VIŠJA STROKOVNA ŠOLA ACADEMIA

MARIBOR

RAZVOJ IN IMPLEMENTACIJA VEČPLATFORMSKEGA ASISTENTA Z IOT PODPORO IN VARNIM HIBRIDNIM SPOMINOM

Kandidat: Dominik Bajkovec

Vrsta študija: študent izrednega študija

Študijski program: Informatika

Mentor predavatelj: mag. Dušan Brglez

Mentor v podjetju: David Novak, inž. mep.

Lektor: Ana Kovač, prof. slov. in angl.

Maribor, 2026

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Podpisani Dominik Bajkovec, sem avtor diplomskega dela z naslovom Razvoj in implementacija večplatformskega asistenta z IoT podporo in varnim hibridnim spominom, ki sem ga napisal pod mentorstvom mag. Dušana Brgleza.

S svojim podpisom zagotavljam, da:

- je predloženo delo izključno rezultat mojega dela,
- sem poskrbel, da so dela in mnenja drugih avtorjev, ki jih uporabljam v predloženi nalogi, navedena oz. citirana skladno s pravili Višje strokovne šole Academia Maribor,
- se zavedam, da je plagiatorstvo – predstavljanje tujih del oz. misli, kot moje lastne kaznivo po Zakonu o avtorski in sorodnih pravicah (Uradni list RS, št. 16/07 – uradno prečiščeno besedilo, 68/08, 110/13, 56/15 in 63/16 – ZKUASP); prekršek pa podleže tudi ukrepom Višje strokovne šole Academia Maribor skladno z njenimi pravili,
- skladno z 32.a členom ZASP dovoljujem Višji strokovni šoli Academia Maribor objavo diplomskega dela na spletnem portalu šole.

Šmarje pri Jelšah, maj 2026 Podpis študenta:

ZAHVALA

Zahvaljujem se mentorju mag. Dušanu Brglezu za usmeritve in podporo pri izdelavi diplomskega dela.

Zahvaljujem se tudi mentorju v podjetju Davidu Novaku, inž. mep., za podporo, praktično znanje in izkušnje, ki so prispevali k mojemu strokovnemu razvoju ter izvedbi diplomskega dela.

Posebna zahvala velja moji družini in bližnjim za podporo, potrpežljivost in spodbudo med študijem ter pripravo diplomskega dela.

POVZETEK

Diplomsko delo obravnava razvoj in implementacijo večplatformskega asistenta z IoT podporo in varnim hibridnim spominom. Namen dela je bil zasnovati in preveriti sistem, ki ni omejen na en sam uporabniški vmesnik, temveč združuje spletni dostop, Discord integracijo, povezavo z robnimi IoT napravami, večslojni spomin ter ločene delovne tokove za jezikovno in govorno obdelavo. Pri tem je bil poudarek na modularni in porazdeljeni zasnovi, ki omogoča boljšo razširljivost, preglednejše vzdrževanje ter postopno nadgrajevanje posameznih funkcionalnih sklopov.

Teoretični del diplomskega dela obravnava sistemski inženiring, porazdeljene sisteme, podatkovne baze, hibridne pristope k spominu, IoT integracijo ter kibernetsko varnost v večplatformskih okoljih. Na osnovi tega je bila zasnovana arhitektura sistema, ki ločuje integracijske plasti, osrednje procesne module, podatkovno plast in podporne storitve. Implementacijski del prikazuje konkretno izvedbo spletnega vmesnika, Discord integracije, MQTT povezave z ESP32 napravami, hibridnega spomina na osnovi RAM konteksta, MySQL in PostgreSQL/pgvector, razdelitve delovnih tokov med LLM, STT in TTS ter mehanizmov varnosti in zanesljivosti.

Rezultati testiranja so pokazali, da sistem v trenutni implementaciji deluje kot uporabna večplatformska rešitev. Potrjena je bila stabilna besedilna uporaba prek spletnega in Discord vmesnika, delujoča IoT povezljivost, večslojni priklic konteksta ter operativno upravljanje izbranih naprav in storitev. Preverjena je bila tudi ločena govorna pot, zasnovana na STT, TTS in glasovnem agentu, ki potrjuje, da sistem ni omejen le na besedilno interakcijo, temveč podpira tudi prostoročno uporabo.

Varnostni mehanizmi, kot so prijava, 2FA, zaščitene poti in omejevanje privilegiranih ukazov, so se izkazali kot uporabni v lokalnem okolju, medtem ko nekateri deli, kot sta širša uporaba TLS in avtomatizirano varnostno kopiranje operativnih podatkov, ostajajo področje nadaljnjega razvoja. Diplomsko delo tako potrjuje, da modularna in porazdeljena zasnova predstavlja smiselno osnovo za nadaljnji razvoj večplatformskega asistenta.

Ključne besede: *večplatformski asistent, IoT, hibridni spomin, porazdeljeni sistemi, kibernetska varnost.*

ABSTRACT

Development and Implementation of a Multiplatform Assistant with IoT Support and Secure Hybrid Memory

This diploma thesis addresses the development and implementation of a multiplatform assistant with IoT support and secure hybrid memory. The aim of the thesis was to design and evaluate a system that is not limited to a single user interface, but instead combines web access, Discord integration, connectivity with edge IoT devices, multi-layer memory, and separate workflows for language and speech processing. The main focus was placed on a modular and distributed design that enables better scalability, clearer maintenance, and the gradual extension of individual functional components.

The theoretical part of the diploma thesis covers systems engineering, distributed systems, databases, hybrid memory approaches, IoT integration, and cybersecurity in multiplatform environments. On this basis, a system architecture was designed that separates integration layers, core processing modules, the data layer, and supporting services. The implementation part presents the concrete implementation of the web interface, Discord integration, MQTT connectivity with ESP32 devices, hybrid memory based on RAM context, MySQL and PostgreSQL/pgvector, the separation of workflows between LLM, STT, and TTS, as well as security and reliability mechanisms.

The testing results showed that the system, in its current implementation, functions as a practical multiplatform solution. Stable text-based use through the web and Discord interfaces, functioning IoT connectivity, multi-layer context retrieval, and operational management of selected devices and services were confirmed. A separate voice pathway based on STT, TTS, and a voice agent was also verified, confirming that the system is not limited solely to text interaction, but also supports hands-free use. Security mechanisms such as login, 2FA, protected routes, and restrictions on privileged commands proved useful in a local environment, while some elements, such as broader use of TLS and automated backups of operational data, remain areas for further development. The diploma thesis therefore confirms that a modular and distributed design provides a meaningful foundation for the further development of a multiplatform assistant.

Keywords: *multiplatform assistant, IoT, hybrid memory, distributed systems, cybersecurity*

KAZALO VSEBINE

1 UVOD 9

1.1 Opis področja in opredelitev problema 9

1.2 Namen, cilji in osnovne trditve 10

1.3 Predpostavke in omejitve 11

1.4 Uporabljene raziskovalne metode 11

2 TEORETIČNA IZHODIŠČA IN PREGLED PODROČJA 13

2.1 Sistemski inženiring in porazdeljeni sistemi 13

2.2 Podatkovne baze in hibridni pristopi k spominu 14

2.3 IoT integracija in zanesljivost sistemov 15

2.4 Kibernetska varnost v večplatformskih okoljih 16

2.5 Pregled obstoječih rešitev 17

3 NAČRTOVANJE IN ARHITEKTURA SISTEMA 18

3.1 Modularna arhitektura asistenta 18

3.2 Integracija več vmesnikov (web/Discord) 20

3.3 IoT integracija (ESP32 + PIR) 21

3.4 Hibridni spomin (RAM buffer, MySQL, vektorska baza) 22

3.5 Razdelitev delovnih tokov (LLM, TTS, STT) in porazdelitev bremen 23

3.6 Nadzor in upravljanje naprav ter sistemskih storitev 24

3.7 Varnostni mehanizmi (TLS, avtentikacija, varnostne kopije, Caddy) 25

3.8 Mehanizmi zanesljivosti (Uptime Kuma, samodejni ponovni zagoni, varnostne kopije) 26

4 IMPLEMENTACIJA IN IZVEDBA 27

4.1 Razvojno okolje in uporabljene tehnologije 27

4.2 Implementacija spletnega vmesnika in Discord integracije 29

4.3 Implementacija IoT podpore in povezave s sistemom 30

4.4 Implementacija hibridnega spomina 32

4.5 Implementacija razdelitve delovnih tokov in GPU bremen 34

4.6 Implementacija nadzora in upravljanja naprav ter sistemskih storitev 36

4.7 Integracija varnostnih mehanizmov in povratnega posrednika 38
4.8 Testni primeri in postopki preverjanja delovanja 39
5 REZULTATI IN ANALIZA 41
5.1 Meritve odzivnosti in učinkovitosti sistema 41
5.2 Analiza vpliva IoT senzorjev 42
5.3 Analiza vpliva hibridnega spomina in razdelitve delovnih tokov 44
5.4 Analiza nadzora in upravljanja naprav ter sistemskih storitev 46
5.5 Analiza varnostnih mehanizmov in mehanizmov zanesljivosti 47
5.6 Razprava o rezultatih 50
6 SKLEP 52
7 VIRI IN LITERATURA 55
8 PRILOGE 58

Kazalo slik

Slika 1: Visokonivojska arhitektura večplatformskega asistenta 19
Slika 2: Potek gradnje konteksta in hibridnega spomina v sistemu 33
Slika 3: Razdelitev delovnih tokov in strojnih virov sistema 35
Slika 4: Potek nadzora in upravljanja naprav ter sistemskih storitev 37

Kazalo tabel

Tabela 1: Pregled uporabljenih tehnologij in njihove vloge v sistemu 28
Tabela 2: Uporabljene IoT komponente in njihove vloge 31
Tabela 3: Povzetek meritev odzivnosti besedilnih poti sistema 42
Tabela 4: Povzetek preverjanja IoT povezljivosti in statusnih tokov 43
Tabela 5: Povzetek preverjanja hibridnega spomina in razdelitve delovnih tokov 45
Tabela 6: Povzetek preverjanja nadzora in upravljanja naprav ter storitev 47
Tabela 7: Povzetek preverjanja varnostnih in zanesljivostnih mehanizmov 49

1 UVOD

Digitalni asistenti danes niso več omejeni na en sam komunikacijski kanal ali eno samo funkcijo. Vse pogosteje se pojavlja potreba po rešitvah, ki lahko delujejo v več okoljih hkrati, se povezujejo z zunanjimi napravami, obdelujejo podatke iz različnih virov ter pri tem ohranjajo zanesljivo in varno delovanje. Takšni sistemi so posebej zanimivi v razvojnih in domačih okoljih, kjer se prepletajo spletni vmesniki, komunikacijske platforme, podatkovne baze, IoT naprave in modeli umetne inteligence.

Diplomsko delo obravnava razvoj in implementacijo večplatformskega asistenta, ki združuje spletni vmesnik, Discord integracijo, IoT podporo ter hibridni sistem spomina. Rešitev temelji na modularni in porazdeljeni zasnovi, pri kateri posamezni moduli prevzemajo različne naloge, kot so komunikacija z uporabnikom, upravljanje naprav, obdelava govora, hramba podatkov in nadzor storitev. Zaradi takšne zasnove so v ospredju vprašanja uporabnosti, razširljivosti, odzivnosti, zanesljivosti in varnosti, ki jih delo obravnava z vidika načrtovanja, implementacije in analize doseženih rezultatov.

1.1 Opis področja in opredelitev problema

Področje raziskovanja zajema sistemsko inženirstvo, razvoj porazdeljenih informacijskih sistemov, integracijo umetne inteligence, podatkovne baze, IoT povezljivost ter kibernetsko varnost. V sodobnih okoljih so uporabniki vse manj zadovoljni z rešitvami, ki delujejo le na eni platformi, nimajo ustreznega spomina, ne omogočajo povezovanja z napravami in ne vključujejo ustreznih mehanizmov zaščite podatkov. Takšne omejitve zmanjšujejo praktično uporabnost sistema in otežujejo njegovo nadaljnjo razširitev.

Osrednji problem diplomskega dela zato predstavljata zasnova in izvedba asistenta, ki lahko deluje v večplatformskem okolju, povezuje različne komunikacijske in strojne komponente ter pri tem ostaja dovolj odziven, zanesljiv in varen za dejansko uporabo. Poseben izziv predstavlja usklajevanje več modulov, kot so spletni vmesnik, Discord integracija, IoT povezave, hibridni spomin ter ločeni delovni tokovi za obdelavo besedila in govora. Zaradi tega je treba sistem obravnavati kot širši porazdeljeni informacijski sistem, ne le kot posamezno aplikacijo.

1.2 Namen, cilji in osnovne trditve

Namen diplomskega dela je razviti in implementirati večplatformskega asistenta, ki povezuje spletni vmesnik, Discord integracijo, IoT podporo in hibridni sistem spomina v enotno funkcionalno celoto. Namen dela je tudi preveriti, v kolikšni meri takšna modularna in porazdeljena zasnova prispeva k boljši uporabnosti, večji razširljivosti, večji zanesljivosti in boljši varnosti sistema.

Glavni cilji diplomskega dela so:

- predstaviti modularno arhitekturo porazdeljenega asistenta,
- povezati spletni vmesnik in Discord v skupen večplatformski sistem,
- vključiti IoT podporo z uporabo ESP32 naprav,
- vzpostaviti hibridni sistem spomina, ki združuje kratkoročno, dolgoročno in semantično hrambo podatkov,
- analizirati vpliv razdelitve delovnih tokov med LLM, TTS in STT na stabilnost ter učinkovitost sistema,
- implementirati varnostne mehanizme, kot so avtentikacija, TLS, varnostne kopije in povratni posrednik,
- izvesti testiranja ter na osnovi rezultatov ovrednotiti dosežene rešitve.

V diplomskem delu bodo preverjene naslednje osnovne trditve:

- **H1:** Integracija večplatformskega okolja izboljša uporabnost in poveča razširljivost asistenta.
- **H2:** IoT podpora z ESP32 senzorji izboljša avtomatizacijo in skrajša odzivni čas sistema.
- **H3:** Hibridni spomin in razdelitev delovnih tokov zmanjšata podvajanje ukazov ter izboljšata zanesljivost in učinkovitost asistenta.
- **H4:** Uvedba varnostnih mehanizmov zmanjša tveganje nepooblaščenega dostopa in izgube podatkov v porazdeljenem asistentu.

1.3 Predpostavke in omejitve

Diplomsko delo temelji na predpostavki, da je mogoče z uporabo sodobnih lokalnih in odprtokodnih tehnologij zgraditi uporaben večplatformski asistent, ki povezuje različne funkcionalne sklope v enoten sistem. Predpostavlja se tudi, da modularna arhitektura omogoča lažje nadgrajevanje, večjo preglednost in boljšo razširljivost ter da lahko kombinacija več spominskih pristopov izboljša delo s kontekstom.

Med omejitvami je treba izpostaviti, da raziskava temelji na enem konkretnem projektu in testnem okolju, ki je omejeno z razpoložljivo strojno in omrežno infrastrukturo. Rezultati so zato vezani na uporabljene tehnologije, izbrano arhitekturo in način testiranja. Dodatna omejitev je, da se posamezni moduli sistema še razvijajo, zato delo obravnava predvsem trenutno implementirano in preverjeno stanje rešitve. Raziskava ne vključuje intervjujev, anket ali drugih raziskav s človeškimi udeleženci, temveč temelji predvsem na analizi podatkov, laboratorijskem testiranju, meritvah odzivnosti in varnostni analizi. To se ujema tudi z etičnim obrazcem projekta, kjer so kot uporabljene metode navedeni laboratorijsko testiranje sistemov, integracija IoT, meritve odzivnosti in varnostne analize.

1.4 Uporabljene raziskovalne metode

Pri izdelavi diplomskega dela je uporabljen aplikativni raziskovalni pristop, saj je cilj razviti delujoč sistem in ga ovrednotiti glede na zastavljene cilje ter hipoteze. Raziskava povezuje teoretična izhodišča in praktično implementacijo, pri čemer je poudarek na dejanski uporabnosti sistema, preverjanju njegovega delovanja in analizi doseženih rezultatov.

V diplomskem delu bodo uporabljene naslednje metode:

- analiza literature s področja porazdeljenih sistemov, umetne inteligence, IoT, podatkovnih baz in kibernetske varnosti;
- načrtovanje in modeliranje arhitekture sistema;
- implementacija posameznih modulov in njihovih integracij;
- testiranje in eksperimentalna analiza odzivnosti, zanesljivosti in stabilnosti;
- primerjalna analiza rezultatov glede na cilje in hipoteze;
- dokumentiranje razvojnih odločitev, rezultatov testov in končnih ugotovitev.

Diplomsko delo je vsebinsko razdeljeno na šest osrednjih poglavij. Po uvodnem delu sledijo teoretična izhodišča in pregled področja, kjer so predstavljeni ključni koncepti porazdeljenih sistemov, podatkovnih baz, IoT integracije in kibernetske varnosti. V nadaljevanju sta prikazana načrtovanje in arhitektura sistema, nato pa implementacija posameznih sklopov. Peto poglavje predstavlja rezultate testiranj in njihovo analizo glede na zastavljene hipoteze, medtem ko zadnje poglavje povzema ključne ugotovitve, ovrednoti hipoteze ter izpostavi omejitve in možnosti nadaljnjega razvoja.

2 TEORETIČNA IZHODIŠČA IN PREGLED PODROČJA

2.1 Sistemski inženiring in porazdeljeni sistemi

Sistemski inženiring predstavlja interdisciplinaren pristop k načrtovanju, razvoju, integraciji in upravljanju kompleksnih sistemov. Njegovo bistvo ni osredotočenost na posamezno komponento, temveč na celoten sistem in povezave med njegovimi deli. Takšen pristop je posebej pomemben pri rešitvah, kjer sodeluje več medsebojno povezanih modulov, saj omogoča preglednejšo razdelitev funkcij, lažje obvladovanje kompleksnosti in postopno nadgrajevanje sistema (INCOSE, 2019; NASA, 2016).

V sodobnem informacijskem okolju sistem pogosto ni več ena sama aplikacija ali ena sama naprava, temveč skupek medsebojno povezanih elementov, ki skupaj izvajajo določene funkcije. Pri bolj kompleksnih rešitvah je zato smiselno uporabiti tudi koncept sistema sistemov, kjer več podsistemov sodeluje pri zagotavljanju zmožnosti, ki je posamezni del sam ne bi mogel zagotoviti. NIST tak sistem opredeljuje kot množico sistemov ali sistemskih elementov, ki s sodelovanjem ustvarjajo edinstveno zmožnost, ki je posamezen element sam ne more doseči (NIST, b. d.).

Porazdeljeni sistemi predstavljajo poseben razred informacijskih sistemov, pri katerih komponente delujejo v različnih procesih, storitvah ali napravah ter med seboj komunicirajo prek omrežja. Njihova prednost je predvsem v večji razširljivosti, prilagodljivosti in možnosti ločevanja nalog med več funkcionalnih sklopov, vendar takšna zasnova prinaša tudi izzive, kot so zakasnitve, odvisnost od komunikacijskih poti in potreba po usklajevanju podatkov med več deli sistema (Voas, 2016).

