



ŠOLSKI CENTER CELJE
Srednja šola za kemijo, elektrotehniko in računalništvo
Pot na Lavo 22, 300 Celje

Odpiranje rampe centra varne vožnje z mobilno aplikacijo

Raziskovalna naloga s področja računalništva

Avtorji: Milan Jekić, Kenan Salkić, Alen Salobir

Mentor: Borut Slemenšek, univ. dipl. inž

Somentor: Tomaž Lotrič, mag. prom. zn.

Mestna občina Celje, Mladi za Celje

Celje, 2025

IZJAVA*

Mentor/-ica **Borut Sllemenšek** v skladu z 20. členom Pravilnika o organizaciji mladinske raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, zagotavljam, da je v raziskovalni nalogi z naslovom **Spletni sistem in aplikacija za nadzor rampe**, katere avtorji so **Milan Jekić, Kenan Salkić in Alen Salobir**:

- besedilo v tiskani in elektronski obliki istovetno,
- pri raziskovanju uporabljeno gradivo navedeno v seznamu uporabljene literature,
- da je za objavo fotografij v nalogi pridobljeno avtorjevo dovoljenje in je hranjeno v šolskem arhivu,
- da sme Osrednja knjižnica Celje objaviti raziskovalno nalogo v polnem besedilu na knjižničnih portalih z navedbo, da je raziskovalna naloga nastala v okviru projekta Mladi za Celje,
- da je raziskovalno nalogo dovoljeno uporabiti za izobraževalne in raziskovalne namene s povzemanjem misli, idej, konceptov oziroma besedil iz naloge ob upoštevanju avtorstva in korektnem citiranju,
- da smo seznanjeni z razpisni pogoji projekta Mladi za Celje.

Celje, 3. 4. 2025



Podpis mentorja

Podpis odgovorne osebe

*

POJASNILO

V skladu z 20. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje je potrebno podpisano izjavo mentorja (-ice) in odgovorne osebe šole vključiti v izvod za knjižnico, dovoljenje za objavo avtorja (-ice) fotografskega gradiva, katerega ni avtor (-ica) raziskovalne naloge, pa hrani šola v svojem arhivu.

KAZALO

1	UVOD.....	1
1.1	Opis/predstavitev problema	1
1.2	Cilji	1
2	ANDROID APLIKACIJA.....	2
2.1	Izdelava načrta	2
2.2	Android Studio.....	3
2.3	Programski jezik Kotlin	4
2.4	Različne verzije.....	5
2.5	Končna verzija aplikacija.....	6
2.5.1	Sistemske vmesnike (Android System APIs)	8
2.5.2	Kotlin koda	9
3	iOS APLIKACIJA	12
3.1	Izdelava načrta	12
3.2	Xcode 10.1	13
3.3	Programski jezik Swift.....	14
3.4	Različne verzije.....	15
3.5	Končna verzija aplikacije.....	16
3.5.1	Sistemske vmesnike (iOS System APIs)	16
4	APLIKACIJA NA RAČUNALNIKU	17
4.1	Definicija ciljev.....	17
4.2	Visual Studio Code (VS Code).....	18
4.3	Programski jeziki za izdelavo spletne aplikacije	18
4.3.1	HTML (HyperText Markup Language)	18
4.3.2	CSS (Cascading Style Sheets)	19
4.3.3	JavaScript	20
4.4	Aplikacija in verzije.....	21
5	ZAKLJUČEK	22
5.1	Hipoteze – potrjene ali ovržene	23
6	VIRI IN LITERATURA.....	24

Kazalo slik

Slika 1: Android Studio	3
Slika 2: Integrirano razvojno okolje Android Studio	4
Slika 3: Domača stran v1	6
Slika 4: Prijavna stran v1	6
Slika 5: Prijavna stran.....	7
Slika 6: Verifikacija strani	7
Slika 7: Zgodovina odpiranj	8
Slika 8: Domača stran.....	8
Slika 9: Android Studio – Kotlin koda	10
Slika 10: XCode	13
Slika 11: XCode IDE.....	13
Slika 12: Aplikacija na računalniku.....	18
Slika 13: Primer html kode za tabelo	19
Slika 14: Urejanje tabele s pomočjo CSS.....	20
Slika 15: Preverjanje vnosa s pomočjo JS	21

Povzetek

Raziskovalna naloga obravnava razvoj spletnega sistema in mobilnih aplikacij za nadzor parkirne zapornice. V nalogi so opisani procesi načrtovanja, implementacije in testiranja sistema, pri čemer sta uporabljena programska jezika Kotlin in Swift za mobilne aplikacije ter HTML, CSS in JavaScript za spletno aplikacijo. Glavni cilj je bil razviti varno in zanesljivo rešitev, ki omogoča nadzor zapornice prek brezžične povezave.

Ključne besede:

iOS, Android, Swift, Kotlin, podatkovna baza

Abstract

This research paper focuses on the development of a web-based system and mobile applications for gate control. It describes the planning, implementation, and testing phases, utilizing Kotlin and Swift for mobile apps and HTML, CSS, and JavaScript for the web application. The primary goal was to develop a secure and reliable solution for managing the gate via a wireless connection.

Key words:

iOS, Android, Swift, Kotlin, data base

Kratice in okrajšave

API – Application Programming Interface (programski vmesnik)

Programski vmesnik, ki omogoča komunikacijo med različnimi programskimi komponentami. API omogoča aplikacijam, da uporabljajo funkcionalnosti drugih aplikacij ali storitev.

IDE – Integrated Development Environment (integrirano razvojno okolje)

Programska oprema, ki zagotavlja vse potrebne funkcionalnosti za razvoj programske opreme. IDE običajno vključuje urejevalnik kode, orodja za razhroščevanje, testiranje in druge funkcije, ki olajšajo proces razvoja.

IntelliJ IDEA – priljubljeno integrirano razvojno okolje (IDE). Uporablja se predvsem za razvoj programske opreme v programskem jeziku Kotlin.

OTP – One-Time Password (enkratno geslo)

Varnostni mehanizem, ki omogoča uporabo enkratnega gesla za prijavo ali preverjanje identitete. OTP se pogosto uporablja v večstopenjskih varnostnih sistemih.

SMS – Short Message Service (kratka sporočila)

Tehnologija za pošiljanje kratkih besedilnih sporočil med mobilnimi napravami. Uporablja se predvsem za pošiljanje enkratnih gesel (OTP) ali drugih obvestil.

CRUD – Create, Read, Update, Delete (ustvari, preberi, posodobi, izbriši)

Osnovne operacije, ki jih aplikacija izvaja na podatkovnih virih. Pomenijo ustvarjanje novih zapisov, branje obstoječih, posodabljanje obstoječih in brisanje zapisov iz baze.

UI/UX – User Interface/User Experience (uporabniški vmesnik/uporabniška izkušnja)

UI se nanaša na zasnovo in izgled uporabniškega vmesnika aplikacije, medtem ko UX opisuje celotno izkušnjo uporabnika med interakcijo z aplikacijo ali storitvijo.

Wi-Fi – Wireless Fidelity (brežžična povezljivost)

Tehnologija za brezžično omrežno povezovanje naprav, ki omogoča prenos podatkov na kratke razdalje brez fizičnih povezav.

