderMarkers” čija je glavna zadaća dohvat i prikaz markera za pojedinu lokaciju menzi. Funkcija za sve menze koje prima od komponente “MapScreen” dodaje marker.

Drugi dio aplikacije omogućuje unos i brisanje zadataka te označavanje istih kao obavljene. Funkcionalost koja je omogućena ovim dijelom služi studentima da unesu zadatak te da isti pritiskom na *button* dodaju na zaslon. Nakon što se uspješno doda zadatak, student ima mogućnost brisanja i označavanja istog kao obavljenog. Glavna svrha ovog dijela aplikacije je olakšavanje praćenja i ispunjavanja obaveza na fakultetu. Izgled zaslona sa dodanim zadacima kao primjer prikazan je slikom 7.



Slika 7. Zaslon sa dodanim zadacima

Komponente koje sadrži ovaj dio aplikacije nazivaju se “TodoScreen”, “Todo” i “AddTodo”. “TodoScreen” se koristi samo za prikaz komponente “Todo” putem navigacije koja se nalazi na dnu aplikacije. “Todo” sadrži komponentu “AddTodo” koja je glavni dio za izvršavanje svih mogućnosti ovog dijela aplikacije.



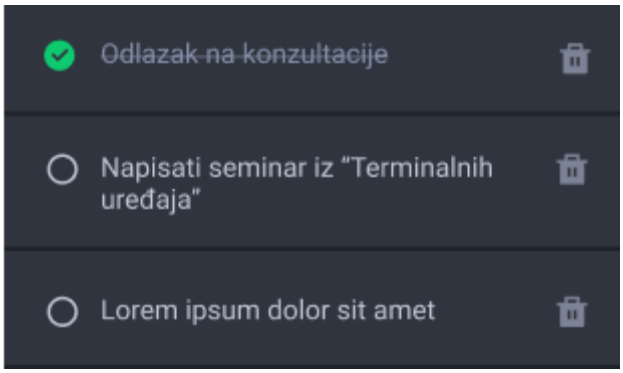
Slika 8. Dodavanje novog zadatka

Kao što je vidljivo na slici 8 na dnu zaslona nalazi se *TextInput* odnosno React Native API koji se koristi za unos željenog teksta, u ovom slučaju to je tekst sa novim zadatakom. Nakon unosa teksta pritiskom na *button* izvršava se funkcija “addTodo” čiji kod je vidljiv na slici 9 te se upisani zadatak prikazuje i sprema u bazu podataka koja će biti opisana kasnije u radu.

```
const addTodo = async () => {
    if (value.length <= 0)
        return;
    try {

        Keyboard.dismiss();
        todos.push({ text: value, key: Math.random().toString(), checked: false });
        loadTodo();
        await updateAsyncStorage(todos);

    } catch (err) {

        alert(err);

    }
}
```

Slika 9. Funkcija „addTodo“

“addTodo” funkcija nakon pritiska na *button* provjerava postoji li vrijednost teksta odnosno ima li duljinu veću od 0. Ukoliko je to ispunjeno vrijednost teksta dodaje se unutar polja *todos* te se nakon toga prikazuje zadatak i baza podataka ažurira se istim. Nakon što je zadatak prikazan na zaslonu uz zadatak dolaze dvije funkcije koje omogućavaju brisanje i označavanje zadatka obavljenim kao što je vidljivo na slici 10.



Slika 10. Označavanje i brisanje zadataka

S lijeve strane zadatka nalazi se *checkbox* te se pritiskom na isti prazan kružić ispunjava oznakom obavljenog zadataka kao što je vidljivo kod prvog zadatka na slici 10. Kako bi to bilo moguće izršava se funkcija "checkedTodo" čiji je kod prikazan na slici 11.