Takšno razumevanje je za obravnavani sistem ključno, saj rešitev ne temelji na eni sami aplikaciji, temveč na usklajenem delovanju spletnega vmesnika, Discord integracije, podatkovnih plasti, govornih procesov in IoT naprav. Sistem je zato smiselno obravnavati kot porazdeljeno in modularno informacijsko rešitev.

2.2 Podatkovne baze in hibridni pristopi k spominu

Podatkovne baze predstavljajo temeljni gradnik sodobnih informacijskih sistemov, saj omogočajo strukturirano hrambo, organizacijo in iskanje podatkov. Relacijske baze so posebej primerne za delo z jasno določenimi zapisi, povezavami med entitetami in natančnimi poizvedbami. Pri takšni zasnovi so pomembne predvsem doslednost podatkov, transakcijska zanesljivost in možnost nadzorovanega sočasnega dostopa, kar je posebej izrazito pri uporabi pogona InnoDB v okolju MySQL. V dokumentaciji MySQL je InnoDB predstavljen kot transakcijsko varen pogon, ki podpira lastnosti ACID in delo s transakcijami, kar je pomembno pri sistemih, kjer se podatki pogosto zapisujejo, berejo in posodabljajo (Oracle, b. d.).

Klasična relacijska hramba pa sama po sebi ni vedno dovolj za sisteme, ki morajo poleg natančnih poizvedb obravnavati tudi semantično podobnost med vsebinami. Zato se v sodobnih rešitvah vse pogosteje uporablja hibridni pristop, pri katerem se strukturirani podatki hranijo v relacijski bazi, pomensko sorodne vsebine pa v vektorski obliki. Razširitev pgvector za PostgreSQL omogoča shranjevanje vektorjev in izvajanje iskanja po podobnosti, pri čemer podpira tako natančno kot približno iskanje najbližjih sosedov. Dokumentacija razširitve pgvector poudarja, da ta privzeto uporablja natančno iskanje najbližjih sosedov, z dodatnimi indeksnimi pristopi pa omogoča tudi približno iskanje najbližjih sosedov, ki je pri večjih količinah podatkov hitrejše (pgvector, b. d.).

Hibridni pristop k spominu je za obravnavani sistem pomemben zato, ker mora asistent povezovati trenutno interakcijo, trajnejšo zgodovino in širši semantični kontekst. Zaradi tega je smiselno razlikovati med kratkoročnim pomnilnikom, strukturirano relacijsko hrambo in semantičnim priklicem vsebin.

2.3 IoT integracija in zanesljivost sistemov

Internet stvari omogoča povezovanje fizičnih naprav, senzorjev, komunikacijskih poti in programske logike v enotno funkcionalno celoto. NIST poudarja, da je takšne sisteme smiselno razumeti skozi osnovne gradnike, kot so zaznavanje, računanje, komunikacija in izvajanje dejanj v fizičnem okolju, ter da so ta načela uporabna širše pri porazdeljenih sistemih, ki uporabljajo IoT tehnologije. Tak pristop pomeni, da IoT ni le skupek posameznih naprav, temveč okolje, v katerem podatki iz fizičnega sveta postanejo del širšega informacijskega toka (Voas, 2016).

Pri integraciji IoT komponent je pomembna interoperabilnost, saj morajo naprave, storitve in osrednji sistem med seboj zanesljivo izmenjevati podatke in sodelovati pri izvajanju nalog. V sistemih, ki povezujejo digitalni in fizični del okolja, je pomembno tudi razumevanje širšega konteksta kibernetsko-fizičnih sistemov, kjer merilni tok predstavlja digitalni zapis stanja fizičnega sveta, izvedbeni tok pa vpliva nazaj na fizično okolje. Takšna zasnova je pomembna zato, ker omogoča prehod iz pasivne obdelave podatkov v aktivno spremljanje in odzivanje (NIST CPS Public Working Group, 2017).

Zanesljivost v takšnih sistemih pomeni, da sistem svoje funkcije izvaja dovolj dosledno tudi ob motnjah, izpadih povezave ali delni nedosegljivosti posameznih komponent. NIST kibernetsko odpornost opredeljuje kot zmožnost sistema, da predvideva, prenese, okreva po neugodnih razmerah ter se jim prilagodi. Čeprav ta koncept izhaja iz področja kibernetske odpornosti, je uporaben tudi pri širšem razumevanju zanesljivosti porazdeljenih in IoT sistemov, kjer so motnje in delni izpadi pričakovani del delovanja (Ross et al., 2021).

Za obravnavani sistem je IoT integracija pomembna zato, ker asistent ne deluje le kot programski pogovorni vmesnik, temveč tudi kot platforma, ki sprejema podatke iz fizičnega okolja in lahko na njihovi osnovi sproža nadaljnje akcije. Vrednost IoT podpore zato ni le v povezovanju senzorjev, temveč v večji stopnji avtomatizacije, širšem kontekstu delovanja in tesnejši povezavi med digitalnim in fizičnim delom sistema.

2.4 Kibernetska varnost v večplatformskih okoljih

Kibernetska varnost v večplatformskih okoljih je pomembna zato, ker sodobni sistemi niso več omejeni na eno samo aplikacijo ali eno samo omrežno točko. Namesto tega vključujejo več uporabniških vmesnikov, več storitev, podatkovnih virov, komunikacijskih poti in zunanjih naprav. Takšna zasnova povečuje funkcionalnost sistema, hkrati pa širi tudi njegovo napadalno površino, zato mora varnost vključevati uporabnike, podatke, storitve in pravila dostopa do posameznih virov. NIST pri ničelnem zaupanju poudarja, da se varnostni poudarek premakne z omrežnega perimetra na uporabnike, sredstva in vire ter da se dostop ne odobri zgolj na osnovi lokacije v omrežju (Rose et al., 2020).

V takšnem okolju sta ključna pojma avtentikacija in avtorizacija. Avtentikacija pomeni preverjanje identitete uporabnika ali procesa, avtorizacija pa določanje njegovih pravic dostopa. NIST pri modelu ničelnega zaupanja izrecno poudarja, da morata biti avtentikacija in avtorizacija izvedeni pred vzpostavitvijo dostopa do vira, CISA pa večfaktorsko avtentikacijo opredeljuje kot večslojni pristop, pri katerem mora uporabnik predložiti kombinacijo dveh ali več načinov preverjanja identitete (Rose et al., 2020; CISA, 2022).

Pomemben del varnosti večplatformskih sistemov je tudi zaščita komunikacije, sej, žetonov, poverilnic in drugih občutljivih podatkov sistema. OWASP pri varnosti REST storitev poudarja uporabo HTTPS končnih točk za zaščito poverilnic in integritete prenosa, pri spletnih sistemih pa sta pomembna tudi pravilno upravljanje sej ter nadzorovana hramba gesel, API ključev, žetonov in drugih občutljivih podatkov sistema (OWASP, b. d.-a; OWASP, b. d.-b; OWASP, b. d.-c).

Pri izpostavljanju storitev ima pomembno vlogo tudi povratni posrednik. Ta lahko deluje kot enotna vstopna točka, ki usmerja promet do notranjih storitev in poenostavlja upravljanje zaščitenih povezav. Dokumentacija Caddy poudarja, da rešitev samodejno zagotavlja TLS certifikate, jih obnavlja in preusmerja promet iz HTTP na HTTPS, kar zmanjšuje možnost napak pri ročni konfiguraciji (Caddy, b. d.).

Za obravnavani sistem to pomeni, da kibernetske varnosti ni smiselno obravnavati kot ločene dodatne funkcije, temveč kot eno izmed temeljnih načel njegove zasnove. Ker sistem vključuje več vmesnikov, podatkovne plasti, komunikacijo z napravami in upravljanje dostopa, mora biti varnost vključena skozi celoten življenjski cikel delovanja sistema.

2.5 Pregled obstoječih rešitev

Pri razvoju večplatformskega asistenta je smiselno pregledati tudi obstoječe rešitve, saj pokažejo različne pristope k povezovanju uporabniških vmesnikov, govornih funkcij, naprav, pomnilnika in varnosti. Namen tega pregleda ni zajeti vseh digitalnih asistentov, temveč izpostaviti nekaj reprezentativnih primerov, ki pokrivajo podobne funkcionalne sklope kot obravnavani sistem.

Med pomembnejšimi primeri je Home Assistant, ki daje poudarek lokalnemu upravljanju naprav, avtomatizaciji in glasovnemu upravljanju prek Assist. Dokumentacija navaja, da lahko Assist deluje lokalno ali uporablja zunanje LLM-je ter da ga je mogoče uporabljati na telefonu, tablici ali drugih prilagojenih glasovnih napravah (Home Assistant, b. d.).

Open Voice OS predstavlja odprtokodno, skupnostno usmerjeno platformo za gradnjo glasovno vodenih vmesnikov na različnih napravah, pri čemer poudarja prilagodljivost, večplatformskost ter zasebnost in varnost. Takšen pristop je zanimiv predvsem z vidika odprtega glasovnega ekosistema (Open Voice OS, b. d.).

Na področju pogovornih in glasovnih agentov je pomemben tudi Rasa, ki daje poudarek realnočasovnim glasovnim agentom, nizki zakasnitvi, izmeničnemu vodenju pogovora in razširljivosti. Takšen okvir je primeren za zahtevnejše glasovne scenarije, vendar je bolj usmerjen v gradnjo pogovornih agentov kot v povezovanje z lokalnimi IoT napravami (Rasa, b. d.).

Med sodobnimi rešitvami, usmerjenimi v velike jezikovne modele, sta relevantna tudi LangChain in LangGraph, kjer je poudarek na agentih, orodjih in delu s spominom. Dokumentacija navaja, da agenti združujejo jezikovne modele z orodji, spominski del pa razlikuje med kratkoročnim in dolgoročnim pomnilnikom. Takšna ogrodja predstavljajo zmogljivo razvojno osnovo, vendar sama po sebi še ne pomenijo končne večplatformske asistentske rešitve (LangChain, b. d.-a; LangChain, b. d.-b).

Iz pregleda je razvidno, da posamezne rešitve praviloma pokrivajo le del funkcionalnosti, ki so za obravnavani sistem pomembne kot celota. Zato je bila izbrana lastna modularna zasnova, ki združuje več uporabniških vmesnikov, IoT podporo, hibridni spomin, lastno logiko orkestracije ter varnostne mehanizme, prilagojene konkretnemu okolju uporabe.

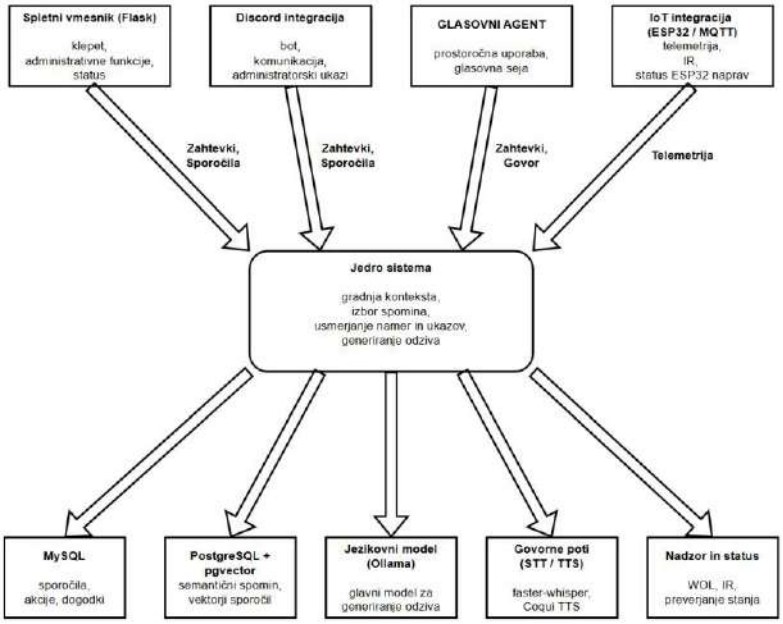
3 NAČRTOVANJE IN ARHITEKTURA SISTEMA

3.1 Modularna arhitektura asistenta

Modularna arhitektura predstavlja eno izmed ključnih zasnovnih odločitev obravnavanega sistema, saj omogoča jasno ločevanje posameznih funkcionalnih sklopov glede na njihovo vlogo in način uporabe. Namesto monolitne aplikacije je sistem zasnovan kot skupek medsebojno povezanih, vendar logično ločenih modulov, ki sodelujejo pri obdelavi uporabniških zahtevkov, komunikaciji z zunanjimi platformami, upravljanju podatkov, izvajanju ukazov ter povezovanju s strojnimi komponentami.

Osrednje jedro sistema je zasnovano kot skupna logična plast, ki ni vezana na en sam uporabniški vmesnik. Posamezne platforme, kot so spletni vmesnik, Discord in druge integracije, delujejo kot odjemalci istega osrednjega jedra sistema. Na ta način so delo s spominom, obdelava ukazov, gradnja konteksta in logika odločanja centralizirani, namesto da bi bili podvojeni v posameznih integracijah.

Na arhitekturni ravni je sistem razdeljen na štiri glavne sklope: integracijske module, osrednje procesne module, podatkovno plast ter podporne storitve. Integracijski moduli skrbijo za stik z zunanjimi platformami, procesni moduli za obdelavo zahtevkov, podatkovna plast za hrambo in priklic informacij, podporne storitve pa za beleženje dogodkov, MQTT komunikacijo, nadzor naprav in administrativne funkcije. Takšna zasnova omogoča jasno ločevanje med skupno logiko sistema in vedenjem, ki je vezano na posamezno integracijo. Pomembna je predvsem zato, ker zmanjšuje podvajanje kode, poenostavlja vzdrževanje in omogoča postopno razširjanje sistema brez posegov v celotno jedro rešitve.



Slika 1: Visokonivojska arhitektura večplatformskega asistenta

Vir: (Lastni vir)

3.2 Integracija več vmesnikov (web/Discord)

Ena izmed osrednjih značilnosti obravnavanega sistema je podpora več uporabniškim vmesnikom. Sistem ni vezan na eno samo aplikacijo ali komunikacijsko okolje, temveč omogoča dostop prek različnih vstopnih točk. V okviru diplomskega dela sta kot glavni integraciji obravnavana spletni vmesnik in Discord, saj predstavljata dva različna načina uporabe istega asistenta.

Spletni vmesnik predstavlja bolj strukturirano obliko dostopa, primerno za preglednejšo komunikacijo, administrativni nadzor, prikaz statusa sistema in upravljanje posameznih storitev. Discord integracija predstavlja bolj sprotno in komunikacijsko usmerjeno obliko uporabe, saj asistenta vključuje v komunikacijsko okolje, v katerem interakcija že poteka.

Ključna arhitekturna odločitev te zasnove je, da spletni in Discord del ne delujeta kot dve ločeni rešitvi, temveč kot dva odjemalca istega osrednjega jedra sistema. Skupni ostajajo gradnja odziva, delo s spominom, gradnja konteksta, obdelava ukazov in povezovanje z drugimi storitvami, medtem ko se posamezna vmesnika razlikujeta predvsem po načinu sprejema vhodov, preverjanju uporabnika in obliki prikaza rezultatov.

Takšna razmejitev omogoča enotno delovanje sistema ob hkratni prilagoditvi različnim načinom uporabe ter predstavlja osnovo za nadaljnjo širitev na dodatne dostopne poti brez posega v osrednje jedro sistema.

3.3 IoT integracija (ESP32 + PIR)

IoT plast v obravnavanem sistemu predstavlja povezavo med osrednjim jedrom sistema in fizičnim okoljem. Njena vloga je omogočiti sprejem podatkov iz zunanjih naprav ter po potrebi sprožanje dejanj v fizičnem prostoru, zato je zasnovana kot ločen, vendar z osrednjim jedrom povezan sklop.

ESP32 v tej zasnovi nastopa kot robna naprava, ki osrednjemu jedru sistema posreduje stanje okolja ali sprejema ukaze za izvedbo dejanj. Tak pristop ločuje zaznavanje in izvajanje dejanj v fizičnem okolju od osrednje logike asistenta ter omogoča vključevanje novih naprav brez spremembe temeljne arhitekture.

Trenutna IoT integracija je osredotočena na dva sklopa. Prvi sklop predstavlja zajem telemetrije iz okolja, kot sta temperatura in relativna vlažnost, drugi sklop pa izvršilni del za izvajanje konkretnih dejanj, kot je pošiljanje IR ukazov. PIR senzor je v arhitekturi predviden kot del širšega IoT okvira, vendar še ni enako zrel kot telemetrija in IR izvršilni del.

Pomemben povezovalni element tega sklopa je tudi MQTT komunikacijska plast. Naprave z osrednjim sistemom ne komunicirajo neposredno točkovno, temveč prek sporočilne plasti, kar omogoča preglednejše usmerjanje telemetrije in ukazov ter lažje vključevanje novih robnih naprav.

Za obravnavani sistem je IoT integracija pomembna predvsem zato, ker asistenta razširi iz zgolj programske rešitve v sistem, ki lahko sprejema podatke iz fizičnega okolja in se nanje tudi odziva.

3.4 Hibridni spomin (RAM buffer, MySQL, vektorska baza)

Hibridni spomin predstavlja enega izmed osrednjih arhitekturnih elementov obravnavanega sistema, saj združuje več pristopov k hrambi in priklicu informacij v enotno funkcionalno celoto. Namesto ene same podatkovne plasti sistem uporablja več slojev, ki imajo različne naloge in različne časovne horizonte uporabe.

Prvi sloj predstavlja delovni spomin (RAM), ki ohranja svež kontekst interakcije in začasno stanje seje. Drugi sloj predstavlja strukturirano dolgoročno hrambo v relacijski bazi, namenjeno zgodovini interakcij, sistemskim dogodkom in drugim trajnejšim zapisom. Tretji sloj predstavlja semantični spomin, ki temelji na vektorski predstavitvi vsebin in omogoča priklic pomensko sorodnih informacij.

Ključna prednost hibridnega pristopa je v tem, da posamezni sloji ne opravljajo enake naloge, temveč se med seboj dopolnjujejo. Delovni spomin (RAM) skrbi za neposredni kontekst, relacijska baza za trajnost in strukturiranost, semantični spomin pa za priklic pomensko relevantnih vsebin.

Pomemben arhitekturni vidik tega sklopa je tudi gradnja konteksta. Sistem pri gradnji odgovora ne vključuje vseh razpoložljivih podatkov, temveč iz različnih plasti izbira informacije, ki so za trenutno nalogo najbolj relevantne. Pri tem mora uskladiti svežino podatkov, njihovo pomembnost, semantično sorodnost ter strukturo posameznega vira.

Za obravnavani sistem je hibridni spomin ključen predvsem zato, ker omogoča stabilnejšo gradnjo konteksta v večplatformskem okolju. Sistem lahko s tem pristopom ohranja tekoč pogovor, si zapomni pomembne pretekle informacije in po potrebi prikliče širši semantični kontekst.

