Izrada hibridne mobilne aplikacije pomoću Capacitora

Srnec, Bruno

Sveučilište Sjever

Završni rad br. 925/MM/2024

Izrada hibridne mobilne aplikacije pomoću Capacitora

Bruno Srnec, 0016135590

Varaždin, rujan 2024. godine

Sveučilište
Sjever

Odjel za Multimediju, oblikovanje i primjenu

Završni rad br. 925/MM/2024

Izrada hibridne mobilne aplikacije pomoću Capacitora

Student

Bruno Srnec, 2224/336

Mentor

Vladimir Stanisavljević, mr.sc.

Varaždin, rujan 2024. 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 Bruno Srnec | JMBAG 0016135590

DATUM 09.09.2024. | KOLEGIJ Programski alati 3

NASLOV RADA Izrada hibridne mobilne aplikacije pomoću Capacitora

NASLOV RADA NA ENGL. JEZIKU Developing a hybrid mobile application with Capacitor

MENTOR Vladimir Stanisavljević | ZVANJE viši predavač

ČLANOVI POVJERENSTVA

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

Zadatak završnog rada

BROJ 925/MM/2024

OPIS

Za izradu hibridnih mobilnih aplikacije koriste se HTML tehnologije kao podloga za programiranje. Za iskorištenje svih ugrađenih mogućnosti pojedine mobilne i stolne platforme poput lokacijskih usluga ili kamere potrebno je mobilnu aplikaciju pisati u nekom specifičnom alatu koji iz HTML-a omogućuje pristup tim mogućnostima. Capacitor je jedan takav alat novije generacije. Podaci se u aplikaciju mogu učitati iz vanjskih izvora na internetu tipično AJAX tehnologijama. U ovom radu potrebno je izraditi višeplatformsku aplikaciju koja koristi HTML, vanjske izvore podataka i Capacitor.
U radu je potrebno:
• opisati osnovne Capacitor alata i pripadne programske zbirke te mogućnosti koje pruža za razvoj aplikacija i usporediti ga sa sličnim platformama,
• istražiti i opisati neko sučelje za pristup udaljenim izvorima podataka i podatke iskoristiti u aplikaciji,
* vizualizirati primljene podatake proizvoljnim grafikonom,
• pomoću Capacitora izraditi složeniju aplikaciju koja koristi ugrađene mogućnosti mobilnih platformi i tablično i grafički prikazuje podatke pribavljene iz udaljenih izvora podataka.

Detaljno opisati sve korištene tehnologije i korake u radu potrebne da bi se ostvarilo traženo te detaljno opisati stečena iskustva i postignute rezultate.

ZADATAK URUČEN 18.09.2024. | POTPIS MENTORA Vladović

REPUBLIKA HRVATSKA
SVEUČILIŠTE SJEVER

Predgovor

Ova tema završnog rada odabrana je na preporuku profesora Vladimira Stanislavljevića sukladno s vlastitim zanimamanjem za razvoj aplikacija, te front-end programiranjem. Cilj izrade mobilne hibrinde aplikacije služeći se Capacitorom je steći novo iskustvo i znanje u radu s manje poznatim frameworkovima kao što je ovaj, te isto iskustvo i znanje prenjeti drugima.

Želio bih se zahvaliti mentoru Vladimiru Stanislavljeviću na prenesenom znanju i uputama, obitelji na financijskoj i emotivnoj podršci koja se također pripisuje i prijateljima.

Sažetak

Hibridne aplikacije postaju sve popularnije zbog svoje sposobnosti da se jednom razvijena aplikacija koristi na više platformi, čime se smanjuju troškovi i vrijeme razvoja. Ovaj rad se bavi izradom hibridne mobilne aplikacije koristeći Capacitor, moderni alat za razvoj mobilnih aplikacija temeljenih na web tehnologijama. Cilj rada je pružiti uvid u mogućnosti Capacitor-a te demonstrirati njegovu primjenu kroz izradu konkretne aplikacije za prikaz mjerenja kvalitete zraka u Hrvatskoj.

U prvom dijelu rada predstavljena je platforma Capacitor, koja omogućava kreiranje hibridnih mobilnih aplikacija koristeći standardne web tehnologije poput HTML-a, CSS-a i JavaScript-a. Capacitor je posebno osmišljen kako bi olakšao integraciju nativnih funkcionalnosti uređaja, čime se postiže gotovo nativno iskustvo za korisnika. Detaljno su opisane ključne značajke Capacitor-a, kao što su podrška za više platformi, jednostavna integracija s popularnim JavaScript frameworkovima i podrška za nativne plugine. Također, u ovom dijelu rada razmatra se konkurencija Capacitoru, uključujući platforme kao što su Apache Cordova i React Native, te su analizirane njihove prednosti i nedostaci u usporedbi s Capacitorom.

Drugi dio rada fokusira se na praktičnu primjenu Capacitor-a kroz izradu mobilne aplikacije koja prikazuje mjerenja kvalitete zraka u Hrvatskoj. Opisan je proces konfiguracije radnog okruženja, uključujući instalaciju Capacitor-a i potrebnih ovisnosti. Razvoj aplikacije obuhvaća korake od inicijalizacije projekta, dodavanja platformi, do integracije API-ja za prikupljanje podataka o kvaliteti zraka. Posebna pažnja posvećena je dizajnu korisničkog sučelja, optimizaciji performansi te testiranju aplikacije.

Kroz razvoj aplikacije, obrađena su i pitanja vezana uz interakciju s korisnicima te prikaz i vizualizaciju podataka o kvaliteti zraka. Aplikacija omogućava korisnicima da u realnom vremenu prate kvalitetu zraka u različitim dijelovima Hrvatske, što je od velike važnosti za informiranje i osvješćivanje javnosti o ekološkim pitanjima.

Zaključno, rad ističe prednosti korištenja Capacitor-a u razvoju hibridnih aplikacija, uključujući bržu isporuku i bolju iskoristivost resursa, te naglašava važnost dostupnosti ekoloških podataka kroz moderne tehnologije.

Ključne riječi: Ionic, Capacitor, HTML, CSS, JavaScript, JSON, jQuery, Tailwindcss, Ajax, hibridne aplikacije

Abstract

Hybrid applications are becoming increasingly popular due to their ability to be used across multiple platforms with a single development effort, reducing both costs and development time. This paper focuses on the development of a hybrid mobile application using Capacitor, a modern tool for developing mobile applications based on web technologies. The goal of the paper is to provide insight into the capabilities of Capacitor and to demonstrate its application through the development of a specific app for displaying air quality measurements in Croatia.

The first part of the paper introduces the Capacitor platform, which allows for the creation of hybrid mobile applications using standard web technologies such as HTML, CSS, and JavaScript. Capacitor is specifically designed to simplify the integration of native device functionalities, providing users with a nearly native experience. The key features of Capacitor are described in detail, including support for multiple platforms, easy integration with popular JavaScript frameworks, and support for native plugins. Additionally, this part of the paper examines Capacitor's competition, including platforms such as Apache Cordova and React Native, analyzing their advantages and disadvantages compared to Capacitor.

The second part of the paper focuses on the practical application of Capacitor through the development of a mobile application that displays air quality measurements in Croatia. The process of configuring the development environment is described, including the installation of Capacitor and the necessary dependencies. The development of the application covers the steps from project initialization and platform addition to API integration for collecting air quality data. Special attention is given to the design of the user interface, performance optimization, and application testing.

Throughout the development of the application, issues related to user interaction and the display and visualization of air quality data are addressed. The application enables users to monitor air quality in real-time across different regions of Croatia, which is crucial for informing and raising public awareness of environmental issues.

In conclusion, the paper highlights the advantages of using Capacitor in hybrid application development, including faster delivery and better resource utilization, and emphasizes the importance of making environmental data accessible through modern technologies.

Keywords: Ionic, Capacitor, HTML, CSS, JavaScript, JSON, jQuery, TailwindCSS, Ajax hybrid applications

Popis korištenih kratica

HTML HyperText Markup Language
-sintaksa za obilježavanje hipertektualnih dokumenata.

API Application Programming Interface
- skup programskih pravila kojih se programeri moraju držati da bi ostvarili željene rezultate kod programa

CSS Cascading Style Sheets
- stilski jezik koji opisuje izgled HTML dokumenta

JSON JavaScript Object Notation
- otvoreni standardni format koji koristi čitljiv tekst za prijenos objekt podataka koji se sastoje od parova atribut-vrijednost

SDK Software Development Kit
- kup alata i resursa koje programeri koriste za razvoj aplikacija za specifičnu platformu ili tehnologiju

JDK Java Development Kit
- zbirka alata i resursa potrebnih za razvoj Java aplikacija

Sadržaj

1. Uvod 1
2. Ionic Team 3
   2.1. Povijest i Razvoj 3
   2.2. Ionic Framework 3
   2.3. Proširenje na Capacitor 4
   2.4. Ionic Pro: profesionalni alati i usluge 4
   2.5. Utjecaj Na Razvoj Mobilnih Aplikacija 5
3. Capacitor 6
   3.1. Osnovne Karakteristike Capacitor-a 6
   3.2. Kako Capacitor Funkcionira? 7
   3.3. Prednosti Capacitor-a 7
   3.4. Upotreba Capacitor-a 8
4. Alternative Capacitora 9
   4.1. Cordova 9
   4.2. React Native 10
   4.3. Flutter 11
   4.4. Xamarin 12
5. Capacitora plugin-ovi 13
6. Hibridne mobilne aplikacije 14
   6.1. Počeci i prve aplikacije 14
   6.2. Razvoj, poboljšanja napredak i inovacije 15
   6.3. Najuspješnije i Najpoznatije Hibridne Aplikacije 15
7. Potrebne tehnologije za rad u Capacitoru 19
   7.1. Android Studio 19
   7.2. Java Development Kit 21
8. Postavljanje projekta i korištene tehnologije 22
   8.1. Instalacija Capacitora, mogućnostti inicijalizacije projekta 22
   8.2. Node.js 23
   8.3. Stvaranje i konfiguracija projekta 24
   8.4. Tailwindcss 28

8.5. jQuery 30
8.6. AJAX 31
9. Izrada hibridne mobilne aplikacije u Capacitoru 32
9.1. „Glava“ html-a 32
9.2. Dohvaćanje najbliže postaje sa @capacitor/geolocation 34
9.3. Primjena AJAX-a i jQuery-a 39
9.4. Testiranje i izgled web aplikacije 46
9.5. Gradnja izvorne aplikacije za android 50
9.6. Zaključak praktičnog dijela 55
10. Zaključak 57
11. Literatura 59

1. Uvod

Ogroman napredak u razvoju mobilnih aplikacija postignut je u zadnjem desetljeću, gdje ključna komponenta u ovoj "revoluciji" postaju hibridne mobilne aplikacije. Upotreba web tehnologija u realizaciji aplikacija koje se mogu izvršavati na višestrukim platformama ima puno prednosti kao što su manji troškovi, jednostavno održavanje i brz razvoj. Među najistaknutijim alatima za izradu hibridnih mobilnih aplikacija jest Capacitor. Ovaj okvir, odnosno framework, koji nam omogućuje razvoj aplikacija visoke kvalitete uz korištenje najpoznatijih web tehnologija, razvila je tvrtka Ionic Team.

Capacitor je alat napravljen kako bi nadjačao ograničenja prijašnjih hibridnih frameworkova kao što je Cordova suvremenijim i prilagodljivijim pristupom. Naime, Capacitor ima mogućnost integracije sa suvremenim okvirima za front-end razvoj poput Reacta, Angulara ai Vue-a što uvelike olakšava rad developerima koji nemaju iskustva s njim. Ovaj okvir isto tako omogućava rad s nativnim funkcionalnostima platforme kroz jednostavne API-je (Application programming interface) što ga čini odličnim izborom za razvoj hibridnih aplikacija.

Ionic Team, poznat po razvoju naprednih alata za izradu mobilnih i web aplikacija, stekao je ugled prvenstveno kroz razvoj okvira Ionic. Povijest Ionic Teama i njegovih proizvoda može se pratiti od ranih 2010-ih godina kada su se pojavili prvi pokušaji da se olakša razvoj aplikacija za različite platforme korištenjem jedinstvenog koda. Njihova vizija bila je stvoriti okvir koji bi omogućio programerima da koriste web tehnologije kao što su HTML, CSS i JavaScript za izradu visokokvalitetnih mobilnih aplikacija koje izgledaju i ponašaju se kao nativne aplikacije.

Prvi značajan korak bio je lansiranje Ionic Frameworka 2013. godine, koji je ubrzo postao vrlo popularan među razvojnim programerima. Ionic Framework je otvoreni izvorni okvir koji omogućava razvoj visokokvalitetnih mobilnih aplikacija pomoću web tehnologija. Njegova sposobnost da pruži nativni izgled i dojam aplikacija postignuta je korištenjem prilagodljivih komponenti koje oponašaju nativne elemente korisničkog sučelja na različitim platformama.

U 2018. godini, Ionic Team je lansirao Capacitor, novi otvoreni izvorni okvir za izradu web-nativnih aplikacija. Capacitor je zamišljen kao modernija i fleksibilnija alternativa Cordovi, s ciljem da se riješe neka od ograničenja starijih hibridnih okvira. Capacitor omogućava programerima pristup nativnim funkcionalnostima uređaja putem web tehnologija, te se lako integrira s modernim front-end okvirima kao što su Angular, React i Vue.

Capacitor nudi brojne prednosti, uključujući platformsku nezavisnost, jedinstvene API-je, web-first pristup, integraciju s modernim framework-ovima i automatsko generiranje projekata za različite platforme. Ove značajke čine Capacitor atraktivnim alatom za razvoj hibridnih aplikacija koje mogu raditi na različitim platformama uz minimalne izmjene koda. Također, Capacitor je

modularan i proširiv, omogućavajući developerima da dodaju ili uklanjaju funkcionalnosti prema potrebi, te da kreiraju vlastite pluginove za specifične nativne funkcionalnosti.

Razvoj hibridnih aplikacija započeo je s potrebom za rješenjima koja bi omogućila rad aplikacija na različitim mobilnim platformama bez potrebe za pisanjem specifičnog koda za svaku od njih. Prvi pokušaji u ovom području pojavili su se početkom 2000-ih, a jedna od najranijih hibridnih aplikacija bila je aplikacija razvijena od strane kompanije Nitobi, poznata kao PhoneGap. PhoneGap je omogućio programerima da koriste HTML, CSS i JavaScript za razvoj mobilnih aplikacija koje su mogle pristupiti nativnim funkcionalnostima uređaja putem JavaScript API-ja. Kasnije je doniran Apache Software Foundation-u i preimenovan u Apache Cordova, čime je nastavljena evolucija hibridnih aplikacija.

Razvoj tehnologija koje omogućuju pristup nativnim funkcionalnostima uređaja kroz JavaScript bio je ključan za ovu evoluciju. Ova inovacija omogućila je programerima da razviju aplikacije koje izgledaju i ponašaju se kao nativne aplikacije, ali koristeći jedinstveni kod za sve platforme. Tijekom godina, hibridne aplikacije su se razvijale kako bi postale učinkovitije i pružile bolje korisničko iskustvo.

Ovaj rad fokusirat će se na izradu mobilne hibridne aplikacije pomoću Capacitora, s posebnim naglaskom na postupak izrade aplikacije za mjerenje kvalitete zraka u Hrvatskoj. Izrada takve aplikacije zahtijeva temeljito razumijevanje Capacitora i njegovih mogućnosti, kao i sposobnost integracije s različitim web tehnologijama i API-ima. Rad će opisati korake potrebne za postavljanje projekta, dodavanje potrebnih platformi i funkcionalnosti, te testiranje i distribuciju aplikacije. Također, bit će objašnjeni izazovi s kojima se developeri mogu susresti tijekom razvoja i kako ih prevladati korištenjem dostupnih alata i resursa.

Cilj ovog rada je pružiti detaljan vodič za izradu mobilne hibridne aplikacije pomoću Capacitora, te pokazati kako se ovaj moderan okvir može iskoristiti za razvoj visokokvalitetnih aplikacija koje kombiniraju fleksibilnost web tehnologija s performansama nativnih aplikacija. Nadamo se da će ovaj rad biti koristan resurs za developere koji žele istražiti mogućnosti hibridnog razvoja i koristiti Capacitor za svoje projekte.

2. Ionic Team

Ionic Team je poznat po razvoju naprednih alata za izradu mobilnih i web aplikacija, a svoj je ugled stekao prvenstveno kroz razvoj okvira Ionic, koji je postao jedan od najpopularnijih alata za izradu hibridnih aplikacija. Povijest Ionic Teama i njegovih proizvoda može se pratiti od ranih 2010-ih godina kada su se pojavili prvi pokušaji da se olakša razvoj aplikacija za različite platforme korištenjem jedinstvenog koda. [1]

Slika 1: Ionic team logo

2.1. Povijest i Razvoj

Ionic Team osnovali su Max Lynch, Ben Sperry i Adam Bradley 2012. godine. Njihova vizija bila je stvoriti okvir koji bi omogućio programerima da koriste web tehnologije kao što su HTML, CSS i JavaScript za izradu visokokvalitetnih mobilnih aplikacija koje izgledaju i ponašaju se kao nativne aplikacije. Prvi značajan korak bio je lansiranje Ionic Frameworka 2013. godine, što je ubrzo postalo vrlo popularno među razvojnim programerima.

2.2. Ionic Framework

Ionic Framework je otvoreni izvorni okvir koji omogućava razvoj visokokvalitetnih mobilnih aplikacija pomoću web tehnologija. Jedan od ključnih elemenata uspjeha Ionic Frameworka je njegova sposobnost da pruži nativni izgled i dojam aplikacija, što je postignuto korištenjem prilagodljivih komponenti koje oponašaju nativne elemente korisničkog sučelja na različitim platformama. [2]

Ionic Framework je ubrzo postao popularan zbog svojih brojnih prednosti, uključujući:

- **Jedinstveni kod**: Programeri mogu pisati jedinstveni kod za sve platforme, što značajno smanjuje vrijeme i resurse potrebne za razvoj.
- **Prilagodljive komponente**: Ionic nudi širok spektar prilagodljivih komponenti koje omogućuju izradu aplikacija s nativnim izgledom i osjećajem.
- **Integracija s Angularom**: Ionic je u početku bio snažno povezan s Angular frameworkom, što je omogućilo lakšu integraciju i korištenje postojećih Angular resursa.
- **Community Support**: Ionic ima veliku i aktivnu zajednicu programera koja kontinuirano doprinosi razvoju okvira i pruža podršku novim korisnicima.

