Aplikacija za praćenje potrošnje goriva

Muktić, Ivan

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI STUDIJ RAČUNARSTVO

Ivan Muktić

Aplikacija za praćenje potrošnje goriva

ZAVRŠNI RAD

Čakovec, svibanj 2024.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI STUDIJ RAČUNARSTVO

Ivan Muktić, 60198321031301842 86

APLIKACIJA ZA PRAĆENJE POTROŠNJE GORIVA

FUEL CONSUMPTION APPLICATION

ZAVRŠNI RAD

Mentor:

Mr.sc, v.pred. Željko Knok

Čakovec, svibanj 2024.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

PRIJAVA TEME I OBRANE ZAVRŠNOG/DIPLOMSKOG RADA

Stručni prijediplomski studij:

☑ Računarstvo ☐ Održivi razvoj ☐ Menadžment turizma i sporta

Stručni diplomski studij Menadžment turizma i sporta: ☐

Pristupnik: Ivan Muktić, JMBAG: 0313018428
(ime i prezime)

Kolegij: Baze podataka I
(na kojem se piše rad)

Mentor: mr.sc. Željko Knok
(ime i prezime, zvanje)

Naslov rada: Aplikacija za praćenje potrošnje goriva

Naslov rada na engleskom jeziku: Application for monitoring fuel consumption

Članovi povjerenstva: 1. mr.sc. Ivan Hegeduš, predsjednik
(ime i prezime, zvanje)
2. Nenad Breslauer, v.pred., član
(ime i prezime, zvanje)
3. mr.sc. Željko Knok, mentor
(ime i prezime, zvanje)
4. Marija Miščančuk,v.pred., zamjenski član
(ime i prezime, zvanje)

Broj zadatka: 2023-RAČ-5

Kratki opis zadatka:

Glavna funkcija aplikacije bi bila praćenje potrošnje goriva i generiranje putnih naloga za potrebe korisnika. Korisnik ima mogućnost odabira vozila(osobno/poslovno) i kod svakog punjenja unosi informacije potrebne za analizu, koje se dalje spremaju kako bi se mogla pratiti potrošnja na mjesečnoj bazi, također bi imala i grafove. Korisnik bi mogao spremati lokacije benzinskih pumpi koje bi se spremale u aplikaciju, da bi se kasnije mogao navigirati do tih istih, ili ih podijeliti s drugim ljudima.

Datum: 3.7.2024. Potpis mentora: Željko Knok

ZAHVALA

Zahvaljujem se mentoru na dragocjenom usmjeravanju i vođenju tijekom procesa pisanja, što je rezultiralo kvalitetnim razlaganjem i ročenjem teme završnog rada u strukturiranu cjelinu. Također, želim izraziti zahvalnost na znanju i izvrsnom učenju koje sam stekao na svim predmetima koje sam pohađao.

Ivan Muktić

Sažetak

Mobilne aplikacije prvobitno su služile za brzu provjeru elektroničke pošte no njihova velika potražnja dovela je do proširenja na druga područja koja korisnicima pomažu u svakodnevnom životu, kao što su bilješke, podsjetnici i kalendar. Cilj ovog rada jest olakšanje praćenja potrošnje gorima svim korisnicima. Aplikacija je rađena u iOS operacijskom sustavu i koristi razvojno okruženje Xcode[1] te programski jezik swift[2] za izradu korisničkog sučelja i sve potrebne logike. Za trajno pohranjivanje podataka koristi se Core Data okvir koji olakšava komunikaciju sa bazom podataka. Također postoji mogućnost i za uključenje vanjske baze podataka preko platforme Firebase Firestore[3]. Ovaj projekt isprva je zahtijevao izradu modela koji su nam potrebni za povezivanje s bazom, zatim klase koje sadrže glavne operacije za dohvaćanje, slanje, uređivanje i brisanje podataka. Na samom kraju napravljeno je korisničko sučelje koje pruža jednostavno korištenje aplikacije. Kod ulaska u aplikaciju korisnik dolazi do glavnog ekrana u kojem ima mogućnost dodavanja novog vozila. Svako vozilo ima ime, početno stanje brojčanika, registracijske oznake, vrstu goriva i sliku. Nakon dodavanja vozila, svako vozilo ima mogućnost dodavanja novog punjenja, a svako punjenje ima datum, cijenu, volumen i vrijednost brojila. Uz samo par informacija koje korisnik mora unijeti dobivaju se informacije o udaljenosti, cijeni po litri, prosječnoj potrošnji i još mnoge druge. Svako punjenje se sprema u listu specifičnu za to vozilo. Također svako punjenje ulazi u statistike koje se nalaze na drugom ekranu. U statistikama korisnik može pratiti grafove koji za svaki mjesec broje prosječnu potrošnju, ukupni trošak te ukupnu količinu goriva. Statistike također nude mogućnost praćenja potrošnje za sva ili pojedina vozila. Zadnji ekran su postavke koje omogućuju odabir valute, mjernih jedinica za duljinu, volumen i prosječnu potrošnju. Postavke još nude mogućnost odabira preko dvadeset različitih boja koje mogu biti tema aplikacije i gumba koji briše sve podatke iz baze.

Ova aplikacija omogućuje korisnicima da bez napora prate i upravljaju potrošnjom goriva. Detaljnim uvidom u informacije poput prosječne kilometraže, prosječne potrošnje goriva i novca pomaže pojedincima u donošenju informiranih odluka o svojim voznim navikama, održavanju vozila i planiranju ruta. Cilj je da korisnici mogu prepoznati prilike za optimizaciju svojih voznih navika i u konačnici smanjenje ukupnih troškova goriva.

Ključne riječi: *Xcode, swift, iOS, Core Data, Firebase Firestore, gorivo, statistike, postavke*

---

[1]Xcode – razvojna okolina za izradu softvera.
[2]Swift – programski jezik.
[3]Firebase Firestore – vanjska baza podataka
[4]Core Data - iOS sistem baze podataka

Abstract

Mobile applications originally served for quick email check but their high demand has led to expansion into other areas that help users in their everyday lives, such as notes, reminders, and calendars. This work is based on facilitating users to track fuel consumption. The application is developed in the iOS operating system, utilizes the xCode development environment, and the Swift programming language for creating the user interface and all necessary logic. For persistent data storage, the Core Data framework is used, which facilitates communication with the database. There is also the possibility of connecting to an external database via the Firebase Firestore platform. This project initially required the creation of models necessary for connecting to the database, then classes containing main operations for fetching, sending, editing, and deleting data. At the end, a user interface was created to provide easy usage of the application. Upon entering the application, the user arrives at the main screen where they have the option to add a new vehicle. Each vehicle has a name, initial odometer reading, license plate number, fuel type and image. After adding a vehicle, each vehicle has the option to add a new refueling, and each refueling includes a date, price, volume and odometer reading. With just a few pieces of information that the user needs to input, they receive informations about distance, price per liter, average consumption, and many more. Each refueling is saved in a list specific to that vehicle. Additionally, each refueling contributes to the statistics found on another screen. In the statistics, the user can track graphs that count the average consumption, total cost, and total fuel quantity for each month. The statistics also offer the option to track consumption for all vehicles or only for specific ones. The last screen is settings, which allows the selection of currency, units for length, volume, and average consumption. The settings also offer the possibility to choose from over twenty different colors that can be the theme of the application and a button that deletes all data from the database.

This application enables users to effortlessly track and manage fuel consumption. By providing detailed insights into information such as average mileage, fuel consumption and costs, it helps individuals make informed decisions about their driving habits, vehicle maintenance, and route planning. The goal is for users to identify opportunities to optimize their driving habits and ultimately reduce overall fuel costs.

Key words: *xCode, swift, iOS, Core Data, Firebase Firestore, fuel, statistics, settings*

---

[1]Xcode – development environment for software development.
[2]Swift – programming language.
[3]Firebase Firestore – external database platform.
[4]Core Data – iOS database system.

Popis korištenih kartica

iOS operacijski sustav tvrtke Apple za mobilne uređaje
macOS operacijski sustav tvrtke Apple za stolne uređaje
WatchOS operacijski sustav tvrtke Apple za pametne satove
tvOS operacijski sustav tvrtke Apple za pametne televizore
VisionOS operacijski sustav tvrtke Apple za pametne naočale
UI korisničko sučelje
UX korisničko iskustvo
iCloud vanjsko spremište podataka

Table of Contents

1. UVOD

Kroz ovaj rad istražuje se proces izrade iOS aplikacije korištenjem SwiftUI-a, novog, deklarativnog okvira (engl. *framework*) ističući njegove ključne komponente i prednosti koje omogućuju izgradnju dinamičkih i interaktivnih korisničkih sučelja.

Neki od čestih problema kod izrade takvih aplikacija su:

1. Izrada jednostavnog i atraktivnog korisničkog sučelja koje se dobro prilagođava različitim veličinama zaslona i orijentacijama.
2. Jasni i intuitivni tokovi navigacije koji korisnicima omogućuju jednostavno kretanje između različitih zaslona i sekcija aplikacije.
3. Učinkovita implementacija koda i tehnike optimizacije kako bi se osigurala glatka izvedba i brzi rad.
4. Ispravno upravljanje i trajno pohranjivanje podataka.
5. Učinkovito rukovanje greškama i informativne poruke o greškama kako bi se korisnici vodili u slučaju greška ili neočekivanih scenarija.

Cilj ovog rada je izrada funkcionalne i korisne aplikacije koja će kroz svoje jednostavno korisničko sučelje olakšati krajnjim korisnicima lako praćenje potrošnje goriva za svoja vozila, vođenje putnih naloga i drugo. Drugi cilj je ojačati znanje, programerske vještine i stjecanje stručnosti u suvremenom okviru za razvoj iOS aplikacija.