HTML – HyperText Markup Language

Spletni jezik za izdelavo ogrodja spletne strani.

CSS – Cascading Style Sheets

Jezik, ki skrbi za vizualno podobo in oblikovanje spletne aplikacije. Njegova naloga je, da uporabniškemu vmesniku doda estetsko vrednost.

JavaScript – spletni skriptni jezik za izdelavo dinamičnih in interaktivnih spletnih strani.

Frontend – del aplikacije, ki je viden uporabniku in omogoča interakcijo; sestavljen iz HTML, CSS in JavaScript.

Backend – strežniški del aplikacije, ki obdeluje zahteve uporabnika, upravlja podatkovno bazo in zagotavlja API-je.

1 UVOD

1.1 Opis/predstavitev problema

V današnjem času digitalizacije in avtomatizacije je učinkovito upravljanje dostopnih sistemov ključnega pomena za varnost in nadzor. Ena izmed takšnih rešitev je sistem za nadzor zapornice, ki omogoča nadzorovan dostop do določenih območij preko mobilnih aplikacij. Namen te raziskovalne naloge je razviti spletni sistem in aplikaciji za mobilne naprave z operacijskim sistemom Android ali iOS, ki bosta omogočali nadzor zapornice prek varne brezžične povezave. Raziskava vključuje analizo obstoječih rešitev, razvoj programske opreme in testiranje delovanja sistema v realnih pogojih.

1.2 Cilji

Cilji raziskovalne naloge so:

- razviti delujočo spletno in mobilno aplikacijo za nadzor zapornice;
- zagotoviti varno in učinkovito avtentikacijo uporabnikov;
- optimizirati sistem za hitro in zanesljivo delovanje.

Hipoteze

1. Implementacija OTP sistema za prijavo bo izboljšala varnost in zmanjšala nepooblaščen dostop.
2. Aplikacija bo omogočala hitro in stabilno povezavo z zapornico, kar bo omogočilo učinkovito upravljanje.
3. Uporabniška izkušnja bo izboljšana z intuitivnim in enostavnim uporabniškim vmesnikom.

2 ANDROID APLIKACIJA

2.1 Izdelava načrta

Pred začetkom kodiranja smo pripravili podroben načrt.

Definicija ciljev: aplikacija mora omogočati povezavo z zapornico preko Wi-Fi povezave/omrežja. Uporabnik mora imeti možnost ročnega odpiranja/zapiranja zapornice z uporabo gumba v aplikaciji. Prav tako mora biti implementirana varnostna zaščita, ki vključuje preverjanje pristnosti uporabnika, kar omogoča, da ima dostop do aplikacije in njenih funkcij le pooblaščen uporabnik.

Raziskava in analiza: pred začetkom razvoja smo naredili raziskavo obstoječih rešitev za upravljanje s sistemi za nadzor zapornic, da bi ugotovili, kako se podobni sistemi integrirajo z mobilnimi napravami in katere metode za varnost ter učinkovitost uporabi večina rešitev.

Programska oprema: izbor IDE za razvoj Android aplikacije je bil ključnega pomena. Izbran je bil Android Studio, saj omogoča napredno podporo za razvoj in testiranje aplikacij na različnih napravah ter sistemih.

Uporabniški vmesnik (UI/UX): pri načrtovanju preprostega in funkcionalnega uporabniškega vmesnika smo se osredotočili na uporabniško prijaznost ter intuitivnost. Glavni elementi vključujejo gumb za odprtje/zaprtje zapornice in zaznavo Wi-Fi omrežij, ki omogoča enostavno prepoznavanje razpoložljivih omrežij v okolici.

Varnostni vidiki: za varnost pri uporabi aplikacije smo se odločili za implementacijo dvostopenjskega načina preverjanja uporabnika in naprave. Prav tako smo poskrbeli za preverjanje omrežja in uporabo šifriranih povezav pri prenosu občutljivih podatkov.

Testni scenariji: za zagotavljanje pravilnega delovanja aplikacije smo pripravili testne scenarije, ki vključujejo preverjanje uspešnosti povezave v različnih omrežjih, testiranje odzivnosti rampe ob uporabi aplikacije, pa tudi simulacijo prekinitve povezave in obnovitve le-te.

2.2 Android Studio

Aplikacijo smo razvili v Android Studiu (verzija 2024.2.1) v programskem jeziku Kotlin, ki je uradno razvojno okolje (IDE) za Android aplikacije. Android Studio temelji na IntelliJ IDEA in ponuja širok nabor orodij za programiranje, testiranje ter optimizacijo mobilnih aplikacij. Zasnovan je posebej za razvoj programske opreme, ki deluje na operacijskem sistemu Android in omogoča enostavno povezovanje s simulatorji aplikacij, fizičnimi napravami ter različnimi API-ji.



Slika 1: Android Studio

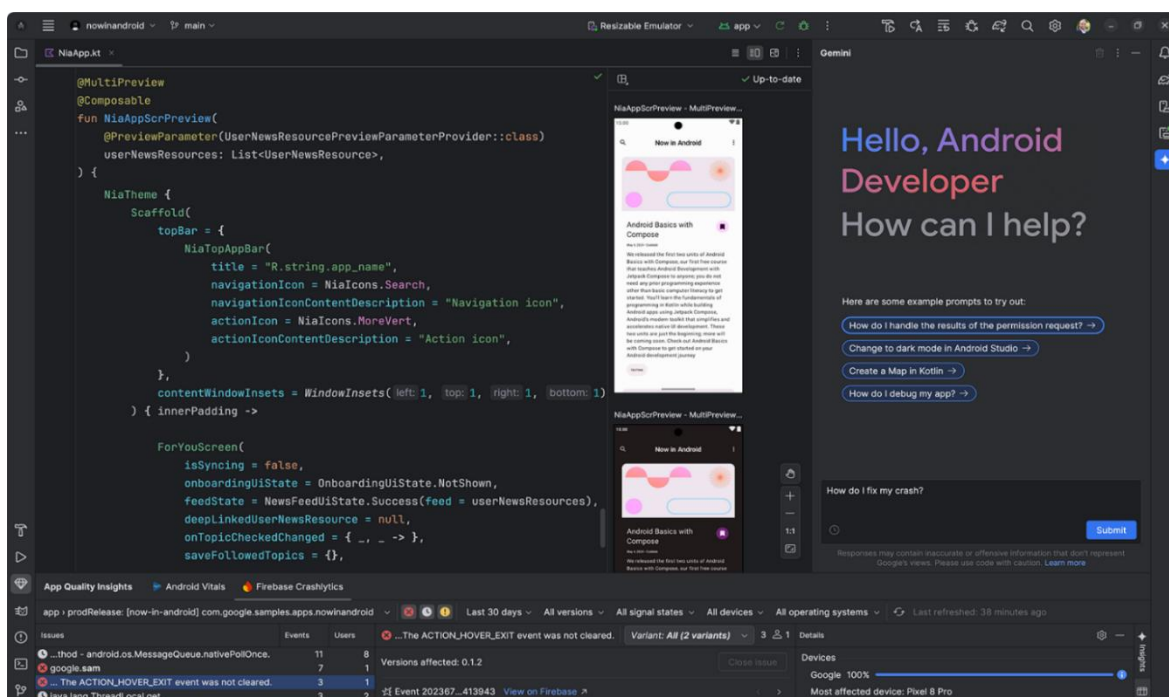
V tej raziskavi smo izdelali aplikacijo s podporo za naprave z operacijskim sistemom Android verzije 11.0 (API 26) ali novejšim. Android Studio omogoča razvijalcem, da prilagodijo aplikacijo različnim velikostim zaslonov, testirajo njeno delovanje na različnih virtualnih in fizičnih napravah ter odpravijo morebitne napake s pomočjo vgrajenega razhroščevalnika (debuggerja).

