SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1458

RAZVOJ KOGNITIVNE USLUGE U PROSTORU INTERNETA STVARI

Dominik Papeš

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1458

RAZVOJ KOGNITIVNE USLUGE U PROSTORU INTERNETA STVARI

Dominik Papeš

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1458

Pristupnik:	**Dominik Papeš (0036543712)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Gordan Ježić

Zadatak: **Razvoj kognitivne usluge u prostoru Interneta stvari**

Opis zadatka:

U današnjem Internetu značajno raste broj jednostavnih uređaja koji imaju mogućnost prikupljanja i odašiljanja podataka. Takvi uređaji Interneta stvari, senzori i aktuatori, korišteni su za različite namjene, primjerice za razvoj pametnih prostora kojima je glavni cilj olakšavanje svakodnevnih radnji korisniku. Vaš zadatak je razviti web-uslugu koja omogućuje korisnicima pregled određenih senzora i aktuatora u prostoru te personaliziranje usluge definiranjem preferencija za određeni kontekst. Na temelju tih preferencija, sustav analizira korisnika i njegove želje te predviđa postavke kako bi maksimalno udovoljio korisnikovim potrebama. Potrebno je izvršiti testiranje funkcionalnosti i analizu razvijene usluge na odabranom studijskom slučaju. Svu potrebnu literaturu i uvjete za rad osigurat će Vam Zavod za telekomunikacije.

Rok za predaju rada: 14. lipnja 2024.

SADRŽAJ

1. Uvod

Pojam Interneta stvari (engl. *Internet of Things, IoT*) odnosi se na mrežu međusobno povezanih uređaja koji se u ovom kontekstu nazivaju "stvarima". Uređaji na koje se pojam odnosi najčešće su okarakterizirani činjenicom da sadrže neku vrstu senzora za mjerenje određenih parametara te određenu vrstu programske podrške i ostalih tehnologija koje im omogućavaju međusobnu komunikaciju, interakciju i razmjenu podataka bilo međusobno ili izravno s korisnicima. Neki primjeri "stvari" su uređaji za praćenje zdravstvenog stanja, komponente sigurnosnih sustava, komponente povezane s automatiziranom vožnjom i slično. Posebno uređaji "stvari" u kontekstu ovog rada bit će pametni kućanski sustavi osvjetljenja, grijanja i klimatizacije prostora.

Pojam Interneta stvari definirao je 1999. Kevin Ashton izjavivši kako su u tom trenutku gotovo sve podatke na Internetu bili stvorili i organizirali ljudi te da se u mnogim zamislima Interneta zanemaruje činjenica da su ljudi spori, griješe i nemaju vremena za razliku od strojeva [3]. Predvidio je kako će računala moći, pomoću podataka koje generiraju ljudi, a računala pasivno prikupljaju, pratiti stanja drugih stvari, znati kada je određene komponente potrebno zamijeniti i općenito smanjiti troškove, gubitke i količinu otpada. Smatrao je kako će Internet stvari donijeti novu revoluciju i promijeniti svijet baš kao što je to učinio i sam Internet. Pokazalo se kako su se ta predviđanja donekle i ostvarila, već 2003. broj povezanih uređaja diljem svijeta iznosio je oko 500 milijuna, tek 7 godina kasnije, u 2010. taj broj iznosio je 12,5 milijardi, a danas iznosi približno 50 milijardi te nastavlja rasti [5].

Uređaji Interneta stvari u praksi su obogaćeni takozvanom pametnom programskom podrškom koja omogućava smanjenu interakciju čovjeka i samog sustava te je u konačnici cilj smanjiti tu interakciju na najmanju moguću razinu i da sama integracija sustava bude neprimjetna krajnjem korisniku. Kako bi se takav neprimjetan sustav mogao integrirati u kontekstu ovog rada koristit će se neuronska mreža treće strane, Keras.io. Neuronska mreža omogućava predviđanje korisničkih postavki na temelju već postojećeg konteksta. Prate se korisničke postavke prilikom njihove svake promjene te se spremaju u bazu konteksta zajedno s ostatkom parametara poput doba dana, tempe-

rature i slično. Na temelju tih podataka usluga automatski određuje postavke kada ih korisnik sam ne definira.

Usluge Interneta stvari mogu učiniti svakodnevni život učinkovitijim, prilagođenijim i učinkovitijim. Također pametne usluge Interneta stvari posebno olakšavaju život osobama s invaliditetom na način da određene poslove uređaji obavljaju autonomno i bez korisničkog unosa te se prilagođavaju korisnikovim potrebama.

Cilj ovog rada je razvoj inteligentne web usluge u prostoru Interneta stvari koja korisnicima omogućava pregled spojenih uređaja i personaliziranje postavki za svaki pojedini uređaj. Sustav na temelju korisničkih preferencija mora imati sposobnost predviđanja postavki kada korisnik to ne učini sam. Prostor Interneta stvari na koji se rad odnosi vezan je uz održavanje takozvane pametne kuće, SmartHome. Sustav omogućava podešavanje osvjetljenja i temperature odnosno klimatizacije prostora. Uz sam razvoj usluge jedan od ciljeva je i analiza usluge na studijskom slučaju.

2. Sredstva za razvoj inteligentnih usluga

2.1. Automatizacija doma

Područje Interneta stvari koje se bavi razvojem usluga koje pružaju povezanost kućanskih uređaja i njihovog autonomnog djelovanja naziva se automatizacija doma. Razvoj kućanskih uređaja započeo je krajem 19. i početkom 20. stoljeća. Daljnjim razvojem tehnologije, pojavom osobnih računala i umrežavanja istih pojavila se i potreba za umrežavanjem kućanskih tehnologija.

Počeci kućanskih automatizacija sežu iz vremena 70-ih godina prošlog stoljeća razvitkom X10 tehnologije, sustav za upravljanje uređajima koji upravljačke signale šalje putem električnih vodova na koji su priključeni kompatibilni uređaji [13].

U 90-im godinama prošlog stoljeća povećava se broj istraživanja u području gerontehnologije, kompozit riječi gerontologija i tehnologija. Gerontologija je interdisciplinarno znanstveno područje koje se bavi proučavanjem psiholoških, zdravstvenih i socijalnih problema starenja i starih ljudi. Tehnologija se u ovom kontekstu odnosi na znanstvenu disciplinu koja se bavi istraživanjem te razvojem tehnika i proizvoda. Gerontehnologija je dakle disciplina koja se bavi istraživanjem i razvojem različitih tehnika i proizvoda zasnovanih na znanstvenim spoznajama o procesu starenja, a relevantna je u kontekstu pametnih kuća jer tehnologije Interneta stvari mogu značajno olakšati život starijim osobama.

Automatizacija doma postaje značajno komercijalno dostupnija početkom 21. stoljeća zahvaljujući povećanju dostupnosti osobnih računala i Interneta koji postaju široko rasprostranjeni. Računala postaju manja i prenosivija. Razvoj mobilnih uređaja dodatno je omogućio efikasniji razvoj usluga automatizacija kao i bolje korisničko iskustvo.

Glavne komponente sustava automatizacije doma su korisničko sučelje, način prijenosa, centralni kontroler i uređaji, odnos između njih prikazan je na slici 2.1. Koris-

ničko sučelje pruža korisniku način upravljanja sustavom i prikaz bitnih informacija o sustavu. Najčešće je ostvareno ugrađenim monitorom, osobnim računalom ili mobitelom. Način prijenosa može biti žični (npr. Ethernet) ili bežični (npr. Bluetooth, WiFi). Centralni kontroler je sklopovsko sučelje koje komunicira s korisničkim sučeljem i upravlja spojenim uređajima. Sadrži određenu programsku podršku koja omogućava funkcionalnosti. Uređaji su spojeni na centralni kontroler i obavljaju neku funkciju, npr. lampe, klima uređaji ili grijanje.

Slika 2.1: Sustav automatizacije doma

2.2. Objekti, senzori i aktuatori

Arhitektura Interneta stvari može se najjednostavnije podijeliti u 3 sloja: percepcijski sloj, transportni sloj i aplikacijski sloj, prikazano na slici 2.2 . Radni princip ove hijerarhijske struktura započinje s percepcijskim slojem na kojem se prikupljaju podaci iz povezanih stvari pomoću tehnologija kao što su na primjer senzori. Prikupljeni podaci se prenose u transportnom sloju do aplikacijskog sloja. Transportni sloj koristi dobro poznate protokole koji se koriste u Internetu i lokalnim mrežama za transport podataka. U aplikacijskom sloju se prikupljeni podaci analiziraju i procesuiraju te se poduzimaju određene akcije ovisno o vrsti podataka i uređaja o kojem je riječ.

Objekt se u prostoru Interneta stvari poistovjećuje sa stvarima. Stvari, odnosno objekti u ovom kontekstu odnose se na bilo koji elektronički uređaj koji se može spojiti na Internet i primati odnosno slati podatke ili obavljati određene akcije. Objekti se mogu podijeliti na pametne i nepametne. Nepametni objekti mogu se podijeliti na senzore i aktuatore, a pametni objekti su u pravilu kompozicije nepametnih objekata.

Senzori su elektronički uređaji koji imaju sposobnost mjerenja stanja i promjena u okolišu. Postoje različite vrste senzora ovisno o namjeni. Senzori blizine mjere udaljenost predmeta od senzora bez potrebe za fizičkim kontaktom što se postiže emitiranjem određene vrste elektromagnetskog zračenja, npr. infracrvenog te se udaljenost određuje na temelju varijacija u povratnom signalu, ovi senzori koriste se u sustavima za parkiranje i sličnim situacijama. Pozicijski senzori mogu detektirati poziciju predmeta ili osobe u određenom području, primjena ovakvih senzora česta je u području

Slika 2.2: Slojevita arhitektura Interneta stvari

kućne sigurnosti te se koriste kako bi detektirali moguće uljeze. Senzori prisutnosti mogu detektirati ljudsku prisutnost ili prisutnost predmeta u određenom području, koriste se kako bi se automatski regulirali parametri poput temperature ili vlage u zraku ovisno o tome nalaze li se u prostoriji ljudi. Senzori detekcije pokreta detektiraju pokrete koriste se u automatskim vratima i automatskim svjetlima, ali mogu se koristiti i u području kućne sigurnosti na način da se spoje na kameru i prilikom detekcije pokreta aktiviraju kameru te se područje fotografira ili započinje video. Temperaturni senzori mjere promjene u temperaturi, između ostalog koriste se i u klima uređajima kako bi se prepoznala željena temperatura i zaustavio rad klima uređaja. Senzori vlage mogu detektirati promjenu vlage u zraku što se također može koristiti u području automatizacije doma i pametnih kuća za automatsku regulaciju vlage u zraku.

Aktuatori su uređaji koji rade neku konkretnu akciju temeljem određenog signala. Akcija može biti mehaničke prirode ili u obliku slanja signala. Primjeri mehaničkih aktuatora su različite vrste motora, servomotora ili bombi. Druga vrsta aktuatora su aktuatori koji između ostalog šalju poruke, upravljaju LED lampicama ili kontroliraju kretnje robota.

Pametni objekti dakle mogu sadržavati senzore ili aktuatore ili oboje, imaju ugrađen operacijski sustav te pritom imaju sposobnost "razmišljanja" što u ovom kontekstu znači da mogu komunicirati s ostalim objektima u prostoru, mogu procesuirati podatke koje dobe od senzora i ovisno o vrijednostima tih podataka poslati signale koji aktiviraju prikladne aktuatore. Sukladno tome M. Woolridge definira pojam programskog agenta kao računalni sustav koji ima mogućnost komunikacije s okolišem i sposobnost donošenja autonomnih odluka u ime korisnika sustava [36]. Odnosno riječ je

o programskoj komponenti koja može samostalno funkcionirati i donositi samostalne odluke u okolišu kojeg okupiraju i druga programska rješenja bez obaveznog unosa korisnika za određenom akcijom. Osim inteligencije u obzir treba uzeti i sljedeća dva svojstva agenata: mobilnost agenta i društvene sposobnosti. Društvene sposobnosti agenta odnose se na njegovu mogućnost da komunicira i surađuje s ostalim agentima prilikom izvršavanja svojih zadataka. Mobilnost se odnosi na mogućnost agenta da migrira, privremeno ili trajno, na drugi čvor mreže u kojoj se nalazi.

