RAZVOJ MOBILNE APLIKACIJE ZA GLAZBENI FESTIVAL

Jugović, Vilko

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

RAZVOJ MOBILNE APLIKACIJE ZA GLAZBENI FESTIVAL

Vilko Jugović

Zagreb, veljača 2023.

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada“.

U Zagrebu, veljača 2023.

Predgovor

Želim izraziti zahvalnost svim profesorima i asistentima koji su mi prenijeli znanje i iskustva tijekom mog preddiplomskog studija. Posebno bih želio zahvaliti svom mentoru, Aleksanderu Radovanu, koji je svojim entuzijazmom potaknuo moju strast za znanjem i programiranjem. Također, veliko hvala mojim roditeljima koji su mi omogućili studiranje na Visokom učilištu Algebra te time postavili temelje za moj daljnji razvoj. Na kraju, želim se zahvaliti svojoj djevojci Petri koja mi je pružala ogromnu podršku u najtežim trenucima i zaslužna je što se nalazim u ovom uspješnom životnom trenutku.

A Gradišćanska 24, HR-10000 Zagreb
T (01) 2222 182
E student@algebra.hr
www.algebra.hr

ALGEBRA

Temeljem članka 8. Pravilnika o završnom radu i završnom ispitu na preddiplomskom studiju Visokog učilišta Algebra sačinjena je ova

Potvrda o dodjeli završnog rada

kojom se potvrđuje da student Vilko Jugović, JMBAG 0023102003, OIB 80209225554 u šk. godini 2021./2022., studij: Primjenjeno računarstvo - Preddiplomski studij, smjer: Programsko inženjerstvo, od strane povjerenstva za provedbu završnog ispita, dana 25.02.2022. godine, ima odobrenu izradu završnog rada

s temom: **RAZVOJ MOBILNE APLIKACIJE ZA GLAZBENI FESTIVAL**

i sažetkom rada: Ovaj završni rad sadrži opis razvijenog aplikacijskog sustava za glazbeni festival s ciljem poboljšanja korisničkog iskustva posjetitelja tijekom festivala. Glavni cilj sustava je kroz tehnološko rješenje posjetiteljima glazbenih festivala omogućiti jednostavniji i ugodniji boravak. Aplikacija omogućuje korisnicima uvid u izvođače te njihov raspored nastupa, kreiranje vlastite liste omiljenih izvođača, jednostavnije snalaženje na festivalskom području, spremanje ulaznice na festival te dobivanje važnih informacija od organizatora.

Mentor je: Aleksander Radovan.

Odobrenjem završnog rada studentu je omogućen upis kolegija "Izrada završnog projekta/Praksa" te je sukladno članku 8. Pravilnika o završnom radu i završnom ispitu dužan najkasnije do početka nastave ljetnog semestra u sljedećoj školskoj godini, uspješno obraniti završni rad uspješnim polaganjem završnog ispita.

U protivnom student može zatražiti novog mentora/icu i temu te ponovo upisati kolegij "Izrada završnog projekta/Praksa" budući da rad koji nije predan i obranjen na završnom ispitu u roku određenom Pravilnikom završnom radu i završnom ispitu prestaje vrijediti. Izrada novog završnog rada se izvodi sukladno rokovima određenima za školsku godinu u kojoj je studentu određen novi mentor/ica i dodijeljen novi završni rad.

Potpis studenta:	Potpis mentora:	Potpis predsjednika povjerenstva:

Ova potvrda izdaje se u 4 (četiri) primjerka od kojih 3 (tri) idu kao prilog završnom radu.

Sažetak

Ovaj završni rad sadrži opis razvijenog aplikacijskog sustava za glazbeni festival s ciljem poboljšanja korisničkog iskustva posjetitelja tijekom festivala. Glavni cilj sustava je kroz tehnološko rješenje posjetiteljima glazbenih festivala omogućiti jednostavniji i ugodniji boravak. Aplikacija omogućuje korisnicima uvid u izvođače te njihov raspored nastupa, kreiranje vlastite liste omiljenih izvođača, jednostavnije snalaženje na festivalskom području, spremanje ulaznice na festival te dobivanje važnih informacija od organizatora.

Ključne riječi: aplikacijski sustav, glazbeni festival, posjetitelji.

Abstract

This final thesis contains a description of the developed application system for a music festival with the aim of improving the user experience of visitors during the festival. The main goal of the system is through a technological solution to provide music festival visitors a simpler and more pleasant stay. The application enables users to view the artists and their schedule of performances, create their own list of favorite artists, easier navigation around the festival area, save festival ticket and obtain important information from the organizers.

Keywords: application system, music festival, visitors, user experience.

Sadržaj

1. Uvod

MusicEvent je sustav namijenjen posjetiteljima glazbenih festivala. Sastoji se od tri komponente: Klijentskog i poslužiteljskog sučelja te baze podataka.

Mobilna aplikacija omogućava posjetiteljima kroz različitu interakciju jednostavniji i ugodniji boravak na festivalu.

Mrežni servis služi za komunikaciju i razmjenu podataka između mobilne aplikacije i baze podataka, te obavlja poslovnu logiku i procesiranje informacija prije prosljeđivanja drugim dijelovima sustava.

Rad sadrži četiri poglavlja. Nakon prvoga – uvodnog, drugo poglavlje sadrži obrazloženje i motivaciju odabira teme rada, kao i trenutne probleme i nedostatke u području koje se želi riješiti razvojem sustava.

Treće poglavlje obuhvaća opis dijelova aplikacije te svih komponenata unutar tih dijelova te opis korištenih tehnologija i arhitekture.

U četvrtome, ujedno i zadnjem poglavlju, izlaže se opis budućih funkcionalnosti koje bi unaprijedile korisničko iskustvo i korisnost aplikacije na osnovu odgovora koji su dali korisnici koji su imali priliku koristiti aplikaciju.

Motivacija za izradu MusicEvent aplikacije proizlazi iz uviđanja da mobilne aplikacije na glazbenim festivalima većinom nisu prilagođene posjetiteljima, nego se fokus stavlja na klijenta odnosno festival. Ovim rješenjem želi se kreirati rješenje koje će u potpunosti biti prilagođeno posjetiteljima i kroz tehnološko rješenje pokušati im učiniti boravak na festivalu što jednostavnijim i ugodnijim.

2. Problematika i prijedlog aplikativnog rješenja za glazbeni festival

Organizacija glazbenih festivala jedan je od najzahtjevnijih poslova u kulturnoj industriji. Odabir izvođača, logističke pripreme i komunikacija s posjetiteljima samo su neki od izazova s kojima se organizatori susreću. Međutim, jedna od najvećih poteškoća s kojom se organizatori suočavaju jest pružanje kvalitetnog iskustva posjetiteljima festivala.

Posjetitelji žele uživati u glazbi, ali i imati jasan pristup najnovijim informacijama, informacijama o rasporedu nastupa, mjestima na kojima se događaju koncerti, ali i praktičnim informacijama poput mjesta za prehranu te žele izbjeći osobnu frustraciju zbog stvaranja ogromnih gužvi na ulazima u festivalsko područje. Kako bi se olakšala interakcija s posjetiteljima, organizatori sve češće koriste mobilne aplikacije koje pružaju informacije o festivalu na jednom mjestu.

2.1. Provođenje i rezultati ankete o potrebama posjetitelja glazbenih festivala

Istraživanje o potrebama posjetitelja glazbenih festivala vršila se generativnim istraživanjem (engl. *generative research*). Generativno istraživanje je metoda istraživanja koja se fokusira na stvaranje novih ideja, koncepta ili rješenja putem otvorenog dijaloga i kreativne suradnje između ljudi koji su uključeni u proces istraživanja([7]). Ova metoda obično se koristi u područjima poput dizajna, marketinga, inovacija i sličnih područja gdje je potrebno stvoriti nešto novo i inovativno. Cilj ovakvog istraživanja je dobivanje inovativnih ideja na temelju motivacija i frustracija korisnika i da se realiziraju one ideje koje donose veliku vrijednost korisniku, a malo ulaganje truda kreatora.

Prvi korak istraživanja bio je provođenje filter ankete nad zaposlenicima mStart plus d.o.o. tvrtke. Filter anketa je vrsta ankete koja se koristi kako bi se ograničila populacija koju se ispituje, odnosno odabrali samo određeni ispitanici koji ispunjavaju određene kriterije. Cilj ankete je bio odabrati ciljanu skupinu korisnika koja ima 18-36 godina te posjećuje glazbene festivale bar jedan put godišnje. U anketi je sudjelovalo 80 osoba. Pitanja koja su postavljana ispitanicima bila su sljedeća:

1. Kojoj dobnoj skupini pripadate?
2. Koliko često idete na glazbene festivale?
3. Jeste li u prošlosti koristili mobilnu aplikaciju na glazbenom festivalu kao dio festivalskog iskustva?

