Aplikacija za pametno upravljanje mobilnim kućicama u kampovima

Rakas, Aleksandar

Sveučilište Jurja Dobrile u Puli

Fakultet informatike

Aleksandar Rakas

APLIKACIJA ZA PAMETNO UPRAVLJANJE MOBILNIM KUĆICAMA U KAMPOVIMA

Diplomski rad

Pula, rujan, 2023. godine

Sveučilište Jurja Dobrile u Puli

Fakultet informatike

ALEKSANDAR RAKAS

APLIKACIJA ZA PAMETNO UPRAVLJANJE MOBILNIM KUĆICAMA U KAMPOVIMA

Diplomski rad

JMBAG: 0303079090, izvanredan student

Studijski smjer: Sveučilišni diplomski studij Informatika

Predmet: Mobilne aplikacije

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Mentor: doc. dr. sc. Siniša Sovilj

Pula, rujan, 2023. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisan **Aleksandar Rakas**, kandidat za magistra **informatike** ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Diplomskog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

U Puli, rujan 2023. godine

IZJAVA O KORIŠTENJU AUTORSKOG DIJELA

Ja, **Aleksandar Rakas,** dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom **„Aplikacija za pametno upravljanje mobilnim kućicama u kampovima“** koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama. Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, rujan 2023. godine

Potpis

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

Pula, 24. kolovoz 2022.

DIPLOMSKI ZADATAK

Pristupnik:	**Aleksandar Rakas (0303079090)**
Studij:	Sveučilišni diplomski studij Informatike
Naslov (hrv.):	**Aplikacija za pametno upravljanje mobilnim kućicama u kampovima**
Naslov (eng.):	Application for smart management of mobile homes in camps
Opis zadatka:	Zadatak je izraditi aplikaciju za pametno upravljanje mobilnim kućicama u kampovima, gdje bi korisnik preko mobilnih uređaja (mobitel, tablet itd.) mogao upravljati mobilnom kućicom. Aplikacija se treba temeljiti na Arduino tehnologijama i MQTT komunikacijskom protokolu. Usporedo izraditi i jedinične te integracijske testove čitavog koda. Istražiti i usporediti alternativne načine komunikacije sustava.

Zadatak uručen pristupniku: 24. kolovoz 2022.
Rok za predaju rada: 24. kolovoz 2023.

Mentor:

_____
doc.dr.sc. Siniša Sovilj

Komentor:

_____
Dalibor Fonović, dipl.ing.

Sadržaj

1. Uvod

Većina ljudi koji žive u gradovima žive brz život, te traže neko mjesto za opuštanjem van grada okruženo prirodom. Idealno rješenje za takvu vrstu odmora bi bili kampovi u prirodi gdje se nalaze mobilne kućice za odmor. Za takvu vrstu odmora gostima je potrebno omogućiti osjećaj vlastitog doma.

Zbog napredovanja tehnologije u današnje vrijeme neophodno je kreirati sustav koji uveliko pomaže u radnim procesima zaposlenih. Razvijanje takvih aplikacija za upravljanje mobilnim kućicama podiže se sama kvaliteta i sposobnost takvog kampa.

Mobilna aplikacija za upravljanje mobilnim kućicama u kampovima namijenjena je korisnicima koji mogu u zasebnim mobilnim kućicama upravljati sa klimom i svjetlom.

Cilj ove aplikacije bi bio da se korisnicima koji održavaju mobilne kućice ubrzava proces obavljana radova i kontrola nad mobilnim kućicama. Aplikacija se može prenamijeniti i za goste kako bi im omogućilo upravljanje dok nisu na lokaciji istih.

Ovaj diplomski rad govori o upravljanje mobilnim kućicama u kampovima kroz razvoj moderne aplikacije koja omogućuje zaposlenicima u kampu bolje upravljanje svojim resursima i vremenom, te pružiti gostima poboljšano iskustvo boravka u prirodi. Kroz analizu postojećih problema o upravljanju mobilnim kućicama i istraživanje najnovijih tehnoloških rješenja, ovaj rad ima za cilj identificirati ključne funkcionalnosti i zahtjeve za razvoj takve aplikacije.

U sljedećim poglavljima govoriti će se o mogućim problemima i prednosti korištenja aplikacije za upravljanje mobilnim kućicama, te tehničke karakteristike njezine implementacije kao i o strukturi aplikacije.

2. Motivacija

Danas na tržištu vidimo da se sve češće automatiziraju određeni poslovi. Ljudi pokušavaju razvijati aplikacije kako bi smanjili udio ljudskog faktora u odrađivanju određenog posla, sa time se dolazi do pitanja da li je moguće skroz ugasiti ljudski faktor.

IOT (Internet of things) se sve češće pojavljuje u domovima, pa tako je i motivacija ove aplikacije da se proširi i u privremenim domovima. Ljudi žele komoditet i ugodnost prilikom samog boravljenja.

Kako bi se omogućilo ljudima jednostavnije upravljenje svojim domom, kako stalnim tako i privremenim boravkom dolazi se do velikog zadovoljstva i uživanja. Samim time podiže se luksuznost i udobnost doma, te podizanjem na višu razinu kampa u kojem gosti borave.

Svaku aplikaciju je potrebno provesti kroz SWOT analizu. SWOT analiza je tehnika preko koje se pokazuje uspješnost poslovanja i strategije. U sebi sadrži 4 kategorije kao što su snaga, slabosti, prilike i prijetnje.

2.1 Konkurencija

Današnje tržište je preplavljeno raznim aplikacijama. IOT aplikacije se sve češće pojavljuju oko nas, te nam pružaju tehnologije sa kojima krajnji korisnik prema svojim željama I mogućnostima unaprijeđuje upravljanje domom. Trenutno ne postoji niti jedna aplikacija isključivo za upravljanje mobilnim kućicama u kampvima, ali postoje razne aplikacije koje obavljaju istu funkciju u vlastitom domu ili prostoru u kojem korisnik boravi. Neke od poznatijih aplikacija su: Amazon Alexa[1], 3Plus Grupa[2], Loxone[3] I YOUVI[4].

[1] https://alexa.amazon.com/
[2] https://www.3plus.hr/
[3] https://www.loxone.com/
[4] https://www.peaknx.com/

2.1.1 Amazon Alexa

Amazon Alexa je aplikacija ili pametni uređaj sa kojim se može obavljati svakodnevne zadatke uporabom glasa. (Slika 1)

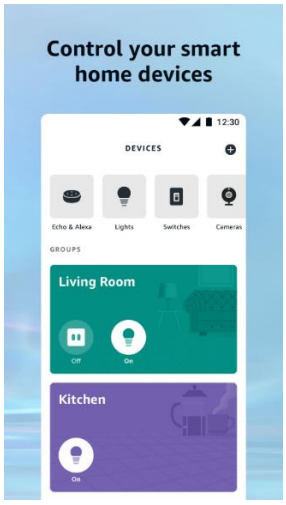
“Alexa čini vaš život lakšim, smislenijim i zabavnijim dopuštajući vam da glasom upravljate svojim svijetom. Alexa vam može pomoći da izvučete više iz stvari koje već volite i otkrijete nove mogućnosti o kojima niste ni sanjali.” (Amazon,2023)

Primjeri Alexe u svakodnevnom životu bi bili upravljanje sa svjetlima, kamerama, televizorima, vratima itd. Svaki pametni uređaj koji ima opciju povezivanja na internet ili preko Bluetooth opcije moguće je povezati sa aplikacijom Alexa. Jedna od stavki je da se može više opcija odraditi od jednom u određeno vrijeme.

“Od svjetala i utičnica do termostata i kamera, Alexa može pomoći da vaš dom postane pametniji i automatiziraniji pojednostavljujući vaše svakodnevne rutine.

Stvaranje povezanog doma je jednostavno. Jednostavno priključite i uključite svoj novi pametni kućni uređaj. Zatim recite: "Alexa, otkrij uređaje.".

Alexa rutine čine da se nekoliko stvari dogodi jednim zahtjevom, tako da možete učiniti više u manje koraka. Na primjer, ujutro biste mogli zamoliti Alexu da upali svjetla, pročita vijesti i najavi sve sastanke u vašem kalendaru za taj dan.” (Amazon, 2023)



Slika 1: Prikaz Amazon Alexa aplikacije

(Izvor:https://play.google.com/store/apps/details?id=com.amazon.dee.app&hl=en_CA)

2.1.2 3Plus Grupa

3Plus Grupa je tvrtka koja se bavi razvojem tehnologije pametne kuće. Njihova platforma omogućava pristup svim uređajima sa bilo koje lokacije.(Slika 2)

“Naša platforma omogućuje vam da preuzmete kontrolu nad svojim domom bilo gdje i bilo kada. Pratite i upravljajte domovima unutar Android i IOS aplikacija. Bilo da se radi o ulaznim vratima, rasvjeti, višezonskom grijanju/hlađenju, sigurnosnim alarmima, sigurnosnim alarmima ili čak glazbi, imamo bezbroj mogućnosti za personalizaciju.” (3Plus, 2021)

Aplikacija omogućava potpunu kontrolu doma i upravljanje svim mogućnostima radi lakšeg svakodnevnog života. Mogućnost pokretanja aplikacije odvija se na Android i IOS platformi. Uz mobilnu aplikaciju nude mogućnost Hardware-a.

“Naša aplikacija za pametni dom omogućuje korisnicima preuzimanje potpune kontrole nad svojim domom. Praćenje, upravljanje vašim pametnim sustavom u bilo koje vrijeme i korištenje svih dostupnih widgeta za brži pristup čini svaki svakodnevni život uvelike lakšim. Sve potrebne administrativne mogućnosti dostupne su unutar mobilne aplikacije.