2.3. Svjesnost konteksta

Kontekst se može definirati na nekoliko načina na primjer kontekst su okolnosti koje formuliraju neki događaj, izraz ili ideju te omogućuju da bude u potpunosti shvaćen, ili je kontekst situacija u kojoj se nešto događa te to omogućuje shvaćanje tog događaja. Još jedna od definicija govori kako je kontekst skup činjenica i okolnosti koje okružuju određenu situaciju ili događaj [19]. Kontekst u smislu kontekstno svjesnog programiranja i pokretnih agenata definira se kao bilo koja informacija kojom se može okarakterizirati situacija u kojoj se određeni objekt nalazi obzirom na njegovu fizičku, društvenu, komunikacijsku i programsku okolinu.

Kontekstno svjesno programiranje prvi put se kao problem istraživanje pojavljuje 1991. u radu M. Weisera [35], u tom radu ujedno je i najavio pojavu sveprisutnoga računarstva. Sveprisutno računarstvo po Weiseru temelji se na nekoliko principa, neki od kojih su:

1. Računalo je tu da korisniku pomogne nešto napraviti, a ne da se korisnik bavi računalom.

2. Najbolje računalo ono je koje se ne primjećuje.

3. Služenje računalom mora biti intuitivno.

4. Tehnologija mora pružiti smirenje.

Jasno je iz principa na kojima se gradi ideja sveprisutnog računarstva da je riječ o zahtjevnom problemu jer povećava složenost sustava koji pokušava implementirati ovakvo rješenje te utječe na funkcionalnosti i efikasnost ovakvog rješenja. Prije spomenuti autonomni inteligentni softverski agenti omogućuju izvedbu sveprisutnog računarstva umrežavanjem agenata u višeagentski sustav.

3. Postojeće koginitivne usluge za upravljanje uređajima u pametnom prostoru

Pojam "pametne kuće" usko je vezan uz tehnologije Interneta stvari, stoga ne treba čuditi kako je tržište aplikacijama zasićeno mnogim aplikacijama koje pružaju mogućnosti upravljanja kućnim uređajima poput klima uređaja, sustava za grijanje, osvjetljenja i kućnih pomagala. Cilj tih aplikacija je korisniku pružiti jednostavno i univerzalno sučelje za lakše upravljanje spojenim uređajima.

Najčešće takve aplikacije integriraju i mogućnosti upravljanja ne samo kućanskim uređajima, već i ostalim uređajima poput pametnih satova i ostalih nosivih uređaja. Nerijetko su popraćene i takozvanim inteligentim glasovnim odnosnom virtualnim asistentima poput Appleove Siri, Amazonove Alexe, Googleovog pomoćnika, Microsoftove Cortane ili Samsungovog Bixbyja koji omogućavaju korištenje aplikacije glasovnim naredbama. Time primjene tih aplikacija postaju šire od samo upravljanja spojenim uređajima, već se asistenti mogu koristiti za akcije poput slanja i čitanja tekstualnih poruka i e-pošte, pokretanje poziva, odgovaranja na jednostavne upite poput provjere vremenske prognoze, postavljanje alarma i podsjetnika, kontroliranje toka medijskog sadržaja s različitih izvora kao što su YouTube, Spotify, Netflix ili lokalne datoteke. Tipičan način na koji funkcioniraju takvi asistenti je da ih korisnik aktivira pobudnom riječi nakon čega prate korisnikov govor, procesuiraju ga i na temelju toga pokreću određenu akciju.

U nastavku će ukratko biti predstavljeni neki primjeri postojećih sustava kućne automatizacije.

3.1. Apple Siri

Vjerojatno najpoznatiji glasovni asistent i onaj koji je većini sinoniman s pojmom glasovnih asistenata je Appleova Siri. Siri kao koncept nastaje u 80-im godinama prošlog stoljeća u radu John Scullyja, tadašnjeg glavnog izvršnog direktora Applea od 1983. do 1993. John Scully predlaže koncept "Navigatora znanja" (engl. *Knowledge navigator*) kojeg opisuje kao digitalni pomoćni uređaj koji će biti sposoban pristupiti velikoj umreženoj bazi hipertekstualnih informacija i koristiti softverske agente kako bi pomogao u pretraživanju istih. Sculley je gledao na izgradnju Navigatora znanja kao "ovisnu o daljnjem razvoju važnih tehnologija", poput poboljšanih baza podataka, pretraživanja informacija i, naravno, napretka u istraživanju umjetne inteligencije [29].

Siri zapravo nije bila izumljena i izgrađena od strane Applea, već je nastala kao jedan od ogranaka DARPA-inog [1] CALO [2] projekta. Agencija za napredne istraživačke projekte obrane je američka vladina agencija stvorena 1958. godine za poticanje istraživanja tehnologije s potencijalnim vojnim primjenama. Većina DARPA-inih projekata su povjerljive tajne, ali mnoge od njezinih vojnih inovacija imale su veliki utjecaj na civilni svijet, osobito u područjima elektronike, telekomunikacija i računalnih znanosti. Vjerojatno je najpoznatija po ARPANET-u[3], ranom mrežnom sustavu računala za dijeljenje vremena koji je postavio temelje za Internet [7]. Apple je pod vodstvom Stevea Jobsa kupio Siri 2010. te je 2011. postala jedna od standardnih aplikacija i usluga integriranih u iOS počevši s iPhonoeom 4[16].

Siri je u to vrijeme bila revolucionaran izum koji je korisnicima po prvi put omogućio beskontaktno, glasovno upravljanje pametnim telefonom. Ranije verzije Siri bazirale su svoje funkcionalnosti na upravljanjem iPhone uređaja, no razvojem tehnologija Interneta stvari Siri danas podržava i mogućnosti upravljanja uređajima pametne kuće.

Programabilnost Siri nalazi se u SiriKitu i takozvanim "planovima" (engl. *Intent*). Planovi su posebno definirana sučelja za specifične aplikacije koji definiraju načine na koji Siri treba procesuirati određenu korisničku naredbu.

[1] (engl. *Defense Advanced Research Projects Agency*)

[2] (engl. *Cognitive Assistant that Learns and Organizes*), Agencija za napredne istraživačke projekte obrane

[3] (engl. *Advanced Research Projects Agency Network*)

3.2. Amazon Alexa

Amazon Echo je bežični pametni zvučnik s glasovnim upravljanjem koji je razvila tvrtka Amazon. Uređaj se povezuje s glasovno kontroliranim inteligentnim osobnim asistentom Alexa, koji odgovara na ime "Alexa". Uređaj je sposoban za glasovnu interakciju, reprodukciju glazbe, izradu popisa zadataka, postavljanje alarma, strujanje podcasta, reprodukciju audioknjiga te pružanje informacija o vremenu, prometu i drugih informacija u stvarnom vremenu. Također može kontrolirati nekoliko pametnih uređaja koristeći se kao središnji hub za automatizaciju doma. Ono što je u početku odvajalo Amazon Alexu od ostalih inteligentnih glasovnih i virtualnih asistenata kao što su Apple Siri, Google Home i Microsoft Cortana bila je neprisutnost zaslona na uređaju Amazon Echo. Na taj način postala je platforma odvojena od uobičajenih platformi za razvijanje asistenata poput iOS-a, Androida i Windows operacijskih sustava [37].

2004. godine ustanovljen je Amazon Lab126, Amazonov tim za razvoj računalnog sklopovlja. Nakon 3 godine istraživanja i razvijanja na tržište su plasirali svoj prvi proizvod globalnog uspjeha, Amazon Kindle, e-čitač [37]. Neki od drugih proizvoda Amazonovog Lab126 prije Amazon Echoa i Alexe su Kindle Fire tablet, Amazon Fire TV i Fire Phone.

Amazon Echo može se koristiti u kombinaciji s Alexa Skills Kit radnim okvirom za programiranje takozvanih vještina (engl. *skills*). Vještine su aplikacije za Amazon Alexu, glasovno su upravljane što korisnicima omogućava uporabu aplikacija bez upotrebe ruku. Vještine obuhvaćaju različite akcije poput svakodnevnih aktivnosti poput pretraživanja vijesti, slušanja glazbe, igranja igara i slično. Vještine također podržavaju upravljanje uređajima koji su povezani u oblaku, na primjer korisnici mogu glasovno zatražiti paljenje svjetla ili promjenu temperature ako su ti uređaji spojeni na Alexine čime se Amazon Alexa može smjestiti u područje automatizacije doma jer olakšava svakodnevne radnje.

Korisnik započinje interakciju s Alexom tako da izgovara riječ pobude (engl. *wake word*), "Alexa" i govori u Alexa podržan uređaj. Uređaj šalje govor prema Alexa usluzi u oblaku gdje se govor prepoznaje, određuje se što korisnik želi učiniti i šalje se POST zahtjev za pokretanjem vještine koja ispunjava korisnikove zahtjeve, POST zahtjev sadrži JSON[4] tijelo koje sadrži parametre potrebne za pravilan rad vještine. Kod programiranja vještine programer ne mora brinuti o implementaciji algoritma za procesuiranje govora, taj dio usluge obavlja se u Alexinom oblaku, ali programer mora

[4](engl. *JavaScript Object Notation*)

dobro definirati interakcije glasovnog modela odnosno očekivane načine na koji će korisnik interagirati s aplikacijom [31].

3.3. SmartThings

SmartThings je američka tvrtka koja se bavi tehnologijama Interneta stvari osnovana 2012. godine te kupljena od strane Samsung Electronics 2014 [32]. Vodeći proizvod tvrtke je istoimena aplikacija za iOS i Android uređaje koja služi za upravljanje povezanim uređajima. Podržan je veliki raspon uređaja od nosivih uređaja kao što su pametni satovi do kućanskih aparata poput perilica rublja.

Aplikacija služi kao svojevrsni daljinski upravljač za upravljanje, ali omogućava i specifičnije akcije poput stvaranja rutina za pojedine uređaje odnosno automatiziranje zadataka [10]. Uređaji se pronalaze i povezuju putem Bluetootha ili WiFi-ja, nakon čega je upravljanje njima moguće putem WiFi-ja. Podržani su svi uređaji građeni na Matter protokolu, neovisno o tvrtki koja ih proizvodi.

Matter protokol, prethodno znan kao CHIP[5], je komunikacijski standard za poboljšanje interoperabilnosti između različitih klasa uređaja Interneta stvari [38]. CSA[6] nadgleda razvoj projekta, a sam cilj projekta je uspostaviti industrijski uniformni standard za komunikaciju uređaja Interneta stvari. Proizvođači koji slijede Matter standard mogu povezati svoje uređaje s uređajima drugih proizvođača te s druge strane korisnici mogu koristiti jednu aplikaciju za upravljanje svim uređajima koji koriste Matter protokol umjesto da moraju koristiti različite aplikacije za svakog pojedinog proizvođača uređaja u njihovom vlasništvu. Također, korisnici ne moraju kupovati zasebnu opremu za spajanje različitih uređaja već koriste standardizirana sučelja [38].

Matter protokol na transportnom i mrežnom sloj može koristiti protokol Thread. Thread je protokol razvijen kao isprepletena mreža, odnosno mreža "svaki sa svakim" (engl. *mesh network*). U pozadini koristi mrežni protokol IPv6 i dobro je prilagođen za uređaje Interneta stvari jer dobro radi na niskom napajanju stoga je odličan izbor za uređaje s baterijom [1].

Samsungova vizija nesmetane povezanosti inspirirana je filozofijom Smirene tehnologije (engl. *Calm technology*) čija je zamisao automatska suradnja uređaja, tako da korisnici mogu uštedjeti vrijeme na postavljanju i odmah uživati u iskustvu. Kako bi ostvario ovu viziju, Samsung je preoblikovao SmartThings i njegove povezane usluge i partnerstva, uključujući Samsungov Hub Everywhere, proširujući njegove mogu-

[5](engl. *Project Connected Home over IP*)

[6](engl. *Connectivity Standard Alliance*)

nosti na cijeli pametni dom s audio i vizualnim podacima, kao i SmartThings Energy, SmartThings Pet i SmartThings Cooking. Samsung je također uspostavio partnerstvo s Philips Hue kako bi integrirao Philips Hue Sync izravno u Galaxy uređaje, omogućujući pametnoj kućnoj rasvjeti da se uskladi s glazbom [10].