2.3. Proširenje na Capacitor

U 2018. godini, Ionic Team je lansirao Capacitor, novi otvoreni izvorni okvir za izradu web-nativnih aplikacija. Capacitor je zamišljen kao modernija i fleksibilnija alternativa Cordovi, s ciljem da se riješe neka od ograničenja starijih hibridnih okvira. Capacitor omogućava programerima pristup nativnim funkcionalnostima uređaja putem web tehnologija, te se lako integrira s modernim front-end okvirima kao što su Angular, React i Vue. [3]

2.4. Ionic Pro: profesionalni alati i usluge

Uz Ionic Framework i Capacitor, Ionic Team nudi i niz profesionalnih alata i usluga pod nazivom Ionic Pro. Ionic Pro uključuje različite alate za razvoj, testiranje, distribuciju i praćenje aplikacija. Neki od ključnih alata i usluga su:

- **Ionic Appflow**: Alat za kontinuiranu integraciju i isporuku (CI/CD) koji omogućava automatsko testiranje, gradnju i distribuciju aplikacija.
- **Ionic Monitor**: Alat za praćenje performansi aplikacija u stvarnom vremenu, omogućavajući programerima da identificiraju i riješe probleme prije nego što utječu na korisnike.
- **Ionic Deploy**: Alat za daljinsko ažuriranje aplikacija, omogućavajući programerima da brzo i jednostavno implementiraju promjene bez potrebe za ponovnim objavljivanjem aplikacije na trgovinama. [1]

2.5. Utjecaj Na Razvoj Mobilnih Aplikacija

Ionic Team je svojim inovacijama značajno utjecao na razvoj mobilnih aplikacija. Kroz svoje alate i okvire, omogućili su programerima da koriste poznate web tehnologije za izradu aplikacija koje izgledaju i ponašaju se kao nativne. Ovo je posebno značajno za manje timove i individualne programere koji nemaju resurse za razvoj i održavanje zasebnih aplikacija za različite platforme.

Ionic Framework i Capacitor su otvorili vrata brojnim mogućnostima za brži i efikasniji razvoj aplikacija, smanjujući pritom troškove i vrijeme potrebno za razvoj. Njihova sposobnost da kombiniraju fleksibilnost web tehnologija s performansama nativnih aplikacija omogućila je programerima da se usmjere na izradu visokokvalitetnih aplikacija koje pružaju izvrsno korisničko iskustvo. [4]

3. Capacitor

Capacitor je moderan framework razvijen od strane Ionic tima, dizajniran za stvaranje visokokvalitetnih hibridnih mobilnih aplikacija koje se mogu izvoditi na različitim platformama, uključujući iOS, Android, web i Progressive Web Apps (PWA). Kao nasljednik Cordove, Capacitor je stvoren kako bi prevladao određene nedostatke svog prethodnika, nudeći modernije pristupe i veće mogućnosti.

Slika 2: Capacitor logo

3.1. Osnovne Karakteristike Capacitor-a

1. **Platformska Nezavisnost**: Capacitor omogućava razvoj aplikacija koje se mogu izvršavati na različitim platformama uz minimalne izmjene koda. To uključuje podršku za iOS, Android i web aplikacije, kao i Progressive Web Apps (PWA).
2. **Jedinstveni API-ji**: Capacitor koristi jedinstveni set API-ja koji omogućavaju pristup nativnim funkcionalnostima uređaja, kao što su kamera, geolokacija, mreža i drugo. Ovi API-ji su dizajnirani da budu jednostavni za korištenje i konzistentni između različitih platformi.
3. **Web-First Pristup**: Capacitor je dizajniran s fokusom na web tehnologije. Aplikacije su uglavnom napisane koristeći HTML, CSS i JavaScript, što omogućava web developerima da iskoriste svoje postojeće vještine za razvoj mobilnih aplikacija.

4. **Integracija s Modernim Framework-ovima**: Capacitor se lako integrira s popularnim JavaScript framework-ovima kao što su Angular, React i Vue. To omogućava developerima da koriste omiljene alate i biblioteke tijekom razvoja aplikacija.
5. **Automatsko Generiranje Projekata**: Capacitor automatizira proces generiranja nativnih projekata za iOS i Android, što olakšava postavljanje i održavanje projekata na različitim platformama.

3.2. Kako Capacitor Funkcionira?

Capacitor se sastoji od dva osnovna dijela: **Core** i **CLI**.

Capacitor Core: Core dio sadrži osnovnu logiku i API-je potrebne za komunikaciju između JavaScript koda i nativnih platformi. On pruža standardizirani set API-ja koji omogućavaju pristup nativnim funkcionalnostima uređaja.

Capacitor CLI (Command Line Interface): CLI alat omogućava developerima upravljanje Capacitor projektima. Pomoću CLI-a, developeri mogu kreirati nove projekte, dodavati platforme (iOS, Android), sinkronizirati web kod s nativnim projektima i još mnogo toga.

3.3. Prednosti Capacitor-a

1. **Jednostavna Migracija s Weba na Mobilne Platforme**: Capacitor omogućava web developerima jednostavnu migraciju njihovih postojećih web aplikacija na mobilne platforme. Korištenjem poznatih tehnologija kao što su HTML, CSS i JavaScript, developeri mogu brzo prilagoditi svoje aplikacije za iOS i Android.
2. **PWA Podrška**: Capacitor omogućava izradu Progressive Web Apps (PWA) aplikacija, koje kombiniraju najbolje značajke web i mobilnih aplikacija. PWA aplikacije mogu raditi offline, imaju brze performanse i pružaju korisnicima iskustvo slično nativnim aplikacijama.
3. **Modularnost i Proširivost**: Capacitor je modularan, što znači da developeri mogu dodavati ili uklanjati funkcionalnosti prema potrebi. Osim toga, lako se može proširiti vlastitim pluginovima koji omogućavaju pristup specifičnim nativnim funkcionalnostima.
4. **Aktivna Zajednica i Dokumentacija**: Capacitor ima aktivnu zajednicu developera i opsežnu dokumentaciju koja pomaže u učenju i rješavanju problema. Ionic tim redovito

ažurira framework i dodaje nove značajke, osiguravajući da Capacitor ostane suvremen i relevantan.

3.4. Upotreba Capacitor-a

Capacitor je dizajniran da pojednostavi proces razvoja hibridnih aplikacija, omogućavajući developerima korištenje već poznatih web tehnologija. Aplikacije razvijene uz pomoć Capacitor-a lako se mogu prilagoditi za rad na različitim platformama, što smanjuje vrijeme razvoja i održavanja. Capacitor je posebno koristan za timove koji žele iskoristiti postojeće web aplikacije i proširiti ih na mobilne platforme bez potrebe za učenjem potpuno novih tehnologija. [3]

4. Alternative Capacitora

Capacitor nudi prednosti kao što su jednostavna migracija s weba, podrška za moderne frontend frameworkove i integracija s nativnim funkcionalnostima bez potrebe za velikim promjenama u web kodu. Međutim, postoje određeni izazovi, poput performansi koje mogu biti slabije u usporedbi s nativnim rješenjima i ograničenjima u pogledu složenijih nativnih funkcionalnosti.

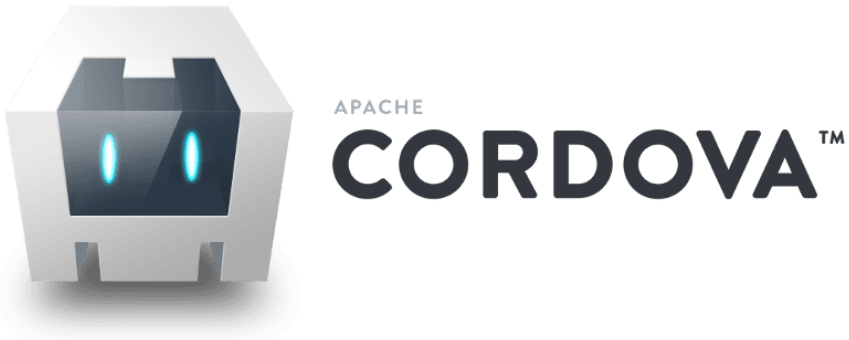
Cordova je najbliža alternativa Capacitor-u zbog sličnog pristupa hibridnom razvoju, ali Capacitor je moderniji i pruža bolje performanse i podršku za moderne razvojne prakse. [5]

React Native i **Flutter** pružaju bolje performanse i nativniji izgled i dojam, ali zahtijevaju više znanja i učenja novih jezika ili razvojnih paradigmi. [6, 7]

Xamarin nudi robustan razvoj s C# i nativnim performansama, ali može biti teži za učenje za developere koji dolaze iz web okruženja.

Svaki od ovih frameworkova ima svoje jedinstvene prednosti i nedostatke, a izbor između njih ovisi o specifičnim potrebama projekta, postojećem znanju tima i ciljevima aplikacije. [8]

4.1. Cordova



Slika 3: Cordova logo

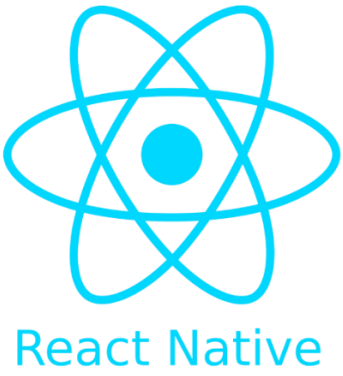
Prednosti:

1. **Široka podrška za pluginove**: Cordova ima veliki broj gotovih pluginova koji omogućuju pristup nativnim funkcionalnostima.
2. **Dugogodišnja upotreba**: Kao jedan od prvih hibridnih frameworkova, Cordova ima veliku zajednicu i mnogo resursa za učenje.
3. **Jednostavna integracija**: Cordova se lako integrira s različitim frontend framework-ovima, poput Angulara i Reacta.

Nedostaci:

1. **Zastarjelost**: Cordova je stariji framework i nije dizajniran za moderne razvojne prakse.
2. **Performanse**: Aplikacije napravljene s Cordovom mogu imati slabije performanse u usporedbi s onima napravljenima u Capacitoru ili nativenim rješenjima.
3. **Teži razvoj**: Razvoj i održavanje Cordova pluginova može biti izazovno zbog fragmentirane dokumentacije i starijih alata.

4.2. React Native



Slika 4: React native logo

Prednosti:

1. **Performanse**: React Native koristi native komponente, što rezultira boljim performansama u usporedbi s hibridnim rješenjima poput Capacitor-a.
2. **Popularnost**: Velika zajednica i podrška od strane Facebooka omogućuju brzo rješavanje problema i bogatu kolekciju biblioteka.

3. **Hot Reloading**: Omogućuje brži razvoj uz trenutne promjene koda vidljive u aplikaciji.

Nedostaci:

1. **Učenje**: Za developere koji dolaze iz web razvoja, učenje React Native-a može biti izazovno zbog potrebe za razumijevanjem nativnih platformskih komponenti.
2. **Kompleksnost**: Više integracija i prilagodbi može povećati kompleksnost razvoja.
3. **Ovisnost o trećim stranama**: Mnogi pluginovi dolaze od trećih strana, što može rezultirati problemima s održavanjem i kompatibilnošću.

4.3. Flutter

Slika 5: Flutter logo

Prednosti:

1. **Performanse**: Flutter aplikacije su gotovo nativne performanse zbog Dart jezika i renderiranja na osnovnoj razini.
2. **Jedinstveni UI**: Flutter omogućuje izradu prilagođenih, visokokvalitetnih UI komponenata koje su iste na svim platformama.
3. **Hot Reload**: Omogućuje brzo iteriranje i brži razvoj aplikacija.

Nedostaci:

1. **Učenje Darta**: Dart nije toliko popularan kao JavaScript, pa developeri moraju naučiti novi jezik.

2. **Velike aplikacije**: Aplikacije napravljene u Flutteru mogu biti veće u veličini zbog ugrađenog Flutter engine-a.
3. **Mlađa tehnologija**: Flutter je relativno noviji framework, što znači manje dostupnih resursa i manje iskusna zajednica u usporedbi s nekim drugim frameworkovima.

4.4. Xamarin

Slika 6: Xamarin logo

Prednosti:

1. **Performanse**: Xamarin aplikacije koriste nativne kontrole i imaju gotovo nativne performanse.
2. **Jedinstveni kod za logiku**: Mogućnost dijeljenja logike aplikacije između različitih platformi koristići C#.
3. **Podrška od strane Microsofta**: Microsoftova podrška donosi stabilnost i integraciju s alatima poput Visual Studio.

Nedostaci:

1. **Učenje**: C# i .NET ekosustav mogu biti izazovni za developere koji dolaze iz JavaScript ili drugih web jezika.
2. **Veličina aplikacija**: Xamarin aplikacije mogu biti veće u veličini zbog uključivanja .NET runtime-a.
3. **Razvoj sučelja**: Razvoj korisničkog sučelja može biti složeniji u usporedbi s nekim drugim frameworkovima jer zahtijeva više prilagodbi za svaku platformu.

5. Capacitora plugin-ovi

Plugin-ovi su ključni alati u modernom softverskom razvoju jer omogućuju proširenje funkcionalnosti aplikacija bez potrebe za izmjenama osnovnog koda. U kontekstu mobilnog razvoja, kao što je s Capacitorom, plug-inovi omogućuju pristup nativnim značajkama uređaja kao što su kamera, geolokacija ili mrežna povezanost, sve kroz jedinstveno sučelje. Ova modularnost ne samo da ubrzava razvojni proces, već i omogućuje jednostavniju integraciju novih tehnologija, osiguravajući da aplikacije mogu brzo evoluirati i prilagoditi se novim zahtjevima tržišta. Iako se u praktičnom dijelu rada koristi samo jedan od njih, radi primjera integracije, vrijedno je navesti najkorisnije plugin-ove koje Capacitor nudi;

Naziv plugina	Opis
@capacitor/camera	Omogućava pristup kameri uređaja i snimanje slika.
@capacitor/geolocation	Pristup lokaciji uređaja putem GPS-a i drugih lokacijskih usluga.
@capacitor/storage	Služi za lokalno spremanje podataka na uređaj u ključ-vrijednost formatu.
@capacitor/filesystem	Omogućava rad s datotekama na uređaju, poput spremanja i učitavanja datoteka.
@capacitor/device	Pristup informacijama o uređaju, kao što su model, verzija operativnog sustava i ostale tehničke informacije.
@capacitor/network	Provjera mrežnog statusa (povezanosti) uređaja.
@capacitor/haptics	Upravljanje haptičkim povratnim informacijama na uređaju.
@capacitor/clipboard	Omogućuje rad s međuspremnikom uređaja (kopiranje i lijepljenje teksta).
@capacitor/toast	Prikazuje kratke poruke korisniku na zaslonu (toast poruke).
@capacitor/splash-screen	Upravljanje splash zaslonom aplikacije prilikom pokretanja.
@capacitor/push-notifications	Implementacija push notifikacija na Androidu i iOS-u.
@capacitor/app	Pruža funkcionalnosti za upravljanje događajima u aplikaciji (npr. izlazak, pozadina).
@capacitor/browser	Otvara vanjske URL-ove u pregledniku uređaja.
@capacitor/status-bar	Omogućava upravljanje statusnom trakom uređaja.
@capacitor/keyboard	Upravljanje softverskom tipkovnicom na uređajima.

Tablica 1: Plugin-ovi

6. Hibridne mobilne aplikacije

Hibridne aplikacije predstavljaju spoj nativnih i web aplikacija. One koriste web tehnologije poput HTML-a, CSS-a i JavaScript-a za izradu aplikacija koje se izvršavaju unutar nativnog spremnika, omogućujući im pristup nativnim značajkama uređaja poput kamere, senzora i GPS-a. Ovaj pristup omogućava izradu aplikacija koje se mogu koristiti na različitim platformama, uključujući iOS, Android i Windows.

6.1. Počeci i prve aplikacije

Razvoj hibridnih aplikacija započeo je s potrebom za rješenjima koja bi omogućila rad aplikacija na različitim mobilnim platformama bez potrebe za pisanjem specifičnog koda za svaku od njih. Prvi pokušaji u ovom području pojavili su se početkom 2000-ih, a jedna od najranijih hibridnih aplikacija bila je aplikacija razvijena od strane kompanije Nitobi, poznata kao PhoneGap.

PhoneGap je bio jedan od prvih značajnih alata koji je omogućio razvoj hibridnih aplikacija. Razvijen 2009. godine, PhoneGap je omogućio programerima da koriste HTML, CSS i JavaScript za razvoj mobilnih aplikacija koje su mogle pristupiti nativnim funkcionalnostima uređaja putem JavaScript API-ja. Ovaj alat ubrzo je postao popularan među programerima jer je omogućio brži i jednostavniji razvoj aplikacija koje su se mogle pokretati na više platformi. Kasnije je doniran Apache Software Foundation-u i preimenovan u Apache Cordova. Cordova je nastavila razvijati i unapređivati koncept hibridnih aplikacija, omogućujući programerima da koriste jedinstveni kod za razvoj aplikacija koje se mogu pokretati na različitim platformama. Cordova je također omogućila razvoj i upotrebu pluginova koji su proširili funkcionalnosti hibridnih aplikacija, omogućujući im pristup nativnim API-jima uređaja. [9]

Slika 7: PhoneGap logo

Jedna od prvih značajnih hibridnih aplikacija bila je Facebookova mobilna aplikacija. Facebook je isprva koristio HTML5 za razvoj svoje mobilne aplikacije, no ubrzo su se pojavili problemi s performansama i korisničkim iskustvom. Iako je Facebook kasnije prešao na nativni razvoj, njihov je pokušaj bio važan korak u razvoju hibridnih aplikacija i pokazao je potencijal i izazove ovog pristupa.

6.2. Razvoj, poboljšanja napredak i inovacije

Nakon prvih pokušaja, mnoge su kompanije počele istraživati hibridni pristup. Ključni ciljevi bili su poboljšanje performansi i korisničkog iskustva, koji su često bili inferiorni u odnosu na nativne aplikacije. Razvoj prilagodljivih komponenti koje oponašaju nativne elemente korisničkog sučelja na različitim platformama značajno je poboljšao korisničko iskustvo hibridnih aplikacija.