Naš iskusni tim programera pruža usluge razvoja računalnog softvera i mobilnih aplikacija za Android i iOS. Ovo, u kombinaciji s našim opsežnim iskustvom u dizajniranju intuitivnih grafičkih korisničkih sučelja, osigurava stvaranje aplikacija visoke klase koje blisko surađuju s našim proizvodima.” (3Plus, 2021)

Slika 2: Prikaz 3Plus aplikacije (Izvor: https://www.3plus.hr/)

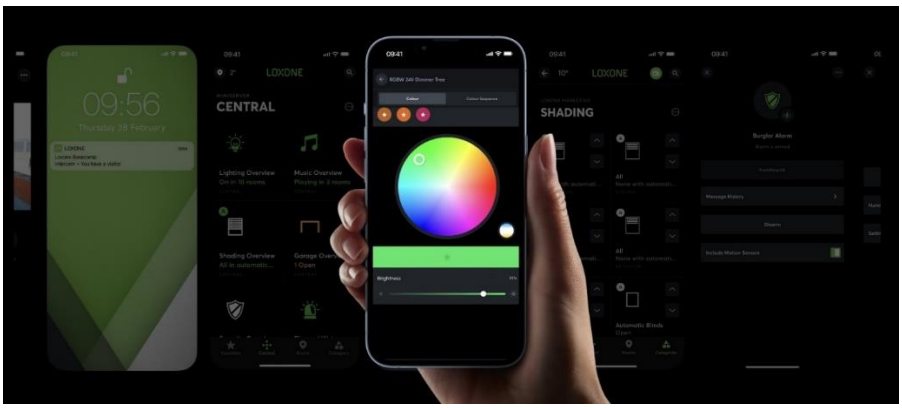
2.1.3 Loxone

Loxone je aplikacija sa kojom se može upravljati svojim domom sa vlastitim kriterijima koji se integriraju unutar aplikacije. Aplikacija se može iskoristiti za unaprijed postavljene paremtre određenih uređaja kako bi se uštedjela energija i novac. Razlika od ostalih IOT aplikacija je u tome što se podaci drže na Miniserveru, a ne u oblaku. (Slika 3)

“Loxone Smart Home više je od same tehnologije. To je način života koji se fokusira na udobnost, sigurnost i energetsku učinkovitost. S Loxoneom možete dizajnirati svoj dom prema svojim željama, potrebama i životnom stilu. A budući da vaš dom obavlja većinu zadataka umjesto vas, imat ćete više vremena za istinski važne stvari u životu.

Od automatiziranog zasjenjenja, rasvjete i multiroom zvuka, pa sve do sigurnosti, kontrole pristupa i upravljanja energijom. Loxone pretvara vaš dom u mjesto koje razumije vaše potrebe i reagira u skladu s tim. Kada niste kod kuće, vaš Loxone Smart Home štedi energiju isključujući grijanje ili hlađenje. Prije nego što se vratite, Loxone će vratiti vaš dom na ugodnu temperaturu, pružajući vam ugodnu klimu u prostoriji u svakom trenutku, uz uštedu energije i novca.

Mnogi pružatelji pametnih kuća prikupljaju ogromnu količinu osobnih podataka – iu najgorem slučaju prodaju ih trećim stranama. Mi u Loxoneu odlučili smo u trenutku kada smo osnovani da se podaci naših klijenata ne prikupljaju niti analiziraju. Ako upalite svjetla u svom domu, to ostaje između vas i vašeg Miniservera.” (Loxone, 2023)



Slika 3: Prikaz Loxone apliakcije (Izvor: https://www.loxone.com/enen/products/apps/)

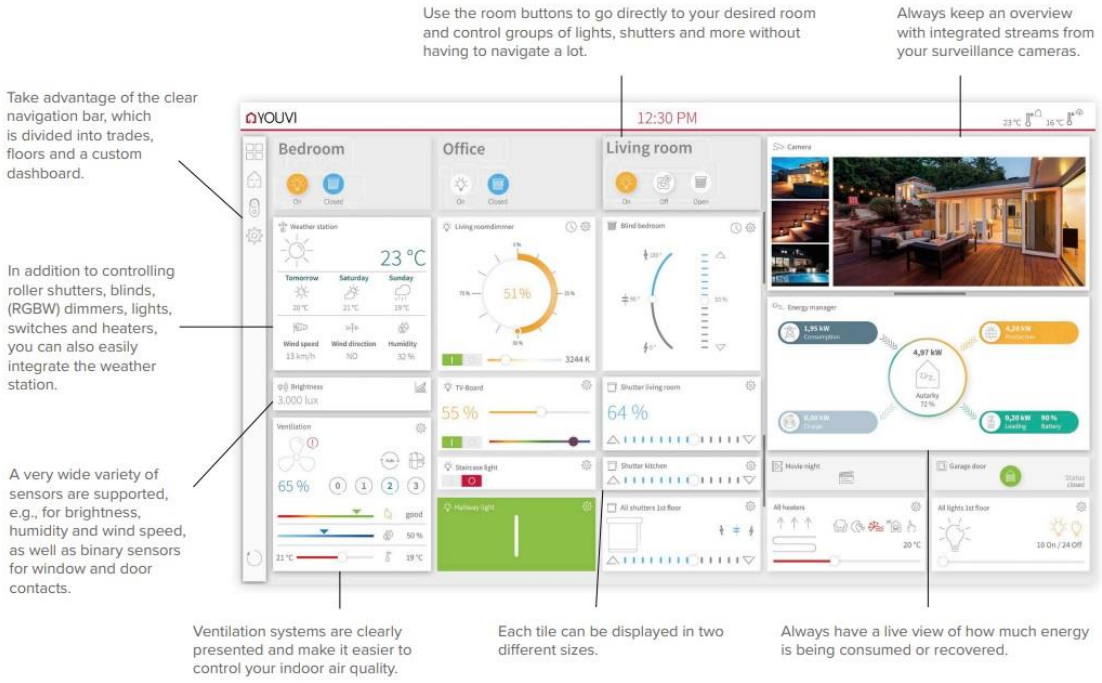
2.1.4 YOUVI

YOUVI je aplikacija koja je integrirana sa hardware-om gdje se može upravljati sa domom preko glasovne naredbe pametnim telefonom ili tabletom.(Slika 4)

“S hardverom i softverom tvrtke PEAKnx možete jednostavno i intuitivno upravljati svojim KNX pametnim domom – sa središnjeg mjesta u zgradi, glasovnom naredbom, pametnim telefonom ili tabletom.

YOUVI se sastoji od VAŠE vizualizacije. Budući da se naša vizualizacija fokusira na korisnika: Učinite YOUVI svojim osobnim softverom za upravljanje i automatizirajte svoj KNX pametni dom jednostavno i intuitivno.

Pouzdana komunikacija s KNX mrežom ključna je za Smart House. Stoga se možete osloniti na naš moćni poslužitelj koji YOUVI sustav pametne kuće dovodi na željeni Windows operativni uređaj, pametni telefon ili tablet.” (Peaknx, 2023)

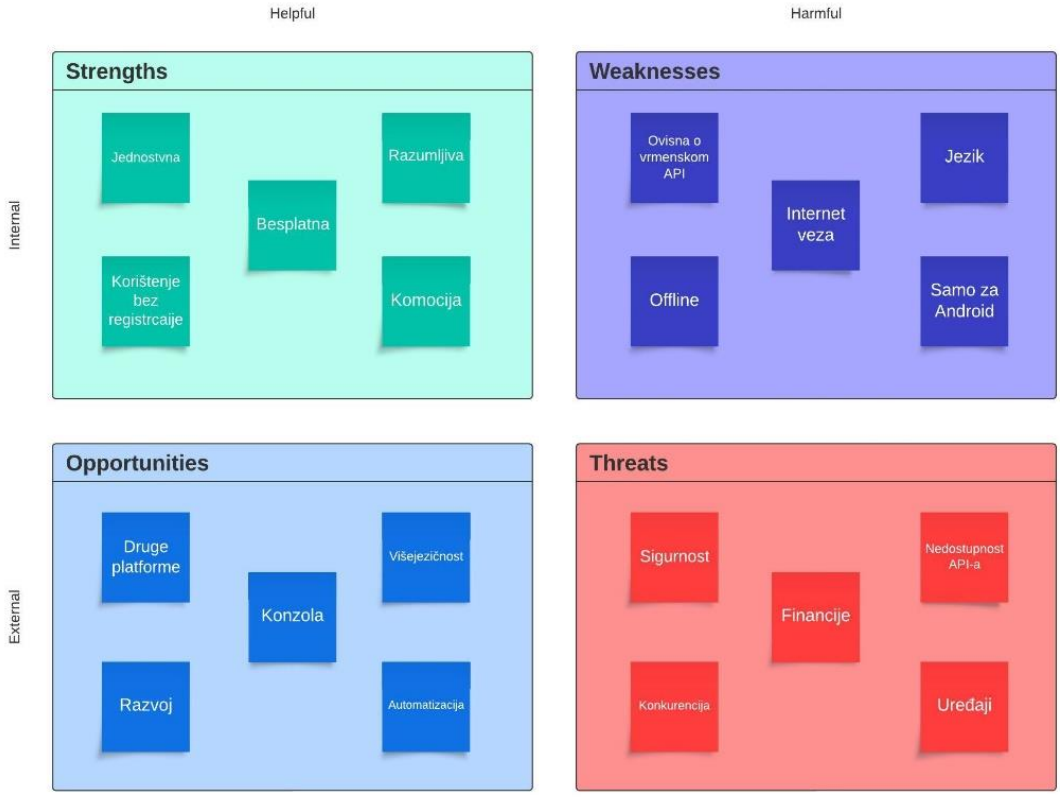


Slika 4: Prikaz YOUVI aplikacije

(Izvor:https://www.peaknx.com/pub/media/peaknx/downloads/documents/factsheets/Controlpro_Factsheet_EN.pdf)

2.2 SWOT analiza

Izradom SWOT analize pokušava se definirati prednosti i nedostaci same aplikacije, kako unutarnje tako i vanjske. Sa time se dolazi do mogućeg rješenja kako u budućnosti unaprijediti aplikaciju i spoznati moguće probleme na koje bi se moglo naići tijekom vremena. U nastavku će biti svaka stavka objašnjena na primjeru aplikacije. (Slika 4)



Slika 4: Prikaz Swot analize (Izvor: autor)

Snaga :

Najveća prednost ove aplikacije je ta što je korisnici mogu koristiti besplatno. Nema reklama koje izlaze na ekranu i pretplate koje se dodatno naplaćuju. Sa time se dolazi do velikog zadovoljstva krajnjeg korisnika. Korisnik kada otvori aplikaciju dolazi do jednostavnog i razumljivog izgleda. Sadržaja mu omogućuje efikasno i brzo uočiti sadržaj, te napraviti potrebnu izmjenu.