Unutar aplikacije SmartThings integriran je i Samsungov glasovni asistent Bixby. Bixby se poput ostalih glasovnih asistenata koristi na način da korisnik izgovori pobudnu riječ koja je u ovom slučaju "Hi, Bixby" nakon čega korisnik verbalno zadaje zadatak. Bixby također podržava opciju za pisanje novih aplikacija koje koriste Bixbyjev API u obliku Bixby Home Studija.

4. Pregled korištenih tehnologija

U nastavku će biti predstavljene ključne tehnologije, radni okviri i biblioteke korištene u implementaciji aplikacije.

4.1. Vite

Vite je moderni gradivni alat i razvojni server za frontend projekte, kreiran od strane Evana Youa, autora Vue.js radnog okvira. Ime "Vite" dolazi od francuske riječi za brzinu, što je ujedno i njegova glavna karakteristika. Vite je dizajniran kako bi pružio izvanredno brzo vrijeme pokretanja i brze nadogradnje tijekom razvoja [34]. Zahvaljujući HMR[1] sustavu, Vite omogućuje brze nadogradnje bez potrebe za ponovnim učitavanjem cijele stranice što značajno ubrzava brzinu razvoja jer uklanja potrebu za čekanjem da se učita nova verzija stranice. Vite pomaže u brzoj inicijalizaciji projekta te dolazi s uključenim opcijama u obliku CLI[2] alata za stvaranje React projekta temeljenog na JavaScriptu ili TypeScriptu te za stvaranje Vue.js projekta koji također može biti temeljen na JavaScriptu ili TypeScriptu.

4.2. React

React je popularna JavaScript knjižnica odnosno biblioteka za izgradnju korisničkih sučelja, razvijena od strane Facebooka 2013. godine. Njegova glavna svrha je olakšavanje izrade interaktivnih, dinamičnih i responzivnih web aplikacija. React se temelji na komponentnom pristupu, omogućujući programerima da kreiraju male, samostalne dijelove korisničkog sučelja koji se mogu ponovno koristiti i kombinirati za izgradnju složenih aplikacija.

Srž Reacta su komponente. Komponente su neovisni, ponovno iskoristivi dijelovi koda koji predstavljaju dijelove korisničkog sučelja. Svaka komponenta može

[1] (engl. *Hot Module Replacement*)
[2] (engl. *Command Line Interface*)

sadržavati svoju logiku i stanje, što omogućuje modularni razvoj aplikacija. Stanje (engl. *state*) i svojstva (engl. *properties*) su ključni za rad s podacima unutar React komponenti. Stanje predstavlja lokalno stanje komponente koje se može mijenjati tijekom vremena, dok su svojstva ulazni podaci koje komponenta prima od roditeljske komponente. Komponente se mogu definirati u obliku funkcija ili klasa.

Klasne komponente pružaju bolju optimizaciju performansi zbog ponovnog korištenja instanci i ažuriranja samo onih dijelova za koje je to potrebno što ih čini boljim izborom za kompleksna sučelja s velikim skupom podataka. Također klasne komponente upravljaju vlastitim internim stanjem pomoću metode *setState*. Ova metoda omogućuje klasnim komponentama ažuriranje i rukovanje podacima koji se mijenjaju tijekom vremena, što omogućuje ponovno renderiranje komponente kada se stanje promijeni te podržavaju precizniju kontrolu nad ponašanjem.

Funkcijske komponente su jednostavniji oblik React komponenti. U osnovi su JavaScript funkcije koje nemaju vlastito unutarnje stanje. Ovisne su o svojstvima odnosno propovima (engl. *properties*) za primanje podataka i vraćanje JSX[3] koda koji predstavlja izlaz komponente.

Funkcijske komponente su postale popularne s uvođenjem React Hookova koji su omogućili klasnim komponentama bez stanja da rukuju stanjem i funkcionalnostima životnog ciklusa. Uvedeni u verziji 16.8, hookovi su smanjili razliku između funkcionalnih i klasnih komponenti. Od tog trenutka, funkcionalne komponente mogle su raditi ono što su mogle i klasne komponente, ali su bile jednostavnije i lakše za ponovnu upotrebu. U sklopu ovog rada koristile su se upravo funkcijske komponente zbog njihove jednostavnosti [27].

Prije spomenuti JSX je sintaksna ekstenzija za JavaScript koja omogućuje pisanje HTML-a[4] unutar JavaScript koda. JSX olakšava kreiranje React komponenti kombiniranjem JavaScript logike s HTML strukturom.

React koristi virtualni DOM[5]. DOM je podatkovna reprezentacija objekata koji sačinjavaju strukturu i sadržaj dokumenta na webu [21]. Virtualni DOM je programski koncept u kojem se idealna ili virtualna reprezentacija korisničkog sučelja čuva u memoriji i sinkronizira s pravim DOM-om putem knjižnice kao što je ReactDOM. Ovaj proces se naziva usklađivanje (engl. *reconciliation*). Ovakav pristup omogućuje deklarativni API Reacta na način da se React sam pobrine da DOM odgovara željenom stanju. To apstrahira manipulaciju atributima, rukovanje događajima i ručno ažuriranje

[3](engl. *JavaScript XML*)

[4](engl. *HyperText Markup Language*)

[5](engl. *Document Object Model*)

DOM-a koje bi se inače moralo koristiti za izgradnju aplikacije [27].

4.3. Typescript

TypeScript je nadskup JavaScripta koji uvodi statičko tipiziranje i dodatne značajke koje olakšavaju razvoj složenih i velikih aplikacija. Razvijen od strane Microsofta i prvi put objavljen 2012. godine, TypeScript se brzo etablirao kao moćan alat za programere koji žele iskoristiti sve prednosti JavaScripta, uz dodatnu sigurnost koju pruža statičko tipiziranje.

TypeScript dodaje tipove varijablama, funkcijama i objektima. Tipovi omogućuju otkrivanje grešaka tijekom razvoja, prije nego što se kod pokrene, što povećava pouzdanost i održivost koda.

Sučelja (engl. *interface*) u TypeScript-u definiraju strukturu objekata. Ona služe kao ugovori koje objekti moraju zadovoljiti, omogućujući provjeru tipova i automatsko dovršavanje u razvojnim alatima [33].

4.4. Bootstrap & React Bootstrap

Bootstrap je jedan od najpopularnijih CSS[6] radnih okvira za razvoj responzivnih i mobilnih web stranica. Prvobitno razvijen od strane Twittera, Bootstrap je postao open-source projekt i široko prihvaćen alat u web zajednici. Njegova jednostavnost korištenja, bogata kolekcija komponenti i fleksibilan sustav rešetke (engl. *grid system*) čine ga idealnim izborom za brzi razvoj modernih web stranica. Za uključivanje Bootstrapa unutar nekog projekta može se koristiti CDN[7] ili se Bootstrap može instalirati pomoću nekog od upravitelja paketa kao što je *npm*[8] [6].

Jedna od glavnih značajki Bootstrapa je njegov rešetkasti sustav koji omogućava lako raspoređivanje sadržaja na stranici. Sustav se temelji na redovima (engl. *rows*) i stupcima (engl. *columns*) kojih na stranici ima 12. Kombiniranjem različitih klasa, može se definirati kako će se sadržaj rasporediti na različitim veličinama ekrana.

Također jedan od glavnih razloga zašto je Bootstrap bio odabran u izradi ovog projekta su njegove preddefinirane komponente i utilitarne klase. Gotove Bootstrap komponente omogućuju brz način stvaranja korisničkog sučelja uniformnog dizajna

[6](engl. *Cascading Style Sheets*)
[7](engl. *Content Delivery Network*)
[8](engl. *Node Packet Manager*)

u rasponu cijele aplikacije. Ove komponente uključuju navigacijske trake (engl. *navbar*), kartice (engl. *cards*), modale (engl. *modals*) , gumbe (engl. *buttons*), obrasce (engl. *forms*) i mnoštvo drugih. Utilitarne klase omogućuju lagano i brzo podešavanje stilova bez potrebe za pisanjem dodatnog CSS-a. Komponente i utilitarne klase koriste se navođenjem u *class* odnosno u slučaju korištenja React biblioteke *className* polja u oznaci elementa (engl. *tag*).

U projektima koji koriste React biblioteku može se instalirati neka od React inačica Bootstrapa, u ovom projektu bio je korišten React Bootstrap [26], koji spaja Reactov deklarativni način pisanja komponenti s preddefiniranim komponentama Bootstrapa. Komponente su implementirane kao React komponente, što olakšava njihovu upotrebu i prilagodbu unutar React aplikacija.

4.5. Django & Django Rest

Django je besplatan web radni okvir otvorenog koda temeljen na Pythonu koji potiče brz razvoj i pragmatičan dizajn koji se primarno koristi za razvoj koda koji se pokreće na poslužiteljskoj strani. Razvoj ovog radnog okvira započeo je u jesen 2003. iz frustracije razvojnog tima u Lawrence Journal-World novinama iz Lawrencea, Kansas u SAD-u, koji je bio pod pritiskom čestih rokova i mnoštvom stalnih zahtjeva za dodavanjem novih funkcionalnosti i izgradnjom novih aplikacija. Kako bi si olakšali razvoj novih aplikacija i njihovo održavanje započeli su s razvojem radnog okvira koji će uokviriti sve najbitnije aspekte web aplikacija i omogućiti brz i jednostavan način izgradnje istih. Radni okvir postao je javno dostupan 2005. godine, a svoje ime dobio je po jazz gitaristu Djangu Reinhardtu. Danas je za razvoj Djanga zaslužena neovisna neprofitna organizacija DSF[9] [15].

Preporučen način instalacije Django radnog okvira je pomoću paketnog upravitelja za Python, *pip* unutar posebnog Python virtualnog okruženja (engl. *virtual environment*). Virtualna okruženja su preporučeni način za pisanje bilo kakvog koda u Pythonu koji iziskuje instaliranje dodatnih paketa jer se na taj način sistemski, odnosno globalni Python interpreter odvaja od interpretera specifičnog za tu pojedinu aplikaciju. Prilikom instalacije dodatnih paketa kao što je to u ovom slučaju Django bi se bez virtualnog okruženja paket instalirao na cijelom sustavu što može dovesti do nekompatibilnosti s ostalim već instaliranim paketima i u konačnici do neočekivanih grešaka u radu sustava drugih nepovezanih aplikacija.

[9](engl. *Django Software Foundation*)

Django slijedi MVC[10] arhitekturu (Model-Pogled-Kontroler), ali koristi drugačiju terminologiju, MVT[11] (Model-Pogled-Predložak) što može biti zbunjujuće s obzirom da se "pogledi" (engl. *views*) u oba konteksta odnose na različite pojmove. Pojam modela odnosi se na dio aplikacije zaslužen za komunikaciju s bazom podataka koristi ORM[12] koji omogućava definiciju tablica koje se koriste u bazi podataka bez potrebe za eksplicitnim pisanjem SQL koda, već se tablice definiraju kao klase u programskom jeziku u kojem je napisan ostatak radnog okvira, u ovom slučaju riječ je dakle o Pythonu. Pogledi u kontekstu Django radnog okvira zapravo preuzimaju ulogu kontrolera iz MVC arhitekture. U pogledima je sadržana poslovna logika koja se obavlja prilikom poziva određene funkcije. Konačno, predlošci (engl. *template*) omogućuju pisanje korisničkog sučelja i odgovaraju pojmu pogleda iz MVC arhitekture.