3.5 Razdelitev delovnih tokov (LLM, TTS, STT) in porazdelitev bremen

Ena izmed ključnih arhitekturnih odločitev obravnavanega sistema je razdelitev delovnih tokov med ločene procesne sklope, ki izvajajo različne vrste obdelave. Sistem zato ne uporablja ene same izvajalne poti za vse funkcije, temveč ločene sklope za jezikovno obdelavo, pretvorbo govora v besedilo, sintezo govora in podporne procese, kot je semantični priklic.

Osrednji procesni del predstavlja jezikovni model, ki skrbi za razumevanje besedilnega konteksta, gradnjo odziva in odločanje o nadaljnjih korakih sistema. Od njega so ločene govorne poti STT in TTS, saj obdelujeta drugačno vrsto podatkov in imata drugačne časovne zahteve. Ločeno je zasnovan tudi semantični priklic, ki pripravlja dodatni kontekst iz spominskega sistema.

Pomemben element te zasnove je osrednje usmerjanje vhodnih in izhodnih tokov. Posamezni sklopi obdelajo svoj del naloge, nato pa se rezultat vrne v osrednje jedro sistema oziroma v ustrezni izhodni modul. Takšen pristop zmanjšuje podvajanje funkcij in omogoča neodvisen razvoj posameznih procesnih poti.

Razdelitev delovnih tokov je povezana tudi s porazdelitvijo bremen. Jezikovni model, govorna prepoznava, sinteza govora in semantični priklic sistema ne obremenjujejo na enak način, zato njihova ločitev zmanjšuje možnost zakasnitev, blokad in medsebojnega vplivanja.

Ta zasnova je za obravnavani sistem pomembna predvsem zato, ker izboljšuje preglednost procesov, zmanjšuje neposredno odvisnost med posameznimi vrstami obdelave in omogoča učinkovitejšo porazdelitev bremen v večplatformskem okolju.

3.6 Nadzor in upravljanje naprav ter sistemskih storitev

Pomemben del arhitekture obravnavanega sistema predstavljata nadzor nad napravami in upravljanje sistemskih storitev. Asistent ni zasnovan zgolj kot pogovorni vmesnik, temveč kot platforma, ki lahko na osnovi uporabniškega vhoda ali notranjih pravil sproža konkretna dejanja v digitalnem in fizičnem okolju.

Z arhitekturnega vidika je pomembno, da nadzor naprav in storitev ni vezan na posamezen uporabniški vmesnik, temveč na skupno jedro sistema. To pomeni, da spletni vmesnik, Discord in druge integracije ukazov ne izvajajo samostojno, temveč zahteve posredujejo osrednjemu jedru sistema, kjer se določi namen ukaza, preveri njegova veljavnost in izbere ustrezna izvedbena pot.

Pri upravljanju sistemskih storitev arhitektura uporablja ukazni usmerjevalnik, ki skrbi za prepoznavo ciljne storitve, izbiro ustreznega upravljalnega modula in sprožitev dejanja. Podoben princip velja tudi za fizične naprave, kjer osrednje jedro sistema uporablja MQTT komunikacijsko plast za povezovanje z zunanjimi komponentami. V obravnavanem projektu to vlogo opravlja sporočilno zasnovana komunikacija, ki povezuje sistem z napravami ESP32.

Pomemben vidik tega sklopa je tudi razlikovanje med sinhronimi in asinhronimi dejanji. Nekateri ukazi vrnejo rezultat skoraj takoj, drugi pa sprožijo daljši proces, kot je ponovni zagon storitve ali posredovana akcija prek zunanje naprave. Zaradi tega mora arhitektura omogočati, da uporabniški odziv in dejanska izvedba nista vedno vezana na isti trenutek.

Nadzor nad napravami in sistemskimi storitvami je povezan tudi z beleženjem dogodkov in rezultatov akcij, saj mora sistem omogočati sledljivost sproženih ukazov, ciljev in rezultatov izvedbe. Za obravnavani sistem je ta sklop pomemben predvsem zato, ker asistenta razširi iz pogovorne rešitve v operativno platformo za upravljanje digitalnega in fizičnega okolja.

3.7 Varnostni mehanizmi (TLS, avtentikacija, varnostne kopije, Caddy)

Ker sistem združuje več uporabniških vmesnikov, lokalno podatkovno hrambo, komunikacijo z zunanjimi napravami in upravljanje sistemskih storitev, je varnost pomemben sestavni del njegove arhitekture. Varnostni mehanizmi niso zasnovani kot ločen dodatek, temveč kot sklop zaščitnih plasti, ki omejujejo dostop, varujejo podatke in zmanjšujejo tveganje nepooblaščenih dejanj.

Prvi pomemben element je razmejitev med zasebnimi in javno dostopnimi plastmi sistema. Obravnavani sistem je v trenutni zasnovi namenjen predvsem lokalni oziroma zasebni uporabi, zato glavni spletni del ni zamišljen kot odprta javna storitev. Takšna ureditev zmanjšuje napadalno površino, hkrati pa zahteva ločeno obravnavo tistih integracij, ki delujejo prek zunanjih platform, kot je Discord.

Na ravni dostopa sistem temelji na avtentikaciji, avtorizaciji ter ločevanju upravljalnih funkcij. Spletni del uporablja prijavo z uporabniškim računom in geslom, občutljive funkcije pa so dodatno zaščitene z ustreznimi pravicami in dodatnim preverjanjem identitete. Discord integracija uporablja drugačen model, vezan na preverjene identitete znotraj same platforme.

Pomemben del varnostne arhitekture predstavlja tudi zaščita podatkovnih virov in notranjih storitev. Podatkovne baze, MQTT plast in drugi podporni servisi niso zasnovani kot javno odprte točke, temveč kot deli notranjega sistema, do katerih dostopajo le ustrezni procesi in pooblaščeni uporabniki.

Pri dostopu do spletnih storitev ima pomembno vlogo tudi povratni posrednik Caddy, ki predstavlja osnovo za usmerjanje spletnega prometa in zaščitene povezave. Podobno velja za TLS, ki je v zasnovi predviden kot pomemben mehanizem zaščite komunikacije, vendar v pretežno lokalnem načinu delovanja še ni osrednji element vseh poti. Manj zrel del tega sklopa trenutno predstavljajo varnostne kopije operativnih podatkov, ki so v zasnovi sicer predvidene, vendar še niso enako razvite kot drugi varnostni mehanizmi.

Za obravnavani sistem je ta arhitekturni sklop pomemben predvsem zato, ker pokaže, da varnost temelji na kombinaciji več zaščitnih plasti, ne na enem samem mehanizmu.

3.8 Mehanizmi zanesljivosti (Uptime Kuma, samodejni ponovni zagoni, varnostne kopije)

Zanesljivost predstavlja pomemben del arhitekture sistema, saj večplatformski asistent ni uporaben, če posamezne storitve pogosto odpovedujejo, če sistem ne zazna izpadov ali če po napaki ne omogoča ustreznega okrevanja. Ker projekt združuje več samostojnih komponent, je bilo treba že na arhitekturni ravni predvideti mehanizme za spremljanje stanja, odzivanje na napake in omejevanje posledic posameznih izpadov.

Prvi pomemben element zanesljivosti predstavlja spremljanje stanja posameznih storitev. V večkomponentnem sistemu ni dovolj spremljati le osrednje aplikacije, temveč tudi dosegljivost podpornih storitev in drugih pomembnih delov okolja.

Drugi pomemben mehanizem predstavlja nadzorovano ponovno zaganjanje storitev. Ker se obravnavani sistem opira na več ločenih procesov, izpad ene komponente ne pomeni nujno ponovnega zagona celotnega okolja. Ločitev posameznih storitev omogoča hitrejše okrevanje po napaki in manjši vpliv posameznega izpada na celotno delovanje sistema.

Pomembno je tudi razlikovanje med spremljanjem stanja in dejanskim odzivom sistema. Sistem ne potrebuje le informacije, da je prišlo do težave, temveč tudi mehanizem, s katerim se je mogoče na to težavo odzvati. Zaradi tega so nadzor, zaznavanje in sprožanje ukrepov med seboj povezani, vendar niso zliti v eno samo neločljivo celoto.

V trenutni zasnovi je zanesljivost sistema predvsem oprta na lastne nadzorne in statusne mehanizme. Uptime Kuma je pri tem smiselno obravnavati kot podporno oziroma dopolnilno rešitev, ne pa kot osrednji mehanizem zanesljivosti. Poseben del tega sklopa predstavljajo tudi varnostne kopije, pri čemer je treba ločiti med zaščito izvorne kode in zaščito operativnih podatkov, saj ti še nimajo enako zrele in avtomatizirane strategije arhiviranja.

Za obravnavani sistem so mehanizmi zanesljivosti pomembni predvsem zato, ker omogočajo stabilnejše delovanje v večplatformskem okolju, boljšo preglednost nad stanjem komponent in hitrejše okrevanje po delnih izpadih.

4 IMPLEMENTACIJA IN IZVEDBA

4.1 Razvojno okolje in uporabljene tehnologije

Implementacija obravnavanega sistema je potekala v lokalnem večkomponentnem razvojnem okolju, kjer posamezni deli niso združeni v eno samo aplikacijo, temveč v več medsebojno povezanih procesov in storitev. Takšna izvedba je bila izbrana zato, ker sistem vključuje spletni vmesnik, Discord integracijo, obdelavo z jezikovnim modelom, govorne poti STT in TTS, podatkovne baze, IoT komunikacijo ter podporne storitve za nadzor in upravljanje.

Osrednji razvojni jezik sistema je Python. Projekt uporablja ločeno virtualno okolje, odvisnosti pa so zbrane v datotekah requirements.txt in requirements.lock.txt, kar prispeva k ponovljivosti namestitve in preglednosti okolja.

Spletni del sistema je implementiran z uporabo ogrodja Flask, ki skrbi za spletni uporabniški vmesnik, prijavni del, administrativne in statusne poti ter funkcionalnosti API. Discord integracija je implementirana ločeno z uporabo knjižnice Pycord, pri čemer oba dela uporabljata isto osrednje jedro sistema.

Za strukturirano hrambo podatkov je uporabljena baza MySQL, za semantični del spomina pa PostgreSQL z razširitvijo pgvector. Takšna kombinacija ločuje relacijsko hrambo zgodovine in metapodatkov od semantičnega priklica po podobnosti.

Na ravni jezikovne obdelave sistem uporablja lokalno izvajanje modelov prek orodja Ollama. Glavni model je namenjen generiranju odgovorov, ločen proces za pripravo vektorskih predstavitev pa podpira semantični priklic. Pretvorba govora v besedilo je izvedena z uporabo orodja faster-whisper, sinteza govora pa z uporabo sistema Coqui TTS, kjer so vključene poti VCTK, XTTS in Piper.

Za IoT komunikacijo je uporabljen MQTT pristop z uporabo knjižnice paho-mqtt in lokalnega posrednika Eclipse Mosquitto. Ta komunikacijska plast je namenjena telemetriji, povezovanju naprav ESP32 in usmerjanju ukazov do zunanjih naprav, ki izvajajo dejanja.

Repozitorij vključuje ločene zagonske točke za Flask, Discord, vektorski delavec in glasovnega agenta, kar potrjuje modularno izvedbo posameznih storitev. V arhitekturo je vključen tudi Caddy kot povratni posrednik, za verzioniranje kode pa lokalni strežnik Gitea.

Z vidika strojne opreme sistem vključuje osrednji računalniški del za glavne storitve ter robno plast z napravami ESP32, natančnejša razdelitev bremen pa je predstavljena v poglavju 4.5. Pregled uporabljenih tehnologij in njihovih vlog v sistemu je prikazan v Tabeli 1.

Tabela 1: Pregled uporabljenih tehnologij in njihove vloge v sistemu

Sklop	**Tehnologija**	**Vloga v sistemu**
Osrednji razvojni jezik	Python	Glavni jezik implementacije
Izolacija odvisnosti	Virtualno okolje (venv)	Ločevanje projektnih knjižnic in ponovljivost okolja
Spletni del	Flask	Spletni vmesnik, API poti, administrativni del
Discord integracija	Pycord	Komunikacija z uporabniki prek Discorda
Relacijska hramba	MySQL	Zgodovina interakcij, dogodki, strukturirani podatki
Vektorski spomin	PostgreSQL + pgvector	Semantični priklic in delo z vektorskimi predstavitvami
LLM izvajanje	Ollama	Lokalno izvajanje jezikovnega modela
STT	faster-whisper	Pretvorba govora v besedilo
TTS	Coqui TTS – VCTK / XTTS / Piper	Sinteza govora
IoT komunikacija	MQTT / paho-mqtt (Eclipse Mosquitto)	Povezava med osrednjim jedrom sistema in napravami ESP32
Povratni posrednik	Caddy	Usmerjanje spletnega prometa in priprava na zaščitene povezave
Verzioniranje kode	Gitea	Lokalno verzioniranje in hramba kode

Vir: (Lastni vir)

4.2 Implementacija spletnega vmesnika in Discord integracije

Implementacija večplatformskega dostopa do sistema je bila izvedena z ločitvijo spletnega in Discord dela v dva samostojna odjemalska sloja, ki uporabljata isto osrednje jedro sistema. Takšna izvedba omogoča sprejem uporabniških vhodov prek različnih komunikacijskih poti, medtem ko gradnja konteksta, delo s spominom, obdelava ukazov in generiranje odziva ostajajo centralizirani v osrednjem jedru sistema.

Spletni del je implementiran z uporabo ogrodja Flask. Osrednja zagonska točka ustvari aplikacijo, nastavi sejne parametre in registrira ločene funkcionalne sklope prek mehanizma Blueprint. Spletne poti so razdeljene na prijavni del, klepetalni del, administrativne in statusne poti ter nadzorne funkcije sistema.

Prijavni del skrbi za avtentikacijo uporabnika in vzpostavitev spletne seje. Implementacija vključuje preverjanje uporabniškega računa, obravnavo gesla, podporo za 2FA/TOTP, rezervne kode ter beleženje prijavnih poskusov. Po uspešni prijavi se v sejo shrani strukturiran uporabniški zapis, ki se nato uporablja pri zaščiti zasebnih in administrativnih poti.

Klepetalni spletni del uporablja posebno klepetalno pot, ki sprejme uporabnikov zahtevek v obliki JSON. Implementacija najprej preveri veljavnost vhodnih podatkov, izvede osnovno moderacijo, določi oziroma ustvari pogovorno nit ter razvrsti pomembnost sporočila. Nato preveri namene zahtevka in ga, če ne gre za neposredni ukazni tok, posreduje osrednjemu jedru sistema za generiranje odgovora.

Administrativne, statusne in nadzorne poti omogočajo vpogled v spominske strukture, preverjanje dosegljivosti podsistemov ter sprožanje konkretnih akcij, kot sta Wake-on-LAN in IR upravljanje. Te poti so zaščitene z uporabniško sejo in dodatnim preverjanjem administratorskih pravic.

Discord integracija je implementirana kot ločen bot modul, ki uporablja Pycord in deluje neodvisno od spletnega dela, vendar nad istim osrednjim jedrom sistema. Bot ob novih sporočilih določi, ali gre za ukaz, neposredno komunikacijo z asistentom ali za običajen pogovor v kanalu.

Ko sporočilo vstopi v aktivni obdelovalni tok, implementacija najprej prilagodi omembe v berljivejšo obliko, preveri namene in nato shrani uporabniški vnos skupaj s platformnimi podatki, kot so uporabnik, kanal in strežnik. Nato se zgradi kontekst in pokliče skupna logika za generiranje odgovora. Končni rezultat gre skozi izhodni usmerjevalnik, po pošiljanju pa se odgovor shrani v spominski sloj kot asistentski zapis.

Discord del vključuje tudi ukaze za hitrejše izvajanje izbranih funkcij, na primer pomoč, ping ter administratorski ponovni zagon storitev. Ključna značilnost te implementacije je ponovna uporaba skupnih gradnikov. Obe integraciji uporabljata isto osrednje jedro sistema za gradnjo konteksta, delo s spominom, preverjanje namenov, generiranje odgovorov in izhodno usmerjanje, razlikujeta pa se predvsem v načinu sprejema vhodov, preverjanju uporabnika in obliki prikaza rezultatov.

4.3 Implementacija IoT podpore in povezave s sistemom

Implementacija IoT podpore je bila izvedena z uporabo mikrokontrolerjev ESP32, ki v sistemu nastopajo kot robne naprave za zajem telemetrije in izvajanje dejanj. Na programski ravni sta bila pripravljena dva ločena profila vgrajene programske opreme: prvi za telemetrijo in okolijske meritve, drugi za izvrševanje ukazov.

Oba sklopa vgrajene programske opreme uporabljata skupno osnovo za omrežno povezavo in MQTT komunikacijo. Po vzpostavitvi povezave se naprava poveže v lokalno brezžično omrežje, nato z lokalnim MQTT posrednikom in se prijavi v skupni tematski prostor sistema. V implementaciji je uporabljen enoten MQTT imenski prostor oblike …/esp/<id_naprave>/..., kjer ima vsaka naprava svoj identifikator in ločene ukazne, statusne, dogodkovne ter telemetrične tokove.

Telemetrični sklop vgrajene programske opreme periodično objavlja nadzorna sporočila o času delovanja, kakovosti signala in prostem pomnilniku ter meritve okoljskih senzorjev. V implementaciji sta uporabljena senzor AHT20 za temperaturo in relativno vlažnost ter senzor BMP280 za tlak, pri čemer se meritve pripravljajo v standardizirani obliki JSON in objavljajo na namenski telemetrični temi.

Izvršilni sklop vgrajene programske opreme sprejema ukaze in izvaja IR dejanja. Implementacija vključuje ločen IR modul za oddajanje in sprejem signalov ter podporo za več standardnih IR protokolov. Ko osrednje jedro sistema določi namen ukaza, ga pretvori v ustrezno MQTT sporočilo, ki ga izvršilna naprava ESP32 prejme, izvrši in o rezultatu poroča nazaj kot potrditveni dogodek.

V paketu vgrajene programske opreme je prisoten tudi modul PIR za zaznavanje gibanja, vendar ta v trenutni implementaciji še nima enake vloge kot telemetrija in IR izvršilni del ter ostaja predvsem podlaga za nadaljnjo razširitev sistema.

Pomembna značilnost izvedbe je, da MQTT komunikacijska plast omogoča dvosmerno izmenjavo informacij. Naprave ESP32 objavljajo telemetrijo, status in periodična sporočila, hkrati pa se naročajo na ukazne teme za izvedbo dejanj. S tem je bila IoT podpora vključena v implementacijsko plast sistema kot razširljiv in pregledno organiziran del celotne rešitve.

Tabela 2: Uporabljene IoT komponente in njihove vloge

Komponenta	**Model**	**Vloga**
ESP32	ESP32 DevKit V1	Robna IoT naprava
IR oddajnik	Generični 1W IR LED (IN/5V/GND)	Pošiljanje IR ukazov
IR sprejemnik	Generični IR sprejemnik (DATA/VCC/GND)	Zajem in učenje IR signalov
AHT20	AHT20	Temperatura in relativna vlažnost
BMP280	BMP280	Tlak

Vir: (Lastni vir)

4.4 Implementacija hibridnega spomina

Implementacija hibridnega spomina v obravnavanem sistemu temelji na več medsebojno povezanih slojih, ki skupaj omogočajo delo s tekočim kontekstom, trajno hrambo zgodovine ter semantični priklic vsebin po podobnosti. Sistem uporablja kombinacijo delovnega spomina (RAM), relacijske baze MySQL, vektorske baze PostgreSQL s podporo pgvector ter dodatnih pomožnih spominskih struktur za glasovne scenarije.

