Izrada aplikacija primjenom React Native JavaScript razvojnog okvira

Đuranec, Nela

Sveučilište
Sjever

Završni rad br. 837/MM/2023

Izrada aplikacija primjenom React Native JavaScript razvojnog okvira

Nela Đuranec, 0336045776

Varaždin, rujan 2023. godine

Sveučilište
Sjever

Odjel za Multimediju, oblikovanje i primjenu

Završni rad br. 837/MM/2023

Izrada aplikacija primjenom React Native JavaScript razvojnog okvira

Student

Nela Đuranec, 0336045776

Mentor

Vladimir Stanisavljević, mr.sc.

Varaždin, rujan 2023. godine

Sveučilište Sjever
Sveučilišni centar Varaždin
104. brigade 3, HR–42000 Varaždin

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL Odjel za multimediju

STUDIJ preddiplomski stručni studij Multimedija, oblikovanje i primjena

PRISTUPNIK Nela Đuranec | JMBAG 0336045776

DATUM 21.08.2023 | KOLEGIJ Programski alati 3

NASLOV RADA Izrada aplikacija primjenom React Native JavaScript razvojnog okvira

NASLOV RADA NA ENGL. JEZIKU Application Development in React Native JavaScript framework

MENTOR Vladimir Stanisavljević | ZVANJE viši predavač

ČLANOVI POVJERENSTVA

1. doc.dr.sc. Andrija Bernik - predsjednik povjerenstva
2. pred. Dražen Crčić, dipl.ing. - član povjerenstva
3. mr.sc. Vladimir Stanisavljević, v.pred. - mentor
4. doc.dr. sc. Domagoj Frank - zamjenski član
5.

Zadatak završnog rada

BROJ 837/MM/2023

OPIS

React Native, jedna je od popularnih tehnologija na temelju JavaScripta za razvoj hibridnih mobilnih aplikacija. Proširenje je popularnog React razvojnog okvira s dodatnim komponetama korisničkog sučelja projektiranim za rad na manjim mobilnim uređajima.

U ovom radu potrebno je:
* opisati povijesni razvoj i osnove React Native programske zbirke te njezinu primjenu za izradu korisničkog sučelja mobilnih aplikacija,
* analizirati prednosti i ograničenja React Native i usproediti ga sa sličnim tehnologijama,
* detaljno opisati dostupne komponente i osnovnu proceduru izrade aplikacija pomoću React Native,
* opisati postupak instalacije i načine ugrađivanja u produkcijske mobilne aplikacije,
* osmisliti i oblikovati proizvoljnu aplikaciju na kojoj će biti demonstrirane glavne mogućnosti React Native-a,
* usporediti rad izrađene aplikacije na uređajima različitih razlučivosti i objasniti kako React Native potpomaže prikaz u različitim skalama i orijentacijama.
* istražiti mogućnosti primjene za aplikacije s podrškom za dnevni i noćni rad te veličinu teksta prilagođenu postavkama sustava.

Detaljno dokumentirati sve korištene tehnologije i izrađeni programski kod te opisati stečena iskustva i postignute rezultate.

ZADATAK URUČEN 22.05.2023. | POTPIS MENTORA Vlado

SVEUČILIŠTE SJEVER

Predgovor

Svrha ovoga završnog rada je da svi zainteresirani bolje upoznaju ovu temu i potencijalno nauče nešto novo. Na internetu su lako dostupni primjeri i tutoriali na ovu temu, no ne postoji dokumentacija na hrvatskom jeziku. Svoju budućnost vidim u programiranju web stranica, no želim saznati ponešto i o mobilnim aplikacijama te React Native razvojnom okviru. U slobodno vrijeme proučavala sam React.js, stoga mi ova tema nije u potpunosti strana.

Zahvaljujem obitelji, dečku i kolegama na faksu na konstantnoj potpori i motivaciji, svim profesorima na prenesenom znanju, a ponajviše mentoru Vladimiru Stanisavljeviću za pomoć i podršku u izradi ovoga završnog rada.

Sažetak

Ovaj rad prikazuje postupak izrade mobilne aplikacije primjenom React Native Javascript razvojnog okvira, s fokusom na izradu aplikacije za vremensku prognozu. Na primjeru ove aplikacije prikazuju se glavne komponente i značajke React Native okvira.

Teorijski dio sadrži opisane najpoznatije Javascript razvojne okvire te podjelu aplikacija na native i više-platformske tj. hibridne. Prije praktičnog dijela rada opisani su osnovni elementi za izradu aplikacija i potrebni alati za React Native. U praktičnom dijelu prikazana je izrada aplikacije od inicijalizacije pa do funkcionalne aplikacije. Kako raste popularnost mobilnih uređaja i aplikacija, tako raste i potraga za programerima u tom području.

Ovaj završni rad prikaz je naučenih tehnika grafičkog i web dizajna te programiranja tijekom studiranja Multimedije, dizajna i oblikovanja. Bitan faktor je i još puno dodatno uloženih sati proučavanja navedenih tehnika.

Ključne riječi: HTML, CSS, JS, React.js, React Native, native aplikacije, hibridne aplikacije, komponente, API

Summary

This final paper outlines the development process of creating a mobile application using the React Native Javascript framework, with a focus on creating a weather forecast application. It showcases the key components and features of the React Native framework.

The theoretical part contains the description of the most famous programming frameworks and the breakdown of applications into native and multi-platform i.e. hybrid. It also provides an overview of the basic elements for creating applications and the necessary tools for React Native development. The practical part shows the step-by-step creation of the application, from initialization to a functional application. As the popularity of mobile devices and mobile applications increases, so does the demand for developers in that field.

This paper is a presentation of the learned techniques in graphic design, web design and programming while studying Multimedia, Design and Application. An important factor is the many additional hours spent studying the mentioned techniques.

Keywords: HTML, CSS, JS, React Native, native apps, hybrid apps, components, API

Popis korištenih kratica

HTML	HyperText Markup Language - prezentacijski jezik za izradu web stranica
CSS	Cascading Style Sheets - jezik za izradu oblikovanja web stranica
JS	Javascript - skriptni programski jezik za izradu interaktivnih stranica i aplikacija
RN	React Native - razvojni okvir za izradu hibridnih aplikacija za Android, iOS i Web
CLI	Command Line Interface - sučelje naredbenog retka; oblik sučelja između operativnog sustava i korisnika u koji korisnik upisuje naredbe u obliku redaka teksta
API	Application Programming Interface - aplikacijsko programsko sučelje služi za spajanje aplikacije i sustava računala, definira način obavljanja pojedinih procesa, u programu se traže podaci od API-ja koje korisnik na kraju vidi
JSX	Javascript syntax exstension - ekstenzija na Javascript koja omogućuje pisanje HTML koda unutar Javascript-a
UI	User Interface - korisničko sučelje
UX	User Experience - korisničko iskustvo kod korištenja korisničkog sučelja
JSON	JavaScript Object Notation - format datoteke koji služi za razmjenu podataka, oblikovan je u parovima atributa i vrijednosti
NPM	Node Package Manager - upravitelj paketa za Javascript

Sadržaj

1. Uvod

Ljudi danas ne mogu funkcionirati bez pametnih telefona. Klasično računanje svodi se na uporabu mobilnih kalkulatora. Hrvati se još uvijek privikavaju na korištenje eura pa koriste aplikaciju za preračunavanje eura u kune. Potreba za preračunavanjem navela je ljude da više koriste kartično plaćanje, većinom kroz mobilne aplikacije.

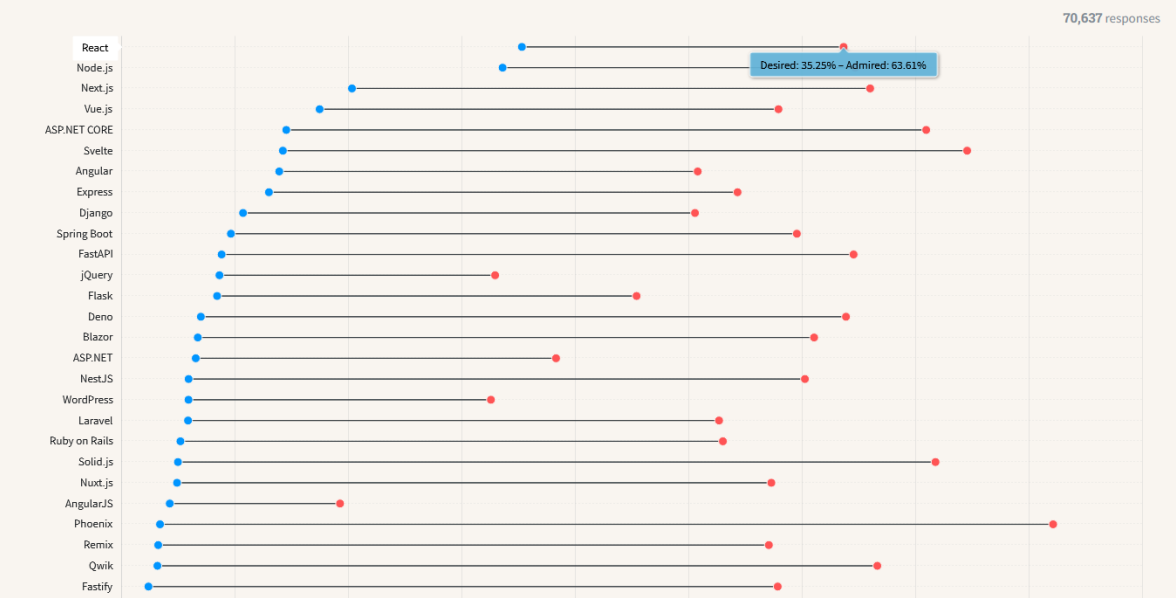
Messenger, Whatsapp i Viber osnovni su komunikacijski kanali, dok TikTok, Instagram, Youtube i Facebook služe za zabavu. Prema podacima iz 2022. čak 81.8% korisnika Facebook koristi isključivo na mobilnim uređajima, 16.7% i na mobitelu i računalu, a 1.5% na računalu ili ostalim uređajima. Veliki razvoj vidi se i na web trgovinama i aplikacijama za naručivanje hrane koje su većinski korištene na mobilnim uređajima. Sve navedeno je dokaz da je velika potražnja za aplikacijama i programerima u tom području. [1]

Za pisanje teorijskog i izradu praktičnog dijela ovoga rada potrebno je savladati vještine dizajniranja korisničkog sučelja i stvaranje pozitivnog korisničkog iskustva (UI - User Interface i UX - User Experience). Potrebno je steći napredno poznavanje programskih jezika HTML (HyperText Markup Language), CSS (Cascading StyleSheets), JS (Javascript) te React Native razvojnog okvira. Za sve navedeno potrebno je osnovno poznavanje programiranja. Dodatno se koristi biblioteka Font Awesome za lakše postavljanje ikona i OpenWeather API (Application Programming Interface) za dohvat podataka o vremenu.

Cilj je prikazati postupak izrade hibridne aplikacije pomoću React Native razvojnog okvira te na primjeru aplikacije za vremensku prognozu opisati glavne značajke tog okvira.

2. Javascript okviri

Najpoznatiji JS okviri (engl. *frameworks*) za izradu web aplikacija su Angular, Vue.js i React.js. Na kraju svakoga internetskog članka usporedbe ova tri okvira, nikad nema pobjednika, već je izbor na programeru. Svaki okvir ima svoje prednosti i nedostatke pa je bitno znati što je u projektu bitno i što se želi postići. Programeri na Stack Overflow anketi svake godine glasaju koji razvojni okvir je najkorišteniji, a usporedno koliko im se sviđa. [2] React je već tri godine na vodećoj poziciji jer je ujedno traženi okvir, a ljudi ga i vole. Sve više ljudi sviđa se Phoenix i Svelte, dok s druge strane ne vole Angular koji se još uvijek koristi u velikoj mjeri.



Slika 2.1 Prikaz potražnje pojedinog okvira

2.1. Angular

Angular je najstariji od navedenih okvira, što ga čini pouzdanim, no učenje je zahtjevnije od preostala dva. Time što je razvojni okvir, nudi sve potrebno pa nisu potrebne dodatne biblioteke. Ne koristi klasičan JS, već Typescript koji omogućuje korištenje tipova varijabli te brže otkrivanje problema. Većinom se koristi u većim tvrtkama s iskusnim programerima. Nema virtualni, već običan DOM pa se cijela aplikacija ažurira što je usporava. Nije fleksibilan jer već ima predefinirane komponente. Na temelju Angular-a izrađen je i okvir Ionic za mobilne aplikacije. [3]

2.2. Vue.js

Vue.js je najmlađi od navedenih okvira. Jednostavan je za učenje, fleksibilan i brže se učitava što ga čini odličnim izborom za manje projekte i timove. U usporedbi s Angular i React.js, nema toliko gotovih komponenta i biblioteka te su korisnička podrška i dokumentacija slabije razvijene. Weex i Ionic se koriste za izradu web i hibridnih mobilnih aplikacija na osnovu Vue.js okvira. [3]

2.3. React.js - okvir ili biblioteka?

React.js je nastao 2013. godine od strane Facebook-a (sadašnja Meta). Zbog velike popularnosti i podrške, budućnost ovog okvira izgleda obećavajuće. Iz tog razloga mnoge tvrtke koriste baš ovaj okvir, a time je i velika potražnja za React programerima. React zapravo nije JS okvir, već JS biblioteka. Nije limitiran kao okviri koji imaju definirani način rješavanja za sve probleme. Koristi se JSX koji omogućuje pisanje HTML koda unutar JS. Kao biblioteka dopušta slobodu programerima jer sami mogu definirati komponente i ključne dijelove projekta. Zbog toga se React koristi za UI, a za dodatne funkcije potrebne su dodatne biblioteke. Za održavanje globalnog state-a dodatno se koristi Redux, dok se za routing koristi react-router-dom ili Next.js okvir koji donosi dodatne mogućnosti. Svojstva React-a su da je deklarativan i sastavljiv - sastoji se od manjih komponenta koje čine cjelinu i mogu se koristiti na više mjesta. React sadrži virtualan DOM (Document Object Model) koji ažurira samo promjene, a ne cijelu aplikaciju. [3]

React.js se rendera pomoću ReactDOM-a:

```
ReactDOM.render(<App />, container)
```

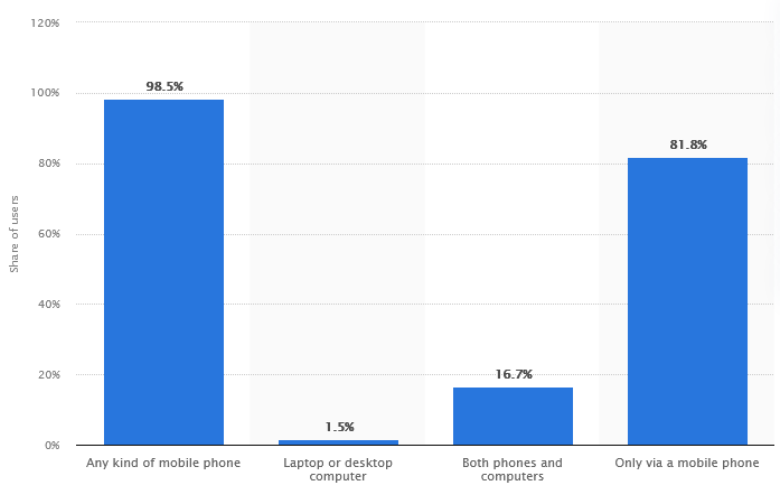
dok je React Native sam renderer:

```
const { AppRegistry } = require('react-native')
AppRegistry.registerComponent('app', () => MainComponent)
```

Zbog „nedostatka“ DOM-a u React Native-u ne koriste se oznake div, span, input, već za svaku oznaku postoji zamjena - View, Text i slično. „Learn once, write anywhere“ - znanje React-a znatno olakšava korištenje React Native okvira za mobilne aplikacije. [4]

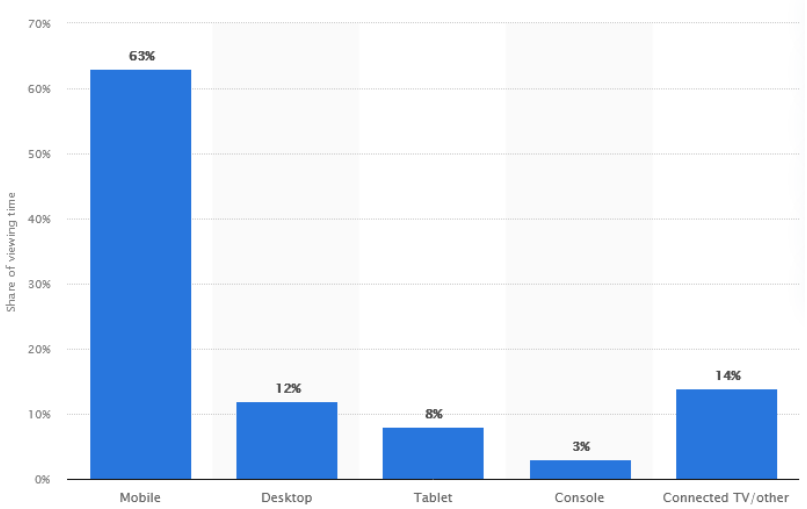
3. Mobilne aplikacije

Popularnost mobilnih uređaja postaje sve veća, a tako i korištenje mobilnih aplikacija. Grafovi na Slika 3.1 i Slika 3.2 prikazuju korištenje aplikacija Facebook i Youtube prema uređajima. Najveći dio korisnika je upravo na mobilnim uređajima. [1] [5]



Slika 3.1 Statistika korištenja Facebook-a prema vrsti uređaja (2022.)