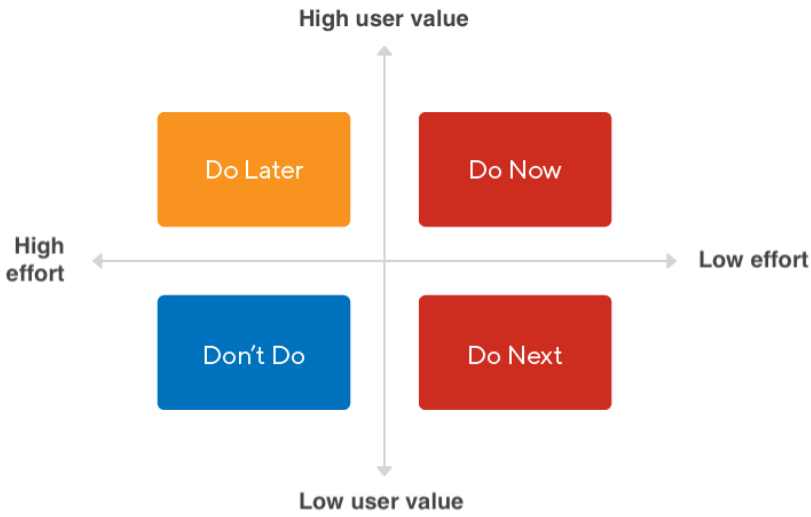
Rezultat ankete dao je 30 osoba koje odgovaraju ciljanoj skupini.

U drugom koraku se vršila kvalitativna metoda istraživanja, jedan na jedan sa svakom od osobom od 30 minuta putem Microsoft Teams programa. Istraživanje se odvijalo u razdoblju od 10. studenog do 10. prosinca 2022. godine. Cilj kvalitativne metode istraživanja je bio razumjeti složenost problema te prikupiti dubinske informacije od osoba. Osobu se tijekom razgovora kroz pitanja navodilo na iskazivanje problema na glazbenim festivalima te se pokušavao razumjeti kontekst u kojem su se problemi događali.

Nakon ispitivanja, odgovori svih ispitanika su se sumirali te se dobio konačni postotak važnosti određenih odgovora:

- 90% ispitanika postane frustrirano zbog gužve na ulazima na festivale
- 83.3% ispitanika žele imati informacije o gužvama na šankovima i toaletima
- 83.3% ispitanika žele imati odmah saznanje o važnim novostima
- 83.3% ispitanika želi imati vizualni raspored nastupa unutar aplikacije
- 56.6% ispitanika želi imati popis nastupa omiljenih izvođača
- 56.6% ispitanika želi imati iscrtanu mapu festivala unutar karte
- 26.6% ispitanika ne mogu do bar-kôda ulaznice zbog lošeg signala

Sljedeći korak je bio uzeti sve rezultate istraživanja i postaviti u matricu prioriteta(Slika 2.1). Matrica prioriteta (engl. *Prioritization matrix*) je alat koji se koristi za rangiranje ideja, projekata, zadataka ili bilo kojeg drugog skupa stavki kojima treba odrediti prioritete([10]).



Slika 2.1 Matrica prioriteta

Na osnovu svih rezultata, pomoću matrice prioriteta definirani su prioritetni zadatci koji će se implementirati u izradi mobilne aplikacije za glazbene festivale.

2.2. Usporedba s postojećim rješenjima

Mobilne aplikacije za glazbene festivale postaju sve popularnije jer nude posjetiteljima festivala mnoge prednosti u usporedbi s postojećim rješenjima.

Ako se pogleda hrvatsko tržište , postoji dosta prostora za razvoj digitalnih rješenja jer se muzički festivali još uvijek oslanjaju na neke tradicionalne pristupe organizaciji poput tiskanja papirnatih ulaznica te izrade rasporeda događaja u tiskanom obliku. Takav način organizacije u današnje vrijeme dovodi do sve veće neučinkovitosti te postaje frustrirajuća za posjetitelje.

S druge strane, festivali koji uvode tehnološke novine postaju popularniji i privlačniji za posjetitelje. Problematika kod postojećih digitalni rješenja je ta da se više fokusiraju na organizatore festivala, pokušavajući ispuniti njihove zahtjeve kroz promociju festivala i povećanje prodaje, nego na posjetitelje festivala. MusicEvent aplikacija ima za cilj usmjeriti se na razvoj digitalnog rješenja u obliku mobilne aplikacije koje će biti u potpunosti prilagođena posjetiteljima te na taj način zadovoljiti potrebe korisnika, a istovremeno ostvariti ciljeve organizatora.

2.3. Prijedlog rješenja

S obzirom na rezultat istraživanja predlaže se implementacija mobilne aplikacije s funkcionalnostima koje su prioritetno najvažnije posjetiteljima festivala. Ova mobilna aplikacija bi omogućila posjetiteljima da prilikom ulaska u aplikaciju imaju uvid u sve izvođače te da mogu vidjeti raspored nastupanja svih izvođača u određenom danu kroz vizualno kreirani prikaz. Također bi se unutar aplikacije mogao spremati bar kôd ulaznice kako bi se riješio problem dohvaćanja ulaznica s različitih web stranica zbog lošeg signala. Problem oko primanja obavijesti tijekom festivala bi se riješio kroz push obavijesti (engl. *push notifications*), a omiljeni izvođači bi se spremali u vlastite liste kako bi posjetitelji imali lakši uvid u iste, te bi se unutar rasporeda nastupanja omiljeni izvođači prikazivali drukčijom bojom radi jednostavnijeg raspoznavanja istih. Iscrtana festivalska mapa s ucrtanim točkama bitnih festivalskih područja bi se prikazivala unutar karte te samim time bi olakšala snalaženje na festivalskom području.

S ovom aplikacijom, organizatori bi mogli učinkovito komunicirati s posjetiteljima, slati im obavijesti i obavještenja, ali i dobivati povratne informacije. Korištenje ovog aplikativnog rješenja olakšalo bi proces organiziranja festivala, ali i stvorilo pozitivno iskustvo za posjetitelje. Uz to, organizatori bi imali bolji uvid u posjećenost festivala, što bi im omogućilo da poboljšaju svoje marketinške strategije i ponude bolje iskustvo za posjetitelje u budućnosti.

3. Implementacija aplikativnog rješenja

U ovom poglavlju bit će opisan sustav i njegove komponente, tehnologije korištene u razvoju aplikacije i načini na koje su izvedeni pojedini dijelovi aplikacije. Poseban naglasak bit će stavljen na mobilnu aplikaciju i njezino korisničko sučelje, budući da je to razina na kojoj korisnici izravno komuniciraju s aplikacijom te značajno utječe na njezin uspjeh i upotrebljivost. Grafičko sučelje mobilne aplikacije predstavlja konkretni segment ovog rada.

U prvom dijelu poglavlja bit će opisana tehnologija koja je korištena za izradu korisničkog sučelja - React Native. Radi se o brzorastućem razvojnom okviru za izradu višestrukih aplikacija.

Nakon toga, svaka pojedina komponenta sustava dobit će vlastito potpoglavlje u kojem će se detaljnije opisati način na koji se sustav razvio, slojevitost unutar svakog dijela aplikacijskog sustava te će se predstaviti primjeri izvedbe kôda.

3.1. Glavne značajke React Native programskog okvira

React Native (skraćeno RN) je popularan razvojni okvir otvorenog kôda (engl. *Open-source*) baziran na ReactJS-u, okviru koji se koristi za izradu sučelja web-stranica([1]). Primarno je namijenjen za razvoj mobilnih aplikacija, ali kasnije se njegova upotreba proširila i na druge popularne platforme. Omogućuje programerima da koriste jedan kôd za izgradnju aplikacija na popularnim mobilnim platformama, što omogućava uštedu vremena razvoja i troškova. RN koristi JavaScript programski jezik (skraćeno JS) za izradu korisničkog sučelja, ali koristeći nativni[1] prikaz operativnog sustava (skraćeno OS).

Ideja o RN je započela još 2012. godine kad je tim Facebooka razmišljao o novom pristupu *mobile-first* razvoja i dizajna, što je značilo usmjeravanje resursa na mobilne uređaje koji su postajali standard na svjetskom tržištu. Facebook je htio pristupiti drukčije od ostalih tvrtki koje su se usmjerile na razvoj nativnih aplikacija za mobilne uređaje te je planirao razvijati

[1] Nativne aplikacije su softverska rješenja razvijena za određenu platformu ili operacijski sustav.

mobilnu web-stranicu koristeći HTML5 jezik. Ta ideja nije dugo živjela jer su uvidjeli kako neće doseći performanse koje su od sustava očekivali.