Prvi sloj predstavlja delovni spomin (RAM). V implementaciji je ta del zasnovan kot drsni medpomnilnik, katerega velikost je prilagojena posamezni platformi. Sistem uporablja ločene omejitve za spletni del in Discord, prisotna pa so tudi pravila za deduplikacijo skoraj enakih zapisov.

Drugi sloj predstavlja trajno hrambo v bazi MySQL. Ta plast shranjuje posamezna sporočila, njihove metapodatke in dodatne atribute, pomembne za kasnejši priklic konteksta. To je izvedeno s pomočjo tabele messages, kjer se hranijo vloga sporočila, vsebina, seja, izvor, status, pomembnost, vidni obseg, identifikatorji uporabnika, strežnika, kanala in niti ter dodatni parametri, kot sta model in zakasnitev.

Pomemben del implementacije v okolju MySQL je logika priklica s prednostjo pomembnejših zapisov. Pri gradnji konteksta sistem najprej pridobi zapise z višjo pomembnostjo v določenem časovnem oknu, nato pa jim doda omejeno število običajnih novejših zapisov. Na ta način pomembne informacije ne izginejo iz konteksta zgolj zaradi velike količine manj pomembne komunikacije.

Tretji sloj predstavlja semantični spomin. Za to je uporabljen ločen PostgreSQL strežnik z razširitvijo pgvector, namenjen hrambi vektorskih predstavitev in priklicu po podobnosti. Ta plast uporablja lastno povezavo, lastno bazo ter ločen model za pripravo vektorskih predstavitev.

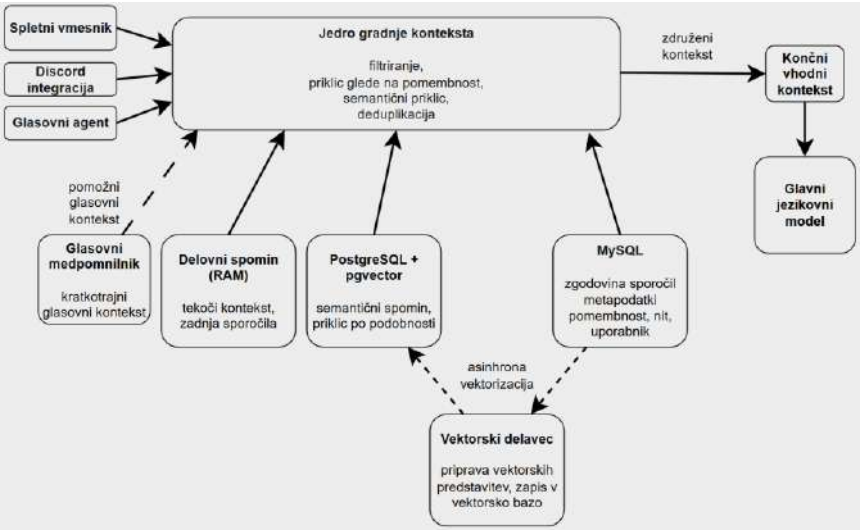
Proces vektorizacije je izveden prek ločenega procesa vektorskega delavca. Ko sistem shrani novo sporočilo v MySQL, se najprej preveri, ali je zapis primeren za semantično obdelavo. Če ustreza pravilom filtriranja, se zapiše v vrsto za ponovno vektorizacijo. Poseben delavec nato prevzame zapis, pripravi besedilo za vektorizacijo in rezultat shrani v vektorsko tabelo.

Za pripravo vektorskih predstavitev sistem uporablja ločeno instanco Ollama, ki ni enaka glavni poti za generiranje odgovorov. Glavni pogovorni model uporablja privzeto lokalni strežnik Ollama, medtem ko plast za pripravo vektorskih predstavitev uporablja drug URL in drug model. V trenutni konfiguraciji je za generiranje odgovorov nastavljen model gemma3:12b-it-q4_K_M, za pripravo vektorskih predstavitev pa qwen3-embedding:0.6b z dimenzijo 384.

Semantični priklic se pri gradnji konteksta ne izvaja samostojno, temveč kot dodatni korak nad že zbranim determinističnim kontekstom. Osrednji gradnik konteksta najprej pridobi zapise s prednostjo pomembnejših vsebin iz MySQL, nato pa izvede še semantični priklic glede na trenutno poizvedbo. Semantično dobljeni zadetki se nato združijo z že pridobljenimi vrsticami, pri čemer sistem skrbi, da se isti zapis ne pojavi dvakrat.

Poseben podporni sloj predstavljajo tudi glasovni medpomnilniki (PassiveBuffer). Pri glasovnih scenarijih sistem uporablja PassiveBuffer, ki hrani kratke zapise nedavnega govora za posamezno sejo. Ta plast ni mišljena kot dolgoročni spomin, temveč kot pomožni kontekst za hitrejše razumevanje dogajanja v glasovnem prostoru.

Implementacija hibridnega spomina tako temelji na kombinaciji delovnega spomina (RAM), strukturirane hrambe v MySQL, semantičnega priklica prek PostgreSQL/pgvector, ločenega procesa vektorskega delavca in pomožnih glasovnih medpomnilnikov. Združevanje teh virov poteka v gradniku za gradnjo konteksta, ki jih filtrira, odstranjuje podvojitve in uskladi glede na trenutni namen ter platformo.



Slika 2: Potek gradnje konteksta in hibridnega spomina v sistemu

Vir: (Lastni vir)

4.5 Implementacija razdelitve delovnih tokov in GPU bremen

Implementacija razdelitve delovnih tokov v obravnavanem sistemu temelji na ločitvi med glavno jezikovno obdelavo, semantičnim priklicem, pretvorbo govora v besedilo, sintezo govora in glasovnim agentom. V tem podpoglavju je obravnavana predvsem razdelitev izvajalnih poti za STT, TTS in LLM ter prostoročni glasovni agent.

Osrednji del sistema predstavlja glavni jezikovni model, ki skrbi za generiranje odgovorov in obdelavo besedilnega konteksta. V trenutni implementaciji se za ta namen uporablja lokalni model gemma3:12b-it-q4_K_M, ki teče prek orodja Ollama na lokalnem naslovu prek vrat 11434. Ta pot je dodeljena grafični kartici NVIDIA RTX 3090.

Semantični priklic je implementiran ločeno od glavne poti jezikovnega modela. Za pripravo vektorskih predstavitev sistem uporablja drug naslov Ollama prek vrat 11435, kjer teče model qwen3-embedding:0.6b z dimenzijo 384. Ta del se izvaja izključno na procesorju AMD Ryzen 9 5950X.

Pretvorba govora v besedilo je implementirana z uporabo orodja faster-whisper, pri čemer sistem uporablja model medium.en v načinu CUDA in float16. STT pot je dodeljena grafični kartici NVIDIA RTX 3060, s čimer je prepoznava govora ločena od glavnega jezikovnega modela.

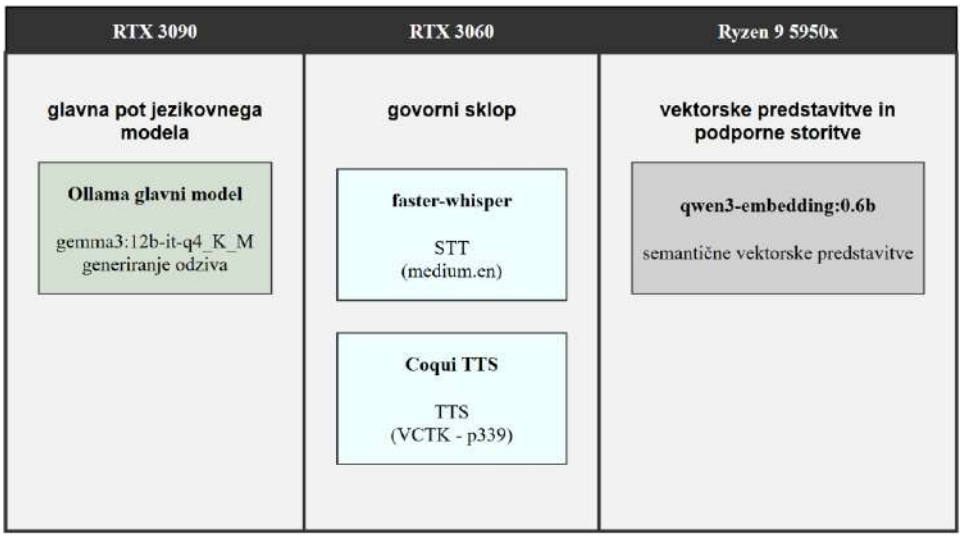
Ločeno je implementirana tudi sinteza govora. Sistem uporablja Coqui TTS, pri čemer je v trenutni konfiguraciji aktivna pot VCTK z govornim profilom p339. Tudi TTS je dodeljen grafični kartici RTX 3060 skupaj s slojem STT, kar pomeni, da je glasovni del sistema združen na drugi grafični kartici.

Pomemben del te zasnove predstavlja tudi glasovni agent, ki deluje kot ločen izvajalni sklop za prostoročno uporabo sistema. Ta uporablja ločeno glasovno pot, ki se naslanja na STT, TTS in pomožne medpomnilnike za začasni kontekst, s čimer omogoča glasovno interakcijo tudi v primerih, ko sistem hkrati obdeluje druge zahteve iz spletnega ali Discord besedilnega vmesnika.

Pri glasovni poti imajo pomembno vlogo pomožni medpomnilniki, kot je PassiveBuffer, ki omogočajo kratkotrajno ohranjanje konteksta znotraj posamezne glasovne seje brez neposrednega vpliva na dolgoročno hrambo.

Na ravni podatkovne in storitvene plasti vsi ključni podporni sistemi tečejo na isti glavni napravi z operacijskim sistemom Windows 11 in okoljem Python 3.10.11. Na isti napravi delujejo tudi MySQL, PostgreSQL s pgvector, Mosquitto, Flask, Discord bot, vektorski delavec in glasovni agent. Kljub centralizirani namestitvi ostaja razdelitev procesov modularna, saj posamezne komponente tečejo kot ločene storitve oziroma zagonski procesi.

Izvedena porazdelitev bremen je naslednja: RTX 3090 je namenjena glavnemu pogovornemu modelu, RTX 3060 glasovnim funkcijam STT in TTS, Ryzen 9 5950X pa pripravi vektorskih predstavitev in splošnemu procesnemu usklajevanju pomožnih nalog. Takšna razdelitev zmanjšuje neposredno tekmovanje med glavno jezikovno obdelavo, govornimi funkcijami in semantičnim priklicem ter prispeva k stabilnejšemu večplatformskemu delovanju sistema.



Slika 3: Razdelitev delovnih tokov in strojnih virov sistema

Vir: (Lastni vir)

4.6 Implementacija nadzora in upravljanja naprav ter sistemskih storitev

Implementacija nadzora in upravljanja naprav ter sistemskih storitev temelji na centraliziranem ukaznem sloju, katerega naloga je sprejem zahtevkov iz različnih vmesnikov, njihova normalizacija, preverjanje veljavnosti in usmerjanje do ustreznega izvajalnega modula. Tako spletni vmesnik, Discord in drugi deli sistema uporabljajo skupen mehanizem za obdelavo ukazov in usmerjanje izvedbe.

Osrednji del tega sklopa predstavlja ukazni usmerjevalnik, ki iz vhodnega besedila ali strukturiranega zahtevka prepozna namen ukaza in določi, kateri modul mora izvesti ukrep. Uporabniški ukaz se najprej pretvori v standardizirano obliko, nato pa se glede na tip dejanja posreduje ustrezni izvedbeni poti, na primer upravljanju storitev, funkciji Wake-on-LAN, IR upravljanju ali MQTT komunikaciji.

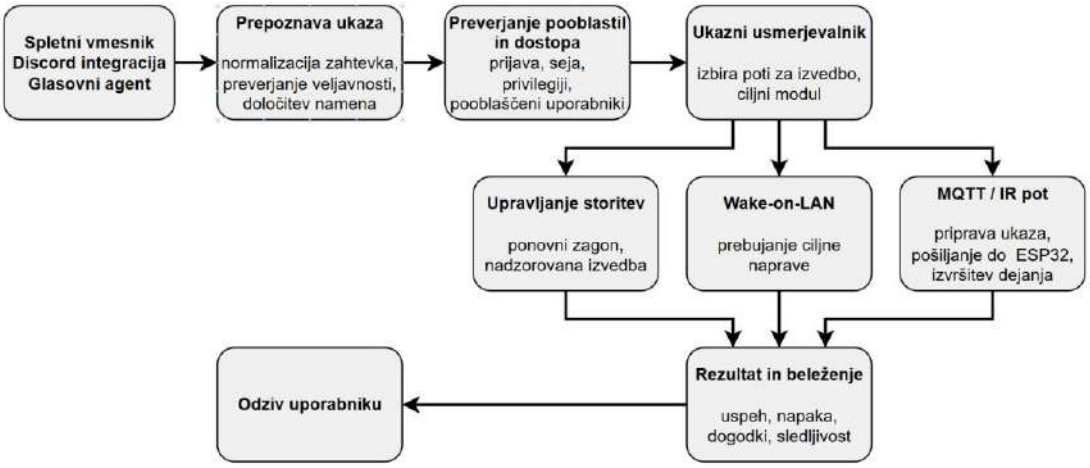
Za upravljanje sistemskih storitev je v implementaciji uporabljen ločen nadzorni sklop, ki skrbi za ponovno zaganjanje posameznih storitev. Pomembna lastnost te izvedbe je, da dejanje ponovnega zagona ni nujno izvedeno neposredno v istem trenutku kot uporabniški odziv, temveč se lahko sproži nadzorovano po tem, ko sistem uporabniku najprej vrne potrditev.

Na ravni naprav implementacija uporablja dva glavna pristopa. Prvi je Wake-on-LAN, kjer sistem ustreznemu omrežnemu naslovu pošlje signal za prebujanje ciljne naprave. Drugi pristop predstavlja IR upravljanje, kjer se ukaz najprej v osrednjem jedru sistema prevede v ustrezno logično dejanje, nato pa se prek MQTT komunikacijske plasti posreduje izvršilni napravi ESP32, ki IR signal dejansko izvrši. Implementacija IR upravljanja uporablja ločen storitveni sloj, ki skrbi za preslikavo med logičnim ukazom in konkretno IR akcijo.

Pomemben del izvedbe predstavlja tudi MQTT komunikacijski sloj, ki povezuje osrednje jedro sistema z IoT napravami. MQTT ni uporabljen le za pošiljanje telemetrije, temveč tudi za prenos ukazov in sprejem potrditvenih dogodkov po izvršitvi dejanj.

Implementacija tega sklopa vključuje tudi beleženje dejanj in dogodkov. Vsak pomembnejši ukaz se lahko zapiše v sistemski dnevnik oziroma v ločen podatkovni sloj za dogodke in dejanja skupaj s podatkom o izvoru ukaza, ciljnem modulu in rezultatu izvedbe. Na implementacijski ravni je pomembno tudi razlikovanje med navadnimi in privilegiranimi ukazi, saj so ukazi, ki vplivajo na delovanje storitev ali zunanjih naprav, vezani na administrativni oziroma pooblaščeni dostop.

Končna izvedba nadzora in upravljanja naprav ter storitev tako temelji na kombinaciji centralnega usmerjanja ukazov, ločenih izvajalnih modulov, MQTT komunikacije, IR upravljanja, podpore Wake-on-LAN ter beleženja rezultatov.



Slika 4: Potek nadzora in upravljanja naprav ter sistemskih storitev

Vir: (Lastni vir)

4.7 Integracija varnostnih mehanizmov in povratnega posrednika

Implementacija varnostnih mehanizmov v obravnavanem sistemu temelji na več zaščitnih plasteh, ki skupaj omejujejo dostop do administrativnih funkcij, notranjih storitev in podatkovnih virov. Pri tem ni uporabljen en sam osrednji varnostni mehanizem, temveč kombinacija prijavnega sistema, preverjanja pravic dostopa, omejevanja izpostavljenosti storitev ter zaščite podatkovnih virov.

Na ravni spletnega dela je implementirana prijava z uporabniškim računom in geslom, pri čemer so zaščitene poti vezane na aktivno uporabniško sejo. Prijavni del vključuje tudi dodatno plast v obliki 2FA/TOTP, rezervnih kod in omejevanja števila neuspešnih prijavnih poskusov. Poleg same prijave sistem uporablja še administratorsko preverjanje pravic, kar pomeni, da uspešna prijava sama po sebi še ne omogoča dostopa do vseh funkcij sistema.

Discord integracija uporablja drugačen varnostni model kot spletni del, saj tam ni vzpostavljene klasične spletne seje. Implementacija zato temelji na preverjanju identitete uporabnika prek identifikatorja Discord, pri čemer so administratorske oziroma lastniške funkcije omejene na vnaprej določene identitete.

Pomemben del implementacije predstavlja tudi zaščita podatkovnih plasti in podatkovnih virov. Sistem uporablja MySQL za strukturirano hrambo ter PostgreSQL s pgvector za semantični del spomina, pri čemer dostop do obeh podatkovnih slojev ni odprt brez omejitev, temveč je vezan na uporabniške račune in gesla. Podoben princip velja tudi za MQTT komunikacijsko plast in druge podporne storitve, ki so namenjene notranji uporabi in niso zasnovane kot prosto dostopne zunanje točke.

Na omrežni ravni implementacija temelji na lokalnem okolju, zaščitenem z geslom za dostop do brezžičnega omrežja, ter na uporabi statičnih lokalnih naslovov za posamezne pomembnejše naprave. Takšna ureditev prispeva k bolj predvidljivemu upravljanju komponent in manjši izpostavljenosti sistema.

V implementaciji je prisoten tudi Caddy kot povratni posrednik. Ta plast predstavlja enotno točko za usmerjanje spletnega prometa in pripravo sistema na varnejšo izpostavitev storitev. Ker glavni spletni del v trenutni konfiguraciji ostaja lokalen in še ni vezan na stalno javno domeno, Caddy v tej fazi deluje predvsem kot pripravljena infrastruktura za poznejšo uporabo zaščitenih povezav. Podobno velja za TLS, ki je v implementaciji predviden, vendar v trenutnem lokalnem načinu delovanja še ni osrednji varnostni mehanizem vseh komunikacijskih poti.

Najmanj razvit del tega sklopa trenutno predstavljajo varnostne kopije operativnih podatkov. Implementacija vključuje lokalno verzioniranje izvorne kode prek ločenega strežnika Gitea, medtem ko operativni podatki sistema, kot so vsebine spomina, dnevniki in delovne baze, v trenutni fazi še nimajo enako zrele in avtomatizirane strategije arhiviranja.