Pred pričetkom dela v Android Studiu smo morali izdelati izvedbeni načrt in pripraviti grafične osnutke aplikacije. Ta korak je bil ključen za oblikovanje osnovne ideje o funkcionalnostih aplikacije ter njenem vizualnem izgledu. Načrtovanje v začetni fazi razvoja omogoča boljšo organizacijo dela, hitreje prilagajanje spremembam in učinkovitejšo implementacijo posameznih komponent.

Android Studio ponuja številne funkcionalnosti, kot so vizualni urejevalnik uporabniškega vmesnika (UI designer), napredna analiza kode, simulatorji naprav, enostavno povezovanje

razvojnega okolja z mobilnimi napravami ter vgrajena orodja za testiranje, kar bistveno olajša razvojno delo in omogoča hitrejši razvoj kakovostnih aplikacij.

Med razvojem so se pojavile nekatere tehnične težave, kot so povezovanje Android Studio s fizično napravo in emulatorjem ter težave s kompatibilnostjo različnih verzij Androida. Te težave so bile deloma rešene s pomočjo uradne dokumentacije (**Android Developer Documentation** - Google) in dodatnih virov, kot so **Kotlinlang.org** (JetBrains) ter **Kotlin for Android Developers** (Leiva, A.).



Slika 2: Integrirano razvojno okolje Android Studio

2.3 Programski jezik Kotlin

Kotlin je statičen programski jezik, ki je popolnoma združljiv z Java in ga je razvilo podjetje JetBrains. Postal je uradni jezik za razvoj aplikacij za Android in se pogosto uporablja za mobilne aplikacije, predvsem zaradi svoje enostavnosti in zmogljivosti. Kotlin ponuja številne prednosti v primerjavi z Java, kot so:

- **Kratkost kode** – Kotlin omogoča pisanje manj kode za enake naloge, kar povečuje produktivnost razvijalcev.
- **Zanesljivost** – Kotlin vključuje napredne mehanizme za preprečevanje napak, kot je »null safety«, ki zmanjšujejo možnosti za napake ob izvajanju aplikacije.

- **Enostavna integracija z Javao** – Kotlin je popolnoma združljiv z obstoječo Java kodo, kar omogoča enostavno migracijo obstoječih projektov na ta jezik.
- **Funkcionalne lastnosti** – Kotlin podpira funkcionalne paradigme, kot so visoke funkcije, kar razvijalcem omogoča bolj fleksibilen in izrazit način programiranja.
- **Vsestranskost** – Kotlin se uporablja v Android razvoju, vendar je primeren tudi za razvoj »backend« aplikacij, spletnih aplikacij in podatkovnih znanstvenih projektov, zahvaljujoč svoji zmogljivosti in prilagodljivosti.

Pri pisanju kode v Kotlinu smo si veliko pomagali z različnimi viri, ki so nam omogočili boljše razumevanje jezika in njegovih funkcionalnosti. Največ informacij smo dobili iz **Android Developer Documentation** (Google), **Kotlinlang.org** (JetBrains) ter **Kotlin for Android Developers** (Leiva, A.), ki so bili glavna pomoč pri razvoju aplikacije in razumevanju osnov ter naprednih konceptov.

Poleg tega je **Kotlin Course – Tutorial for Beginners** pripomoček, ki je pomagal približati sintakso jezika s praktičnimi primeri in razlago, ki so olajšali delo. Med razvojem aplikacije je prišla zelo prav tudi spletna stran **w3schools**, kjer lahko hitro dobimo odgovore na vprašanja, povezana s sintakso in osnovnimi funkcionalnostmi jezika.

Ti viri so nam omogočili, da smo kljub omejitvam pri povezovanju Android Studia s fizičnim telefonom in emulatorjem uspešno razvili funkcionalno rešitev za upravljanje rampe.

2.4 Različne verzije

Pri izdelavi aplikacije je od začetka do konca nastalo več verzij. Slika na naslednji strani levo prikazuje prvo verzijo mobilne aplikacije, kjer smo se osredotočili predvsem na implementacijo izgleda aplikacije. Sledile so verzije z ostalimi nadgradnjami in izboljšavami. Za funkcionalnost mobilne aplikacije je bila dodana še povezava s podatkovno bazo in, kot smo že zapisali, avtentikacija za preverjanje uporabnika.



Slika 4: Prijavna stran v1



Slika 3: Domača stran v1

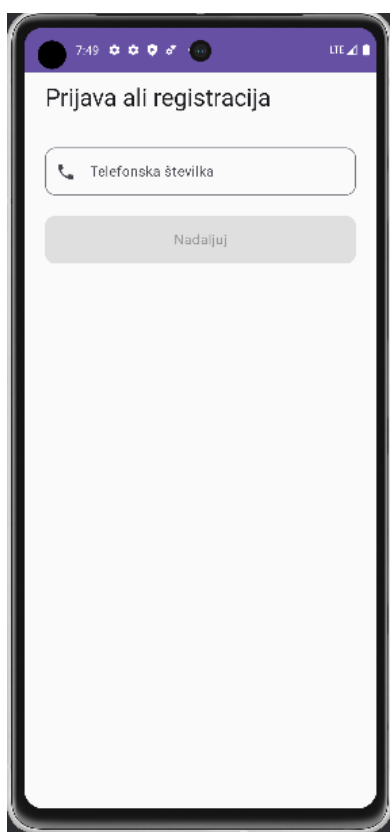
2.5 Končna verzija aplikacija

Pred končanjem zadnje verzije aplikacije smo se odločili za sistem **OTP SMS preverjanja**. Ta sistem deluje po naslednjem postopku.