Već iduće godine razvojni programer (engl. *developer*) Jordan Walke osmislio je način kojim bi se JavaScript kôdom generirali elementi za korisničko sučelje (engl. *User interface*, skraćeno UI) za iOS uređaje. Potaknuti tom idejom Facebook organizira *Hackathon*[2] kako bi dobili bolji uvid u mogućnosti budućeg razvoja mobilnih aplikacija kroz globalno poznat i korišten programski jezik.

Tako je React Native prvo zaživio kao razvojni alat za iOS uređaje, no ubrzo se pojavljuje i podrška za Android uređaje. Prva inačica React Native razvojnog okvira javno je objavljena 2015. godine. Od tada React Native raste velikom brzinom te postaje zajednica s više stotina tisuća razvojnih programera i preuzimanja više od 1 milijun puta svakoga tjedna([12]).

Višeplatformski (engl. *Cross-platform*) razvoj aplikacijskih rješenja postao je popularan trend u svijetu razvoja mobilnih aplikacija. S porastom raznih platformi i uređaja, postalo je ključno za poslovne subjekte razvijati aplikacije koje se mogu izvoditi na više platformi. React Native je jedan od najpopularnijih okvira za razvoj višeplatformskih rješenja. Inovacija koju je React Native uveo je pozadinski pristup. Do tada su ostali konkurentski okviri koristili `WebView` komponentu koja je služila za prikaz web stranice unutar mobilne aplikacije. RN, za razliku od konkurentskih okvira , koristi standardne jezične prevoditelje (engl. *compiler*) za svaku platformu kako bi napisani JS kôd i korištene komponente preveo u nativne komponente svake platforme ili operacijskog sustava (engl. *Operating System,* skraćeno OS) čime je dodao novi inovativni mehanizam razvoja mobilnih aplikacija.

Arhitektura RN sastoji se od tri ključna dijela: JavaScript kôd, JavaScript Bridge, platformski kôd([8]):

1. JavaScript kôd se koristi za opisivanje izgleda i ponašanja aplikacije. Programeri koriste React Native API-e kako bi definirali izgled aplikacije i kako bi definirali funkcionalnost. Ova funkcionalnost se zatim prevodi u niz naredbi koje se izvršavaju na uređaju.
2. JavaScript Bridge je veza između JavaScript koda i platformskog kôda. On omogućuje JavaScript kodu da poziva platformski kôd, a platformskom kodu da poziva JavaScript

[2] Natjecanje u kojemu programeri rješavaju zadane probleme i zadatke smišljajući vlastita softverska rješenja.

kôd. Ovo je bitno zbog toga što platformski kôd pruža funkcionalnost koja nije dostupna u JavaScript kodu, kao što su pristup uređaju, senzorima, kameri i drugim uređajima.

3. Platformski kôd je kôd napisan u programskom jeziku specifičnom za određenu platformu (Objective-C, Swift, Java, Kotlin, itd.) koji se izvršava na uređaju. On pruža funkcionalnost koja nije dostupna u JavaScript kôdu, kao što su upravljanje kamerom, uređajem i drugim hardverskim funkcijama. Platformski kôd se prevodi u izvršni kôd koji se izvršava na uređaju.

Glavna značajka ovakve arhitekture je višedretvenost[3] (engl. *multithreading*), koju React Native koristi kako bi postigao bolju izvedbu i responzivnost aplikacije. Jedna od glavnih prednosti višedretvenosti je omogućavanje aplikaciji da izvršava više zadataka istovremeno, što je ključno za mobilne aplikacije koje moraju reagirati brzo.

Hot Reloading je još jedna od bitnih značajki za programere, koja React Native čine veoma popularnim. Ova značajka omogućava programerima da vide promjene u stvarnome vremenu prilikom razvoja aplikacije što značajno ubrzava razvojni proces.

React Native također ima sve širu upotrebu radi toga što omogućava brz razvoj aplikacija jer podržava upotrebu modula i biblioteka koje mogu biti napisane u bilo kojem programskom jeziku.

3.2. Klijentski dio aplikacije

MusicEvent sustav posjeduje grafičko sučelje – mobilnu aplikaciju u React Native razvojnom okviru koristeći Expo. Expo je razvojni okvir za RN koji dolazi s nizom alata za pojednostavljenje procesa razvoja aplikacija.

Korisnici MusicEvent mobilne aplikacije imaju mogućnost pregleda izvođača te dodavanja izvođača u favorite, raspored njihovih nastupa, prikaz ucrtane mape festivala, primanje obavijesti od organizatora te spremanje festivalske ulaznice u aplikaciju.

[3] U računalnoj arhitekturi, višedretvenost je sposobnost središnje procesorske jedinice da pruži više niti istovremenog izvršavanja ([11])

3.2.1. Kreiranje i pokretanje aplikacije

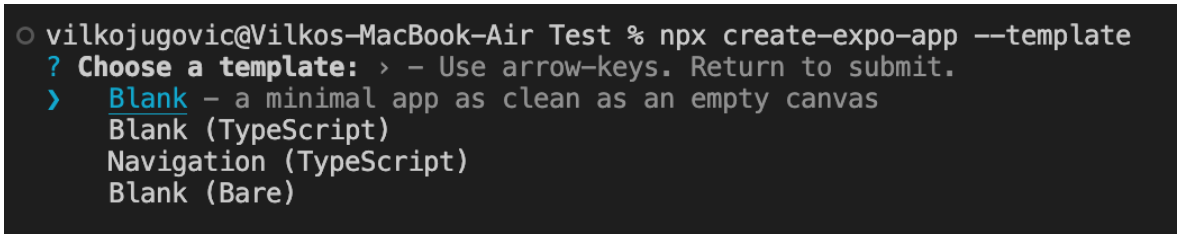
Za razvoj aplikacije pomoću Expo-a, potrebna su dva alata. Alata naredbenog retka (engl. *Command-line*) Expo CLI (engl. Expo *command-line interface*) i mobilna klijentska aplikacija Expo Go za otvaranje projekta na Android i iOS platformi. Osim toga može se koristiti bilo koji web preglednik za pokretanje projekta.

Nakon instalacije alata, novi React Native projekt stvaramo kroz Expo putem komandne linije (engl. *Command Prompt*, skraćeno cmd) naredbom:

```
npx create-expo-app naziv_projekta
```

gdje se umjesto `naziv_projekta` postavlja naziv aplikacije. Također se može koristiti `–template` opcija unutar komandne linije kako bi se prilikom pokretanja mogla vidjeti lista dostupnih šablona (engl. *Template*)(Slika 3.1).

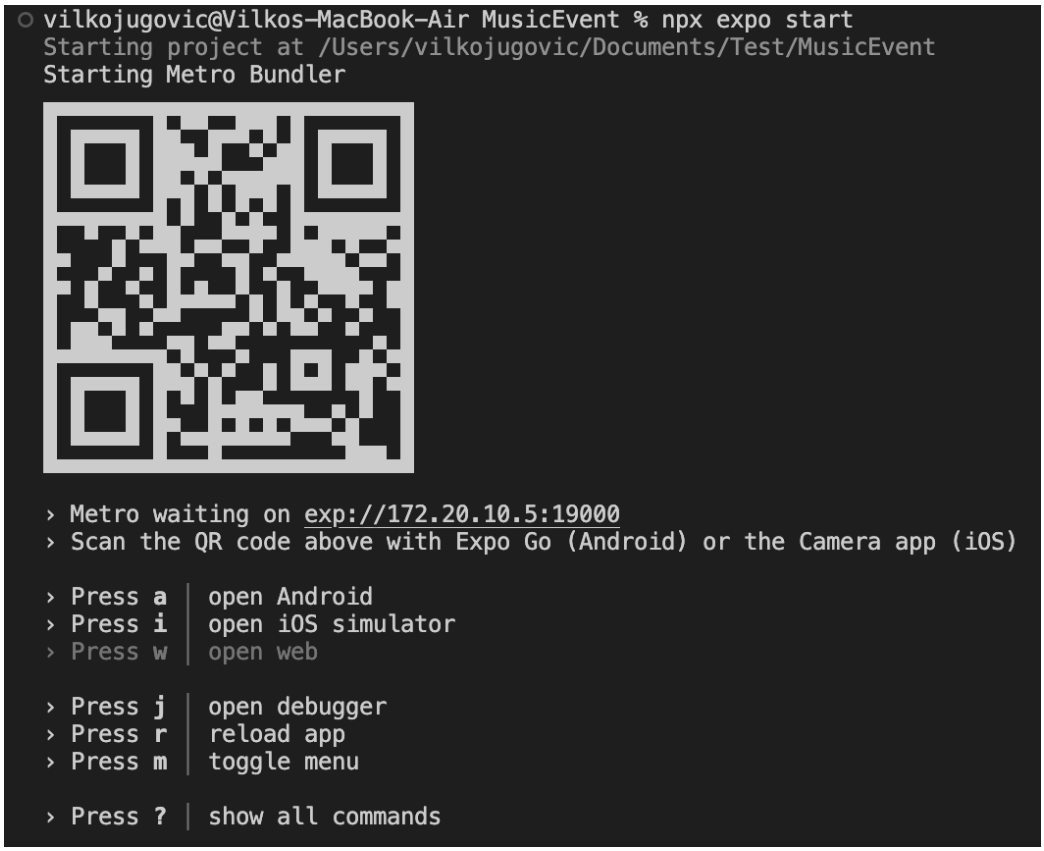
```
npx create-expo-app --template
```



Slika 3.1 Odabir vrste expo projekta prilikom stvaranja

Za mobilnu aplikaciju MusicEvent odabrana je opcija - *blank*. Nakon odabira, skripta se izvršava kako bi se stvorio početni temelj aplikacije. Na tom temelju će se graditi sve potrebne komponente, ekrani i funkcionalnosti.