2. APPLE EKOSUSTAV

U Apple-ovom ekosustavu, tehnološki proizvodi su integrirani i podržavaju se unutar šire mreže usluga i platformi. Različiti uređaji poput iPhonea, iPada, Mac računala i Apple sata povezani su kroz zajednički operacijski sustav i tehnološke standarde. Kroz ovaj integrirani ekosustav, korisnicima se pruža dosljedno iskustvo korištenja, a razvojnim programerima omogućuje jednostavno stvaranje aplikacija koje se mogu koristiti na više uređaja.

U okviru Apple-ovog ekosustava, razvoj aplikacija odvija se kroz različite alate i tehnologije koje pruža tvrtka. Xcode, kao glavno razvojno okruženje, omogućuje programerima izradu aplikacija za iOS, macOS, watchOS, tvOS i VisionOS platforme. Kroz Xcode, programeri mogu pristupiti različitim okvirima kao što su SwiftUI i UIKit za izgradnju korisničkih sučelja, te Core Data za upravljanje podacima. Osim toga, Appleov ekosustav pruža podršku za integraciju s različitim servisima poput iCloud-a za pohranu podataka i Apple Pay-a za online plaćanja.

Kroz kontinuirano ažuriranje operacijskih sustava i tehnoloških alata, Apple osigurava da njihov ekosustav ostane siguran, stabilan i funkcionalan. Ovo omogućuje korisnicima da uživaju u najnovijim značajkama i performansama, dok programeri imaju pristup najnovijim tehnologijama i alatima za stvaranje inovativnih aplikacija.

2.1 Programski jezik Swift

Swift je moderni programski jezik razvijen od strane Apple-a za izradu aplikacija na iOS, macOS, watchOS i tvOS platformama. Ovaj jezik je poznat po svojoj jednostavnosti, čitljivosti i sigurnosti. S obzirom na brzu prilagodbu od strane programera i podršku tvrtke Apple, Swift je postao ključni alat u svijetu razvoja mobilnih i desktop aplikacija.

Jedna od ključnih karakteristika Swift-a je njegova jednostavnost i izražajnost, što omogućuje programerima da napišu čist, strukturiran i efikasan kod. Osim toga, Swift podržava različite paradigme programiranja kao što su objektno-orijentirano, funkcionalno i protokol-orijentirano programiranje. Zahvaljujući automatskom upravljanju memorijom, tipizaciji tijekom izvođenja i opcionalnim vrijednostima, Swift također pruža visoku razinu sigurnosti i smanjuje mogućnost grešaka u programiranju. Kroz redovita ažuriranja i dodavanje novih značajki, Swift ostaje relevantan i moćan alat za razvoj aplikacija u Apple-ovom ekosustavu.

2.2 Razvojno okruženje Xcode

Xcode je integrirano razvojno okruženje (engl. *IDE*) razvijeno od strane tvrtke Apple za razvoj softvera. Ono pruža programerima moćan set alata i resursa za izradu visoko kvalitetnih aplikacija.

Xcode uključuje bogat skup alata za upravljanje kodom, uključujući uređivač koda s podrškom za funkcije poput automatskog dovršavanja, popravljanje koda, integraciju s kontrolom verzija te integrirani alat za pronalaženje i ispravljanje grešaka u kodu.

Xcode olakšava programerima cijeli proces razvoja softvera za Apple uređaje, pružajući integriranu podršku za razvoj, testiranje i distribuciju aplikacija. Zahvaljujući svojoj sveobuhvatnosti, Xcode postaje nezamjenjiv alat za programere u Apple ekosustavu.

Slika 1. Xcode ikona

Izvor: https://developer.apple.com/news/?id=ufox7ci5 (20.03.2024.)

2.3 Glavni okvir (engl. *framework*) SwiftUI

SwiftUI je okvir koji se koristi za izradu korisničkih sučelja na Apple platformama. Ovaj okvir omogućava jednostavno definiranje i organiziranje elemenata sučelja putem deklarativnog pristupa.

SwiftUI pruža ugrađene komponente i stilove koji olakšavaju razvoj aplikacija. SwiftUI automatski reagira na promjene podataka i osvježava korisničko sučelje, što smanjuje potrebu za ručnim upravljanjem sučeljem.

U ovom radu SwiftUI će se koristiti za izradu korisničkog sučelja, ekrana, određenih animacija i gesti. Također će se služiti i za integraciju s drugim dijelovima iOS ekosustava, poput Core Date za upravljanje podacima ili Combine za reaktivno programiranje.

Slika 2. SwiftUI ikona (20.03.2024.)

Izvor: https://developer.apple.com/xcode/swiftui/

2.4 Ostali okviri

Kod izrade aplikacije koriste se i neki drugi okviri kao što su Combine, UIKit, Foundation.

Kôd 1. Poziv okvira u aplikaciju

```
import Combine
import Foundation
import UIKit
```

Izvor: Autor

Kod 1 prikazuje način poziva knjižnica potrebnih za aplikaciju.

2.4.1 Foundation

Foundation je temeljni okvir u Swiftu koji pruža osnovne funkcije i tipove podataka za rad s podacima, datotekama, mrežama, i drugim operacijama. Ovaj okvir sadrži klase za rad s nizovima, datotekama, linkovima datumima, JSON-om, i drugim osnovnim operacijama potrebnim za razvoj aplikacija.

2.4.2 Combine

Combine je okvir koji omogućuje reaktivno programiranje u Swiftu. Omogućuje obradu i manipulaciju tokova podataka kao niz asinkronih događaja. Koristi se za upravljanje asinkronim zadacima, rukovanje događajima i upravljanje tokovima podataka.

2.4.3 UIKit

UIKit je stariji alat za izradu sučelja na iOS-u s mnogim komponentama koje SwiftUI ne podržava izravno. Kad su navedene komponente potrebne u SwiftUI aplikaciji, koriste se UIKit prikazivače poput UIImagePickerController. Da bi bili funkcionalni u SwiftUI, koriste se protokoli poput UIViewRepresentable ili UIViewControllerRepresentable. To nam omogućuje korištenje funkcionalnosti UIKit-a unutar SwiftUI-a, tako da aplikacija ima sve što treba.

2.5 Deklarativno i Imperativno Programiranje

Deklarativno programiranje se fokusira na opisivanje željenog rezultata ili stanja, bez eksplicitnog navođenja koraka za postizanje tog cilja.

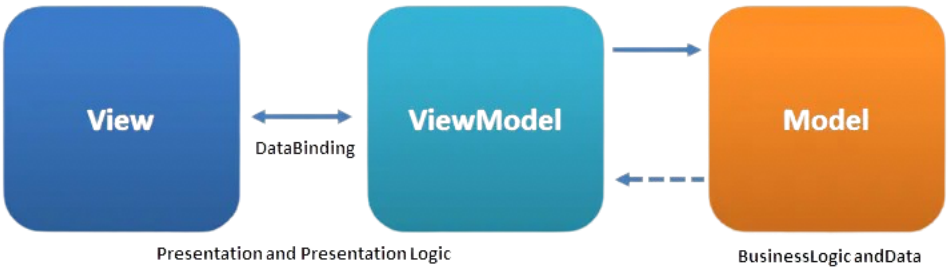
Imperativno programiranje detaljno opisuje korake i naredbe koje računalu treba izvršiti kako bi postiglo željeno stanje ili rezultat.

U SwiftUI okviru, koji koristi deklarativni pristup, programeri opisuju strukturu korisničkog sučelja i odnose između elemenata, dok se u UIKit okviru, koji koristi imperativni pristup, programeri fokusiraju na naredbe i promjene stanja elemenata sučelja.

3. MVVM (engl. *Model-View-ViewModel*) ARHITEKTURA

U modernom razvoju mobilnih aplikacija, arhitekture igraju ključnu ulogu u održavanju skalabilnosti, čitljivosti i upravljivosti koda. MVVM je jedna od popularnih arhitektura.

Slika 3. MVVM arhitektura



Izvor: Autor

Ova arhitektura razdvaja logiku rada od korisničkog sučelja kroz jasno definirane komponente: model koji sadrži podatke i radnu logiku, ekran koji predstavlja korisničko sučelje te model ekrana koji posreduje između modela i ekrana, obrađujući logiku rada i pripremajući podatke za prikazivanje.

3.1 Model

Model predstavlja podatke i radnu logiku aplikacije. Uključuje strukture podataka, baze podataka, mrežne pozive i druge komponente odgovorne za upravljanje podacima. Model je neovisan o korisničkom sučelju i modelu ekrana.