Cordova je postavila temelje za daljnji razvoj hibridnih aplikacija, a novi alati i okviri počeli su se pojavljivati kako bi unaprijedili ovaj pristup. Ionic Framework, lansiran 2013. godine, bio je jedan od takvih alata koji je kombinirao snagu Cordove s modernim web tehnologijama kao što su AngularJS, HTML5 i CSS3. Ionic je omogućio brži razvoj i jednostavnije održavanje hibridnih aplikacija, čime je postao popularan izbor među programerima. [10]

Tijekom godina, hibridne aplikacije su se razvijale kako bi postale učinkovitije i pružile bolje korisničko iskustvo. Razvoj tehnologija koje omogućuju pristup nativnim funkcionalnostima uređaja kroz JavaScript bio je ključan za ovu evoluciju. Ova inovacija omogućila je programerima da razviju aplikacije koje izgledaju i ponašaju se kao nativne aplikacije, ali koristeći jedinstveni kod za sve platforme.

6.3. Najuspješnije i Najpoznatije Hibridne Aplikacije

Hibridne aplikacije postale su izuzetno popularne zbog svoje sposobnosti da rade na više platformi koristeći jedan kodni osnov. Ovo omogućuje tvrtkama da smanje troškove razvoja i vrijeme potrebno za lansiranje novih funkcionalnosti. Neke od najuspješnijih i najpoznatijih hibridnih aplikacija uključuju Instagram, Uber, Twitter, Sworkit, Untappd i Pacifica.

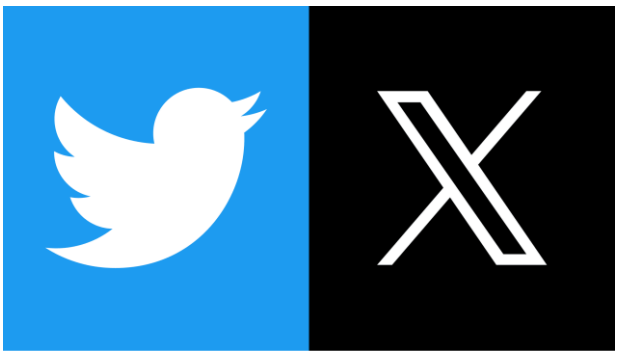
Instagram je jedna od najpopularnijih društvenih mreža za dijeljenje fotografija i videozapisa, s milijardama korisnika širom svijeta. Tvrtka koja stoji iza Instagrama koristi hibridni pristup za razvoj nekih svojih komponenti. Instagram koristi React Native za određene dijelove aplikacije, omogućujući timovima da dijele kod između Android i iOS platformi. Ovaj hibridni pristup omogućio je brže iteracije i poboljšanja performansi, pružajući korisnicima konzistentno iskustvo na različitim platformama.

Slika 8: Instagram logo

Uber je globalna platforma za prijevoz koja povezuje vozače s putnicima putem mobilne aplikacije. Uberova aplikacija koristi hibridni pristup kako bi pružila konzistentno korisničko iskustvo na različitim uređajima. Uber koristi vlastitu verziju frameworka koji se temelji na React Native kako bi postigli brzu i efikasnu izvedbu. Njihov glavni fokus je na pružanju brzih reakcija aplikacije, što je ključno za korisničko iskustvo. Time su omogućili brze performanse i odzivnost aplikacije, konzistentnost u funkcionalnosti i dizajnu na različitim platformama te mogućnost brzih ažuriranja i poboljšanja.

Slika 9: Uber logo

Twitter je još jedna velika društvena mreža koja koristi hibridni pristup za mobilne aplikacije kako bi osigurala dosljedno iskustvo na svim uređajima. U početku, Twitter je koristio različite nativne pristupe za svoje mobilne aplikacije, ali je prešao na hibridni pristup kako bi smanjio složenost razvoja i održavanja. Ovo im je omogućilo bržu integraciju novih značajki i funkcionalnosti te dosljednost u korisničkom iskustvu na svim platformama. Danas ova platforma nosi ime **X**.



Slika 10: Twitter i X logo

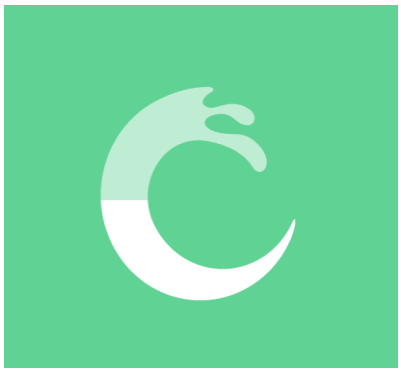
Sworkit je popularna fitness aplikacija koja nudi personalizirane video vježbe i planove treniranja. Sworkit koristi hibridni pristup za razvoj svoje aplikacije, omogućujući im da dosegnu širu publiku s minimalnim troškovima razvoja. Hibridna priroda aplikacije omogućava jednostavnu integraciju novih značajki i prilagodbu sadržaja različitim korisnicima.

Slika 11: Sworkit logo

Untappd je društvena mreža za ljubitelje piva koja omogućava korisnicima da ocjenjuju i dijele svoja iskustva s različitim pivima. Ova aplikacija koristi hibridni pristup kako bi osigurala dosljedno iskustvo na različitim platformama i omogućila korisnicima da lako pristupaju svim značajkama aplikacije. Hibridni pristup omogućava Untappdu da brzo implementira nove značajke i prilagodi se potrebama korisnika.

Slika 12: Untappd logo

Pacifica je aplikacija za mentalno zdravlje koja pomaže korisnicima u upravljanju stresom i anksioznošću kroz različite alate i tehnike. Pacifica koristi hibridni pristup kako bi omogućila korisnicima pristup aplikaciji na različitim uređajima s minimalnim troškovima razvoja. Hibridna priroda aplikacije omogućava timu da brzo dodaje nove funkcionalnosti i poboljšava korisničko iskustvo.



Slika 13: Pacifica logo

7. Potrebne tehnologije za rad u Capacitoru

Pošto je Capacitor framework koji nam omogućava razvoj aplikacija s web tehnologijama, za funkcionalnost tih aplikacija u nativnom okruženju potrebne su tehnologije i alati koji omogućuju prijevod i komunikaciju javascript koda sa hardware-om.

7.1. Android Studio

Android Studio službeno je integrirano razvojno okruženje (IDE) za razvoj aplikacija na Android platformi. Razvila ga je i održava tvrtka Google, a prvi put je najavljen na Google I/O konferenciji u svibnju 2013. godine. Android Studio temelji se na IntelliJ IDEA, popularnom IDE-u tvrtke JetBrains, poznatom po moćnim alatima za razvoj softvera. Ovo razvojno okruženje nudi specifične značajke i alate koji olakšavaju razvoj aplikacija za Android uređaje.

Android Studio omogućuje programerima da pišu, testiraju i implementiraju Android aplikacije. Osim razvoja aplikacija, nudi i alate za testiranje aplikacija na emulatorima, optimizaciju koda i otklanjanje pogrešaka. Kao službeni razvojni alat za Android, pruža potpuno integrirano okruženje s podrškom za sve aspekte razvoja Android aplikacija, uključujući dizajn korisničkog sučelja, razvoj logike aplikacije i integraciju s Android platformom.

Jedna od glavnih prednosti Android Studija je alat za dizajn korisničkog sučelja. Omogućuje jednostavno vizualno dizajniranje korisničkog sučelja, gdje programeri mogu povlačiti i ispuštati elemente kao što su gumbi, tekstualna polja i slike kako bi kreirali raspored aplikacije. Uz to, alat omogućuje pregled aplikacije u stvarnom vremenu na različitim uređajima i rezolucijama. Android Studio dolazi i s ugrađenim emulatorom koji omogućuje testiranje aplikacija bez potrebe za fizičkim uređajem. Emulator može simulirati različite Android uređaje, verzije operativnog sustava, pa čak i različite hardverske komponente poput senzora i kamera. To omogućuje brzo testiranje aplikacija u različitim uvjetima i na različitim uređajima.

Android Studio koristi napredni editor koda koji podržava automatsko dovršavanje koda, refaktoriranje i statičku analizu koda, što omogućuje brži rad s manje pogrešaka. Također, editor prepoznaje Android specifične elemente, poput XML datoteka za korisničko sučelje, te daje korisne sugestije tijekom razvoja. Sustav za izgradnju projekata temelji se na Gradleu, koji omogućuje lako upravljanje ovisnostima i prilagodbu izgradnji. Gradle sustav omogućuje optimizaciju aplikacija za različite verzije Androida i različite uređaje, što je ključno za široku kompatibilnost aplikacija.

Osim toga, Android Studio ima ugrađenu podršku za Git i druge sustave za verzioniranje, što olakšava suradnju unutar timova. Programeri mogu lako pratiti promjene u kodu, vraćati se na

prethodne verzije i upravljati različitim granama projekta. Još jedna bitna značajka Android Studija su alati za otklanjanje grešaka (debugging) koji omogućuju praćenje aplikacije dok se izvršava, postavljanje točaka zaustavljanja i pregledavanje varijabli i memorije. Alati za profiliranje pružaju detaljan uvid u performanse aplikacije, korištenje memorije, CPU-a i mrežnih zahtjeva, što je ključno za optimizaciju i izbjegavanje problema poput curenja memorije.

Android Studio podržava više programskih jezika, uključujući Javu, Kotlin, koji je sada službeni jezik za Android razvoj, te C++ za razvoj aplikacija koje zahtijevaju visoke performanse putem Android NDK-a. Ova fleksibilnost omogućuje programerima da biraju jezik koji najbolje odgovara njihovim potrebama.

Korištenje Android Studija donosi brojne prednosti. Osim što pruža najkompletnije razvojno okruženje za razvoj Android aplikacija, nudi sve potrebne alate za cijeli proces razvoja – od pisanja koda, testiranja i otklanjanja pogrešaka, pa sve do implementacije aplikacije na Google Play Store. Programeri tako mogu biti sigurni da koriste najsuvremenije alate i najbolje prakse koje preporučuje Google za razvoj Android aplikacija. Android Studio je besplatan i dostupan na svim glavnim operativnim sustavima, uključujući Windows, macOS i Linux, čime je dostupan širokom krugu programera. [13]

Slika 14: Android studio logo

7.2. Java Development Kit

JDK (Java Development Kit) je zbirka softverskih alata koja omogućuje razvoj aplikacija na programskom jeziku Java. JDK je osnovni alat potreban za stvaranje Java aplikacija i sastoji se od raznih komponenti kao što su Java kompajler, Java Virtual Machine (JVM), te biblioteke i alati potrebni za pokretanje i testiranje Java programa.

Glavna komponenta JDK-a je kompajler (javac), koji prevodi Java izvorni kod u bajtkod koji može biti izvršen na JVM-u. JVM je ključan jer omogućuje da Java programi budu prenosivi i kompatibilni na različitim platformama, što je jedna od glavnih prednosti Jave. Također, JDK uključuje alat za otklanjanje pogrešaka (debugger), biblioteke klasa koje olakšavaju razvoj aplikacija, te dodatne alate kao što su Java arhivski alat (jar) za pakiranje aplikacija.

Postoji u različitim verzijama, a svaka nova verzija donosi poboljšanja u smislu performansi, sigurnosti i novih značajki jezika. Na primjer, u JDK 11, Java je postala modularna, što omogućuje lakše upravljanje ovisnostima i smanjenje veličine aplikacija.

JDK je besplatan za osobnu upotrebu, no za komercijalne svrhe Oracle pruža licencirane verzije s dodatnim podrškama. Pored Oraclovog JDK-a, postoje i druge implementacije kao što su OpenJDK, koje su open-source i mogu se slobodno koristiti.

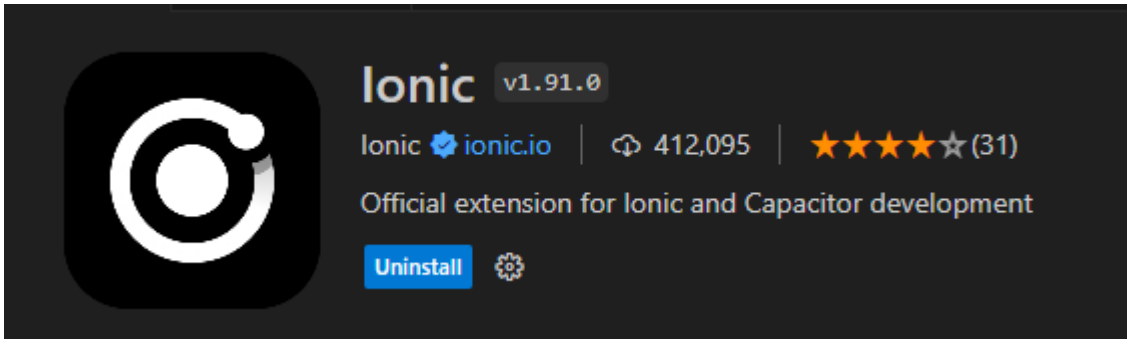
Ovo je ključni alat za sve koji razvijaju Java aplikacije, bilo da se radi o jednostavnim desktop aplikacijama, web aplikacijama ili složenim sustavima za poduzeća. Kao takav, JDK igra važnu ulogu u svijetu softverskog razvoja zbog svoje stabilnosti, sigurnosti i široke upotrebe.

Slika 15: JDK logo

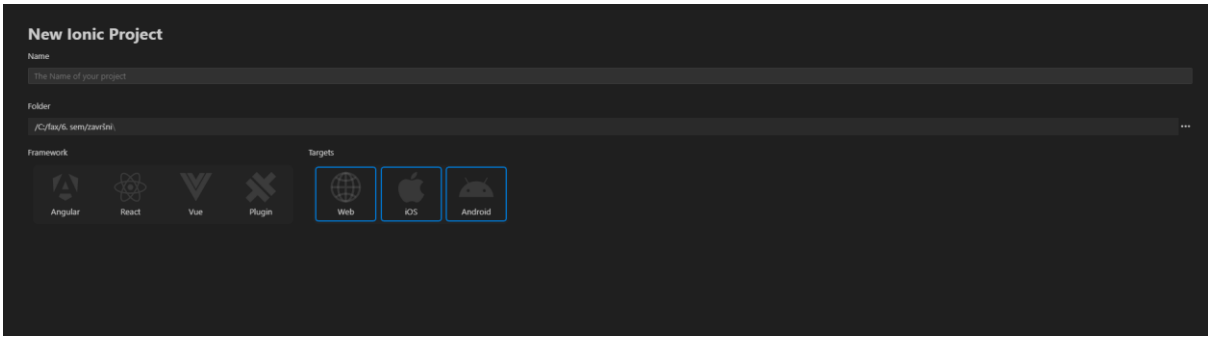
8. Postavljanje projekta i korištene tehnologije

8.1. Instalacija Capacitora, mogućnosti inicijalizacije projekta

Capacitor se može koristiti unutar Visual studio code-a, editora mog odabira, instalacijom Ionic plugina [*Slika 16*]. Nakon instalacije nam se u lijevoj vertikalnoj alatnoj traci pojavljuje mali Ionic gumb pomoću kojeg stvaramo projekt u određenom frameworku. Ponuđeni su Angular, React, Vue, te Plugin opcija, te možemo birati ime aplikacije, destinaciju datoteke i ciljane platforme (Web, IOS i Android) [*Slika 17*].



Slika 16: Ionic plugin



Slika 17: Stvaranje projekta

Pošto neću raditi u nijednom frameworku, već ću se služiti samo običnim javascriptom, projekt ću stvoriti u Windows PowerShell terminalu. Prije stvaranja projekta potrebno je osigurati postojanje node.js kako bismo se mogli služiti javascriptom u nativnom razvoju.

8.2. Node.js

Node.js je otvorena platforma koja omogućava izvođenje JavaScript koda izvan web preglednika. Razvijen je 2009. godine od strane Ryana Dahla i od tada je postao ključni alat za razvoj modernih aplikacija. Node.js koristi V8 JavaScript engine, isti onaj koji pokreće Google Chrome, što omogućava brzo i efikasno izvođenje JavaScript koda na strani servera.

Jedna od osnovnih karakteristika Node.js-a je njegov asinkroni I/O model. Ovaj model omogućava događajima vođeno, neblokirajuće rukovanje operacijama, što ga čini vrlo efikasnim I skalabilnim. Asinkroni model omogućava rukovanje velikim brojem simultanih konekcija bez potrebe za kreiranjem novih niti za svaku konekciju, čime se smanjuje memorijsko opterećenje I povećava brzina izvođenja.

Node.js koristi jednotračni event loop umjesto korištenja više niti za istovremeno izvršavanje zadataka. Kada Node.js mora čekati na neki zadatak, poput čitanja datoteke s diska ili čekanja na odgovor baze podataka, koristi callback funkcije koje će se izvršiti nakon što se zadatak završi. Ovaj pristup omogućava efikasno upravljanje resursima I izbjegavanje blokirajućih operacija.

Velika prednost Node.js-a je njegov ekosistem paketa, poznat 23erver23 (Node Package Manager). Npm je najveći ekosistem otvorenih knjižnica na svijetu, omogućavajući programerima da jednostavno instaliraju, dijele I ažuriraju pakete koda. Ovaj ekosistem ubrzava razvoj aplikacija I olakšava upravljanje ovisnostima, što dodatno povećava produktivnost programera.

Brzina I efikasnost su ključne prednosti Node.js-a. Zahvaljujući V8 engineu I asinkronom modelu, Node.js omogućava brzo izvođenje koda I efikasno rukovanje velikim brojem simultanih konekcija. Ovo ga čini idealnim za stvaranje real-time aplikacija, poput chat aplikacija I online igara, gdje je brzina odziva kritična.

Jednostavnost razvoja također je značajna prednost Node.js-a. Korištenje istog programskog jezika, JavaScripta, na klijentskoj I serverskoj 23erver pojednostavljuje 23erver23 razvoja. Programeri ne moraju prelaziti između različitih jezika I okruženja, što smanjuje kompleksnost I ubrzava razvoj.