Djangov pretpostavljeni način rada temelji se na ideji da će korisničko sučelje biti pisano unutar Django aplikacije i dostavljeno do klijenta putem Django predložaka. U slučaju implementacije aplikacije na ovom projektu nije bio korišten takav način rada, već je korisničko sučelje napisano u zasebnoj React aplikaciji, stoga je za potrebe ovog projekta potrebno instalirati popratni radni okvir Django radnom okviru imena Django Rest Framework. Django Rest radni okvir omogućuje jednostavnu implementaciju REST API-ja unutar Django aplikacije. Django Rest također se lagano instalira paketskim upraviteljem *pip* te ga je isto tako preporučeno instalirati u virtualnom okruženju [8]. Kako bi se Django Rest mogao pravilno koristiti potrebno je definiriati serijalizatore, funkcije koje prevode modele iz ORM-a u JSON objekte ili u neki drugi prikladni format za slanje preko mreže. Radni okvir također podržava pretpostavljene postavke za autentikaciju i autorizaciju korisnika uključujući tokensku autorizaciju, OAuth1a i OAuth2. Model korisnika je preddefiniran te ga je moguće prilagoditi potrebama projekta. Funkcije pogleda mogu se prilagoditi dekoratorima koji između ostalog definiraju dopuštenu REST metodu i provjeravaju razinu autorizacije korisnika.

4.6. Tensorflow & Keras

TensorFlow je fleksibilna i skalabilna softverska knjižnica za numeričke izračune te pritom koristi dijagramske tokove podataka. Ova knjižnica i povezani alati omogućuju korisnicima učinkovito programiranje i treniranje neuronskih mreža i drugih modela

[10](engl. *Model-View-Controller*)
[11](engl. *Model-View-Template*)
[12](engl. *Object Relational Mapper*)

strojnog učenja te njihovo postavljanje u proizvodnju. TensorFlow je razvijen od strane Googlea. Prvotno je lansiran 2015. godine, a njegov glavni cilj je omogućiti jednostavno treniranje i implementaciju modela dubokog učenja. TensorFlow podržava širok spektar zadataka u strojnom učenju, uključujući regresiju, klasifikaciju, klasteriranje i generativne modele. Osnovni algoritmi TensorFlowa napisani su u visoko optimiziranom C++ i CUDA[13], platformi za paralelno računanje i API-ju koji je kreirala NVIDIA. API-ji su dostupni na nekoliko jezika, među kojima je Python API najstabilniji i najkompletniji. Ostali službeno podržani jezici uključuju JavaScript, C++, Javu, Go i Swift. Treće strane nude pakete za više jezika poput C# i Ruby [25].

Keras je visoko razinski API za duboko učenje koji radi povrh TensorFlowa. Razvijen je s ciljem da bude jednostavan za korištenje i omogućava brzi razvoj modela neuronskih mreža. Keras je posebno popularan zbog svoje jednostavnosti i čitljivosti, omogućujući istraživačima i inženjerima da brzo i učinkovito razvijaju složene modele dubokog učenja. Iako je trenutno najčešće korišten s TensorFlowom, Keras podržava i druge backendove kao što su Theano i CNTK[14].

Kako bi se mogla odabrati optimalna arhitektura mreže zajedno s optimalnim hiperparametrima za dani zadatak može se koristiti dodatna biblioteka KerasTuner [24]. Hiperparametri su ključni aspekti koji mogu značajno utjecati na performanse modela strojnog učenja, a ručno podešavanje može biti vremenski zahtjevno i neučinkovito. Keras Tuner olakšava ovaj proces pružajući sustavne metode za pronalaženje najboljih kombinacija hiperparametara za određeni model.

[13](engl. *Compute Unified Device Architecture*)

[14](engl. *Microsoft Cognitive Toolkit*)

5. Implementacija web aplikacije

Web aplikacija implementirana u sklopu ovog rada generičkog imena SmartHome omogućava prijavljenom korisniku upravljanje klima uređajima i svjetlima koja se nalaze u sobama koje definira korisnik. Aplikacija prati vrijeme u danu i temperaturu sobe, prate se korisnikove promjene te se na temelju tih podataka automatski određuju vrijednosti postavki pojedinih uređaja bez potrebe za korisničkim unosom.

Arhitektura aplikacije je zamišljena kao MVC model. Klijent pokreće klijentsku aplikaciju napisanu u Reactu te koristi REST API za komunikaciju s poslužiteljem odakle dobiva podatke za prikaz. Poslužiteljska aplikacija napisana je u Pythonu u radnom okviru Django te su njeni glavni zadaci autorizacija i autentifikacija korisnika, obrada zahtjeva, rad s bazom podataka i automatsko predviđanje stanja za pojedine uređaje.

Izvorni kod aplikacije dostupan je na GitHub repozitoriju na sljedećoj poveznici: `https://github.com/dominikpapes/smarthome`

5.1. Klijentska strana

5.1.1. Pokretanje klijentske aplikacije

Klijentska aplikacija implementirana je u Reactu te se kao frontend server koristi Vite. Kako bi se aplikacija pokrenula lokalno potrebno je klonirati repozitorij s dane poveznice, pozicionirati se u direktorij *SmartHome/frontend/smarthome* i instalirati potrebne pakete izvođenjem sljedeće naredbe u komandnoj liniji

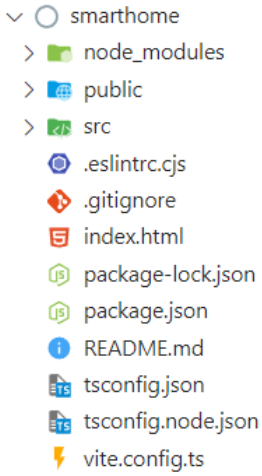
```
npm i
```

Pritom *npm* mora biti prethodno instaliran na računalu na kojem se pokreće naredba. Nakon uspješnog instaliranja svih potrebnih paketa potrebno je pokrenuti produkcijski frontend server koji će pokrenuti stranicu izvođenjem naredbe:

```
npm run dev
```

5.1.2. Struktura klijentske aplikacije

Na slici 5.1 prikazana je struktura korijenskog direktorija klijentske aplikacije. U nastavku će biti pojašnjene uloge najznačajnijih direktorija i datoteka.



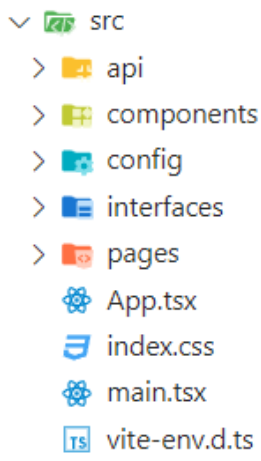
Slika 5.1: Struktura direktorija klijentske aplikacije

Direktorij *node_modules* sadrži pakete instalirane putem NPM-a te su ti paketi instalirani lokalno, odnosno pristupiti im se može jedino unutar projekata u direktoriju *smarthome*, za sve projekte van tog direktorija potrebno je ponovno instalirati dane pakete unutar tog projekta kako bi se mogli koristiti. Ovaj direktorij neće se tu nalaziti prilikom inicijalnog povlačenja s gita, već će se inicijalizirati instalacijom paketa koja je objašnjena u poglavlju 5.1.1.

Direktorij *public* sadrži javno dostupne datoteke, odnosno datoteke dostupne pregledniku koje ne moraju biti upakirane s ostatkom aplikacije, najčešće je riječ o slikama.

Direktorij *src* sadrži izvorni kod React aplikacije, te ima ulogu centralnog mjesta za razvoj aplikacije. Na slici 5.2 prikazana je struktura *src* direktorija. Direktorij *api* sadrži funkcije za pristup REST uslugama sa servera. Kako se React temelji na filozofiji gradnje sučelja na temelju ponovno uporabivih modularnih komponenata, u direktoriju *components* smještene su sve pojedinačne komponente. Direktorij *config* sadrži konfiguracijske detalje i inicijalna stanja za određene objekte. Direktorij *interfaces* sadrži TypeScript sučelja koja definiraju tip podataka koji se razmjenjuje između komponenata. Direktorij *pages* sadrži pojedinačne stranice preciznije, stranicu za prijavu, stvaranje korisničkog profila i stranicu za pregledavanje uređaja. Datoteka *App.tsx* glavna je komponenta aplikacije u koju se uključuju ostale komponente te unutar koje

se definiraju rute to jest URL-ovi[1] ostalih stranica. Datoteka *main.tsx* je ulazna točka aplikacije, u ovoj datoteci se poziva funkcija *ReactDOM.createRoot()* koja prikazuje glavnu komponentu aplikacije u DOM, također u ovoj datoteci je moguće uvesti *.css* datoteke koje će se koristiti u komponentama kako ne bi bilo potrebno uvažati te iste datoteke u svakoj nadolazećoj komponenti.



Slika 5.2: Struktura *src* direktorija

5.2. Poslužiteljska strana

5.2.1. Pokretanje poslužiteljske aplikacije

Poslužiteljska aplikacija implementirana je u Pythonu pomoću Django radnog okvira. Kako bi se aplikacija pokrenula lokalno potrebno je klonirati repozitorij s dane poveznice te prije svega pozicionirati se u direktorij *backend*. U direktoriju *backend* potrebno je stvoriti virtualno okruženje kako paketi koji će se se instalirati za korištenje s aplikacijom ne bi remetili uobičajen rad sistemskog Python interpretera, odnosno kako bi se spriječili konflikti. U direktoriju *backend* potrebno je pokrenuti sljedeću naredbu iz komandne linije[2]:

```
pyhton -m venv .venv
```

Ova naredba koristi alat *venv* iz standardne Python biblioteke koji će stvoriti virtualno okruženje u direktoriju *.venv*. Bitno je naglasiti kako se za direktorij *.venv* moglo odabarati bilo koje drugo proizvoljno ime, ali je *.venv* konvencionalno ime za taj direk-

[1] (engl. *Uniform Resource Locator*)

[2] Prikazane upute odnose se na operacijski sustav Windows

torij. Virtualno okruženje sada je stvoreno, ali ga je potrebno još i aktivirati kako bi se zaista moglo koristiti. Aktivacija virtualnog okruženja postiže se sljedećom naredbom:

```
.venv\Scripts\activate
```

S uključenim virtualnim okruženjem mogu se sigurno instalirati sve ovisnosti odnosno paketi pozicioniranje u direktorij *SmartHome/backend/smarthome* i izvršavanjem naredbe:

```
python -m pip install -r requirements.txt
```

Za konačno pokretanje poslužiteljske aplikacije potrebno je koristiti sljedeću naredbu:

```
python manage.py runserver
```

Ako prilikom pokretanja aplikacije dođe do problema vezanih uz migracije potrebno je izvršiti naredbu:

```
python manage.py migrate
```

Migracije su Djangov način upravljanja promjenama u modelima baze podataka i održavanja baze podataka sinkronizirane s definicijama modela u Django kodu.

5.2.2. Struktura poslužiteljske aplikacije

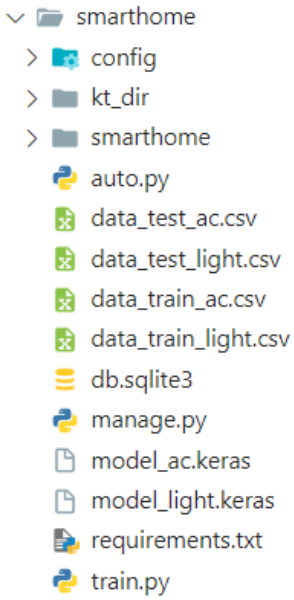
Na slici 5.3 prikazana je struktura korijenskog direktorija poslužiteljske aplikacije. U nastavku će biti objašnjeni pojedini elementi aplikacije.

Prije svega bitno je razlikovati između Django projekta i Django aplikacije. Django projekt je cjelokupna konfiguracija za određeno web sjedište, koja može uključivati jednu ili više aplikacija. Projekt sadrži postavke, konfiguracije URL-a i druge globalne konfiguracije. Projekt je u ovoj strukturi sadržan u direktoriju *config* te je njegova struktura prikazana na slici 5.4. Datoteka *db.sqlite3* sadrži podrazumijevanu bazu podataka. Skripta *manage.py* omogućava interakciju s projektom kao što su pokretanje servera ili migracija baze podataka.

Direktorij *__pycache__* kao i datoteka *__init__.py* neće biti posebno razmatrani jer je riječ o sustavnim datotekama programskog jezika Python. Datoteka *settings.py* je centralna konfiguracijska datoteka za Django projekt u kojoj se mogu odrediti postavke baze podataka, postavke aplikacije, middlewarea i ostali. Datoteke *asgi.py* i *wsgi.py* su ulazne točke za ASGI[3] odnosno za WSGI[4] kompatiblne servere. Datoteka *urls.py* koristi se za usmjeravanje URL-ova na odgovarajuće aplikacije unutar Django projekta.