Izvor: https://www.statista.com/statistics/377808/distribution-of-facebook-users-by-device/



Slika 3.2 Statistika korištenja Youtube-a prema vrsti uređaja (2021.)

Izvor: https://www.statista.com/statistics/1173543/youtube-viewing-time-share-device/

Mobilne aplikacije mogu se podijeliti na native i hibridne. Ovisno o kompleksnosti projekta, financijama i timu ljudi, bira se hoće li aplikacija biti native ili hibridna. Za kompleksnu aplikaciju i brze performanse bira se native, a za jeftiniju i bržu izradu bira se hibridna aplikacija.

3.1. Native aplikacije

Native aplikacije su razvijene za određene platforme: iOS ili Android. Iz tog se razloga iOS aplikacija ne može koristiti na Android uređaju i suprotno. Potreba za dva odvojena tima je nedostatak programiranja u nativnim jezicima.

3.1.1. Android

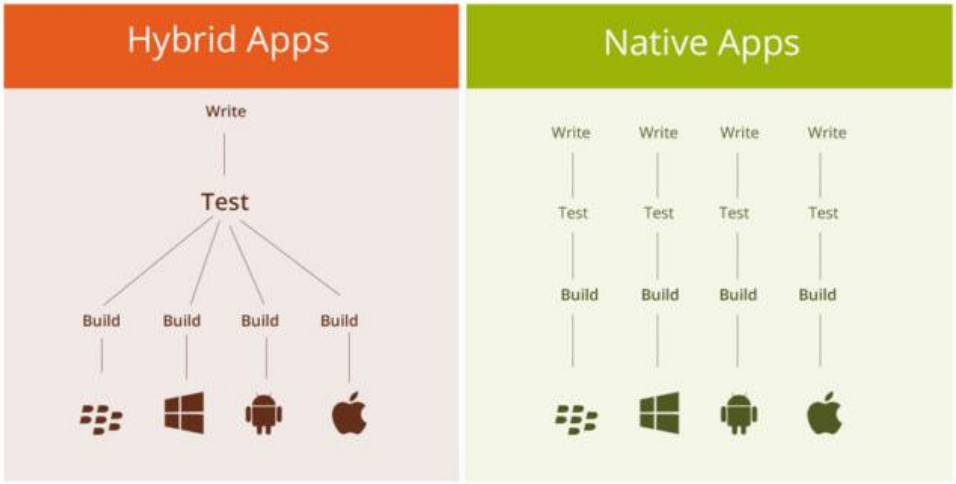
Za izradu Android aplikacija potrebno je poznavati programski jezik Java ili Kotlin. Kod izrade aplikacije koristi se Android Studio s emulatorom.

3.1.2. iOS

Za izradu iOS aplikacija potrebno je poznavati programski jezik Objective-C ili Swift te programsko sučelje Xcode.

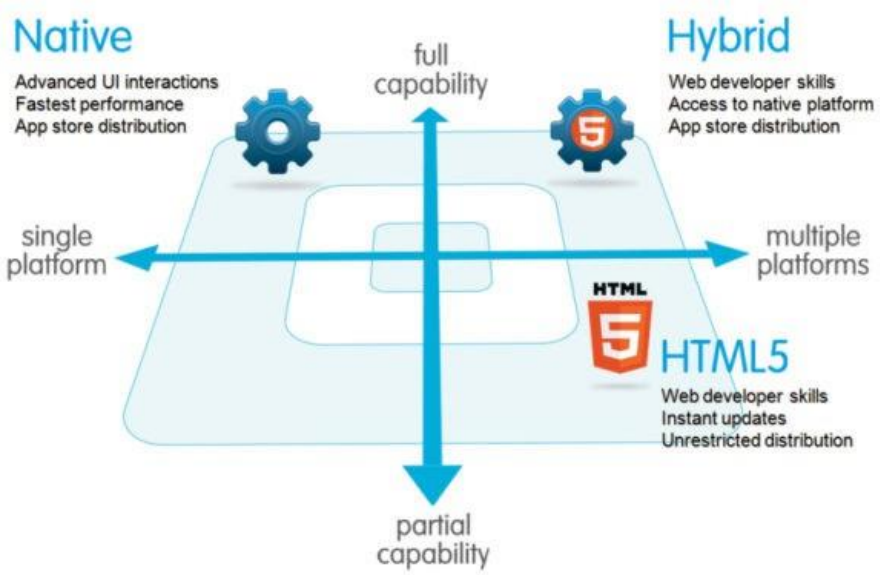
3.2. Hibridne aplikacije

Hibridne aplikacije uključuju izradu iOS i Android aplikacije koristeći isti kod koji radi na obje platforme. Nije potrebno znati nove programske jezike kao što su Kotlin i Swift, već se koriste HTML, CSS i JS.



Slika 3.3 Razlika hibridnih i nativnih aplikacija

Izvor: https://www.webiotic.com/best-way-to-develop-hybrid-mobile-apps/



Slika 3.4 Prikaz razlike native, web i hibridnih aplikacija

Izvor https://www.testgrid.io/blog/how-to-create-a-strong-mobile-app-hybrid-architecture/

3.2.1. Višeplatformski (hibridni) razvojni okviri

Neki od najpoznatijih višeplatformskih okvira su Cordova, Ionic, Xamarin, Flutter i React Native. Tradicionalne hibridne aplikacije rađene u Cordova i Ionic okvirima koriste WebView. To je komponenta koja omogućava prikazivanje HTML, CSS i JS sadržaja unutar aplikacije. S druge strane, Flutter i React Native koriste native komponente. Xamarin i Flutter zahtijevaju učenje novih jezika. Xamarin koristi C# i .NET okvir, dok Flutter koristi programski jezik Dart. Zbog navedenog, React Native je najbolji izbor za programere koji žele iskoristiti znanje i iskustvo iz frontenda - HTML, CSS, JS te dobiti native osjećaj aplikacije. [6]

3.3. React Native

Facebook je 2015. godine izradio React Native razvojni okvir. React Native, iako naziv sadrži „native“, nije native, već hibridan razvojni okvir koji koristi native UI komponente.

Piše se u JS programskom jeziku. Neke od poznatih aplikacija koje su programirane u React Native-u su Instagram, Facebook, Skype i Pinterest. Puno je korisnika pa je lakše pronaći odgovore

na pitanja. React Native je, kao i React, deklarativan i sastavljiv. Sastoji se od komponenta koje se koriste kroz cijelu aplikaciju.

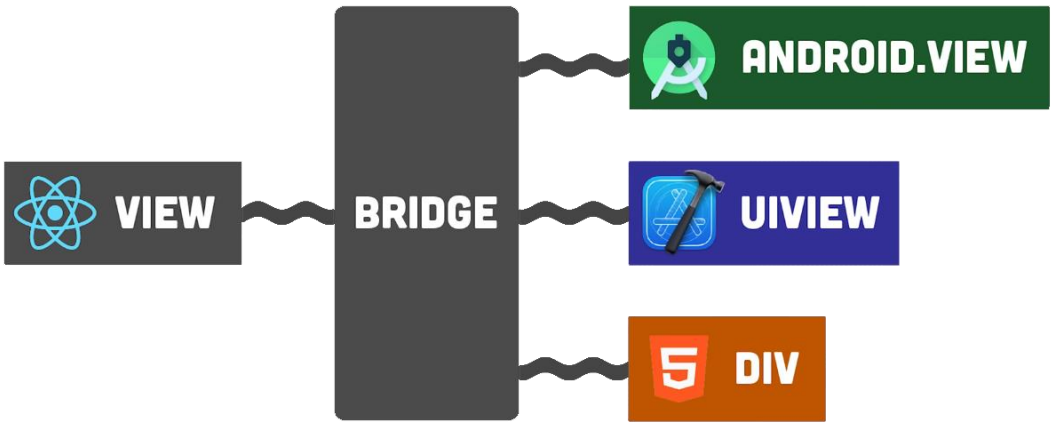
3.3.1. Prednosti i mane

Hibridne aplikacije su isplativije jer zahtijevaju samo jedan tim, a time je i sam proces izrade aplikacije brži. Također je olakšano i održavanje aplikacije. Koriste se specifične native komponente poput View i Text koje se jednako ponašaju na svim platformama.

Performanse tj. izvedba aplikacije je sporija nego kod native aplikacije zbog toga što ovisi o npr. Expo platformi koja je dodatni sloj. Ako aplikacija zahtijeva kompleksne komponente, hibridni način tu često nema podršku. Dolaskom novije verzije Native-a moguće je da se neke biblioteke neće poklapati pa je potrebno čekati i njihovu noviju verziju.

3.3.2. Nova React Native arhitektura

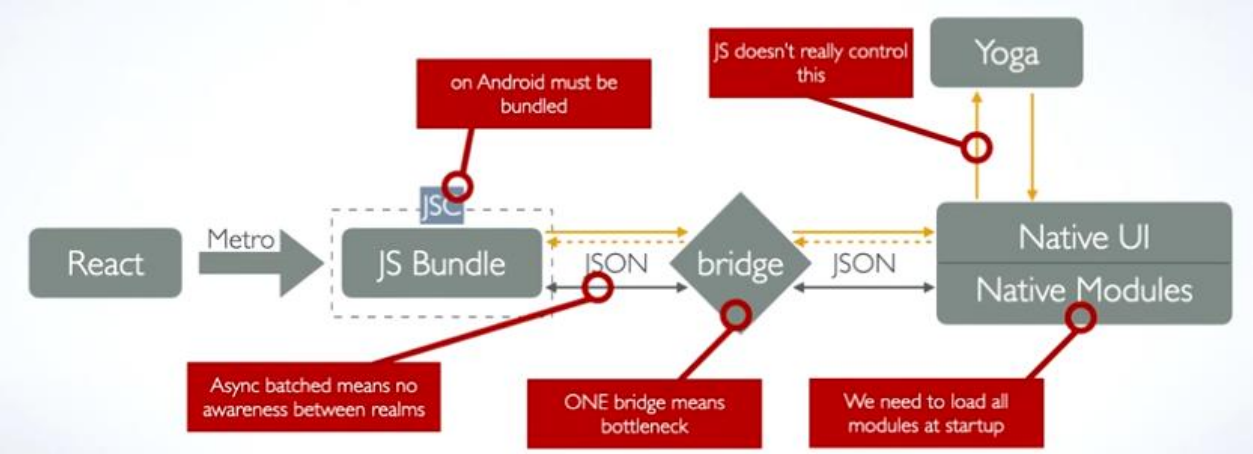
React Native uvodi novu arhitekturu. Dokumentacija je još uvijek eksperimentalna pa su korisnici pozvani da doprinose i sudjeluju u raspravama. Stara arhitektura sadrži React Native Bridge koji komponente pretvara u odgovarajuće komponente za korištenu platformu, npr. View pretvara u UIView kod iOS, na Androidu u Android.View, a na webu se ponaša kao div. [7]



Slika 3.5 Pretvaranje React Native komponente u native komponente

Izvor: https://www.youtube.com/watch?v=gvkqT_Uoahw

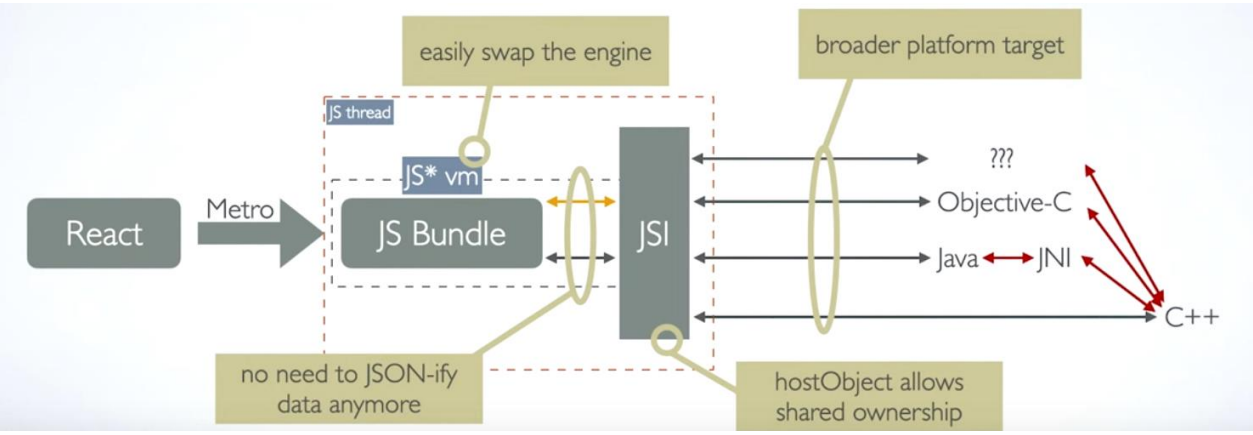
Taj „most“ kod dugih lista, ili općenito kod većih aplikacija, sporo učitava podatke. Te podatke je potrebno uvesti u JSON (JavaScript Object Notation) obliku. Također, učitavaju se svi moduli, a ne samo koji su potrebni. Zato je nastala nova arhitektura. [8]



Slika 3.6 Stara React Native arhitektura

Izvor: https://www.youtube.com/watch?v=8wAj0KG5H6U

Glavna promjena u novoj arhitekturi je JSI (Javascript Interface) koji omogućuje direktnu komunikaciju između JS i C++ tj. native strane. Turbomoduli izdvajaju i pokreću samo module koji se trenutno koriste. Više nije potrebno pretvarati podatke u JSON.



Slika 3.7 Nova React Native arhitektura

Izvor: https://www.youtube.com/watch?v=8wAj0KG5H6U

4. Usporedba React.js i React Native

4.1. Postavljanje projekta

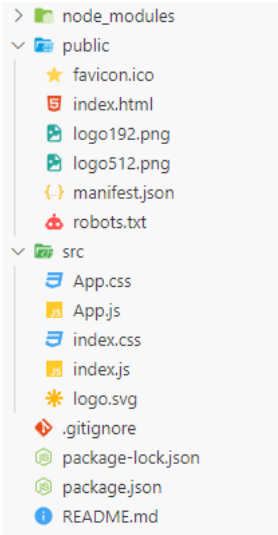
React projekt započinje se od nule ili korištenjem predefinirane konfiguracije create-react-app. Potrebno se pozicionirati u mapu gdje će se nalaziti nova mapa s React projektom. Nakon toga u konzolu se upiše:

```
npx create-react-app reactproject
```

Potrebno se pozicionirati u novo stvorenu mapu:

```
cd reactproject
```

Nakon instaliranja vidljive su 3 mape i još nekoliko datoteka. Mapa node_modules sadrži sve potrebne datoteke za rad React aplikacije. Mape public i src sadrže datoteke koje opisuju i prikazuju sadržaj aplikacije. Unutar mape public nalaze se favicon, slike i index.html datoteka gdje se nalazi div element s klasom „root“ u koju se ubacuje sav sadržaj iz App.js.



Slika 4.1 Popis datoteka nakon instaliranja create-react-app

React projekt pokreće se naredbom:

```
npm start
```

nakon čega se pokreće index.html s predloškom koji sadrži animiran logo i tekst. Za početak programiranja, izbriše se sadržaj iz predloška i može se početi s programiranjem. Sličan postupak je i za React Native.

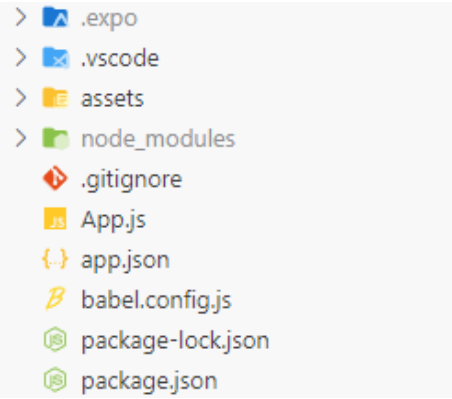
4.1.1. Expo

Expo je platforma koja olakšava izradu i pokretanje React Native aplikacija u samo par minuta. Druge opcije uključuju korištenje Android Studio i Xcode razvojnih okruženja te emulatora. Takvo postavljanje traje oko sat vremena, kompliciranije je, a zauzima i više memorije. Xcode je dostupan samo na macOS sustavima. [9]

Za početak je lakše rješenje Expo. Brine se za konfiguraciju i u potpunosti smanjuje ručno postavljanje projekta. Uključuje vlastiti server koji se pokreće naredbom npx expo start. Nudi opciju objavljivanja aplikacija na Expo naredbom expo publish.