Za pokretanje aplikacije u razvojnom okruženju, potrebno je izvršiti naredbu `npx expo start` u cmd kako bi se pokrenuo Expo CLI. Nakon toga, aplikacija se može otvoriti na AVD-u. Na iPhone virtualnom stroju, aplikacija se može otvoriti putem web preglednika s "i", a na web-pregledniku s "w". Alternativno, kôd se može skenirati putem QR čitača koji se generirao u CLI-ju(Slika 3.2). Jedini uvjet za uspostavljanje veze i otvaranje aplikacije je da su oba uređaja spojena na istu mrežu ako se razvojno okruženje otvara na lokalnom stroju.



Slika 3.2 Pokretanje Expo CLI

Za stvaranje i distribuciju MusicEvent aplikacije koristio se EAS. EAS (engl. *Expo Application Services*) su duboko integrirane usluge u oblaku (engl. *Cloud services*) za Expo i React Native aplikacije, kreirane od Expo tima. To je set alata za automatiziranje postupka stvaranja i distribucije aplikacija. EAS kombinira najbolje iz Expo-ove postojeće platforme i dodaje dodatne značajke koje pomažu programerima da ubrzaju proces stvaranja i distribucije aplikacija.

Za povezivanje aplikacije s EAS-om potrebno je prvo instalirati EAS CLI putem komandne linije naredbom:

```
npm install -g eas-cli
```

Zatim kroz cmd naredbom:

```
Eas build:configure
```

podešavamo iOS i Android projekt za EAS. Potrebno je dodati nekoliko stvari u konfiguraciju kako bi projekt bio spreman za korištenje EAS-a. To uključuje dodavanje Apple Developer računa, certifikata i profila, kao i povezivanje s distribucijskim kanalom.

Sada kada je projekt konfiguriran za EAS, sljedeći korak je stvaranje artefakta. Artefakt je *prebuilt*[4] verzija aplikacije koja je spremna za distribuciju. Stvara se pomoću komandne naredbe `eas build`, gdje unutar komandne linije imam opciju definiranja platforme i profila.

Nakon sto smo kreirali aplikaciju, moguće ju je objaviti u trgovinu aplikacija ili instalirati direktno na uređaj.

3.2.2. Navigacija među komponentama i lokalizacija

Kako bi se olakšalo korištenje mobilnih aplikacija i stvorilo intuitivno korisničko sučelje, kompleksnije aplikacije se često dijele na cjeline ili module. Bez modularizacije, sve bi se funkcionalnosti okupile na jednom mjestu što bi stvorilo zabunu i otežalo povezivanje između komponenti, formi i elemenata koje aplikacija posjeduje.

Svaki modul aplikacije bi trebao imati jedan ili više ekrana kako bi se omogućilo korištenje specifičnih funkcionalnosti ugrađenih u aplikaciju. U takvom slučaju, navigacija postaje ključni dio aplikacije koji omogućuje korisniku jednostavno kretanje između ekrana.

U projektu MusicEvent, navigacija se implementira korištenjem React Navigation paketa (verzija 5.x). Ovaj paket sadrži ključne usluge React Native okvira za stvaranje navigacijske strukture za kretanje kroz aplikaciju.

React navigacija je biblioteka za usmjeravanje i navigaciju u React Native aplikacijama([9]). Pruža jednostavan i učinkovit način rukovanja navigacijom i tokom korisnika u mobilnim aplikacijama. U ovom poglavlju ćemo opisati proces instalacije i opis funkcionalnosti korištenih u MusicEvent aplikaciji.

React Navigation se dodaje u projekt pokrećući komandnu liniju u korijenskom direktoriju (engl. *Root directory*) RN aplikacije i upisujući komandu:

```
npm install @react-navigation/native@^5.9
```

Da bi se održala potpuna funkcionalnost navigacije u React Navigation-u, potrebno je uključiti dodatne pakete u aplikaciju kako bi se izbjegla neuobičajena ponašanja i pogreške

[4] Unaprijed izgrađena

prilikom korištenja. Stoga, nakon izvršavanja prve naredbe, preporučuje se instaliranje tih paketa pokretanjem sljedeće komande:

```
npm install react-native-reanimated
react-native-gesture-handler
react-native-screens
react-native-safe-area-context
@react-native-community/masked-view
```

Kôd 3.1 Dodavanje paketa potrebnih za navigaciju

Nakon uspješne instalacije, React Navigation se može koristiti unutar projekta.

U ovom radu, kada se govori o navigaciji, komponente i instance koje su stvorene korištenjem React Navigation paketa za navigaciju u aplikaciji, bit će nazvane navigatori.

Unutar projekta MusicEvent svaki navigator je zaseban JavaScript datoteka koja se sastoji od tri bloka koda: `import`, `create` i `export` (Slika 3.3).

```
1  import React from "react";
2  import { createStackNavigator } from "@react-navigation/stack";
3  import UserTopTab from "./user-top-tab";
4
5  const Stack = createStackNavigator();
6
7  const TopTabStack = () => {
8    return (
9      <Stack.Navigator
10       headerMode="none">
11       <Stack.Screen
12         name="TopTab"
13         component={UserTopTab}
14       />
15     </Stack.Navigator>
16   );
17 };
18
19 export default TopTabStack;
20
```

Slika 3.3 Primjer navigacije u MusicEvent aplikaciji

U `import` dijelu uvoze se paketi, drugi navigatori te ekrani ili komponente potrebni za prelazak s jednog na drugi. `Create` dio je mjesto gdje se navigator inicijalizira te mu se dodaju ekrani ili komponente između kojih će se vršiti kretanje, a `export` dio dolazi na kraju i omogućava korištenje navigatora u nekoj drugoj JavaScript datoteci unutar projekta.

Kako bi se navigacija učinila dostupna kroz čitavu aplikaciju potrebno je definirati unutar početne JS datoteke *App.js* glavni navigator koji će se pokrenuti prilikom pokretanja aplikacije. Glavni navigator mora biti omotan u `NavigationContainer` komponentu koja upravlja našim navigacijskim stablom i sadrži stanje navigacije.

Nakon definiranja navigacija, prelazak s jednog ekrana na drugi vrši se unutar koda samih ekrana. Prelazak se događa pozivom funkcije unutar koje je definiran prelazak. Na primjeru sa slike(Slika 3.4) vidi se prelazak na ekran `Artist` te prosljeđivanje na sljedeći ekran atributa `artistId` kroz navigaciju.

```
press={() => navigation.navigate("Artist",
  {
    artistId: item.id
  }
)}
```

Slika 3.4 Funkcija prelaska s ekrana na ekran

U aplikaciji MusicEvent korištena su tri tipa navigatora: Stack, Drawer i Top Tab. Svaki od ovih navigatora ima jedinstvena svojstva, način stvaranja i funkcioniranja te se koristi u različite svrhe kako bi se osiguralo intuitivno korištenje aplikacije.

Drawer navigacija je jedan od načina organiziranja navigacije unutar React Native aplikacija. Ova biblioteka omogućuje korisnicima da se kreću kroz različite zaslone u aplikaciji, a svaki zaslon se otvara iz bočne ladice (engl. *drawer*). Kreira se pomoću funkcije `createDrawerNavigator` koja se uvozi iz paketa `@react-navigation/drawer`.

Implementacija Drawer navigatora u React Navigation-u uključuje stvaranje ladice i definiranje zaslona koji će se otvarati. Osim toga, korisnici mogu otvoriti ladicu pritiskom na određeni gumb ili potezom prsta po zaslonu, te se na taj način se kretati kroz zaslone.