[3](engl. *Asynchronous Server Gateway Interface*)

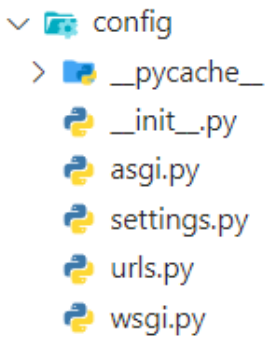
[4](engl. *Web Server Gateway Interface*)



Slika 5.3: Struktura poslužiteljske aplikacije

Na primjer sljedeći isječak koda iz *urls.py* će sve URL-ove koji počinju sa *smarthome/* preusmjeriti na aplikaciju *smarthome* te će se URL dalje parsirati koristeći njenu *urls.py* datoteku.

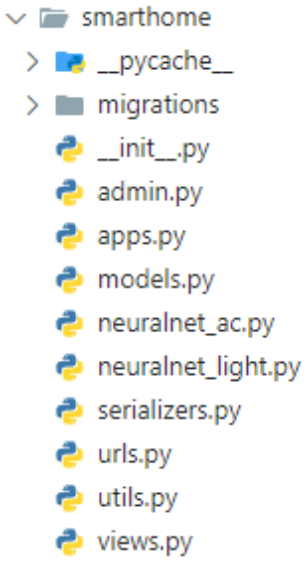
```
urlpatterns = [
    path("smarthome/", include("smarthome.urls")),
    path('admin/', admin.site.urls),
]
```



Slika 5.4: Struktura projekta *config*

S druge strane Django aplikacija je specifična funkcionalnost ili modul unutar projekta, kao što su blog, ankete, ili korisnički sustav. Aplikacije su dizajnirane tako da budu višekratno upotrebljive i neovisne o ostatku projekta. Na slici 5.5 prikazana je struktura aplikacije *smarthome* u direktoriju *backend/smarthome/smarthome*.

Datoteka *admin.py* pruža mogućnosti konfiguracije administratorskog sučelja za



Slika 5.5: Struktura aplikacije *smarthome*

modele definirane unutar aplikacije, administratorsko sučelje općenito nudi mogućnosti pregledavanja baze podataka i mijenjanja podataka u njoj. Datoteka *apps.py* omogućava definiranje određenih postavki koje su specifične za pojedinu aplikaciju, na primjer može se definirati dodatna funkcija koja će se pokretati prilikom svakog pokretanja aplikacije. U datoteku *models.py* pišu se definicije modela, odnosno strukture za bazu podataka. Modeli se pišu u obliku Python klasa koje nasljeđuju *models.Model* iz *django.db* te se zahvaljujući tome automatski mapiraju kao tablice u bazi podataka čime Django postiže funkcionalnost ORM-a. Datoteke *neuralnet_ac.py* i *neuralnet_light.py* sadrže definicije neuronskih mreža čija je implementacija opisana u poglavlju 5.2.5. Datoteka *serializers.py* dio je Django Rest radnog okvira te ova datoteka sadrži klase čija je funkcija mapiranje modela iz baze podataka u JSON objekte. Datoteka *urls.py* brine o usmjeravanju URL-ova specifičnih za aplikaciju. Datoteka *utils.py* sadrži pomoćne funkcije, na primjer funkciju za generiranje skupa za treniranje. Datoteka *views.py* sadrži poslovnu logiku, takozvane "poglede" u Djangovom MVT modelu to jest "kontrolere" u ekvivalentnom, ali poznatijem modelu MVC. Pogledi su zasluženi za obrađivanje HTTP zahtjeva i slanje odgovora.

5.2.3. Arhitektura baze podataka

Odabrana baza podataka za ovaj projekt je SQLite. SQLite je popularan, otvoren, samostalni relacijski sustav za upravljanje bazama podataka, DBMS[5] koji se koristi

[5](engl. *Database Management System*)

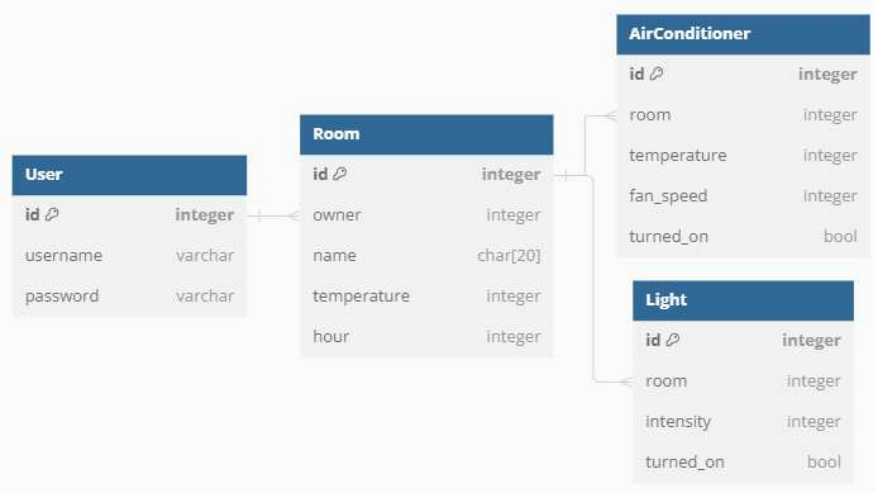
kao ugrađena relacijska SQL baza podataka. SQLite je iznimno lagan i jednostavan za korištenje, što ga čini idealnim izborom za mnoge aplikacije koje trebaju bazu podataka, ali ne zahtijevaju složene funkcionalnosti koje pružaju teži DBMS-ovi poput MySQL-a ili PostgreSQL-a. Njegova jednostavnost, prenosivost i minimalna potreba za konfiguracijom čine ga popularnim izborom za mobilne aplikacije, desktop softver i male web aplikacije [14].

Glavne značajke SQLitea su da je to "serverless" baza podataka, odnosno ne zahtijeva postavljanje i održavanje zasebnog procesa servera, podaci se pohranjuju izravno u datoteku na disku, u ovom slučaju riječ je o datoteci *db.sqlite3*.

U sklopu ovog rada SQLite pruža sasvim dovoljne performanse, ali valja napomenuti kako je SQLite ponajprije zamišljen kao sustav za upravljanje bazama podataka u razvoju te bi ga u slučaju puštanja ovog projekta u produkcijsko okruženje trebalo zamijeniti moćnijim DBMS-om kao što je na primjer PostgreSQL. Usprkos tome SQLite je odabran zbog odlične integracije s Django radnim okvirom koja pruža automatsku konfiguraciju SQLite kao odabrane baze podataka.

Na slici 5.6 prikazana je shema baze podataka. Baza je relativno jednostavna, sastoji se od 4 tablice *User*, *Room*, *AirConditioner* i *Light*. *User* sadrži tri polja: *id*, *username* i *password* u koje se spremaju id, korisničko ime i lozinka. *Room* sadrži 5 polja: *id*, *owner*, *name*, *temperature* i *hour* u koja se spremaju id sobe, id vlasnika sobe, ime sobe, zadnja izmjerena temperatura u sobi i sat zadnjeg mjerenja temperature. *AirConditioner* sadrži 5 polja: *id*, *room*, *temperature* i *fan_speed* i *turned_on* u koja se spremaju id klima uređaja, id sobe u kojoj se uređaj nalazi, temperatura na koju je postavljen klima uređaj, brzina kojom se okreće ventilator klima uređaja i stanje uređaja koje govori je li uređaj uključen. *Light* sadrži 4 polja: *id*, *room*, *intensity* i *turned_on* u koja se spremaju id svjetla, id sobe u kojoj se nalazi svjetlo, intenzitet na koje je svjetlo postavljeno i stanje svjetla koje je bool varijabla i govori o uključenosti svjetla. Jedan User može imati više soba. U jednoj sobi može se nalaziti više klima uređaja i svjetla.

Ako je iz nekog razloga potrebno upravljati bazom podataka direktno to se može postići pokretanjem poslužiteljske aplikacije i odlaskom na `http://127.0.0.1:8000/admin/`, podaci za prijavu administratora su korisničko ime "admin" s lozinkom "admin".



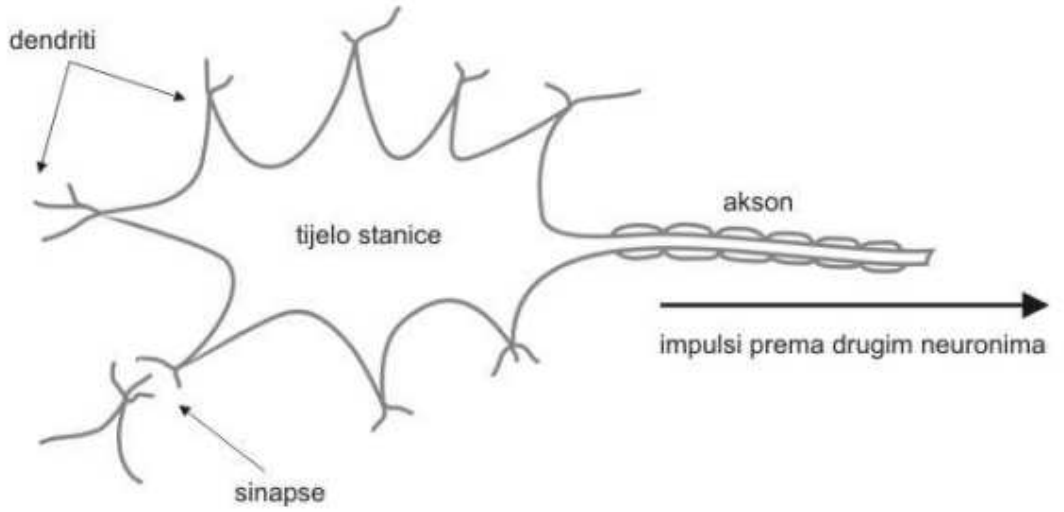
Slika 5.6: Shema baze podataka

5.2.4. Neuronske mreže

Ljudski mozak kao i računala konstantno procesuira informacije te je poznato da se mozak sastoji od velikog broja živčanih stanica (neurona) koji paralelno procesuiraju informacije. Ta činjenica navela je McCullocha i Pittsa još 1940. godine da opišu matematički model neuronske mreže u okviru teorije automata [4]. Međutim, zbog nedostatne tehnologije u to vrijeme prvi praktični primjeri neuronskih mreža javljaju se tek u 50-im godinama prošlog stoljeća, ali i tada su brzo pale u zaborav sve do 90-ih godina prošlog stoljeća te je njihov razvoj nastavljen sve do danas i još traje te one čine jedan od nezaobilaznih kocepata pri razvoju inteligentnih sustava.

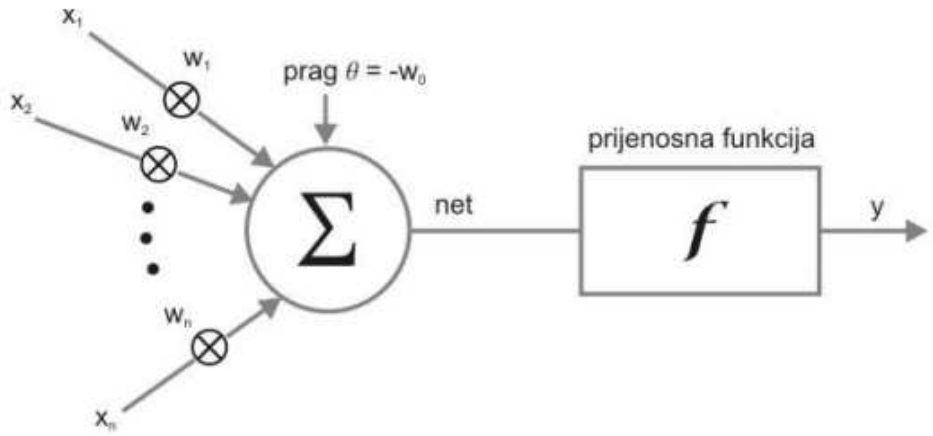
Biološki neuron sastoji se od tijela stanice, skupa dendrita (ogranaka), aksona (dugačke cijevčice koje prenose električke poruke) i niza završnih članaka. Na slici 5.7 prikazana je građa biološkog neurona. Tijelo stanice sadrži informaciju predstavljenu električnim potencijalom između okoline i unutrašnjosti. Neuroni su spojeni sinapsama koje se nalaze na dendritima te se preko njih primaju informacije koje se očituju u neuronu povećanjem ili smanjenjem potencijala tijela. Ako ukupni napon pređe određen prag pokreće se mehanizam slanja informacija pomoću neurotransmitera, kemikalije zadužene za slanje informacija, preko aksona šalje se novi impuls koji će inicirati ovakve reakcije s ostalim povezanim neuronima [4].