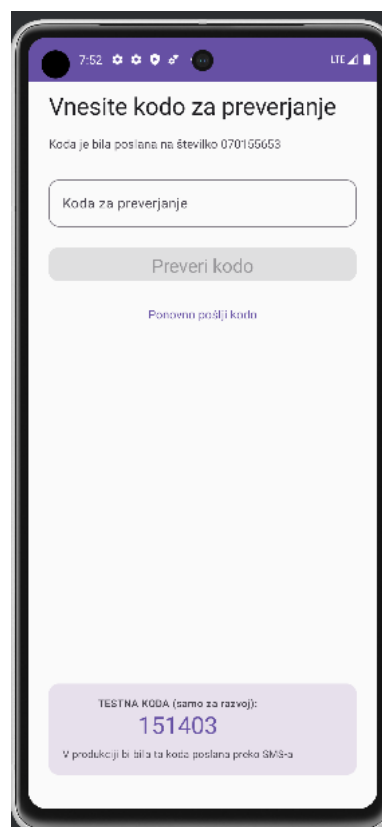
1. **Vnos telefonske številke:** uporabnik vpiše svojo telefonsko številko v aplikacijo.
2. **Pošiljanje številke preko API-ja:** aplikacija pošlje vpisano številko v izbrani vključeni OTP sistem (na predhodno vpisano številko) preko njihovega API-ja ob pritisku na gumb "Nadaljuj".
3. **Preverjanje OTP kode:** po prejemu kode na vpisano telefonsko številko uporabnik vpiše kodo v aplikacijo in pritisne "Preveri kodo". Če se koda ujema, aplikacija preusmeri uporabnika na domači zaslon aplikacije. V nasprotnem primeru prikaže napako (v testnem okolju je koda prikazana neposredno v aplikaciji).

4. **Shranjevanje podatkov v podatkovno bazo:** po uspešni prijavi se podatki o uporabniku shranijo v podatkovno bazo. Na domačem zaslonu se prikažejo podatki o uporabniku, vključno z imenom (nastavljeno na privzeto "Uporabnik"), telefonsko številko in veljavnostjo računa, ki je v osnovi nastavljena na en mesec od prve prijave. Shranjujejo so tudi podatki o datumu in času obiska vsakega uporabnika ob njegovem vходу in izhodu.
5. **Dodatna varnost:** za zagotavljanje varnosti pri uporabi funkcionalnosti za odpiranje zapornice mora biti uporabnik povezan na omrežje Wi-Fi, ki je vnaprej določeno v aplikaciji. To povečuje varnost celotnega sistema.

V spodnjem delu aplikacije se nahaja navigacijska vrstica, ki omogoča preklapljanje med različnimi zavihki, vključno z **Zgodovino**. Ta zavihek nam prikaže podatke vseh vходов in izhodov glede na datum in čas teh dogodkov.



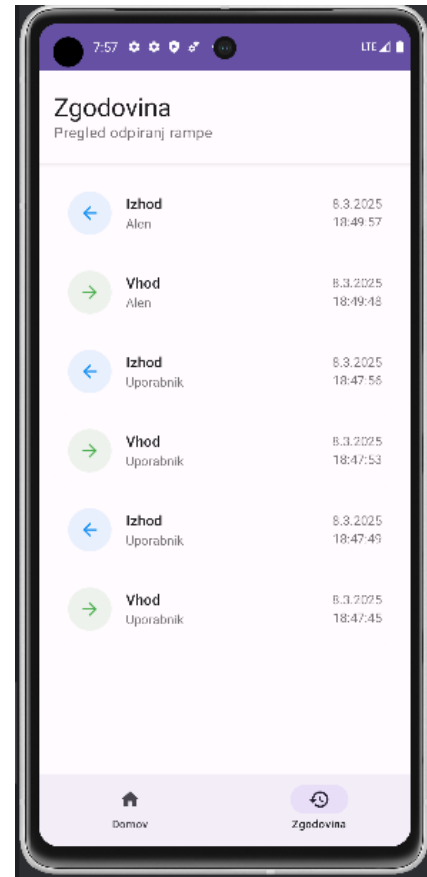
Slika 5: Prijavna stran



Slika 6: Verifikacija strani



Slika 8: Domača stran



Slika 7: Zgodovina odpiranj

2.5.1 Sistemski vmesniki (Android System APIs)

Aplikacija vključuje naslednje sistemske vmesnike (API-je).

Supabase API

Aplikacija uporablja Supabase kot backend, kar vključuje dostop do podatkovne baze in avtentikacijo.

Supabase omogoča izvajanje operacij, kot so pridobivanje uporabnikov, posodabljanje podatkov o uporabnikih, beleženje dogodkov (vstopi/izhodi) in upravljanje z uporabniškimi podatki.

PostgREST API

Supabase uporablja PostgREST za izpostavljanje RESTful API-jev, ki omogočajo izvajanje CRUD (Create, Read, Update, Delete) operacij na podatkovnih tabelah.

Aplikacija uporablja te API-je za interakcijo z več različnimi tabelami v podatkovni bazi, kjer so shranjeni različni podatki o uporabnikih.

GoTrue API

Supabase vključuje GoTrue za upravljanje avtentikacije uporabnikov, kar omogoča prijavo, registracijo in upravljanje sej.

Aplikacija uporablja GoTrue za preverjanje telefonskih števil in upravljanje uporabniških računov.

Klici za omrežno povezljivost

Aplikacija preverja omrežno povezljivost in stanje WiFi povezave, kar je ključno za delovanje funkcionalnosti, ki zahtevajo dostop do interneta.

Klici za obdelavo podatkov

Za obdelavo in posodabljanje podatkov so v aplikaciji uporabljene različne metode vključno z asinhronimi klici za pridobivanje ter posodabljanje podatkov o uporabnikih in dogodkih. Te API-je aplikacija uporablja za interakcijo s podatkovno bazo, upravljanje uporabnikov in zagotavljanje funkcionalnosti, kot so vstopi ter izhodi.

2.5.2 Kotlin koda

Kot je bilo že omenjeno, je Kotlin dokaj enostaven jezik. Na sliki na naslednji strani je prikazan del kode, ki je bila uporabljena v naši aplikaciji. Prikazana koda predstavlja zahtevnejši del kode, zato jo tudi na kratko pokomentiramo.

Konstanta `REQUIRED_SSID` (zahtevan_SSID):

- V njej je shranjeno ime WiFi omrežja, na katerega se želimo povezati (npr. "WifiHome2").

Spremenljivka `isConnectedToRequiredWifi` (jePovezanNaZahtevanWifi):

- V spremenljivki, ki je namenjena branju v aplikaciji, je shranjena vrednost `true` ali `false` glede na to, ali je naprava povezana s točno določenim omrežjem ali ne.
- Najprej je poklicana metoda `getConnectedWifiName()`, ki poskuša pridobiti ime omrežja.
- Če je pridobivanje imena omrežja (SSID) uspešno, se le-to primerja z `REQUIRED_SSID`.

```
17
18 class WifiControllerImpl(private val context: Context) : WifiController {
19     companion object {
20         private const val REQUIRED_SSID = "WifiHome2"
21     }
22
23     override val isConnectedToRequiredWifi: Boolean
24     get() {
25         val name = getConnectedWifiName() ?: return false
26         return name.trim().equals(REQUIRED_SSID, ignoreCase = true)
27     }
28
29     override fun getRequiredWifiName(): String = REQUIRED_SSID
30
31     override fun getConnectedWifiName(): String? {
32         if (!hasLocationPermission() || !isLocationEnabled()) return "Neznan WiFi"
33         val wifiManager = context.getSystemService(Context.WIFI_SERVICE) as WifiManager
34         if (!wifiManager.isWifiEnabled && Build.VERSION.SDK_INT < Build.VERSION_CODES.Q) return null
35         val connectionInfo = wifiManager.connectionInfo ?: return null
36         if (connectionInfo.networkId == -1) return "Neznan WiFi"
37         val rawSsid = connectionInfo.ssid ?: return null
38         val cleanSsid = rawSsid.removeSurrounding(delimiter: "\\*")
39         return if (cleanSsid.isEmpty() || cleanSsid == "<unknown ssid>") "Unknown WiFi" else cleanSsid
40     }
41
42     private fun hasLocationPermission(): Boolean =
43         ContextCompat.checkSelfPermission(context, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED
44 }
```