Nije potrebno instalirati Android Studio ni Xcode, već je dovoljan samo VS Code i mobilni uređaj s Expo Go aplikacijom. Stvaranje projekta pomoću platforme Expo:

```
npx create-expo-app weather
cd weather
```



Slika 4.2 Popis instaliranih datoteka i mapa

Pokretanje aplikacije vrši se naredbom expo start, a gašenje ctrl+C.

```
npx expo start
```

Ova naredba generira QR kod. Potrebno je na mobilni uređaj instalirati Expo Go aplikaciju. Taj se kod skenira te se otvara Expo Go aplikacija s predloškom aplikacije. Prikazivanje aplikacije u Expo nije moguće ako mobitel i računalo na kojem se programira nisu spojeni na isti Wi-Fi.

Osim pokretanja na mobitelu, moguće je pokretanje i na webu (u konzolu se upiše slovo „w“). Za to je potrebno instalirati:

```
npx expo install react-native-web @expo/webpack-config
```

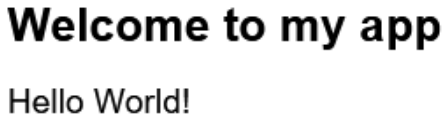
Može se pokrenuti i emulator unutar Android Studia te se unutar konzole pritisne slovo „a“. Ako je potrebno osvježiti (engl. *refresh*) aplikaciju, unutar konzole upiše se slovo „r“. Komponente

Bitno je da se komponente uvijek pišu velikim početnim slovom, u suprotnom dolazi do errora. Javascript se unutar JSX-a upisuje unutar vitičastih zagrada {}. HTML unutar JSX-a razlikuje se od običnog HTML-a jer umjesto class koristi className i svi ostali atributi pišu se u camelCase obliku. To je način pisanja varijabli, funkcija i slično na tako da započinju malim slovom, a svaka nova riječ započinje velikim slovom, bez razmaka (npr. reactNativeComponent). Export default određuje koja komponenta je „glavna“, tj. koja se može koristiti i izvan datoteke u kojoj je definirana. Kod korištenja dva ili više elementa unutar komponente, potrebno ih je objediniti unutar jednog glavnog div-a ili fragmenta < > </ >.

4.1.2. React komponente

Primjer React komponente MyApp unutar koje se dodaje komponenta MyElement:

```
import React from „react“;

function MyElement() {
  return (
    <div>
      Hello World!
    </div>
  );
}

export default function MyApp() {
  return (
    <>
      <h1>Welcome to my app</h1>
      <MyElement />
    </>
  );
}
```



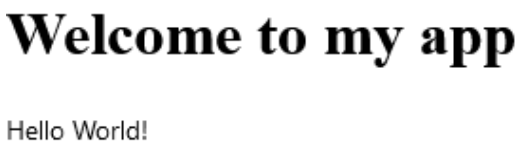
Slika 4.3 Ispis koda u React.js

Izvor: https://codesandbox.io/s/modest-cerf-thjk23

4.1.3. React Native komponente

React Native ima predefinirane oznake, npr. <View> koji se ponaša kao div i unutar njega <Text> u kojem se nalazi tekst. Svaka oznaka se uključuje iz „react-native“ biblioteke. Za isti rezultat kao kod React.js, u React Native-u je potrebno napraviti navedene promjene:

```
import {Text, View} from 'react-native';

function MyElement() {
  return (
    <View>
      <Text>Hello World!</Text>
    </View>
  );
}

export default function MyApp() {
  return (
    <View>
      <h1>Welcome to my app</h1>
      <MyElement />
    </View>
  );
}
```



Slika 4.4 Ispis koda u React Native

Izvor: https://snack.expo.dev/UDlMoHtXa

Rezultat je isti. Font je drugačiji jer se za pisanje koda koriste različita web razvojna okruženja, Sandbox za React.js i Snack za React Native. U navedenim web okruženjima još je više ubrzan postupak izrade aplikacija jer nije potrebno instalirati module, što je super za početnike jer ne moraju prolaziti konfiguraciju. Također rezultat je odmah vidljiv za web, a kod Snack-a i za iOS i Android uređaje. Još jedna opcija je skeniranje koda i otvaranje unutar Expo Go aplikacije.

Neke od najbitnijih komponenta su već navedene View i Text, te dodatno Image, ScrollView i TextInput. Postoji još puno dodatnih komponenta: Pressable, TouchableOpacity, Button, itd.

Tablica 1 Najbitnije React Native komponente

REACT NATIVE UI COMPONENT	ANDROID VIEW	IOS VIEW	WEB ANALOG	DESCRIPTION
`<View>`	`<ViewGroup>`	`<UIView>`	A non-scrolling `<div>`	A container that supports layout with flexbox, style, some touch handling, and accessibility controls
`<Text>`	`<TextView>`	`<UITextView>`	`<p>`	Displays, styles, and nests strings of text and even handles touch events
`<Image>`	`<ImageView>`	`<UIImageView>`	`<img>`	Displays different types of images
`<ScrollView>`	`<ScrollView>`	`<UIScrollView>`	`<div>`	A generic scrolling container that can contain multiple components and views
`<TextInput>`	`<EditText>`	`<UITextField>`	`<input type="text">`	Allows the user to enter text

Osim osnovnih komponenta, programer može izraditi vlastite ili koristiti tuđe komponente i biblioteke. Postoje dvije vrste komponenta: funkcionalne i klasne. Klasne komponente su stariji pristup te se danas sve manje koriste jer ne sadrže kuke kao funkcionalne. Funkcionalne se pišu kao funkcija ili kao konstanta, dok klasna sadrži render:

```
const Custom = () => {     ili    function Custom(){
  return <Text>Ovo je custom komponenta</Text>
}
class Custom extends Component {
  render() {
    return <Text>Ovo je custom komponenta</Text>
  }
}
```

4.1.4. Linking

Još jedna razlika je stavljanje poveznice na određeni tekst. Postupak za React.js je isti kao u HTML. Tekst se zatvori unutar oznaka te se unutar href atributa stavi vrijednost, tj. link na željenu stranicu. S druge strane, React Native radi na kompliciraniji način, korištenjem Linking biblioteke. [10]

Za početak je potrebno uključiti Linking u datoteku:

```
import { Linking } from "react-native";
```

Kako bi se moglo kliknuti na link, prvo je potrebno tekst „OpenWeather“ zatvoriti u oznaku <Pressable> ili <TouchableOpacity>. Razlika je da TouchableOpacity smanji vidljivost elementa pa se dobije efekt klika. Sadrže atribut onPress u koji se upiše ime funkcije s linkom kao svojstvom.

```
<TouchableOpacity onPress={() =>
handleWebsitePress("https://openweathermap.org/")}>
     <Text styles={styles.APIcolor}>OpenWeather</Text>
</TouchableOpacity>
```

Funkcija provjerava je li stranica pouzdana, odnosno, sadrži li link http ili https. Ako nije pouzdana, u konzoli se ispiše „Don't know how to open this URL:“ i ispiše se link.

```
const handleWebsitePress = async (url) => {
    const supported = await Linking.canOpenURL(url);
    if (supported) {
      await Linking.openURL(url);
    } else {
      Alert.alert(`Don't know how to open this URL: ${url}`);
    }
};
```

4.2. React hooks i props

Najbitnije funkcije React-a su props i hooks. Props ili properties su vrijednosti koje se šalju iz komponente roditelja u komponentu djeteta. Služe za korištenje jedne komponente više puta, ali s različitim vrijednostima npr. imena. Hooks tj. kuke su funkcije koje zaviru u varijable. Osnovne kuke su useState, useEffect i useContext, a moguće je izraditi i vlastite kuke. Primjena je ista u React.js i React Native, a bitno je da se kuke pozicioniraju unutar komponente i prije return().

4.2.1. useState

Kuka useState služi za spremanje stanja varijable. Koristi se za podatke koji se mijenjaju s vremenom ili na zahtjev korisnika. U sljedećem primjeru mijenja se je li mačka gladna ili sita. Klikom na gumb mačka se nahrani i promijeni se stanje te varijable.

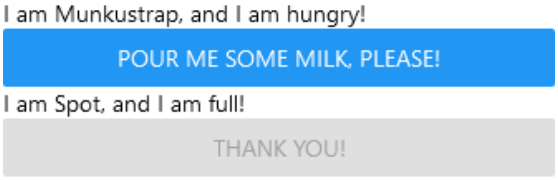
Primjer korištenja useState kuke i props na jednostavnom primjeru [11]:

```
import {useState} from 'react';
import {Button, Text, View} from 'react-native';

function Cat (props) {
  const [isHungry, setIsHungry] = useState(true);
  return (
    <View>
      <Text>
        I am {props.name}, and I am {isHungry ? 'hungry' : 'full'}!
      </Text>
```

```
      <Button
        onPress={() => {
          setIsHungry(false);
        }}
        disabled={!isHungry}
        title={isHungry ? 'Pour me some milk, please!' : 'Thank you!'}
      />
    </View>
  );
};
export default function Cafe () {
  return (
    <>
      <Cat name="Munkustrap" />
      <Cat name="Spot" />
    </>
  );
};
```

Varijabla isHungry sprema vrijednost „true“ ili „false“, dok se funkcija setIsHungry koristi za promjenu te varijable kod klika na gumb. Početno stanje je „true“, što označava da je mačka gladna. Gumb sadrži funkciju onPress koja pomoću setIsHungry(false) rerendera komponentu i mijenja to stanje u „false“. Ovisno o vrijednosti isHungry prikazuje se drugačiji tekst. Kad je vrijednost „true“ prikazat će se tekst „hungry“. Klikom na gumb ta se vrijednost mijenja u „full“, gumb je onesposobljen i promijeni tekst u „Thank you“. useState je privremen i individualan za svakog korisnika. Vrijednost se ne sprema u bazu podataka pa se kod refreshanja aplikacije useState briše.

Korištenje propova vidljivo je na imenima mačaka. „Cafe“ je komponenta roditelj unutar koje se dvaput poziva komponenta „Cat“ koja prima dvije različite vrijednosti imena: Munkustrap i Spot. Ta se vrijednost unutar komponente „Cat“ poziva sa {props.name}.



Slika 4.5 Rezultat promjene stanja varijable isHungry

4.2.2. useEffect

Unutar kuke useEffect pozivaju se podaci s API-ja. Potrebno je postaviti „dependency“ tj. prazni niz [ ] ili unutar njega varijablu o kojoj ovisi pozivanje te kuke. Ako se ta varijabla ne postavi, useEffect će se pozivati na svaki render u beskonačnost, što često dovodi do errora.

Moguće je imati useEffect bez dependency-a,

```
useEffect(() => {...
})
```

prazan dependency niz poziva useEffect samo jednom te je stoga idealno rješenje,

```
useEffect(() => {...
},[])
```

a niz s vrijednostima poziva useEffect na promjenu određenih varijabli.

```
useEffect(() => {...
},[varijabla1, varijabla2])
```

Ovisno o primjeru, dovoljno je postaviti prazan niz, no taj niz može sadržavati i jednu ili više varijabli ako promjena ovisi o toj varijabli.

Na primjeru aplikacije za vremensku prognozu useEffect će se koristiti kod fetchanja podataka:

```
  useEffect(() => {
    loadForecast();
  }, [language, cityName]);
```

Funkcija loadForecast fetcha API podatke, a dependency niz sadrži dvije varijable: language i cityName. Aplikacija se osvježava kod promjene grada ili jezika aplikacije.

4.2.3. useContext

Kuka useContext služi za slanje informacija, bez da se šalje pomoću propova. Definira se u zasebnoj datoteci i u glavnoj roditelj komponenti te je tako dostupna u cijeloj aplikaciji. Context se može koristiti za promjenu lokacije, fonta, jezika i svega što se koristi kroz cijelu aplikaciju.

Za promjenu jezika potrebno je u odvojenoj datoteci postaviti „LanguageProvider“ te unutar App.js sav sadržaj staviti unutar te komponente. Na taj će način promjena jezika biti dostupna u cijeloj aplikaciji.

```
import { createContext, useContext, useState } from "react";

const LanguageContext = createContext();

export const LanguageProvider = ({ children }) => {
  const [language, setLanguage] = useState("en");

  const toggleLanguage = () => {
```

```
    setLanguage((prevLanguage) => (prevLanguage === "en" ? "hr" :
"en"));
  };
  const contextValue = { language, toggleLanguage };

  return (
    <LanguageContext.Provider value={contextValue}>
      {children}
    </LanguageContext.Provider>
  );
};

export const useLanguage = () => {
  return useContext(LanguageContext);
};
```

Još su potrebni en.json i hr.json koji sadrže ime varijable i vrijednosti za pojedini jezik. Za varijablu homeTitle i settings izgledaju kao navedeno:

```
//en.json
{
  "homeTitle": "Current weather",
  "settings": "Settings"
}
//hr.json
{
  "homeTitle": "Trenutno vrijeme",
  "settings": "Postavke"
}
```

Gumb za promjenu jezika:

```
import { Button } from "react-native";
import hrTranslations from "../../locales/hr";
import enTranslations from "../../locales/en";
import { useLanguage } from "./LanguageContext";

export default function LanguageSwitcher() {
  const { language, toggleLanguage } = useLanguage();
  const t = language === "hr" ? hrTranslations : enTranslations;

  return <Button title="Toggle Language" onPress={toggleLanguage} />;
}
```

Za svaku komponentu u kojoj se koristi Context za promjenu jezika potreban je sljedeći kod na početku komponente:

```
import { useLanguage } from "../components/api/LanguageContext";
import hrTranslations from "../locales/hr.json";
import enTranslations from "../locales/en.json";
```

a unutar komponente:

```
const { language, toggleLanguage } = useLanguage();
const t = language === "hr" ? hrTranslations : enTranslations;
```

Umjesto teksta „Settings“ upisuje se „{t.settings}“:

```
<Text style={styles.title}>{t.settings}</Text>
```

4.2.4. Tema i font prema postavkama sustava

Za automatsko mijenjanje teme ovisno o postavkama uređaja koristi se useColorScheme. Za prilagodbu fonta ovisno o veličini i postavkama uređaja, potrebno je koristiti PixelRatio. Ta je opcija dostupna samo za Android, dok će kod iOS uređaja uvijek postaviti default vrijednost.

```
import { useColorScheme, PixelRatio } from "react-native";
const fontSize = PixelRatio.getFontScale() * 16;
```

Unutar komponente postavljaju se i pozivaju konstante:

```
const colorScheme = useColorScheme();

const styleContainer = {
  backgroundColor: colorScheme === "dark" ? "black" : "white",
};
const styleText = {
  color: colorScheme === "dark" ? "white" : "black",
};
return (
  <View style={ styleContainer }>
    <Text style={[ styleText, fontSize]}>Ovo je tekst.</Text>
  </View>
);
```

Kao i font i tema, također se može prepoznati i jezik koji korisnik koristi. Za to je potrebna posebna biblioteka i18next. [12]

Potrebno je instalirati:

```
npm install i18next react-i18next
```

Unutar glavne datoteke App.js potrebno je definirati sve moguće jezike i default jezik, ako neki od navedenih nije dostupan. Sav sadržaj je potrebno zatvoriti unutar I18nextProvider komponente:

```
import { I18nextProvider } from 'react-i18next';
import i18n from 'i18next';
```

```
import LanguageDetector from 'i18next-browser-languagedetector';

import enTranslation from './locales/en.json';
import hrTranslation from './locales/hr.json';

i18n
  .use(LanguageDetector)
  .init({
    resources: {
      en: {
        translation: enTranslation,
      },
      hr: {
        translation: hrTranslation,
      },
    },
    fallbackLng: 'en',
  });

export default function App () {
  return (
    <I18nextProvider i18n={i18n}>
      {children}
    </I18nextProvider>
  );
};
```

4.2.5. Popularni pluginovi

React ima veliku zajednicu pa je podrška i dokumentacija razvijena. Aktivna zajednica sudjeluje u izradi dodatnih biblioteka. React Native Directory na GitHubu sadrži oko 1260 biblioteka koje podupiru izradu aplikacija. Jedna od njih je i već spomenuta biblioteka i18next. [13]

Neke od najpoznatijih i najviše korištenih:

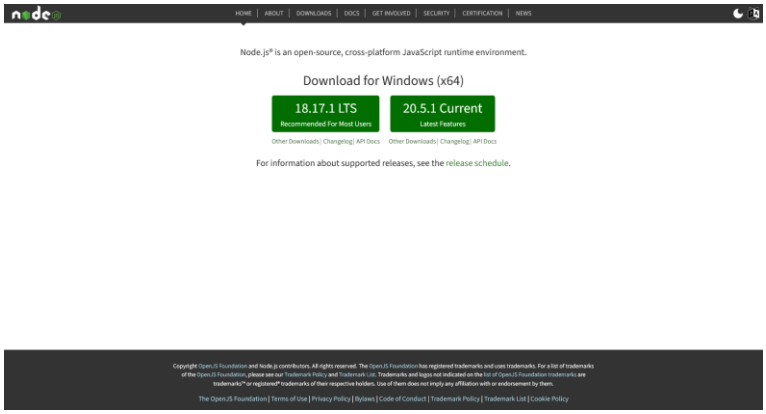
- react-native-reanimated https://docs.swmansion.com/react-native-reanimated/
- react-navigation https://reactnavigation.org/
- react-native-screens https://github.com/software-mansion/react-native-screens
- react-native-maps https://github.com/react-native-maps/react-native-maps
- react-native-tab-view https://github.com/satya164/react-native-tab-view
- react-native-gesture-handler https://github.com/software-mansion/react-native-gesture-handler

5. React Native - potrebni alati

Za početak nužno je instalirati Node.js, Visual Studio Code, Expo Go na mobitelu, Android Studio ili Xcode na računalu. React Native ima svoju službenu stranicu (https://reactnative.dev/) gdje se nalazi sva dokumentacija i postupci izrade React Native projekta. Za dohvaćanje podataka o prognozi, koristi se Open Weather API (https://openweathermap.org/).