Na slici 5.8 prikazan je model neurona umjetne neuronske mreže. Model koristi slijedeću analogiju s biološkim neuronom: signali su opisani numeričkim iznosom i na ulazu u neuron množe se težinskim faktorom koji opisuje jakost sinapse; signali pomnoženi težinskim faktorima zatim se sumiraju analogno sumiranju potencijala u



Slika 5.7: Građa biološkog neurona [4]

tijelu stanice; ako je dobiveni iznos iznad definirana praga, neuron daje izlazni signal. Model može umjesto funkcije praga imati i takozvanu prijenosnu funkciju.



Slika 5.8: Neuron umjetne neuronske mreže [4]

Ulazne signale kojih ukupno ima n, označe se sa $x_1, x_2, \ldots, x_n$. Težine se označe sa $\omega_1, \omega_2, \ldots, \omega_n$.

Ulazni signali općenito su realni brojevi u intervalu $[-1, 1]$, $[0, 1]$ ili samo elementi iz $\{0, 1\}$, ako je riječ o Booleovom ulazu. Težinska suma net dana je s

$$\text{net} = \omega_1 x_1 + \omega_2 x_2 + \ldots + \omega_n x_n - \theta \tag{5.1}$$

ali se zbog kompaktnosti često dogovorno uzima da je vrijednost praga $\theta = -\omega_0$ te se dodaje ulazni signal x_0 s fiksiranom vrijednošću 1 te se jednostavnije može zapisati

$$\text{net} = \sum_{i=0}^{n} \omega_i x_i = \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \ldots + \omega_n x_n \tag{5.2}$$

dok je izlaz y rezultat prijenosne funkcije

$$y = f\left(\sum_{i=0}^{n} \omega_i x_i\right) = f(\text{net}) \tag{5.3}$$

Prijenosna funkcija može se definirati na mnogo načina, a najčešći odabir je sigmoidalna funkcija koja je derivabilna što je bitno prednost prilikom učenja neuronske mreže. Sigmoidalna funkcija definirana je kao:

$$f(\text{net}) = \frac{1}{1 + e^{-a \cdot \text{net}}} \tag{5.4}$$

Neuronske mreže se dakle sastoje od velikog broja povezanih neurona. Neuroni su poredani u slojeve. Prvi ulazni sloj sastoji se od ulaznih podataka, nakon čega slijede skriveni slojevi koji obrađuju podatke te zadnji sloj predstavlja izlaz odnosno predikciju koju je mreža izračunala na temelju ulaznih podataka. Neuronsku mrežu prije korištenja potrebno je trenirati odnosno naučiti na testnom skupu podataka. Testni skup podataka još se dodatno podijeli na skup za učenje i skup za testiranje kako bi se mogla provjeriti točnost predikcija neuronske mreže. Učenje se svodi na podešavanje težina u mreži koja utječu na konačni rezultat.

5.2.5. Implementaicija neuronske mreže u sklopu aplikacije *SmartHome*

U sklopu ovog rada korištena je neuronska mreže implementirana u Kerasu s TensorFlow backendom. Keras pruža visoko razinski API za modeliranje neuronskih mreža koji je odličan za početnike u polju neuronskih mreža.

Keras nudi dva glavna modela za pisanje neuronskim mreža: sekvencijalni i funkcionalni API. Sekvencijalni model omogućava jednostavno stvaranje unaprijedne mreže dok funkcionalni API nudi više mogućnosti prilagodbe mreže korisnikovim potrebama. U ovom radu korišten je funkcionalni model kako bi određeni parametri mreže mogli bolje prilagoditi potrebama rada.

Ulaz mreže u slučaju klima uređaja sastoji se od sljedećih značajki: id uređaja, koji je indirektno povezan s vlasnikom, trenutna temperatura sobe i trenutni sat u danu. Izlaz neuronske mreže sastoji se predviđenog stanja uređaja (uključen ili isključen), predviđene temperature na koju bi uređaj trebao biti podešen i predviđene brzine ventilatora.

U slučaju neuronske mreže za predviđanje postavki svjetla ulaz se sastoji od značajki idja svjetla i doba dana, a izlazi se sastoje od predviđenog intenziteta svjetla.

Funkcionalni model korišten je zbog činjenice da se na ulaz neuronske mreže dovodi ID uređaja koji, iako je numerička varijabla, nema jednako značenje kao ostale vrijednosti, već ima kategoričko značenje. Te se u tu svrhu koristi *Embedding* sloj. Osim toga ostali ulazi i izlazi kodirani su *One-Hot Encoding* tehnikom što znači da se mogući ulazi, odnosno izlazi kodiraju na način da se podijele na moguće klase te se klasa kojoj određeni ulaz odnosno izlaz pripada označi s 1, a ostale s 0.

6. Studijski slučaj: korištenje aplikacije s podacima pretpostavljenog korisnika

Aplikacija je zamišljena u klijent-poslužitelj arhitekturi. Kompatibilni uređaji mogli bi se spojiti s aplikacijom te bi korisnik na svom računalu ili pametnom telefonu mogao upravljati uređajima, a sav rad s neuronskim mrežama, spremanjem u bazu podataka i slično obavlja se na udaljenom poslužitelju.

Korisniku se pri svakom pokretanju aplikacije javlja ekran s opcijom za prijavljivanje u aplikaciju i opcijom za stvaranje novog računa. Na slici 6.1 prikazano je sučelje za prijavu.

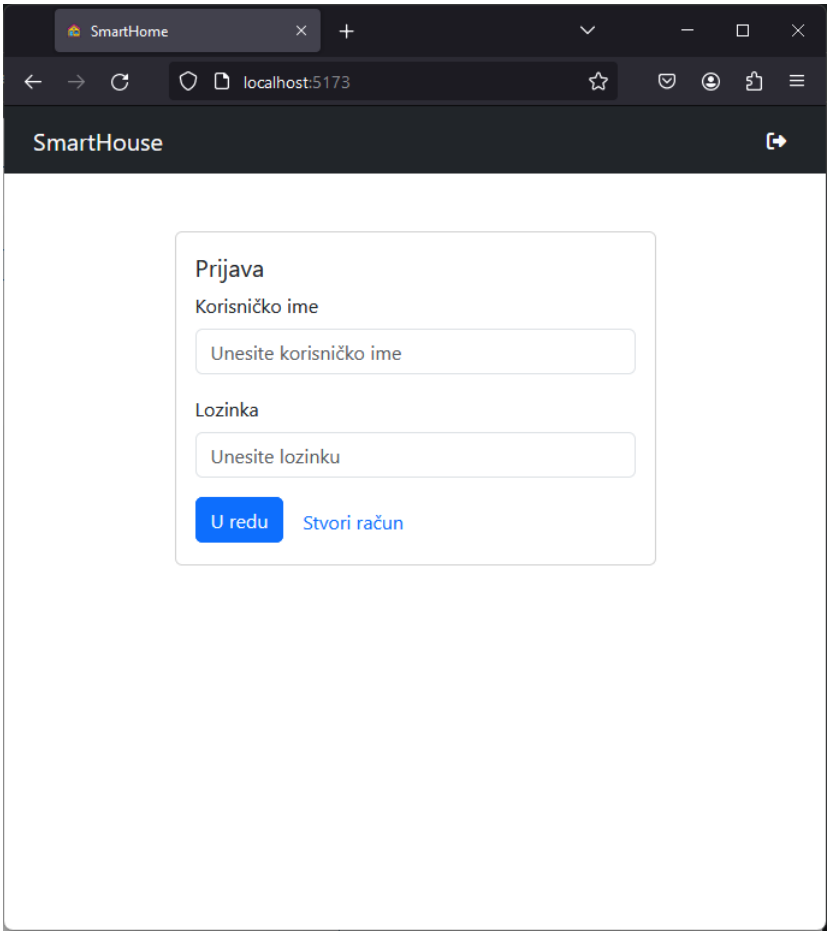
Korisnik će kod prvog pristupa aplikaciji odabrati opciju "Stvori račun" što će ga odvesti na sučelje za stvaranje računa. Potrebno je odabrati korisničko ime i lozinku. Korisnik kreira svoj račun na primjer s korisničkim imenom "Korisnik" i lozinkom "Lozinka123". Na slici 6.2 prikazano je sučelje za stvaranje novog korisnika.

Korisnik se prijavljuje i prikazuje mu se ekran s mogućnošću stvaranja nove sobe ili odjave iz sustava. Korisnik može stvoriti novu sobu klikom na plavi plus. Na slici 6.3 prikazano je sučelje za dodavanje nove sobe u kojem se mogu dodati novi uređaji i podesiti njihove početne vrijednosti.

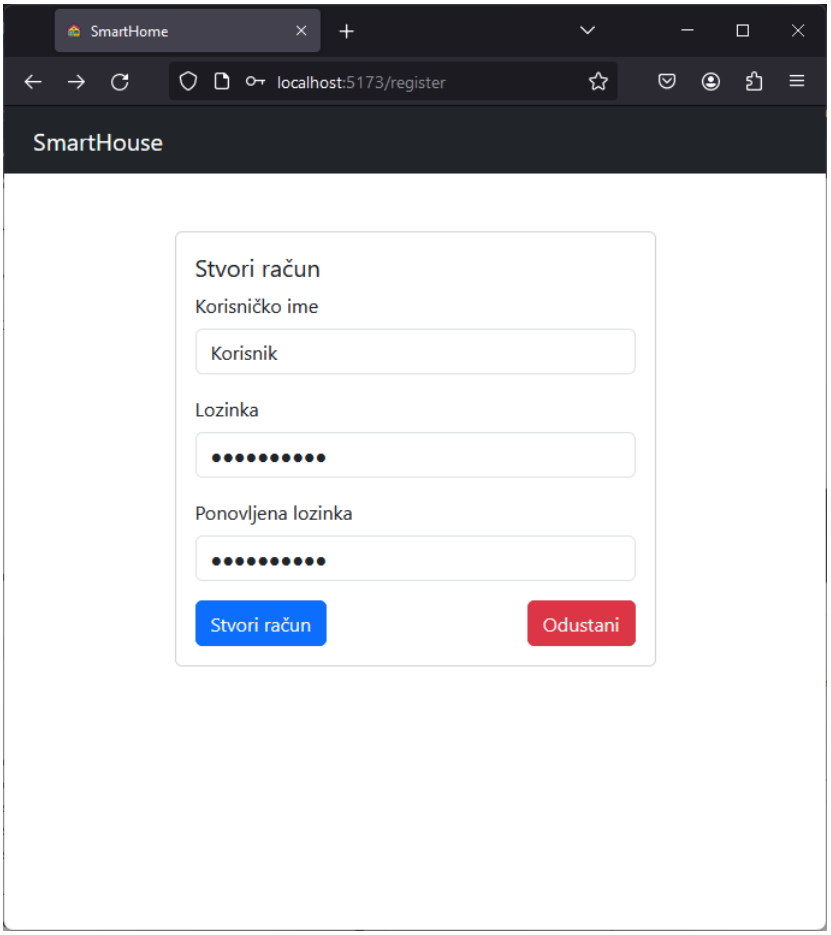
Pritiskom na gumb "OK" dodaju se novi uređaji i njihovo stanje može se provjeriti u početnom ekranu, prikazano na slici 6.4.

Korisnik ima opciju brisanja sobe klikom na crveni X prije čega mu se pojavljuje prozor za potvrdu, također korisnik može mijenjati parametre uređaja klikom na ikonu olovke. Te je u istom sučelju moguće dodati novi uređaj u sobi ili izbrisati neki od postojećih. Sučelje za mijenjanje parametara prikazano je na slici 6.5.