Slika 9: Android Studio – Kotlin koda

Metoda `getConnectedWifiName()` (pridobiImePovezanegaWifiOmrežja):

- Najprej preveri, ali ima aplikacija **dovoljenje za dostop do lokacije** in če je **lokacija sploh vklopljena**. To preverjanje opravi z dvema metodama (`hasLocationPermission()` in `isLocationEnabled()`). Če pogoja nista izpolnjena, vrne "Neznan WiFi". V novejših verzijah androida je to dovoljenje obvezno.
- Drugi preverjanje ugotavlja, če je WiFi izklopljen in je različica Androida nižja od Q. V obeh primerih metoda vrne vrednost `null`, kar predstavlja `false`.
- S tem je preverjena ustreznost okolja v napravi.

- Nato iz `wifiManager.connectionInfo` pridobi podatke o trenutni povezavi.
- Če je vrednost v `connectionInfo.networkId` enaka -1, naprava ni povezana v WiFi in vrne odgovor "Neznan WiFi", sicer iz `connectionInfo.ssid` odstrani morebitne narekovaje.
- Če je rezultat prazen ali je vsebina "<unknown ssid>", vrne "Unknown WiFi", sicer vrne ime povezanega omrežja (SSID).

Metoda `hasLocationPermission()` (imaDovoljenjezaLokacijo)

- Preveri, ali ima aplikacija dovoljenje `ACCESS_FINE_LOCATION`, ki predstavlja dovoljenje za branje imen povezanih omrežij.

Metoda `isLocationEnabled()` (LokacijaVključena):

- V sistemskih nastavitvah (`LocationManager`) v operacijskem sistemu Android) preveri, ali je vključena lokacija.

3 iOS APLIKACIJA

3.1 Izdelava načrta

Pred začetkom kodiranja smo pripravili podroben načrt.

Definicija ciljev: aplikacija mora omogočati povezavo z zapornico preko Wi-Fi povezave/omrežja. Uporabnik mora imeti možnost ročnega odpiranja/zapiranja zapornice z uporabo gumba v aplikaciji. Prav tako mora biti implementirana varnostna zaščita, ki vključuje preverjanje pristnosti uporabnika, kar omogoča, da ima dostop do aplikacije in njenih funkcij le pooblaščen uporabnik.

Raziskava in analiza: pred začetkom razvoja smo naredili raziskavo obstoječih rešitev za upravljanje s sistemi za nadzor zapornic, da bi ugotovili, kako se podobni sistemi integrirajo z mobilnimi napravami in katere metode za varnost ter učinkovitost uporabi večina rešitev.

Programska oprema: izbor IDE za razvoj iOS aplikacije je bil ključnega pomena. Izbran je bil Xcode 10.1, saj omogoča napredno podporo za razvoj in testiranje aplikacij na različnih napravah ter sistemih.

Uporabniški vmesnik (UI/UX): pri načrtovanju preprostega in funkcionalnega uporabniškega vmesnika smo se osredotočili na uporabniško prijaznost ter intuitivnost. Glavni elementi vključujejo gumb za odprtje/zaprtje zapornice in zaznavo Wi-Fi omrežij, ki omogoča enostavno prepoznavanje razpoložljivih omrežij v okolici.

Varnostni vidiki: za varnost pri uporabi aplikacije smo se odločili za implementacijo dvostopenjskega načina preverjanja uporabnika in naprave. Prav tako smo poskrbeli za preverjanje omrežja in uporabo šifriranih povezav pri prenosu občutljivih podatkov.

Testni scenariji: za zagotavljanje pravilnega delovanja aplikacije smo pripravili testne scenarije, ki vključujejo preverjanje uspešnosti povezave v različnih omrežjih, testiranje odzivnosti rampe ob uporabi aplikacije, pa tudi simulacijo prekinitve povezave in obnovitve le-te.

3.2 Xcode 10.1

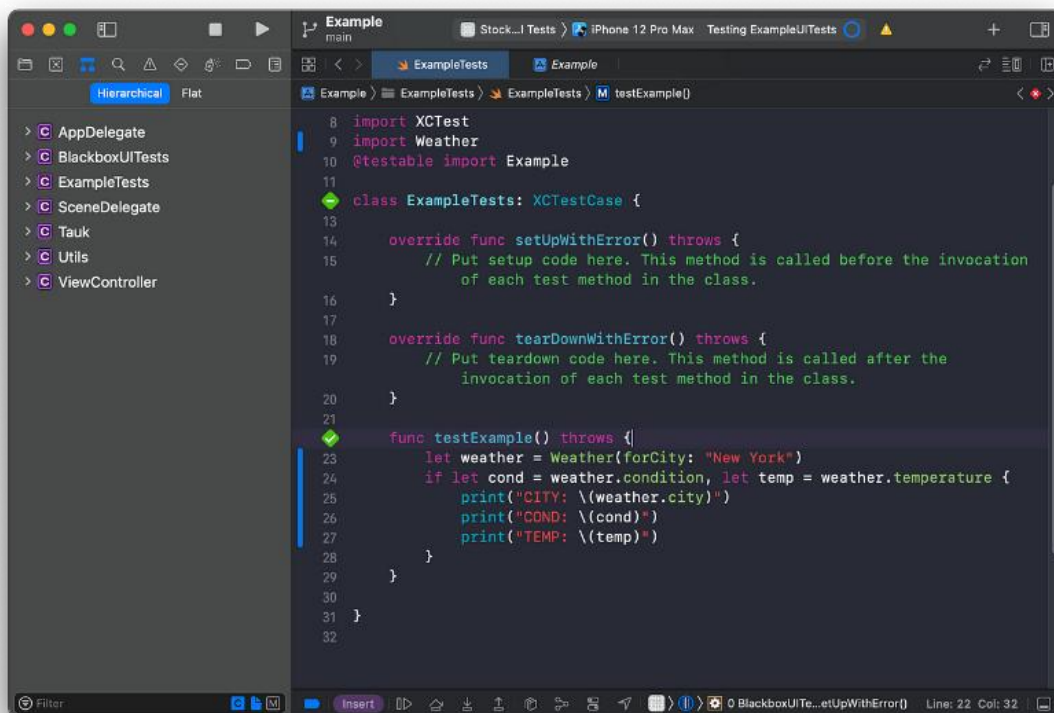
Aplikacija je bila razvita v **Xcode 10.1**, ki je uradno razvojno okolje (IDE) za iOS aplikacije. Xcode temelji na programskem jeziku **Swift** in ponuja širok nabor orodij za programiranje, testiranje ter optimizacijo mobilnih aplikacij. Zasnovan je posebej za razvoj programske opreme, ki deluje na operacijskem sistemu **iOS**, in omogoča enostavno povezovanje s simulatorji aplikacij, fizičnimi napravami ter različnimi **API-ji**.