Aplikacija nema registraciju i prijavu, što zapravo korisniku nudi mogućnost brže odrade zadatka kojeg želi odraditi.

Komocija je jedna od bitnih stavki prednosti. Danas sve više ljudi žele biti komotni u svom domu, te preko pametnih telefona sa bilo koje lokacije odrađivati zadatke za koje bi inače morali biti u istom, ali i odrađivanje zadataka iako se nalaze u svom domu kao što je na primjer ugasiti svjetlo ako su već legli u krevet i slično.

Slabosti:

Jedna od slabosti aplikacije je ta što za upravljanje aplikacijom je potrebna internetska veza. Lokacije na kojima je slabo pokrivena internetska veza aplikacija ne radi ispravno, sa time i ako je slab Wireless internet. Zbog nemogućeg ili lošeg spajanja na mrežu aplikacija se ne može koristiti. Problem korištenjem aplikacije nastaje zbog dohvaćanja API-a temperature. Trenutno API za vremensku prognozu je ograničen na 10 000 poziva dnevno, što je malo, ali sa pretplatom se može broj povećati na neograničeno.

Aplikacije je napravljena samo na hrvatskom jeziku što ograničava produkciju za strane državljane i mogućnost korištenja je samo na Android platformi, što smanjuje veliku brojku populacije koje koriste IOS platforme.

Prilike:

Izrada konzole bi bila jedna od najboljih prilika za razvijanje sustava. Sa konzolom bi se moglo unaprijediti cijeli sustav i ponuditi gotovo idejno rješenje za određene potrebe korisnika. Unaprijedila bi se aplikacija sa mogućnosti odabira stranih jezika kako bi se proširila produkciju na više država.

Za povećanjem broja korisnika aplikacija bi trebala moći podržavati i druge platforme koje koriste pametni uređaji.

Trenutno sa ovom aplikacijom ima se mogućnost upravljati svjetlom i klimom, ali postoji mogućnost za razvoj u smislu upravljanje puno više uređaja i komponenti u domu.

Automatizacija određenih uređaja i komponenti bi doprinijela uspješnosti aplikacije kako bi korisnici mogli upravljati po rasporedu s obzirom na svoje životne navike.

Prijetnje:

Cijeli sustav izrade takvog sustava zatijeva velika financijska sretstva, gdje bi bilo potrebno pronaći invenstitore koji bi uložili u ovu aplikaciju. Sa razvijanjem takvog sustava, aplikacija kao software I konzola kao hardware dolazi se do velikog troška same izrade. Za implementaciju sustava u svome domu, korisnik mora imati odgovarajuće uređaje koji imaju mogućnost spajanja na konzolu ili aplikaciju što iziskuje dodatne troškove.

Ukoliko ne postoje senzore kao što je slučaj kod ove aplikacije, ona dohvaća temperaturu preko API-a, te je usko povezana sa serverom preko kojeg se šalju podaci. Ukoliko je taj server nedostupan određeni dio aplikacije neće funkcionirati.

Korisnici pretežito ako imaju starije uređaje nemaju mogućnost spajanja na aplikaciju, što dolazi do dodatnih problema.

Aplikacija je otvorenog koda što smanjuje sigurnost samih uređaja, gdje postoji mogućnost hakerskih napada i upravljanje domom sa treće strane.

Trenutno na tržištu postoji jako puno konkurentnih aplikacija koje nude veće mogućnosti što dolazi smanjenjem potrebe za ovom aplikacijom.

2.3 Mogućnost daljnjeg razvoja

Tehnologija napreduje iz dana u dan, samim time i aplikacije koje su napravljene zahtjevaju dodatna unaprijeđivanja i poboljšanja same aplikacije. Trenutno ova aplikacija ima puno raznih mogućnosti kako unaprijediti i poboljšati sustav da bi konkurirala na tržištu.

Potrebno je korisnicima koji koriste aplikaciju ponuditi ocjenjivanje i mogućnost nadopunjavanja komentarima koje potrebe njih zadovoljavaju kako bi se poboljšala aplikacija, sa time bi se dobilo najiskrenije mišljenje korisnika. Nakon toga se kreće u razradu plana i ideje za promjenama.

Aplikaciju je potrebno proširiti na druge platforme i prevesti na više jezika kako bi se povećala populaciju koja ih koristi.

Potrebno je uvesti razne mogućnosti upravljanja, kako bi korisnik mogao sve svoje uređaje koji imaju mogućnost spajanja da konfigurira u aplikaciji, te njima upravlja.

Korisnicima je potrebno omogućiti automatizaciju kako bi svoje dnevne navike prenijeli na odrađivanje aplikacije. U to bi ulazilo na primjer da se korisniku svako jutro u određeno vrijeme, koje sam korisnik postavi, upali svijelo, podignu zastori na prozorima, te se uključi aparat za kavu ltd.

Sa takvim velikim upravljanjem cijelog sustava dolazi se do problema sigurnosti. Za daljnje razvijanje ove aplikacije potrebno je napraviti server koji se nalazi u kući kako bi se moglo maksimalno zaštiti od hakerskih napada i krađe podataka.

Kako bi aplikacija mogla spadati u gotovo rješenje potrebno je napraviti konzolu na kojoj se nalazi ova aplikacija i ona upravlja cijelim domom, na koju su spojeni uređaji za kontroliranje.

Trebali bi se ugraditi odgovarajući senzori koji bi prikupljali informacije i slali na aplikacijsko sučelje ili kao obavijesti na pametni uređaj ukoliko senzori očitaju neku informaciju koja nije dozvoljena. Odrađivanje automatizacije kao na primjer senzor pokreta ukoliko je netko u našem domu tada se pozove policija ili povećanje temerature kada se automatski pali klima.

Sa time bi korisnici dobili proizvod koji im omogućava cijeli sustav na jednom mjestu.

3. IoT

IoT tehnologija je koncept povezivanja fizičkih objekata sa uređajima u kojima su integrirani senzori sa sofware-om povezani sa mrežom kako bi prikupljali podatke.

Sa njima se omogućuje i međusobno povezivanje sa više uređaja stvarajući veliku mrežu.

“Internet stvari (IoT) odnosi se na mrežu fizičkih uređaja, vozila, uređaja i drugih fizičkih objekata koji su ugrađeni sa senzorima, softverom i mrežnim povezivanjem koji im omogućuje prikupljanje i dijeljenje podataka.” (IBM, 2023)

IoT danas u svijetu sve više prevladava u društvu. IoT ima mnoge primjene u različitim industrijama, uključujući pametne gradove, industriju, zdravstvo, poljoprivredu, promet, energetiku i mnoge druge. Ova tehnologija donosi mnoge prednosti, ali također postavlja izazove u vezi s sigurnošću podataka, privatnošću i upravljanjem velikim količinama podataka generiranih IoT uređajima.

Primjeri uređaja uključeni u IoT obuhvaćaju pametne telefone, pametne termostate, pametne kamere, senzore za praćenje okoliša, pametne kućanske aparate, automobile s internetskom vezom, medicinske uređaje i mnoge druge. Ovi uređaji prikupljaju podatke, šalju ih putem interneta na udaljene servere ili druge uređaje i omogućavaju korisnicima da ih nadziru i upravljaju putem aplikacija ili računalnih sučelja.

“Potencijalne primjene IoT-a su goleme i raznolike, a njegov utjecaj već se osjeća u širokom rasponu industrija, uključujući proizvodnju, transport, zdravstvo i poljoprivredu. Kako broj uređaja povezanih s internetom nastavlja rasti, IoT će vjerojatno igrati sve važniju ulogu u oblikovanju našeg svijeta i transformaciji načina na koji živimo, radimo i komuniciramo jedni s drugima.

IoT omogućuje tim pametnim uređajima da komuniciraju međusobno i s drugim uređajima s omogućenim internetom, poput pametnih telefona i pristupnika, stvarajući ogromnu mrežu međusobno povezanih uređaja koji mogu razmjenjivati podatke i obavljati različite zadatke autonomno.” (IBM, 2023)

IoT će vjerojatno imati značajnu ulogu u budućnosti, a razvoj tehnologije IoT-a donosi brojne mogućnosti i izazove

S obzirom na te trendove, IoT će se vjerojatno širiti i produbljivati u različitim industrijama, a njegova uloga u svakodnevnom životu će rasti. Potrebno je pronaći odgovarajuće rješenje za sigurnost i privatnost korisnika, kako bi se osiguralo da se tehnologija IoT-a koristi na odgovoran i povjerljiv način.

3.1 Prednosti IoT-a

IoT nisu samo uređaji koji se nalaze oko nas, IoT je ustvari nešto sasvim više, ono donosi brojne prednosti u različitim aspektima oko života ljudi i poslovanja.

Neke od prednosti IoT-a su:

- Povećana učinkovitost: IoT omogućava automatizaciju procesa i optimizaciju resursa. To znači manje gubitaka vremena, energije i drugih resursa te bolju iskorištenost dostupnih kapaciteta.

- Bolja upravljanja resursima: U poslovanju, IoT omogućava bolje praćenje i upravljanje resursima kao što su zalihe, oprema i vozila. To može smanjiti troškove i povećati produktivnost.

- Poboljšane usluge: IoT omogućava pružanje personaliziranih i poboljšanih usluga korisnicima. Na primjer, pametni uređaji mogu automatski prilagoditi svoje funkcije prema korisničkim preferencijama.

- Brža i bolja analiza podataka: IoT uređaji prikupljaju velike količine podataka koji se mogu analizirati za dobivanje vrijednih uvida. To može pomoći u boljem razumijevanju potreba korisnika i donošenju informiranih odluka.

- Poboljšana sigurnost: IoT se koristi u nadzoru i sigurnosti, uključujući pametne kamere, senzore za detekciju požara, alarme i sustave za kontrolu pristupa. To pomaže u zaštiti ljudi i imovine.