U MusicEvent aplikaciji, najviše se koristi Stack navigator. Prema zadanim postavkama funkcionira na način da je prilagođen kretanjem među ekranima baš onoj platformi na kojoj se pokreće. Tako na iOS uređajima prelazak s jednog ekrana na drugi vrši se *slide in*[5] animacijom, dok kod Android uređaja to je *fade in*[6] animacija. Ovaj navigator funkcionira

[5] Standardni animacijski prelazak s jednog ekrana na drugi na iOS uređajima
[6] Standardi animacijski prelazak s jednog ekrana na drugi na Android uređajima

na način da kada se prelazi s jednog ekrana na drugi, stvara se stog (eng. *stack*) ekrana koji pamti put kojim se došlo do trenutno prikazanog ekrana. Takav pristup omogućuje povratak na prethodne ekrane i održava logički slijed navigacije. Za kreiranje Stack navigatora koristi se funkcija `createStackNavigator` koja se uvozi iz `@react-navigation/stack` paketa.

Navigator s gornjim oznakama (engl. *Top Tabs Navigator*) je navigacijski sustav koji omogućava prebacivanje između različitih zaslona korištenjem kartica na vrhu zaslona. Unutar aplikacije MusicEvent koristi se za prikaz novosti na različitim društvenim mrežama. Za kreiranje ovog navigatora koristi se funkcija `createMaterialTopTabNavigator` koja se uvozi iz paketa `@react-navigation/material-top-tabs`, a da bi ovaj navigator funkcionirao potrebno je dodati i paket `react-native-tab-view`. Primjer navigacije s gornjim oznakama (Slika 3.5) prikazuje tri taba Twitter, Facebook i Instagram.

Slika 3.5 Primjer navigacije s gornjim oznakama

Lokalizacija omogućava da aplikacija bude dostupna na više jezika, što olakšava proces prijevoda i dodavanja novih jezika, te širi ciljno tržište aplikacije. U slučaju MusicEvent aplikacije, koji je namijenjen i gostujućoj populaciji iz različitih zemalja, nužno je da

podržava različite jezike kako bi se zadovoljilo što veće govorno područje. Trenutna inačica aplikacije podržava engleski i hrvatski jezik. Sve funkcionalnosti lokalizacije nalaze se u mapi `localization` (Slika 3.6) unutar React Native projekta.

Slika 3.6 Lokalizacijska mapa

U MusicEvent aplikacije lokalizacija se rješava na način da aplikacija prepoznaje jezik postavljen na uređaju i prema tome vrši prijevod. Za takvu realizaciju lokalizacije potreban je `expo-localization` modul i `i18n-js` paket. Expo-localization modul pristupa nativnom dijelu mobilnog uređaja i kroz konstantu `Localization.locale` vraća jezik postavljen na uređaju te prilikom pokretanja aplikacije i18n.locale varijabli se dodjeljuje dohvaćena vrijednost. Zatim kada se funkcija `localization` pozove u kodu aplikacije, prijevodni ključ se šalje kao argument `i18n.t` metodi koja prema trenutno postavljenom jeziku dohvaća prijevodni tekst iz prijevodnog rječnika.

```
import * as Localization from "expo-localization";
import i18n from "i18n-js";
import en from "./languages/en";
import hr from "./languages/hr";

i18n.translations = {
  en: en,
  hr: hr
};

i18n.locale = Localization.locale;

export const localization = (translationItem) => {
  return i18n.t(translationItem);
```

```
};
```

Kôd 3.2 Inicijalizacija lokalizacije

3.2.3. Komunikacija s mrežnim servisom i spremanje podataka

Tri osnovne komponente MusicEvent projekta su: Mobilna aplikacija, mrežni servis i baza podataka. Da bi mobilna aplikacija mogla dohvatiti iz baze podatke ili spremiti na bazu, potrebno je poslati zahtjev na mrežni servis koji će u odgovoru vratiti tražene informacije.

Komunikacija između Mrežnog servisa te mobilne aplikacije odvija se putem JavaScript Fetch API sučelja namijenjenog za komunikaciju putem HTTP protokola (engl. *HyperText Transfer Protocol*) s već odrađenom logikom komunikacije([2]).

```
export const getArtists = async () => {
  const response = await fetch(artists);
  return await response.json();
};
```

Kôd 3.3 Asinkrona funkcija dohvaćanja podataka fetch metodom

U MusicEvent aplikaciji kôd za komunikaciju s mrežnim servisom nalazi se u mapi *endpoints.*

Korištenjem Fetch API-ja dostupna je javna funkcija `fetch` koja omogućuje slanje HTTP zahtjeva na određeni URL putem prvog obveznog parametra. Drugi parametar sadrži različite postavke za zahtjev, od kojih se najčešće koriste `method`, `body` i `headers`. `Method` parametrom definira se vrsta zahtjeva (GET ili POST), dok se `headers` parametrom kao objektom mogu postaviti sva zaglavlja HTTP protokola i pridijeliti željene vrijednosti. U slučaju korištenja POST metode, `body` parametar sadrži podatke u JSON formatu koje je potrebno poslati poslužitelju.

```
const response = await fetch(token, {
    method: "POST",
    headers: {
      "Content-Type": "application/json"
    },
    body: JSON.stringify(bodyData)
  });
```

Kôd 3.4 Primjer POST metode iz MusicEvent aplikacije

Unutra mobilne aplikacije MusicEvent mrežni servis se poziva asinkronim funkcijama koje se izvode u jednoj dretvi i ne blokiraju izvođenje drugog kôda. Nakon što mobilna aplikacija primi odgovor (engl. *response*), podatci u JSON formatu pretvaraju se u čitljive informacije za JavaScript kôd.

Mobilne aplikacije koriste različite skupove podataka s kojima se može upravljati na različite načine. Jedna od važnih funkcionalnosti koju moraju imati je sposobnost spremanja podataka na uređaj. U MusicEvent aplikaciji koriste se dva osnovna načina spremanja podataka koji se razlikuju po trajanju i mjestu pohrane podataka.

Prvi način spremanja je AsyncStorage. To je sustav pohrane podataka koji se koristi za spremanje malih količina podataka unutar memorije uređaja. Podatci spremljeni u uređaj imaju trajanje duže od sesije aplikacije. MusicEvent aplikacija koristi `react-native-async-storage/async-storage` paket za spremanje i čitanje podatka iz memorije. Kako bi se mogao koristiti ovaj paket potrebno je pokrenuti sljedeću komandnu liniju:

```
npx expo install @react-native-async-storage/async-storage
```

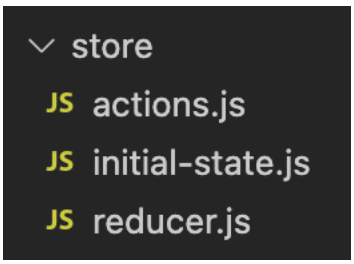
Nakon instalacije paket je spreman za korištenje. AsyncStorage može spremati samo podatke tipa `String`, u slučaju da se spremaju objektni tipovi podatka tada ih je potrebno prvo serijalizirati. Podatci se pohranjuju u parovima ključ (engl. *key*) vrijednost (engl. *value*) pozivanjem funkcije `setItem('@key', value).`

Dohvaćanje podatka se vrši funkcijom `getItem('@key')` koja prima kao parametar ključ te vraća vrijednost za taj ključ.

Drugi način spremanja podataka je pohrana u stanje (engl. State). Podatci pohranjeni u stanju (engl. *state*) su pohranjeni u memoriji aplikacije i mogu se koristiti samo unutar komponenti u kojima su pohranjeni ili u komponentama koje su im djeca (engl. *Child components*). To znači da se podatci pohranjeni u stanju ne mogu dijeliti između različitih sesija aplikacije, već se brišu nakon zatvaranja aplikacije ili osvježavanja stranice. Stoga, ako je potrebno pohraniti podatke koji će se koristiti između različitih sesija aplikacije, stanje nije najbolja opcija za pohranjivanje podataka.

U MusicEvent aplikaciji koristio se React Context API za upravljanje stanjima (engl. *State management*) jer je jednostavan za implementacija te nije potrebno dodavati dodane pakete

u projekt jer je već uključen u React. Implementacija state managementa se nalazi u mapi *store* (Slika 3.7).