Slika 10: XCode

V tej raziskavi smo izdelali aplikacijo s podporo za naprave z **iOS verzijo 12.0 ali novejšim**. Xcode omogoča razvijalcem, da prilagodijo aplikacijo različnim velikostim zaslonov, testirajo njeno delovanje na različnih virtualnih in fizičnih napravah ter odpravijo morebitne napake s pomočjo vgrajenega razhroščevalnika (**debuggerja**).

Pred pričetkom dela v Xcode smo morali izdelati izvedbeni načrt in pripraviti grafične osnutke aplikacije. Ta korak je bil ključen za oblikovanje osnovne ideje o funkcionalnostih aplikacije ter njenem vizualnem izgledu. Načrtovanje v začetni fazi razvoja omogoča boljšo



Slika 11: XCode IDE

organizacijo dela, hitreje prilagajanje spremembam in učinkovitejšo implementacijo posameznih komponent.

Xcode ponuja številne funkcionalnosti, kot so **vizualni urejevalnik uporabniškega vmesnika (Storyboard)**, napredna analiza kode, simulatorji naprav, enostavno povezovanje razvojnega okolja z mobilnimi napravami ter vgrajena orodja za testiranje. Te funkcionalnosti bistveno olajšajo razvojno delo in omogočajo hitrejši razvoj kakovostnih aplikacij.

Pri delu so se pojavile nekatere tehnične težave, kot so omejitve **starejše verzije Xcode (10.1)** in težave pri testiranju aplikacije na simulatorju. Za reševanje teh težav smo uporabili različne viri, kot so **Apple Developer Documentation** (Apple Inc.), **Ray Wenderlich Tutorials** in **Hacking with Swift** (Hudson, P.), ki so nudili dragocene informacije za odpravljanje težav in optimizacijo kode.

3.3 Programski jezik Swift

Swift je **sodoben programski jezik**, ki ga je razvilo podjetje **Apple**. Postal je **uradni jezik za razvoj aplikacij za iOS** in se pogosto uporablja za mobilne aplikacije, predvsem zaradi svoje enostavnosti in zmogljivosti. Swift ponuja številne prednosti v primerjavi z drugimi jeziki, kot so:

- **Kratkost kode** – Swift omogoča pisanje manj kode za enake naloge, kar povečuje produktivnost razvijalcev.
- **Zanesljivost** – Swift vključuje napredne mehanizme za preprečevanje napak, kot je »null safety«, ki zmanjšujejo možnosti za napake ob izvajanju aplikacije.
- **Enostavna integracija z Objective-C** – Swift je popolnoma združljiv z obstoječo Objective-C kodo, kar omogoča enostavno migracijo obstoječih projektov na ta jezik.
- **Funkcionalne lastnosti** – Swift podpira funkcionalne paradigme, kot so visoke funkcije, kar razvijalcem omogoča bolj fleksibilen in izrazit način programiranja.

- **Vsestranskost** – Swift se uporablja v iOS razvoju, vendar je primeren tudi za razvoj **macOS, watchOS in tvOS aplikacij**, zahvaljujoč svoji zmogljivosti in prilagodljivosti.

Pri pisanju kode v Swiftu smo si veliko pomagali z različnimi viri, ki so omogočili boljše razumevanje jezika in njegovih funkcionalnosti. Opisani so bili že zgoraj, kjer je bilo omenjeno reševanje težav, ki so se pojavile pri delu. Ti viri so nam pomagali tudi pri razvoju aplikacije in razumevanju osnov ter naprednih konceptov.

Swift Tutorial – Full Course for Beginners pa je bil nadvse praktičen pripomoček, ki nam je pomagal približati sintakso jezika s **praktičnimi primeri in razlago**, kar je zelo olajšalo delo. V pomoč nam je bila tudi spletna stran **w3schools**, ki je bila že omenjena, saj je tam mogoče dobiti hitre odgovore na vprašanja, povezana s sintakso in osnovnimi funkcionalnostmi jezika.

Kljub omejitvam, ki so povezane s **starejšo verzijo Xcode (10.1)** in težavami s **testiranjem aplikacije na simulatorju**, smo s pomočjo teh virov uspešno izdelali aplikacijo s funkcionalno rešitvijo za upravljanje zapornice.

3.4 Različne verzije

Po opravljenem skupnem načrtovanju mobilnih aplikacij za oba operacijska sistema smo se tudi tu najprej osredotočili predvsem na implementacijo izgleda aplikacije, da bi dobili jasen vpogled v vizualno podobo in uporabniško izkušnjo. Po končanem oblikovanju smo se lotil pisanja kode, pri tem pa je hitro vzniknila prva težava. Zaradi zastarele opreme, na kateri je nastajala naša aplikacija, ni bilo mogoče pravilno pognati simulatorja v Xcode. To je onemogočalo spremljanje delovanja aplikacije v realnem času in posledično tudi testiranje njenega delovanja. Zaradi teh tehničnih omejitev ni bilo mogoče preveriti, kaj se dogaja v programu, kar je otežilo nadaljnji razvoj in odpravljanje morebitnih logičnih napak, ki jih ob prevajanju aplikacije prevajalnik ne more javiti.

Upamo, da nam v kratkem uspe prevesti aplikacijo na novejšem MAC sistemu.

3.5 Končna verzija aplikacije

Da smo zagotovili enako funkcionalnost aplikacije kot pri Androidu, smo se tudi tu odločili za sistem **OTP SMS preverjanja**. Postopek delovanja tega sistema je bil opisan že v prejšnjem poglavju.

Aplikacija je pripravljena za mobilne naprave z operacijskim sistemom iOS, trudili pa smo se, da bi bila vizualno in izkustveno čimbolj podobna aplikaciji, ki teče na napravah z operacijskim sistemom Android.

3.5.1 Sistemski vmesniki (iOS System APIs)

Aplikacija vključuje naslednje systemske vmesnike (API-je).

CoreLocation API: aplikacija uporablja CoreLocation za preverjanje omrežne povezljivosti in stanja Wi-Fi povezave, kar je ključno za delovanje funkcionalnosti, ki zahtevajo dostop do interneta.

SQLite API: aplikacija uporablja SQLite za shranjevanje podatkov o uporabnikih in zgodovini vhodov/izhodov.

NSURLSession API: aplikacija uporablja URLSession za izvajanje omrežnih klicev, kot so pošiljanje telefonske številke na OTP API in preverjanje OTP kode.

UserNotifications API: aplikacija uporablja UserNotifications za obveščanje uporabnika o uspešni prijavi ali napakah.

Te API-je aplikacija uporablja za interakcijo s podatkovno bazo, upravljanje uporabnikov in zagotavljanje funkcionalnosti, kot so vstopi in izhodi.