- Smanjenje troškova i otpada: IoT omogućava bolje upravljanje energijom, vodom i drugim resursima, što može dovesti do smanjenja troškova i ekološki prihvatljivijeg poslovanja.

- Poboljšana kvaliteta života: U kućanstvima, IoT uređaji kao što su pametni termostati, pametne svjetiljke i pametni uređaji za kućnu zabavu čine svakodnevni život praktičnijim i udobnijim.

- Povećana dostupnost informacija: IoT omogućava pristup informacijama iz udaljenih mjesta putem interneta, što može biti korisno u mnogim situacijama, uključujući zdravstvenu skrb, obrazovanje i posao.

- Novi poslovni modeli: IoT otvara vrata novim poslovnim modelima, kao što su pretplate na usluge povezane s IoT-om, prodaja podataka ili usluge temeljene na analizi podataka.

- Bolje upravljanje gradovima: Pametni gradovi koriste IoT za optimizaciju prometa, upravljanje energijom, praćenje okoliša i poboljšanje kvalitete života građana.

“IoT uređaji nisu samo svakodnevni predmeti u vašem domu. Uređaji se također koriste na radnom mjestu, često u tvornicama i na proizvodnoj traci. Uređaji se mogu povezati kako bi se pojednostavili procesi. Međutim, oni se mogu koristiti i inteligentnije

Jedan veliki razlog popularnosti pametnih i povezanih uređaja je koliko su korisni za okoliš.

IoT je jasan prikaz inovacije. To je kombinacija svakodnevnih uređaja i pojednostavljenih podataka.

IoT uvodi mnoštvo prilika za razne industrije. Zdravstvo je jedno od najperspektivnijih. Već stvaramo više aplikacija povezanih sa zdravljem, koje mogu prikupljati i pohranjivati osobne podatke u vezi s zdravstvenom poviješću ljudi, iznenadnim simptomima i cjelokupnim zdravljem” (Kiesha Frue, 2019)

3.2 Mane IoT-a

Unatoč mnogim prednostima, IoT također ima nekoliko mana i izazova koje treba uzeti u obzir:

- Sigurnost i privatnost: Jedan od najvećih izazova IoT-a je sigurnost podataka i uređaja. Mnogi IoT uređaji su ranjivi na sigurnosne prijetnje, a ako ne budu pravilno zaštićeni, mogu biti mete hakera. Osim toga, prikupljeni podaci o korisnicima i njihovim navikama mogu predstavljati ozbiljnu prijetnju privatnosti.

- Upravljanje: Upravljanje velikim brojem IoT uređaja može biti izazovno. Potrebna je učinkovita infrastruktura za upravljanje i održavanje tih uređaja kako bi se osigurala njihova pouzdanost i funkcionalnost.

- Troškovi implementacije: Implementacija IoT rješenja može biti skupa, posebno za mala poduzeća i pojedince. Troškovi uključuju nabavu uređaja, povezivanje na mrežu i razvoj softverske infrastrukture.

- Održivost baterije: Mnogi IoT uređaji koriste bateriju kao izvor napajanja, a održavanje dugog vijeka trajanja baterije može biti izazov.

- Velike količine podataka: IoT uređaji generiraju velike količine podataka, a njihovo učinkovito prikupljanje, obrada i analiza mogu biti izazovni. Potrebna je infrastruktura za pohranu podataka.

- Ekološki utjecaj: Porast broja IoT uređaja može povećati potrošnju energije i elektroničkog otpada, što može imati negativan utjecaj na okoliš.

- Zloupotreba podataka: Prikupljeni podaci mogu se zloupotrijebiti ili prodavati trećim stranama bez pristanka korisnika, što postavlja pitanja o etici i privatnosti.

- Regulacija i pravna pitanja: Brzi razvoj IoT-a postavlja izazove u pogledu regulacije i pravnih pitanja, uključujući pitanja odgovornosti za štetu i povrede koje mogu proizaći iz korištenja IoT uređaja.

- Smanjenje ljudskog elementa: Automatizacija putem IoT-a može dovesti do smanjenja potrebe za ljudskom interakcijom u određenim industrijama, što može rezultirati gubicima radnih mjesta.

“Najveća prijetnja IoT uređajima je ranjivost hakiranja. Ako netko želi pristup vašim podacima, naći će način. Ali za razliku od računala koja imaju antivirusni program za lociranje zlonamjernog softvera ili omogućavanje trenutnih ažuriranja kada se pojavi novo ažuriranje, IoT-u ozbiljno nedostaju preventivne mjere kibernetičkog kriminala..

IoT je bio (i još uvijek jest) uzbudljiva industrija, ali trenutno ima ograničenja. Ne možete nastaviti razgovor sa svojim pametnim zvučnicima jer nema potrebne razine umjetne inteligencije.

Ovi se uređaji oslanjaju na prikupljanje podataka za poboljšanje ili pružanje usluga. Ali ovu ogromnu količinu podataka, koju prikuplja svaki povezani uređaj, treba negdje pohraniti i analizirati. Trenutačno nemamo najbolju infrastrukturu za obavljanje tako velikog posla, a to bi u budućnosti moglo dovesti do velikih problema povezanih s podacima.” (Kiesha Frue, 2019)

4. Tehnologije

Tehnologije koje su korištene za izradu aplikacije su programski jezik Kotlin[5] I Express Js[6]. U sljedećem poglavlju će biti svaka tehnologija objašnjenja pojedinačno.

4.1 Kotlin

Kotlin je programski jezik koji se koristi za razvoj različitih vrsta aplikacija, uključujući mobilne aplikacije, web aplikacije, desktop aplikacije i backend sustave. Kotlin je razvijen od strane tvrtke JetBrains i prvi put je predstavljen 2011. godine. Od tada je postao popularan među programerima zbog svoje jednostavnosti, izražajnosti i sigurnosti.

Kotlin ima čitljivu sintaksu koja je jako slična drugim programskim jezicima, te je kodiranje brže i lakše

[5] https://kotlinlang.org/
[6] https://expressjs.com/

Usko je povezan sa Javom tjst. potpuno je kompatibilan, te se mogu postojeće biblioteke koristiti u Kotlin projektima.

Kotlin je postao službeni jezik za razvoj Android aplikacija od strane Googla, te je sa time zamijenio Javu kao preferirani jezik. Brzo se širi i postaje popularan izbor među programerima za mnoge vrste razvojnih projekata.

“Preko 50% profesionalnih Android programera koristi Kotlin kao primarni jezik, dok samo 30% koristi Javu kao glavni jezik. 70% programera čiji je primarni jezik Kotlin kažu da ih Kotlin čini produktivnijima.

Od svog nastanka 2011., Kotlin se kontinuirano razvijao, ne samo kao jezik već i kao cijeli ekosustav s robusnim alatima. Sada je neprimjetno integriran u Android Studio i aktivno ga koriste mnoge tvrtke za razvoj Android aplikacija.” (Kotlin, 2023)

4.2 Express Js

Express.js, često nazivan samo Express, je popularni web okvir (framework) za razvoj web aplikacija temeljenih na JavaScriptu. Brz, fleksibilan i efikasan okvir koji se često koristi za izgradnju web aplikacija i API-a (Application Programming Interface).

Express.js je izgrađen na vrhu Node.js, što znači da se koristi za izradu serverske strane web aplikacije.

Express omogućava definiranje ruta koje odgovaraju HTTP zahtjevima. Svaka ruta može imati vezanu funkciju koja se izvršava kad se zahtjev podudara s tom rutom. Olakšava rukovanje HTTP metodama kao što su GET, POST, PUT i DELETE.

Express omogućava integraciju s različitim vrstama baza podataka, uključujući relacijske baze podataka poput MySQL, PostgreSQL, kao i NoSQL baze podataka poput MongoDB.

Pruža podršku za upravljanje korisničkim sesijama i implementaciju autentikacije, što je bitno za razvoj sigurnih web aplikacija i analizu tijela HTTP zahtjeva, što je korisno za primanje i obradu JSON-a, XML-a i drugih formata podataka poslanih putem zahtjeva.

Express.js se često koristi za izgradnju API-ja (RESTful) i serverskih strana web aplikacija. Budući da je lagan i jednostavan za učenje, postao je popularan izbor među programerima za izradu raznih vrsta web aplikacija.

“Express je minimalan i fleksibilan okvir web aplikacije Node.js koji pruža robustan skup značajki za web i mobilne aplikacije.

Uz mnoštvo HTTP uslužnih metoda i međuprograma koji su vam na raspolaganju, stvaranje robusnog API-ja brzo je i jednostavno.” (Express Js, 2017)

4.3 HTTP & MQTT

HTTP je protokol koji služi za prijenos dokumenata. Podaci koji su potrebni spremljeni su u URL obliku.

Potreban je za komunikaciju između klijenta i Web servisa.

Web servis može biti konfiguriran na lokalnij mašini ili na Web serveru.

„HTTP (HyperText Transfer Protocol) je izumljen kao komponenta World Wide Weba za prijenos dokumenata. Najpoznatija nam je kao jedna od tehnologija koja omogućuje rad web preglednika.“ (Ian Craggs, 2022)

Klijenti HTTP protokola postavljaju zahtjeve kao što su metode GET, PUT, DELETE i POST koje koriste jednostavne formate podataka kao što su XML ili JSON. REST arhitektura predstavlja način organiziranja aplikacije i resursa sa navedenim metodama.

HTTP se uobičajeno upotrebljuje kako bi uređajima omogućili POST metodu resursima koji se reprezentiraju na uređajima.

Poslužitelj ima opciju da odgovara podacima samo kada klijent to zahtjeva. Koristi TCP/IP protokol i poruke su tekstualne.

MQTT (Message Queuing Telemetry Transport) je protokol za komunikaciju između uređaja IOT-a i drugih aplikacija.

Dizajniran je da bude lagan i efikasan protokol, što ga čini pogodnim za uređaje s ograničenim resursima, poput senzora i aktuatora u IoT ekosustavu. Poruke su komprimirane i šalju se u binarnom formatu.

Koriste se naredbe kao što su CONNECT, SUBSCRIBE, PUBLISH, UNSUBSCRIBE i DISCONNECT. Dopušta porukama prolaznost u oba smjera između poslužitelja i klijenta.