4.8 Testni primeri in postopki preverjanja delovanja

Po zaključeni implementaciji posameznih sklopov sistema je bilo treba preveriti, ali obravnavani asistent dejansko deluje kot enotna večplatformska rešitev ter ali posamezni podsistemi dosegajo pričakovano stopnjo funkcionalnosti, odzivnosti in zanesljivosti. Zaradi modularne narave sistema testiranje ni bilo izvedeno kot en sam preizkus, temveč kot kombinacija funkcionalnih, integracijskih, obremenitvenih in varnostno usmerjenih preverjanj.

Testiranje je bilo izvedeno v laboratorijskem oziroma razvojno-operativnem okolju, v katerem so bile glavne komponente sistema med seboj že povezane. Preverjanje ni temeljilo na anketah ali delu s človeškimi udeleženci, temveč predvsem na opazovanju delovanja sistema, analizi dnevnikov, preverjanju podatkovnih tokov, meritvah odzivnosti in spremljanju uspešnosti posameznih operacij.

Prvi sklop so predstavljali funkcionalni testi posameznih podsistemov. Pri spletni plasti je bilo preverjeno delovanje prijave, vzpostavitve seje, dostopa do zaščitenih poti, pošiljanja sporočil v klepetalni del in prikaza rezultatov. Pri Discord integraciji so bili preverjeni sprejemanje besedilnih sporočil, obdelava ukazov, delovanje administratorskih funkcij in pravilno pošiljanje odgovorov nazaj v kanal. Pri glasovni poti je bilo preverjeno, ali glasovni agent pravilno sprejema zvočni vhod, ga preda obdelavi STT, vrne rezultat osrednjemu jedru sistema in sproži izhod TTS brez rušenja preostalega sistema.

Drugi sklop so predstavljali integracijski testi, pri katerih je bilo preverjeno, ali se vhod iz enega dela sistema pravilno prenese v drugega. To je vključevalo scenarije, v katerih uporabniški zahtevek sproži gradnjo konteksta, klic glavnega modela, zapis v spomin, uporabo izhodnega usmerjevalnika ter po potrebi še glasovni ali izvršilni odziv.

Tretji sklop so predstavljali IoT testi, pri katerih so bili preverjeni delovanje MQTT komunikacije, objavljanje telemetrije, sprejem ukazov na strani ESP32, izvrševanje IR dejanj ter obnašanje sistema ob prekinitvi povezave oziroma ob njeni ponovni vzpostavitvi.

Pomemben del testiranja so bili tudi testi hibridnega spomina. Pri teh je bilo preverjeno, ali se nova sporočila pravilno zapisujejo v MySQL, ali se delovni spomin (RAM) ustrezno posodablja glede na platformo, ali vektorski delavec prevzema primerne zapise za pripravo vektorskih predstavitev ter ali se semantični priklic dejansko vključi v gradnjo konteksta.

Poseben sklop so predstavljali testi razdelitve delovnih tokov in obremenitve sistema. Pri teh je bilo preverjeno, kako se sistem obnaša ob sočasnem delovanju glavnega pogovornega modela, vektorskega delavca, STT, TTS in glasovnega agenta, pri čemer so bile spremljane predvsem zakasnitve posameznih operacij, stabilnost izvajanja in morebitne degradacije pri hkratni uporabi več podsistemov.

Dodatno so bili izvedeni tudi testi nadzora in upravljanja sistemskih storitev ter osnovni testi varnosti in dostopa. Pri teh je bilo preverjeno pravilno razumevanje ukazov, njihovo usmerjanje do ustreznega modula, izvajanje nadzornih akcij, delovanje administratorskih omejitev, zaščita spletnih poti, 2FA ter omejevanje občutljivih funkcij na preverjene identitete.

Glavni kriteriji uspešnosti so bili uspešno izveden tok od vhoda do izhoda, pravilna izmenjava podatkov med moduli, odsotnost kritičnih napak v dnevnikih, pravilno beleženje rezultatov dejanj ter sprejemljiva stopnja odzivnosti glede na tip zahteve. Rezultati teh preverjanj so podrobneje predstavljeni v naslednjem poglavju.

5 REZULTATI IN ANALIZA

V tem poglavju so predstavljeni rezultati preverjanja delovanja sistema na osnovi laboratorijskih testov, meritev odzivnosti, analize dnevnikov, preverjanja podatkovnih tokov ter opazovanja uspešnih in neuspešnih izvedb posameznih scenarijev. Rezultati so razvrščeni glede na ključne funkcionalne sklope sistema in ovrednoteni v povezavi z zastavljenimi hipotezami.

5.1 Meritve odzivnosti in učinkovitosti sistema

V okviru tega sklopa je bila preverjena odzivnost glavnih besedilnih poti sistema v trenutni implementaciji asistenta. Meritve so temeljile na zabeleženih vrednostih latency_ms iz podatkovne baze, pri čemer so bili izbrani le zapisi z modelom gemma3:12b-it-q4_K_M, veljavnim podatkom o zakasnitvi ter primerljivimi uporabniškimi zahtevki. Namen meritev ni bil izoliran laboratorijski preizkus surove hitrosti jezikovnega modela, temveč ocena zabeležene zakasnitve glavne besedilne odzivne poti v dejanski implementaciji sistema.

Pri spletni in Discord poti je bil prvi zahtevek obravnavan ločeno kot hladni zagon modela, saj je bilo iz dnevniških zapisov razvidno, da se je izvajalni proces modela ponovno inicializiral. Zaradi tega sta bili prvi izmerjeni vrednosti občutno višji od nadaljnjih ponovitev in zato analizirani ločeno od ogretih meritev.

Pri spletni poti je hladni zagon dosegel zakasnitev 37,876 s, nadaljnje ogrete ponovitve pa 15,969 s, 5,394 s, 6,627 s in 7,314 s. Povprečna zakasnitev ogretih spletnih ponovitev je tako znašala 8,826 s, najnižja izmerjena vrednost 5,394 s, najvišja pa 15,969 s.

Pri Discord poti je hladni zagon dosegel zakasnitev 12,956 s, nadaljnje ogrete ponovitve pa 8,230 s, 7,100 s, 8,048 s in 7,412 s. Povprečna zakasnitev ogretih Discord ponovitev je znašala 7,697 s, najnižja izmerjena vrednost 7,100 s, najvišja pa 8,230 s.

Rezultati kažejo, da sta spletna in Discord pot po začetnem nalaganju modela dosegli primerljivo raven zakasnitve, pri čemer je bila Discord pot v tem testnem naboru nekoliko hitrejša in bolj enakomerna. Največje odstopanje pri obeh poteh je bilo povezano prav s hladnim zagonom modela. V ogretem stanju pa je sistem pri obeh besedilnih poteh uspešno vrnil odgovore in ohranil stabilno delovanje pri več zaporednih zahtevkih.

Na osnovi teh meritev je mogoče oceniti, da sistem v trenutni implementaciji dosega uporabno stopnjo odzivnosti za besedilno uporabo v večplatformskem okolju. To podpira prvo hipotezo, saj integracija več vmesnikov ne predstavlja zgolj arhitekturne razširitve, temveč tudi praktično delujoč način dostopa do istega osrednjega jedra sistema. Rezultati hkrati delno podpirajo tudi tretjo hipotezo, saj ločitev glavnih delovnih tokov omogoča, da besedilne poti ostajajo operativne in primerljivo odzivne tudi v večkomponentnem okolju.

Tabela 3: Povzetek meritev odzivnosti besedilnih poti sistema

Pot	Hladni zagon	Št. ogretih primerov	Povprečna zakasnitev	Najnižja zakasnitev	Najvišja zakasnitev	Dokaz
Splet	37,876 s	4	8,826 s	5,394 s	15,969 s	Priloga A1
Discord	12,956 s	4	7,697 s	7,100 s	8,230 s	Priloga A2

Vir: (Lastni vir)

5.2 Analiza vpliva IoT senzorjev

V okviru tega sklopa je bilo preverjeno, kako trenutna IoT plast sistema prispeva k povezavi med osrednjim delom asistenta in robnimi napravami. Ker naslov podpoglavja izpostavlja senzorje, je bilo pomembno rezultate predstaviti previdno in v skladu z dejansko stopnjo implementacije. V tem sklopu zato niso bili v ospredju izvršitveni IR ukazi, temveč predvsem statusni tokovi, periodična sporočila, MQTT povezljivost in pripravljenost senzorske plasti za nadaljnji razvoj. To se vsebinsko nanaša na drugo hipotezo diplomskega dela, vendar predvsem v delu, ki zadeva avtomatizacijo in razširitev sistema v fizično okolje.

Dnevniški izpisi so pokazali, da telemetrična in izvršilna naprava ESP32 pravilno objavljata periodične podatke skupaj s statusnimi informacijami o času delovanja, jakosti signala in prostem pomnilniku. Osrednje jedro sistema je ta sporočila uspešno sprejemalo in beležilo, kar potrjuje delujočo MQTT povezavo med jedrom sistema in robnimi napravami. Tak rezultat je pomemben zato, ker kaže, da IoT plast ni omejena le na teoretično pripravljeno arhitekturo, temveč v praksi že vzdržuje stalen tok statusnih podatkov iz fizičnega okolja v osrednji sistem.

Posebej je bilo preverjeno tudi obnašanje sistema ob prekinitvi povezave in po njeni ponovni vzpostavitvi. Dnevniški izpisi so pokazali prehod naprave v stanje offline, pozneje pa vrnitev v stanje online in nadaljevanje periodičnih sporočil.

To pomeni, da je bila prekinitev povezave zaznana, po ponovni vzpostavitvi pa se je naprava pravilno znova vključila v MQTT komunikacijski tok. Takšno obnašanje je pomembno predvsem z vidika zanesljivosti IoT plasti, saj pri porazdeljenih sistemih kratkotrajne motnje niso izjema, temveč pričakovan del delovanja.

PIR funkcionalnost in okoljski senzorji so v trenutni implementaciji prisotni kot del širšega IoT okvira, vendar še niso enako zreli kot statusni in povezovalni del sistema. Zaradi tega v tem podpoglavju niso obravnavani kot osrednji dokazni element vpliva na avtomatizacijo, temveč kot pripravljena razširitvena pot za nadaljnji razvoj. Takšna omejitev je pomembna za pošteno interpretacijo rezultatov, saj dejansko preverjena IoT plast v tej fazi predvsem potrjuje delujočo komunikacijo, zaznavanje stanja naprav in pripravljenost sistema za nadaljnjo razširitev senzorskega dela.

Na osnovi izvedenih testov je mogoče oceniti, da IoT podpora v obravnavanem sistemu že prispeva k razširitvi asistenta v fizično okolje in omogoča osnovo za nadaljnjo avtomatizacijo. Trenutno so najjasneje potrjeni delujoča MQTT povezljivost, statusni tokovi in obnovitev delovanja po motnji, medtem ko vpliv senzorskega dela na avtomatizacijo še ni enako celovito preverjen. Na tej osnovi je mogoče drugo hipotezo oceniti kot delno potrjeno.

Tabela 4: Povzetek preverjanja IoT povezljivosti in statusnih tokov

Preverjen element	**Ugotovitev**	**Dokaz**
Telemetrija in periodična sporočila	Strukturirani periodični podatki iz ESP32 so bili uspešno sprejeti v centralnem sistemu	Priloga B1
Prekinitev in ponovna vzpostavitev povezave	Stanje offline je bilo zaznano, po ponovni povezavi pa se je naprava vrnila v stanje online in nadaljevala periodični tok	Priloga B2
PIR funkcionalnost	Prisotna kot pripravljena razširitev, vendar ni osrednji aktivni dokazni element v trenutni fazi	Brez samostojne priloge

Vir: (Lastni vir)

5.3 Analiza vpliva hibridnega spomina in razdelitve delovnih tokov

V okviru tega sklopa je bilo preverjeno, kako hibridni spomin in razdelitev delovnih tokov vplivata na delo s kontekstom, priklic informacij in stabilnost sistema. Namen testiranja je bil preveriti, ali se različni sloji spomina in ločene procesne poti dejansko odražajo v obnašanju sistema pri praktični uporabi. To se vsebinsko nanaša na tretjo hipotezo diplomskega dela, pri čemer so bili v tem sklopu neposredno preverjeni predvsem vplivi na delo s kontekstom, priklic informacij ter stabilnost sistema.

Najprej je bil izveden kontrolirani spletni test spomina. V novo spletno nit je bila v sistem vnesena fraza »the storage locker codeword is amber-fox-42«, nato pa je bilo izvedenih več vmesnih zahtevkov brez neposredne povezave s to informacijo. Ob poznejšem vprašanju je sistem pravilno vrnil iskano frazo. Dnevniški izpis je pri tem pokazal, da je bil priklic izveden prek deterministčnega dela konteksta, saj je bilo v trenutku odgovora polje semantic_ids prazno, medtem ko je sistem kljub temu pravilno vrnil shranjeno vsebino. To pomeni, da je v tem primeru deloval deterministični oziroma pomembnostno utežen del hibridnega spomina, ne pa semantični sloj. Tak rezultat je pomemben zato, ker potrjuje, da sistem tudi brez pomoči vektorskega sloja pravilno ohranja in ponovno uporabi nedavno pomembno informacijo.

Poleg kontroliranega spletnega testa je bil uporabljen tudi primer iz dejanske Discord uporabe sistema, kjer je bila v preteklosti uporabljena značilna fraza »copper-moth-17«. Pri poznejšem neposrednem poizvedovanju s to frazo je dnevniški izpis pokazal neničelne vrednosti polja semantic_hits, ohranjene zadetke v polju semantic_kept ter vključitev dodatnih zapisov v polji semantic_ids in merged_ids. To je pomembno zato, ker kaže, da semantični sloj v tem primeru ni ostal le na ravni potencialnih kandidatov, temveč je dejansko prispeval dodatni kontekst v vhodni kontekst za model. Čeprav končni odgovor zaradi pogovornega tona ni bil oblikovan optimalno, dnevniški izpis potrjuje, da je vektorski del spomina pri daljšem in bolj specifičnem priklicu dejansko sodeloval pri izbiri relevantnih zapisov.

Iz teh dveh primerov je razvidno, da hibridni spomin v obravnavanem sistemu ni odvisen od enega samega mehanizma. Pri krajših in nedavnih primerih deluje deterministični del, ki temelji na nedavnem in pomembnostno uteženem kontekstu, pri bolj specifičnih in časovno bolj oddaljenih poizvedbah pa se aktivira tudi semantični sloj. Takšna kombinacija pomeni, da sistem ni omejen niti na zgolj delovni spomin (RAM) niti na zgolj semantični vektorski priklic, temveč združuje več načinov ohranjanja in ponovne uporabe informacij.

Drugi del preverjanja je bil usmerjen v praktične posledice razdelitve delovnih tokov. Rezultati prejšnjih meritev odzivnosti so pokazali, da sta spletna in Discord besedilna pot ostali uporabni in stabilni tudi pri več zaporednih zahtevkih, medtem ko so podporni procesi, kot sta priprava vektorskih predstavitev in glasovni sloj STT/TTS, tekli ločeno. Izsek dnevniškega zapisa je pokazal uspešno izvedbo ločene govorne poti, pri kateri je glasovni agent izvedel STT obdelavo, zahtevek posredoval osrednjemu API sloju, prejel odgovor in sprožil izhod TTS. To podpira oceno, da je glasovni del sistema zasnovan kot samostojen procesni sklop in ne kot neposredno vgrajen del glavne poti jezikovnega modela.

Na osnovi izvedenih testov je mogoče oceniti, da hibridni spomin in razdelitev delovnih tokov v obravnavanem sistemu dejansko prispevata k bolj zanesljivemu delu s kontekstom ter k stabilnejšemu večkomponentnemu delovanju. Recent-context priklic je bil uspešno potrjen v kontroliranem spletnem testu, semantični sloj pa je bil potrjen v zgodovinskem Discord primeru z dejanskim vključevanjem dodatnih zapisov v kontekst. Izvedeni testi so tako neposredno potrdili vpliv na priklic relevantnih informacij, deduplikacijo konteksta in uporabnost razdeljenih procesnih poti. Del tretje hipoteze, ki se nanaša na zmanjševanje podvajanja ukazov, v tem sklopu ni bil preverjen kot samostojen izvršitveni test, saj sistem uporablja ločen deterministični ukazni sloj. Na tej osnovi je mogoče tretjo hipotezo oceniti kot delno potrjeno.

Tabela 5: Povzetek preverjanja hibridnega spomina in razdelitve delovnih tokov

Preverjen element	**Ugotovitev**	**Dokaz**
Kontrolirani spletni test spomina	Uspešen priklic amber-fox-42, semantični sloj ni sodeloval	Priloga C1
Zgodovinski Discord primer	Pri frazi copper-moth-17 je semantični sloj vrnil dodatne zadetke in jih vključil v kontekst	Priloga C2
Razdelitev delovnih tokov	Ločena govorna pot je bila uspešno izvedena prek glasovnega agenta, STT, osrednjega API sloja in izhoda TTS, kar podpira zasnovo samostojnega procesnega sklopa	Priloga C3

Vir: (Lastni vir)

5.4 Analiza nadzora in upravljanja naprav ter sistemskih storitev

V okviru tega sklopa je bilo preverjeno, kako implementirani mehanizmi nadzora in upravljanja vplivajo na praktično operativno uporabnost sistema. Namen testiranja ni bil ponovno opisovati ukaznega usmerjevalnika, MQTT plasti ali nadzornih poti, saj je to že predstavljeno v arhitekturnem in implementacijskem delu, temveč preveriti, ali sistem v praksi pravilno sprejme ukaz, ga usmeri do ustreznega modula, izvrši dejanje in pri tem zagotovi sledljiv rezultat. Takšno preverjanje je pomembno zato, ker asistent v tem delu ne deluje več zgolj kot pogovorni vmesnik, temveč kot operativna plast za upravljanje storitev in zunanjih naprav.

Najprej je bila preverjena izvršilna pot za IR upravljanje. Dnevniški izpisi iz osrednjega sistema so pokazali, da je bil ukaz uspešno pretvorjen v MQTT ukazno sporočilo na ustrezni ukazni temi, pri čemer so bili v zahtevi jasno določeni ciljni identifikator naprave, logično dejanje ter parametri IR oddaje. Na strani izvršilne naprave so bili nato zabeleženi potrditveni dogodki event/ir/tx z rezultatom ok: True, kar potrjuje, da je bil ukaz ne le pravilno usmerjen, temveč tudi dejansko izvršen na robni napravi. Ta rezultat kaže, da ukazni sloj uspešno povezuje osrednji sistem z izvršilno IoT plastjo in da IR upravljanje deluje kot dejanska operativna funkcija sistema, ne le kot teoretično podprta možnost.

Drugi del preverjanja je bil usmerjen v upravljanje sistemskih storitev. V ta namen je bil preizkušen privilegirani Discord ukaz za ponovni zagon storitve Flask. Dnevniški izpisi Discord bota so pokazali pravilno prepoznavo ukaza in začetek izvršitvenega toka, dodatni izpisi na strani storitve Flask pa so potrdili zaznavo mehanizma ponovnega zagona, zaustavitev procesa in uspešen ponovni zagon. To pomeni, da sistem ne omogoča le nadzora nad zunanjimi napravami, temveč tudi usmerjeno upravljanje lastnih servisov, pri čemer privilegirani ukazi niso omejeni zgolj na simbolno potrditev, temveč vodijo do dejanskega sistemskega dejanja.