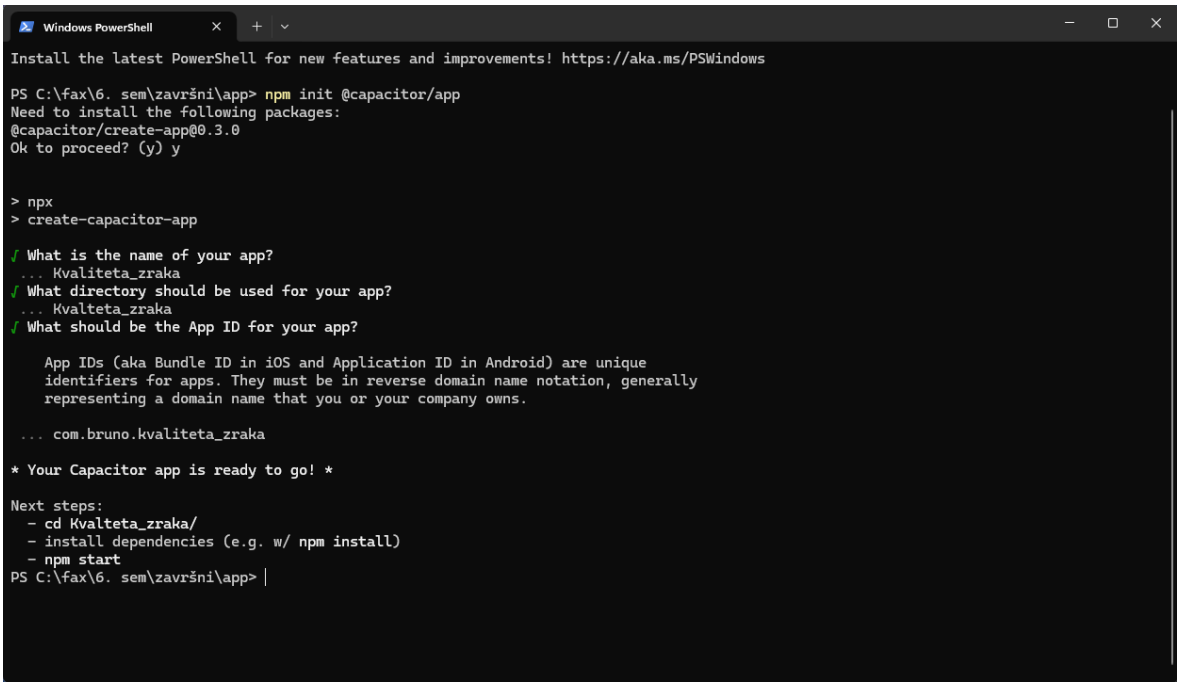
Velika zajednica korisnika I bogata dokumentacija također doprinose popularnosti Node.js-a. Kao popularna platforma, Node.js ima veliku zajednicu programera koji aktivno doprinose njegovom razvoju. Postoji obilje online resursa, tutorijala I dokumentacije, što olakšava učenje I rješavanje problema, te omogućava brži razvoj I implementaciju aplikacija.

Node.js je posebno pogodan za aplikacije koje zahtijevaju brzu I efikasnu obradu velikog broja simultanih zahtjeva. Web 23erver su jedan od najčešćih primjena Node.js-a, jer može rukovati velikim brojem istovremenih zahtjeva bez preopterećenja servera. Real-time aplikacije, kao što su chat aplikacije I online igre, također koriste Node.js zbog njegove brzine I efikasnosti. [11]

Slika 18: node.js logo

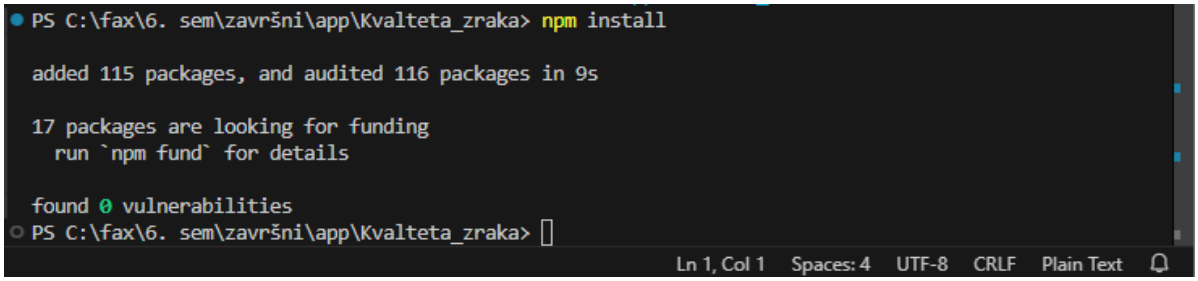
8.3. Stvaranje i konfiguracija projekta

Desnim klikom na mapu u kojoj želimo napraviti projekt, otvara nam se niz opcija od kojih je potrebno odabrati „Otvori u terminalu“. Ta opcija nam otvara PowerShell prozor koji nam navodi putanju do odabrane mape. Prva naredba koju upisujemo je „**npm init @capaitor/app**“ na što dobivamo par pitanja. Prvo je kako želimo da nam se aplikacija zove. Nakon unosa zadovoljavajućeg imena aplikacije (Kvaliteta_zraka u mojem slučaju), pritisnemo Enter, te slijedi koji direktorij želimo koristiti za aplikaciju. Unosom imena aplikacije nam se na destinaciji stvara jednakoimena mapa (Kvaliteta_zraka). Za kraj dobivamo pitanje koji App ID želimo, što nam je bitno za razvoj za platforme Android i IOS. Format App ID je „**com.example.app**“. [*Slika 19*]



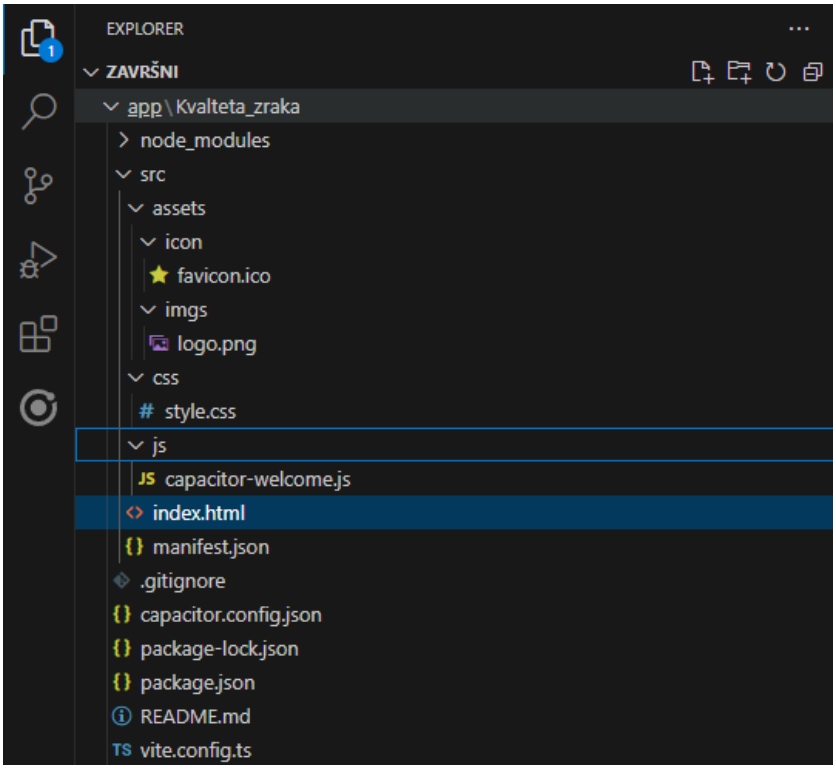
Slika 19: Stvaranje Capacitor projekta preko PowerShell-a

Unosom naredbe „**code Kvaliteta_zraka/**“ u terminal, možemo na brži način otvoriti projekt u zadanom code editoru, koji je u mom slučaju Visual Studio Code. Za ostalo postavljanje projekta koristi se terminal u samom VSC-u, pa kao prvi korak u editoru, uterminal unosimo naredbu „**npm install**“ koja izvršava instalaciju svih vanjskih biblioteka, frameworkova i modula na koje se naš projekt oslanja kako bi normalno funkcionirao. [*Slika 20*]



Slika 20: npm install

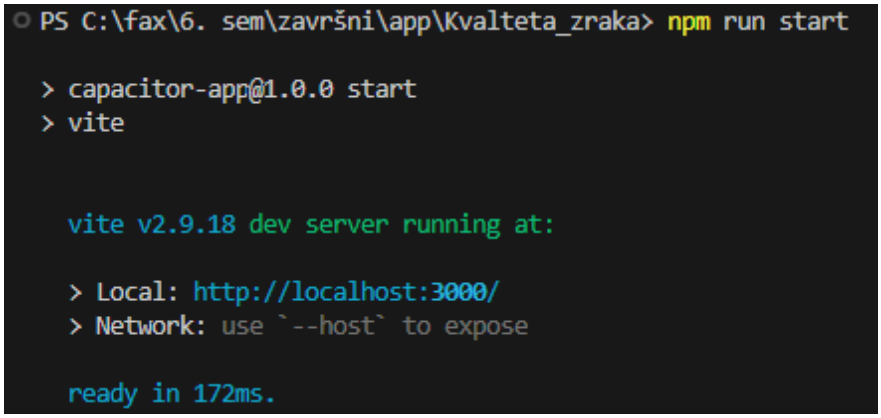
U Explorer sekciji, unutar datoteke u kojoj smo inicijalizirali projekt, u mom slučaju „app“ datoteka, sada imamo sljedeće komponente [*Slika 21*]. Ukoliko želimo promjeniti App ID ili ime aplikacije, to možemo napraviti u „**capacitor.config.json**“.



Slika 21: Komponente aplikacije

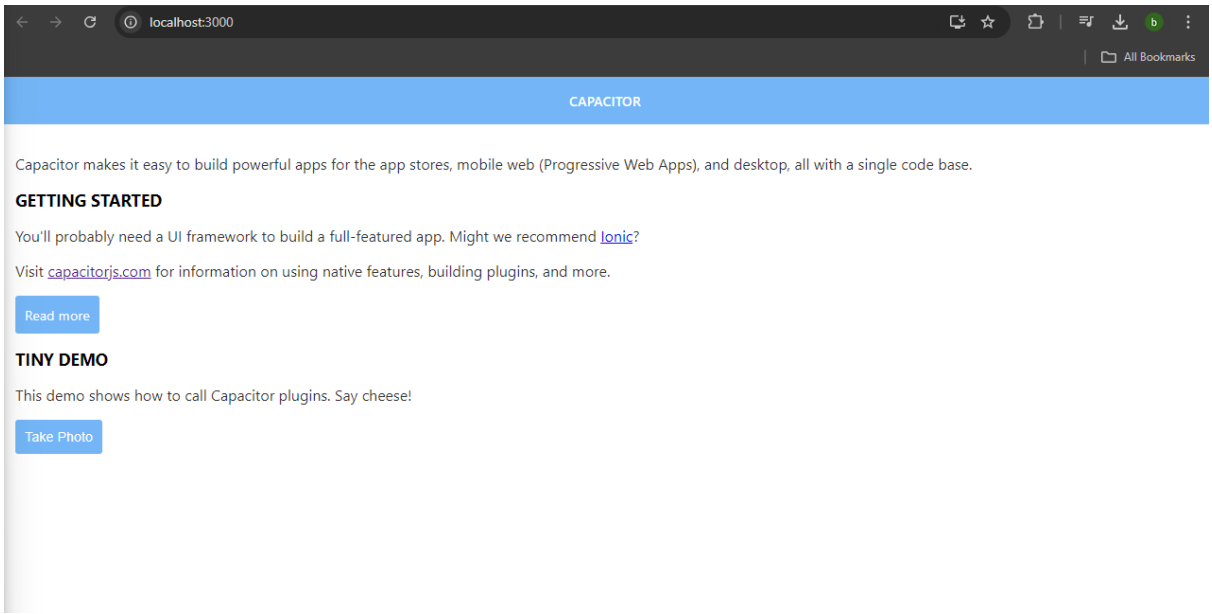
Kao što vidimo, već nam je stvorena template aplikacija, odnosno imamo već stvorene **index.html**, **capacitor-welcome.js**, te nekakve multimedijske sadržaje (**favicon.ico** i **logo.png**)

koji nam se učitavaju na istoj. Unosom naredbe „**npm run start**“ u VSC-ov terminal, dobivamo odgovor, ako je lokalni server pokrenut, te dobivamo link na njega. **Ctrl + lijevim klikom** miša na **http://localhost:3000/ .** [*Slika 22*]



Slika 22: Prvi test

U odabranom pregledniku, otvara nam se template aplikacija sa nekakvim naslovom, tekstom koji nam preporučuje Ionic framework za izradu aplikacija, te nas šalje na **capacitorjs.com** za više informacija, funkcija i gradivnih pluginova. Na navedenu stranicu možemo pristupiti pomoću linka i gumba. Ispod informacija se nalazi demonstracija nativne funkcionalnosti uređaja. Klikom na „**Take photo**“ gumb otvara se kamera. U mom slučaju se vraća crni prozor koji kaže „No camera found“ te se ispod naslova nalazi gumb „Choose image“ nam dopušta da odaberemo sliku s uređaja i ona će se prikazati ispod gumba „**Take photo**“. [*Slika 23*]



Slika 23: Pretpregled template aplikacije

Za brže i lakše oblikovanje izgleda stranice koristit će se Tailwind CSS. On se instalira unosom naredbe **„npm install -D tailwindcss postcss autoprefixer**“ u VSC terminal. Naredbom **„npx tailwind init**“ inicijaliziramo novu komponentu **tailwind.config.js**, odnosno Tailwind konfiguraciju u kojoj samo moramo navesti relativnu putanju do sadržaja unutar “content“ varijable (**[“./src/**/*.{html,js}”]**). [*Kod 1*]

```
/** @type {import('tailwindcss').config} */
module.exports = {
  content: ["./src/**.{html,js}"],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

Kod 1: tailwind.config.js

Naknadno potrebno je stvoriti **postcss.config.js** komponentu u koju dodajemo tailwindcss i autoprefixer koji smo ranije instalirali.

```
module.exports = {
    plugins:  {
        tailwindcss: {},
        autoprefixer: {},
    },
};
```

Kod 2: postcss.config.js

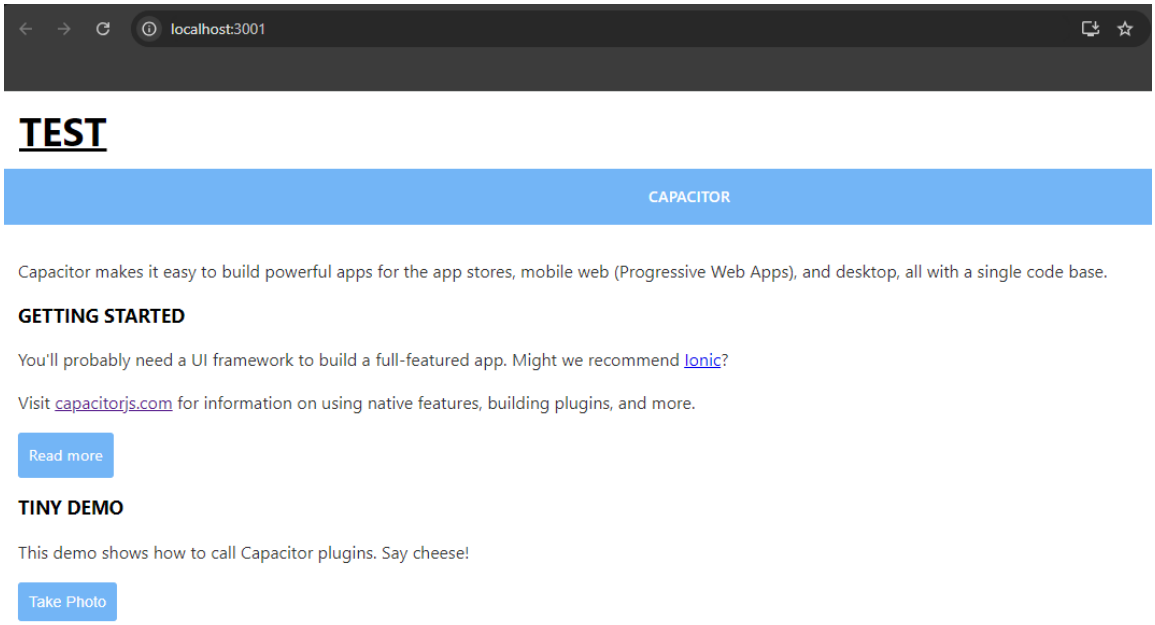
Za završni korak konfiguracije Tailwindcss-a, potrebno je otvriti **style.css** komponentu, te u nju dodati tailwind klase; **@tailwind base**, **@tailwind components** i **@tailwind utilities**.