MQTT je također prikladan za komunikaciju s mobilnim uređajima, što ga čini pogodnim za pametne kuće i IoT rješenja koja uključuju mobilne aplikacije.

Koristi se u mnogim IoT aplikacijama, uključujući pametne kuće, industrijske automatizacije, praćenje vozila, pametne gradove i mnoge druge.

Temelji se na modelu "publish-subscribe", gdje uređaji mogu slati poruke na teme (topics), a drugi uređaji (pretplatnici) se mogu pretplatiti na te teme kako bi primili poruke koje ih zanimaju. Ovaj model omogućava fleksibilnu i skalabilnu komunikaciju između uređaja u IoT mreži.

„MQTT naredbe su CONNECT, SUBSCRIBE, PUBLISH, UNSUBSCRIBE i DISCONNECT. MQTT teme su jedinica distribucije, na koju klijenti mogu OBJAVITI i PRETPLATITI se. Svi ovlašteni pretplatnici na temu primit će sve poruke objavljene na njoj. MQTT teme ne moraju biti unaprijed definirane: aplikacije ih mogu kreirati jednostavno koristeći ih.“ (Ian Craggs, 2022)

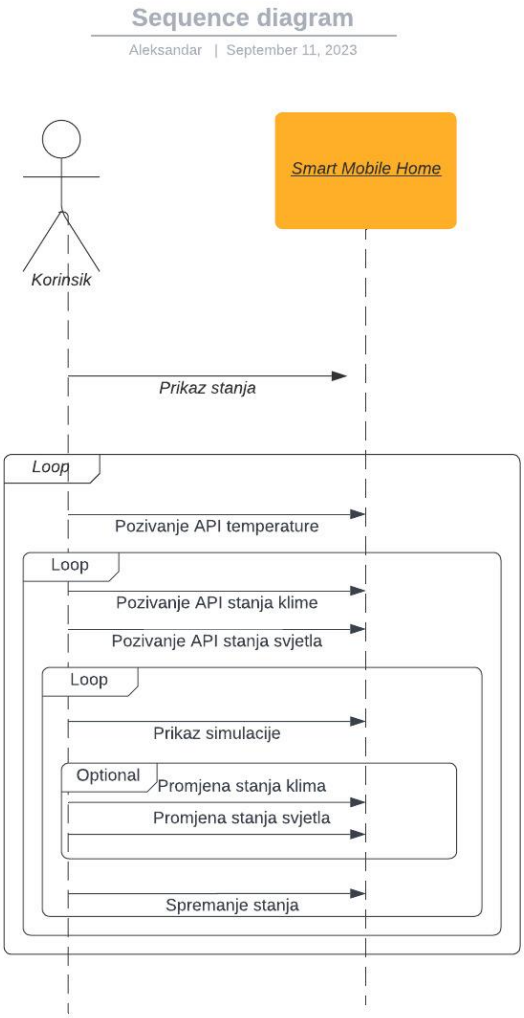
Izbor programera između HTTP i MQTT ovise o potrebama IoT projekta. HTTP se koristi ako je potrebna jednostavna integracija s aplikacijom i usmjerena komunikacija GET i POST metode. MQTT se koristi ako je IoT projekt s velikim brojem uređaja i ograničenim resursima. Također, nije neuobičajeno da se u IoT ekosustavima koriste oba protokola zajedno, svaki za svoje specifične zadatke.

Ova aplikacija koristi HTTP restful arhitekturu zbog lakše povezanosti poslužitelja i klijenta, te zbog bliskosti programiranja.

5. Dijagrami

U sljedećim poglavljima će biti napravljeni i objašnjeni dijagram slijeda (Slika 5) i dijagram slučajeva korištenja (Slika 6)

5.1 Dijagram slijeda



Slika 5: Prikaz Dijagrama slijeda (Izvor: Autor)

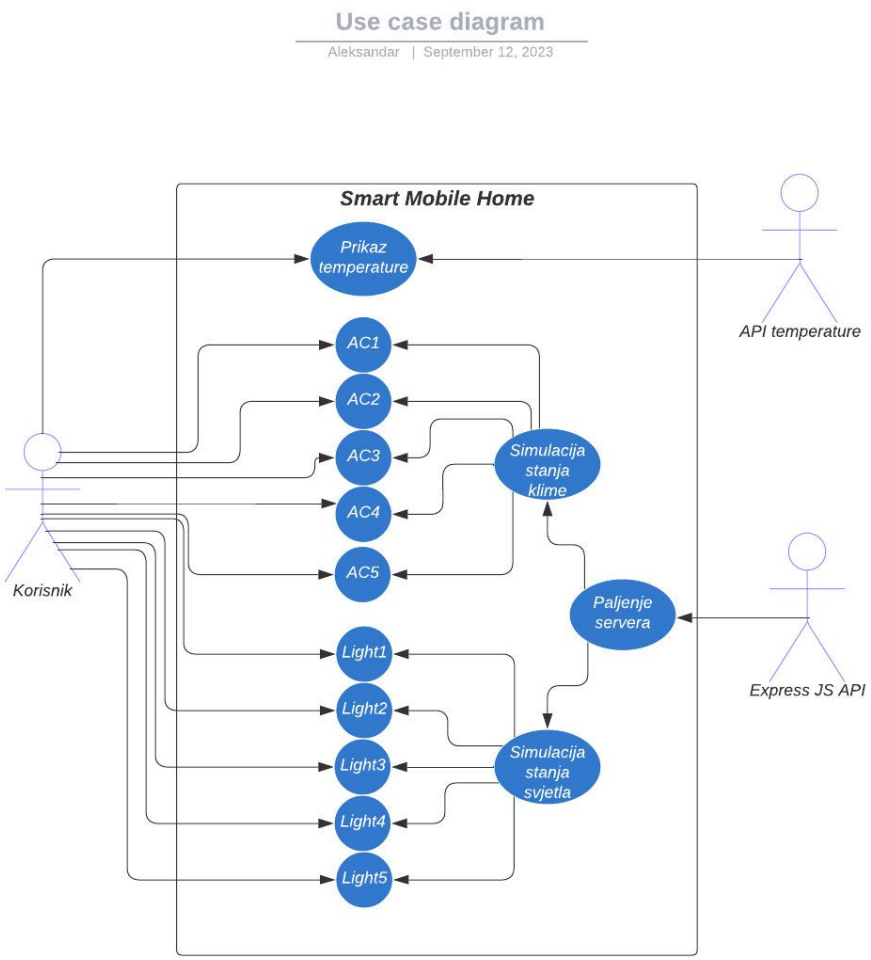
Upravljanje svjetlom i klimom je glavni segment aplikacije. Na početnom zaslonu korisniku su prikazane informacije o trenutnoj temperaturi, tlaku, vlazi i vjetru koji se dohvaća preko API-a Open Meteo smješten na web serveru. Na zaslonu su prikazane 5 mobilnih kućica sa switch button-om kojeg korisnik može kontrolirati sa gašanjem/paljenjem klime i svjetla.

Vanjski program Express Js je podignut na lokalnoj mašini koji generira „random“ slučajeve upaljenih i ugašenih klima i svjetla, te aplikacija poziva svakih 3 minute API Call.

U aplikaciji je namješteno da se svakih 10 sekundi očitava parametri Expressa te na osnovu toga mijenja stanje switch button-a kojeg korisnik može kontrolirati.

Promijena koju korisnik napravi ostaje spremljena do ponovnog pozivanja API-a

5.2 Dijagram slučajeva korištenja



Slika 6: Prikaz Dijagrama slučajeva korištenja (Izvor: Autor)

Dijagram slučajeva korištenja pokazuje kako korisnik ima pristup informacijama o temperaturi. Glavna funkcionalnost aplikacije je upravljanje klimama i svjetlom. U aplikaciji postoje dva API-a. Jedan API se nalazi na web serveru preko kojeg dobivamo informacije o temperaturi, a drugi API je podešen na lokalnoj mašini koji mijenja senzore u kućici i nasumično odrađuje zadatak koji je potreban.

Prije pokretanja aplikacije potrebno je ručno upaliti server kako bi bio povezan sa aplikacijom.

Sa time aplikacija simulira podatke i prikazuje unutar nje.

6. Implementacija aplikacije

Prilikom izrade android aplikacije “Smart Mobile Home” korišten je program Android studio u kojem se razvijala aplikacija. Bilo je potrebno znanje tehnologije koje su prije navedene Kotlin i Express js.

U aplikaciji se koriste dva API-a. Jedan API je korišten u svrhu prikupljanja podataka meteorološke stanice koji se nelazi na serveru web stranice Open-Meteo.

Drugi API se koristi na lokalnoj mašini koji je napravljen u Express Js kako bi simulirao senzore i slao nasumične podatke koje se nalaze u Mobilnim kućicama.

Aplikacija provjerava GPS lokaciju i ukoiliko nije dozvoljena na uređaju, aplikacije neće raditi.

Kao početak izrade aplikacije preuzet je inicijalni predložak sa GitHub-a philipplackner[7].

Aplikacija bi omogućila korisnicima pregled informacija o uređajima (npr. stanje svjetla, klime) i prikazala povratne informacije nakon što korisnik izvrši određene akcije (npr. uključivanje svjetla).

[7] https://github.com/philipplackner/WeatherApp/tree/initial/app

Izrada je bila na način da je korisničko sučelje prihavatljivo korisniku na način da je aplikacija intuitivna i lako se koristila.

Ova implementacija bi omogućila korisnicima nadzor nad različitim uređajima u njihovim pametnim kućicama putem mobilnih uređaja. Važno je napomenuti da bi stvarna implementacija ovisila o specifičnim uređajima i njihovoj kompatibilnosti s Express.js serverom i Android aplikacijom. Također bi se morala uzeti u obzir pitanja sigurnosti i privatnosti kako bi se zaštitili podaci i uređaji korisnika.

6.1 Vremenska prognoza

Prvi dio aplikacije se sastoji od vremenske prognoze koja je prikazana na vrhu početnog zaslona. Preko API-a se prikupljaju podatci kao što su temperatura, vlaga, vjetar I tlak. Uz prikupljenje podatke prikazane su i ikonice koje označavaju vremensko stanje.