Slika 3.7 Struktura datoteka za upravljanje stanjima

Context omogućava prosljeđivanje podataka kroz stablo komponenti bez potrebe za ručnim prosljeđivanjem podataka. React Context API logika se dijeli na dvije cjeline *actions* i *reducers*([3]). Reduceri su sastavnice sustava u koje se pohranjuje stanje aplikacije i podataka. Funkcija `reducer` prima dva parametra `state` (trenutno stanje) *i* `action` koji je pokrenuo funkciju te ovisno o njemu ulazi u zadani `case`[7] u kojem se nalazi kôd za izmjenu stanja `reducera`.

```
export const reducer = (state, action) => {
  switch (action.type) {
    case actions.START_LOADER:
      return {
        ...state,
        isLoading: true
      };
    case actions.END_LOADER:
      return {
        ...state,
        isLoading: false
      };
}};
```

Kôd 3.5 Primjer reducer funkcije

`createActions` je funkcija koje se poziva iz komponente i ekrana te sadrži logiku za dohvaćanje i procesiranje podataka i okidanje događaja za promjenu stanja u *reducerima*.

```
export const createAction = (type, payload) => {
  return {
```

[7] stanje

```
    type,
    payload
  };
};
```

Kôd 3.6 Primjer *action* funkcije

3.2.4. Korištenje naprednih animacija i slanje obavijesti

Jedna od glavnih značajki React Nativea su animacije koje omogućavaju programerima da svojim aplikacijama dodaju interaktivnost i poboljšaju korisničko iskustvo. Postoje različiti tipovi animacija u RN, a najpopularnije od njih su:

- Nativne animacije
- Animacije interpolacije
- Paralelne animacije
- Sekvencijalne animacije

U MusicEvent aplikaciji korištene su nativne animacije. Nativne animacije u programskom jeziku React Native koriste `Animated` API kako bi stvorile glatke i brze animacije koje rade na nativnom okruženju mobilnih uređaja. Korištenjem Native Driver[8] tehnologije, nativne animacije smanjuju potrošnju procesorskog vremena i baterije mobilnog uređaja, što ih čini idealnim za aplikacije koje zahtijevaju visoku razinu interaktivnosti. Također, nativne animacije pružaju programerima mnoge mogućnosti za stvaranje interaktivnih animacija koje poboljšavaju korisničko iskustvo.

```
const scrollRef = useRef(null);
  const [currentIndex, setCurrentIndex] = useState(0);

  useEffect(() => {
    const x = currentIndex * 200;
    const y = 0;
    scrollRef.current.scrollTo({ x, y, animated: true });
  }, [currentIndex]);

  const handleNext = () => {
    const nextIndex = (currentIndex + 1) % images.length;
```

[8] Native driver pomiče sve korake izvršavanja u nativni dio aplikacije

```
    setCurrentIndex(nextIndex);
  };

  return (
    <View style={styles.container}>
      <ScrollView
        ref={scrollRef}
        horizontal={true}
        showsHorizontalScrollIndicator={false}
        pagingEnabled={true}>
        {images.map((image, index) => (
          <Image key={index}
            source={image.source} style={styles.image} />
        ))}
      </ScrollView>
      <TouchableOpacity
            style={styles.button} onPress={handleNext}>
        <Image source={require('./images/arrow.png')} />
      </TouchableOpacity>
    </View>
  );
```

Kôd 3.7 Primjer nativne animacije

Na ovom primjeru je prikazano korištenje nativne animacije na horizontalnoj listi slika koja se pomiče kada korisnik klikne gumb. Lista se pomiče u lijevo i sljedeća se slika pozicionira na mjesto prve, kada lista dođe do kraja, sljedećim klikom na gumb animacija vraća listu na početak.

Da bi se postigla animacija pomaka liste slika, koristi se funkcija `scrollTo` koja pomiče `ScrollView` komponentu prema novoj poziciji. Promjenom `currentIndex` varijable poziva se `UseEffect` funkcija, koja se koristi da bi se slike unutar `ScrollView` komponente automatski pomicale.

Kada se klikne na gumb, poziva se funkcija `handleNext` koja povećava `currentIndex` za 1. Ako se dođe do kraja liste, `currentIndex` se vraća na 0 da bi se omogućilo ponovno animiranje kroz listu.

Ovo je samo jedan od primjera mogućnosti korištenja nativnih animacija. Korištenjem ovakvih tipova animacije poboljšava se korisničko iskustvo jer aplikaciju čine dinamičnijom i privlačnijom.

Slanje obavijesti je ključni aspekt mobilnih aplikacija koji poboljšava korisničko iskustvo i povećava korisnički angažman. Aplikacija MusicEvent koristi Expo Push notifikacije za slanje push obavijesti (engl. *Push notifications*). Expo *push* notifikacije predstavljaju brz i jednostavan način slanja obavijesti korisnicima React Native aplikacije. Unutar mrežnog servisa implementiran je proces slanja push obavijesti svim korisnicima koji koriste mobilnu aplikaciju MusicEvent preko Expo push API-ja (engl. *Application Programming Interface*, skraćeno API). Osim što je jednostavan za upotrebu, Expo Push API ima i druge prednosti, kao što su automatska konfiguracija i upravljanje registracijom uređaja, podrška za iOS i Android uređaje, ugrađena analitika, te mogućnost planiranja i personalizacije obavijesti.

Expo je platforma koja omogućuje brzo i jednostavno slanje *push* obavijesti na Android i iOS uređaje. Korištenjem Expo *push* API-ja programeri mogu integrirati *push* notifikacije u svoje aplikacije.

Za korištenje Expo Push notifikacija u mobilnoj aplikaciji, potrebno je kreirati račun na Expo platformi i instalirati Expo CLI. Zatim na mobilnoj aplikaciji potrebno je dodati dva Expo modula:

```
npx expo install expo-notifications expo-device
```

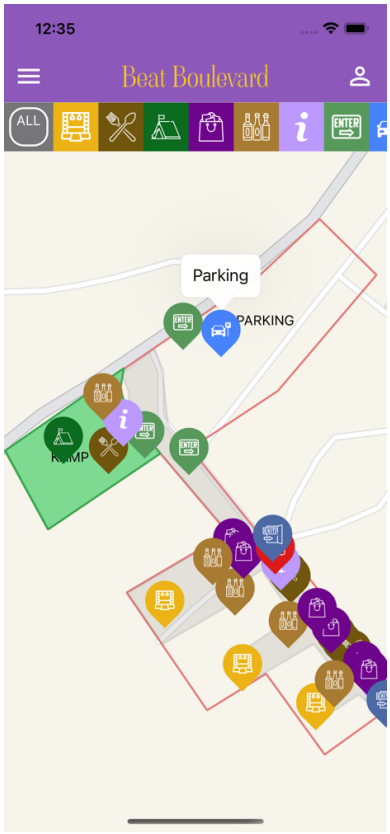
Unutar asinkrone funkcije `registerForPushNotificationsAsync` prvo se provjerava korištenjem `expo-device` paketa naredbom `Device.isDevice` provjeravamo da li je uređaj stvaran ili je simulator, zatim ako se radi o stvarnom uređaju pokrećemo funkciju `Notifications.requestPermissionsAsync` koja nam vraća status da li korisnik dopušta primanje obavijesti kroz mobilnu aplikaciju, a u slučaju da prihvaća pokreće se funkcija `Notifications.getExpoPushTokenAsync` koja unutar objekta `data` vraća `expoPushToken`. Taj token se prilikom pokretanja MusicEvent aplikacije sprema u bazu podataka.

Slanje push obavijesti svim uređajima se šalje kroz sučelje kreirano unutar mrežnog servisa. Gdje popunjavamo formu te klikom gumba `send`, preko HTTP/2 API-ja se šalje push obavijest svim uređajima.

Expo Push notifikacije su brz i jednostavan način slanja obavijesti korisnicima React Native aplikacija. Korištenje Expo Push notifikacija poboljšava korisničko iskustvo i povećava uključenost korisnika u aplikaciji. Integracija Expo Push notifikacija u React Native aplikaciju je jednostavna i ne zahtijeva puno vremena.

3.2.5. Korištenja mapa i geolokacije

Mape su ključni dio mnogih aplikacija koje se temelje na lokaciji, a u aplikacijama za glazbene evente posebno su korisne za prikaz mjesta održavanja određenih događaja te pomažu u snalaženju u prostoru. MusicEvent mobilna aplikacija nudi korisnicima prikaz iscrtane mape festivalskog područja sa vizualno ucrtanim mjestima najvažnijih stvari na festivalu(Slika 3.8). Prilikom učitavanja karte, pregled je postavljen na iscrtano područje festivala. Karta se po želji može zumirati ili smanjiti. Unutar iscrtanog područja na karti iscrtane su ikone koje predstavljaju bitne točke na lokaciji festivala. Klikom na određenu ikonu otvara se oblak koji prikazuje minimalni opis toga što se nalazi na tom mjestu. Radi lakšeg pronalaska određene stvari na festivalu. Na vrhu ekrana je postavljen horizontalni izbornik za filtriranje određenih stvari na festivalu.



Slika 3.8 Ekran u MusicEvent aplikaciji s mapom i ucrtanim lokacijama

Cijeli prikaz mapa kreiran je pomoću paketa *react-native-maps*, pokretanjem komandne linije: `npx expo install react-native-maps`.