Kôd 2. Model vozila

```
class Vehicle: ObservableObject, Identifiable,
Equatable, Codable {
      static func == (lhs: Vehicle, rhs:
Vehicle) -> Bool {
      lhs.id == rhs.id
}

      var id: UUID
      var name: String
      var odometerInitalValue: Int?
      var fuelType: FuelType
      var licensePlate: String
      var imageName: String?
      var lastService: Double?
      var spanForService: Double?

      var refuels: [Refuel] = []

      init(
            id: UUID = UUID(),
      name: String,
      odometerInitalValue: Int,
      fuelType: FuelType,
      licensePlate: String,
      imageName: String?,
      lastService: Double? = nil,
      spanForService: Double? = nil,
      refuels: [Refuel] = []
) {
      self.id = id
      self.name = name
      self.odometerInitalValue = o
      dometerInitalValue
      self.fuelType = fuelType
      self.licensePlate = licensePlate
      self.refuels = refuels
      self.imageName = imageName
      self.lastService = lastService
      self.spanForService = spanForService
      }
}
```

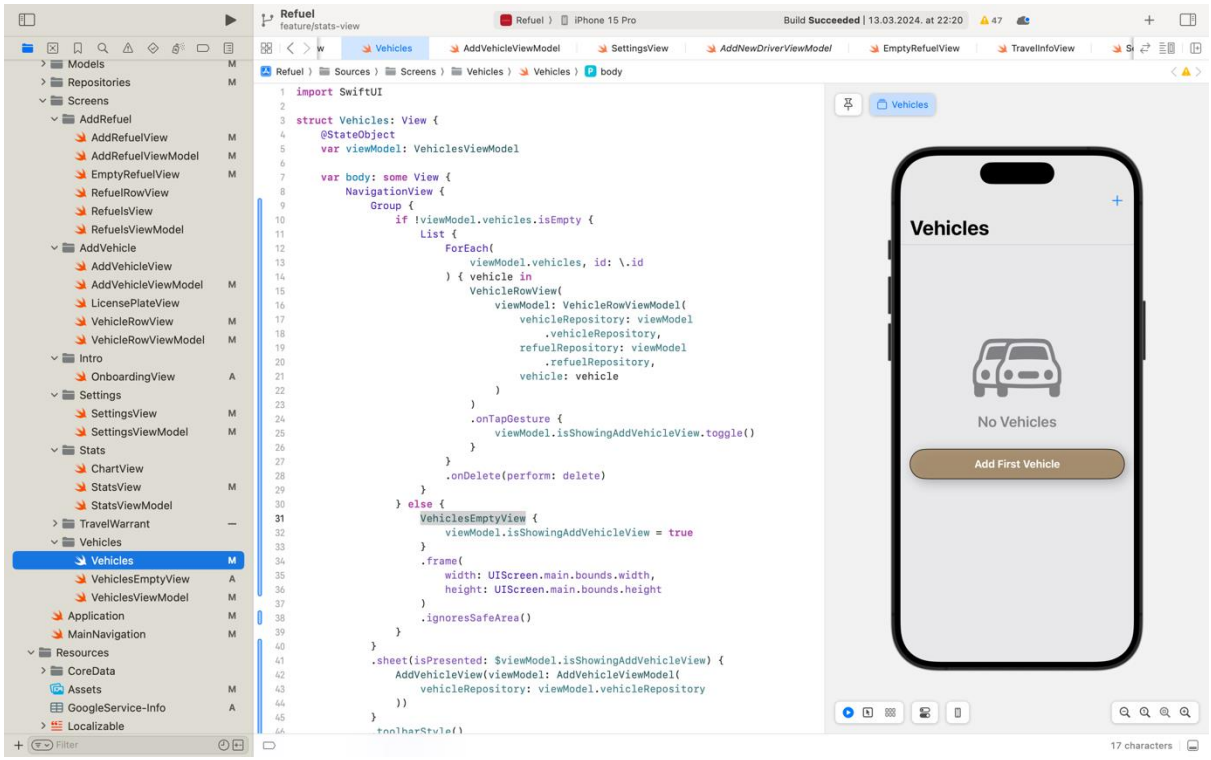
Izvor: Autor

Kod 2 prikazuje model vozila zajedno sa svim svojim svojstvima i konstruktorom.

3.2 Ekran (engl. *View*)

Ovo je sloj korisničkog sučelja u kojem korisnik komunicira s aplikacijom. Uključuje komponente korisničkog sučelja poput gumba, oznaka, polja za unos teksta i prikaza. Ekran je odgovoran za prikazivanje podataka i prikupljanje korisničkih ulaza.

Slika 4. Navigacija, kod I izgled ekrana u simulatoru



Izvor: Autor

Slika 4. prikazuje s lijeve strane navigacijski prozor, u sredini uređivač, a desno pregled na kojem se nalazi uređaj.

1. Navigacijski prozor (engl. *Navigator*): Omogućuje pristup projektu, datotekama i resursima te olakšava upravljanje projektom.
2. Uređivač (engl. *Editor*): Središnji prozor kodiranja omogućuje uređivanje i pregledavanje izvornog koda, storyboarda ili drugih datoteka u projektu.
3. Pregled (engl. *Preview*): Omogućuje vizualni prikaz sučelja ili izgleda aplikacije kako bi se mogao provjeriti izgled i funkcionalnost na različitim uređajima.

3.3 Model ekrana (engl. *ViewModel*)

Model zaslona predstavlja ključni dio koji upravlja radnim logikama i manipulacijom podataka. Odvojen je od samog ekrana što omogućuje čisto razdvajanje logike prikaza od radne logike.

Model ekrana u SwiftUI-u često drži stanje aplikacije i pruža podatke koje ekrani koriste za prikazivanje. Također se brine o interakciji korisnika, kao što su obrada korisničkih akcija i ažuriranje stanja aplikacije na temelju tih akcija. Kroz ovakav pristup, model ekrana postaje središnje mjesto za upravljanje stanjem aplikacije i komunikaciju s modelom putem koje osigurava ažuriranje podataka i održavanje dosljednosti aplikacije.

Kôd 3. Funkcije modela ekrana vozila

```
init(
        vehicleRepository: VehicleRepository,
        refuelRepository: RefuelRepository,
        travelWarrantRepository:
TravelWarrantRepository
    ) {
        self.vehicleRepository =
vehicleRepository
        self.refuelRepository =
refuelRepository
        self.travelWarrantRepository =
travelWarrantRepository
        self.vehicleRepository.$vehicles
            .receive(on: RunLoop.main)
            .sink(receiveValue: { _ in
                self.reload()
            })
            .store(in: &anyCancellables)
    }

    func reload() {
        Task {
            vehicles = []
            vehicles.append(contentsOf:
vehicleRepository.vehicles)
        }
    }
    func delete(at offsets: IndexSet) {
        Task {
            try await
vehicleRepository.remove(at: offsets)
             try await
refuelRepository.getAll()
         }
```

Izvor: Autor

Kod 3 prikazuje funkcije koje sadrži model ekrana i njegov konstruktor koji sadrži sva potrebna svojstva te funkciju za automatsko ažuriranje podataka.

4. BAZE PODATAKA

Baze podataka su temeljni dijelovi softverskih sustava koji omogućuju organiziranje, spremanje i manipuliranje podacima. One predstavljaju strukturirane skupove informacija koje se mogu pohraniti, upravljati, dohvaćati i brisati putem odgovarajućih upita.

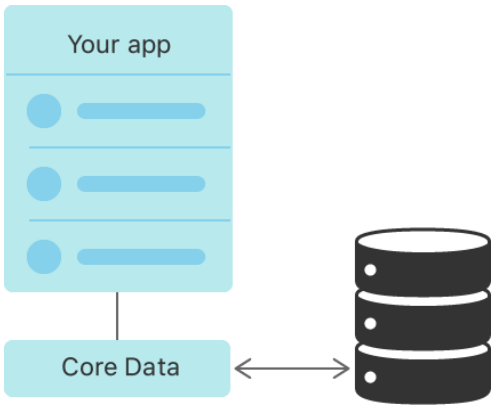
Dva glavna tipa baza podataka su relacijske i nerelacijske. Relacijske baze podataka koriste tablice povezane prema relacijama, dok se nerelacijske baze koriste za nestrukturirane podatke. Primjeri relacijskih baza su MySQL, PostgreSQL, SQLite, Oracle Database, a primjeri nerelacijskih su MongoDB, Cassandra i Redis.
U ovom projektu koristiti će se Core Data i Firebase okviri preko kojih aplikacija komunicira s bazom.

4.1 Core Data

Core Data je okvir za lokalno spremanje podataka na Apple uređajima. Obično se koristi za lokalno spremanje podataka na uređajima korisnika, omogućujući brz i učinkovit pristup podacima.

Koristi SQLite relacijsku bazu podataka koja omogućuje razvoj aplikacija s kompleksnim modelima podataka, na način da olakšava manipulaciju podacima kroz objekte i veze između njih.

Slika 5. Relacija između uređaja, Core Date i baze podataka



Izvor: https://developer.apple.com/documentation/coredata

Slika 5 prikazuje relaciju između uređaje i baze podataka.

4.2 Firebase Firestore

Firebase je platforma za razvoj aplikacija koja pruža različite usluge, uključujući autentifikaciju, analitiku, upravljanje korisnicima i bazu podataka u oblaku.

Firebase Firestore je jedna od ključnih usluga koju nudi Firebase platforma, koju je razvio Google. Ona omogućuje razvoj aplikacija s brzom i skalabilnom bazom podataka u oblaku. Firestore se ističe svojom nerelacijskom strukturom podataka, koristeći NoSQL pristup

umjesto tradicionalnog relacijskog modela. Podaci se organiziraju u obliku kolekcija dokumenata, gdje svaki dokument može sadržavati raznovrsne podatke.

4.3 Uzorak repozitorija (engl. *Repository Pattern*)

Uzorak repozitorija koristi se za odvajanje pristupa podacima od ostatka aplikacije. U MVVM arhitekturi, uzorak repozitorija služi za komunikaciju između modela ekrana i izvora podataka.

Kôd 4. Protokol s akcijama modela

```
import Foundation

enum RepositoryError: Error {
    case modelNotFound
    case vehicleNotFound
}

protocol Repository {
    associatedtype T

    func add(_ model: T) async throws -> T
    func get(id: UUID) async throws -> T?
    func getAll() async throws -> [T]
    func update(id: UUID, model: T) async throws -> T
    func delete(id: UUID) async throws
    func remove(at offsets: IndexSet) async throws
}
```

Izvor: Autor

Kod 4 prikazuje protokol koji omogućuje osnovne radnje s modelima, poput dodavanja, dohvaćanja, ažuriranja i brisanja. Također podržava asinkrono ponašanje i upravljanje greškama poput nepostojećeg modela ili vozila.