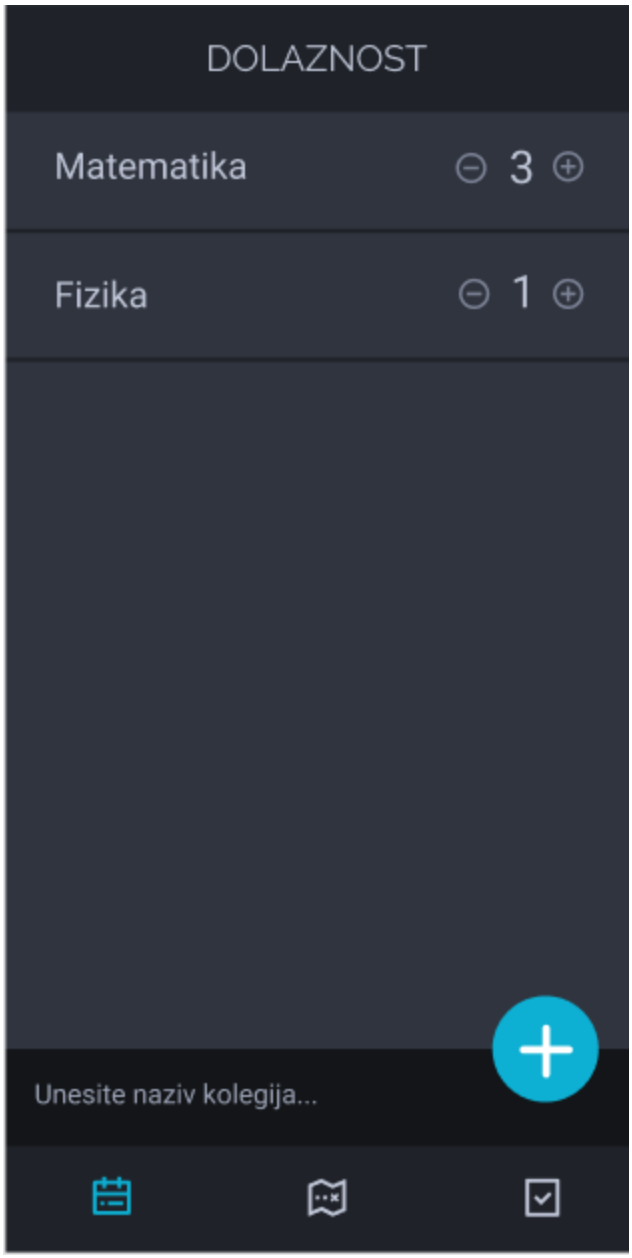
```
const checkedTodo = (id) => {

    setTimeout(() => {
        try {

            todos.map(async (todo, i) => {

                if (todo.key == id) {
                    todo.checked = !todo.checked;
                    await updateAsyncStorage(todos);
                    setTodos(todos);
                }

            });
        } catch (err) {
            alert(err);

        }
    }, 300)
}
```

Slika 11. Funkcija "checkedTodo"

Kao što je vidljivo funkcija prima "id" koji je jedinstvena oznaka svakog zadatka te na temelju njega pregledava polje "todos" i pronalazi određeni zadatak, mijenja mu stanje te ga označava izvršenim i isti je prekrižen ili obrnuto ako je već izvršen. Desno od zadataka nalazi se ikona na koju se pritiskom izvršava funkcija "removeTodo" te se na temelju "id"-a pronalazi zadatak i briše se iz baze podataka te nestaje sa zaslona. Funkcija koja se izvršava vidljiva je na slici 12.

Slika 12. Funkcija “removeTodo“

Treći dio aplikacije koji služi za dodavanje predavanja i brojanje dolazaka na iste vidljiv je na slici 13. Sastoji se od komponenti koje se nazivaju “CounterScreen”, “Counter” i “AddCounter”. Funkcionalost ovog dijela aplikacije slična je prijašnje opisanom dijelu koji se sastoji od zadataka. Glavna razlika je to što umjesto zadatka korisnik dodaje predavanje i za isto može dodavati ili oduzimati broj dolazak. Uz to kod ovog zaslona brisanje dodanog predavanja izvršava se dugim pritiskom na dio zaslona na kojem se nalazi predavanje koje korisnik želi obrisati.



Slika 13. Zaslon za kolegije i dolaznost

Na slici 13 vidljivi su primjeri dodanih predavanja, a desno od predavanja nalazi se broj dolazaka na to predavanje. Pored broja moguće je pritisnuti plus ili minus ikonu koje aktiviraju funkcije s kodom prikazanim slikom 14 koja dodaje u slučaju plusa odnosno oduzima broj dolazaka u slučaju minusa.

```
const incrementCount = (id) => {
    setTimeout(() => {
        try {
            counters.map(async (counter, i) => {
                if (counter.key == id) {
                    counter.count = counter.count + 1;
                    await updateAsyncStorage(counters);
                    setCounters(counters);
                }
            });
        } catch (err) {
            alert(err);

        }
    }, 300)
}
const decrementCount = (id) => {
    setTimeout(() => {
        try {
            counters.map(async (counter, i) => {

                if (counter.key == id) {
                    counter.count = counter.count - 1;
                    await updateAsyncStorage(counters);
                    setCounters(counters);
                }
            });
        } catch (err) {
            alert(err);
        }
    }, 300)
}
```

Slika 14. Funkcije „incrementCount“ i „decrementCount“

4.4 AsyncStorage

Baza podataka koja se koristila kroz aplikaciju za spremanje predavanja i zadataka naziva se „AsyncStorage“. „AsyncStorage“ je nešifrirani, asinkroni sustav za pohranu ključeva i vrijednosti koji je globalan za aplikaciju. Na iOS-u, „AsyncStorage“ podržan je izvornim kodom koji pohranjuje male vrijednosti u serializirani rječnik, a veće vrijednosti u zasebne datoteke. Na Androidu OS-u, „AsyncStorage“ će koristiti „RocksDB“ ili „SQLite“, [22].