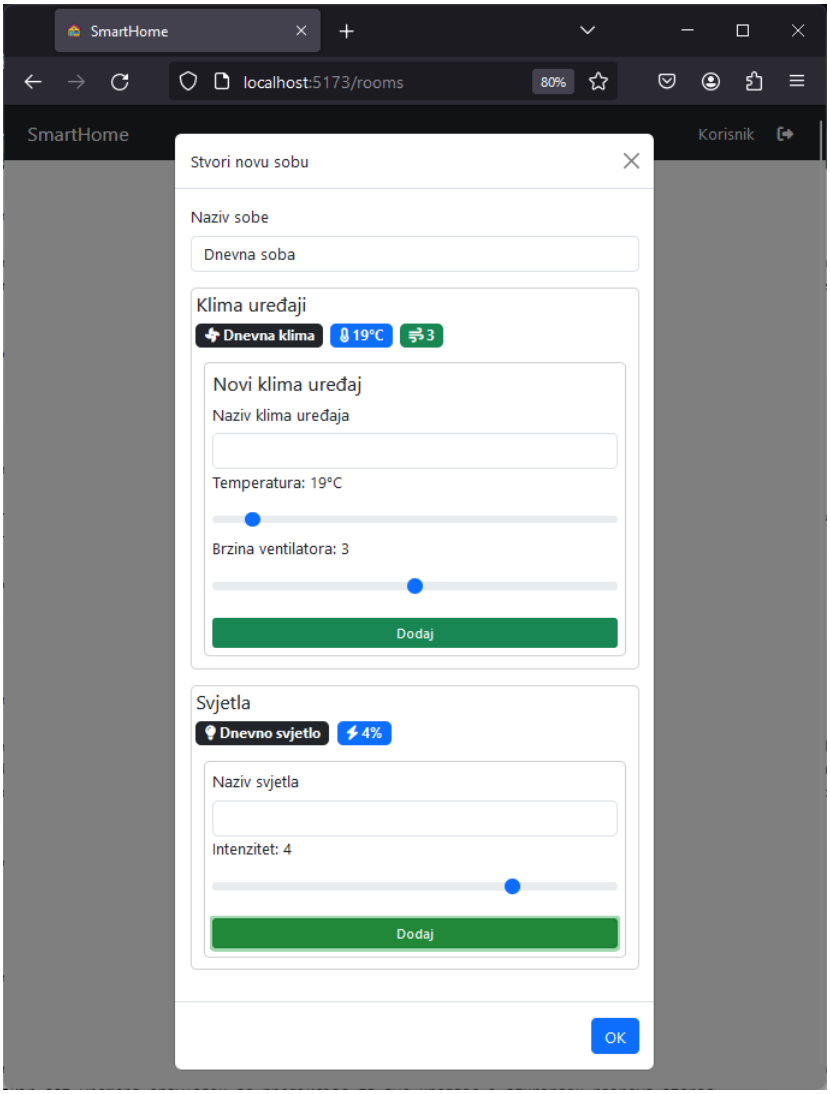
Korisnikove promjene parametara bilježe se te se model neuronske trenira sukladno s novim promjenama. Svakih sat vremena obavljaju se predikcije za sve uređaje i ažuriraju njihova stanja.



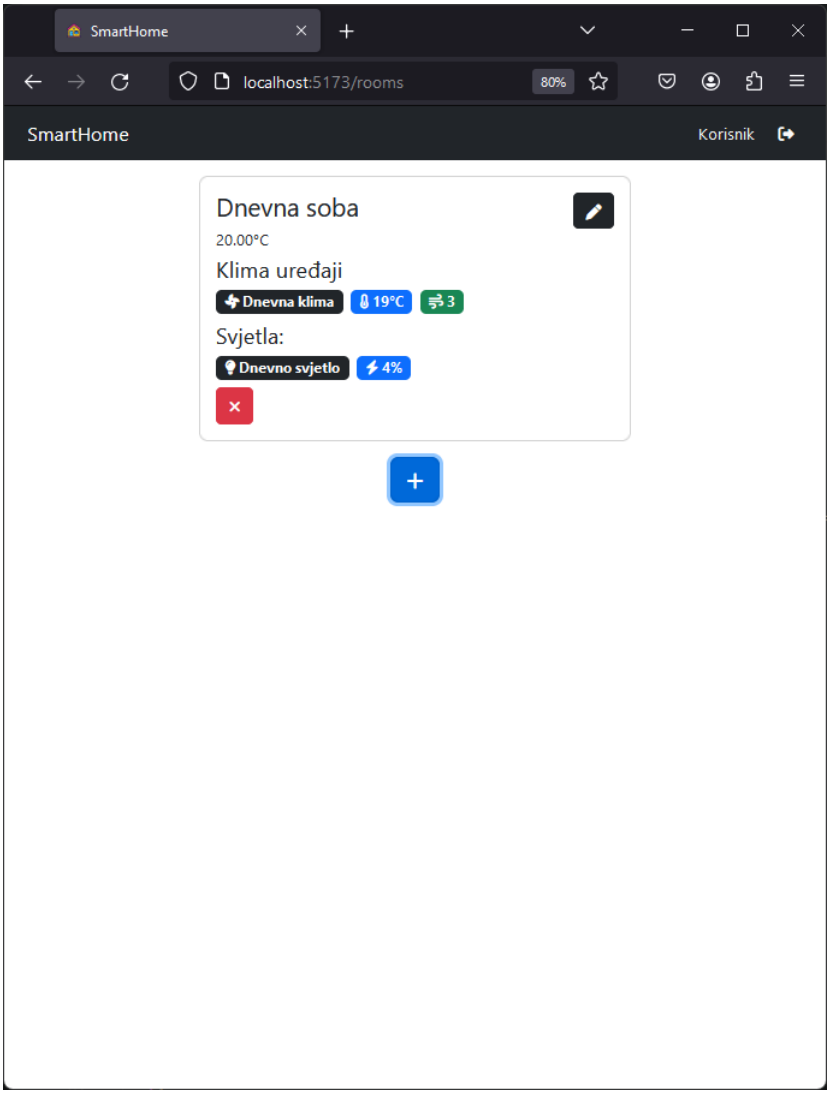
Slika 6.1: Sučelje za prijavu



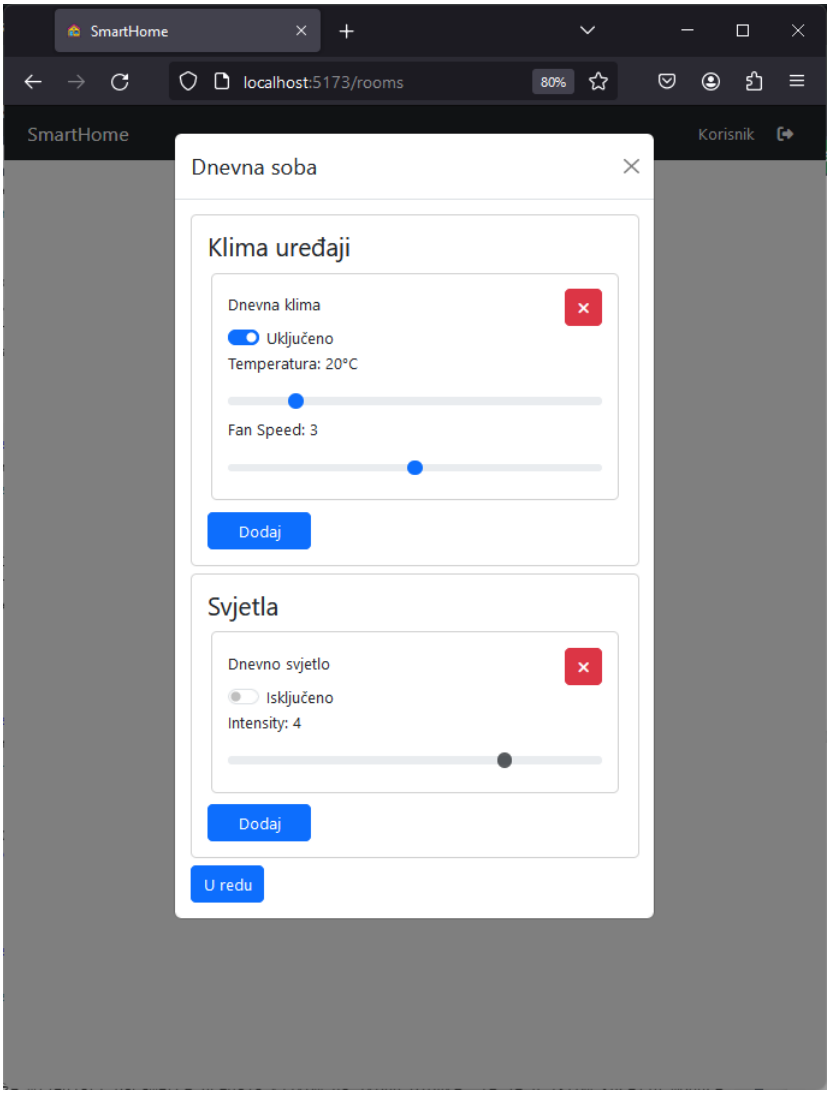
Slika 6.2: Stvaranje korisničkog računa



Slika 6.3: Stvaranje nove sobe



Slika 6.4: Prikaz uređaja



Slika 6.5: Sučelje za mijenjanje parametara uređaja

U sklopu ovog rada implementirane su i skripte *auto.py* i *train.py* koje simuliraju rad aplikacije. Nakon stvaranja određenog broja korisnika i uređaja za te korisnike prikazanim postupkom potrebno je aktivirati skriptu *train.py* koja će inicijalizirati model neuronske mreže i započeti njeno treniranje, ovaj postupak potrebno je učiniti samo prilikom prvog pokretanja te je skriptu potrebno pokrenuti unutar zasebne komandne linije. Skripta će najprije simulirati skup podataka za treniranje mreže te će potom generirati prikladni model mreže.

Kako bi se simulirao rad same aplikacije uz ažuriranje stanja svakih sat vremena potrebno je pokrenuti skriptu *auto.py*. Skripta prima 1 argument *"–single"* te ako je argument naveden skripta će ažurirati stanja samo jednom, bez toga argumenta skripta radi kontinuirano gdje je jedan sat simuliran kao 10 sekundi.

7. Zaključak

U ovom radu istraženi su osnovni koncepti povezivanja uređaja pomoću inteligentnih usluga u prostoru Interneta stvari. Definirani se pojmovi objekta, senzora i aktuatora, kao i koncept svjesnosti konteksta koji je ključan u kognitivnom aspektu usluga Interneta stvari. Pojam svjesnosti konteksta odnosi se na sposobnost sustava da prikuplja, interpretira i koristi podatke o okolnostima u kojima se uređaji nalaze, kako bi automatski donosio odluke i prilagođavao svoje ponašanje.

Predstavljena su neka postojeća rješenja iz područja rada koja uključuju Apple Siri, Amazon Alexu i SmartThings. Ova rješenja demonstriraju kako se umjetna inteligencija i kognitivne usluge mogu integrirati u svakodnevne uređaje, omogućujući im interakciju s korisnicima na prirodniji i intuitivniji način. Pokazane su mogućnosti pojedine platforme s korisničke perspektive kao i s perspektive inženjera sklopovske i programske podrške opisom pripadajućih sučelja za razvoj i prilagođavanje platformi.

Opisana je implementacija web aplikacije SmartHome koja korisnicima pruža mogućnosti upravljanja spojenim klima uređajima i svjetlima. Upravljanje uključuje mogućnosti gašenja i paljenja određenog uređaja, mijenjanja vrijednosti temperature i brzine ventilatora za klima uređaje te promjenu intenziteta svjetla za svjetlosne uređaje. Aplikacija također nudi kognitivnu uslugu koja prati korisničke unose za pojedine uređaje koji pripadaju pojedinim korisnicima te na temelju tih unosa i određenog konteksta automatski predviđa stanja pojedinih uređaja. Ova funkcionalnost omogućuje personalizirano iskustvo za svakog korisnika povećavajući lakoću korištenja, udobnost i ergonomiju.