```
html,
body {
  margin: 0 15%;
}
```

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Kod 3: Tailwind klase u style.css

Kako bi bili sigurni da je Tailwindcss pravilno konfiguriran i instaliran, na vrh stranice dodajem novi testni naslov sa tailwind klasama oblikovanja teksta **<h1 class="m-4 font-bold text-4xl underline">TEST</h1>.** Za prikaz nove verzije aplikacije u web-pregledniku, potrebno je zatvoriti staru, te u VSC terminal opet unesti naredbu "**npm run start**", na što nam se generira novi link; **http://localhost:3001/**. Pojava naslova na [*Slika 24*] sa svim željenim oblikovanjima potvrđuje ispravan rad Tailwindcss-a.



Slika 24: Tailwind test

8.4. Tailwindcss

TailwindCSS je razvio Adam Wathan zajedno s timom suradnika. Adam Wathan je kanadski developer i autor poznat po svom radu u području razvoja softvera i edukaciji programera. TailwindCSS je prvi put objavljen 2017. godine i brzo je stekao popularnost zbog svog inovativnog pristupa stiliziranju web stranica. Adam Wathan i njegov tim kontinuirano rade na unapređenju TailwindCSS-a, redovito izdaju nove verzije i dodatke koji poboljšavaju funkcionalnost i korisničko iskustvo.

Ovo je moderni utility-first CSS framework koji omogućava brzo i efikasno stiliziranje web stranica. Za razliku od tradicionalnih CSS frameworkova koji pružaju unaprijed definirane

komponente, TailwindCSS se fokusira na pružanje malih, izravnih CSS klasa koje se koriste za izgradnju vlastitih komponenti. Ovaj pristup omogućava veću fleksibilnost i kontrolu nad dizajnom, te smanjuje količinu potrebnog prilagođenog CSS koda.

Jedna od glavnih prednosti TailwindCSS-a je njegova brzina razvoja. Programeri mogu brzo primijeniti stilove koristeći utility klase izravno u HTML-u, bez potrebe za pisanjem dodatnog CSS-a. Na primjer, za postavljanje margine, boje pozadine ili veličine fonta, programer može jednostavno dodati odgovarajuću TailwindCSS klasu, kao što su m-4 za marginu, bg-blue-500 za pozadinu ili text-xl za veličinu teksta. Ovakav način rada omogućava brze iteracije i promjene u dizajnu bez puno truda.

TailwindCSS također nudi iznimnu prilagodljivost. Svojstva poput boja, fontova, razmaka i drugih mogu se jednostavno prilagoditi pomoću konfiguracijske datoteke. To znači da se stilovi mogu lako uskladiti s brendom ili specifičnim dizajnerskim zahtjevima projekta. Osim toga, TailwindCSS podržava prilagođene pluginove, što omogućava proširenje funkcionalnosti frameworka prema potrebama projekta.

Jedna od značajnih prednosti TailwindCSS-a je i smanjenje dupliciranja koda. Korištenjem utility-first pristupa, mnogo manje prilagođenog CSS koda je potrebno, što smanjuje mogućnost pojave redundancije. To dovodi do čišćeg i održivijeg koda, koji je lakše razumjeti i održavati. Također, TailwindCSS automatski uklanja neiskorištene stilove prilikom proizvodnje, što smanjuje veličinu CSS datoteka i poboljšava performanse web stranice.

TailwindCSS je vrlo popularan među modernim razvojnim timovima zbog svoje jednostavnosti i učinkovitosti. Pruža sve potrebne alate za brzi razvoj i omogućava programerima da se fokusiraju na funkcionalnost i dizajn bez gubljenja vremena na pisanje prilagođenog CSS-a. Njegova kompatibilnost s različitim alatima za izgradnju i integraciju s popularnim frontend frameworkovima, kao što su React, Vue i Angular, dodatno povećava njegovu vrijednost.

Primjena TailwindCSS-a može značajno ubrzati razvojne procese i poboljšati kvalitetu koda. Mnogi veliki projekti i tvrtke usvojili su TailwindCSS zbog njegovih prednosti u smislu brzine, fleksibilnosti i održavanja. Osim toga, bogata zajednica i odlična dokumentacija pružaju izvrsnu podršku za programere, olakšavajući učenje i implementaciju ovog frameworka.

U zaključku, TailwindCSS predstavlja suvremeno rješenje za stiliziranje web stranica koje kombinira brzinu razvoja, prilagodljivost i učinkovitost. Njegov utility-first pristup donosi značajne prednosti u odnosu na tradicionalne metode stiliziranja, čineći ga iznimno korisnim alatom za moderne web developere. [12]

Slika 25: Tailwindcss logo

8.5. jQuery

jQuery je popularna, brza i jednostavna JavaScript biblioteka koja je osmišljena kako bi olakšala manipulaciju HTML dokumentima, upravljanje događajima, animaciju i interakciju s AJAX tehnologijom. Nastao je 2006. godine, a njegov autor je John Resig. Od tada, jQuery je postao jedna od najkorištenijih JavaScript biblioteka u svijetu web razvoja.

Glavna prednost ove biblioteke je njena jednostavnost. Tradicionalne JavaScript operacije koje zahtijevaju mnogo redova koda u jQueryju se mogu ostvariti s nekoliko linija. Na primjer, dok je u "čistom" JavaScriptu potrebno puno kodiranja za pronalazak i manipulaciju DOM (Document Object Model) elemenata, jQuery to omogućava s jednostavnim selektorima koji nalikuju CSS-u. To omogućuje brže i intuitivnije kodiranje, što je posebno korisno za početnike, ali i za iskusne programere koji žele ubrzati svoj rad.

Jedna od ključnih karakteristika jQueryja je njegova sposobnost kompatibilnosti s različitim preglednicima. U ranijim fazama razvoja weba, programeri su se često susretali s problemima nekompatibilnosti između različitih preglednika poput Internet Explorera, Firefoxa i Safarija. jQuery je značajno pojednostavio taj proces jer skriva te razlike iza svoje API (Application Programming Interface) funkcionalnosti, omogućujući developerima da pišu univerzalni kod koji će jednako raditi na svim preglednicima.

jQuery također omogućava rad s AJAX tehnologijom na vrlo jednostavan način. AJAX (Asynchronous JavaScript and XML) omogućava asinkronu komunikaciju s poslužiteljem bez potrebe za ponovnim učitavanjem cijele stranice. U jQueryju je rad s AJAX-om sveden na jednostavne funkcije, što omogućava lako slanje i primanje podataka u pozadini, poboljšavajući korisničko iskustvo.

Osim toga, jQuery dolazi s velikim brojem gotovih dodataka (pluginova) koji proširuju njegovu funkcionalnost. Postoji ogroman broj jQuery pluginova koji nude rješenja za različite potrebe, kao što su animacije, efekti, validacija obrazaca i još mnogo toga.

Iako su moderni JavaScript okviri poput Reacta, Vue.js-a i Angulara danas popularniji, jQuery se i dalje koristi u mnogim projektima, posebno starijim aplikacijama koje zahtijevaju jednostavne interakcije s DOM-om.

8.6. AJAX

AJAX (Asynchronous JavaScript and XML) je tehnika koja omogućuje web stranicama dinamičnu komunikaciju s poslužiteljem bez potrebe za ponovnim učitavanjem cijele stranice. Ova tehnologija omogućava slanje i primanje podataka u pozadini, što znatno poboljšava korisničko iskustvo i čini web aplikacije interaktivnijima i bržima.

Prije pojave AJAX-a, svaka interakcija korisnika koja je zahtijevala podatke s poslužitelja (kao što je slanje obrasca ili preuzimanje novih podataka) rezultirala bi ponovnim učitavanjem cijele web stranice. S AJAX-om, moguće je slati asinkrone zahtjeve, što znači da stranica može nastaviti funkcionirati dok se zahtjev obrađuje, a podaci s poslužitelja vraćaju bez prekida korisničkog sučelja.

Iako se AJAX izvorno odnosi na razmjenu podataka u XML formatu, danas se često koristi JSON (JavaScript Object Notation) zbog svoje jednostavnosti i lakoće obrade u JavaScriptu.

Korištenje AJAX-a je jednostavno putem JavaScript-a, a biblioteke poput jQuery-ja dodatno olakšavaju njegovu implementaciju. U jQuery-ju, funkcije poput $.ajax(), $.get(), ili $.post() omogućuju slanje zahtjeva i dohvaćanje odgovora s poslužitelja s vrlo malo koda.

Na primjer, AJAX se koristi kada se želi dinamički prikazati lista proizvoda na web stranici bez ponovnog učitavanja stranice, prikazati rezultate pretrage u realnom vremenu ili poslati podatke obrasca i odmah dobiti povratnu informaciju.

AJAX je ključan za izradu modernih web aplikacija jer omogućuje bržu interakciju i smanjuje opterećenje na korisnikov uređaj. Zbog toga je osnova mnogih popularnih web aplikacija i funkcionalnosti kao što su automatsko dovršavanje unosa, dinamično učitavanje podataka ili komentara, te sinkronizacija u stvarnom vremenu.

9. Izrada hibridne mobilne aplikacije u Capacitoru

U ovom dijelu ću pokušati objasniti html i javascript kod koji je potreban za realizaciju praktičnog dijela, te ću u kratkim crtama objasniti klase tailwind css-a na korištenim primjerima.

9.1. „Glava“ html-a

Za početak je specificirano kodiranje znakova za web, u ovom slučaju UTF-8, što omogućava prikaz svih hrvatskih slozvnih znakova. Zatim je određen naslov aplikacije „Kvaliteta zraka“ koji će se prikazivati na kartici preglednika na webu. Sljedeće je definiran format stranice na mobilnim uređajima; stranica zauzima cijeli zaslona, širina je responzivna, zoom je postavljen na 1:1 i postavljeno je ograničenje korisnika da zumira aplikaciju. Zbog sigurnosti, sljedeći metatag sprječava automatsko prepoznavanje telefonskog broja radi zlouporabe.

```
<head>
  <meta charset="UTF-8" />
  <title>Kvaliteta zraka</title>
  <meta name="viewport"
    content="viewport-fit=cover, width=device-width, initial-scale=1.0, minimum-scale=1.0,
maximum-scale=1.0, user-scalable=no" />
  <meta name="format-detection" content="telephone=no" />
  <meta name="msapplication-tap-highlight" content="no" />
```

Kod 4: <head> 1

U sljedećem dijelu je u <style> tagovima definirano polje u kojem će se iscrtavati graf što nam omogućava D3.js biblioteka koja je odmah ispod i uključena. Zatim slijedi uključivanje jquery biblioteka koje su ključne za realizaciju ajaxa, UI komponenti (dijalozi, odabir datuma...) i stiliziranje istih.

```
  <style>
    .line-chart {
      width: 100%;
      height: 500px;
    }
  </style>
  <script src="https://d3js.org/d3.v7.min.js"></script>
```

```
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <script src="https://code.jquery.com/ui/1.12.1/jquery-ui.min.js"></script>
  <link rel="stylesheet" href="https://code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css">
```

Kod 5: <head> 2

Nakon jQuery-a su uključeni Ionic PWA (Progressive Web App) elementi koji omogućavaju interakciju s nativnim značajkama mobilnih uređaja kao što je kamera, GPS i slično, te njihova kompatibilnost s različitim preglednicima.

```
  <script type="module"
    src="https://unpkg.com/@ionic/pwa-
elements@latest/dist/ionicpwaelements/ionicpwaelements.esm.js"></script>
  <script nomodule
    src="https://unpkg.com/@ionic/pwa-
elements@latest/dist/ionicpwaelements/ionicpwaelements.js"></script>
```

Kod 6: <head> 3

Za kraj je uključen manifest za web aplikaciju koji definira naziv, početnu stranicu, ikone i boje, postavljena je tema i dodana je moja JavaScript i CSS datoteka.

```
    <link rel="manifest"
href="data:application/json;base64,ewogICJuYW1lIjogIkFwcCIsCiAgInNob3J0X25hbWUiOiA
iQXBwIiwKICAic3RhcnRfdXJsIjogImluZGV4Lmh0bWwiLAogICJkaXNwbGF5IjogInN0YW
5kYWxvbmUiLAogICJpY29ucyI6IFt7CiAgICAic3JjIjogImFzc2V0cy9pbWdzL2xvZ28ucG5nIi
wKICAgICJzaXplcyI6ICI1MTJ4NTEyIiwKICAgICJ0eXBlIjogImltYWdlL3BuZyIKICB9XSw
KICAiYmFja2dyb3VuZF9jb2xvciI6ICIjMzFkNTNkIiwKICAidGhlbWVfY29sb3IiOiAiIzMxZ
DUzZCIKfQo=" />
  <meta name="theme-color" content="#31d53d" />
  <script type="module" crossorigin src="/assets/index.8dd5c188.js"></script>
  <link rel="stylesheet" href="/assets/index.8724677c.css">
</head>
```

Kod 7: <head> 4

9.2. Dohvaćanje najbliže postaje sa @capacitor/geolocation

```
<h1 class="m-4 font-bold text-4xl text-center">Kvaliteta zraka</h1>
 <div class="mt-auto mb-auto flex flex-row justify-center">
  <button id="saznaj" class="text-center w-full mb-5">Koja mi je najbliža postaja?</button>
 </div>

 <div id="resultModal" class="modal hidden bg-white shadow-md rounded-lg p-0 mb-5 border-2 border-blue-600">
  <div class="modal-content">
   <div class="p-3"><span id="closeModal"
     class="close cursor-pointer pt-1 pb-1 pr-2 pl-2 bg-blue-600 text-white font-semibold py-2 rounded-md hover:bg-blue-700 transition duration-200">&times;</span>
   </div>
   <p id="resultMessage" class="text-center p-3">Najbliža postaja će biti prikazana ovdje.</p>
   <div class="flex justify-center">
    <button id="selectStation"
     class="text-center 2xl:w-1/6 xl:w-1/6 lg:w-1/6 w-2/4 mt-2,5 mb-5 ml-auto mr-auto">Odaberi ovu postaju</button>
   </div>
  </div>
 </div>
```

Kod 8: html 1

Na početku stranice sam stavio njen naslov te ispod nje gumb „Koja mi je najbliža postaja?“ koji za izvršavanje funkcije mora imati **id=“saznaj“**. U kodu ispod se dobro vidi i korištenje tailwind klasa; u naslovu **m-4** predstavlja marginu od 16 px, dakle dodavanje ili smanjivanje broja u tom pravilu mjenja marginu u inkrementu od 4px, ako želimo definirati određenu marginu, koristimo **mt**, **mb**, **ml** i **mr** kao gornju, donju, lijevu i desnu. Ostali razmaci Više ću se fokusirati na apstraktnije klase jer je općenito tailwindcss dosta intuitivan alat. **Text-4xl** je ekvivalent pravila font-size u css-u gdje 4xl označava veličinu od 36 px. Možemo primjetiti da se tailwindcss bazira na pravilima koja se u većini slučajeva povećavaju u nekakvim predodređenim inkrementima. Flexbox se definira samo pravilom **flex**.

Ispod se nalazi inicajlno skrivena sekcija koja sadržava polje za prikaz povratne informacije o najbližoj postaji, gumb za unos iste postaje kao odabrane u formularu i gumb za izlaz. Svaki

klikabilan element koji izvršava nekakvu funkciju ima svoj id za potrebe selektiranja. Gumbi imaju animaciju zatamnjivanja prilikom hoveranja, te se kursor mijenja iz strelice u „pointer“. Ovi stilovi su definirani u posebnoj datoteci „**styles.css**“ u sljedećem obliku:

```
@apply h-20 w-20 bg-blue-600 text-white font-semibold py-2 rounded-md hover:bg-blue-700
transition duration-200;
```

Kod 9: Definiranje tailwindcss pravila u styles.css

Lako je primjetiti da ista pravilu u inline css-u ima gumb za zatvaranje koji je definiran kao **** element. Razlog tome je što jQuery koristi spam elemente u svom modulu datepicker-a koji će kasnije biti potreban i kada bi pravilo bilo definirano preko span selektora, taj bi se modul raspao.

Kako bi geolokacija funkcionirala, potrebno je u terminalu pokrenuti naredbu „**npm install @capacitor/geolocation**“ za instalaciju ovog plugina. Zatim u posebnoj datoteci „**geolokacija.js**“ treba importirati željenu funkcionalnost iz Capacitorove biblioteke na sljedeći način:

```
import { Geolocation } from '@capacitor/geolocation';
```

Kod 10: Importiranje geolokacije u js

Kako bi skripta mogla izračunavati udaljenost najbliže postaje za mjerenje zraka, prema podacima Informacijskog sustava zaštite zraka [16], stvoren je objekt:

```
const postaje = {

   14: { name: "Zajci", lat: 45.204147, lon: 14.070761, polutanti: ["5", "2", "4", "3"] },
   20: { name: "Čambarelići", lat: 45.181331, lon: 14.101011, polutanti: ["5", "2", "4"] },
   37: { name: "Zoljan", lat: 45.536200, lon: 17.607800, polutanti: ["1", "5", "2"]},
   38: { name: "Vrhovec", lat: 45.818300, lon: 15.947800, polutanti: ["1", "38"] },
   39: { name: "Pula Fižela", lat: 44.866000, lon: 13.852500, polutanti: ["1", "38", "31", "5", "2",
"3"] },
}
```

Kod 11: Objekt sa svim postajama

Za potrebe primjera, naveo sam prvih 5 postaja od ukupno njih 70. Id svake postaje je jednak vrijednosti opcije u formularu koji ću opisivati niže, a nakon slijedi njeno ime, te geografska dužina, širina i vrijednost opcije svih polutanata koje mjeri (također vezano na formular niže).

Prva funkcija u **geolokacija.js** služi za izračunavanje udaljenosti između dvije točke na Zemljinoj površini koristeći njihove geografske dužine i širine Haversinevom formulom koja u obzir uzima i zakrivljenost zemlje, te vraća udaljenost u kilometrima.

```
const getDistance = (lat1, lon1, lat2, lon2) => {
  const R = 6371; // Radijus Zemlje u kilometrima
  const dLat = (lat2 - lat1) * (Math.PI / 180); // Razlika u geografskim širinama (pretvoreno u radijane)
  const dLon = (lon2 - lon1) * (Math.PI / 180); // Razlika u geografskim dužinama (pretvoreno u radijane)

  // Haversineova formula
  const a = Math.sin(dLat / 2) * Math.sin(dLat / 2) +
    Math.cos(lat1 * (Math.PI / 180)) * Math.cos(lat2 * (Math.PI / 180)) *
    Math.sin(dLon / 2) * Math.sin(dLon / 2);

  const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

  return R * c; // Vraćanje udaljenosti u kilometrima
};
```

Kod 12: getDistance

Nakon toga slijedi funkcija koja koristi lokaciju uređaja dohvaćenu Capacitorovim geolocation API-jem i prolazi kroz sve postaje iz gornjeg objekta i izračunava njihovu udaljenost. Dok pronađe najbližu, prikazuje povratnu infromaciju u inicijalno skrivenom modalu i daje opciju korisniku da ju odabere.