Slika 7: Prikaz koda GET zahtjev za API (Izvor: Autor)

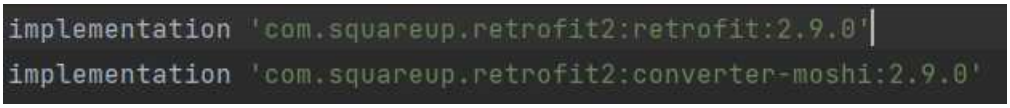
Slika pokazuje da se radi o HTTP GET zahtjevu na resurs definiranom atributom. (Slika 7) Resurs je specificiran kao "v1/forecast" s određenim parametrima kao što su geografska širina i dužina (latitude i longitude) te popis željenih podataka (temperature, relativna vlažnost, brzina vjetra, tlak na razini mora) koji će se dohvatiti u odgovoru.

Ukratko, ovaj dio koda definira funkciju za dohvat vremenskih podataka putem HTTP GET zahtjeva prema određenom resursu. Parametri za geografsku širinu i dužinu prenose se putem URL-a, a rezultat će biti deserijaliziran u obliku „WeatherDto“ objekta.

Slika 8: Prikaz koda funkcije provideWeatherAPi (Izvor: Autor)

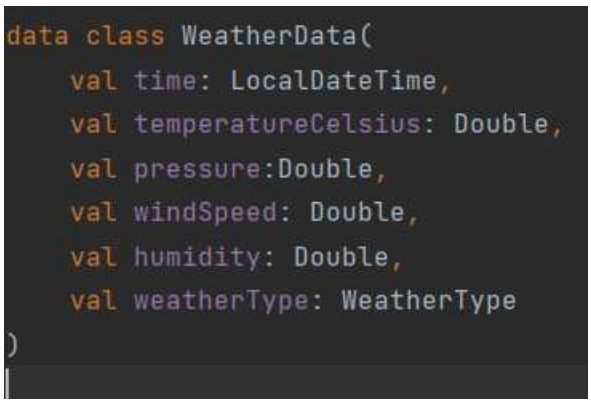
Ova linija koda predstavlja metodu koja se koristi za pružanje (provide) implementacije „WeatherApi“ sučelja, a ta implementacija se koristi za izradu HTTP zahtjeva prema API-u za vremensku prognozu. (Slika 8)

Postavlja se bazna URL adresa za API, što znači da će svi HTTP zahtjevi prema „WeatherApi“ biti upućeni na ovu adresu, te se dodaje konverter za obradu odgovora API-a. U ovom slučaju, koristi se MoshiConverterFactory za pretvaranje JSON odgovora u objekte koji se implementira u “build.gradle (:app)” sa jednostavnim linijama koda. (Slika 9)



Slika 9: Prikaz koda implementacije Moshi (Izvor: Autor)

U suštini, ova metoda „provideWeatherApi“ konfigurira i vraća instancu „WeatherApi“ koja je spremna za upotrebu u cijeloj aplikaciji



Slika 10: Prikaz koda klase WeatherData (Izvor: Autor)

Napravljena je klasa “WeatherData” u koju spremamo vrijednosti koje su dobivene preko API-a. Svaka varijabla je definirana kao “Double” što se koristi kao tip podataka za pohranu decimalnih brojeva. (Slika 10)

“val weatherType: WeatherType” predstavlja tip vremena. Koristi se za opisivanje je li sunčano, oblačno, kišovito, snježno i sl. “WeatherType” je klasa koja sadrži različite moguće tipove vremena.

```
@Composable
fun WeatherCard(
    state: WeatherState,
    backgroundColor: Color,
    modifier: Modifier = Modifier
) {
    state.weatherInfo?.currentWeatherdata?.let { data ->
    Card (
        backgroundColor = backgroundColor,
        shape = RoundedCornerShape(30.dp),
        modifier = modifier.padding(40.dp)
        ){
```

Slika 11: Prikaz koda funkcije WeatherCard (Izvor: Autor)

Ova dio koda definira Kotlin funkciju nazvanu “WeatherCard” koja služi za stvaranje komponente (widgeta) korisničkog sučelja (UI) u kojoj se nalaze 3 argumenta. (Slika 11)

“state.weatherInfo?.currentWeatherdata?.let { data ->” Koristi se Kotlinov blok za obradu objekta ako je dostupan. “state” sadrži informacije o vremenskim podacima, a ovom provjerom se osigurava da će se daljnja obrada izvršiti samo ako su vremenski podaci dostupni.

U tijelu funkcije je kreiran jedan stupac u kojeg se povlače parametri koji su potrebni, kao što su slike, tekst, te funkcija u kojoj je definirana slika tlaka,vlage, vjetra i tekst sa podacima o njihovoj vrijednosti, te funkciju ”WeatherCard” kasnije pozivamo u “MainActivity”.

```
@Composable
fun WeatherDataDisplay(
    value: Int,
    unit: String,
    icon: ImageVector,
    modifier: Modifier = Modifier,
    textStyle: TextStyle = TextStyle(),
    iconTint: Color = Color.White
)
```

Slika 12: Prikaz koda funkcije WeatherDataDisplay (Izvor: Autor)

Kreirana je funkcija „WeatherDataDisplay“ (Slika12).

```
            Icon(
                imageVector = icon,
                contentDescription =null,
                tint = iconTint,
            modifier = Modifier.size(25.dp)
            )

        Spacer(modifier = Modifier.width(4.dp))

        Text(
            text = "$value$unit",
            style = textStyle
        )



    }
}
```

Slika 13: Prikaz tijela koda funkcije WeatherDataDisplay (Izvor: Autor)

U funkciji „WeatherDataDisplay“ ubacuje se ikona tlaka, vjetra i vlage, te njihove vrijednosti. (Slika 13)

6.2 Express Js i implentacija

Da bi postojao osjećaj stvarnih uvjeta prilikom korištenja aplikacije bilo je potrebno stvoriti okruženje koji će reprezentirati realne događaje.

U ovom primjeru simuliraju se situacije korištenja klime i svjetla te obaviještavanje korisnika o njihovom trenutnom stanju uključenosti ili isključenosti.

Express je korišten kao alat, kako bi se simulirali podaci od senzora.

Napravljen je da aplikacija bude funkcionalna, te da korisnik može upravljati nad resursima.

```
const express = require('express');
const app = express();
app.use(express.json());

const houseACValues = {
  AC1: false,
  AC2: false,
  AC3: false,
  AC4: false,
  AC5: false,
  light1: false,
  light2: false,
  light3: false,
  light4: false,
  light5: false
};

function updatehouseACValues() {
  Object.keys(houseACValues).forEach((key) => {
    houseACValues[key] = Math.random() < 0.5;
  });
}

setInterval(updatehouseACValues, 3 * 60 * 1000);
updatehouseACValues();
```

Slika 13: Prikaz koda Express.js (Izvor: Autor)

Ovaj dio koda napisan je u JavaScript programskom jeziku koristeći Node.js okolinu i Express.js web okvir. (Slika 13)

Definira se objekt “houseACValues”, koji sadrži informacije o različitim uređajima u kući (klima uređaji i svjetla). Inicijalno su svi uređaji postavljeni na “false”, što znači da su isključeni.

Kreira se funkcija “updatehouseACValues”. Unutar ove funkcije koristi se petlja za iteriranje kroz objekte i svakom uređaju se dodjeljuje slučajna vrijednost (true ili false) s vjerojatnošću 50%. Ova funkcija simulira promjene statusa uređaja u kući.

Postavlja se periodički interval od 3 minute (3 * 60 * 1000 ms) i poziva funkciju “updatehouseACValues” na svakih 3 minute. To znači da će se status uređaja u kući nasumično mijenjati svakih 3 minute.

Slika 14: Prikaz koda GET I PUT u Express js. (Izvor: Autor)

Ovaj dio koda definira nekoliko ruta i povezanih funkcija za upravljanje zahtjevima (HTTP GET i PUT zahtjevima). (Slika 14)

Prva ruta /house/all definirana je kao HTTP GET zahtjev. Kada se na ovoj ruti izvrši GET zahtjev, izvršava se funkcija koja prima “req” (zahtjev) i “res” (odgovor) kao argumente. Zatim se koristi “res.json(houseACValues)” kako bi se poslao odgovor koji sadrži sadržaj objekta “houseACValues” kao JSON objekt. Ovaj odgovor će sadržavati status uređaja u kućici.

Druga ruta /house/:id sadržava identifikator uređaja, koristi “req.params” kako bi se dohvatila vrijednost parametra "id" iz rute, te provjerava da li postoji uređaj u kući s određenim identifikatorom.

Ako uređaj postoji, šalje se odgovor u JSON formatu koji sadrži identifikator uređaja (id) i njegov trenutni status (value).

Sa funkcijom “app.put” omogućavamo da se šalje status “id” I “value” na server.

```
fun fetchDataFromApi() {

    val client = OkHttpClient()
    val url = "http://10.0.2.2:3000/house/all"
    val request = Request.Builder()
        .url(url)
        .build()

    try {
        // Execute the request
        val response: Response = client.newCall(request).execute()

        // Check if the request was successful (status code 200)
        if (response.isSuccessful) {
            val responseBody = response.body?.string()
```

Slika 15: Prikaz koda funkcije fetchDataFromApi (Izvor: Autor)

U Kotlinu je kreirana funkcija “fetchDataFromApi” koja se koristi za izvršavanje GET zahtjeva prema određenom URL-u i dohvaćanje podataka. Ako je zahtjev ispravan spremamo vrijednosti koje smo dobili od API-a u varijablu kao string. (Slika 15)

Da bi kasnije koristili podatke iz tog stringa potrebno je podatke prebaciti u objekt. Za to je korištena OkHttpClient I Gson biblioteka. Stvaranjem instance koristi se za parsiranje JSON podataka u objekte.