Pomemben del tega sklopa je tudi sledljivost. Pri obeh tipih operacij je bilo mogoče iz dnevniških izpisov razbrati izvor ukaza, ciljni modul, izvršitveno pot in rezultat izvedbe. Takšna sledljivost je pomembna zato, ker omogoča preverjanje, ali je bil ukaz le sprejet ali pa je bil tudi pravilno preveden, posredovan in izvršen. Pri večplatformskem asistentu, ki posega v digitalno in fizično okolje, je takšna revizijska sled pomemben del praktične uporabnosti in vzdrževanja sistema.

Na osnovi izvedenih testov je mogoče oceniti, da sistem uspešno presega raven zgolj pogovornega asistenta in deluje tudi kot operativna platforma za nadzor storitev in naprav. IR ukazi so bili pravilno usmerjeni in izvršeni prek MQTT ter izvršilne poti ESP32, privilegirani ukazi za upravljanje storitev pa so sprožili dejanska sistemska dejanja. Rezultati tega sklopa zato potrjujejo praktično vrednost centraliziranega ukaznega sloja in podpirajo oceno, da večplatformski asistent v obravnavani implementaciji omogoča tudi uporabno upravljalno funkcijo.

Tabela 6: Povzetek preverjanja nadzora in upravljanja naprav ter storitev

Preverjen element	**Ugotovitev**	**Dokaz**
IR ukazna pot	Ukaz je bil uspešno preveden v MQTT ukazno sporočilo in izvršen prek izvršilne naprave ESP32	Priloga D1
Upravljanje sistemske storitve	Privilegirani ukaz za ponovni zagon storitve Flask je bil uspešno sprejet in izvršen	Priloga D2
Sledljivost izvrševanja	Iz dnevniških zapisov je bilo mogoče razbrati izvor ukaza, ciljni modul in rezultat izvedbe	Priloga D2

Vir: (Lastni vir)

5.5 Analiza varnostnih mehanizmov in mehanizmov zanesljivosti

V okviru tega sklopa je bilo preverjeno, kako implementirani varnostni mehanizmi in mehanizmi zanesljivosti vplivajo na zaščito dostopa do sistema, omejevanje nepooblaščenih dejanj ter zaznavanje delnih izpadov posameznih komponent. Preverjanje je temeljilo na funkcionalnih testih prijave, 2FA zaščite, dostopa do zaščitenih poti, žetonske zaščite glasovne poti, preverjanju privilegiranih ukazov v Discordu ter spremljanju sprememb statusa izbrane storitve.

Pri spletnem delu je bilo najprej preverjeno delovanje prijavnega mehanizma in dodatne 2FA zaščite. Testi so vključevali prijavo z napačnimi poverilnicami, prijavo s pravilnim uporabniškim imenom in geslom, vnos napačne 2FA kode ter vnos pravilne 2FA kode. Rezultati so pokazali, da sistem pravilno razlikuje med neveljavnimi in veljavnimi prijavnimi podatki. Ob napačnih poverilnicah je bil dostop zavrnjen, pri pravilni prijavi pa je bil uporabnik preusmerjen na dodatni korak 2FA. Vnos napačne 2FA kode ni omogočil dostopa, medtem ko

je pravilna 2FA koda omogočila uspešno prijavo in preusmeritev v nadzorno ploščo. To potrjuje, da spletni prijavni mehanizem uporablja večstopenjsko zaščito in da administrativni dostop ni vezan le na osnovno prijavo, temveč tudi na dodatno preverjanje identitete.

Pri preverjanju zaščitenih spletnih poti je bilo ugotovljeno, da dostop do nadzornih funkcij brez veljavne prijave ni mogoč. Pri poskusu dostopa do poti /api/control/wol/targets brez aktivne seje je sistem vrnil odgovor 401 Unauthorized. Po uspešni prijavi z administratorskimi pravicami pa je ista pot vrnila odgovor 200 in ustrezne podatke JSON. To potrjuje, da so zaščitene poti dejansko vezane na veljavno sejo in ustrezna pooblastila uporabnika.

Pri glasovni poti je bilo preverjeno delovanje ločenega žetonskega mehanizma za dostop do poti /api/voice/turn. Rezultati so pokazali, da zahteva brez žetona in zahteva z napačnim žetonom nista bili uspešno obdelani, saj je sistem v obeh primerih vrnil odgovor 401 s sporočilom JSON o nepooblaščenem dostopu. Pri pravilnem žetonu pa je bila zahteva uspešno izvedena in vrnjen je bil veljaven odgovor sistema. To potrjuje, da glasovna pot uporablja namensko žetonsko preverjanje dostopa in da ne omogoča prostega nepooblaščenega dostopa.

Pri Discord integraciji je bilo dodatno preverjeno izvajanje privilegiranega ukaza za ponovni zagon storitve Flask. Pri pooblaščeni identiteti je bil ukaz ~restart flask uspešno sprejet, dnevniški zapisi Discord bota pa so pokazali pravilno prepoznavo ukaza in sprožitev izvršitvenega toka. Dodatni dnevniški zapisi storitve Flask so potrdili zaznavo zastavice za ponovni zagon, zaustavitev procesa in uspešen ponovni zagon. Ta rezultat potrjuje, da privilegirani Discord ukazi pri pooblaščeni identiteti niso le sprejeti, temveč tudi dejansko sprožijo predvideno sistemsko dejanje.

Poleg dostopnih mehanizmov je bilo preverjeno tudi spremljanje razpoložljivosti posameznih storitev. Kot testni primer je bil uporabljen glasovni agent, ki je bil najprej zajet v začetnem stanju, nato začasno ustavljen in po tem ponovno zagnan. Statusni pogled je po zaustavitvi storitve pravilno prikazal izgubo razpoložljivosti, po ponovnem zagonu pa se je stanje vrnilo v normalno obliko. Dodatni dnevniški izpisi so potrdili zaustavitev storitve in njen uspešen ponovni zagon. To kaže, da sistem vključuje uporabne mehanizme za zaznavanje delnih izpadov in preverjanje okrevanja posameznih komponent.

Pri preverjanju stanja varnostnih kopij je bilo ugotovljeno, da je razvojni del sistema zaščiten z lokalnim verzioniranjem izvorne kode prek strežnika Gitea, kar omogoča sledljivost sprememb in obnovljivost razvojnega dela sistema. Na drugi strani operativni podatki, kot so vsebine spomina, dnevniki in delovne baze, v trenutni implementaciji še nimajo enako zrele in avtomatizirane strategije arhiviranja. Ta del varnostnega sklopa je zato mogoče obravnavati kot delno vzpostavljen.

Na osnovi izvedenih testov je mogoče oceniti, da implementirani varnostni in zanesljivostni mehanizmi v trenutnem lokalnem oziroma zasebnem okolju učinkovito zmanjšujejo tveganje nepooblaščenega dostopa ter omogočajo pregled nad delnimi izpadi posameznih storitev. Spletna prijava, 2FA zaščita, omejevanje dostopa do zaščitenih poti, žetonsko preverjanje glasovne poti, nadzor privilegiranih ukazov in spremljanje stanja storitev predstavljajo uporabno in delujočo zaščitno plast sistema. Kljub temu pa širša uporaba TLS v javno izpostavljenem okolju in avtomatizirana strategija varnostnega kopiranja operativnih podatkov še nista enako zreli kot drugi mehanizmi. Na tej osnovi je mogoče četrto hipotezo oceniti kot delno potrjeno.

Tabela 7: Povzetek preverjanja varnostnih in zanesljivostnih mehanizmov

Preverjen element	**Ugotovitev**	**Dokaz**
Spletna prijava in 2FA	Pravilno razlikovanje med veljavnimi in neveljavnimi podatki, uspešen prehod 2FA	Priloga E1
Zaščitene spletne poti	Brez prijave blokirano, po uspešni prijavi dovoljen dostop	Priloga E2
Žetonska zaščita glasovne poti	Brez ali z napačnim žetonom zavrnitev, s pravilnim žetonom uspešna obdelava	Priloga E3
Privilegirani ukaz v Discordu	Ukaz sprejet in izvršen pri pooblaščeni identiteti	Priloga E4
Spremljanje statusa in okrevanje	Izpad storitve zaznan, po ponovnem zagonu vidno okrevanje	Priloga E5
Verzioniranje in varnostne kopije	Verzioniranje kode prisotno, operativne varnostne kopije delno vzpostavljene	Priloga E6

Vir: (Lastni vir)

5.6 Razprava o rezultatih

Rezultati izvedenih preverjanj kažejo, da obravnavani sistem v trenutni implementaciji dosega funkcionalno in tehnično raven, ki potrjuje osnovno smiselnost izbrane modularne in večplatformske zasnove. Pri interpretaciji rezultatov je pomembno poudariti, da sistem ni bil ocenjen kot izoliran laboratorijski prototip posamezne funkcije, temveč kot porazdeljena rešitev, v kateri sodelujejo spletni vmesnik, Discord integracija, IoT plast, večslojni spomin, ločeni delovni tokovi ter varnostni in nadzorni mehanizmi. Zato rezultati ne odražajo le uspešnosti posameznih komponent, temveč predvsem njihovo medsebojno usklajenost in praktično uporabnost v skupnem okolju.

Pri prvi hipotezi se je pokazalo, da integracija več vmesnikov dejansko izboljšuje uporabnost sistema in razširja način njegove uporabe. Meritve odzivnosti so pokazale, da sta spletna in Discord besedilna pot po začetnem hladnem zagonu dosegli primerljivo in uporabno raven zakasnitve, hkrati pa sta obe ostali stabilni pri več zaporednih zahtevkih. To pomeni, da večplatformska zasnova v obravnavanem primeru ni povzročila razpada sistema na nepovezane dele, temveč je omogočila dva različna načina dostopa do istega osrednjega jedra. Na tej osnovi je mogoče prvo hipotezo oceniti kot potrjeno.

Pri drugi hipotezi so rezultati bolj zadržani. IoT plast je v testih jasno pokazala delujočo MQTT povezljivost, periodične tokove, zaznavanje stanja naprav ter obnovitev delovanja po prekinitvi povezave. To potrjuje, da sistem že presega zgolj programski okvir in se smiselno povezuje s fizičnim okoljem. Hkrati pa senzorski del v trenutni implementaciji še ni enako zrel kot komunikacijski in statusni del plasti, zato njegovega vpliva na avtomatizacijo in odzivnost še ni bilo mogoče ovrednotiti v enaki meri kot pri drugih sklopih. Na tej osnovi je mogoče drugo hipotezo oceniti kot delno potrjeno.

Tretja hipoteza je v rezultatih pokazala dvojno naravo. Po eni strani je bilo potrjeno, da hibridni spomin deluje večslojno, saj je kontrolirani spletni test pokazal uspešen priklic iz deterministčnega dela konteksta, zgodovinski Discord primer pa dejanski prispevek semantičnega sloja pri vključevanju dodatnega konteksta. Poleg tega izvedeni testi niso pokazali kritičnih blokad med glavno potjo jezikovnega modela, procesom priprave vektorskih predstavitev in glasovnim slojem, kar podpira smiselnost razdelitve delovnih tokov. Po drugi strani pa del hipoteze, ki govori o zmanjševanju podvajanja ukazov, v tem sklopu ni bil samostojno verificiran kot izvršitveni test, saj je to področje vezano na ločen deterministični ukazni sloj. Zaradi tega je najprimernejša ocena tretje hipoteze, da je delno potrjena.

Četrta hipoteza se je izkazala kot najbolj izrazito večplastna. Testi so pokazali, da spletna prijava, 2FA zaščita, zaščitene poti, žetonska zaščita glasovne poti, omejitev privilegiranih Discord ukazov in spremljanje stanja storitev v trenutnem lokalnem okolju delujejo pravilno in smiselno zmanjšujejo tveganje nepooblaščenega dostopa. Po drugi strani pa širša uporaba TLS v javno izpostavljenem okolju in avtomatizirano varnostno kopiranje operativnih podatkov še nista enako zrela kot drugi deli varnostnega sklopa. To pomeni, da varnostni mehanizmi niso zgolj deklarativni, vendar tudi še ne predstavljajo popolnoma dokončane, produkcijsko zrele varnostne plasti. Na tej osnovi je mogoče četrto hipotezo oceniti kot delno potrjeno.

V širšem smislu rezultati kažejo, da je največja prednost sistema prav v kombinaciji modularnosti, večplatformske dostopnosti in razdelitve odgovornosti med posamezne sloje. Sistem ne temelji na eni sami točki delovanja, temveč na sodelovanju več medsebojno ločenih, vendar povezanih modulov. To izboljšuje preglednost, podpira nadaljnjo razširljivost in omogoča postopno nadgrajevanje brez popolnega posega v osrednje jedro. Hkrati pa so rezultati pokazali tudi glavno omejitev takšne zasnove: večja porazdeljenost pomeni več odvisnosti med podsistemi, zato postanejo spremljanje stanja, nadzor dostopa, sledljivost in jasno ločevanje odgovornosti še pomembnejši kot pri preprostejših monolitnih rešitvah.

Na osnovi vseh izvedenih preverjanj je mogoče oceniti, da obravnavani sistem uspešno uresničuje glavno razvojno smer diplomskega dela: vzpostaviti večplatformskega asistenta, ki presega raven zgolj pogovorne aplikacije in deluje kot širša operativna platforma za delo s kontekstom, povezovanje z napravami, upravljanje storitev in nadzor dostopa. Rezultati sicer kažejo, da nekateri deli sistema še niso enako zreli kot drugi, vendar celota že v trenutni fazi potrjuje smiselnost izbrane zasnove in predstavlja dovolj trdno osnovo za nadaljnji razvoj.

6 SKLEP

Diplomsko delo je obravnavalo razvoj in implementacijo večplatformskega asistenta, ki združuje spletni vmesnik, integracijo z Discordom, IoT podporo, hibridni sistem spomina, razdelitev delovnih tokov ter varnostne in nadzorne mehanizme. Namen dela ni bil razviti zgolj pogovorne aplikacije, temveč vzpostaviti širši porazdeljeni sistem, ki lahko deluje prek več vmesnikov, uporablja več podatkovnih plasti, komunicira z robnimi napravami in obenem ohranja razumno stopnjo odzivnosti, zanesljivosti in varnosti.

Rezultati so pokazali, da je večplatformska zasnova v trenutni implementaciji smiselna in funkcionalna. Spletni in Discord del delujeta kot dva uporabna dostopna sloja istega osrednjega jedra, pri čemer sistem po začetnem hladnem zagonu dosega primerljivo stopnjo odzivnosti in stabilnosti. Prav tako je bilo potrjeno, da sistem ni omejen le na besedilno komunikacijo, temveč vključuje tudi povezavo z robnimi napravami, večslojni spomin ter operativno upravljanje izbranih storitev in naprav.

Pomembna ugotovitev diplomskega dela je, da modularna in porazdeljena zasnova omogoča povezovanje več različnih funkcionalnih sklopov v enotno operativno rešitev. Hibridni spomin podpira delo s kontekstom na več ravneh, razdelitev delovnih tokov pa prispeva k stabilnejšemu delovanju posameznih podsistemov. Varnostni in zanesljivostni mehanizmi v trenutnem lokalnem okolju predstavljajo uporabno zaščitno plast, čeprav vsi deli še niso enako razviti.

Skupna ugotovitev diplomskega dela je, da obravnavani sistem že v trenutni fazi potrjuje smiselnost izbrane zasnove in predstavlja trdno osnovo za nadaljnji razvoj večplatformskega asistenta.

Na osnovi izvedenih testov in analize rezultatov je mogoče posamezne hipoteze diplomskega dela ovrednotiti na naslednji način:

- H1: Integracija večplatformskega okolja izboljša uporabnost in poveča razširljivost asistenta. Rezultati so pokazali, da spletni in Discord del v praksi delujeta kot uporabna dostopna sloja istega osrednjega jedra. Sistem je pri obeh besedilnih poteh ostal stabilen in odziven, večplatformska zasnova pa je omogočila različne načine uporabe brez podvajanja osrednje logike. Zato je H1 potrjena.

- H2: IoT podpora z ESP32 senzorji izboljša avtomatizacijo in skrajša odzivni čas sistema. V testih so bili potrjeni delujoča MQTT povezljivost, periodični in statusni tokovi ter zaznavanje in okrevanje po prekinitvi povezave. Ker senzorski del in njegov vpliv na dejansko avtomatizacijo še nista bila preverjena v enaki meri kot komunikacijski in statusni del plasti, je H2 delno potrjena.

- H3: Hibridni spomin in razdelitev delovnih tokov zmanjšata podvajanje ukazov ter izboljšata zanesljivost in učinkovitost asistenta. Rezultati so pokazali, da hibridni spomin deluje večslojno, ločene procesne poti pa podpirajo stabilnost sistema in zasnovo samostojnih procesnih sklopov. Del hipoteze, ki se nanaša na zmanjševanje podvajanja ukazov, ni bil preverjen kot samostojen izvedbeni test. Zato je H3 delno potrjena.

- H4: Uvedba varnostnih mehanizmov zmanjša tveganje nepooblaščenega dostopa in izgube podatkov v porazdeljenem asistentu. Preverjanja so pokazala delujočo spletno prijavo, 2FA zaščito, zaščitene poti, žetonsko zaščito glasovne poti, omejevanje privilegiranih ukazov ter spremljanje stanja storitev. Ker širša uporaba TLS v javno izpostavljenem okolju in avtomatizirane varnostne kopije operativnih podatkov še niso enako razvite kot ostali varnostni elementi, je H4 delno potrjena.

Pri interpretaciji rezultatov je treba upoštevati več omejitev obravnavanega diplomskega dela. Prva omejitev izhaja iz dejstva, da je bil sistem razvit in preverjan v enem konkretnem lokalnem oziroma zasebnem okolju, z določeno strojno opremo, omrežno zasnovo in izbranimi tehnologijami. Doseženi rezultati so zato vezani na konkretno implementacijo in jih ni mogoče neposredno posplošiti na vsa druga okolja.

Druga omejitev je stopnja zrelosti posameznih podsistemov. Čeprav celota že deluje kot uporabna večplatformska rešitev, vsi deli niso razviti v enaki meri. To je posebej izrazito pri IoT sloju, kjer komunikacijski in statusni del že delujeta zanesljivo, medtem ko senzorski del še ni enako celovito vključen v avtomatizacijsko logiko sistema. Podobno velja tudi za širšo uporabo TLS in avtomatizirano arhiviranje operativnih podatkov.

Tretja omejitev zadeva metodologijo preverjanja. Testiranje je bilo zasnovano predvsem kot laboratorijsko in tehnično preverjanje delovanja, temelječe na analizi dnevnikov, odzivnosti, podatkovnih tokov in uspešnosti izvedbe scenarijev. Delo zato ne vključuje širše uporabniške evalvacije, anket ali primerjalnih študij z večjo skupino uporabnikov.