„AsyncStorage“ koristi razne metode za dohvat, brisanje, dodavanje itd. podataka koje se nalaze u tablici 6.

Tablica 6. Metode „AsyncStorage“-a, [22]

Metode	
getItem	Dohvat određenog podatka
setItem	Postavljanje vrijednosti određenog podatka
removeItem	Brisanje određenog podatka
mergeItem	Dodaje novu vrijednost postojećoj Vrijednosti
Clear	Brisanje cijele baze podataka
getAllKeys	Dohvat svih jedinstvenih oznaka
flushGetRequests	Ispušta sve zahtjeve na čekanju pomoću jednog „multiGeta“
multiGet	„multiGet“ poziva skup parova jedinstvenih oznaka i vrijednosti koji odgovara *input*-u formata „multiSet“
multiSet	„multiSet“ i „multiMerge“ uzimaju niz skupa parova jedinstvenih oznaka i vrijednosti koji odgovaraju ishodu „multiGet“-a
multiRemove	Brisanje svih jedinstvenih oznaka
multiMerge	Dodaje novu vrijednost postojećim vrijednostima

5. ZAKLJUČAK

Aplikacije za mobilne uređaje su sadašnjost i budućnost ljudskih života te iz dana u dan postaju neizostavni dio svakodnevnice. Mogućnosti takvih aplikacija su bezbrojne i pitanje je vremena kada će pametni telefoni u potpunosti zamjeniti ostale uređaje kao što su osobno računalo, kamera, GPS navigacija.

Android OS je trenutno vodeći na tržištu operacijskih sustava, a sa znanjem da kao takav još uvijek ima prostora za napredak izgledno je održavanje tog statusa i u budućnosti. Razvoj OS-a na Android i iOS-a na Apple uređajima svakim se danom sve brže i sve više razvija. Zahvaljujući međusobnoj kompeticiji između Apple-a i Android-a i jedni i drugi prisiljeni su iz dana u dan razvijati nove i bolje tehnologije te ih prezentirati korisnicima.

React Native radni okvir unutar tekstualnog uređivača “Visual Studio Code” i alat Expo korišteni su zbog dostupnosti dokumentacije koja pomaže u korištenju te zbog činjenice da je sve besplatno. Uz pravilno korištenje i odgovarajuće alate moguće je razviti korisnu i kvalitetnu aplikaciju. U bližoj budućnosti razvoj mobilnih aplikacija će biti pristupačniji, lakši i brži zbog ekstremno brzog razvoja i napretka tehnologija za tu svrhu. Zbog pritiska konkurencije React Native će iz godine u godinu biti brži i efikasniji.

StudentKit je jednostavna aplikacija koja služi kako bi olakšala studentski život studentima grada Zagreba. Uz par klikova korisnik bilježi svoje zadatake koje mora obaviti na fakultetu te prati svoju prisutnost na kolegijima, a za vrijeme odmora ili slobodnog vremena korisnik može pronaći željenu menzu.

Aplikacija sadrži osnovne funkcije koje su potrebne da ispuni svoju svrhu, ali mogućnosti za povećanje broja i poboljšanje postojećih funkcija su velike. Npr., može se omogućiti dodavanje lista i grupiranje kolegija i zadataka prema pojedinom kolegiju, povezivanje studenata u svrhu razmjenjivanja poruka putem aplikacije te proširenje aplikacije na teritorij Republike Hrvatske ili na cijeli svijet.

LITERATURA

[1] PCWorld, Preuzeto sa: https://www.pcworld.com/article/2034723/android-founder we-aimed-to-make-a-camera-os.html[Pristupljeno: srpanj 2020.]

[2] Android službena dokumentacija, Preuzeto sa: https://source.android.com/

[Pristupljeno: srpanj 2020.]

[3] Android službena dokumentacija, Preuzeto sa: https://source.android.com/

devices/architecture [Pristupljeno: srpanj 2020.]

[4] Google Operating System blog, Preuzeto sa: https://googlesystem.blogspot.com /2007/11/google-launches-android-open-mobile.html [Pristupljeno: srpanj 2020.]

[5] https://developer.android.com/guide/topics/manifest/uses-sdk-element#ApiLevels

[Pristupljeno: srpanj 2020.]