5.1. Node.js

Kako bi se uopće mogao koristiti React potrebno je instalirati Node.js.



Slika 5.1 Instaliranje Node.js

5.2. Visual Studio Code

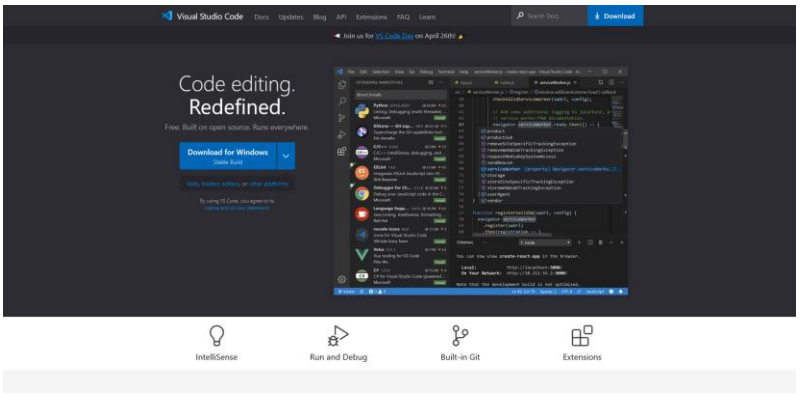
Visual Studio Code je jedan od najpopularnijih integriranih razvojnih okruženja (engl. *IDE - Integrated development environment*). Ima mogućnost instalacije raznih ekstenzija.

U ovom slučaju potrebne ekstenzije su:

- React Native Tools
- React-Native/React/Redux snippets

Dodatne ekstenzije za formatiranje koda:

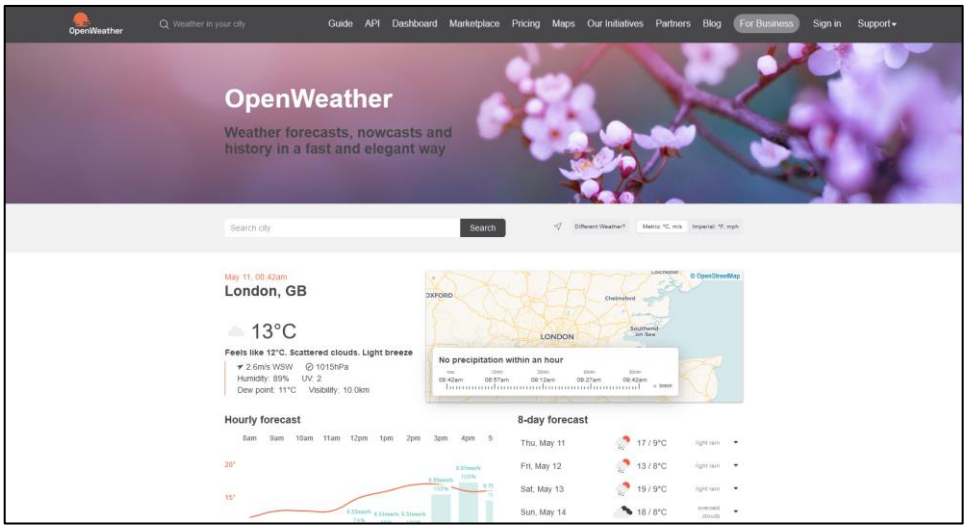
- Prettier - Code formatter
- Material Icon Theme
- Color Highlight



Slika 5.2 Instaliranje code editora Visual Studio Code

5.3. Open Weather API

API (engl. *Application Programming Interface*) omogućuje komunikaciju aplikacije s bazom podataka pomoću određenih funkcija. Korisnik korištenjem aplikacije zapravo šalje zahtjev za dobivanje podataka s drugog računala. [14]



Slika 5.3 Open Weather API

Open Weather API je besplatan API koji sadrži trenutne, ali i povijesne podatke o vremenskoj prognozi. Za početak potrebno se registrirati, nakon čega se dobije personalizirani API ključ. On se koristi za „fetching“, odnosno dohvaćanje podataka. Kao student moguće je koristiti „Developer“ plan koji uključuje dohvat podataka za više dana.

Iz toga slijedi da je konstanta s vlastitim ključem i link za dohvaćanje podataka:

```
const openWeatherKey = "abcdefghijklmnoprstuv";
```

```
let url =
`https://api.openweathermap.org/data/2.5/weather?q=London,uk&APPID=${
openWeatherKey}`
```

odnosno s unesenim ključem:

```
https://api.openweathermap.org/data/2.5/weather?q=London,uk&APPID=
abcdefghijklmnoprstuv
```

Kod postavljanja projekta, potrebno je instalirati i expo-location:

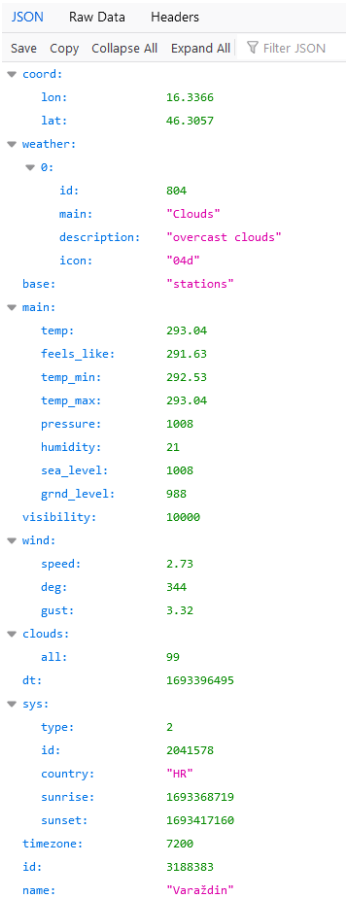
```
npx expo install expo-location
```

kojim se kasnije pristupa lokaciji korisnika za precizno određivanje mjesta i vremena.

Za dohvat lokacije, umjesto navedenog url-a, koristi se početni dio na koji se dodaje lokacija:

```
https://api.openweathermap.org/data/2.5/weather?
&lat=${location.coords.latitude}&lon=${location.coords.longitude}&APP
ID=${openWeatherKey}
```

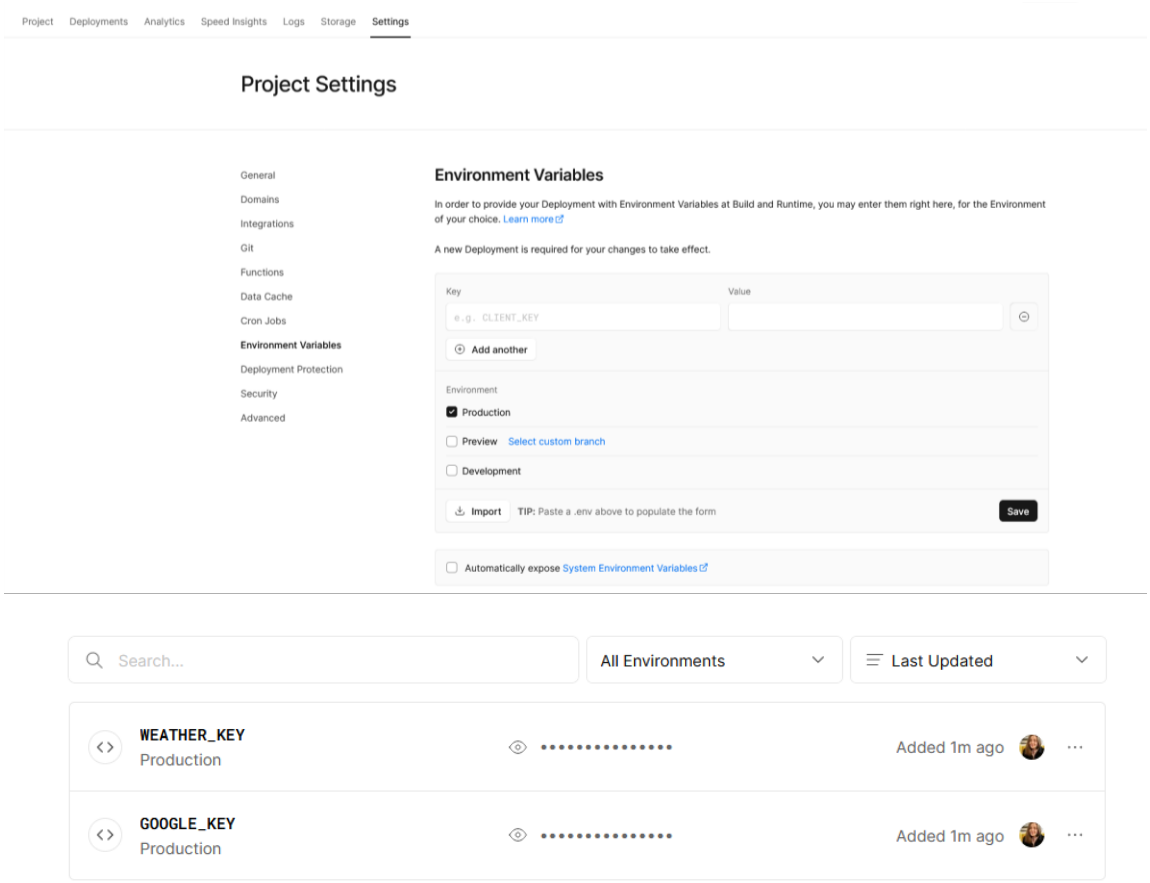
Iz navedenog dobije se JSON ovisno o trenutnoj lokaciji.



Slika 5.4 JSON podaci

5.3.1. Zaštita API ključeva

Navedeni API ključ „openWeatherKey“ je izmišljen zbog zaštite podataka autorice ovoga rada. U projektu je poželjno ključeve odvojiti u .env datoteku iz koje se oni kasnije pozivaju. Kod postavljanja na GitHub .env datoteku je potrebno uključiti u .gitignore. Vercel za svaku novu promjenu projekta uzima kod s GitHub-a. Pošto je .env datoteka spremljena samo lokalno, potrebno je u Vercel projekt upisati ili ubaciti .env datoteku s ključevima. Na taj se način ključevi ne izlažu direktno, već su definirani lokalno i unutar Vercel-a. [15]



Slika 5.5 Dodavanje API ključeva u Vercel

Env datoteka sastoji se od naziva i vrijednosti, piše se bez razmaka i navodnika.

```
WEATHER_KEY=abcdefghijklmnoprstuv
```

Unutar datoteka gdje se ključevi koriste poziva se:

```
import { WEATHER_KEY } from "@env";
const openWeatherKey = WEATHER_KEY;
```

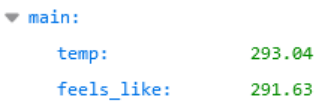
5.3.2. Pretvaranje jedinica

Neke podatke iz JSON-a ne može se direktno ispisati, već je potrebno pretvoriti u određenu jedinicu. Najjednostavniji primjer je pretvaranje Kelvina u Celzijuse:

```
<Text style={styles.currentTemp}>
{Math.round((forecast.main.temp - 272.15) * 10) / 10}&deg;C
</Text>
```

Temperatura se iz 293.88K pretvara u 21.7°C. Prva varijabla „forecast“ dohvaća JSON podatke iz API-ja. Nastavno na nju upisuju se željeni atributi. Oni se mogu odvajati točkom ili se upisuju unutar uglatih zagrada i navodnika, rezultat je u oba slučaja isti [16]:

```
forecast.main.temp
forecast[„main“].temp
```



Slika 5.6 Dohvat podatka „temp“

Drugi način koristi se kad je atribut broj:

```
forecast.weather[0].description
```

Slika 5.7 Dohvat podatka „description“

5.3.2.1. Pretvaranje Unix jedinice

Podaci o vremenu izlaska i zalaska sunca te datum mjerenja izraženi su u unix jedinici. To je jedinica koja broji koliko je sekundi prošlo od 01.01.1970., stoga je za pretvaranje vremena izlaska sunca „1690342108“ potrebno:

```
const sunrise = forecast.sys.sunrise;
const timeSunrise = new Date(sunrise * 1000);
```

Prvo se iz API-ja dohvaća vrijednost sunrise koja je 1690342108, zatim se postavlja novi datum prema tom podatku i množi s 1000 kako bi se pretvorio u milisekunde. Za ispis datuma i vremena dovoljan je samo console.log(timeSunrise); te će se ispisati:

```
Wed Jul 26 2023 05:28:28 GMT+0200 (Central European Summer Time)
```

Ipak, za željeni format vremena izlaska sunca 05:28, potreban je kod:

```
const sunrise = forecast.sys.sunrise;
const timeSunrise = new Date(sunrise * 1000);
const hoursSunrise = timeSunrise.getHours();
const minutesSunrise = timeSunrise.getMinutes();

const formattedHoursSunrise =
    hoursSunrise < 10 ? `0${hoursSunrise}` : hoursSunrise;
const formattedMinutesSunrise =
  minutesSunrise < 10 ? `0${minutesSunrise}` : minutesSunrise;
```

Iz timeSunrise uzimaju se samo sati i minute. Formatiranjem se satima i minutama manjima od 10 na prvo mjesto dodaje 0 kako bi uvijek bili dvoznamenkasti. U kodu se vrijeme ispisuje:

```
<Text style={styles.text}>
    {`${formattedHoursSunrise}:${formattedMinutesSunrise}`}
</Text>
```

5.3.2.2. Pretvaranje stupnjeva u smjer vjetra

Drugo pretvaranje je iz stupnjeva u strane svijeta za prikaz smjera vjetra. Sjever se može iskazati u vrijednostima 0 i 360 stupnjeva, iz tog razloga potrebno je dva puta koristiti vrijednost „N“ kao „North“ tj. Sjever. To znači da je umjesto 16 strana svijeta, potrebno ispisati 17 strana od kojih su prva i zadnja „N“. [17]

Tablica 2 Vrijednosti strana svijeta

Values	Compass Sectors
1	N
2	NNE
3	NE
4	ENE
5	E
6	ESE
7	SE
8	SSE
9	S
10	SSW
11	SW
12	WSW
13	W
14	WNW
15	NW
16	NNW
17	N

što u kodu rezultira nizom:

```
const compassSector = ["N", "NNE", "NE", "ENE", "E", "ESE", "SE", "SSE",
"S", "SSW", "SW", "WSW", "W", "WNW", "NW", "NNW", "N",];
```

ili hrvatski:

```
const compassSector = ["S", "SSI", "I", "IIJ", "I", "IJJ", "J",
"JJI", "S", "SJZ", "Z", "ZZJ", "Z", "ZZI", "I", "IIS", "S",];
```

a računa se:

```
windDirection = compassSector[(forecast.wind.deg / 22.5).toFixed(0)]
```

Dijeljenje s 22.5 dijeli 360 stupnjeva na 16 dijelova, a .toFixed(0) zaokružuje rezultat na najbliži cijeli broj. Taj broj određuje koji će se element iz niza prikazati kao smjer.