```
const findNearestStation = async () => {
  try {
    const { coords } = await Geolocation.getCurrentPosition(); // Dohvaćanje trenutne lokacije korisnika
    const userLat = coords.latitude;
```

```
    const userLon = coords.longitude;

    let nearestStation = null;
    let minDistance = Infinity; // Postavljanje početne vrijednosti minimalne udaljenosti na beskonačno

    // Iščitavanje svih postaja po redu
    for (const [id, postaja] of Object.entries(postaje)) {
      const distance = getDistance(userLat, userLon, postaja.lat, postaja.lon); // Izračunavanje udaljenosti do svake postaje
      if (distance < minDistance) { // Ako je udaljenost manja od minimalne udaljenosti
        minDistance = distance; // Postavljanje nove minimalne udaljenosti
        nearestStation = { id, ...postaja }; // Postavljanje najbliže postaje
      }
    }

    // Prikaz rezultata u modalu
    const resultMessage = nearestStation
      ? `Najbliža postaja je ${nearestStation.name} koja se nalazi ${minDistance.toFixed(2)} km od vas.`
      : 'Nema dostupnih postaja.';

    document.getElementById('resultMessage').innerText = resultMessage; // Postavljanje teksta rezultata
    document.getElementById('resultModal').style.display = 'block'; // Prikaz modal s rezultatom

    // Postavlja događaj za odabir najbliže postaje
    document.getElementById('selectStation').onclick = () => {
      selectStation(nearestStation.id); // Odabir najbliže postaje
      document.getElementById('resultModal').style.display = 'none'; // Zatvara modal
    };

  } catch (error) {
    console.error('Greška pri dohvaćanju lokacije:', error); // Ako postoji greška, ispisuje je
    alert('Došlo je do pogreške pri dohvaćanju lokacije.'); // Prikaz greške korisniku
```

```
  }
};
```

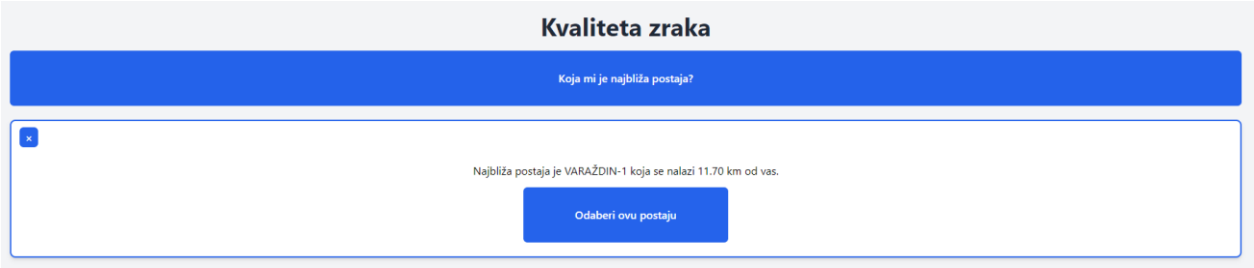
Kod 13: findNearestStation

Sada dok su sve funkcije navedene, potrebno je dodati događaje koji će ih pozivati; likom na gumb „Koja mi je najbliža postaja?“, pokreće se funkcija *findNearestStation*, klikom na „Odaberi ovu postaju“ postavljanja se vrijednosti izbornika u vraćenu najbližu postaju i klikom na gumb za izlaz se prozor s najbližom postajom zatvara.

```
const selectStation = (stationId) => {
  const stationSelect = document.getElementById('postaja');
  stationSelect.value = stationId;
};

document.getElementById('saznaj').addEventListener('click', findNearestStation);

document.querySelector('.close').addEventListener('click', () => {
  document.getElementById('resultModal').style.display = 'none';
});
```

Kod 14: Pozivanje funkcija u geolokacija.js

Opisani dijelovi aplikacije u pregledniku izgledaju kao na [*Slika 26*]. Kao što je već više puta spomenuto, modal pri učitavanju stranice nije vidljiv.



Slika 26: App 1

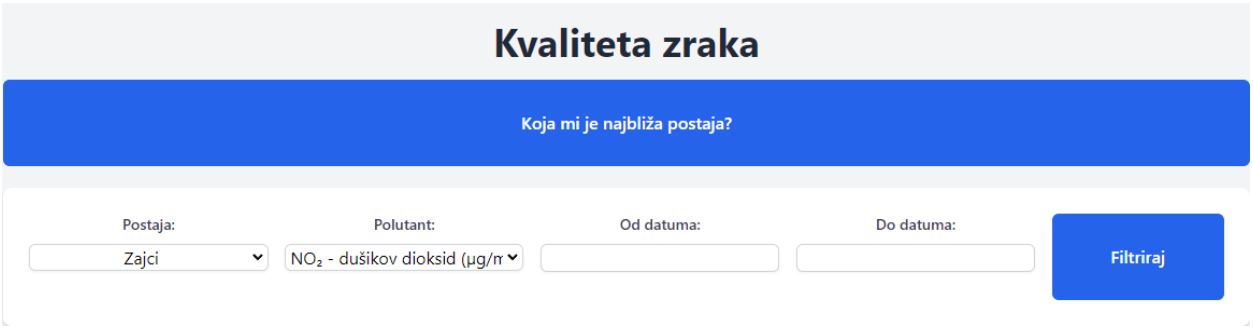
9.3. Primjena AJAX-a i jQuery-a

Za izvlačenje podataka o kvaliteti zraka iz Informacijskog sustava zaštite zraka potreban je formular koji korisniku na izbor daje unos samo dostupnih postaja i polutanata iz istog. Prema tome sam napravio listu 70 postaja koje rade danas i 445 polutanata koje mjere do kojih sam došao preko Informacijskog sustava zaštite zraka [16]. Kao primjer sam ostavio selekciju samo prvih pet za oba izbornika. Unos postaje može biti izabran dropdown izbornikom ili gumbom za odabir najbliže postaje. Korisnik također bira vremenski raspon mjerenja, te kad je zadovoljan unesenim podacima može kliknuti gumb „Filtriraj“. [Slika 27]

```
<div class="formular">
  <div class="formular_input">
   <label for="postaja">Postaja:</label>
   <select id="postaja" name="postaja">
     <option value="14">Zajci</option>
     <option value="20">Čambarelići</option>
     <option value="37">Zoljan</option>
     <option value="38">Vrhovec</option>
     <option value="39">Pula Fižela</option>
      <!-- I tako dalje -->
   </select>
  </div>
  <div class="formular_input">
   <div>
     <label for="polutant">Polutant:</label>
   </div>
   <div>
     <select id="polutant" name="polutant">
       <option value="1">NO₂ - dušikov dioksid (µg/m3)</option>
       <option value="2">SO₂ - sumporov dioksid (µg/m3)</option>
       <option value="3">CO - ugljikov monoksid (mg/m3)</option>
       <option value="4">H₂S - sumporovodik (µg/m3)</option>
       <option value="5">PM₁₀ - lebdeće čestice (&lt;10µm) (µg/m3)</option>
      <!-- I tako dalje –->
     </select>
```

```
    </div>


  </div>
  <div class="formular_input">
    <label for="datumOd">Od datuma:</label>
    <input type="text" id="datumOd" name="datumOd">
  </div>
  <div class="formular_input">
    <label for="datumDo">Do datuma:</label>
    <input type="text" id="datumDo" name="datumDo">
  </div>
  <div class="mt-auto mb-auto flex flex-row justify-around">
    <button id="filtriraj" class="text-center w-full 2xl:w-40">Filtriraj</button>
  </div>
</div>
```

Kod 15: Formular



Slika 27: App 2

Ispod formulara se nalazi tablica za prikaz dohvaćenih podataka i dodana su joj tailwind ograničenja maksimalne visine od 400px i ako podaci ne stanu u taj prostor, da se ona može pomicati klizačem. Tablica ima 3 stupca koja prikazuju vrijeme mjerenja, izmjerenju vrijednost odabranog polutanta te njegovu mjernu jedinicu. Ispod nje se nalazi polje koje će koristiti za crtanje grafa iz prikazanih podataka u tablici. Ovi elementi su pri učitavanju stranice prazni, pa nema smisla da se prikazuju i stoga su skriveni.

```
<div id="dataContainer" class="hidden">
  <div class="table-container">
```

```
    <h2 class="text-center">Tablica podataka</h2>
    <div class="overflow-auto max-h-[400px]">
      <table id="podaci" class="min-w-full border border-gray-300">
        <thead class="table-header">
          <tr>
            <th class="text-center">Vrijeme</th>
            <th class="text-center">Vrijednost</th>
            <th class="text-center">Mjerna jedinica</th>
          </tr>
        </thead>
        <tbody class="table-body">

        </tbody>
      </table>
    </div>
  </div>
  <div class="overflow-x-auto">
    <div id="lineGraph"
      class="mt-4 h-60 bg-gray-200 border border-gray-300 rounded-lg flex justify-center items-
center mb-10 min-w-[1000px]">
    </div>
  </div>
 </div>
```

Kod 16: Tablica i polje grafa

Željenim podacima može pristupiti bilo tko tako da upiše u tražilicu poveznicu koje izgleda slično ovome:

http://iszz.azo.hr/iskzl/rs/podatak/export/json?postaja=173&polutant=1&tipPodatka=0&vrijemeOd=11.08.2016&vrijemeDo=25.08.2016

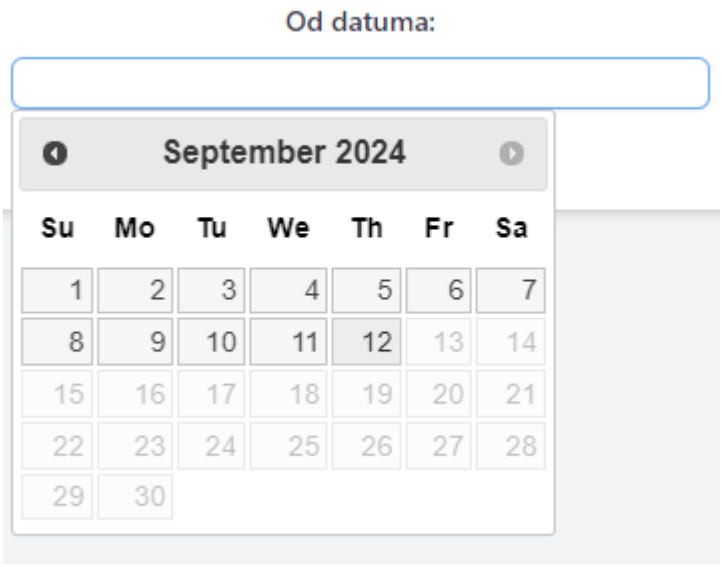
Iz poveznice možemo iščitati da se radi o postaji 173, a to je prema ISZZ-u AMS1 Kaštel Sućurac, mjereni polutant je 1, odnosno NO_2 - dušikov dioksid (µg/m3) i razdoblju od 11.08.2016. do 25.8.2016. Tip podataka je json jer je ta vrijednost 0, kad bi bila 1, radilo bi se o xml-u.

Zbog prioriteta izvršavanja JavaScript koda, funkcije koje se odnose na čitanje i prikaz izvađenih podataka iz ISZZ-a stavio sam u html dokumentu unutar <script> tagova te ih obgrlio document ready funkcijom koja osigurava da se kod može izvršavati tek kad je html document

učitan i spreman za interakciju. Za korištenje jQuery datepickera, potrebna je funkcija koja će korisniku omogući bilo koji prošli datum za datum od i do sa ograničenjem da najviši mogući bude današnji.

```
$(document).ready(function () {

    $('#datumOd').datepicker({
      dateFormat: 'dd.mm.yy',
      minDate: null,
      maxDate: new Date()
    });

    $('#datumDo').datepicker({
      dateFormat: 'dd.mm.yy',
      minDate: null,
      maxDate: new Date()
    });
```

Kod 17: Datepicker



Slika 28: Datepicker

Funkcija u html-u dohvaća vrijednosti odabranih u formi i kreira url koji se može popunjavati po gore navedenom principu. Nakon inicijalizacije varijabli, brišu se prethodni redovi iz tablice i prethodni grafikon ako ih ima. Zatim se AJAX-om šalje zahtjev na generirani url i očekuje se

odgovor u json format. Ako je zahtjev uspješan, pokreće se funkcija koja stvara prazno polje gdje će se spremati podaci za graf i iterira kroz svaki redak odgovora. Prema iščitanim podacima, za svaki se stvara redak u tablici i podaci se dodaju u nju. Podaci se također dodaju u prazno polje za crtanje grafa i pretvaraju u oblik s kojim funkcije ds3.js biblioteke mogu baratati. Ako ima podataka, miče se hidden pravilo sa tablice i grafa da mogu biti vidljivi. Naravno, funkcija ne bi bila potpuna bez povratne informacije o pogrešci, ukoliko je ima i definiranju poziva klikom na gumb “Filtriraj”

```
    function fetchData() {
      var postaja = $('#postaja').val();
      var polutant = $('#polutant').val();
      var tipPodatka = '0';
      var datumOd = $('#datumOd').val();
      var datumDo = $('#datumDo').val();


      var url =
`https://iszz.azo.hr/iskzl/rs/podatak/export/json?postaja=${postaja}&polutant=${polutant}&tipP
odatka=${tipPodatka}&vrijemeOd=${datumOd}&vrijemeDo=${datumDo}`;


      $('#podaci tbody').empty();
      d3.select("#lineGraph").selectAll("*").remove();

      $.ajax({
        url: url,
        dataType: 'json',
        cache: false,
        success: function (data) {
          const lineData = [];

          $.each(data, function (i, item) {
            var row = '<tr>' +
              '<td>' + formatDate(new Date(item.vrijeme)) + '</td>' +
              '<td>' + item.vrijednost + '</td>' +
              '<td>' + item.mjernaJedinica + '</td>' +
```

```
            '</tr>';
          $('#podaci tbody').append(row);


          lineData.push({
            value: parseFloat(item.vrijednost),
            date: new Date(item.vrijeme)
          });
        });


        if (lineData.length > 0) {
          drawLineGraph(lineData);
          $('#dataContainer').removeClass('hidden');
        } else {
          d3.select("#lineGraph").selectAll("*").remove();
          $('#dataContainer').addClass('hidden');
        }
      },
      error: function () {
        alert('Došlo je do pogreške prilikom učitavanja podataka');
      }
    });
  }
$('#filtriraj').click(fetchData);
```

Kod 18: fetchData

Nakon ove funkcije slijedi formatiranje datuma koji će se prikazivati u grafu u oblik pogodan za hrvatsko tržište. Redom se dohvaćaju dani, mjeseci, sati i minute, pa se vraća formatirani.

```
function formatDate(date) {
    const day = String(date.getDate()).padStart(2, '0');
    const month = String(date.getMonth() + 1).padStart(2, '0');
    const year = date.getFullYear();
    const hours = String(date.getHours()).padStart(2, '0');
    const minutes = String(date.getMinutes()).padStart(2, '0');
```

```
  return `${day}.${month}.${year} ${hours}:${minutes}`;
}
```

Kod 19: formatDate

Kako bi se upotpunio prikaz podataka, dodao sam funkciju koja u zadano polje crta graf sa trendom podataka. Započinje tako da se postojeći graf, ako ga ima, uklanja nakon čega se definiraju margine i dimenzije grafa. Zatim se stvara SVG element unutar polja u html-u i postavlja mu se ukupna širina, visina i dodaj se se grupni element za manipulaciju unutarnjeg dijela grafa nakon čega se graf pomiče unutar margina. Korištenjem vremenske skale definirana je X os koja se proteže po širini dok je Y os definirana linearnom i proteže se po visini. Definirana je i linija koja povezuje točke na grafu tako da joj X koordinata ovisi o datumu a Y o vrijednosti. Crtanje grafa se vrši tako da se podaci prvo vežu, pa se crta linija koja se kasnije ispunjava bojom i dodaje joj se nekakva debljina. Oblik linije se definira prema funkciji line. Na kraju se na dno grafa crta X os i Y os na lijevi rub sa pripadajućim oznakama.

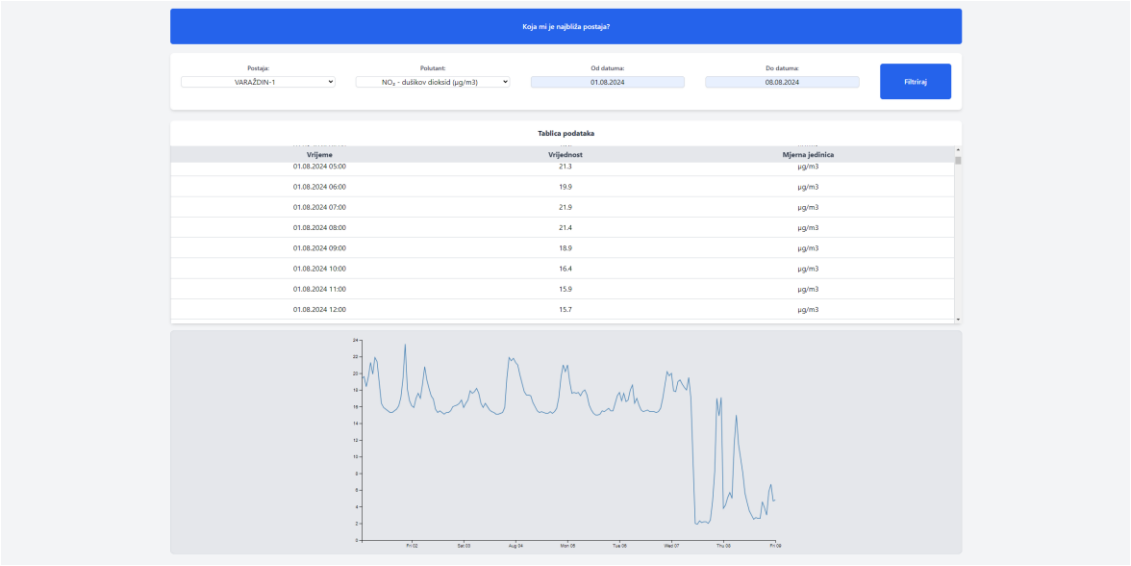
```
function drawLineGraph(data) {
    d3.select("#lineGraph").selectAll("*").remove();

    const margin = { top: 20, right: 30, bottom: 30, left: 40 };
    const width = 1000 - margin.left - margin.right;
    const height = 500 - margin.top - margin.bottom;

    const svg = d3.select("#lineGraph")
      .append("svg")
      .attr("width", width + margin.left + margin.right)
      .attr("height", height + margin.top + margin.bottom)
      .append("g")
      .attr("transform", `translate(${margin.left},${margin.top})`);

    const x = d3.scaleTime()
      .domain(d3.extent(data, d => d.date))
      .range([0, width]);

    const y = d3.scaleLinear()
      .domain([0, d3.max(data, d => d.value)]).nice()
```