[6] Statcounter portal, Preuzeto sa: https://gs.statcounter.com/os-market-share#yearly-2009-2020 [Pristupljeno: srpanj 2020.]

[7] Diffen portal, Preuzeto sa: https://www.diffen.com/difference/Android_vs_iOS

[Pristupljeno: srpanj 2020.]

[8] Raluca Budiu, Mobile: Native Apps, Web Apps, and Hybrid Apps, 2013.Preuzeto sa:

https://www.nngroup.com/articles/mobile-native-apps/ [Pristupljeno: srpanj 2020.]

[9] Službena Apple dokumentacija, preuzeto sa: https://developer.apple.com/library/

documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introductio n.html [Pristupljeno: srpanj 2020.]

[10] Swift dokumentacija, preuzeto sa: https://swift.org/about/ [Pristupljeno: srpanj

2020.]

[11] Go Java portal, preuzeto sa: https://go.java/developer-opportunities/ [Pristupljeno: srpanj 2020.]

[12] Kotlin službena dokumentacija, preuzeto sa: https://kotlinlang.org/docs/reference/

faq.html [Pristupljeno: srpanj 2020.]

[13] Web arhiva, preuzeto sa: https://web.archive.org/web/20070916144913/

http:/wp.netscape.com/newsref/pr/newsrelease67.html [Pristupljeno: srpanj 2020.]

[14] MDN web dokumentacija, preuzeto sa: https://developer.mozilla.org/en-US/

docs/Web/CSS [Pristupljeno: srpanj 2020.]

[15] MDN web dokumentacija, preuzeto sa: https://developer.mozilla.org/en-US/

docs/Web/HTML [Pristupljeno: srpanj 2020.]

[16] React Native službena dokumentacija, preuzeto sa: https://reactnative.dev/docs/getting-started.html [Pristupljeno: srpanj 2020.]

[17] Microsoft službena dokumentacija, preuzeto sa: https://docs.microsoft.com/en-

us/xamarin/get-started/what-is-xamarin [Pristupljeno: srpanj 2020.]

[18] Ionic službena dokumentacija, preuzeto sa: https://ionicframework.com/

[Pristupljeno: srpanj 2020.]

[19] Medium portal, preuzeto sa: https://medium.com/all-technology-feeds/

crossplatform-vs-native-mobile-app-development-choosing-the-right-dev-tools-for-your-app-project-47d0abafee81 [Pristupljeno: srpanj 2020.]

[20] W3school portal, preuzeto sa: https://www.w3schools.com/react/react_state.asp

[Pristupljeno: srpanj 2020.]

[21] React službena dokumentacija, preuzeto sa: https://reactjs.org/docs/hooks-

overview.html [Pristupljeno: srpanj 2020.]

[22] React službena dokumentacija, preuzeto sa: [21] React službena dokumentacija, preuzeto sa: https://reactnative.dev/docs/asyncstorage.html [Pristupljeno: srpanj 2020.]

POPIS KRATICA

OS	(Operating system) operacijski sustav
API	(Application programming interface) aplikacijsko programsko sučelje
IPC	(Inter-process communication) međuprocesna komunikacija
HAL	(Hardware abstraction layer) sloj apstrakcije hardvera
IDE	(Integrated development enviroment) integrirano razvojno okruženje
GPS	(Global Positioning System) globalni položajni sustav
HTML	(Hypertext Markup Language) prezentacijski jezik za izradu web stranica
CSS	(Cascading Style Sheets)
URL	(Universal Resource Locator) usklađeni lokator sadržaja
SDK	(Software development kit) komplet za razvoj softvera
OTA	(Over-the-air)

POPIS SLIKA

POPIS TABLICA

POPIS GRAFOVA

Sveučilište u Zagrebu

Fakultet prometnih znanosti

Vukelićeva 4, 10000 Zagreb

IZJAVA O AKADEMSKOJ ČESTITOSTI I SUGLASNOSTI

Izjavljujem i svojim potpisom potvrđujem da je ZAVRŠNI RAD (*vrsta rada*) isključivo rezultat mojega vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu, a što pokazuju upotrijebljene bilješke i bibliografija. Izjavljujem da nijedan dio rada nije napisan na nedopušten način, odnosno da je prepisan iz necitiranog rada te da nijedan dio rada ne krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Svojim potpisom potvrđujem i dajem suglasnost za javnu objavu završnog rada pod naslovom STUDENTKIT – APLIKACIJA ZA STUDENTE GRADA ZAGREBA TEMELJENA NA ANDROID OPERACIJSKOM SUSTAVU, u Nacionalni repozitorij završnih i diplomskih radova ZIR.

Student:

Gerić

(*potpis*)

U Zagrebu 9.9.2020.