U *react-native-maps* paketu, `<MapView>` komponenta se upotrebljava za prikaz mape. Ova komponenta pruža mogućnost učitavanja prikaza mape iz druge aplikacije koja je namijenjena za to unutar druge aplikacije. Na Android sustavu, `MapView` koristi Google Maps, dok se na iOS sustavima učitava Apple Maps.

Unutar `MapView` komponente dodavaju se dodatne komponente ovisno o željama prikaza na samoj karti. U MusicEvent aplikaciji koristile su se `Polygon` i `Marker` komponente. `Polygon` komponenta služila je za iscrtavanje granica festivalskog područja, a `Marker` komponenta za dodavanje različitih pinova unutar festivalskog područja.

Korištenje geolokacije unutar mape pruža korisnicima jasnu predodžbu o njihovoj lokaciji u odnosu na okolinu. Interaktivni prikaz na mapi pomaže korisnicima da bolje shvate svoju lokaciju i smjer u kojem se kreću. Osim toga, geolokacija unutar mape može pružiti korisnicima korisne informacije o događajima i lokacijama na festivalu koje su u blizini njihove lokacije.

Implementacija geolokacije vrši se dodavanjem dodatnog paketa u projekt kroz naredbu: `npx expo install expo-location.`

Nakon uspješne instalacije automatski se dodaju tri nova dopuštenja u projekt:

- ACCESS_COARSE_LOCATION
- ACCESS_FINE_LOCATION
- FOREGROUND_SERVICE

Ova dopuštenja se koriste za dobivanje približne lokacije uređaja, precizne lokacije uređaja te dopuštenje pretplate na ažuriranja lokacije dok je aplikacija u upotrebi.

Geo lokacija se dodaje na mapu na način da prilikom pokretanja aplikacije se zatraži dopuštenje korištenja lokacije, te zatim pozivamo funkciju `Location.getCurrentPositionAsync({})`, funkcija nam vraća u obliku objekta koordinate lokacije.

Korištenjem mapa i geolokacije posjetiteljima festivala se omogućava jednostavnije snalaženje i ugodniji boravak na festivalu.

3.3. Servisni sloj aplikacije

Servisni sloj aplikacije čini mrežni API (engl. *Web API*). Aplikacijsko programsko sučelje (engl. *Application Programming Interface*, skraćeno API) omogućuje računalnim programerima pristup funkcionalnosti objavljenih softverskih modula i usluga. API definira strukture podataka i potprogramne pozive koji se mogu koristiti za proširenje postojećih aplikacija s novim značajkama i izgradnju potpuno novih aplikacija na vrhu ostalih softverskih komponenti.

3.3.1. Korištene tehnologije

Za mrežni servis unutar sustava MusicEvent koristio se mrežni API, u nastavku će se koristit naziv web API. Da bi se razumjelo što znači web API, potrebno je razmotriti drugu riječ u nazivu: API, što je kratica za Aplikacijsko programsko sučelje (engl. *Application Programming Interface*). API sučelje je sastavljeno od skupa funkcija koje omogućuju programerima pristup i korištenje specifičnih funkcija, alata ili podataka jedne aplikacije, usluge ili operacijskog sustava.

Web API je jedna vrsta API-ja koja se razlikuje od ostalih po tome što se pristupa putem mrežnog HTTP ili HTTPS protokola. Krajnje točke aplikacija, koje su dostupne putem javnih URL[9]-ova (engl. *Uniform Resource Locator*, skraćeno URL), pokreću funkcije koje su povezane s tim sučeljem.

U MusicEvent sustavu, Web API se koristi kao posrednik za komunikaciju između mobilne aplikacije i baze podataka, a razvijao se u C# ASP.NET(.NET Framework) razvojnome okviru i za razvoj se koristio Microsoftov IDE Visual Studio 2019. Nakon završenog razvoja, WebAPI je hostan na Azure App Service.

3.3.2. Arhitektura

ASP.NET Web API je popularan mrežni programski okvir (engl. *web framework*) za izgradnju skalabilnih i sigurnih web aplikacija. ADO.NET model podataka entiteta(engl. *Entity Data Model,* skraćeno EDM), s druge strane, je objektno-relacijsko mapiranje (engl. *Object-Relational Mapping*, skraćeno ORM) alat koji se koristi za mapiranje podataka iz

[9] URL je putanja do određenog sadržaja na Internetu te se obično naziva poveznica, ponekad i mrežna adresa

baze podataka u objekte koji se koriste u programiranju. Kombinacijom ova dva alata omogućava se izgradnja kompleksne i efikasne web aplikacije koja je integrirana s bazom podataka.

Arhitektura servisnog dijela aplikacije može se opisati kao MVC arhitekturni obrazac. Model-View-Controller (skraćeno MVC) je arhitekturni obrazac za izgradnju softverskoga rješenja. Kako i sam naziv aludira, sastavljen je od triju glavnih komponenti:

- Model sloj: Ovaj sloj sadrži ADO.NET Entity Data modele koji se koriste za mapiranje podataka iz baze podataka u objekte koje koristite u projektu. Model sloj također sadrži klase koje opisuju podatke i poslovnu logiku aplikacije.
- Pogled sloj (engl. *View layer*) je odgovoran za prikazivanje podataka koji se dobivaju iz kontroler sloja korisniku. Ovaj sloj koristi HTML, CSS i JavaScript za oblikovanje i stiliziranje web stranice. MusicEvent web API koristi korisničko sučelje za prikaz forme za unos podataka koji će se zatim klikom na gumb slati kao push obavijesti svim mobilnim uređajima kojim je spremljen token uređaja u bazu podataka.
- Kontroler sloj (engl. *Controller layer*): Kontroler sloj je odgovoran za obradu zahtjeva korisnika i komunikaciju s model slojem kako bi dobio potrebne podatke. Kontroler sloj također obavlja validaciju podataka koje prima od korisnika i pomaže u spremanju podataka u bazu podataka.

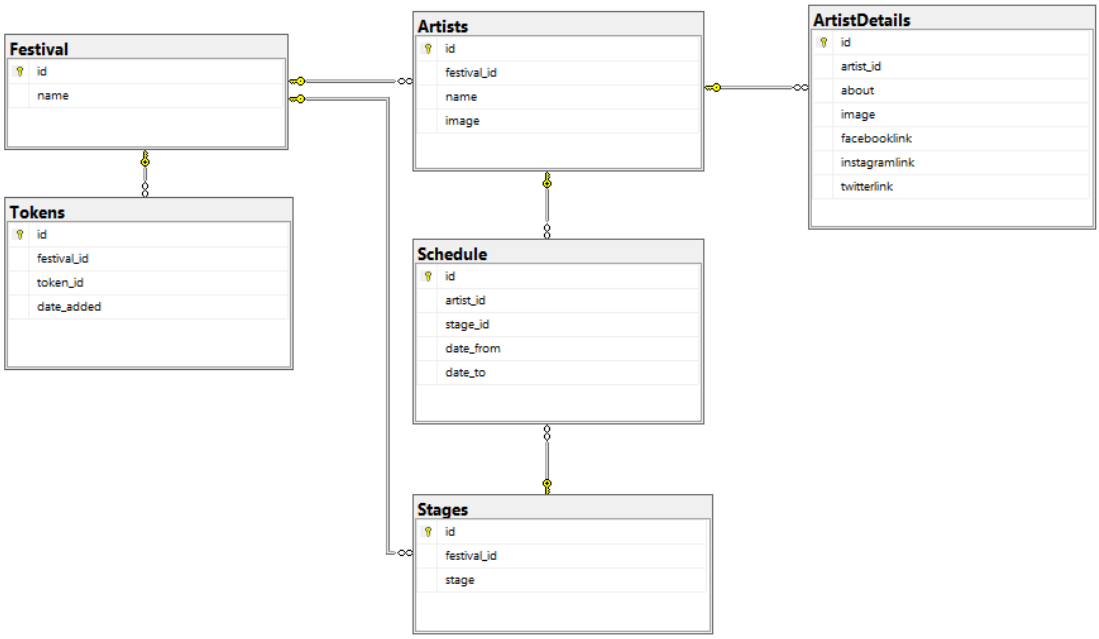
Uz korištenje ADO.NET Entity Data modela, arhitektura projekta ASP.NET MVC Web API omogućava da se razdvoje poslovna logika aplikacije od baze podataka, što olakšava održavanje i skalabilnost aplikacije. Također, ADO.NET Entity Data model pruža intuitivno sučelje za upravljanje podatcima u bazi podataka, što olakšava izradu aplikacije.