4.4 Implementacija uzorka repozitorija

Kod implementacije uzorka repozitorija u aplikaciju modeli se dijele u dvije skupine: podatkovni modeli i modeli domene, koji zajedno igraju ključnu ulogu u osiguravanju učinkovitog rada aplikacije.

Modeli domene se fokusiraju na prikaz poslovne logike i koncepata domene aplikacije, dok se modeli podataka bave pohranom i dohvaćanjem podataka u bazi. Modeli domene komuniciraju sa ekranima, pripremajući i pružajući podatke za prikaz, dok dana modeli izravno komuniciraju s bazom podataka radi trajnog upravljanja podataka. Ova podjela pomaže u održavanju jasne, sigurne i lako održive arhitekture u razvoju aplikacija.

Glavna dodirna točka između modela domene i podatkovnih modela su funkcije koje mapiraju te podatke. Koriste se za mapiranje podataka iz jednog formata ili modela u drugi. Ovaj

postupak omogućuje kompatibilnost između različitih dijelova aplikacije. U ovom primjeru se koristi za povezivanje između modela domene i entiteta baze podataka.

Kôd 5. Mikro servis za mapiranje modela

```
struct VehicleMapper: DomainModelMapper {
    typealias domainModel = Vehicle
    typealias entityModel = VehicleEntity

    static func mapToDomainModel(from entity: VehicleEntity) -> Vehicle {
        return Vehicle(
            id: entity.id ?? UUID(),
            name: entity.name ?? "Unknown name",
            odometerInitalValue: Int(entity.odometerInitialValue),
            fuelType: FuelType(rawValue: entity.fuelType ?? "") ?? .autogas,
            licensePlate: entity.licensePlate ?? "Unknown license plate",
            imageName: entity.imageName ?? "Invalid photo name",
            lastService: entity.lastService,
            spanForService: entity.spanForService
        )
    }

    static func mapToEntity(from domain: Vehicle, entity: inout VehicleEntity) {
        entity.id = domain.id
        entity.name = domain.name
        entity.odometerInitialValue = Int32(domain.odometerInitalValue ?? 0)
        entity.fuelType = domain.fuelType.rawValue
        entity.licensePlate = domain.licensePlate
        entity.imageName = domain.imageName
        entity.lastService = domain.lastService ?? 0.0
        entity.spanForService = domain.spanForService ?? 0.0
    }
}
```

Izvor: Autor

Kod 5 prikazuje kako podaci entiteta iz baze podataka trebaju biti prikazani u korisničkom sučelju, koristi se prva funkcija za mapiranje tih podataka u modele domene koje se obrađuju u modelu ekrana i prikazuju na korisničkom sučelju.

Kada korisnik promijeni podatke u korisničkom sučelju (na primjer, uređuje informacije o vozilu), koristi se druga funkcija za ažuriranje entiteta u bazi podataka s novim informacijama unesenim od strane korisnika.

5. OSTALE KOMPONENTE

Upotreba raznih komponenti igra ključnu ulogu u poboljšanju organizacije koda, čitljivosti i mogućnosti ponovne upotrebe istog koda na različitim mjestima. Te komponente djeluju poput stupova koji nam pomažu učinkovito strukturirati naš kod i rješavati složene zadatke.

Aplikacija Refuel u mnogočemu se oslanja na razne komponente poput prikaza koji se pojavljuju na više različitih ekrana, obrade pogrešaka, ekstenzija, uređivača prikaza i raznoraznih servisa.

Razbijanjem velikih zadataka na manje komade i korištenjem ovih komponenti, programeri mogu stvoriti aplikacije koje su dobro organizirane, učinkovite i lake za održavanje.

5.1 Ekstenzije (engl. *extensions*)

Ekstenzije u Swiftu omogućuju programerima dodavanje novih funkcionalnosti postojećim tipovima, uključujući klase, strukture, enumeracije, protokole, pa čak i ugrađene tipove poput niza znakova (*engl. string*), cijelih brojeva (*engl. integer*), datuma.

Po tom principu ekstenzije poboljšavaju čitljivost, održavanje i ponovnu upotrebljivost koda jer omogućuju logično grupiranje srodnih funkcija. Osim toga, ekstenzije olakšavaju korištenje protokol orijentiranog programiranja, što omogućuje programerima pisanje fleksibilnijeg i kvalitetnijeg koda.

Kôd 6. Ekstenzija koja pretvara cijeli broj u decimalni

```
extension Binding where Value == Int? {
    public func double() -> Binding<Double?> {
        return Binding<Double?>(
            get: {
                guard let value = self.wrappedValue else {
                    return nil
                }
                return Double(value)

            },
            set: {
                guard let value = $0 else {
                    self.wrappedValue = nil
                    return
                }
                self.wrappedValue = Int(value)
            }
        )
    }
}
```

Izvor: Autor

Kod 6 prikazuje ekstenziju tipa podataka koji pruža dvosmjernu komunikaciju objekta (engl. *Binding*). Unutar ekstenzije definirana je funkcija koja omogućuje pretvaranje cjelobrojnih

vrijednosti u decimalne. Prilikom čitanja vrijednosti, funkcija provjerava da li varijabla sadrži vrijednost. Ako sadrži, pretvara je u decimalni broj i vraća rezultat, ako ne sadrži, vraća informaciju da varijabla ne sadrži vrijednost.

5.2 Uređivač prikaza (engl. *View Modifier*)

Uređivači prikaza su alati kojima se oblikuje izgled i ponašanje SwiftUI prikaza. Pružaju programerima mogućnost da mijenjaju izgled, raspored i ponašanja prikaza bez potrebe za direktnim mijenjanjem osnovnog koda. Uređivači prikaza su neophodni za kreiranje vizualno privlačnih i interaktivnih korisničkih sučelja u SwiftUI-u.

Kôd 7. Uređivač prikaza koji mijenja izgled alatne trake

```
struct ToolbarModifier: ViewModifier {
    func body(content: Content) -> some View {
        if #available(iOS 16.0, *) {
            content
                .toolbarBackground(
                    Color.gray.opacity(0.2),
                    for: .navigationBar, .tabBar
                )
                .toolbarBackground(
                    .visible,
                    for: .navigationBar, .tabBar
                )
                .background(Color.gray.opacity(0.2))
        }
    }
}

extension View {
    func toolbarStyle() -> some View {
        modifier(ToolbarModifier())
    }
}
```

Izvor: Autor

Kod 7 prikazuje prilagođeni SwiftUI uređivač prikaza nazvan 'ToolbarModifier'. Svaki uređivač se mora pridržavati protokola koji zahtijeva implementaciju funkcije body s jednim ulaznim parametrom content. Sadržaj (engl. *content*) predstavlja prikaz na koji kasnije pozivamo uređivač prikaza. Na sadržaj dodajemo druge standardne uređivače, u ovom slučaju za promjenu boje navigacijskih traka. Na samom kraju radimo ekstenziju tipu na kojem želimo primijeniti uređivač. Ta ekstenzija sadrži metodu koja omotava uređivač radi jednostavnijeg korištenja diljem aplikacije.

5.3 Servisi

U Swiftu, servisi se obično odnose na klase ili strukture odgovorne za grupiranje određene funkcionalnosti ili operacija unutar aplikacije. One služe kao centralizirano mjesto za rukovanje određenim zadacima ili funkcionalnostima koje nisu izravno povezane s logikom korisničkog sučelja.

Kôd 8. Servis za pohranu i dohvaćanje podataka

```
import SwiftUI

struct ToolbarModifier: ViewModifier {
    func body(content: Content) -> some View {
        if #available(iOS 16.0, *) {
            content
                .toolbarBackground(
                    Color.gray.opacity(0.2),
                    for: .navigationBar, .tabBar
                )
                .toolbarBackground(
                    .visible,
                    for: .navigationBar, .tabBar
                )
                .background(Color.gray.opacity(0.2))
        }
    }
}

extension View {
    func toolbarStyle() -> some View {
        modifier(ToolbarModifier())
    }
}
```

Izvor: Autor

Kod 8 prikazuje servis koji služi za pohranu i dohvaćanje podataka koristeći ugrađeni servis za pohranu podataka (engl. *UserDefaults*). Veća fleksibilnost i ponovna upotreba koda omogućene su upotrebom generičkih funkcija, s obzirom da te funkcije mogu biti korištene s različitim vrstama podataka, što rezultira čišćim i efikasnijim kodom.

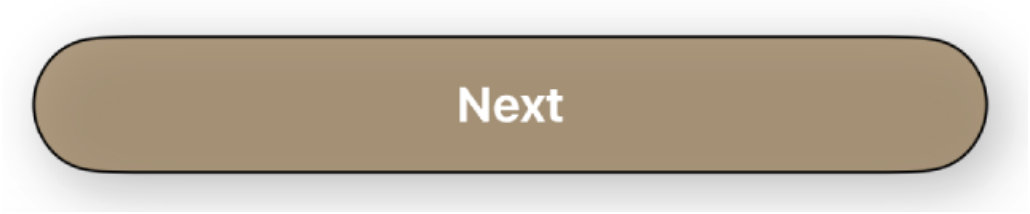
5.4 Zajednički prikazi

Prikazi se tipično odnose na vizualne elemente koji čine korisničko sučelje (UI) aplikacije.

Tekst, slika, gumb, polje za unos, lista i prekidač samo su neki od primjera raznih prikaza dostupnih u Swiftu. Prikazi igraju ključnu ulogu u oblikovanju korisničkog iskustva te potiču rast broja aktivnih korisnika i zadržavanje aktivnih korisnika u aplikaciji. Fokusiranjem na upotrebljivost, jednostavnost, pristupačnost i funkcionalnost, programeri mogu stvoriti prikaze koji oduševljavaju korisnike i doprinose dugoročnom uspjehu aplikacije.