```
    .range([height, 0]);

  const line = d3.line()
    .x(d => x(d.date))
    .y(d => y(d.value));

  svg.append("path")
    .data([data])
    .attr("fill", "none")
    .attr("stroke", "steelblue")
    .attr("stroke-width", 1.5)
    .attr("d", line);

  svg.append("g")
    .attr("transform", `translate(0,${height})`)
    .call(d3.axisBottom(x));

  svg.append("g")
    .call(d3.axisLeft(y));
}
```

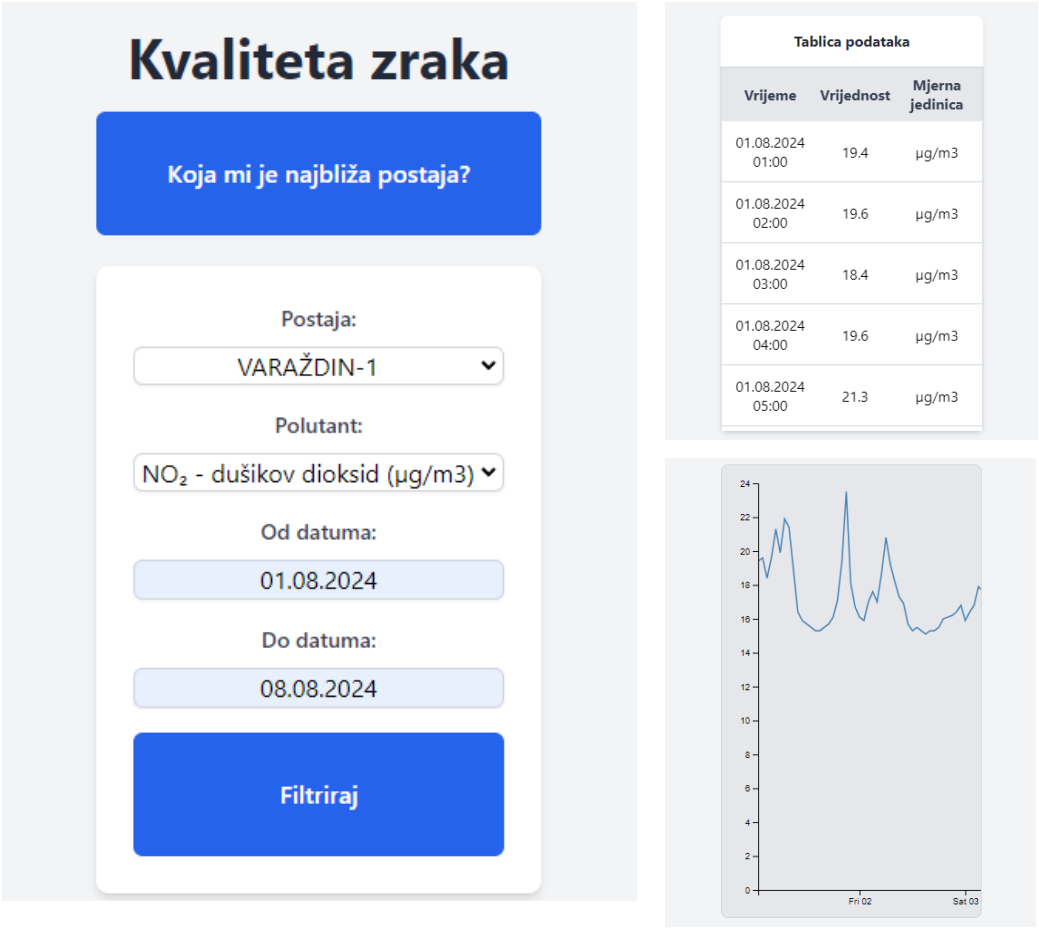
Kod 20: drawLineGraph

9.4. Testiranje i izgled web aplikacije

Klikom na gumb „Koja mi je najbliža postaja?“ otvara se prozor koji mi daje do znanja da je to Varaždin-1 i da se nalazi 11.7 km od mene [*Slika 26*]. Klikom na gumb „Odaberi ovu postaju“ ona se odabire automatski sa padajućeg izbornika i prozor sa povratnom informacijom se zatvara. Odabirem polutant NO_2 - dušikov dioksid (µg/m3) vremenski raspon 01.08.2024 - 03.08.2024. Klikom na gumb filtriraj, prikazuje se tablica svih mjerenih podataka u tom vremenskom rasponu i pripadajući graf. [*Slika 29*] U pregledniku sam također testirao responzivnost na formatu Samsung Galaxy S20 Ultra te se tablica i graf mogu klizati u smjeru prikaza podataka ukoliko ne stanu u definirano polje prikaza. Aplikacija je u potpunosti responzivna i prilagođena za više uređaja različitih veličina.



Slika 29: Testiranje web aplikacije



Slika 30: Testiranje mobilne web aplikacije

Responzivnost u tailwindu se uređuje dodavanjem npr.: xl:pravilo i ima predodređene generične veličine zaslona. Stiliziranje index.html-a napravio sam u styles.css datoteci definiranjem sljedećih tailwind pravila:

```
html,
body {
  margin: 0 15%;
}

@tailwind base;
@tailwind components;
@tailwind utilities;

body {
  @apply bg-gray-100 font-sans;
}

h1 {
  @apply text-4xl font-bold text-gray-800 text-center my-4;
}

.formular {
  @apply flex 2xl:flex-row flex-col justify-between text-center mx-auto p-6 bg-white shadow-md
rounded-lg;
}

.formular_input {
  @apply flex flex-col mb-4 2xl:xl:w-1/5;
}

label {
  @apply block text-sm font-medium text-gray-700 mb-1;
}

select,
input {
  @apply mt-1 block w-full border border-gray-300 rounded-md shadow-sm focus:ring
focus:ring-blue-300 focus:border-blue-500;
```

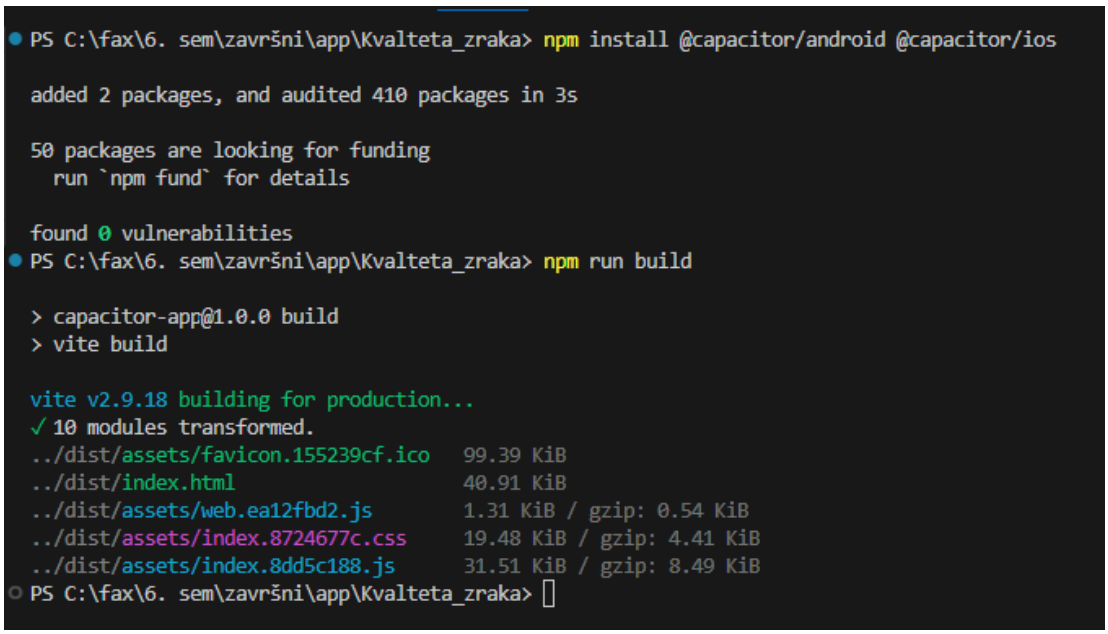
```
}

option:hover,
input:hover {
  @apply border-blue-400;
}

button {
  @apply h-20 w-20 bg-blue-600 text-white font-semibold py-2 rounded-md hover:bg-blue-700
transition duration-200;
}


.table-container {
  @apply bg-white shadow-md rounded-lg p-0 mt-6 ;
}

.table-header {
  @apply bg-gray-200 sticky top-0;
}

.table-header th {
  @apply py-2 text-gray-700 border-b;
}

.table-body {
  @apply bg-white;
}

.table-body td {
  @apply py-3 text-center border-b border-gray-300;
}

#podaci {
  @apply min-w-full;
```

```
}

#lineGraph {
  @apply w-full bg-gray-200 border border-gray-300 rounded-lg flex justify-center items-center
mb-10;
}

h2 {
  @apply text-center font-bold text-gray-800 pt-4 mb-4;
}

svg {
  @apply min-h-fit;
}

select, input {
  @apply text-center;
}

#modal {
  @apply flex 2xl:flex-row flex-col justify-between text-center mx-auto p-6 bg-white shadow-md
rounded-lg;
}
```

Kod 21: styles.css

9.5. Gradnja izvorne aplikacije za android

Da bismo započeli gradnju izvorne aplikacije za android, potrebno je u terminalu pokrenuti naredbu „**npm install @capacitor/android @capacitor/ios**“ koja nam instalira sav nužan kod vezan za platformu, biblioteke i konfiguracije. Uključio sam i potporu za Apple IOS iako neću razvijati aplikaciju za taj sustav, no dobro je imati mogućnost. Nakon toga unosim naredbu „**npm run build**“ koji će u capacitor.config.json datoteci pronaći direktorij ("webDir": "dist") u koji će instalirati aplikaciju.



Slika 31: Gradnja aplikacije

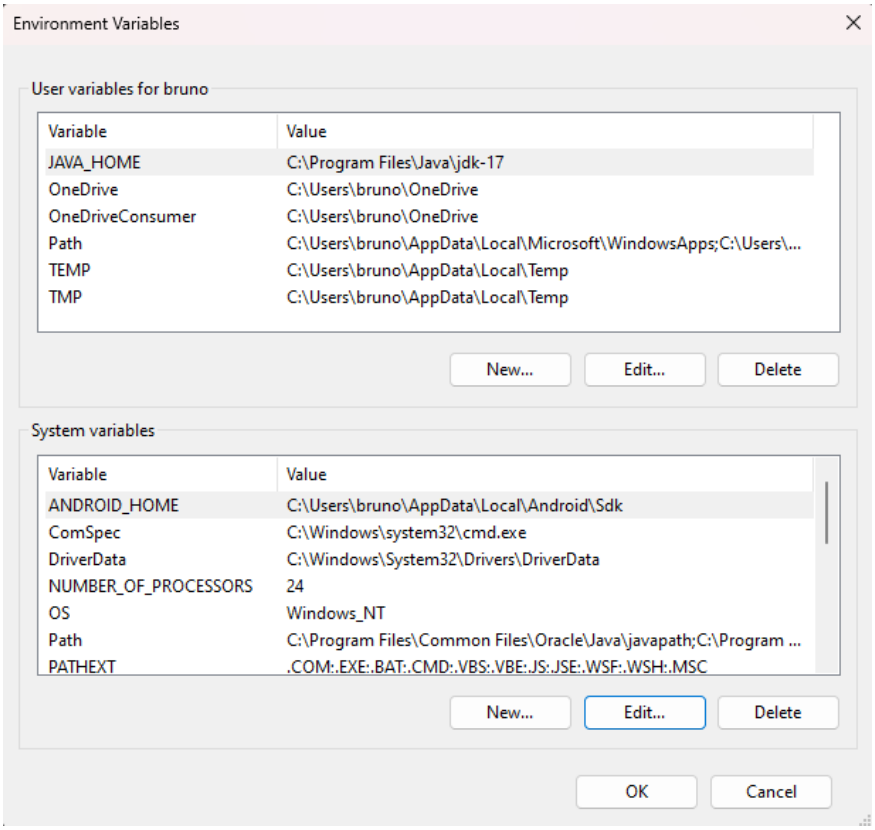
```
{
 "appId": "com.bruno.kvaliteta_zraka",
 "appName": "Kvaliteta_zraka",
 "webDir": "dist",
 "plugins": {
  "SplashScreen": {
   "launchShowDuration": 0
  }
 }
}
```

Kod 22: capacitor.config.json

Nakon toga u terminal unosi se naredbu „**npx cap add android**“ za dodavanje androida kao ciljanu platformu. Capacitor stvara novi Android projekt u android/ direktoriju unutar datoteke projekta. Ovo omogućava Capacitoru komunikaciju s Android platformom. Nakon toga može se otvoriti generiran Android projekt u Android Studiu i razviti ga na emulatoru ili pravom uređaju. Za potrebe ovog rada koristit će se emulator. Nakon bilo kakve izmjene u aplikaciji sad možemo unesti u terminal naredbe „**npm run build**“ i nakon toga „**npx cap sync**“ ili „**npx cap copy**“. Sync je malo sporiji ali sigurniji, pa se u glavnom preporuča korištenje te naredbe. Unosom naredbe „**npx cap run android**“ terminal će izbaciti poruku (u ovom slučaju) ukoliko nije instaliran

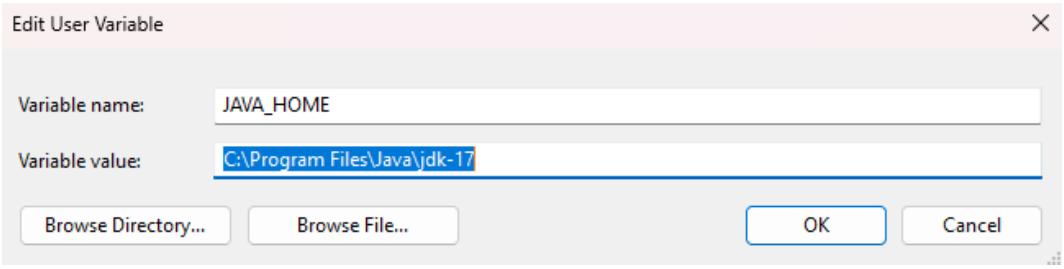
Android Studio (SDK), Java Development Kit (JDK) i njihove „Environment“ variable. Također je važno napomenuti da je za rad u Capacitoru potrebno instalirati JDK 17.

Varijable koje je potrebno dodati nakon instalacije SDK i JDK su JAVA_HOME i ANDROID_HOME. Putanja do željenog odredišta je: Desni klik na This PC > Advanced System Settings > Environment Variables.

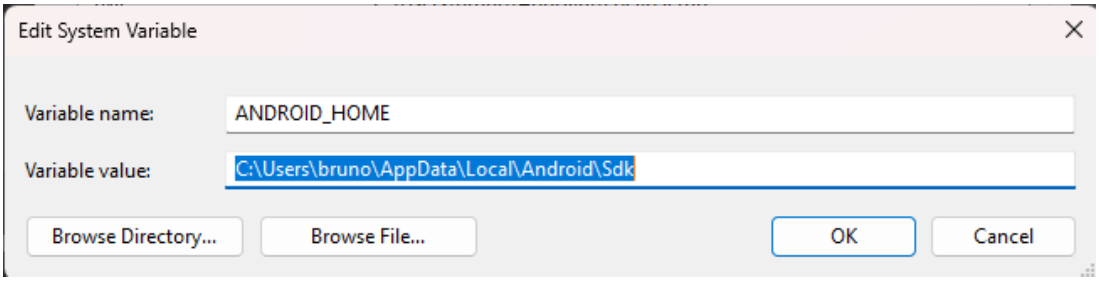


Slika 32: Environment Variables

Funkcija u html-u dohvaća vrijednosti odabranih u formi i kreira url koji se može popunjavati po gore navedenom principu. Nakon inicijalizacije varijabli, brišu se prethodni redovi iz tablice i prethodni grafikon ako ih ima. Zatim se AJAX-om šalje zahtjev na generirani url i očekuje seJAVA_HOME dodaje se kao „User variable“ klikom na gumb“ New“, a ANDROID_HOME kao „System variable“.

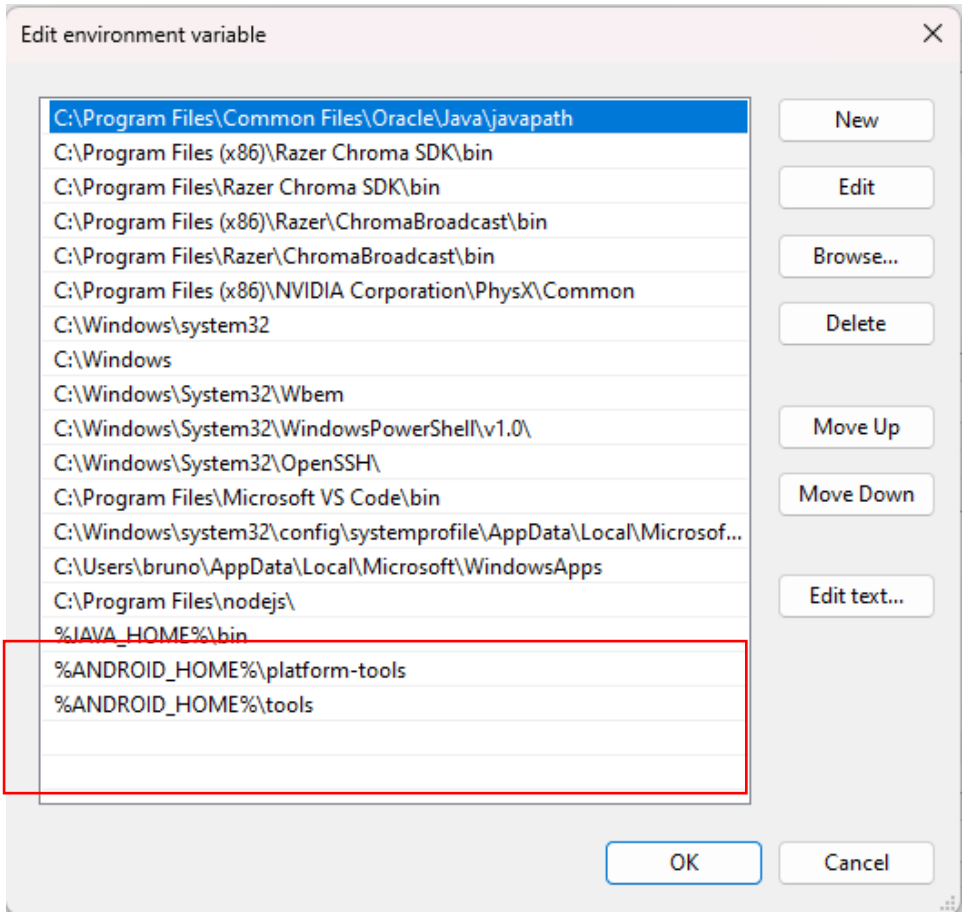


Slika 33: JAVA_HOME



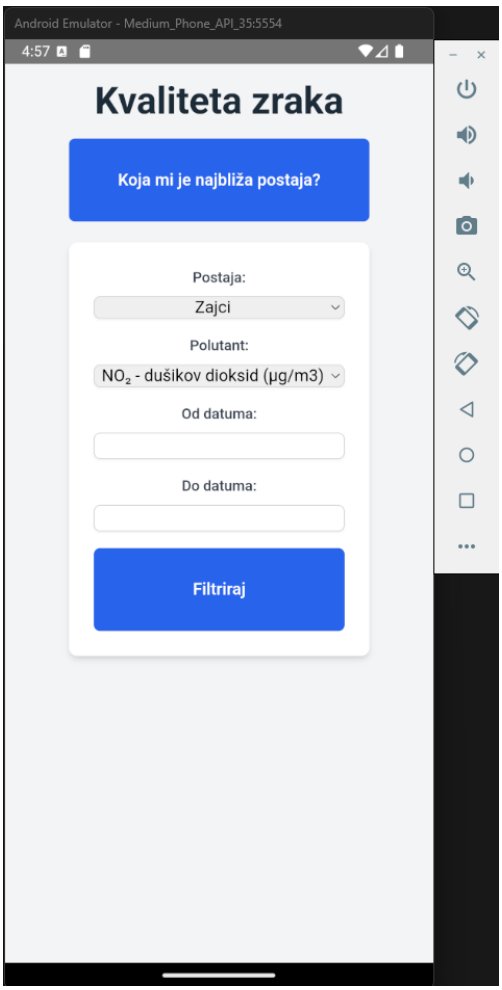
Slika 34: ANDROID_HOME

Nakon dodavanja varijabli, potrebno je odabrati sistemsku varijablu „Path“ i klikom na gumb Edit urediti varijablu tako što dodamo 3 retka kao na slici:



Slika 35: Path

Sada je osigurana pravilna podrška i okolina za građenje aplikacije, pa se u VSC terminal može unesti naredba „**npx cap run android**“ i to će pokrenuti emulator android uređaja. Emulator ima mnoge postavke kojima možemo upravljati kao veličina uređaja, lokacija, fingerprint senzor, kamera u kojoj možemo dodavati slike i slično.



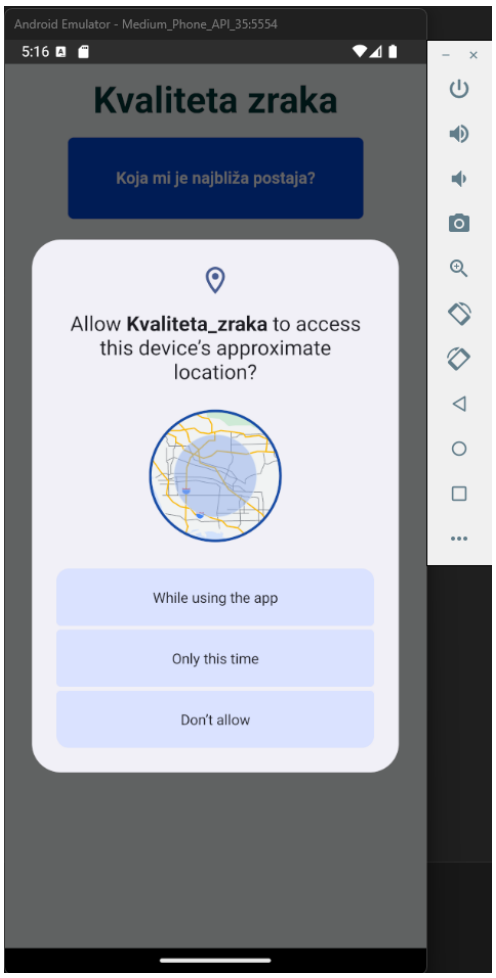
Slika 36: Emulator

Sad se javlja problem sa funkcionalnošću geolokacije, pošto je to plugin Capacitora. Klikom na gumb za prikaz najbliže postaje, aplikacija vraća poruku o grešci. Kako bi se to popravilo potrebno je dodati dopuštenja na android > app > src > main > AndroidManifest.xml u VSC direktoriju ispod komentra „<!-- Permissions -->“:

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-feature android:name="android.hardware.location.gps" />
```