3.4. Baza podataka

Kao i mnogi drugi moderni aplikacijski sustavi, MusicEvent aplikacija zahtijeva stalno upravljanje podatcima. Ti podatci moraju biti spremljeni na jednom mjestu, a istovremeno dostupni u bilo koje doba dana kada korisnik otvori aplikaciju na svom uređaju.

Za bazu podataka odabran je Azure SQL Server koji pruža kreiranje relacijske baze podataka u oblaku. Bazi podataka se pristupa i upravlja kroz Microsoft SQL Server Management Studio (skraćeno SSMS).

Nakon pristupanja Azure serveru preko SSMS, na novo podignutu *musicFestival* bazu spajaju se ostale komponente sustava. Sustavne datoteke obuhvaćaju dvije datoteke koje sadrže upite na bazu podataka: jedna datoteka služi za stvaranje neophodnih tablica i veza, dok se druga koristi za unapređivanje baze podataka s prethodno generiranim podatcima koji su ključni za očekivano ponašanje aplikacije.



Slika 3.9 Entity Relationship (skraćeno ER) dijagram baze podataka

ER dijagram predstavlja entitete koji se nalaze u bazi podataka i njihovu povezanost. U MusicEvent sustavu, bazu čini šest entiteta(Slika 3.9): Festival, Tokens, Artists, ArtistDetails, Schedule i Stages. Tablica Artists sadrži popis svih izvođača na festivalu te njihove osnovne podatke: sliku i ime. Ona je povezana s ArtistDetails tablicom koja sadrži sve detalje vezane za određenog izvođača. Tablica Schedule povezana je s tablicama Artists i Stages te se u nju spremaju vremena nastupanja određenog izvođača te pozornica na kojoj nastupa. Tokens tablica je povezana s festivalom i sadrži sve tokene uređaja kojima će se slati push obavijesti.

Baza podataka kreirana na Azure SQL serveru ima mnoge prednosti u odnosu na tradicionalne SQL servere, a neke od prednosti su: skalabilnost, sigurnost, dostupnost te automatizacija.

4. Povratne informacije na razvijeno aplikativno rješenje

Primarni cilj analize je bio predstaviti sustav zajedno s njegovim glavnim funkcionalnostima potencijalnim korisnicima, odnosno posjetiteljima glazbenih festivala.

Zbog vremenske limitiranosti, testiranje nije moglo biti provedeno na velikom broju korisnika, ali se uspjelo dobiti nekoliko povratnih informacija od strane ciljne grupe osoba odnosno osoba koje posjećuju festivala bar jedan put godišnje. Informacije su se prikupljale razgovorom s korisnicima koji su testirali mobilnu aplikaciju.

Tijekom razgovora, dobivene su neke korisne informacije o njihovim dojmovima i mišljenjima o sustavu. Većina korisnika se složila da su glavne funkcionalnosti sustava vrlo korisne te da bi im ovakav tip aplikacije uvelike olakšao i učinio boravak na festivalu ugodnijim.

Najčešća povratna informacija bila je da bi korisnici voljeli imati integriran cjelokupan sustav kupovine ulaznica unutar aplikacije, a ne samo spremanja festivalske ulaznice unutar aplikacije.

Druga važna povratna informacija bila je da bi korisnici voljeli imati mogućnost ostavljanja povratne informacije na festival kako bi se organizator mogao poboljšati na sljedećem festivalu.

Neke manje važne povratne informacije uključivale su želju za mogućnostima dodavanja događaja u osobni kalendar i praćenja vremenske prognoze za festival.

Sve u svemu, korisnici su bili vrlo zadovoljni funkcionalnostima sustava, ali bi voljeli vidjeti neke dodatne značajke koje bi im mogle pomoći tijekom boravka na festivalu. Uzimajući u obzir ove povratne informacije, u budućnosti se može pristupiti daljnjem razvoju funkcionalnosti kako bi se zadovoljile potrebe i želje korisnika te na taj način privukla sve veća i veća pozornost posjetitelja festivala.

Zaključak

React Native je moćan alat za izradu mobilnih aplikacija, koji omogućuje brz razvoj aplikacija za više platformi. Konverzijom React komponenti u nativne elemente, dobije se korisničko iskustvo na razini onoga što nude nativne aplikacije, a pritom da se ne narušavaju performanse. Ovaj rad pokazuje kako se React Native može koristiti za izradu mobilne aplikacije za glazbeni festival i predstavlja temelj za daljnji razvoj i poboljšanje mobilne aplikacije u budućnosti.

Na temelju trenutne situacije u svijetu glazbenih festivala za koju je MusicEvent sustav osmišljen, jasno je da je potrebno tehnološko unaprjeđenje. Aplikacija bi bila korisna posjetitelje festivala koji bi kroz aplikaciju mogli osigurati jednostavniji i ugodniji boravak na festivalu.

Glavne funkcionalnosti MusicEvent mobilne aplikacije su razvijene, komunikacija među sustavima je postavljena. Naime, još bi trebalo uložiti vrijeme kako bi se dodale funkcionalnosti koje su korisnici aplikacije predložili, a koje se smatraju bitne za sustav kako bi pružio još bolje iskustvo prilikom korištenja.

Ovim završnim radom stekao sam iskustvo oblikovanja i izrade cjelokupnog aplikativnog rješenja koje se sastojalo od mobilne aplikacije, mrežnog servisa te baze podataka. Kroz izradu cjelokupnog rješenja unaprijedio sam svoje znanje programiranja te stekao potpuno nova znanja iz dizajna i planiranja razvoja. Završno, postoji motivacija za daljnji razvoj i unapređenje aplikacije kako bi se predstavila produkcijska verzija potencijalnim klijentima odnosno organizatorima festivala. Zbog toga, MusicEvent sustav smatram uspješnim projektom.

Popis kratica

API	*Application Programming Interface*	Aplikacijsko programsko sučelje
CLI	*Command-Line Interface*	sučelje naredbenoga retka
CMD	*Command Prompt*	komandna linija
EDM	*Entity Dana Model*	Model podataka entiteta
EF	*Entity Framework*	*Entity Framework*
HTTP	*HyperText Transfer Protocol*	*HyperText Transfer Protocol*
JS	*JavaScript*	*JavaScript*
MVC	*Model-View-Controller*	*Model-View-Controller* arhitektura
ORM	*Object-Relational Mapping*	objektno-relacijsko mapiranje
OS	*Operating System*	operacijski sustav
RN	*React Native*	*React Native*
SSMS	*SQL Server Management Studio*	*SQL Server Management Studio*
UI	*User Interface*	korisničko sučelje
URL	*Uniform Resource Locator*	ujednačeni lokator sadržaja
JSON	*JavaScript Object Notation*	*JavaScript Object Notation*
CSS	*Cascading Style Sheets*	*Cascading Style Sheets*

Popis slika

Popis kôdova

Literatura

[1] React Native, React Native dokumentacija - Introduction, https://reactnative.dev/docs, siječanj 2023.

[2] Mozzila, Fetch API dokumentacija, Fetch API – Web APIs – MDN Web Docs, https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API, siječanj 2023.

[3] Medium, React Context API: Quickest State Management Implementation, levelup.gitconnected.com/react-context-api-quickest-state-management-implementation-f9abd6658e66, Dan Halperin, 2019., siječanj 2023.

[4] Code Magazine, ASP.NET MVC and the ADO.NET Entity Framework, codemag.com/article/1009071/ASP.NET-MVC-and-the-ADO.NET-Entity-Framework, Morgridge Dane, 2022., siječanj 2023.

[5] Microsoft Docs, Create web APIs with ASP.NET Core, learn.microsoft.com/en-us/aspnet/core/web-api/?WT.mc_id=dotnet-35129-website&view=aspnetcore-7.0/, siječanj 2023.

[6] Microsoft Docs, Sql server technical documentation, https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16&viewFallbackFrom=sql-server-ver16%2F, siječanj 2023.

[7] Hubspot, Generative Research: Everything You Need to Know, blog.hubspot.com/service/generative-research, Cassie Wilson, 2023., siječanj 2023.

[8] Dev, Understanding React Native Architecture, https://dev.to/goodpic/understanding-react-native-architecture-22hh, Jun Kaneko, 2020., siječanj 2023.

[9] React Navigation, React Navigation dokumentacija, Fundamentals, https://reactnavigation.org/docs/getting-started, siječanj 2023.

[10] Nielsen Norman Group, Using Prioritization Matrices to Inform UX Decisions, nngroup.com/articles/prioritization-matrices, Sarah Gibbons, 2018., siječanj 2023.

[11] Techopedia, Multithreading, https://www.techopedia.com/definition/24297/multithreading-computer-architecture, Justin Stoltzfus, 2022., siječanj 2023.

[12] Npm, https://www.npmjs.com/package/react-native, siječanj 2023.