Prilagođeni prikaz je komponenta koju stvara programer. Ona ima vlastiti raspored elemenata i različitu funkcionalnost te se može koristiti kroz cijelu aplikaciju. Prednosti stvaranja prilagođenih prikaza su brojne. Istu komponentu može se ponovno koristiti u različitim dijelovima aplikacije. Ne samo da štedi vrijeme, već osigurava i dosljednost u cijeloj aplikaciji. Također, kod postaje modularniji i lakši za održavanje.

Slika 6. Prikaz izgleda gumba iz aplikacije



Izvor: Korisnik

Slika 6 prikazuje izgled gumba korištenog u aplikaciji koji se koristi kroz cijelu aplikaciju i ima mogućnosti mijenjanja teksta, boje pozadine, ruba i slova te svojstva učitavanja i onemogućenosti klika.

6. KORISNIČKO SUČELJE

Dizajn korisničkog sučelja (UI) ključan je dio razvoja (iOS) aplikacija koji izravno utječe na korisničko iskustvo (UX), broj zadovoljnih korisnika, pristupačnost, performanse i identitet aplikacije.

Tablica 1. Dijelovi korisničkog sučelja

Karakteristika	Opis
Navigacijska traka	Omogućava korisnicima navigaciju kroz aplikaciju, prikazuje trenutni položaj korisnika unutar aplikacije te pruža mogućnost brzog pristupa važnim funkcijama i sadržajima.
Glavni sadržajni prostor	Predstavlja centralno područje u kojem se prikazuje osnovni sadržaj aplikacije, kao što su tekstovi, slike, tablice ili druge vrste informacija koje korisnik konzumira ili s kojima komunicira.
Interaktivni elementi	Omogućavaju korisnicima izvođenje različitih akcija unutar aplikacije, kao što su slanje obrazaca, odabir opcija ili kontroliranje postavki.
Izbornici	Izbornici pružaju korisnicima pristup različitim funkcijama i opcijama aplikacije, dok padajući izbornici omogućuju odabir između više mogućnosti koje se prikažu nakon određene radnje korisnika.
Obavijesti	Korisnicima pružaju informacije o trenutnom stanju aplikacije, uspješnosti izvršenih radnji ili o potrebi korisničke interakcije radi rješavanja nekih situacija ili problema.

Izvor: Autor

Tablica 1 prikazuje elemente koji zajedno čine glavni dio korisničkog sučelja i ključni su za oblikovanje korisničkog iskustva koje je intuitivno, ugodno i učinkovito za korisnike. Kvalitetno oblikovanje glavnog dijela UI-a doprinosi zadovoljstvu korisnika i uspjehu aplikacije na tržištu.

6.1. Uvod u aplikaciju

Uvod u aplikaciju u mobilnom razvoju je poput prijateljskog vodiča koji dočekuje korisnike u aplikaciju, prikuplja najbitnije podatke i pomaže im da počnu. Uvod u aplikaciju je važan jer osigurava da korisnici razumiju što aplikacija radi, kako je koristiti i zašto je vrijedna. Dobar uvod u aplikaciju može zadržati korisnike angažiranima i zadovoljnima, što dovodi do boljeg usvajanja i duljeg korištenja aplikacije.

Svaki dobar uvod u aplikaciju trebao bi biti jasan, interaktivan i prilagođen korisniku. To znači da bi trebao jasno objasniti što aplikacija radi, omogućiti korisnicima da sudjeluju u učenju o njoj te uzeti u obzir njihove osobne preferencije.

6.2. Navigacija

Navigacija u SwiftUI-u omogućuje korisnicima da se kreću između različitih dijelova aplikacije kako bi pronašli ono što traže ili izvršili određene akcije. Važna je jer pruža korisnicima intuitivno iskustvo korištenja aplikacije, pomažući im da brzo pronađu ono što im je potrebno. Dobra navigacija uključuje jasnu strukturu aplikacije, jednostavne i konzistentne kontrole za kretanje te intuitivno raspoređene sadržaje.

Tablica 2. Vrste navigacija

Naziv	Opis
Stog navigacija	Koristi se za prikazivanje više pogleda jedan na vrhu drugoga u obliku stoga ili skupa.
Tab navigacija	Koristi se za prikazivanje različitih sekcija ili funkcionalnosti aplikacije u karticama na dnu zaslona.
Modalna navigacija	Koristi se za prikazivanje privremenih ili kontekstualnih prozora koji se pojavljuju preko glavnog sadržaja.
List navigacija	Koristi se za prikazivanje prozora u obliku lista koji se dignu do vrha ekrana preko glavnog sadržaja.

Izvor: Autor

Tablica 2 prikazuje neke od najpoznatijih i najboljih vrsta navigacije koje postoje. U ovoj aplikaciji koriste se sve nabrojene navigacije, savršeno ukomponirane i usklađene za najbolje korisničko iskustvo.

6.3. Komponente sučelja

Komponente korisničkog sučelja su interaktivni elementi koje korisnici vide i uz pomoć kojih korisnici komuniciraju sa aplikacijom.

Uključuju gumbe za radnje, tekstualna polja za unos, oznake za informacije, slike za vizualni dio, okvire za potvrđivanje i radio gumbe za odabir, liste i tablice za prikaz podataka, klizače i brojače za prilagođavanje vrijednosti, prekidače za prebacivanje opcija i pokazatelje napretka za povratnu informaciju o zadatku.

UI komponente obično se organiziraju i stiliziraju prema dizajnerskim načelima i smjernicama platforme ili okvira koji se koristi. Učinkovitom upotrebom UI komponenti, programeri mogu stvoriti intuitivna i vizualno privlačna korisnička sučelja koja poboljšavaju ukupno korisničko iskustvo aplikacije.

6.4 Lokalizacija

Lokalizacija u SwiftUI-u omogućuje prilagodbu aplikacije različitim jezicima i regionalnim postavkama.

To znači da se tekstovi, slike i druge komponente sučelja mogu prilagoditi tako da budu prikladni za korisnike diljem svijeta. Lokalizacija je korisna jer omogućuje širenje publike i poboljšava korisničko iskustvo tako što korisnicima pruža informacije na njihovom materinjem jeziku. Ovaj rad podržava dva jezika: hrvatski i engleski.

7. APLIKACIJA REFUEL

Ovaj dio rada govori o samoj aplikaciji, kako je napravljena, koja je ciljana publika, koje značajke nudi, po čemu se razlikuje od drugih i zašto bi bila sjajna aplikacija za imati na telefonu.

Važne smjernice koje treba imati na umu tijekom ovog projekta su ključne za kreiranje kvalitetne i privlačne aplikacije. Takva aplikacija ne samo da će privući korisnike, već će ih i zadržati, nudeći izvrsno korisničko iskustvo koje je intuitivno i bez smetnji. Bitno je osigurati dobar korisnički doživljaj, stabilnost, sigurnost i kvalitetan dizajn korisničkog sučelja. Također je važno izbjegavati čimbenike koji bi mogli uzrokovati nezadovoljstvo korisnika ili čak dovesti do brisanja aplikacije, kao što su loše korisničko iskustvo, naplata za pristup određenim značajkama, prikazivanje oglasa, problemi s funkcionalnošću aplikacije poput rušenja ili curenja memorije.

Slika 7. Logo aplikacije

Izvor: Autor

Slika 7 prikazuje ručno izrađeni logo koji krasi izgled svakog mobilnog uređaja koji posjeduje aplikaciju.

7.1 Ideja

Ciljana publika ove aplikacije su vlasnici iPhone uređaja svih dobnih skupina koji posjeduju jedno ili više vozila te žele pratiti potrošnju goriva svih svojih vozila na jednom mjestu. Također, aplikacija bi imala funkcionalnosti koje bi bile korisne i za poslovne svrhe, posebno za tvrtke koje održavaju vlastiti prijevoz i trebaju evidentirati putne naloge. Na taj način, moguće je ukloniti potrebu za korištenjem papira.

Glavni cilj ovog projekta je stvoriti prvu verziju aplikacije koja će biti privlačna i korisna korisnicima. Također je bitno da bude dobro koncipirana, organizirana i izrađena kako bi olakšala unapređenje u budućim verzijama.

Ova aplikacija postigla je to svojim privlačnim i elegantnim dizajnom, intuitivnim korisničkim sučeljem te obiljem korisnih usluga koje pruža. Istovremeno, koristi jednu od najmoćnijih arhitektura, uspostavljajući čvrste veze i komunikaciju s bazom podataka te prateći najnovije standarde programskog jezika i razvojnog okruženja.

7.2. Uvod u aplikaciju

Uvod u aplikaciju je ključni proces osmišljen za upoznavanje i vođenje novih korisnika kroz značajke i funkcionalnosti mobilne aplikacije. Ova početna interakcija je presudna jer postavlja ton za korisničko iskustvo (UX) i određuje hoće li korisnik nastaviti koristiti aplikaciju. Učinkoviti uvod u aplikaciju pomaže korisnicima brzo i efikasno razumjeti vrijednost aplikacije, osiguravajući da mogu koristiti njezine značajke bez frustracije.