Kod 23: Dodaci u AndroidManifest.xml

Prva dva dopuštenja traže i fine i grube lokalne podatke, dok treće koje je opcionalno, ali u nužno ako aplikacija koristi GPS za funkciju. Kao što je prije spomenuto, potrebnoo je izvršiti naredbe „**npm run build**“, „**npx cap sync**“ i „**npx cap run android**“



Slika 37: Dopuštenje

Klikom na gumb za prikaz najbliže postaje, sada se otvara prozor za korisničko dopuštenje da aplikacija koristi nativnu funkciju geolokacije uređaja. Odabirom jedne od prvih dviju opcija, gumb je sada funkcionalan. Naredobom „**npx cap open android**“ aplikacija se otvara u Androd studiu gdje se preko wifi mreže ili usb kabla može spojiti fizički uređaj za testiranje aplikacije. Ova opcija se čini jako korisna, no na žalost spojeni uređaj mora imati Android 11 verziju operacijskog sustava, što sprečava u ovom slučaju istraživanje navedene mogućnosti.

9.6. Zaključak praktičnog dijela

Praktični dio rada bavio se izradom aplikacije za prikaz mjerenja kvalitete zraka u Hrvatskoj. Aplikacija koristi javno dostupne podatke o kvaliteti zraka, koje dohvaća putem API-ja i prikazuje ih korisnicima na jednostavan i intuitivan način. Postavke radnog okruženja su uključivale instalaciju Capacitora, Node.js-a te drugih potrebnih alata za razvoj. Nakon inicijalizacije projekta, aplikacija je razvijena koristeći HTML, CSS i JavaScript, uz značajnu upotrebu TailwindCSS-a za stiliziranje korisničkog sučelja.

Jedan od ključnih aspekata aplikacije bio je korištenje geolokacijskih podataka. Uz pomoć @capacitor/geolocation plugina, aplikacija je mogla dohvatiti korisnikovu trenutnu lokaciju i na temelju toga izračunati najbližu mjernu postaju za kvalitetu zraka. Ovaj podatak korisniku je prikazan na sučelju, a zatim se, putem API-ja, dohvaćaju podaci o kvaliteti zraka na toj lokaciji. Integracija geolokacijskih funkcija omogućila je personalizirano korisničko iskustvo, jer svaki korisnik može brzo dobiti informaciju o kvaliteti zraka u svom trenutnom okruženju.

Za prikaz prikupljenih podataka korišten je D3.js za grafičku vizualizaciju, što je omogućilo prikaz dinamičkih grafova o razinama različitih tvari u zraku. Ovi grafovi su bili ključni za prikaz složenih podataka na razumljiv i pregledan način. Korištenje grafova značajno je pridonijelo boljem razumijevanju informacija, jer korisnici mogu uočiti promjene u kvaliteti zraka u realnom vremenu.

Korištenjem jQuery-a i AJAX-a, aplikacija je omogućila dinamičko dohvaćanje podataka bez potrebe za ponovnim učitavanjem stranice. To je značajno poboljšalo korisničko iskustvo jer su informacije o kvaliteti zraka bile dostupne u stvarnom vremenu. Na taj način korisnik može brzo dobiti informacije i reagirati na promjene u kvaliteti zraka, što aplikaciju čini korisnom za svakodnevnu upotrebu.

TailwindCSS je korišten za vizualno oblikovanje aplikacije, što je omogućilo brzu izradu responzivnog sučelja prilagođenog različitim veličinama ekrana i uređajima. Ovaj framework omogućio je da aplikacija izgleda profesionalno i moderno, uz minimalno pisanje prilagođenog CSS-a. Osim što je ubrzao proces razvoja, TailwindCSS je omogućio i lakše održavanje dizajna aplikacije.

10. Zaključak

U ovom završnom radu obrađena je tema izrade hibridne mobilne aplikacije pomoću Capacitora, s naglaskom na konkretan primjer aplikacije za prikaz podataka o kvaliteti zraka u Hrvatskoj. Kroz rad je prikazan cijeli proces razvoja, počevši od postavljanja okruženja, preko tehničkih izazova, sve do konačne implementacije i testiranja aplikacije. Ovdje donosimo pregled ključnih spoznaja prema strukturi rada.

Zbog svoje multiplatformske prirode, hibridne aplikacije omogućuju brži razvoj, smanjuju troškove te omogućuju jednom razvijenoj aplikaciji da se koristi na različitim platformama. Cilj rada bio je pokazati prednosti i mogućnosti koje Capacitor, kao moderni alat za razvoj hibridnih aplikacija, pruža kroz razvoj konkretne aplikacije.

U odnosu na alternative, kao što su Apache Cordova, React Native i Flutter, Capacitor se pokazao kao fleksibilno i jednostavno rješenje koje omogućuje brzi prijelaz web developera u svijet mobilnog razvoja, bez potrebe za učenjem novih jezika poput Jave ili Swifta. Iako druge platforme nude određene prednosti, Capacitor je zadržao balans između jednostavnosti i funkcionalnosti, što ga čini idealnim alatom za izradu hibridnih aplikacija.

Testiranje aplikacije provedeno je na različitim uređajima s Android operativnim sustavom, a rezultati su pokazali da aplikacija zadovoljava sve funkcionalne zahtjeve. Korištenje nativnih funkcionalnosti poput geolokacije i pristupa mreži bilo je stabilno, a vizualizacija podataka o kvaliteti zraka u realnom vremenu radila je bez poteškoća. Optimizacija aplikacije uključivala je provjere performansi, posebice brzine dohvaćanja podataka i interakcije korisnika s aplikacijom.

Zaključno, ovaj rad pokazuje da Capacitor predstavlja izuzetno koristan alat za razvoj hibridnih mobilnih aplikacija. Kroz konkretan primjer aplikacije za praćenje kvalitete zraka u Hrvatskoj, pokazano je kako se Capacitor može koristiti za jednostavan i brz razvoj aplikacija koje su funkcionalne na više platformi. Korištenje web tehnologija u kombinaciji s nativnim funkcionalnostima omogućilo je izradu aplikacije koja korisnicima pruža gotovo nativno iskustvo.

Prednosti hibridnog razvoja uključuju smanjenje troškova i vremena razvoja, mogućnost korištenja već postojećih web znanja te jednostavno održavanje aplikacije. Aplikacija izrađena u ovom radu nudi korisnicima relevantne informacije o kvaliteti zraka u realnom vremenu, što je od velike važnosti za informiranje javnosti i osvješćivanje o ekološkim problemima. Tehnologije koje omogućuju lakši pristup ekološkim podacima, kao što su API integracije, postaju sve važnije u kontekstu globalnih ekoloških izazova.

Sveučilište
Sjever

SVEUČILIŠTE
SJEVER

IZJAVA O AUTORSTVU

Završni/diplomski/specijalistički 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, Bruno Srnec *(ime i prezime)* pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ica završnog/diplomskog/specijalističkog (*obrisati nepotrebno*) rada pod naslovom Izrada hibridne mobilne aplikacije pomoću Capacitora *(upisati naslov)* te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:
(upisati ime i prezime)

Bruno Srnec
(vlastoručni potpis)

Sukladno članku 58., 59. i 61. Zakona o visokom obrazovanju i znanstvenoj djelatnosti završne/diplomske/specijalističke radove sveučilišta su dužna objaviti u roku od 30 dana od dana obrane na nacionalnom repozitoriju odnosno repozitoriju visokog učilišta.

Sukladno članku 111. Zakona o autorskom pravu i srodnim pravima student se ne može protiviti da se njegov završni rad stvoren na bilo kojem studiju na visokom učilištu učini dostupnim javnosti na odgovarajućoj javnoj mrežnoj bazi sveučilišne knjižnice, knjižnice sastavnice sveučilišta, knjižnice veleučilišta ili visoke škole i/ili na javnoj mrežnoj bazi završnih radova Nacionalne i sveučilišne knjižnice, sukladno zakonu kojim se uređuje umjetnička djelatnost i visoko obrazovanje.

11. Literatura

[1] Lynch, M., Sperry, B., & Bradley, A. (2019). Ionic Framework: Bridging the Gap Between Web and Mobile Development. Proceedings of the International Conference on Web Engineering. Springer. doi:10.1007/978-3-030-20526-7_9 (Preuzeto: 10.7.2024.)

[2] Lynch, M., Sperry, B., & Bradley, A. (2013). Ionic Framework Documentation. Ionic. Izvor: https://ionicframework.com/docs (Preuzeto: 10.7.2024.

[3] Ionic. (2018). Introducing Capacitor: The Native Bridge for Modern Web Apps. Ionic Blog. Izvor: https://ionicframework.com/blog/introducing-capacitor (Preuzeto: 10.7.2024.)

[4] Krawczyk, M., & Strzelecki, M. (2020). Hybrid Mobile Development with Ionic. Packt Publishing. (Preuzeto: 10.7.2024.)

[5] Apache Cordova. (n.d.). Apache Cordova Documentation. Izvor: https://cordova.apache.org/docs/en/latest/ (Preuzeto 11.7.2024.)

[6] Akhtar, F., & Ahsan, K. (2019). Learning React Native. Packt Publishing. (Preuzeto: 11.7.2024.)

[7] Chin, P., & Larsson, J. (2019). Flutter Complete Reference: Create beautiful, fast and native-like apps for any device. Packt Publishing. (Preuzeto: 11.7.2024.)

[8] Krueger, R. (2018). Mastering Xamarin.Forms. Packt Publishing. (Preuzeto: 11.7.2024.)

[9] Brooks, C. (2012). PhoneGap Beginner's Guide. Packt Publishing. (Preuzeto: 13.7.2024.)

[10] Charland, A., & Leroux, B. (2011). Mobile Application Development: Web vs. Native. Communications of the ACM, 54(5), 49-53. (Preuzeto: 13.7.2024.)

[11] Rodriguez-Leon, C. (2020). Node.js: Asynchronous event-driven JavaScript runtime. Journal of Web Engineering, 19(3), 197-220. (Preuzeto: 21.7.2024.)

[12] Rodriguez, C. (2020). "Utility-First CSS Frameworks: Enhancing Front-End Development Efficiency". Journal of Web Engineering, 19(4), 234-250. (Preuzeto: 28.7.2024.)

[13] Developer.com, "Android Studio - Android Programming IDE Overview", 2017. Izvor: https://www.developer.com/mobile/android/android-studio-android-programming-ide-overview/ (Preuzeto: 6.9.2024.)

[14] Hrvatska agencija za okoliš i prirodu. (n.d.). Informacijski sustav za praćenje kvalitete zraka u Republici Hrvatskoj. Izvor: https://iszz.azo.hr/iskzl/emeta.htm (Preuzeto 12.9.2024.)

Popis slika

Slika 1: Ionic team logo,
izvor: https://github.com/ionic-team 3
Slika 2: Capacitor logo,
izvor: https://worldvectorlogo.com/logo/capacitor-js 6
Slika 3: Cordova logo,
izvor: https://commons.wikimedia.org/wiki/File:Apache_Cordova_logo.png 9
Slika 4: React native logo,
izvor: https://worldvectorlogo.com/logo/react-native-1 10
Slika 5: Flutter logo,
izvor: https://en.m.wikipedia.org/wiki/File:Google-flutter-logo.svg 11
Slika 6: Xamarin logo,
izvor: https://www.pngegg.com/en/search?q=xamarin 12
Slika 7: PhoneGap logo,
izvor: https://www.cleanpng.com/png-apache-cordova-mobile-app-development-mobile-backe-1299169/ 14
Slika 8: Instagram logo,
izvor: https://www.pinterest.com/pin/869968852998457878/ 16
Slika 9: Uber logo,
izvor: https://seeklogo.com/vector-logo/299630/uber 16
Slika 10: Twitter i X logo,
izvor: https://logocreator.io/blog/twitter-logo/ 17
Slika 11: Sworkit logo,
izvor: https://www.pngegg.com/en/png-wixjl 17
Slika 12: Untappd logo,
izvor: https://www.behance.net/gallery/60819863/Untappd-Bar-App-Advertisement 17
Slika 13: Pacifica logo,
izvor: https://www.rtor.org/2015/06/25/mental-health-app-review-pacifica/ 18
Slika 14: Android studio logo,
izvor: https://en.wikipedia.org/wiki/Android_Studio 20
Slika 15: JDK logo,
izvor: https://medium.com/@hvirkar36/what-is-jdk-java-development-kit-763756af832e 21
Slika 16: Ionic plugin 22

Slika 17: Stvaranje projekta 22
Slika 18: node.js logo,
izvor: https://en.wikipedia.org/wiki/Node.js 24
Slika 19: Stvaranje Capacitor projekta preko PowerShell-a 24
Slika 20: npm install 25
Slika 21: Komponente aplikacije 25
Slika 22: Prvi test 26
Slika 23: Pretpregled template aplikacije 26
Slika 24: Tailwind test 28
Slika 25: Tailwindcss logo,
izvor: https://getlogovector.com/tailwind-css-logo-vector-svg/ 30
Slika 26: App 1 38
Slika 27: App 2 40
Slika 28: Datepicker 40
Slika 29: Testiranje web aplikacije 47
Slika 30: Testiranje mobilne web aplikacije 47
Slika 31: Gradnja aplikacije 51
Slika 32: Environment Variables 52
Slika 33: JAVA_HOME 52
Slika 34: ANDROID_HOME 53
Slika 35: Path 53
Slika 36: Emulator 54
Slika 37: Dopuštenje 55

Popis kodova

Kod 1: tailwind.config.js 27
Kod 2: postcss.config.js 27
Kod 3: Tailwind klase u style.css 27
Kod 4: <head> 1 32
Kod 5: <head> 2 32
Kod 6: <head> 3 33
Kod 7: <head> 3 33
Kod 8: html 1 34
Kod 9: Definiranje tailwindcss pravila u styles.css 35
Kod 10: Importiranje geolokacije u js 35
Kod 11: Objekt sa svim postajama 35
Kod 11: Objekt sa svim postajama 36
Kod 13: findNearestStation 36
Kod 14: Pozivanje funkcija u geolokacija.js 38
Kod 15: Formular 39
Kod 16: Tablica i polje grafa 40
Kod 17: Datepicker 42
Kod 18: fetchData 43
Kod 19: formatDate 44
Kod 20: drawLineGraph 45
Kod 21: styles.css 48
Kod 22: capacitor.config.json 51
Kod 23: Dodaci u AndroidManifest.xml 54

Popis tablica

Tablica 1: Plugin-ovi 13