4 APLIKACIJA NA RAČUNALNIKU

Spletna aplikacija za upravljanje nadzora zapornice je integralni del celovitega sistema za nadzor dostopa in upravljanja uporabnikov. Glavna naloga je zagotavljanje enotne platforme, kjer se podatki o zaposlenih in prisotnih osebah centralizirajo, urejajo in sinhronizirajo. Aplikacija je zasnovana tako, da omogoča preprost in učinkovit način upravljanja z uporabniki ter zagotavlja nemoteno komunikacijo med spletnim vmesnikom, mobilnimi aplikacijami in podatkovno bazo. Povezava s podatkovno bazo omogoča hitro obdelavo podatkov, medtem ko se podatki prikazujejo v uporabniku prijaznem spletnem vmesniku. Spletna aplikacija je ključna točka za dodajanje novozaposlenih, posodabljanje obstoječih podatkov in prikaz ter spremljanje prisotnosti kot tudi prikazu statističnih pregledov. Aplikacija mora biti povezana tudi z aplikacijama za mobilne naprave z operacijskima sistemoma Android in iOS, kar omogoča administratorju lažji pregled nad sistemom oz. nadziranje prisotnosti. Za razvoj aplikacije smo uporabili Visual Studio Code.

4.1 Definicija ciljev

- Enostavno upravljanje uporabnikov: omogočanje dodajanja, urejanja in brisanja zaposlenih ter prisotnih oseb.
- Sinhronizacija podatkov: samodejno posodabljanje podatkov med spletno aplikacijo, mobilnimi aplikacijami in bazo podatkov.
- Pregled nad prisotnostjo: prikaz seznama prisotnih oseb z možnostjo ažurnega posodabljanja podatkov v primeru sprememb.
- Centralizirana baza: baza podatkov prisotnih, za lažje upravljanje celotnega sistema.
- Uporabniška prijaznost: oblikovanje preprostega, estetsko dovršenega in intuitivnega vmesnika za učinkovito in enostavno uporabo.
- Podpora več napravam: povezljivost z mobilnimi aplikacijami za Android in iOS, ki uporabljata iste podatke in omogočata enako funkcionalnost uporabnikom (med mobilnima aplikacijama).



Slika 12: Aplikacija na računalniku

4.2 Visual Studio Code (VS Code)

Je zmogljivo in prilagodljivo integrirano razvojno okolje (IDE), ki ga je razvilo podjetje Microsoft. Gre za odprtokodno orodje, ki je na voljo brezplačno in omogoča razvoj različnih vrst aplikacij – od spletnih, mobilnih, strežniških do podatkovnih projektov. Zaradi svoje hitrosti, razširjenosti in široke podpore za številne programske jezike je izjemno priljubljeno med razvijalci. Pri razvoju te spletne aplikacije smo uporabljali Visual Studio Code 1.86.2.

4.3 Programski jeziki za izdelavo spletne aplikacije

4.3.1 HTML (HyperText Markup Language)

Je temeljni jezik za izdelavo strukture spletnih strani in definira osnovno ogrodje aplikacije. Deluje kot okostje aplikacije, pri kateri se določajo elementi, ki jih uporabnik vidi in preko njih izvaja interakcije. V tej aplikaciji HTML omogoča prikaz podatkov, uporabniških kontrol in organizacijo vsebine.

Kako se HTML uporablja v tej aplikaciji?

HTML določa osnovno postavitev elementov, kot so: navigacijski meni za preklapljanje med zavihki (“Prisotni”, “Zaposleni”, “Vsi”), tabela za prikaz podatkov, gumbi za dodajanje, urejanje in brisanje uporabnikov, obrazci za vnos podatkov ...

HTML omogoča vnos podatkov, ki se nato pošljejo backend aplikaciji in shranijo v SQL bazo.

Povezava z drugimi tehnologijami:

HTML povezuje CSS za oblikovanje in JavaScript za interaktivnost ter omogoča klicanje API-jev za delo s podatki.

```
<div id="dodaj-zaposlenega-div" style="display: none;">
  <button id="dodaj-zaposlenega" class="btn">Dodaj Zaposlenega</button>
</div>

<table id="data-table">
  <thead>
    <tr>
      <th>Ime</th>
      <th>Priimek</th>
      <th>EMŠO</th>
      <th>Telefon</th>
      <th>Čas prihoda</th>
      <th>Kategorija</th>
      <th>Dejanja</th>
    </tr>
  </thead>
  <tbody>
```

Slika 13: Primer html kode za tabelo

4.3.2 CSS (Cascading Style Sheets)

CSS skrbi za vizualno podobo aplikacije in omogoča, da je uporabniški vmesnik estetsko dovršen, pregleden in odziven. Omogoča ločevanje vsebine (HTML) od predstavitve (CSS), kar olajša vzdrževanje in nadgradnjo aplikacije.

Kako se CSS uporablja v tej aplikaciji?

- **Oblikovanje elementov:** CSS določa barve, velikosti pisav, razmike med elementi, obrobe in postavitev.
- **Stilno oblikvanje tabel:** CSS skrbi, da so podatki jasno in pregledno prikazani z uporabo obrob, razmikov ter barv ozadja.

- **Animacije in učinki:** CSS dodaja vizualne učinke, kot so spremembe barve gumbov ob prehodu z miško (hover) in poudarki interaktivnih elementov.

```
#data-table {
  width: 100%;
  border-collapse: collapse;
}

#data-table th, #data-table td {
  border: 1px solid #bdc3c7;
  padding: 8px;
  text-align: left;
}

#data-table th {
  background-color: #ecf0f1;
}
```

Slika 14: Urejanje tabele s pomočjo CSS

4.3.3 JavaScript

JavaScript je spletni jezik, ki omogoča interaktivnost in dinamičnost spletne aplikacije. Uporablja se za izvajanje logike na strani odjemalca in za komunikacijo s strežnikom preko HTTP zahtev.

Kako se JavaScript uporablja v tej primeru?

- **Upravljanje dogodkov:** JavaScript omogoča izvajanje funkcij ob kliku na gumbe, spremembi podatkov ali izbiri zavihkov.
- **Dinamičen prikaz podatkov:** posodablja vsebino strani brez ponovnega nalaganja.
- **Komunikacija z backendom:** JavaScript uporablja fetch() metodo za pošiljanje zahtev (GET, POST, PUT, DELETE) na API-je, ki upravljajo podatke v SQL bazi. Preverja tudi pravilnost vnesenih podatkov pred pošiljanjem na strežnik (npr. dolžina EMŠO in številke pri telefonu).

```
const ime = document.getElementById('ime').value;
const priimek = document.getElementById('priimek').value;
const emso = document.getElementById('emso').value;
const telefon = document.getElementById('telefon').value;

if (!ime || !priimek || !emso || !telefon) {
  alert('Vsa polja morajo biti izpolnjena.');
```

Slika 15: Preverjanje vnosa s pomočjo JS

4.4 Aplikacija in verzije

Aplikacija na računalniku bo omogočala pregled prisotnosti in vodenje statistike gostov kot tudi zaposlenih. V meniju na levi strani so omogočene različne funkcije aplikacije od tekočih pregledov prisotnih uporabnikov poligona, zaposlenih ali celotnih seznamov vpisanih uporabnikov. Pomemben je predvsem pregled uporabnikov, ki uporabljajo poligon za urjenje. Pregledi uporabnikov so opremljeni z dodatnimi filtri za lažje iskanje po prikazanih podatkih.