Opisani su alati korišteni u implementaciji navedenih funkcionalnosti: React, Typescript i Bootstrap za korisničku stranu te Django na poslužiteljskoj strani. React omogućuje izradu dinamičkih korisničkih sučelja, Typescript dodaje statičku tipizaciju kako bi se smanjile pogreške tijekom razvoja, a Bootstrap pruža unaprijed definirane stilove za bržu i lakšu izradu responzivnog dizajna. Django, kao robustan radni okvir za web, omogućuje brzo i sigurno upravljanje podacima na poslužiteljskoj strani.

Prikazani su osnovni koncepti rada neuronskih mreža te je opisana implementacija

neuronske mreže u Kerasu, što omogućuje primjenu naprednih tehnika strojnog učenja za predviđanje stanja uređaja. Neuronske mreže koriste se za analizu povijesnih podataka korisničkih unosa i kontekstualnih informacija, te na temelju toga donose predikcije koje unapređuju funkcionalnost aplikacije.

Dan je studijski slučaj korištenja aplikacije s podacima pretpostavljenog korisnika, pri čemu su pokazani svi postupci korištenja aplikacije i prikaz djelovanja predviđanja neuronske mreže. Kroz ovaj slučaj detaljno su prikazani koraci od inicijalne konfiguracije aplikacije, stvaranja novog korisnika, novih soba i dodavanja uređaj sobe do svakodnevne upotrebe i automatskih prilagodbi koje aplikacija pruža.

Moguća unapređenja rada uključuju otklanjanje rada kojeg obavlja poslužitelj prilikom svakog treniranja mreže na način da se neuronske mreže treniraju u takozvanim "klasterim" odnosno trebalo bi implementirati raspodijeljeno učenje na način da se dio treniranja mreže obavlja na klijentima. Također trebalo bi pružiti podršku za više vrsta uređaja i implementirati mobilnu aplikaciju za Android i iOS operacijske sustave.

LITERATURA

[1] Thread Group - What is Thread. URL \url{https://www.threadgroup.org/What-is-Thread/Overview}. Zadnji pristup: 2024-06-14.

[2] Apple Inc. Siri overview. https://developer.apple.com/siri/, 2023. Zadnji pristup: 2024-06-14.

[3] Kevin Ashton et al. That 'internet of things' thing. *RFID journal*, 22(7):97–114, 2009.

[4] Bojana Dalbelo Bašić, Marko Čupić, i Jan Šnajder. Umjetne neuronske mreže. *Zagreb: Fakultet elektrotehnike i računarstva*, stranice 7–15, 2008.

[5] Harald Bauer, Sebastian Schink, i Florian Thalmayr. The internet of things: Sizing up the opportunity. *McKinsey Company*, 2020. URL https://www.mckinsey.com/industries/semiconductors/our-insights/the-internet-of-things-sizing-up-the-opportunity. Zadnji pristup: 2024-06-14.

[6] Bootstrap Development Team. Bootstrap documentation. https://getbootstrap.com/, 2023. Zadnji pristup: June 6, 2024.

[7] Michael Aaron Dennis. Defense advanced research projects agency | united states government. *Encyclopædia Britannica*, December 23 2022. URL https://www.britannica.com/topic/Defense-Advanced-Research-Projects-Agency. Zadnji pristup: 2024-06-14.

[8] Django REST Framework Development Team. Django rest framework. https://www.django-rest-framework.org/, 2023. Zadnji pristup: 2024-06-14.

[9] Django Software Foundation. Django: The web framework for perfectionists with deadlines. https://www.djangoproject.com/, 2023. Zadnji pristup: June 6, 2024.

[10] Samsung Electronics. Smartthings, 2024. URL https://www.samsung.com/us/smartthings/. Zadnji pristup: 2024-06-05.

[11] Cristian González García, Daniel Meana-Llorián, Juan Manuel Cueva Lovelle, et al. A review about smart objects, sensors, and actuators. *International Journal of Interactive Multimedia & Artificial Intelligence*, 4(3), 2017.

[12] Vaishnavi S Gunge i Pratibha S Yalagi. Smart home automation: a literature review. *International Journal of Computer Applications*, 975(8887-8891), 2016.

[13] HedgeThink. Home automation – a brief history and why it's so important. *Hedge Think*, 2023. URL https://www.hedgethink.com/home-automation-brief-history-important/. Zadnji pristup: 2024-06-14.

[14] D. Richard Hipp. Sqlite, 2024. URL https://sqlite.org/. Zadnji pristup: 2024-06-11.

[15] Adrian Holovaty i Jacob Kaplan-Moss. *The definitive guide to Django: Web development done right*. Apress, 2009.

[16] M. B. Hoy. Alexa, siri, cortana, and more: An introduction to voice assistants. *Medical Reference Services Quarterly*, 37(1):81–88, 2018. doi: 10.1080/02763869.2018.1404391. URL https://doi.org/10.1080/02763869.2018.1404391.

[17] Ljiljana Kaliterna-Lipovčan, Spomenka Tomek-Roksandić, Goran Perko, Diana Mihok, Hrvoje Radašević, Ana Puljak, i Stjepan Turek. Gerontehnologija u europi iu hrvatskoj. *Medicus*, 14(2_Gerijatrija):301–304, 2005.

[18] Achal S Kaundinya, Nikhil SP Atreyas, Smrithi Srinivas, Vidya Kehav, i NMR Kumar. Voice enabled home automation using amazon echo. *Int. Res. J. Eng. Technol*, 4:682–684, 2017.

[19] Ignac Lovrek. Svjesnost konteksta u mreži pokretnih programskih agenata. *Rad Hrvatske akademije znanosti i umjetnosti. Tehničke znanosti*, (513= 15):7–28, 2012.

[20] Somayya Madakam, Ramya Ramaswamy, i Siddharth Tripathi. Internet of things (iot): A literature review. *Journal of Computer and Communications*, 3(5):164–173, 2015.

[21] Mozilla Developer Network. Mdn web docs. `https://developer.mozilla.org/en-US/`, 2023. Zadnji pristup: 2024-06-14.

[22] Anas M Mzahm, Mohd Sharifuddin Ahmad, i Alicia YC Tang. Enhancing the internet of things (iot) via the concept of agent of things (aot). *Journal of Network and Innovative Computing*, 2(2014):101–110, 2014.

[23] Nathalia Nascimento, Paulo Alencar, Donald Cowan, i Carlos Lucena. A reference model for iot embodied agents controlled by neural networks. U *2020 IEEE international conference on big data (Big Data)*, stranice 3500–3505. IEEE, 2020.

[24] Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, et al. Kerastuner. `https://github.com/keras-team/keras-tuner`, 2019. Zadnji pristup: 2024-06-14.

[25] Bo Pang, Erik Nijkamp, i Ying Nian Wu. Deep learning with tensorflow: A review. *Journal of Educational and Behavioral Statistics*, 45(2):227–248, 2020.

[26] React Bootstrap Development Team. React bootstrap documentation. `https://react-bootstrap.netlify.app/`, 2023. Zadnji pristup: June 6, 2024.

[27] React Development Team. React documentation. `https://react.dev/`, 2023. Zadnji pristup: 2024-06-14.

[28] Bill Schilit i Marvin Theimer. Disseminating active map information to mobile hosts. U *Proceedings of the 6th Annual ACM Symposium on Mobile Computing and Networking (MOBICOM '94)*, stranice 85–91, 1994. doi: 10.1145/190314.190324.

[29] John Sculley i John A. Byrne. *Odyssey: Pepsi to Apple ... A Journey of Adventure, Ideas, and the Future*. Harper & Row, New York, 1st izdanju, 1987. ISBN 0-06-015780-1.

[30] Deepti Sehrawat i Nasib Singh Gill. Smart sensors: Analysis of different types of iot sensors. U *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, stranice 523–528. IEEE, 2019.

[31] Amazon Developer Services. Alexa skills kit. https://developer.amazon.com/en-US/alexa/alexa-skills-kit, 2024. Zadnji pristup: 2024-06-14.

[32] Aaron Tilley. Samsung acquires smartthings, a fast-growing home automation startup. *Forbes*, 2014. URL https://www.forbes.com/sites/aarontilley/2014/08/14/samsung-smartthings-acquisition-2/. Zadnji pristup: 2024-06-14.

[33] TypeScript Development Team. Typescript: Javascript with syntax for types. https://www.typescriptlang.org/, 2023. Zadnji pristup: 2024-06-14.

[34] Vite Development Team. Vite: Next generation frontend tooling. https://vitejs.dev/, 2023. Zadnji pristup: 2024-06-14.

[35] Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3): 94–104, 1991. doi: 10.1038/scientificamerican0991-94.

[36] Michael Wooldridge. *An introduction to multiagent systems*. John wiley & sons, 2009.

[37] Ryan R. Wu. Alexa, who are you — a brief history of amazon's voice assistant and beyond. *Catalium*, 2018. URL https://catalium.net/2018/alexa-who-are-you. Zadnji pristup: 2024-06-14.

[38] Wondimu Zegeye, Ahamed Jemal, i Kevin Kornegay. Connected smart home over matter protocol. U *2023 IEEE International Conference on Consumer Electronics (ICCE)*, stranice 1–7. IEEE, 2023.

Razvoj koginitivne usluge u prostoru Interneta stvari

Sažetak

U ovom radu istražuju se osnovni koncepti povezivanja uređaja pomoću inteligentnih usluga u prostoru Interneta stvari. Definiraju se pojmovi objekta, senzora i aktuatora kao i koncept svjesnosti konteksta koji je ključan u kognitivnom aspektu usluga Interneta stvari. Predstavljena su neka postojeća rješenja iz područja rada koja uključuju Apple Siri, Amazon Alexu i SmartThings. Opisana je implementacija web aplikacije SmartHome koja korisnicima pruža mogućnosti upravljanja spojenim klima uređajima i svjetlima pri čemu upravljanje podrazumijeva mogućnosti gašenja i paljenja određenog uređaja te mijenjanja vrijednosti temperature i brzine ventilatora, za klima uređaje, te promjena intenziteta svjetla, za svjetlosne uređaje. Aplikacija također nudi kognitivnu uslugu koja prati korisničke unose za pojedine uređaje koji pripadaju pojedinim korisnicima te na temelju tih unosa i određenog konteksta automatski predviđa stanja pojedinih uređaja. Opisani su alati korišteni u implementaciji navedenih funkcionalnosti: React, Typescript i Bootstrap za korisničku stranu te Django na poslužiteljskoj strani. Prikazani su osnovni koncepti rada neuronskih mreža te je opisana implementacija neuronske mreže u Kerasu. Dan je studijski slučaj korištenja aplikacije s podacima pretpostavljenog korisnika pri čemu su pokazani svi postupci korištenja aplikacije i prikaz djelovanja predviđanja neuronske mreže. Rad je komentiran u cjelini te su iznesene ideje za buduća poboljšanja rada.

Ključne riječi: Internet stvari, kognitivne usluge, web aplikacija, virtualni asistenti, React, Django, Tensorflow

Development of cognitive services within the Internet of Things

Abstract

This paper explores the basic concepts of connecting devices using intelligent services in the Internet of Things (IoT) space. It defines the concepts of objects, sensors, and actuators, as well as the concept of context awareness, which is crucial in the cognitive aspect of IoT services. Some existing solutions in the field are presented, including Apple Siri, Amazon Alexa, and SmartThings. The implementation of the SmartHome web application is described, which provides users with the ability to control connected air conditioners and lights. Control involves turning specific devices on and off, as well as adjusting temperature and fan speed for air conditioners, and changing light intensity for light devices. The application also offers a cognitive service that monitors user inputs for individual devices belonging to specific users and it automatically predicts device states based on these inputs and specific contexts. The tools used in implementing these functionalities are outlined: React, Typescript, and Bootstrap for the client-side, and Django for the server-side. Basic concepts of neural networks are presented, and the implementation of a neural network in Keras is described. A case study depticting everyday usage of the application with data from a hypothetical user is provided, demonstrating all the steps including configuring the application and showcasing the operation of the neural network predictions. The paper is commented on as a whole, and ideas for future improvements are presented.

Keywords: Internet of Things, cognitive services, web application, virtual assistants, Reac, Django, Tensorflow