Tablica 3. Prednosti uvoda u aplikaciju

Prednosti	Opis
Zadržavanje korisnika	Dobro osmišljen proces uvoda korisnika u aplikaciju može značajno poboljšati njihovo zadržavanje.
Poboljšano korisničko iskustvo	Uvođenje korisnika smanjuje krivulju učenja za nove korisnike, što dovodi do boljeg i ugodnijeg iskustva.
Veće stope konverzije	Učinkoviti uvod u aplikaciju povećava vjerojatnost da će korisnik izvršiti željene radnje, poput kupnje ili pretplate na uslugu.
Smanjeni troškovi podrške	Uz sveobuhvatan proces uvođenja korisnika, manja je vjerojatnost da će korisnici naići na probleme koji zahtijevaju podršku.

Izvor: Autor

Tablica 3 prikazuje prednosti uključivanja promišljenog i korisnički orijentiranog procesa uvođenja korisnika u mobilnu aplikaciju.

7.2.1 Praćenje potrošnje goriva

Aplikacija nudi jednostavan način praćenja potrošnje goriva za sva vozila koja posjeduju fizičke ili pravne osobe. Kroz unos samo tri ključna podatka – količine goriva, cijene i vrijednosti na kilometar satu, aplikacija prikazuje prijeđenu udaljenost, cijenu po jedinici potrošnje i potrošnju od posljednjeg točenja goriva.

Nakon spremanja unosa o točenju goriva, aplikacija automatski ažurira popis koji se može pregledati na posebnom zaslonu. Ovdje korisnici mogu vidjeti ukupnu cijenu i prosječnu potrošnju goriva za sva vozila. Osim toga, unos o točenju goriva šalje informacije na statistički zaslon, koji automatski ažurira grafove.

Statistički zaslon nudi prikaz filtriranih informacija o potrošnji goriva. Korisnici mogu vidjeti ukupnu cijenu, ukupnu potrošnju i ukupni volumen za sva vozila ili za odabrano vozilo. Ovo je posebno korisno za usporedbu između različitih vozila.

Tablica 4. Karakteristike praćenja potrošnje goriva

Karakteristika	Opis
Praćenje potrošnje goriva	Omogućava korisnicima da bolje razumiju potrošnju goriva i identificiraju neučinkovitosti. Ovo je posebno važno za tvrtke koje žele smanjiti operativne troškove i optimizirati korištenje svojih vozila.
Usporedbe potrošnje između različitih vozila	Omogućava brzu analizu i uvid u potrošnju goriva bez potrebe za ručnim obračunom. Ovo štedi vrijeme i smanjuje mogućnost pogrešaka.
Automatsko ažuriranje statističkih podataka i grafova	Pomaže korisnicima da donesu informirane odluke o održavanju vozila i potencijalnim zamjenama. Na primjer, ako jedno vozilo troši znatno više goriva od ostalih, to može biti signal za tehnički pregled ili razmatranje zamjene vozila.

Izvor: Autor

U tablici 4 vidljive su mnogobrojne karakteristike jednostavnog praćenja potrošnje goriva.

7.2.2 Putni nalog

Putni nalozi predstavljaju jednostavan način za praćenje poslovnih ruta, zamjenjujući staromodno pisanje na papiru unutar automobila. Ova funkcionalnost aplikacije omogućava korisnicima da digitalno upravljaju svojim poslovnim putovanjima na učinkovit i pregledan način.

Unošenjem vrijednosti kilometar sata, udaljenost se automatski izračunava u stvarnom vremenu. Odabir vozača, dodavanje početnog i krajnjeg mjesta pružaju sve potrebne informacije za kreiranje putnog naloga. Ovaj pristup pojednostavljuje proces evidentiranja putovanja, omogućujući korisnicima da se usredotoče na svoje zadatke bez dodatnog administrativnog opterećenja.

Nakon spremanja putnog naloga, aplikacija automatski ažurira popis koji uključuje ukupnu udaljenost svih ruta. Ovaj popis sadrži i gumb za kreiranje PDF datoteke putnog naloga, što dodatno štedi vrijeme korisnicima i pruža bolju kontrolu za tvrtke koje trebaju putne naloge.

Tablica 5. Karakteristike korištenja putnih naloga

Karakteristika	Opis
Eliminira potrebu za korištenjem papira	Eliminacijom potrebe za ručnim zapisivanjem podataka na papir smanjuje se mogućnost pogrešaka i gubitka važnih informacija.
Automatski izračun udaljenosti	Omogućava preciznije i pouzdanije podatke o putovanjima, pojednostavljuje proces unosa podataka i osigurava točnost informacija koje su ključne za poslovno planiranje i analizu.
Generiranja PDF datoteka	Korisnici mogu brzo kreirati i podijeliti potrebne dokumente. Ovo je posebno korisno za tvrtke koje trebaju redovite i točne izvještaje o putovanjima svojih zaposlenika.

Izvor: Autor

U tablici 5 prikazane su neke od glavnih prednosti korištenja aplikacije kod izrade putnih naloga.

7.2.3 Obavijest o servisu

Podsjetnik za servis vozila predstavlja izuzetno korisnu funkcionalnost, posebno za tvrtke s velikim brojem vozila, ali i za pojedince koji posjeduju više automobila i ne žele razmišljati ili provjeravati kada bi sljedeći servis trebao biti obavljen. Ova značajka pomaže u održavanju redovitog i pravovremenog održavanja vozila, čime se smanjuje rizik od neočekivanih kvarova i povećava sigurnost na cesti.

Unosom datuma posljednjeg servisa i željenog intervala za sljedeći servis, aplikacija automatski zakazuje obavijest. Korisnici su obaviješteni tjedan dana prije datuma servisa, na sam dan servisa i tjedan dana nakon datuma servisa. Ova trostruka obavijest osigurava da korisnici ne zaborave na važan termin, omogućujući im da planiraju i organiziraju svoje vrijeme i resurse na najbolji mogući način.

8. OSOBNE POSTAVKE

Osobne preferencije korisnika predstavljaju izvrstan način za održavanje interesa korisnika i pomažu im u lakšem korištenju aplikacije. U današnje vrijeme, s rastom tehnologije, sve više ljudi prilagođava način na koji koriste aplikacije, a opcije poput tamnog i svijetlog načina rada najbolji su primjer toga.

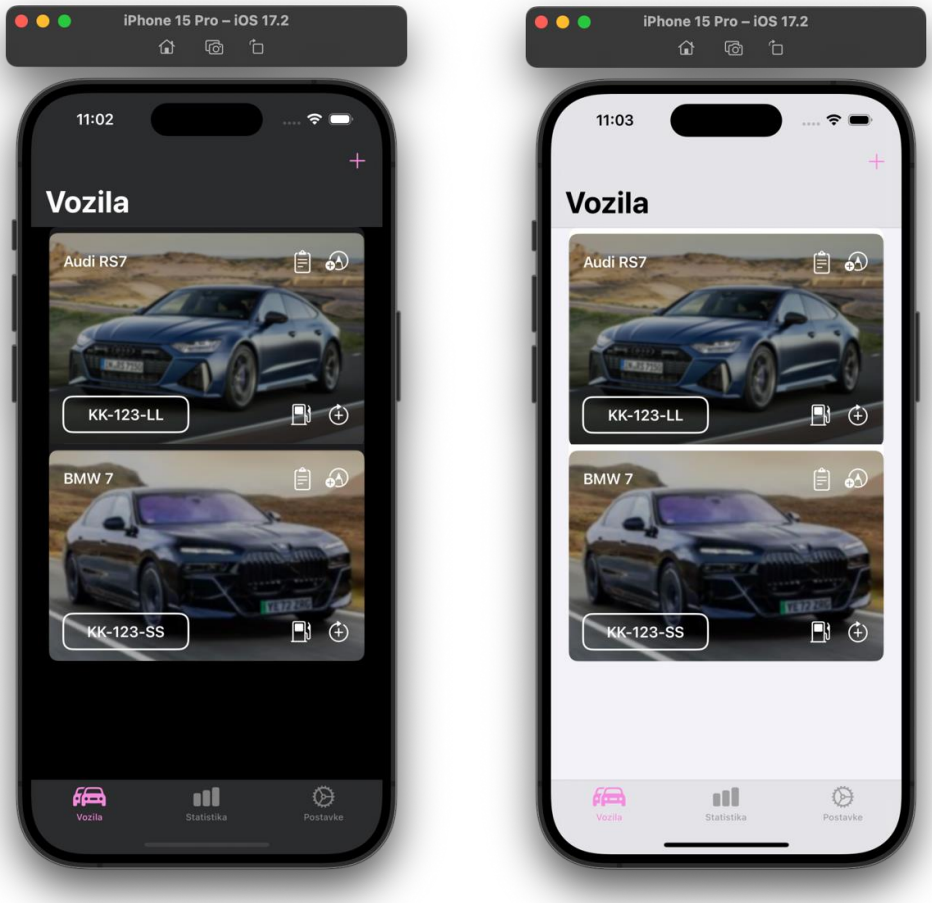
Implementacija osobnih preferencija u aplikaciji donosi brojne prednosti. Korisnici mogu prilagoditi aplikaciju svojim potrebama i željama, što povećava zadovoljstvo i angažman.

Podrška koju aplikacija nudi za tamni i svijetli način rada, prilagodbu boja tema i lokalizaciju jezika čini aplikaciju pristupačnijom i korisnijom za širu publiku.

8.1 Svijetli i tamni način rada

Tamni način rada postao je izuzetno popularan među korisnicima. Mnogi ljudi preferiraju tamni način rada jer smanjuje naprezanje očiju, posebno pri korištenju aplikacija u uvjetima slabog osvjetljenja. Nemogućnost aplikacije da podržava tamni način rada može dovesti do prestanka korištenja od strane korisnika. Stoga, podrška za oba načina rada, tamni i svijetli, postaje sve važnija za zadržavanje korisnika i poboljšanje njihovog iskustva.