Gson se koristi za pretvaranje JSON odgovora (spremljenog u varijablu “responseBody”) u objekt. Konkretno, ovaj kod pretvara JSON podatke u objekt tipa Map, gdje su ključevi JSON objekta postavljeni kao ključevi u mapi, a vrijednosti JSON objekta postavljene kao vrijednosti u mapi. Ovaj objekt “map” sadržava sve podatke koji su dobiveni iz JSON odgovora.

Ovisno o sadržaju JSON odgovora, ove varijable će sadržavati “true” ili “false” koje označavaju status tih uređaja.

Nakon toga svaki “id” i njegov “value” se sprema u varijablu koja se kasnije koristi u aplikaciji.(Slika 16)

```
val gson = Gson()
val jsonObject = gson.fromJson(responseBody, Map::class.java)

var ac1 = jsonObject["AC1"]
val ac2 = jsonObject["AC2"]
val ac3 = jsonObject["AC3"]
val ac4 = jsonObject["AC4"]
val ac5 = jsonObject["AC5"]
val lght1 = jsonObject["light1"]
val lght2 = jsonObject["light2"]
val lght3 = jsonObject["light3"]
val lght4 = jsonObject["light4"]
val lght5 = jsonObject["light5"]
```

Slika 16: Prikaz koda spremanja varijabliu objekt (Izvor: Autor)

6.3 Upravljanje resursima

Radi lakšeg pregleda korisniku o mogućim funkcijama omogućeno mu je na početnom zaslonu prikaz rednog broja mobilnih kućica, te njihovo upravljanje sa Switch ikonom.

U daljenjem nastavku poglavlja biti će objašnjeno kako su kreirani Switch ikone, te njihove funkcije.

Slika 17: Prikaz koda funkcije MobileHomeCard (Izvor: Autor)

Kreirana je funkcija “MobileHomeCard” koja prikuplja sve potrebne informacije o mobilnim kućicama. Tako se u njoj nalazi još jedna funkcija za prikupljanje podataka o API-u, te izgled sučelja koje korisnik vidi na početnom zaslonu aplikacije. (Slika 17)

Slika 18: Prikaz koda inicijalizacije varijabli (Izvor: Autor)

Ovaj dio koda se koristi za inicijalizaciju i praćenje stanja različitih uređaja u aplikaciji. Varijable će se automatski ažurirati kada se promijeni njihovo stanje, a ovo praćenje omogućuje da se promjene u statusu uređaja odraze u korisničkom sučelju aplikacije. Na primjer, ako se status uređaja promijeni na "uključeno" ili "isključeno", tada će se i ove varijable ažurirati kako bi odražavale te promjene, što omogućuje dinamičko osvježavanje prikaza u aplikaciji. (Slika 18)

```
Row(
    modifier = modifier
        .fillMaxWidth()
        .padding(8.dp),
    ) { this: RowScope
    Text(
        text = "Redni broj mobilnih kućica:",
        fontSize = 28.sp,
        color = Color.White
    )
}
Row(
    modifier = modifier
        .fillMaxWidth()
        .padding(16.dp),

    ) { this: RowScope
    Text(
        text = "1:",
        fontSize = 28.sp,
        color = Color.White
    )

    Spacer(modifier = Modifier.width(25.dp))

    Text(
        text = "Klima",
        fontSize = 28.sp,
        color = Color.White
```

Slika 19: Prikaz koda sučelja aplikacije (Izvor: Autor)

U aplikaciji je napravljeno šest redova, prvi red je napravljen običan tekst u kojem radi jasnoće aplikacije opisuje sljedeće redove pod nazivom “Redni broj kućica”. (Slika 19)

Nakon toga reda ostali redovi su isti, prvi dio reda govori redni broj kućice, te u ostatku je tekst “klima” i “svjetlo”.

Nakon svakog dijela tekst se nalazi ikona koji ima svoju funkciju, te ona pokazuje trenutno simulirano stanje.

```
Switch(
    checked = AC1 ,
    onCheckedChange = { isChecked ->
        AC1 = isChecked
    }
)
```

Slika 20: Prikaz koda Switch Button (Izvor: Autor)

Ovaj dio koda implementira prekidač za upravljanje stanjem uređaja “AC1” unutar korisničkog sučelja. Kada korisnik promijeni stanje prekidača, ta promjena će se odraziti na varijablu “AC1”, što omogućuje dinamičko upravljanje uređajem putem korisničkog sučelja. (Slika 20)

Pozivanje Switch-a predstavlja upotrebu Compose komponente "Switch" (prekidač) unutar Compose korisničkog sučelja u Android aplikaciji.

Switch ikona i text su dodani u aplikaciju dvije jednostavne naredbe import koje se vide na slici, te su kasnije pozvani u kodu. (Slika 21)

Za ostale 4 kućice primjer, radnja i funkcija su isti.

```
import androidx.compose.material.Switch
import androidx.compose.material.Text
```

Slika 21: Prikaz koda implementacije Switch Button (Izvor: Autor)

6.4 Korisničko sučelje

Kako bi korisniku bilo što jednostavije shvatiti funkcionalnost aplikacije sve komponente i dijelovi koda koji su korišteni prikazani su u klasi “MainActivity”. U nastavku poglavlja biti će pobliže objašnjeni dijelovi koda koji su korišteni.

```
class MainActivity : ComponentActivity() {


    private val viewModel: WeatherViewModel by viewModels()
    private lateinit var permissionLauncher: ActivityResultLauncher<Array<String>>
```

Slika 22: Prikaz koda klase MainActivity (Izvor: Autor)

Kreirana je klasa “MainActivity”, koje je glavna aktivnost aplikacije. Nasljeđuje “ComponentActivity”, što je osnovna klasa za Android aktivnosti. (Slika 22)

Deklarirana je varijabla “viewModel” koja predstavlja instancu “WeatherViewModel”. Korištenjem “by viewModels()” uzimamo “ViewModel” koji je povezan s aktivnošću i automatski će se kreirati i ponovno koristiti ako je već stvoren.

Nakon toga se deklarira još jedna varijabla “permissionLauncher”, koja će se koristiti za pokretanje aktivnosti koja zahtijeva rezultat aktivnosti (npr. zahtjev za dozvolama). Varijabla je označena kao “lateinit”, što znači da će biti inicijalizirana prije nego što se koristi.

Slika 23: Prikaz koda metode onCreate (Izvor: Autor)

Unutar klase “MainActivity” kreirana je metoda “onCreate”. (Slika 23)

U ovom dijelu koda se definira i implementira “onCreate metoda”, koja je metoda Android aktivnosti i izvršava se prilikom njenog stvaranja. Osnovna implementacija “onCreate” metode obavlja osnovne zadatke inicijalizacije aktivnosti.

Inicijalizira se “permissionLauncher”, što je varijabla koja će se koristiti za registraciju rezultata aktivnosti (Activity Result).

Definira se izraz koji će se izvršiti kada se dozvole zahtijevaju. U ovom slučaju, izvršava se funkcija “loadWeatherInfo()” koja pripada “viewModel” instanci, te “viewModel” sadrži logiku za dohvaćanje vremenskih informacija.

U cjelini, ovaj dio koda koristi Android Activity Result API kako bi omogućio aplikaciji zahtjev za dozvolama za pristup lokaciji (FINE i COARSE lokacijske dozvole). Kada korisnik odgovori na zahtjev za dozvolama, “onActivityResult” metoda će se automatski pozvati, a funkcija “loadWeatherInfo()” unutar “viewModel” će se izvršiti kako bi se dohvatile vremenske informacije nakon što su dozvole odobrene.

Slika 24: Prikaz koda sučelja apliakcije pomoću Compose (Izvor: Autor)

Ovaj dio koda odnosi se na postavljanje korisničkog sučelja (UI) za Android aplikaciju pomoću Compose-a, modernog Android alata za izgradnju korisničkog sučelja. Compose omogućuje jednostavno definiranje korisničkog sučelja i komponenata pomoću Kotlin sintakse. (Slika 24)

“setContent { ... }” je Compose funkcija koja postavlja korisničko sučelje za aktivnost. Sadržaj unutar ove funkcije će definirati kako će izgledati korisničko sučelje aplikacije.

Primjenjuje se Compose tema “WeatherAppTheme” na cijelo korisničko sučelje. Tema može definirati stilove i boje koje će se primijeniti na komponente unutar sučelja.

“Box” je kompozicijski kontejner u Compose-u koji se koristi za postavljanje i organiziranje drugih komponenata unutar sebe.

Kreiran je jedan stupac “Column” koji je kompozicijski kontejner, a koristi se za vertikalno postavljanje komponenata.

“WeatherCard(...)” i “MobileHomeCard(...)” su komponente koje se prikazuju unutar „Column“. “WeatherCard” i “MobileHomeCard” su komponente koje prikazuju informacije o vremenu i upravljanje mobilnim kućicama. Te komponente primaju stanje (state) iz “viewModel.state” i mogu prikazivati informacije na temelju tog stanja.

Na kraju je dodan tekst "Vremenska Prognoza Pula" koristeći Compose “Text” komponentu. Tekst ima određeni stil (veličinu, boju, itd.) definiran pomoću parametara

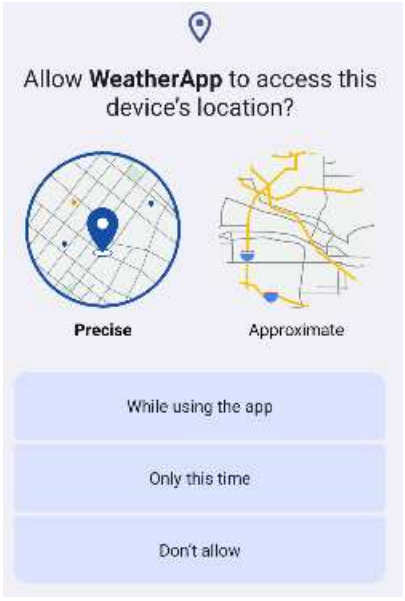
7. Korisničke upute

Pri svakoj inicijalizaciji aplikacije, otvara se prozor za dopuštanje lokacije na uređaju u svrhu korištenja aplikacije kako je prikazano. (Slika 25)

Ukoliko korisnik odabere “While using the app”, prilikom ponovnog ulaska u aplikaciju prozor se više ne pojavljuje.