Ker se podatki o uporabnikih lahko tudi spremenijo ali dopolnjujejo, je le-to omogočeno skrbniku sistema. Tu gre predvsem za podatke o zaposlenih in seveda veljavnost dostopa za goste, ki ga je mogoče na željo uporabnika tudi podaljšati (vnaprej določen čas dostopa za goste je 1 mesec). Spet drugačni so pogoji za goste – inštruktorje.

Pripravljeni so tudi statistični pregledi uporabnikov, njihovih dostopov, časa uporabe poligona... Slednji se lahko seveda še dopolnijo glede na potrebe.

V toku izdelave aplikacije je bilo izdelanih kar nekaj različic, odvisno od idej, ki smo jih v skaldu z mentorjem in somentorejm dorekli oziroma dopolnjevali. Končni program pa bo najverjetneje pridobil še kakšen pregled, ki bo še olajšal preglede dnevnega, tedenskega, mesečnega oziroma letnega dogajanja na poligonu centra varne vožnje.

5 ZAKLJUČEK

Naše delo v raziskovalni nalogi je pokazalo, da je mogoče razviti učinkovit sistem za nadzor zapornice z uporabo sodobnih programskih tehnologij. Med razvojem smo se srečali z različnimi izzivi, kot so zagotavljanje varnosti, stabilnosti povezave in optimizacija uporabniškega vmesnika. Implementacija OTP sistema za avtentikacijo je ključno izboljšala varnost dostopa. Sistem je bil uspešno testiran in omogoča hitro ter varno upravljanje rampe.

Kljub uspešni izvedbi smo se med razvojem soočali s številnimi tehničnimi težavami. Ena izmed večjih težav je bila povezava podatkovne baze s programom, kjer smo naleteli na težave pri vzpostavljanju stabilne komunikacije. Prav tako smo imeli težave z verifikacijo telefonske številke, saj slednje ni mogoče pridobiti avtomatsko iz SIM kartice. Telefonska številka se namreč »klicu pridruži« šele v mobilnem omrežju. Pri razvoju Android aplikacije smo se srečevali z izzivi glede različic operacijskega sistema, prav tako pa smo imeli težave s povezovanjem Android Studia s fizičnim telefonom in emulatorjem.

Pri razvoju iOS aplikacije smo se soočili s pomanjkljivo opremo, saj je bil MAC, na katerem smo delali, zastarel, prav tako pa je bila verzija Xcodea (10.1) zelo stara. Zaradi tega je bilo potrebno prilagoditi kodo, saj nekatere sodobne funkcionalnosti niso bile podprte. Poleg tega simulator ni deloval, kar je otežilo sprotno preverjanje rezultatov razvoja. Dogovarjamo se za prenos aplikacije na novejši sistem, s katerim bomo lahko pripravili tudi to aplikacijo in opravili njeno testiranje v realnem okolju.

In še pogled v prihodnost. Ker skozi zapornico prihajajo zaposleni in gosti, bi bilo za zaposlene pametno pripraviti enostavnejši vstop in izstop z avtomatskim prepoznavanjem vozil in ga integrirati v aplikacijo. S tem zaposlenim ne bi bilo potrebno uporabljati mobilne aplikacije, ki bi bila tako namenjena le gostujočim uporabnikom. Morda pa bodo zaposleni s časom našli še kakšno idejo, ki bi izboljšala ali olajšala njihovo izkušnjo s prihodi in odhodi.

5.1 Hipoteze – potrjene ali ovržene

1. Implementacija OTP sistema za prijavo bi izboljšala varnost in zmanjšala nepooblaščen dostop.

Potrjeno: Hipoteza je potrjena, saj smo z uporabo OTP sistema zagotovili višjo raven varnosti. Vsak uporabnik mora za vstop skozi zapornico naložiti aplikacijo in vpisati svoje podatke, ki se navezujejo na telefonsko številko uporabnika. Za potrditev podatkov prejme na svojo mobilno napravo enkratno geslo (OTP) prek sporočila, kar preprečuje vpis napačne telefonske številke, ki je potrebna za vstop preko zapornice. Vsi ostali podatki se naknadno preverijo ali dopolnijo v pisarni.

2. Aplikacija bo omogočala hitro in stabilno povezavo z zapornico, kar bo omogočilo učinkovito upravljanje.

Ovržemo: Hipoteze ni mogoče potrditi, saj smo aplikacijo testirali le v testnem okolju ne pa tudi fizično na zapornici. Zaradi pomanjkanja časa za testiranje do dneva pisanja tega poročila ne moremo z gotovostjo trditi, da aplikacija omogoča hitro in stabilno povezavo v realnem okolju.

3. Uporabniška izkušnja bo izboljšana z intuitivnim in enostavnim uporabniškim vmesnikom.

Ovržemo: Hipoteze ni mogoče potrditi, saj zaradi omejenega časa nismo izvedli dovolj obsežnega testiranja uporabniškega vmesnika z večjim številom uporabnikov. Posledično ne moremo oceniti, ali aplikacija dejansko zagotavlja izboljšano uporabniško izkušnjo. Dodatno so nam tu zagodle tehnične težave z razvojnim okoljem za iOS.

6 VIRI IN LITERATURA

1. Apple Developer Documentation. [Online]. Apple Inc. [28. dec. 2024; 9.00]. Dostopno na spletnem naslovu: <https://developer.apple.com/documentation/>
2. iOS Tutorials. [Online]. Ray Wenderlich Tutorials. [28. dec. 2024; 9.50]. Dostopno na spletnem naslovu: <https://www.raywenderlich.com/ios>
3. Swift Tutorial – Full Course for Beginners. [Online]. Youtube. [3. jan. 2025; 10.35]. Dostopno na spletnem naslovu: <https://www.youtube.com/watch?v=comQ1-x2a1Q&t=1s>
4. Hacking with Swift. [Online]. Hudson, P. [10. jan. 2025; 11.20]. Dostopno na spletnem naslovu: <https://www.hackingwithswift.com>
5. Android Developer Documentation. [Online]. Google. [14. dec. 2024; 20.00]. Dostopno na spletnem naslovu: <https://developer.android.com/docs>
6. Kotlin Course – Tutorial for Begginers. [Online]. Youtube. [15. dec. 2024; 11.00]. Dostopno na spletnem naslovu: <https://www.youtube.com/watch?v=F9UC9DY-vIU>
7. Kotlin Documentation, JetBrains. [Online]. Kotlinlang.org. [26. dec. 2024; 10.10]. Dostopno na spletnem naslovu: <https://kotlinlang.org/docs/home.html>
8. HTML, SCC, JavaScript. [Online]. W3schools. [20. dec. 2024; 19.15]. Dostopno na spletnem naslovu: <https://www.w3schools.com/>
9. Mathias, M., Gallagher, J. Swift Programming: The Big Nerd Ranch Guide. Big Nerd Ranch Guides. 2016.
10. Phillips, B., Stewart, C. Android Programming: The Big Nerd Ranch Guide. Big Nerd Ranch Guides. 2019.

11. Leiva, A. Kotlin for Android Developers. Lean Publishing. 2016.