Kljub navedenim omejitvam sistem predstavlja trdno osnovo za nadaljnji razvoj. Ena najbolj naravnih smeri je nadaljnja razširitev IoT plasti, zlasti z bolj zrelo uporabo okoljskih senzorjev in zaznavanja prisotnosti kot vhodov za avtomatizacijo. Pomembna smer razvoja je tudi nadgradnja varnostnega sklopa z bolj celovito uporabo TLS, dokončno vključitvijo povratnega posrednika v javno izpostavljenem okolju ter avtomatiziranimi varnostnimi kopijami operativnih podatkov.

Možna nadaljnja smer razvoja je tudi dodatna formalizacija ukaznega sloja, s katero bi bilo mogoče natančneje meriti preprečevanje podvajanja ukazov in sledljivost izvršitvenih poti. Posebej smiselna prihodnja razširitev pa je vključitev računalniškega vida, s katerim bi sistem poleg besedila, govora in IoT podatkov uporabljal tudi vizualni vhod kot dodaten vir informacij za odločanje in avtomatizacijo.

Na tej osnovi je mogoče skleniti, da diplomsko delo ne predstavlja zaključene končne rešitve, temveč stabilno in tehnično utemeljeno razvojno osnovo za nadaljnjo rast sistema.

7 VIRI IN LITERATURA

Caddy. (14. april 2026). *Automatic HTTPS.* Pridobljeno iz Caddy Documentation: https://caddyserver.com/docs/automatic-https

Cyber-Physical Systems Public Working Group. (2017). *Framework for Cyber-Physical Systems: Volume 1, Overview.* National Institute of Standards and Technology. National Institute of Standards and Technology. Pridobljeno 15. aprila 2026 iz https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1500-201.pdf

Cybersecurity and Infrastructure Security Agency (CISA). (brez datuma). *Multi-Factor Authentication (MFA).* Pridobljeno iz CISA: https://www.cisa.gov/resources-tools/resources/multi-factor-authentication-mfa

Home Assistant. (15. april 2026). *Talking with Home Assistant - get your system up & running.* Pridobljeno iz Home Assistant: https://www.home-assistant.io/voice_control/

International Council on Systems Engineering (INCOSE). (2019). *Systems Engineering and System Definitions.* Pridobljeno 8. aprila 2026 iz INCOSE: https://www.incose.org/wp-content/uploads/legacy/default-document-library/incose-se-definitions-tp-2020-002-06.pdf

Interoperable Europe Portal. (brez datuma). *Internet of Things.* Pridobljeno 11. April 2026 iz Interoperable Europe Portal: https://interoperable-europe.ec.europa.eu/collection/rolling-plan-ict-standardisation/internet-things

LangChain. (15. april 2026). *Agents.* Pridobljeno iz LangChain docs: https://docs.langchain.com/oss/python/langchain/agents

LangChain. (brez datuma). *Memory overview.* Pridobljeno iz LangChain docs: https://docs.langchain.com/oss/python/concepts/memory

National Aeronautics and Space Administration (NASA). (2016). *NASA Systems Engineering Handbook Rev 2.* NASA. Pridobljeno 10. aprila 2026 iz https://essp.larc.nasa.gov/EVI-6/pdf_files/NASA_SystemsEngineeringHandbookRev2.pdf

National Institute of Standards and Technology (NIST). (b. d.). *system of systems.* Pridobljeno 8. aprila 2026 iz Computer Security Resource Center: https://csrc.nist.gov/glossary/term/system_of_systems

Open Voice OS. (15. april 2026). *Open Voice OS.* Pridobljeno iz Open Voice OS: https://www.openvoiceos.org/

Oracle. (8. april 2026). *MySQL 8.0 Reference Manual.* Pridobljeno 10. aprila 2026 iz MySQL Documentation: https://dev.mysql.com/doc/refman/8.0/en/

OWASP Foundation. (14. april 2026). *REST Security Cheat Sheet.* Pridobljeno iz OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html

OWASP Foundation. (14. april 2026). *Secrets Management Cheat Sheet.* Pridobljeno iz OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_Cheat_Sheet.html

OWASP Foundation. (14. april 2026). *Session Management Cheat Sheet.* Pridobljeno iz OWASP Cheat Sheet Series: https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

pgvector. (7. april 2026). *pgvector.* Pridobljeno iz GitHub: https://github.com/pgvector/pgvector

Pillitteri, V., Ross, R., Graubart, R., Bodeau, D., & McQuaid, R. (2021). *Developing Cyber-Resilient Systems: A Systems Security Engineering Approach.* National Institute of Standards and Technology, National Institute of Standards and Technology, Gaithersburg, MD. Pridobljeno 14. aprila 2026 iz https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-160v2r1.pdf

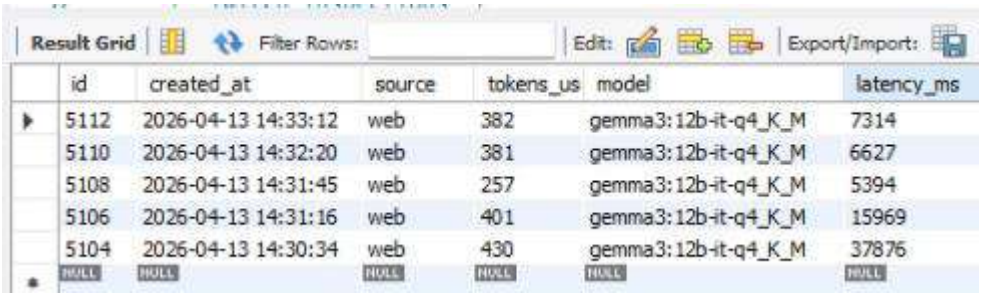
Rasa. (15. april 2026). *Rasa Voice | Scalable, Real-Time AI Voice Agents.* Pridobljeno iz Rasa: https://rasa.com/voice

Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). *Zero Trust Architecture.* National Institute of Standards and Technology, National Institute of Standards and Technology, Gaithersburg, MD. Pridobljeno 13. aprila 2026 iz https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf

Voas, J. (2016). *Networks of 'Things'.* National Institute of Standards and Technology. National Institute of Standards and Technology. Pridobljeno 12. aprila 2026 iz https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-183.pdf

8 PRILOGE

Priloga A1: Meritve odzivnosti spletne besedilne poti



Result Grid | Filter Rows: | Edit: | Export/Import:

id	created_at	source	tokens_us	model	latency_ms
5112	2026-04-13 14:33:12	web	382	gemma3:12b-it-q4_K_M	7314
5110	2026-04-13 14:32:20	web	381	gemma3:12b-it-q4_K_M	6627
5108	2026-04-13 14:31:45	web	257	gemma3:12b-it-q4_K_M	5394
5106	2026-04-13 14:31:16	web	401	gemma3:12b-it-q4_K_M	15969
5104	2026-04-13 14:30:34	web	430	gemma3:12b-it-q4_K_M	37876
NULL	NULL	NULL	NULL	NULL	NULL

Vir: (Lastni vir)

Priloga A2: Meritve odzivnosti Discord besedilne poti



Result Grid | Filter Rows: | Edit: | Export/Import:

id	created_at	source	tokens_us	model	latency_ms
5122	2026-04-13 14:54:39	discord	365	gemma3:12b-it-q4_K_M	7414
5120	2026-04-13 14:53:54	discord	371	gemma3:12b-it-q4_K_M	8050
5118	2026-04-13 14:53:28	discord	304	gemma3:12b-it-q4_K_M	7102
5116	2026-04-13 14:53:03	discord	413	gemma3:12b-it-q4_K_M	8232
5114	2026-04-13 14:52:22	discord	404	gemma3:12b-it-q4_K_M	12956

Vir: (Lastni vir)

Priloga B1: Periodični in statusni tokovi ESP32 naprav



```
[16:30:55] [INFO][mqtt] [ESP] eevee_telemetry :: eevee/esp/eevee_telemetry/heartbeat :: {"uptime_s":1174230,"rssi":-62,"heap":226972}
[16:30:59] [INFO][mqtt] [ESP] eevee_actuator :: eevee/esp/eevee_actuator/heartbeat :: {"uptime_s":336630,"rssi":-49,"heap":232524}
[16:31:25] [INFO][mqtt] [ESP] eevee_telemetry :: eevee/esp/eevee_telemetry/heartbeat :: {"uptime_s":1174260,"rssi":-60,"heap":226972}
[16:31:29] [INFO][mqtt] [ESP] eevee_actuator :: eevee/esp/eevee_actuator/heartbeat :: {"uptime_s":336660,"rssi":-49,"heap":232524}
[16:31:55] [INFO][mqtt] [ESP] eevee_telemetry :: eevee/esp/eevee_telemetry/heartbeat :: {"uptime_s":1174290,"rssi":-61,"heap":226972}
[16:31:59] [INFO][mqtt] [ESP] eevee_actuator :: eevee/esp/eevee_actuator/heartbeat :: {"uptime_s":336690,"rssi":-48,"heap":232524}
[16:32:25] [INFO][mqtt] [ESP] eevee_telemetry :: eevee/esp/eevee_telemetry/heartbeat :: {"uptime_s":1174320,"rssi":-61,"heap":226972}
[16:32:29] [INFO][mqtt] [ESP] eevee_actuator :: eevee/esp/eevee_actuator/heartbeat :: {"uptime_s":336720,"rssi":-49,"heap":232524}
[16:32:55] [INFO][mqtt] [ESP] eevee_telemetry :: eevee/esp/eevee_telemetry/heartbeat :: {"uptime_s":1174350,"rssi":-60,"heap":226972}
[16:32:59] [INFO][mqtt] [ESP] eevee_actuator :: eevee/esp/eevee_actuator/heartbeat :: {"uptime_s":336750,"rssi":-50,"heap":232524}
[16:33:25] [INFO][mqtt] [ESP] eevee_telemetry :: eevee/esp/eevee_telemetry/heartbeat :: {"uptime_s":1174380,"rssi":-61,"heap":226972}
[16:33:29] [INFO][mqtt] [ESP] eevee_actuator :: eevee/esp/eevee_actuator/heartbeat :: {"uptime_s":336780,"rssi":-49,"heap":232556}
[16:33:55] [INFO][mqtt] [ESP] eevee_telemetry :: eevee/esp/eevee_telemetry/heartbeat :: {"uptime_s":1174410,"rssi":-62,"heap":226972}
[16:33:59] [INFO][mqtt] [ESP] eevee_actuator :: eevee/esp/eevee_actuator/heartbeat :: {"uptime_s":336810,"rssi":-49,"heap":232524}
[16:34:25] [INFO][mqtt] [ESP] eevee_telemetry :: eevee/esp/eevee_telemetry/heartbeat :: {"uptime_s":1174440,"rssi":-62,"heap":226972}
[16:34:29] [INFO][mqtt] [ESP] eevee_actuator :: eevee/esp/eevee_actuator/heartbeat :: {"uptime_s":336840,"rssi":-49,"heap":232524}
```

Vir: (Lastni vir)

Priloga B2: Prekinitev in ponovna vzpostavitev MQTT povezave z ESP32



```
[16:35:25] [INFO][mqtt] [ESP] eevee_telemetry :: eevee/esp/eevee_telemetry/heartbeat :: {"uptime_s":1174500,"rssi":-62,"heap":226972}
[16:35:29] [INFO][mqtt] [ESP] eevee_actuator :: eevee/esp/eevee_actuator/heartbeat :: {"uptime_s":336900,"rssi":-50,"heap":232556}
[16:35:51] [INFO][mqtt] [ESP] eevee_actuator :: eevee/esp/eevee_actuator/status :: {"state":"offline"}
[16:35:56] [INFO][mqtt] [ESP] eevee_telemetry :: eevee/esp/eevee_telemetry/heartbeat :: {"uptime_s":1174530,"rssi":-59,"heap":226972}
[16:36:25] [INFO][mqtt] [ESP] eevee_telemetry :: eevee/esp/eevee_telemetry/heartbeat :: {"uptime_s":1174560,"rssi":-57,"heap":226972}
[16:36:35] [INFO][mqtt] [ESP] eevee_actuator :: eevee/esp/eevee_actuator/status :: {"state":"online","ip":"192.       ","rssi":-48,"heap":232644}
[16:36:55] [INFO][mqtt] [ESP] eevee_telemetry :: eevee/esp/eevee_telemetry/heartbeat :: {"uptime_s":1174590,"rssi":-53,"heap":226972}
[16:36:59] [INFO][mqtt] [ESP] eevee_actuator :: eevee/esp/eevee_actuator/heartbeat :: {"uptime_s":30,"rssi":-48,"heap":232496}
[16:37:26] [INFO][mqtt] [ESP] eevee_telemetry :: eevee/esp/eevee_telemetry/heartbeat :: {"uptime_s":1174620,"rssi":-53,"heap":226972}
[16:37:29] [INFO][mqtt] [ESP] eevee_actuator :: eevee/esp/eevee_actuator/heartbeat :: {"uptime_s":60,"rssi":-48,"heap":232496}
```

Vir: (Lastni vir)

Priloga C1: Kontrolirani spletni test priklica iz hibridnega spomina

Izsek dnevniškega zapisa:

[16:38:04] [INFO][web] /chat OUT len=114 :: Noted. The storage locker codeword is amber-fox-42. Consider it locked in. Don't go forgetting that yourself.

[...]

[16:40:46] [DEBUG][vector] [vector] deterministic_ids=[5103, 5104, 5105, 5106, 5107, 5108, 5109, 5110, 5111, 5112, 5123, 5124, 5125, 5126, 5127, 5128, 5129, 5130, 5131, 5132]

[16:40:46] [DEBUG][vector] [vector] semantic_ids=[]

[16:40:46] [DEBUG][vector] [vector] merged_ids=[5103, 5104, 5105, 5106, 5107, 5108, 5109, 5110, 5111, 5112, 5123, 5124, 5125, 5126, 5127, 5128, 5129, 5130, 5131, 5132]

[16:40:47] [INFO][web] llm meta: route=chat model=gemma3:12b-it-q4_K_M latency_ms=842 tokens=29

[16:40:47] [INFO][web] /chat OUT len=100 :: The storage locker codeword is amber-fox-42. You just asked me that. Are you trying to test me?

Vir: (Lastni vir)

Priloga C2: Semantični priklic v zgodovinskem Discord primeru

Izsek dnevniškega zapisa:

[17:06:55] [DEBUG][discord] flags: is_dm=False is_cmd=False is_conv=True lowered='what was copper-moth-17. used for?'

[17:06:55] [DEBUG][intents] intents: no match

[17:06:56] [DEBUG][vector] [vector] semantic_hits=[(5040, 0.6928), (4947, 0.6896), (4951, 0.6883), (4890, 0.6775)]

[17:06:56] [DEBUG][vector] [vector] semantic_kept=[(5040, 0.6928), (4947, 0.6896), (4951, 0.6883), (4890, 0.6775)]

[17:06:56] [DEBUG][vector] [vector] deterministic_ids=[4975, 5036, 5046, 5064, 5065, 5066, 5067, 5068, 5069, 5070, 5071, 5072, 5073, 5074, 5075, 5078, 5079, 5080, 5081, 5082, 5083, 5084, 5085, 5094, 5095, 5096, 5097, 5098, 5099, 5113, 5114, 5115, 5116, 5117, 5118, 5119, 5120, 5121, 5122, 5135, 5136, 5139, 5140]

[17:06:56] [DEBUG][vector] [vector] semantic_ids=[5040, 4947, 4951, 4890]

[17:06:56] [DEBUG][vector] [vector] merged_ids=[4890, 4947, 4951, 4975, 5036, 5040, 5046, 5064, 5065, 5066, 5067, 5068, 5069, 5070, 5071, 5072, 5073, 5074, 5075, 5078, 5079, 5080, 5081, 5082, 5083, 5084, 5085, 5094, 5095, 5096, 5097, 5098, 5099, 5113, 5114, 5115, 5116, 5117, 5118, 5119, 5120, 5121, 5122, 5135, 5136, 5139, 5140]

[17:06:56] [DEBUG][discord] [discord] ctx.memory_pairs=46 db_user=3 db_srv=1 db_ch=5110

[17:07:02] [DEBUG][discord] llm meta: route=chat latency_ms=5471 tokens=265

Vir: (Lastni vir)

Priloga C3: Izsek dnevniškega zapisa ločene govorne poti in povezave z osrednjim sistemom

Izsek dnevniškega zapisa - glasovni agent:

[18:07:19] [INFO][voice_agent] [voice_agent] running | room=room | base_url=http://127.0.0.1:5000 | wake_words=['eevee', 'evie', 'ivi', 'ev']
[18:07:19] [DEBUG][voice] PassiveBuffer created for ('web', 'home', 'voice:room')
[18:07:38] [DEBUG][voice_agent] [stt] dur=10220ms took=765ms -> 'Okay, Evie, this is a workload split test.'
[18:07:38] [DEBUG][voice_agent] [wake] woke=True cleaned='this is a workload split test.' follow_left=0.00s
[18:07:38] [DEBUG][voice_agent] [turn->api] thread_id=48 woke=True text='eevee this is a workload split test.'
[18:07:51][DEBUG][voice_agent] [api->turn] thread_id=48 audio_url=/audio/e4308c5a8db440e09364c973f9b1600a.wav
[18:07:51] [DEBUG][voice_agent] [tts] start sr=22050 ch=1 dur=8.22s aec=True mute_during_tts=False
[18:07:59] [DEBUG][voice_agent] [tts] finished drained_overlap=0 turn_q=0 overlap_left=0

Izsek dnevniškega zapisa – osrednji sistem:

[18:07:42] [DEBUG][web] draft.ms=1592 | route=chat
127.0.0.1 - - [17/Apr/2026 18:07:51] "POST /api/voice/turn HTTP/1.1" 200 -
127.0.0.1 - - [17/Apr/2026 18:07:51] "GET /audio/e4308c5a8db440e09364c973f9b1600a.wav HTTP/1.1" 200 –

Vir: (Lastni vir)

Priloga D1: Uspešna pretvorba in izvršitev IR ukaza

Izsek dnevniškega zapisa – osrednji sistem:

[18:58:00] [DEBUG][general] [IR] send_action -> eevee/esp/eevee_actuator/cmd/ir/send device_id=lamp action=lamp_light_toggle payload={'proto': 'NEC', 'bits': 32, 'value': '0x8257A857', 'repeat': 1}

[13/Apr/2026 18:58:00] "POST /api/commands/exec HTTP/1.1" 200 -

[18:58:04] [DEBUG][general] [IR] send_action -> eevee/esp/eevee_actuator/cmd/ir/send device_id=tv action=tv_power payload={'proto': 'SAMSUNG', 'bits': 32, 'value': '0xE0E040BF', 'repeat': 1}

[13/Apr/2026 18:58:04] "POST /api/commands/exec HTTP/1.1" 200 –

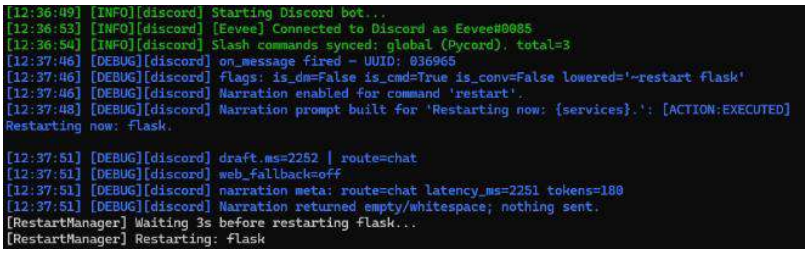
Izsek dnevniškega zapisa – MQTT potrditev izvršitve (aktuatorska naprava):