Slika 8. Prilagodba teme: tamni i svijetli način rada



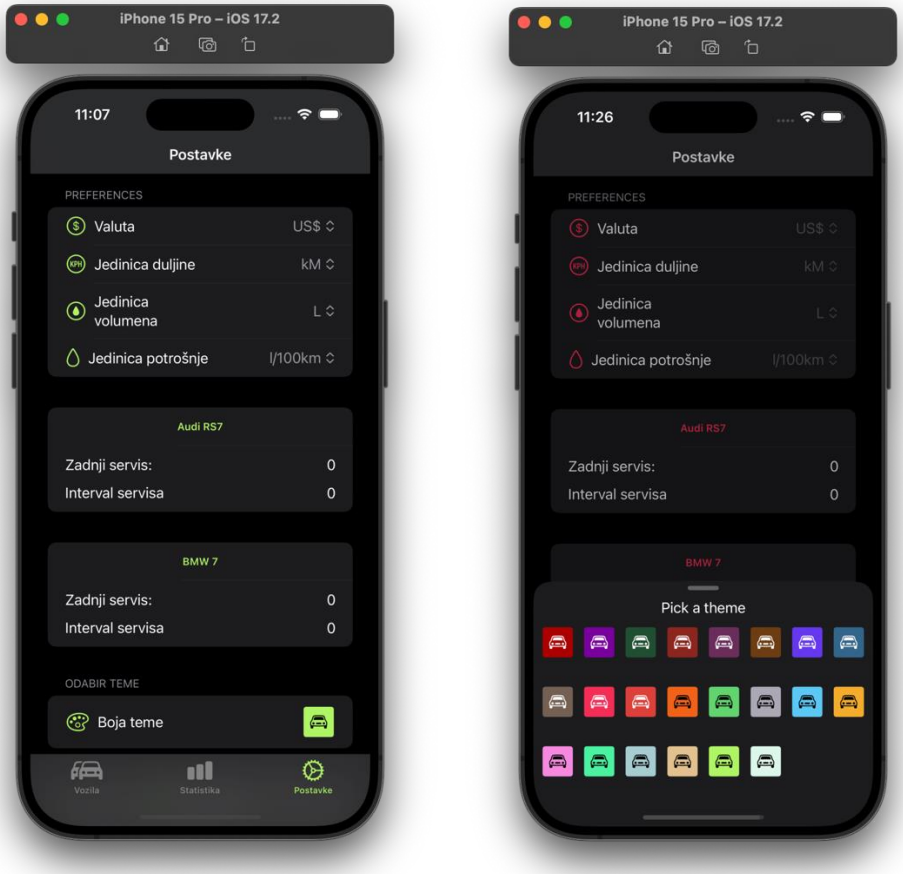
Izvor: Autor

Slika 8 prikazuje glavni ekran aplikacije u oba načina rada, tamni način i svijetli način. Aplikacija mora osigurati optimalnu vidljivost i upotrebljivost u svakom okruženju.

8.2. Prilagodba tematske boje

Prilagodba tematskih boja nije previše zastupljena značajka, mnoge aplikacije je nemaju. Međutim, ova značajka može biti vrlo korisna za održavanje zanimljivosti aplikacije za korisnike. Kada se korisnici umore od iste glavne boje, mogu je jednostavno promijeniti odabirom jedne od više od 20 mogućih boja. Ova mogućnost prilagodbe pruža korisnicima osjećaj kontrole i personalizacije, što može povećati njihovo zadovoljstvo i odanost aplikaciji.

Slika 9. Promjena tematske boje aplikacije



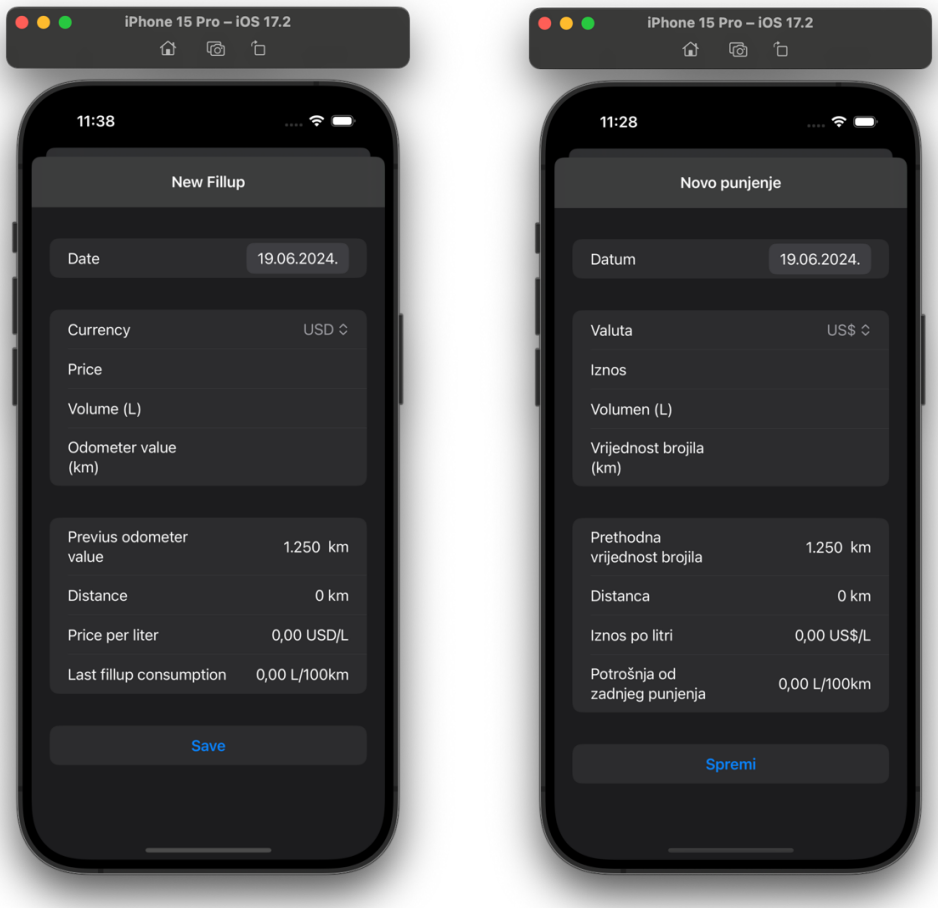
Izvor: Autor

Slika 9 prikazuje odabir boje u aplikaciji, gdje su vidljive dvije različite teme aplikacije. Na jednoj strani prikazana je jedna tema u pozadini birača boja, dok druga strana prikazuje drugu boju teme i kako se ona reflektira na aplikaciji. Ovaj prikaz jasno pokazuje kako promjena boje teme utječe na cjelokupni izgled i dojam aplikacije, ističući mogućnosti prilagodbe prema osobnim preferencijama korisnika.

8.2.3 Podrška za više jezika

Mnogi korisnici aplikacija imaju probleme s jezikom, budući da neki korisnici ne znaju engleski jezik. Lokalizacija je odličan način za prilagodbu jezika kroz cijelu aplikaciju. Štoviše, dodavanje novih jezika vrlo je jednostavno nakon što se uspostavi značajka lokalizacije. Ova funkcionalnost omogućuje korisnicima da koriste aplikaciju na svom materinjem jeziku, što znatno poboljšava njihovo iskustvo i olakšava korištenje aplikacije.

Slika 10. Podrška za engleski i hrvatski jezik



Izvor: Autor

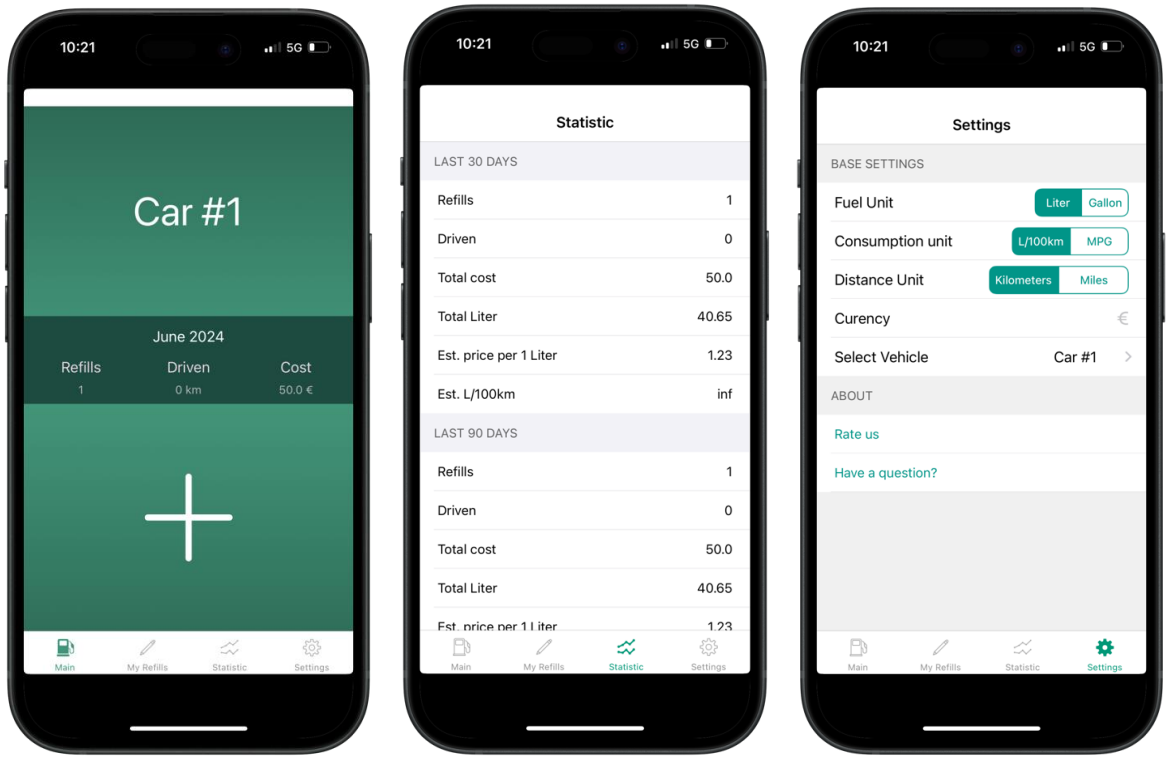
Slika 10 prikazuje aplikaciju lokaliziranu na dva jezika: s lijeve strane na engleskom jeziku, a s desne strane na hrvatskom jeziku. Lokalizacija doprinosi širenju baze korisnika i osigurava najbolje korisničko iskustvo za širu publiku.