5.3.3. Ikone za opis vremena

Ikone koje dodatno opisuju vremensku prognozu dohvaćaju se linkom:

```
http://openweathermap.org/img/wn/${current.icon}@4x.png
```

Tablica 3 Moguće ikone ovisno o vremenu

Izvor: https://openweathermap.org/weather-conditions

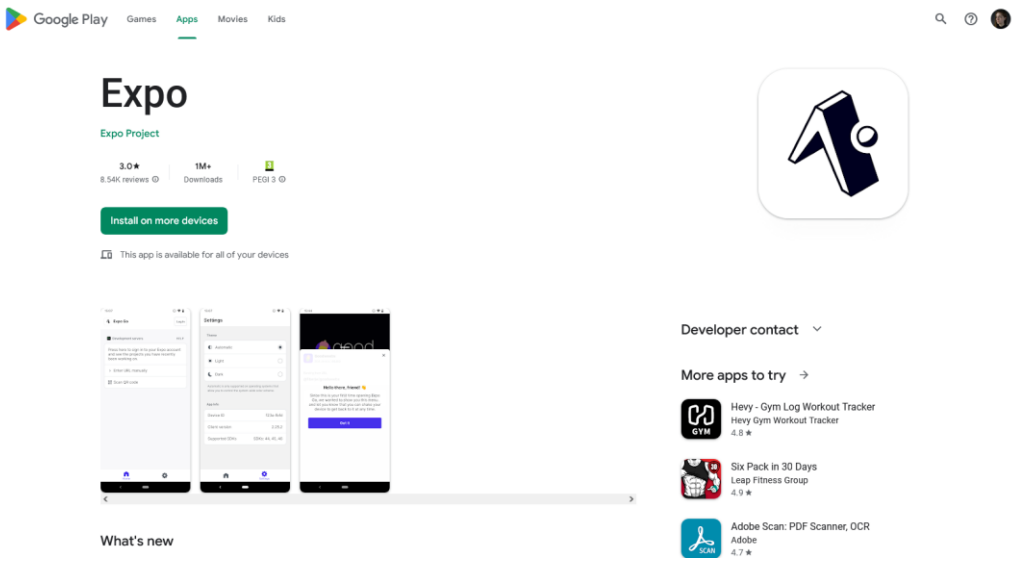
Day icon	Night icon	Description
01d.png	01n.png	clear sky
02d.png	02n.png	few clouds
03d.png	03n.png	scattered clouds
04d.png	04n.png	broken clouds
09d.png	09n.png	shower rain
10d.png	10n.png	rain
11d.png	11n.png	thunderstorm
13d.png	13n.png	snow
50d.png	50n.png	mist

6. Praktični dio

Praktični dio uključuje postavljanje projekta kao što je opisano u poglavlju 4.1., zatim pokretanje aplikacije, dodavanje navigacije i ikona, rješavanje bugova i oblikovanje stilova.

6.1. Pokretanje aplikacije

Više je mogućnosti za pokretanje aplikacije. Moguće je koristiti emulator unutar „Android Studio“ ili instalirati „Expo Go“ aplikaciju na mobitel te pokrenuti direktno na mobitelu.



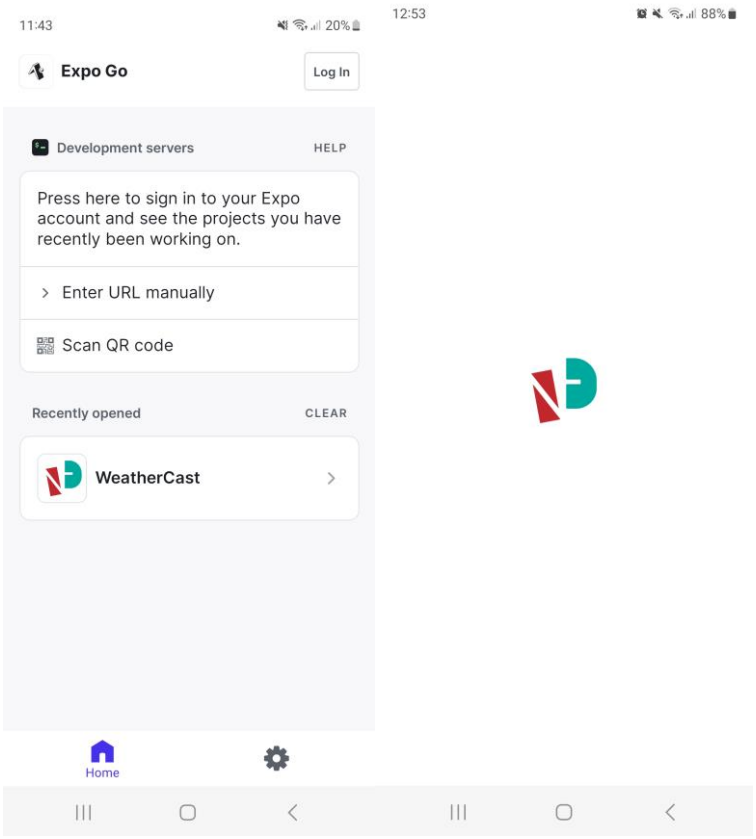
Slika 6.1 Mobilna aplikacija Expo

Na slici Slika 6.2 vidljivo je korisničko sučelje aplikacije Expo Go. Na početnom zaslonu nalaze se skener QR koda za pokretanje nove aplikacije i mjesto za unos URL-a. Sljedeće je „Recently opened“ gdje se nalazi React Native projekt WeatherCast s odgovarajućom ikonom.

Aplikacija se pokreće u VS Code naredbom:

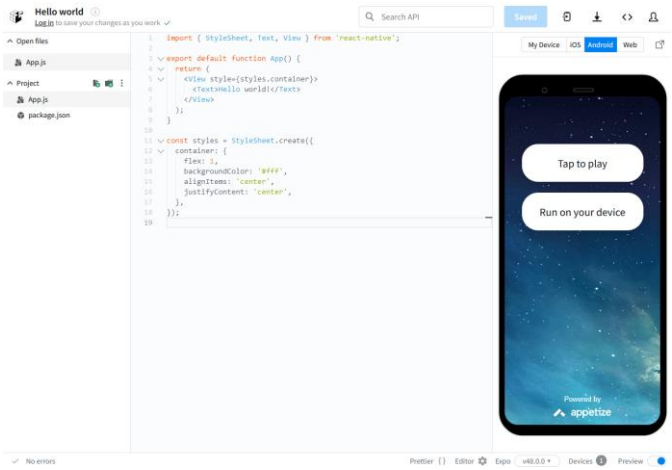
```
npx expo start --web
```

Za otvaranje na mobitelu potrebno je skenirati dobiveni QR kod u konzoli ili kliknuti na WeatherCast, a za otvaranje na računalu koristi se link: http://localhost:19006/.



Slika 6.2 Expo Go aplikacija i ulazak u aplikaciju

Još jedna opcija je korištenje „Snack“ aplikacije u pregledniku koja sadrži emulatore za iOS, Android i web. Sav kod se upisuje na webu pa se potrebno prijaviti kako bi se taj kod spremao.



Slika 6.3 Web aplikacija Snack

Za razliku od React.js koji koristi ReactDOM za renderiranje:

```
ReactDOM.render(<App />, container)
```

react-native je sam po sebi renderer:

```
const { AppRegistry } = require('react-native')
AppRegistry.registerComponent('app', () => MainComponent)
```

Dok React.js koristi alate poput Webpack i Babel, React Native koristi Metro Bundler kao alat za pakiranje, minimiziranje i optimiziranje JS koda za mobilne platforme. Metro stvara posebne JS pakete za pojedinu platformu.

6.2. Navigacija

Za razliku od React.js koji koristi react-router, React Native koristi react-navigation.

```
npm install @react-navigation/native
npm install @react-navigation/stack

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

Instaliraju se potrebne biblioteke, a u App.js doda se NavigationContainer komponenta unutar koje se stavlja ostatak koda.

```
<NavigationContainer>
    <Stack.Navigator initialRouteName="Home">
      <Stack.Group
        screenOptions={{
          headerShown: false,
        }}
      >
        <Stack.Screen name="Home" component={Home} />
        <Stack.Screen name="Pollution" component={Pollution} />
        <Stack.Screen name="Map" component={Map} />
        <Stack.Screen name="Settings" component={Settings} />
      </Stack.Group>
    </Stack.Navigator>
</NavigationContainer>
```

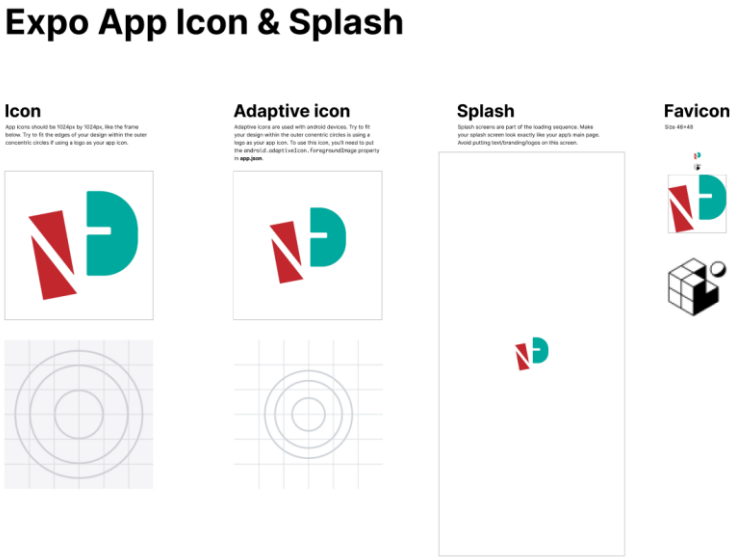
U ovom primjeru navigacija sadrži 4 ekrana, od kojih je Home početni ekran. Nalaze se unutar Stack.Group jer se na sva 4 ekrana sakriva header. [18]

Postoje dvije vrste navigacije u mobilnim aplikacijama. Donja navigacija i stack navigacija. Donja navigacija za primjer može sadržavati ikone kuće, čovjeka i srca, što označava početnu stranicu, korisnički profil i stvari koje nam se sviđaju. „Stack“ navigacija odnosi se na navigaciju

koja se klikom na neku od komponenta otvara preko te stranice, a za povratak je ponuđena strelica. Prijevod na hrvatski je gomilanje, a takav je i princip rada.

6.3. Oblikovanje (stilovi)

Unutar mape assets nalazi se favicon koji se pojavljuje na webu i ikone koje se pojavljuju kod pokretanja aplikacije u Expo. Korišten je logo koji predstavlja ime i prezime Nela Đuranec izrađen u sklopu kolegija Grafički dizajn. Dimenzije datoteka icon.png i adaptive-icon.png su 1024x1024px, splash.png 1284x2778px, dok je favicon 48x48px što je jako sitno pa je dodatno prikazano i u dimenzijama 400x400px za bolju vidljivost. Unutar Figme stavljeni su obrubi kako bi veličina ikona bila vidljivija, no kod izvoza (engl. *export*) potrebno ih je maknuti. Ispod ikona prikazan je i originalni izgled ikona kod postavljanja aplikacije.



Slika 6.4 Figma dizajn ikona

U React Native ne koristi se CSS, već svojstva ili props (engl. *properties*) koji se uređuju unutar StyleSheet komponenti.

```
import { StyleSheet } from "react-native";

export const globalStyles = StyleSheet.create({
  loading: {
    justifyContent: "center",
    alignItems: "center",
    height: "100%",
  },
});
```

Stilovi se mogu pisati unutar datoteke u kojoj se koriste, no bolje je pisati ih u odvojenoj datoteci.

```
import { StyleSheet, View } from "react-native";

export default function App(){
  return <View style={styles.container}>Container</View>
}
const styles = StyleSheet.create({
  container: {
    width: "100%",
    backgroundColor: "black",
  },
})
```

Slika 6.5 Smještaj stilova unutar datoteke u kojoj se koristi

Datoteka GlobalStyles sadrži dva stila: globalStyles koji se koristi u više datoteka i layout u kojem je definiran raspored, poput flexRow i flexColumn.

```
import { StyleSheet } from "react-native";

export const globalStyles = StyleSheet.create({
  loading: {
    justifyContent: "center",
    alignItems: "center",
    height: "100%",
  },
  text: {
    fontWeight: "600",
    fontSize: 20,
  },
  textWhite: {
    color: "#f7f7f7",
    fontWeight: 600,
  },
  bold: {
    fontWeight: 600,
  },
...
});
export const layout = StyleSheet.create({
  container: {
    userSelect: "none",
    width: "100%",
    backgroundColor: "rgba(0, 0, 0, .2)",
  },
  flexRow: {
    flexDirection: "row",
    alignItems: "center",
```

```
    justifyContent: "center",
  },
  flexRowTop: {
    alignItems: "flex-start",
  },
  flexColumn: {
    alignItems: "center",
    justifyContent: "center",
  },
  flexCenter: {
    alignItems: "center",
    justifyContent: "center",
    height: "100%",
    width: "100%",
  },
});
```

6.4. Pozadinski gifovi

Za provjeru dan ili noć koristi se isDark. Na osnovu trenutnog vremena određuje je li dan ili noć i prema tome postavlja pozadinski gif. Na početku se odrede varijable koje pozivaju gifove:

```
const bgImgDark = require("../assets/night.gif");
const bgImgDarkRain = require("../assets/rain.gif");
const bgImgDarkThunder = require("../assets/thunder.gif");
const bgImgLight = require("../assets/day.gif");
const bgImgLightRain = require("../assets/rainDay.gif");
const bgImgLightThunder = require("../assets/thunderDay.gif");
const bgImgSun = require("../assets/sun.gif");
const bgImgSnow = require("../assets/snow.gif");
```

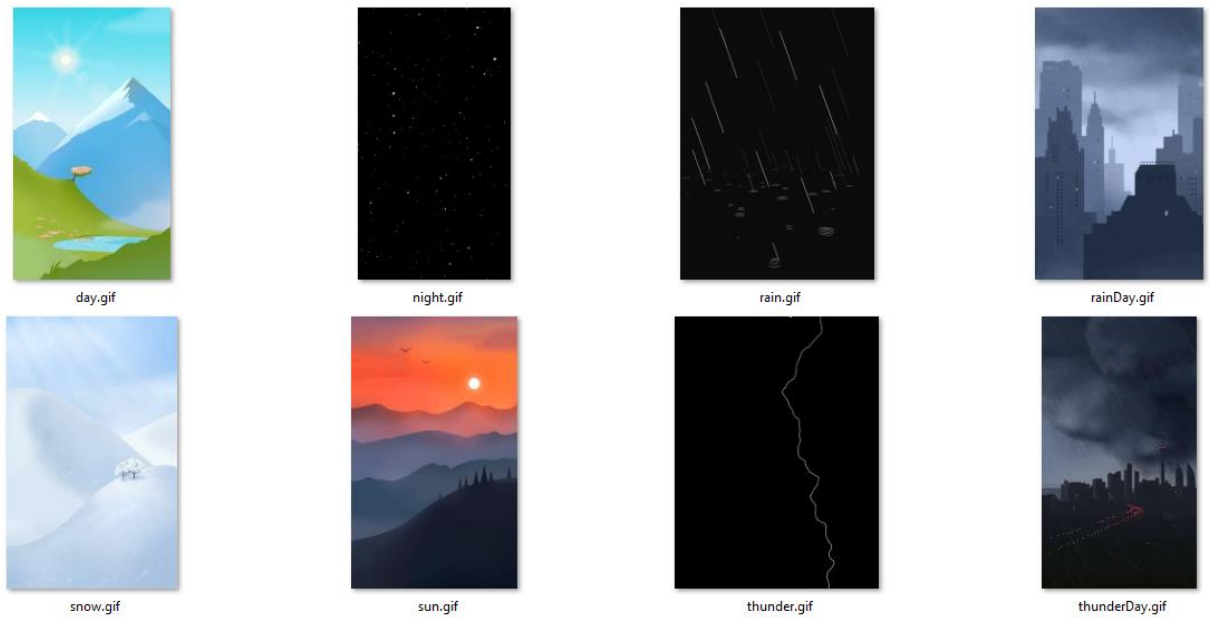
Zatim se postavi useState koji sprema vrijednost isDark ovisno o vremenu:

```
const [isDark, setIsDark] = useState(Boolean);

const hoursForTheme = new Date().getHours();
function checkTheme() {
  if (hoursForTheme >= 20 || hoursForTheme <= 6) {
    setIsDark(true);
  } else {
    setIsDark(false);
  }
}

function changeBg() {
    let id = current.id.toString();
    const group = id[0];
    if (
      (5 <= hoursForTheme && hoursForTheme <= 6) ||
```

```
      (18 <= hoursForTheme && hoursForTheme <= 19)
    ) {
      return bgImgSun;
    } else {
      switch (group) {
        case "2":
          return isDark ? bgImgDarkThunder : bgImgLightThunder;
        case "3":
        case "5":
          return isDark ? bgImgDarkRain : bgImgLightRain;
        case "6":
          return bgImgSnow;
        default:
          return isDark ? bgImgDark : bgImgLight;
      }
    }
  }
```

Funkcija changeBg() poziva se unutar ImageBackground native komponente:

```
<ImageBackground source={changeBg()} style={globalStyles.bgImg}>
...ostatak koda
</ImageBackground>
```



Slika 6.6 Korišteni gifovi

Izvor: https://www.behance.net/gallery/54827733/Weather-animation

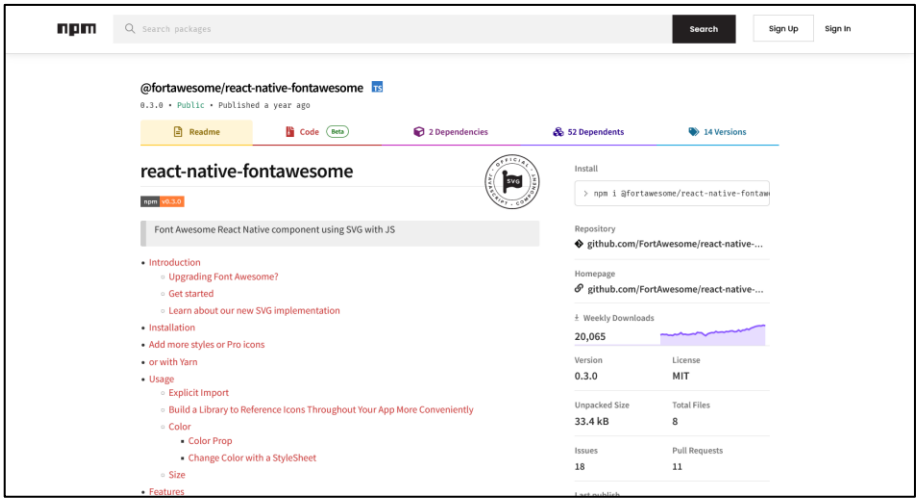
https://www.pinterest.com/pin/197595502383768840/

https://www.pinterest.com/pin/719309371736446879/

6.5. Ikone

Za korištenje Font Awesome ikona potrebno je instalirati biblioteku s besplatnim ikonama [19]:

```
npm i @fortawesome/react-native-fontawesome
npm i --save @fortawesome/free-solid-svg-icons
npm i --save @fortawesome/fontawesome-svg-core
```



Slika 6.7 npm stranica s uputama za korištenje Font Awesome

Na vrhu dokumenta u kojem se koristi ikona potrebno je navesti:

```
import { FontAwesomeIcon } from "@fortawesome/react-native-
fontawesome";
import {
  faCompass,
  faDroplet,
  faSun,
  faWind,
} from "@fortawesome/free-solid-svg-icons";
```

te pozivati ikone:

```
<FontAwesomeIcon icon={faTemperature4} />
<FontAwesomeIcon icon={faCompass} />
```

6.6. Fontovi

Za dodavanje fontova je potreban još jedan context, FontContext.js. Unutar njega stvore se varijable koja pozivaju fontove iz mape koji se zatim mogu koristiti u svim datotekama. App.js zatvori se u FontProvider koji omogućuje korištenje određenih fontova u cijeloj aplikaciji. Prije svega navedenog potrebno je instalirati dodatnu biblioteku:

```
npx expo install expo-font
```

Fontovi su definirani na jednom mjestu pa se promjenom na jednom mjestu vrlo lako zamijene fontovi u cijeloj aplikaciji. Definirana su dva fonta, Lato Regular i Lato Bold:

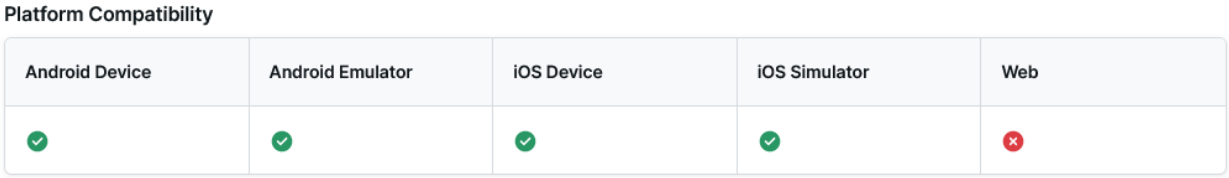
```
useEffect(() => {
    const loadFonts = async () => {
      await Font.loadAsync({
        Font: require("../../assets/fonts/Lato/Lato-Regular.ttf"),
        FontBold: require("../../assets/fonts/Lato/Lato-Bold.ttf"),
      });
      setFontsLoaded(true);
    };

    loadFonts();
  }, []);
```

a pozivaju se:

```
  text: {
    fontFamily: "FontBold",
    fontSize: 20,
    lineHeight: 20,
    color: "#222",
  },
  left: {
    position: "absolute",
    left: 20,
    top: 10,
    fontFamily: "Font",
  },
```

6.7. Google mapa

Dodavanje mape trenutno je moguće samo na mobilnom uređaju jer biblioteka nije kompatibilna za web. To je jedan od problema React Native-a koji se još uvijek rješava. Iz tog razloga mapa je trenutno dostupna samo na mobilnoj verziji aplikacije. Sama prisutnost react-native-maps biblioteke dovodi do greške i nemogućnosti korištenja web verzije.



Slika 6.8 Nedostupnost MapVIew komponente u web pregledniku

Zato je potrebna postinstall.js datoteka koja u web verziji zanemaruje react-native-maps biblioteku. Uključuje se unutar package.json, uz ostale skripte:

```
 "scripts": {
    "postinstall": "node postinstall.js"
  },
```

postinstall.js:

```
async function runPostInstall() {
  const fs = require("fs").promises;

  function log(...args) {
    console.log(...args);
  }

  log(
    "Creating web compatibility of react-native-maps using an empty
module loaded on web builds"
  );
  const modulePath = "node_modules/react-native-maps";

  await fs.writeFile(
    `${modulePath}/lib/index.web.js`,
    "module.exports = {}",
    "utf-8"
  );
  await fs.copyFile(
    `${modulePath}/lib/index.d.ts`,
    `${modulePath}/lib/index.web.d.ts`
  );

  const pkg = JSON.parse(await
fs.readFile(`${modulePath}/package.json`));
  pkg["react-native"] = "lib/index.js";
  pkg["main"] = "lib/index.web.js";

  await fs.writeFile(
    `${modulePath}/package.json`,
    JSON.stringify(pkg, null, 2),
    "utf-8"
  );
  log("Script ran successfully");
}

runPostInstall().catch((error) => {
  console.error("An error occurred:", error);
  process.exit(1);
});
```

Nakon toga potrebno je MapView komponentu uključiti u mobilnoj verziji, a u web verziji ukloniti mogućnost otvaranja Maps ekrana ili postaviti drugačiji sadržaj:

```
  {Platform.OS === "web" ? (
        <>
          <NavigationWeb />
           <Text> Maps is currently unavailable on web version of
WeatherCast. For better experience, please use mobile version of
WeatherCast with Expo.
            </Text>
        </>
      ) : (
        <View style={styles.container}>
          <MapView style={styles.map} />
        </View>
      )}
```

Slika 6.9 Izgled ekrana Mapa na web verziji

6.8. Optimizacija

React je koristan jer je sastavljiv, odnosno, lako ga je rastaviti u više manjih komponenta. U komponentama se lakše snalaziti nego u jednoj datoteci koja sadrži apsolutno sav kod. Iz tog razloga loša praksa je imati puno linija koda kao na Slika 6.10:

```
564   infoBox: {
565     backgroundColor: "rgba(255, 255, 255, 0.8)",
566     width: 150,
567     borderRadius: 10,
568     margin: 10,
569     padding: 20,
570     paddingTop: 50,
571   },
572 });
573
```

Slika 6.10 VS Code - 572 linije koda

Home komponenta sadrži web i mobilnu verziju te stilove. Taj je kod potrebno razdvojiti na više manjih komponenta. Za početak dobra je praksa odvojiti stilove u posebnu datoteku (posebno ako se ponavljaju u više datoteka kao npr. container).

```
import { StyleSheet } from "react-native";

export const styles = StyleSheet.create({
  container: {
     userSelect: "none",
     width: "100%",
backgroundColor: "rgba(0, 0, 0, .2)",
  },   flexCenter: {
    alignItems: "center",
    justifyContent: "center",
    height: "100%",
    width: "100%",
  },
});
```

Ovime se kod smanjio za 100 linija. Ti se stilovi pozivaju ovisno gdje su potrebni:

```
import { styles } from "../components/HomeStyles";
```

Komponente HomeMobile i HomeWeb izdvojene su i svaka ima oko 140 linija koda, što znači da je broj linija Home komponente smanjen s 572 na ukupno otprilike 200 linija koda.

Također, stilovi se mogu dodatno optimizirati tako da se ne ponavljaju iste linije koda. Kod stvaranja tamne i svijetle teme aplikacije potrebno je imati dva stila s jednakim svojstvima, ali s različitom bojom teksta.

```
lightCurrentLocation: {
    color: "#fff",
    textAlign: "center",
    fontSize: 24,
    fontWeight: "600",
  },
darkCurrentLocation: {
    color: "#d7d7d7",
    textAlign: "center",
    fontSize: 24,
    fontWeight: "600",
},
```

Ako je tema tamna primjenjuje se tamni stil, a svijetli ako nije tamna. Navedeni stil poziva se:

```
style={isDark ? styles.darkCurrentLocation:
     styles.lightCurrentLocation}
```

Da bi se izbjeglo ponavljanje istih linija koda, potrebno je taj kod rastaviti na dva dijela: na osnovni stil koji se koristi u obje teme i na dodatan stil koji sadrži boju za tamnu temu:

```
currentDescription: {
    color: "#fff",
    textAlign: "center",
    fontSize: 24,
    fontWeight: "600",
},
darkCurrentLocation: {
    color: "#d7d7d7",
},
```

Kod korištenja više stilova, pozivaju se u nizu i potrebno ih je zatvoriti u uglate zagrade te odvojiti zarezom. Za svijetlu temu je dovoljan osnovni stil jer sadrži boju, dok je za tamnu temu potreban i osnovni i tamni stil.

```
style={isDark
              ? [styles.currentLocation, styles.darkCurrentLocation]
              :  styles.currentLocation
}
```

6.9. Responzivnost

Aplikacija u svim komponentama provjerava koristi li korisnik web preglednik ili aplikaciju na mobilnom uređaju. Iz tog razloga aplikacija na web pregledniku mobilnog uređaja nije responzivna. U tom su slučaju potrebne tri verzije „layouta“: web na računalu, web na mobilnom uređaju te aplikacija na mobilnom uređaju. U odvojenoj datoteci postavi se funkcija:

```
import { useWindowDimensions } from "react-native";

export function isWeb() {
  const windowDimensions = useWindowDimensions();
  return windowDimensions.width >= 875;
}
```

Funkcija provjerava je li širina ekrana veća od 875px, ako je, izvrši se prvi dio komponente. Ako je širina ekrana manja provjerava se je li uređaj Android ili iOS. Na kraju, ako ni taj uvjet nije ispunjen, prikazuje se layout za mobilni web preglednik. To je potrebno jer je web layout drukčiji, a aplikacija ne sadrži navigaciju za web preglednik. Stoga, treća verzija sadrži layout aplikacije, uz dodatak navigacije za web.

Primjer komponente Pollution s tri verzije:

```
{webLayout ? (
  <>
    <NavigationWeb />
    <View style={layout.flexRow}>
      <PollutionLayout airPollution={airPollution} t={t} />
    </View>
  </>
) : Platform.OS === "android" || Platform.OS === "ios" ? (
  <PollutionLayout airPollution={airPollution} t={t} />
) : (
  <>
    <NavigationWeb />
    <PollutionLayout airPollution={airPollution} t={t} />
  </>
)
}
```

Skraćeno je na dvije opcije. Web verzija s navigacijom i mobilna verzija s ili bez navigacije.

```
{webLayout ? (
  <>
    <NavigationWeb />
    <View style={layout.flexRow}>
      <PollutionLayout airPollution={airPollution} t={t} />
    </View>
  </>
) : (
  <>
    {Platform.OS === "web" && <NavigationWeb />}
    <PollutionLayout airPollution={airPollution} t={t} />
  </>
)
}
```

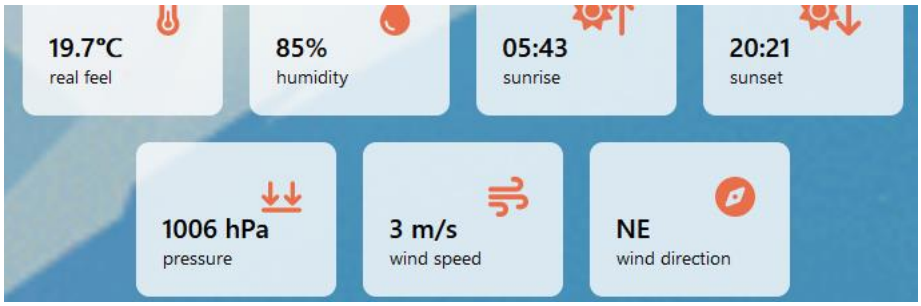
6.10. Testiranje

Slika 6.11 Podatak za ubrzanje vjetra nije izmjeren

Na Slika 6.11 prikazano je kako izgleda kad podatak iz API-ja nije zabilježen. Iz tog razloga potrebno je za svaki podatak prvo provjeriti je li taj podatak izmjeren i zabilježen u API. Konkretan primjer je podatak o ubrzanju vjetra. Potrebno je „View“ kod upisati unutar navedenog:

```
{forecast.wind?.gust && (
     <View style={styles.infoBox}>
            <FontAwesomeIcon
              icon={faArrowUpLong}
              style={[styles.icon, styles.arrow]}
            />
            <FontAwesomeIcon
              icon={faWind}
              style={[styles.icon, styles.iconLeft]}
            />
            <Text style={globalStyles.text}>
              {Math.round(forecast.wind.gust)} m/s
            </Text>
            <Text>wind gust</Text>
     </View>
)
}
```

Rezultat je da se taj podatak ne prikaže ako nije dostupna njegova vrijednost, što je vidljivo na slici Slika 6.12.



Slika 6.12 Izgled bez podatka koji nije dostupan

6.11. Objavljivanje aplikacije

Za šire korištenje aplikacije, potrebno ju je postaviti na Google Play i Apple Store. Za to je potreban „build“ aplikacije. Tim postupkom dobiva se build aplikacije koji se objavljuje na mobilne trgovine. Lakši način je korištenje Expo Go mogućnosti objavljivanja. Krajem 2023. godine dolazi nova verzija SDK koji više neće podržavati taj način, tj. naredbu expo publish. Zamjena će biti EAS Updates. [20] Kako bi se koristio EAS Updates potrebno je slijediti upute iz službene dokumentacije. [21]

6.11.1.Mobilni uređaj

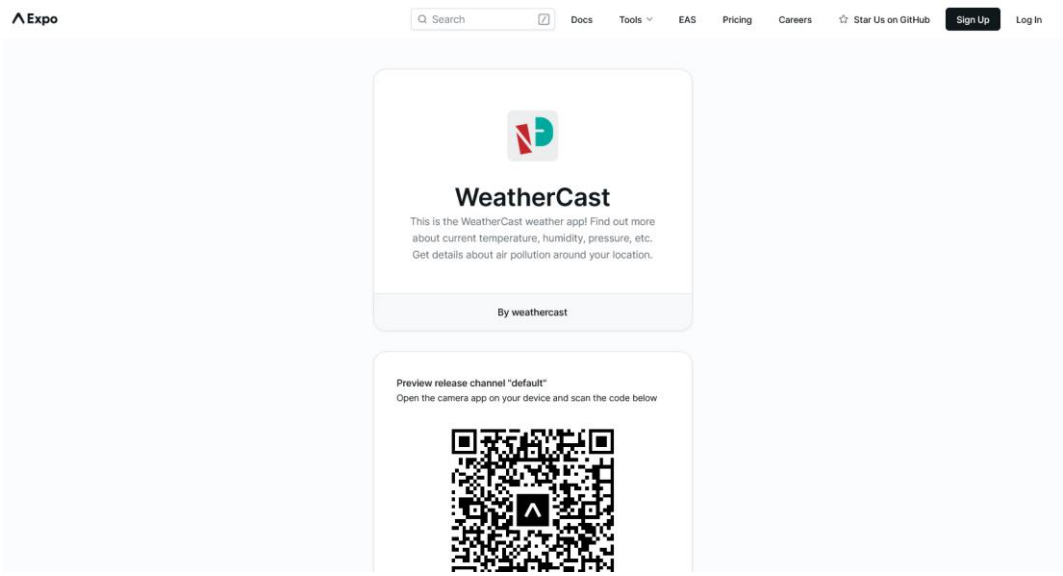
Postoje tri načina za objavljivanje React Native aplikacije za mobilni uređaj. Dva načina uključuju korištenje aplikacije Expo Go. Classic Updates - koristi se biblioteka expo-updates i naredba expo publish. Dolaskom nove verzije koristi se EAS Update tj. naredba eas update. Treća opcija se plaća, a to je build - dobiva se aplikacija za objavljivanje na mobilne trgovine.

6.11.1.1. Classic Updates (Expo publish)

Unutar konzole upiše se:

```
expo publish
```

Nakon instalacije dobije se link koji vodi na stranicu s QR kodom:



Slika 6.13 Stranica na koju vodi link iz konzole

Izvor: https://expo.dev/@lan-e/weather-cast?serviceType=classic&distribution

Prilikom svake nove promjene potrebno je ponovno objaviti aplikaciju, koristi se ista naredba:

```
expo publish
```

Android uređaji skeniranjem QR koda automatski otvore aplikaciju unutar Expo Go aplikacije, dok je za iOS uređaje potrebno stvoriti Expo račun. Također, unutar Expo aplikacije potrebno je sve iOS korisnike pozvati kao članove projekta kako bi mogli vidjeti aplikaciju.

Iako dolaskom nove verzije ova naredba više neće biti dostupna, aplikacije koje trenutno koriste tu naredbu i dalje će joj moći pristupiti.

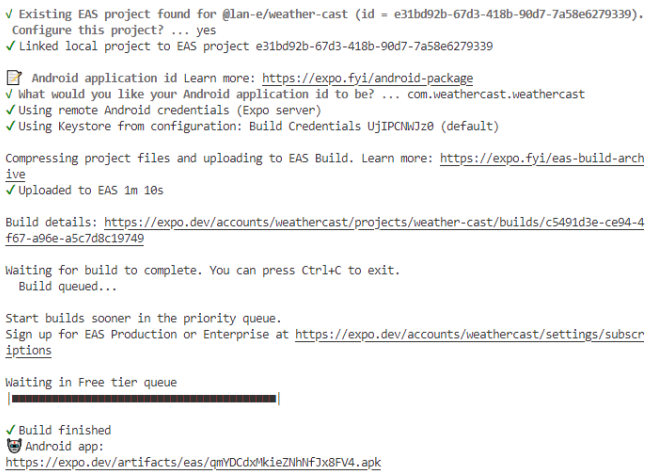
6.11.1.2. Build aplikacije

Expo Application Services (EAS) su usluge za Expo i React Native aplikacije. Služe za buildanje aplikacije. Za početak je potrebno instalirati eas-cli (EAS Command Line Interface):

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

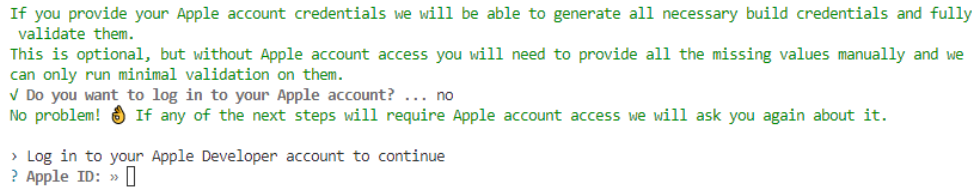
Zatim se pokreće build za Android i iOS:

```
eas build --platform android
```



Slika 6.14 Postupak buildanja Android aplikacije

```
eas build --platform ios
```



Slika 6.15 Postupak buildanja iOS aplikacije

Problem je što se iOS build ne može izvesti bez plaćenog Apple Developer profila, koji iznosi 99 dolara godišnje. Taj isti profil omogućuje i objavljivanje aplikacije na Apple App Store.

Moguće je izvesti build za Android, no potrebno je platiti objavljivanje na Google Play Store. Stoga je moguće korisnicima poslati aplikaciju preko Google Diska ili na sličan način, što nije poželjno za veću distribuciju. iOS je nemoguće buildati ili staviti na Apple Store bez plaćenog profila. Za manji projekt, bez dodatnih troškova, moguće je koristiti aplikaciju kroz Expo Go aplikaciju na mobilnom uređaju, ili koristiti web verziju.

6.11.1.3. EAS Update

Dolaskom nove verzije Expo SDK uvodi se obavezno korištenje EAS Update. Potrebno je instalirati već prije spomenuti eas-cli i prijaviti se u Expo profil:

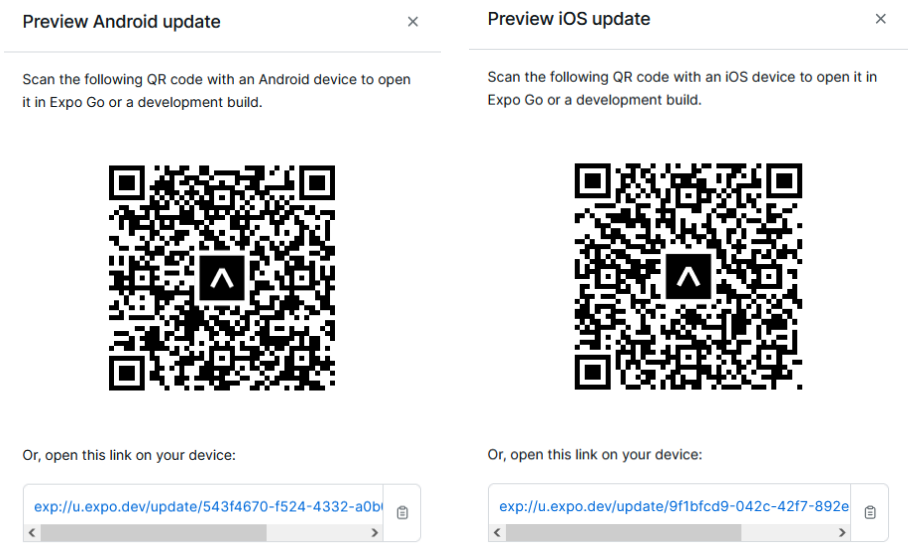
```
npm install -g eas-cli
eas login
```

Potrebna je i dodatna datoteka eas.json:

```
{"build": {
    "development": {
      "developmentClient": true,
      "distribution": "internal"
    },
    "preview": {
      "distribution": "internal",
      "channel": "preview"
    },
    "production": {
      "channel": "production"
    }
  }}
```

Pokrene se naredba eas update:configure, nakon navedenog pokreće se naredba eas update koja rezultira linkom za Expo profil koji sadrži dva QR koda, za Android i iOS:

```
eas update:configure
eas update --channel production --message "first update"
```

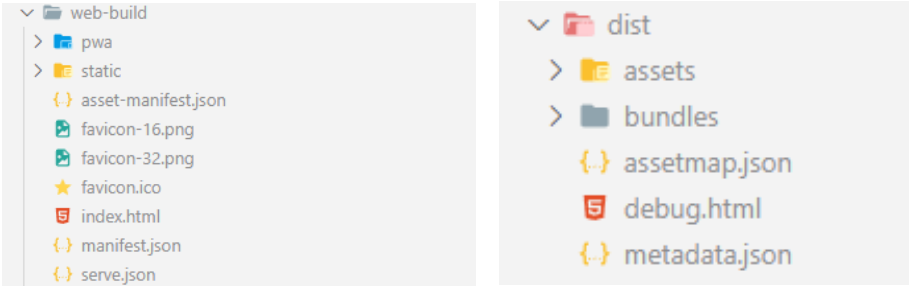


Slika 6.16 Dobiveni QR kodovi za testiranje aplikacije na Android i iOS

6.11.2. Web (Vercel)

Za objavljivanje web verzije, lokalno ili na Vercel, prvo je potrebno izvesti sve Javascript datoteke, slike i ikone - statičan paket (engl. *static bundle*). [22] Upisuje se naredba:

```
npx expo export:web
```



Slika 6.17 Mape web-build i dist nakon buildanja

Stvore se mape .vercel, web-build i dist. Potrebno je u projekt dodati vercel.json koji definira vercel konfiguraciju:

```
vercel.json {
  "buildCommand": "expo export:web",
  "outputDirectory": "web-build",
  "devCommand": "expo",
  "cleanUrls": true,
  "framework": null,
  "rewrites": [
    {
      "source": "/:path*",
      "destination": "/"
    }]}
```

Za lokalno pokretanje web aplikacije potrebno je upisati:

```
npx serve web-build --single
```

prilikom čega se web otvara na poveznici: http://localhost:3000.

Za objavljivanje na Vercel potrebno je instalirati vercel:

```
npm install -g vercel@latest
```

i pokrenuti naredbom:

```
vercel
```

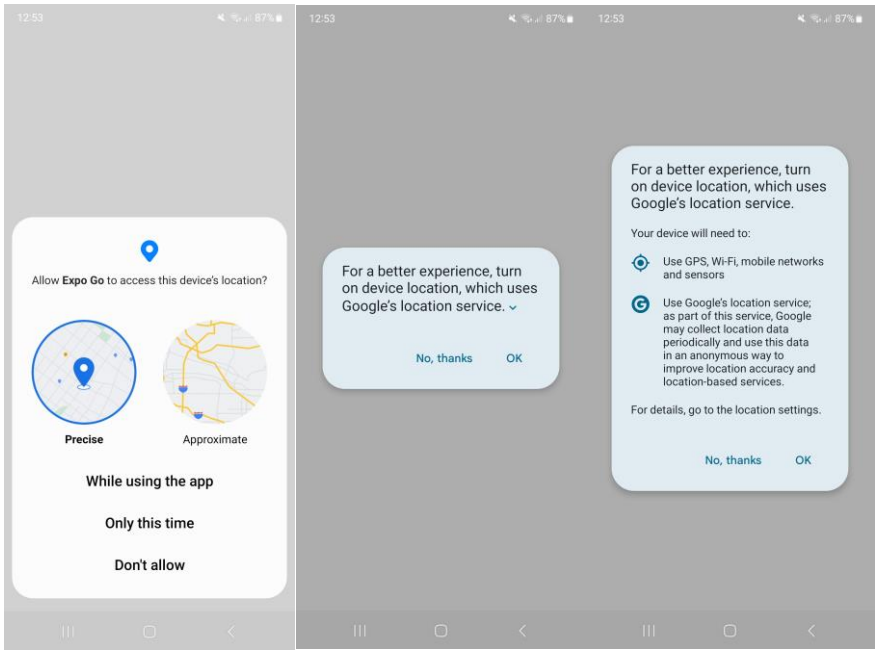
Ako je projekt spojen na GitHub i Vercel, dovoljno je pushati novu verziju projekta na GitHub.

7. Analiza rezultata

7.1. Pokretanje aplikacije

U web pregledniku, mobilnom ili na računalu, aplikacija se pokreće pomoću vercel linka https://final-weathercast.vercel.app/. Za pokretanje aplikacije na mobilnom uređaju potrebno je skenirati QR kod s linka https://expo.dev/@lan-e/weather-cast (zaslon Map u web verziji).

Kod učitavanja aplikacije vidljiv je „Splash screen“ sa Slika 6.2. Učitana aplikacija traži dozvolu za dohvat lokacije korisnika. Nakon dozvole za lokaciju, lokacija se automatski unosi u link API-ja i dohvaćaju se podaci o vremenu. Ako se ne dozvoli dohvat lokacije, lokacija je postavljena na grad Varaždin.

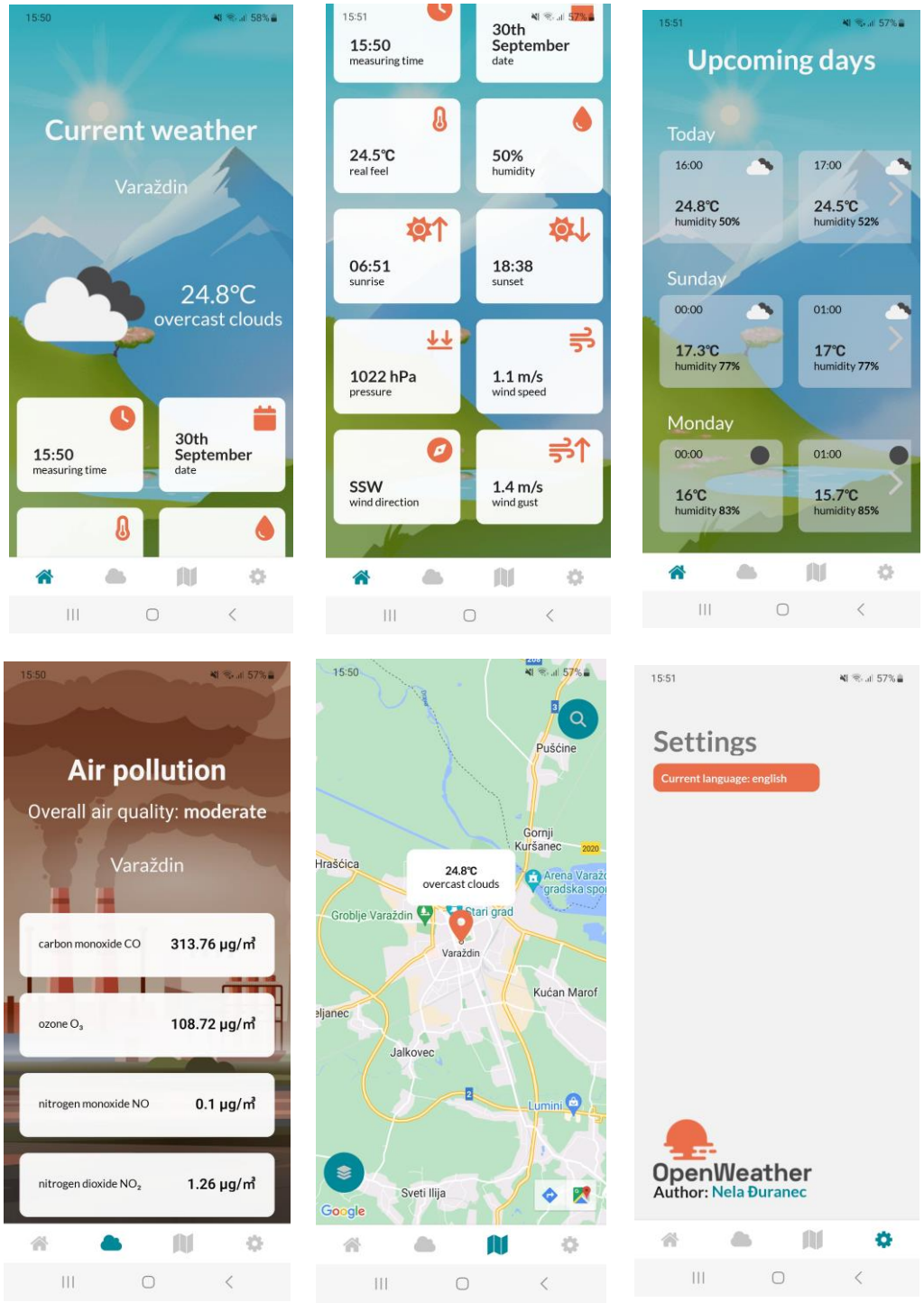


Slika 7.1 Dohvat lokacije na mobilnom uređaju

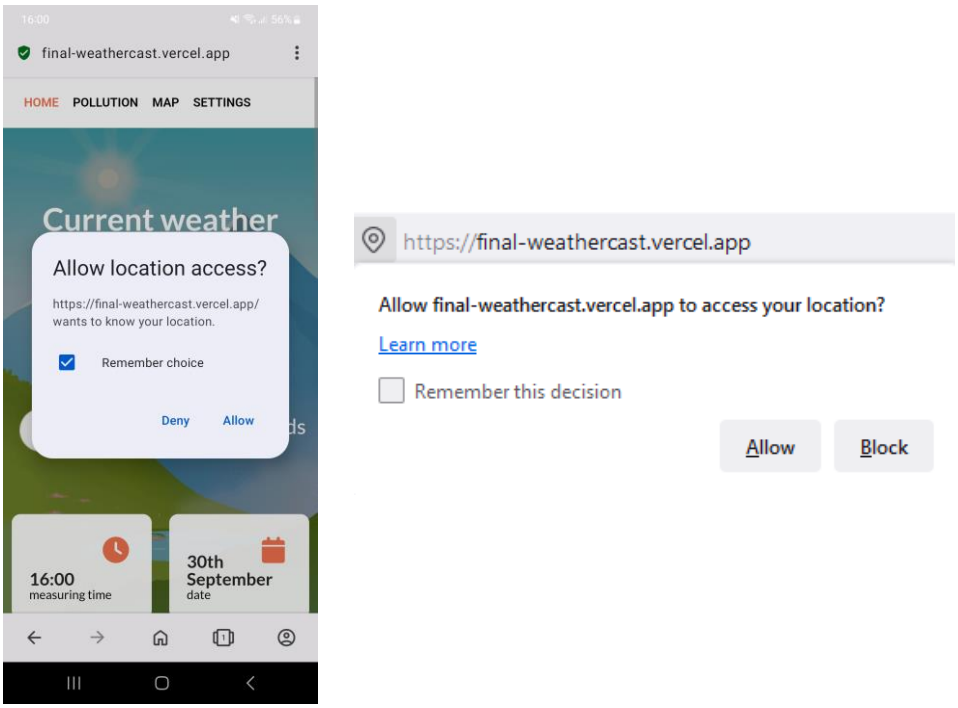
7.2. Zasloni

Četiri su glavna zaslona: Home, Pollution, Map, Settings. Zaslon Home sadrži glavne podatke o vremenu, a to su: lokacija, trenutna temperatura, vrijeme, datum, stvarni osjet temperature, vlažnost zraka, vrijeme izlaska i zalaska sunca, tlak zraka, brzina vjetra, smjer vjetra i naleti vjetra. Ispod navedenog nalazi se i prognoza za ostatak dana i nadolazeće dane. Drugi zaslon, Pollution, prikazuje količinu određenih čestica u zraku i zagađenost zraka. Treći zaslon Map sadrži mapu s

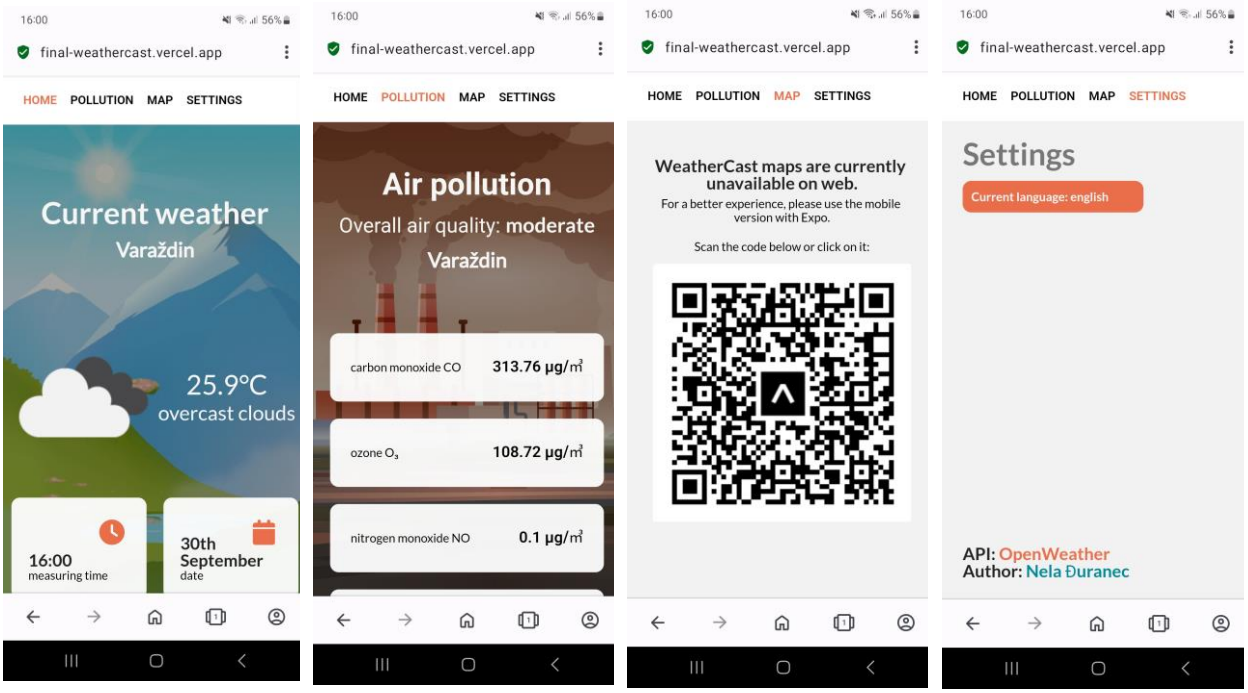
trenutnom lokacijom, a klikom na oznaku prikazuje se trenutno vrijeme. Na sva tri zaslona moguće je promijeniti lokaciju. Zaslon Settings sadrži postavke za promjenu jezika (engleski i hrvatski), podatke o autoru aplikacije (autorici ovoga rada) te link na API koji je korišten.



Slika 7.2 Izgled sva 4 ekrana na mobilnom uređaju



Slika 7.3 Dohvat lokacije na mobilnom i web pregledniku

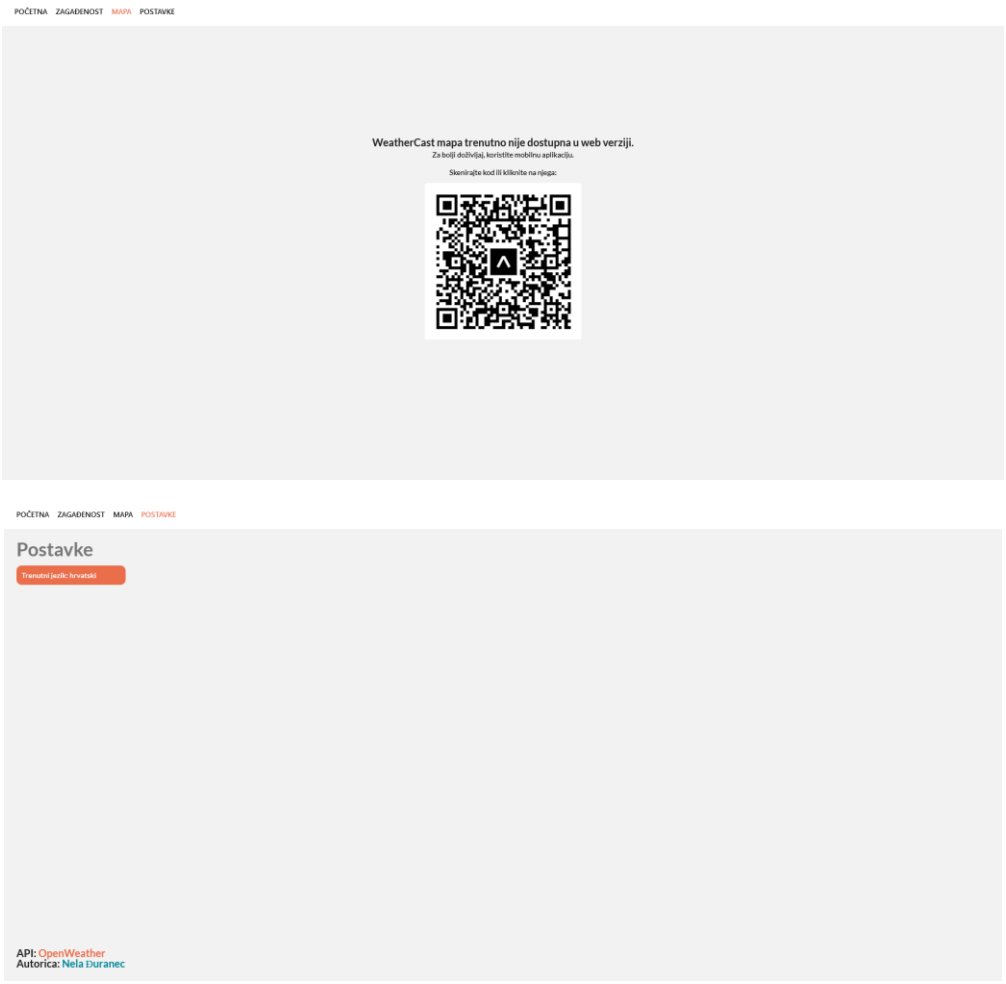


Slika 7.4 Izgled sva 4 ekrana na mobilnom pregledniku


ZAGAĐENOST MAPA POSTAVKE
Trenutno vrijeme
Varaždin
26.5°C
oblačno
16:05
vrijeme mjerenja
30. rujan
datum
26.4°C
stvarni osjet
50%
vlažnost
06:51
izlazak
18:38
zalazak
1022 hPa
tlak zraka
1.1 m/s
brzina vjetra
SJZ
smjer vjetra
1.4 m/s
naleti vjetra
Nadolazeći dani
Danas
Nedjelja
Ponedjeljak
Utorak
Srijeda


26.4°C
stvarni osjet
50%
vlažnost
06:51
izlazak
18:38
zalazak
1022 hPa
tlak zraka
1.1 m/s
brzina vjetra
SJZ
smjer vjetra
1.4 m/s
naleti vjetra
Nadolazeći dani
Danas
Nedjelja
Ponedjeljak
Utorak
Srijeda


POČETNA ZAGAĐENOST MAPA POSTAVKE
Zagađenost zraka
Kvaliteta zraka: prosječna
Varaždin
ugljikov monoksid CO 313.76 µg/m³
sumporni dioksid SO₂ 0.27 µg/m³
ozon O₃ 108.72 µg/m³
amonijak NH₃ 1.73 µg/m³
dušikov monoksid NO 0.1 µg/m³
fine čestice tvari 1.61 µg/m³
dušikov dioksid NO₂ 1.26 µg/m³
grube čestice tvari 1.93 µg/m³



Slika 7.5 Izgled sva 4 ekrana na web pregledniku

8. Zaključak

Kod izrade ovoga rada vidljiva je velika sličnost između React.js i React Native-a. Iako je u React Native-u moguće izraditi i web verziju, rezultat nije savršen pa je za web bolje odvojeno koristiti React.js. React.js je odličan i dovoljan alat za izradu jednostavnijih web aplikacija i jednostavnije je izraditi responzivnost. React Native koristi se za veće projekte i mobilne aplikacije kojima je cilj bolje korisničko iskustvo i native osjećaj.

Neki od problema kod izrade aplikacije bili su nedovršene ili nekompatibilne biblioteke. Tako je došlo do problema korištenja react-native-maps biblioteke koja nije podržana za internetski (web) preglednik i još uvijek ne postoji rješenje. Iz tog razloga bilo je nemoguće implementirati mapu na webu. Aplikacija se testirala pomoću Expo Go aplikacije i mobitela. Mana takvog testiranja je sporo prvo učitavanje, trošenje baterije mobitela i neprestano usmjeravanje pogleda s računala na mobitel. Također, jedan mobilni uređaj nije dovoljan za testiranje responzivnosti aplikacije pa je potrebno imati iOS i Android uređaje različitih veličina. Rješenje bi bilo korištenje emulatora zbog lakšeg snalaženja.

Expo platforma znatno je olakšala izradu i objavljivanje React Native aplikacije. React ima veliku zajednicu korisnika pa je time opširnija i veća dokumentacija i podrška. Potvrđeno je da jedna osoba može izraditi aplikaciju koja će biti funkcionalna na iOS i Android uređajima pa čak i na webu. To je povoljnija opcija kod izrade aplikacija jer nisu potrebna dva posebna tima. Za jedan kod, koji radi na tri platforme, dovoljna je jedna osoba ili tim. Stoga je React Native sve popularniji kao i slični okviri poput Fluttera. Porastom popularnosti mobilnih uređaja i mobilnih aplikacija, raste i potražnja za programerima u tom području.

Bilo bi korisno projekt nadograditi Typescript-om u kojem se određuju tipovi varijabla što znatno olakšava pronalazak bugova. Poželjno je provesti UI/UX anketu za testiranje intuitivnosti aplikacije te samog izgleda aplikacije. Potrebno je pratiti razvoj pluginova i biblioteka te nadograditi aplikaciju mapom za web verziju.

9. Literatura

[1] statista, »Facebook users reach by device 2022,« 24. 2. 2022. [Mrežno]. Available: https://www.statista.com/statistics/377808/distribution-of-facebook-users-by-device/. [Pokušaj pristupa 7. 8. 2023.].

[2] Stack Overflow, »Developer Survey 2023,« [Mrežno]. Available: https://survey.stackoverflow.co/2023/#section-admired-and-desired-web-frameworks-and-technologies. [Pokušaj pristupa 18. 9. 2023.].

[3] Incora, »Angular vs React vs Vue: The Main Differences and Use Cases,« 4. 2. 2022. [Mrežno]. Available: https://incora.software/insights/react-vs-angular-vs-vue. [Pokušaj pristupa 24. 7. 2023].

[4] A. Boduch, React and React Native, 2017.

[5] statista, »Global YouTube viewing time share by device 2021,« 23. 11. 2021. [Mrežno]. Available: https://www.statista.com/statistics/1173543/youtube-viewing-time-share-device/. [Pokušaj pristupa 7. 8. 2023.].

[6] webiotic, »Best Way to Develop Hybrid Mobile Apps?,« 22. 2. 2023. [Mrežno]. Available: https://www.webiotic.com/best-way-to-develop-hybrid-mobile-apps/. [Pokušaj pristupa 10. 7. 2023.].

[7] React Native, »Introduction,« [Mrežno]. Available: https://reactnative.dev/docs/the-new-architecture/landing-page. [Pokušaj pristupa 11. 7. 2023.].

[8] S. Shaw, »JavaScript in Plain English,« Medium, 12. 8. 2022. [Mrežno]. Available: https://javascript.plainenglish.io/react-native-new-architecture-old-vs-new-d0130f42bc79. [Pokušaj pristupa 7. 9. 2023.].

[9] React Native, »Setting up the development environment,« [Mrežno]. Available: https://reactnative.dev/docs/environment-setup. [Pokušaj pristupa 10. 7. 2023.].

[10] React Native, »Linking,« [Mrežno]. Available: https://reactnative.dev/docs/linking. [Pokušaj pristupa 26. 7. 2023.].

[11] React Native, »React Fundamentals,« [Mrežno]. Available: https://reactnative.dev/docs/intro-react. [Pokušaj pristupa 23. 7. 2023.].

[12] i18next, »Introduction,« [Mrežno]. Available: https://react.i18next.com/.

[13] Vercel, »React Native Directory,« [Mrežno]. Available: https://reactnative.directory/. [Pokušaj pristupa 23. 8 2023.].

[14] Amazon, »What Is An API (Application Programming Interface)?,« [Mrežno]. Available: https://aws.amazon.com/what-is/api/. [Pokušaj pristupa 31. 8. 2023.].

[15] npm, »react-native-dotenv,« [Mrežno]. Available: https://www.npmjs.com/package/react-native-dotenv. [Pokušaj pristupa 24. 9. 2023.].

[16] D. Crockford, Javascript: The Good Parts, 2008.

[17] J. Davis, »How to Convert Wind Directions in Degrees to Compass Directions,« Campbell Scientific, 6. 1. 2016. [Mrežno]. Available: https://www.campbellsci.in/blog/convert-wind-directions. [Pokušaj pristupa 21. 7. 2023.].

[18] React Navigation, »Getting started,« [Mrežno]. Available: https://reactnavigation.org/. [Pokušaj pristupa 11. 7. 2023.].

[19] FontAwesome, »React Native,« [Mrežno]. Available: https://fontawesome.com/docs/web/use-with/react-native. [Pokušaj pristupa 31. 7. 2023.].

[20] J. Samp, »Sunsetting “expo publish” and Classic Updates,« Medium, 10. 2. 2023. [Mrežno]. Available: https://blog.expo.dev/sunsetting-expo-publish-and-classic-updates-6cb9cd295378. [Pokušaj pristupa 14. 9. 2023.].

[21] Expo, »Migrate from Classic Updates,« [Mrežno]. Available: https://docs.expo.dev/eas-update/migrate-from-classic-updates. [Pokušaj pristupa 14. 9. 2023.].

[22] Expo, »Publish websites,« [Mrežno]. Available: https://docs.expo.dev/distribution/publishing-websites/. [Pokušaj pristupa 31. 8. 2023.].

Popis slika

Popis tablica

Sveučilište Sjever

SVEUČILIŠTE SJEVER

IZJAVA O AUTORSTVU
I
SUGLASNOST ZA JAVNU OBJAVU

Završni/diplomski rad isključivo je autorsko djelo studenta koji je isti izradio te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, Nela Duranec *(ime i prezime)* pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ica završnog/~~diplomskog~~ *(obrisati nepotrebno)* rada pod naslovom IZRADA APLIKACIJE POMOĆU REACT NATIVE JAVASCRIPT RAZVOJNOG OKVIRA *(upisati naslov)* te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:
(upisati ime i prezime)

Nela Duranec
(vlastoručni potpis)

Sukladno Zakonu o znanstvenoj djelatnost i visokom obrazovanju završne/diplomske radove sveučilišta su dužna trajno objaviti na javnoj internetskoj bazi sveučilišne knjižnice u sastavu sveučilišta te kopirati u javnu internetsku bazu završnih/diplomskih radova Nacionalne i sveučilišne knjižnice. Završni radovi istovrsnih umjetničkih studija koji se realiziraju kroz umjetnička ostvarenja objavljuju se na odgovarajući način.

Ja, Nela Duranec *(ime i prezime)* neopozivo izjavljujem da sam suglasan/na s javnom objavom završnog/~~diplomskog~~ *(obrisati nepotrebno)* rada pod naslovom IZRADA APLIKACIJE POMOĆU REACT NATIVE JAVASCRIPT RAZVOJNOG OKVIRA *(upisati naslov)* čiji sam autor/ica.

Student/ica:
(upisati ime i prezime)

Nela Duranec
(vlastoručni potpis)