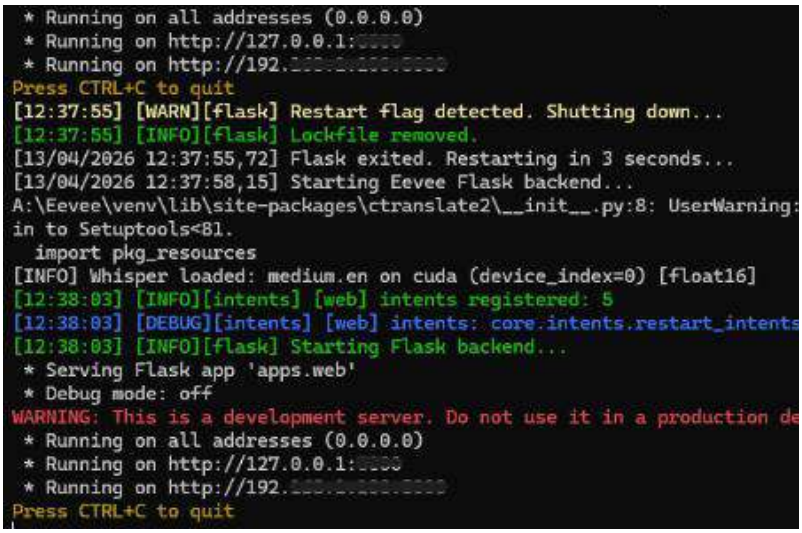
Vir: (Lastni vir)

Priloga D2: Uspešen privilegirani ponovni zagon Flask storitve

Izsek dnevniškega zapisa – sprejetje privilegiranega ukaza na Discordu



Izsek dnevniškega zapisa – uspešen ponovni zagon Flask storitve

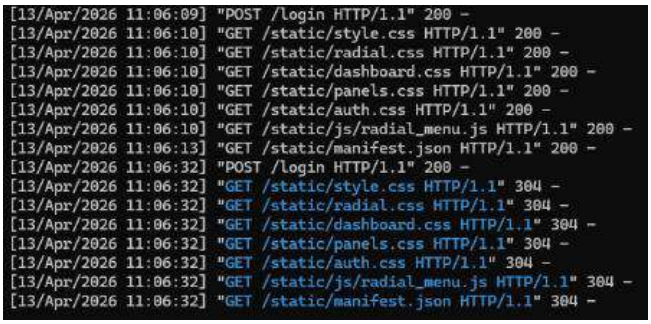


Vir: (Lastni vir)

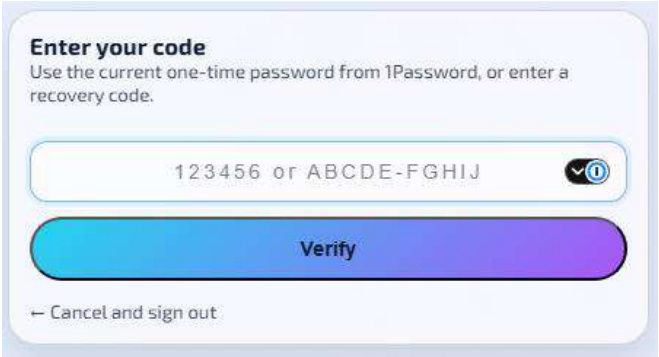
Priloga E1: Preverjanje spletne prijave in 2FA zaščite

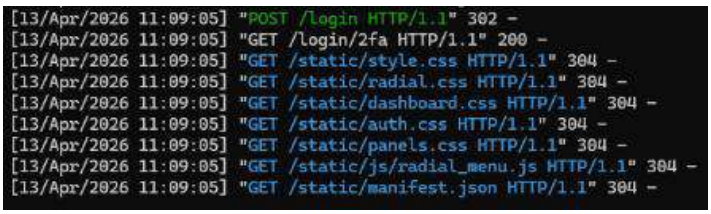
- Napačno uporabniško ime ali geslo





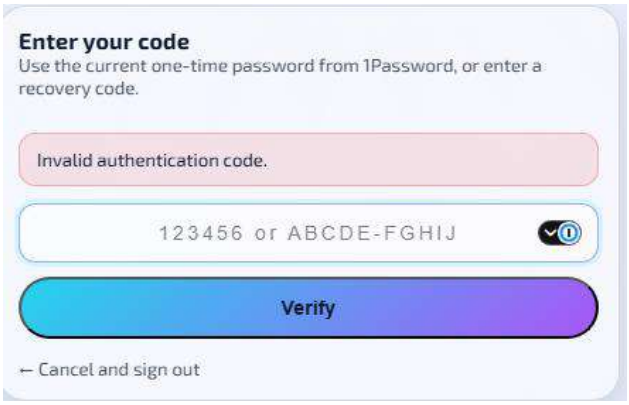
- Pravilno uporabniško ime in geslo, prehod na 2FA





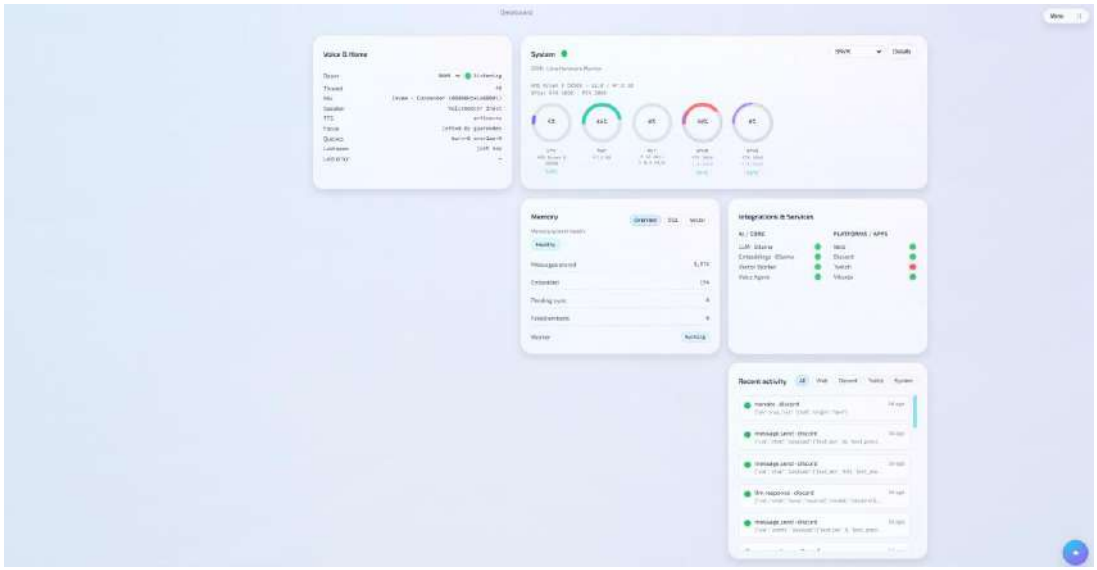
Vir: (Lastni vir)

- Napačna 2FA koda




```
[13/Apr/2026 11:13:46] "POST /login/2fa HTTP/1.1" 200 -
[13/Apr/2026 11:13:46] "GET /static/style.css HTTP/1.1" 304 -
[13/Apr/2026 11:13:46] "GET /static/radial.css HTTP/1.1" 304 -
[13/Apr/2026 11:13:46] "GET /static/dashboard.css HTTP/1.1" 304 -
[13/Apr/2026 11:13:46] "GET /static/panels.css HTTP/1.1" 304 -
[13/Apr/2026 11:13:46] "GET /static/auth.css HTTP/1.1" 304 -
[13/Apr/2026 11:13:46] "GET /static/js/radial_menu.js HTTP/1.1" 304 -
[13/Apr/2026 11:13:46] "GET /static/manifest.json HTTP/1.1" 304 -
```

- Pravilna 2FA koda in uspešna prijava



```
[13/Apr/2026 11:16:27] "POST /login/2fa HTTP/1.1" 302 -
[13/Apr/2026 11:16:27] "GET / HTTP/1.1" 200 -
[13/Apr/2026 11:16:32] "GET /static/style.css HTTP/1.1" 304 -
[13/Apr/2026 11:16:32] "GET /static/dashboard.css HTTP/1.1" 304 -
[13/Apr/2026 11:16:32] "GET /static/radial.css HTTP/1.1" 304 -
[13/Apr/2026 11:16:32] "GET /static/manifest.json HTTP/1.1" 304 -
[13/Apr/2026 11:16:34] "GET /chat-ui HTTP/1.1" 200 -
[13/Apr/2026 11:16:34] "GET /static/js/dashboard.js HTTP/1.1" 200 -
[13/Apr/2026 11:16:34] "GET /static/js/radial_menu.js HTTP/1.1" 304 -
[13/Apr/2026 11:16:34] "GET /static/style.css HTTP/1.1" 304 -
[13/Apr/2026 11:16:34] "GET /static/radial.css HTTP/1.1" 304 -
[13/Apr/2026 11:16:34] "GET /static/dashboard.css HTTP/1.1" 304 -
[13/Apr/2026 11:16:34] "GET /static/chat.css HTTP/1.1" 200 -
```

Vir: (Lastni vir)

Priloga E2: Preverjanje zaščitenih spletnih poti

- Dostop do zaščitene poti brez prijave

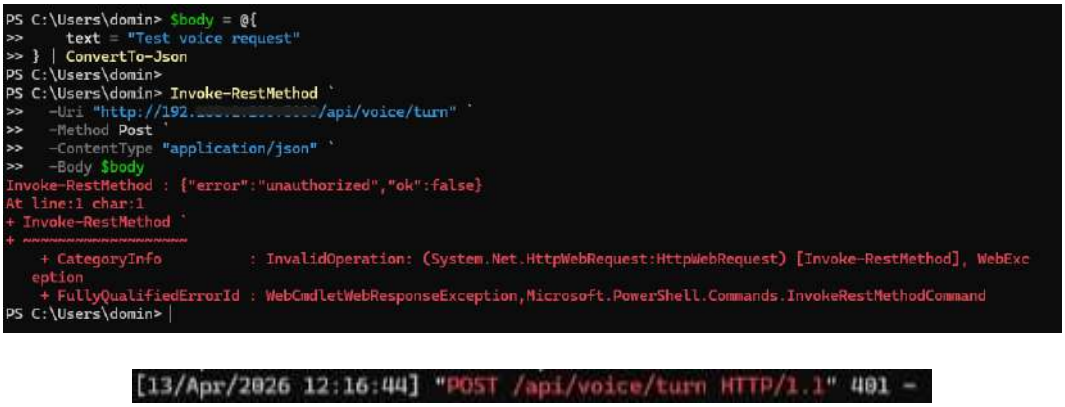


- Dostop do zaščitene poti po uspešni prijavi

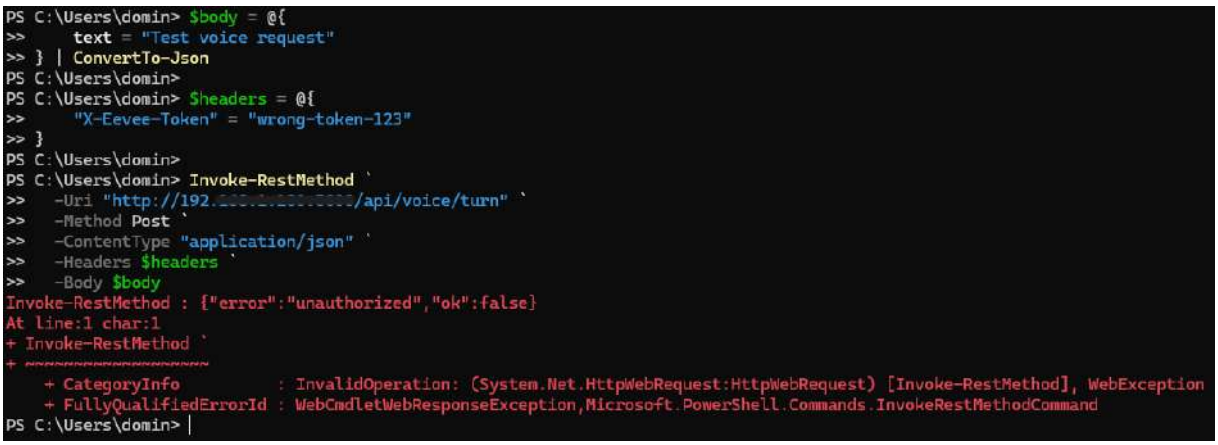


Priloga E3: Preverjanje žetonske zaščite glasovne poti

- Zahteva brez žetona



- Zahteva z napačnim žetonom



Vir: (Lastni vir)

- Zahteva s pravilnim žetonom

```
PS C:\Users\domin> $body = @{
>>     text = "Test voice request"
>> } | ConvertTo-Json
PS C:\Users\domin>
PS C:\Users\domin> $headers = @{
>>     "X-Eevee-Token" = "[redacted]"
>> }
PS C:\Users\domin>
PS C:\Users\domin> Invoke-RestMethod `
>>   -Uri "http://192.[redacted]/api/voice/turn" `
>>   -Method Post `
>>   -ContentType "application/json" `
>>   -Headers $headers `
>>   -Body $body


audio_url :
meta      : @{blocked=True; latency_ms=16765}
ok        : True
text      :
thread_id : 53



PS C:\Users\domin>
```

```
[12:14:44] [INFO][web] moderation: loaded 85 banwords from A:\Eevee\core\moderation\banwords.txt
[12:14:44] [INFO][web] moderation: loaded 743 profanity tokens from A:\Eevee\core\moderation\profanity.txt
[12:14:44] [DEBUG][intents] intents gate: no wake word
[12:14:47] [DEBUG][vector] [vector] semantic_exclude_ids=[]
[12:14:47] [DEBUG][vector] [vector] deterministic_ids=[]
[12:14:47] [DEBUG][vector] [vector] semantic_ids=[]
[12:14:47] [DEBUG][vector] [vector] merged_ids=[]
[12:14:47] [DEBUG][context] web ops system_extra len=0 preview=''
[12:15:04] [DEBUG][web] draft.ms=16766 | route=chat
[12:15:04] [DEBUG][web] web_fallback=off
192.[redacted] - - [13/Apr/2026 12:15:04] "POST /api/voice/turn HTTP/1.1" 200 -
```

Priloga E4: Preverjanje privilegiranega Discord ukaza

- Nepooblaščen poskus privilegiranega ukaza

```
[17:16:01] [INFO][discord] Starting Discord bot...
[17:16:07] [INFO][discord] [Eevee] Connected to Discord as Eevee#0085
[17:16:09] [INFO][discord] Slash commands synced: global (Pycord). total=3
[17:27:17] [DEBUG][discord] on_message fired - UUID: 2c7c46
[17:27:17] [DEBUG][discord] flags: is_dm=False is_cmd=True is_conv=False lowered='~restart flask'
[17:27:18] [DEBUG][discord] on_message fired - UUID: 2f67b3
```

- Pooblaščen ukaz in začetek izvršitvenega toka

```
[12:36:49] [INFO][discord] Starting Discord bot...
[12:36:53] [INFO][discord] [Eevee] Connected to Discord as Eevee#0085
[12:36:54] [INFO][discord] Slash commands synced: global (Pycord). total=3
[12:37:46] [DEBUG][discord] on_message fired - UUID: 036965
[12:37:46] [DEBUG][discord] flags: is_dm=False is_cmd=True is_conv=False lowered='~restart flask'
[12:37:46] [DEBUG][discord] Narration enabled for command 'restart'.
[12:37:48] [DEBUG][discord] Narration prompt built for 'Restarting now: {services}.': [ACTION:EXECUTED]
Restarting now: flask.

[12:37:51] [DEBUG][discord] draft.ms=2252 | route=chat
[12:37:51] [DEBUG][discord] web_fallback=off
[12:37:51] [DEBUG][discord] narration meta: route=chat latency_ms=2251 tokens=180
[12:37:51] [DEBUG][discord] Narration returned empty/whitespace; nothing sent.
[RestartManager] Waiting 3s before restarting flask...
[RestartManager] Restarting: flask
```

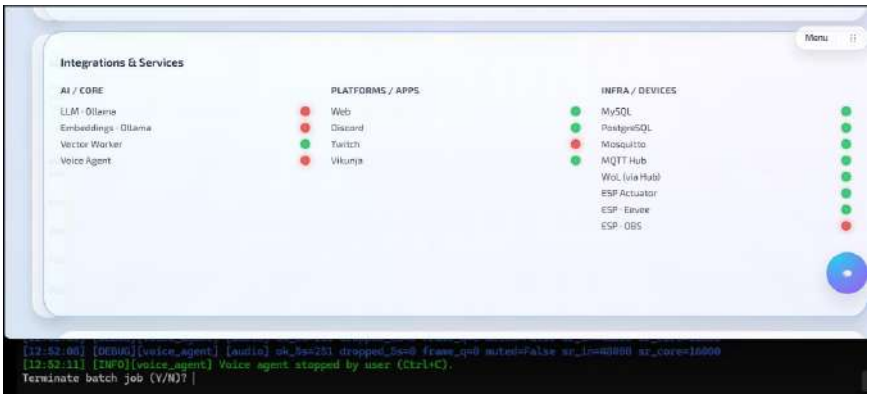
Vir: (Lastni vir)

Priloga E5: Spremljanje statusa storitve in okrevanje

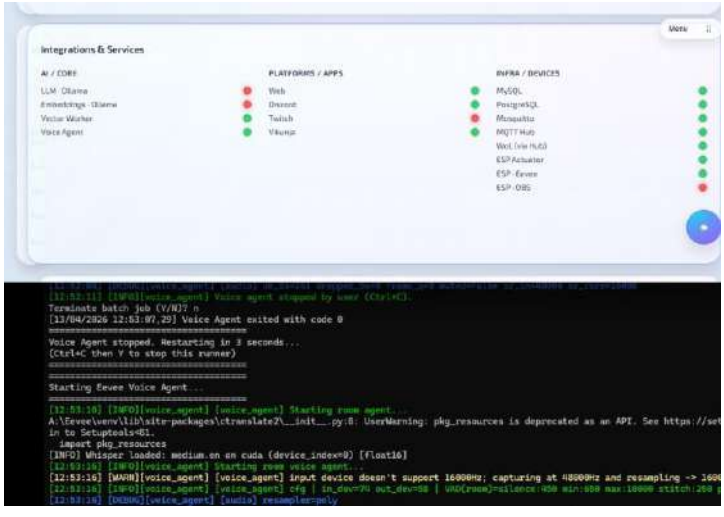
- Začetno stanje storitve



- Stanje po zaustavitvi storitev



- Stanje po ponovnem zagonu storitve



Vir: (Lastni vir)

Priloga E6: Lokalno verzioniranje razvojnega dela sistema

- Aktivnost repozitorija v okolju Gitea



Vir: (Lastni vir)