Odabirom na “Only this time” aplikacija nastavlja sa radom, ali svaki put kada korisnik ulazi u aplikaciju ponovno se pojavljuje prozor kao što je prikazano na slici.

Ukoliko korisnik odabere “Don´t allow”, što bi značilo da ne dopušta praćenje lokacija za potrebe aplikacije, aplikacija staje sa radom, te se ne pokazuje daljni sadržaj.



Slika 25: Prikaz prozora za dopuštenje loakcije (Izvor: Autor)

Kada korisnik odabere pristup lokaciji pojavljuje mu se sučelje aplikacije. Na ekranu u gorenjem lijevom kutu prikazan mu je tekst “Vremenska prognoza Pula”. (Slika 26)

Ispod toga je prozor na kojem se nalaze podaci o vremenskoj prognozi na odabranoj lokaciji.

U prozoru nalazi se:

- Slika prikaza trenutne prognoze
- Trenutna temperatura
- Tekstualni opis vremena
- Podaci o tlaku
- Podaci o vlazi u zraku
- Jačina vjetra

Ispod prozora o temperaturi, pojavljuje se tekst “Redni broj mobilnih kućica” kako bi aplikacija bila što jednostavnije i preglednija korisniku koji je koristi.

Ispod toga u 5 redova nalaze se redni brojevi mobilnih kućica, u svakom redu za svaku pojedinu kućicu u aplikaciji se ispsuje “Klima”, te prekidač koji ima funkciju da se pomiče desno ili lijevo i zavisi o kriteriju korisnika ili API-a.

U toj ravnini se isto pojavljuje “Svjetlo” sa prekidačem. Prekidač pomaknut u lijevo označava da je resurs ugašen, a pomaknut u desno da je resurs upaljen.

Pozadinski program, koji se pokreće, šalje API svakih 3 minute sa simuliranim podacima senzora.

Korisnik u aplikaciji može po svom odabiru paliti ili gasiti resurse.

Svakih 10 sekundi aplikacija poziva API, te ukoliko promijeni status prekidača nakon 10 sekundi ga vraća na vrijednost koja je pročitana iz API-a. Tek nakon 3 minute se mijenja API i resursi u aplikaciji.

Slika 26: Prikaz komplentne apliakcije (Izvor: Autor)

API za vremensku prognozu je ograničen na 10 000 upita u 24h, ako aplikacija iskoristi sve potrošene poziva sa servera, gornji dio aplikacije neće raditi. (Slika 27)

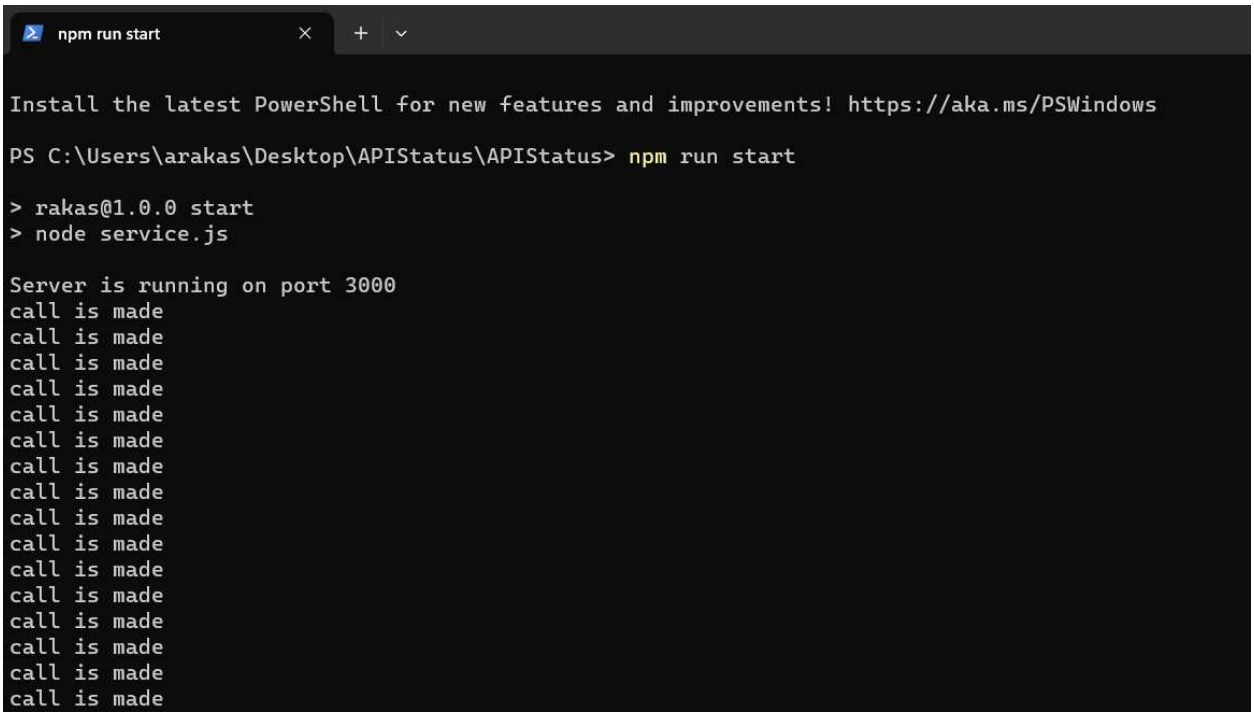
Slika 27: Prikaz neispravne aplikacije (Izvor: Autor)

Prije pokretanja aplikacije potrebno je pokrenuti program koji simulira podatke od senzora. On se pokreće kako bi aplikacija bila funkcionalna i imali podatke o resursima sa kojima korisnik može upravljati.

Program se otvara u terminalnoj konzoli, te se pokreće sa naredbom “npm run start”. (Slika 28)

Ukoliko je program uspješno pokrenut, prikazan je localhost port, te ako aplikacija pozove vrijednosti svakih 10 sekundi se ispiše poruka “call is made”.

Ako program nije pokrenut, prije otvaranja aplikacije, vrijednosti resursa će svi biti “ugašeni”, te se funkcija programa neće odvijati.



Slika 28: Prikaz terminalne konzole (Izvor: Autor)

8. Zaključak

U ovom diplomskom radu predstavljen je razvoj i implementacija aplikacije "Smart Mobile Home" koja omogućava korisnicima daljinsko upravljanje i nadzor njihovih kućanskih uređaja i sistema putem pametnih mobilnih uređaja.

Tijekom razvoja aplikacije, analizirane su tehnologije i protokoli za Internet stvari (IoT) te je implementirana simulirana integracija s uređajima kao što su klima uređaji i svjetlo.

Iako su postignuti značajni rezultati, postoji prostor za daljnje poboljšanje aplikacije. To uključuje proširenje podrške za različite platforme, dodatne sigurnosne funkcije kako bi se osigurala privatnost korisnika i kontinuirano praćenje, te ažuriranje aplikacije kako bi se održala kompatibilnost s najnovijim tehnologijama i uređajima.

Vjeruje se da bi ova aplikacija mogla biti korisna za široku publiku i da bi mogla pridonijeti stvaranju udobnijeg i energetski učinkovitijeg načina života u modernim domovima.

9. Sažetak

Rad obrađuje razvijanje aplikacije sa kojom se može imati kvalitetniji i komotniji boravak tijekom godišnjih odmora turista. Radi konkurentnijeg nastupa na tržištu, ova aplikacija bi mogla doprinijeti u tom osjećaju.

Smart Mobile Home aplikacija je jedna verzija toga.

Korištenjem pametnih telefona ili tableta android platforme se omogućava korisniku da uvidom u vremenske uvjete područja gdje se nalazi smješteni objekt da regulira temperaturne razine unutar objekta, te ima mogućnost osvjetljavanja unutar objekta kao i prilaznog terena.

U radu je prikazana struktura programskog rješenja, te je popraćen cijeli programski tijek sa objašnjavanjima iskodiranog dijela. Vizualnim prikazima prošlo se detaljno kroz cijeli programski kod te se svaki pojedinačni prikaz objasnio.

Programski jezik koji je korišten unutar aplikacije pisan je u programu Kotlin, a API za simulaciju podataka od senzora korišten je Express js.

10. Summary

The paper deals with the development of an application that can be used to have a better quality and more comfortable stay during tourists' vacations. For a more competitive performance on the market, this application could contribute to that feeling.

The Smart Mobile Home app is one version of that.

By using smartphones or tablets of the android platform, the user is enabled to regulate temperature levels inside object by viewing weather conditions of the area where the object is located. It has the option of lighting inside the object as well as the access terrain.

The paper shows the structure of the program solution, and the entire program flow is accompanied with explanations of the coded part. With visual displays, the entire program code was gone through in detail, and each individual display was explained.

The programming language used within the application was written in Kotlin, and the API for simulating sensor data was Express js.

11. Literatura

1. Amazon, (2023) Dostupno na:

https://www.amazon.com/alexa-smart-home/b?ie=UTF8&node=21442899011&ref=pe_alxhub_aucc_en_us_IC_HP_1_HUB_SMA [18.7.2023]

2. 3plus, (2021) Dostupno na:

https://www.3plus.hr/ [18.7.2023]

3. Loxone, (2023) Dostupno na:

https://www.loxone.com/int/smart-home/ [19.7.2023]

4. Peaknx, (2023) Dostupno na:

https://www.peaknx.com/ [24.7.2023]

5. IBM, (2023) Dostupno na:

https://www.ibm.com/topics/internet-of-things [19.7.2023]

6. Keisha Frue, (2019) Dostupno na:

https://pestleanalysis.com/swot-analysis-of-the-internet-of-things/ [10.8.2023]

7. Kotlin, (2023) Dostupno na:

https://kotlinlang.org/docs/android-overview.html [15.8.2023]

8. Express, (2017) Dostupno na:

https://expressjs.com/ [17.8.2023]

9. HivemQ, (2022) Dostupno na:

https://www.hivemq.com/article/mqtt-vs-http-protocols-in-iot-iiot/ [15.9.2023]

12. Popis slika

13. Prilog

1. https://github.com/arakas998/WeatherApp
2. https://github.com/arakas998/APIStatus