8.2.3 Usporedba aplikacija

U današnje vrijeme gotovo je nemoguće stvoriti nešto potpuno novo. Umjesto toga, najbolji način za stvaranje uspješne aplikacije je uzeti ideju koja već postoji i učiniti je boljom, ljepšom, s više značajki i boljim performansama.

U svijetu razvoja aplikacija, ključno je usporediti vlastitu aplikaciju s tržišnim konkurentima. Time programeri prate trendove, korisničke preferencije i tehnološke inovacije koje oblikuju industriju.

Slika 11. Izgled konkurentske aplikacije



Izvor: Autor

Slika 11 prikazuje aplikaciju koja nema vizualnu privlačnost. Sučelje je neuredno i teško je shvatiti čemu aplikacija služi. Aplikacija ima samo jednu glavnu funkcionalnost, što ograničava njezinu korisnost i atraktivnost za širu publiku.

Slika 12. Izgled aplikacije Refuel

Izvor: Autor

Slika 12 prikazuje aplikaciju koja je vizualno vrlo privlačna i zanimljiva. Sučelje je korisnički orijentirano i intuitivno, omogućujući korisnicima lako snalaženje. Aplikacija nudi više funkcionalnosti i pruža veće mogućnosti prilagodbe, čime zadovoljava različite potrebe i preferencije korisnika.

Tablica 6. Usporedba s konkurentom aplikacijom

Značajke	**Refuel**	**Konkurencija**
1.	Praćenje potrošnje goriva	Praćenje potrošnje goriva
2.	Statistike i grafovi	Statistike
3.	Podržava različite mjerne jedinice	Podržava različite mjerne jedinice
4.	Tamni i svijetli način rada	-
5.	Lokalizacija	-
6.	Podsjetnik za servis	-
7.	Putni nalozi	-
8.	Uvod u aplikaciju	-
9.	Odabir tematske boje	-

Izvor: Autor

Tablica 6 prikazuje usporedbu značajki između aplikacije Refuel i konkurencije. Kao što se vidi iz tablice, aplikacija Refuel nudi značajno više funkcionalnosti u odnosu na konkurentnu aplikaciju.

9. DALJNI RAZVOJ APLIKACIJE

Razvoj aplikacija ne završava kada je prva verzija gotova, naprotiv, tada počinje kontinuirani proces unapređenja. Iako je aplikacija izrađena kao završni rad, njen potencijal nadilazi tu svrhu. Objavljivanjem aplikacije na trgovinu aplikacija (engl. *App Store*) otvara se mogućnost dodavanja kupnji unutar aplikacije, što može postati značajan izvor prihoda.

Isto tako uvijek postoji prostor za poboljšanje izgleda aplikacije i dodavanje novih značajki koje će je učiniti privlačnijom korisnicima. Svaka nova verzija može donijeti bolji dizajn, poboljšanu funkcionalnost i bolju optimizaciju, čime se povećava korisničko zadovoljstvo.

9.1 Dodavanje aplikacije na trgovinu

Dodavanje aplikacije na App Store bio bi izvrstan sljedeći korak jer otvara mogućnost zarade i dosezanja globalne publike.

Da bi aplikacija bila objavljena u trgovini, potrebno je dovršiti još ove korake:

1. Prijava u Apple Developer Program: Potrebno je registriranje i plaćanje godišnje naknade za pridruživanje Apple Developer Programu.
2. Testiranje aplikacije: Temeljito testiranje je ključno da bi se spriječilo što više neželjenih kvarova na aplikaciji, prije nego se uopće pošalje u trgovinu.
3. Priprema za predaju: Potrebno je poslati podatke, uključujući naziv aplikacije, opis, snimke zaslona i druge meta podatke.
4. Predaja aplikacije: Aplikacija se predaje na pregled. Ovaj proces može trajati nekoliko dana do tjedan dana. U slučaju odbijanja aplikacije, potrebno je ispuniti sve zahtjeve i zatim je ponovno poslati na pregled.
5. Objava aplikacije: Nakon odobrenja, aplikacija se može objaviti na App Store-u.

9.2 Dohvaćanje stanja brojila OBD-II adapterom

Dodavanje značajke dohvaćanja podataka o stanju brojila iz OBD-II adaptera u novu verziju aplikacije nudi nekoliko prednosti. Smanjuje potrebu za ručnim unosom podataka prilikom bilježenja punjenja goriva. Ovo poboljšanje ne samo da štedi vrijeme, već i minimizira mogućnost pogrešaka pri unosu podataka, osiguravajući točnost podataka.

Implementacija ove značajke uključuje nekoliko ključnih koraka. Prvo, odaberi kompatibilnog OBD-II adapter koji može komunicirati s širokim rasponom marki i modela automobila. Zatim integriranje funkcionalnost adaptera u aplikaciju, uspostavljajući sigurnu i pouzdanu vezu putem Bluetootha ili Wi-Fi mreže.

10. ZAKLJUČAK

Završni rad podijeljen je u dva glavna dijela, od kojih svaki obrađuje bitne aspekte razvoja i unaprjeđenja iOS aplikacije. Prvi dio razrađuje različite tehnologije i znanja potrebna za kreiranje iOS aplikacije, nudeći sveobuhvatan pregled alata, okvira i najboljih praksi koji čine temelj razvoja iOS aplikacija. Ovaj dio pokriva sve od osnova Swift programskog jezika do detalja o Xcode-u i alata za izradu korisničkog sučelja, naglašavajući važnost pridržavanja Apple-ovih smjernica za korisničko sučelje kako bi se osiguralo besprijekorno i intuitivno korisničko iskustvo.

Drugi dio rada usredotočuje se na aplikaciju, opisujući njezine značajke i funkcionalnosti te objašnjavajući zašto se izdvaja u svojoj kategoriji. Aplikacija nudi robustan set značajki za pružanje izvanrednog korisničkog iskustva. Rad također razmatra superiornost aplikacije u odnosu na slične na tržištu, navodeći njezin sveobuhvatan set funkcionalnosti, jednostavnost korištenja i pažnju prema potrebama korisnika kao ključne prednosti. Za razliku od mnogih konkurentskih aplikacija koje nude ograničene mogućnosti i nedostatak korisnički orijentiranog dizajna, ova aplikacija pruža idealno rješenje za praćenje i upravljanje potrošnjom goriva uz uvid u statistike i prilagodljive postavke.

Zaključno, ovaj završni rad pruža temeljit pregled tehnoloških i praktičnih aspekata razvoja iOS aplikacija, dok također predstavlja detaljnu studiju slučaja aplikacije bogate značajkama. Kombinirajući tehničku stručnost s pristupom usmjerenim na korisnike, rad pokazuje kako stvoriti iOS aplikaciju koja ne samo da zadovoljava, već i nadmašuje očekivanja korisnika. Uvidi i preporuke predstavljeni ovdje nude vrijedne smjernice za buduće razvojne projekte, osiguravajući da aplikacija nastavi evoluirati i ostati konkurentna na dinamičnom i stalno promjenjivom tržištu.

Izjava o autorstvu

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

Bana Josipa Jelačića 22/a, Čakovec

IZJAVA O AUTORSTVU

Završni/diplomski rad isključivo je autorsko djelo studenta 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, internetskih i drugih izvora) bez pravilnog citiranja. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom i nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, IVAN MUKTIĆ (ime i prezime studenta) pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ica završnog/diplomskog rada pod naslovom APLIKACIJA ZA PRAĆENJE POTROŠNJE GORIVA

te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:

(vlastoručni potpis)

Literatura

1. Xcode, https://developer.apple.com/xcode/, (Datum pristupa: 20.03.2024)
2. Apple eko sustav, https://techjourneyman.com/blog/apple-ecosystem-explained-2024/, (Datum pristupa: 20.03.2024)
3. Swift, https://developer.apple.com/swift/, (Datum pristupa: 20.03.2024)
4. SwiftUI, https://developer.apple.com/xcode/swiftui/, (Datum pristupa: 22.03.2024)
5. Combine, https://developer.apple.com/documentation/combine, (Datum pristupa: 22.03.2024)
6. MVVM, https://medium.com/@zebayasmeen76/mvvm-in-ios-swift-6afb150458fd, (Datum pristupa: 25.03.2024)
7. Core Data, https://developer.apple.com/documentation/coredata, (Datum pristupa: 03.04.2024)
8. Firebase, https://firebase.google.com, (Datum pristupa: 05.05.2024)
9. Korisničko sučelje, https://developer.apple.com/xcode/swiftui/, (Datum pristupa: 12.05)
10. Navigacija, https://developer.apple.com/documentation/swiftui/navigation, (Datum pristupa: 18.05)
11. Lokalizacija, https://developer.apple.com/documentation/xcode/localization, (Datum pristupa: 23.05)

Popis ilustracija