VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
INFORMATIKOS KATEDRA

Baigiamasis bakalauro darbas

Daiktų interneto prietaisų informacinė sistema

(Internet of things information system)

Atliko: 4 kurso 2 grupės studentas
Šarūnas Ramanauskas (parašas)
Darbo vadovas:
dr. Igor Katin (parašas)

Vilnius
2022

Turinys

Įvadas

Sąvoka daiktų internetas (angl. Internet of Things – IoT) atsirado gana atsitiktinai. Ją 1999 metais pirmą kartą konferencijos metu panaudojo mokslinių tyrimų grupės „Auto-ID Labs“ tuometinis vykdomasis direktorius Kevinas Ashtonas.

„Jeigu turėsime kompiuterius, kurie gebės surinkti bei panaudoti informaciją apie mus supančius daiktus be mūsų įsikišimo, galėsime ne tik ženkliai sumažinti kasdieninį eikvojimą, bet taip pat gebėsime veikti produktyviau ir našiau.“ – 1999 metais pristatymo „Procter & Gamble“ metu teigė Kevinas Ashtonas.

Pradinė idėja buvo siejama su radijo dažnio identifikavimo technologija RFID (angl. Radio-Frequency Identification). Vis dėlto jau tada buvo aišku, jog daiktų internetas turi tapti tuo pagalbininku, padėsiančiu žmonėms apdoroti, analizuoti bei panaudoti nuolatos augantį skaitmeninių duomenų srautą. Kitaip sakant, nuo pat pirmosios akimirkos visuomenei buvo bandoma pasakyti, kad daiktų internetas ruošiasi modernizuoti buitį bei mus supančiam inventoriui suteikti „protą“.

Įsivaizduokite, ryte suskambęs žadintuvas virtuvėje stovinčiam kavos aparatui liepia paruošti kavą, o darbo kambaryje stovintis spausdintuvas pats seka spausdinimui skirtų lapų likutį bei atėjus metui automatiškai pasirūpina jų užsakymu iš parduotuvės. Lygiai taip pat galėtų elgtis reikalaujamą maistą be žmogaus įsikišimo į namus užsakantis išmanusis šaldytuvas. Akivaizdu, daiktų internetas gali plėtoti autonominius skaitmeninius duomenų transporto tinklus, kurie padėtų kurti ir įgyvendinti visavertę išmaniųjų namų, gamyklų bei miestų idėją.

Kita vertus, integruoti didelio našumo procesorius į kiekvieną buityje naudojamą prietaisą yra nepraktiška. Daiktų interneto prietaisai susideda iš dviejų esminių komponentų - sensorių ir valdiklių. Sensorius fiksuoja atitinkamus paties įrenginio arbo jo aplikos duomenis ar parametrus. Valdiklis skirtas prietaiso funkcijoms aktyvuoti. Prietaiso fiksuojamų duomenų apdorojimas ir jo valdiklių aktyvavimas gali vykti centralizuotai taip sujungiant daug skirtingų prietaisų į bendrą tinklą. Įrenginys savaime geba tik išsiųsti savo surinktus duomenis, priimti ir įvykdyti funkcijas kurioms jis buvo sukurtas. Iš šio konteksto kylantis uždavinys - reikalinga programinė įranga suteikianti galimybę valdyti interneto daiktų prietaisus, rinkti ir stebėti jų renkamus duomenis.

Šio darbo tikslas - sukurti daiktų interneto valdymo sistemos prototipą. Uždaviniai:

- Išnagrinėti dalykinę sritį. Atlikti panašių, daiktų interneto prietaisus administruojančių, informacinių sistemų analizę ir palyginimą.
- Sukurti universalios daiktų interneto prietaisų informacinės sistemos projektą. Atlikti reikalavimų ir projektavimo specifikacijas.
- Išrinkti programavimo technologijas skirtas vartotojo sąsajos, duomenų saugojimo ir apdorojimo komponentų kūrimui.
- Realizuoti universalios informacinės sistemos projektą, skirtą administruoti daiktų interneto prietaisus.

1. Daiktų internetas

Daiktų internetas (angl. Internet of things - IOT) apibūdina fizinių objektų - „daiktų“ ar objektų - tinklą, įterptą su jutikliais, programine įranga ir kitomis technologijomis, siekiant prisijungti ir keistis duomenimis su kitais prietaisais ir sistemomis internete. Pagrindą tam suteikė realaus laiko analizės, mašininio mokymosi, jutiklių ir įterptųjų sistemų vystymasis. Tradiciniai įterptųjų sistemų, belaidžių jutiklių tinklų, valdymo sistemų, automatikos (įskaitant namų ir pastatų automatizavimą) ir kitos taikymo sritys prisideda prie daiktų interneto įgalinimo. Vartotojų rinkoje daiktų interneto technologija yra labiausiai sinonimas produktams, susijusiems su „protingo namo“ sąvoka, įskaitant prietaisus ir įrenginius (tokius kaip šviestuvai, termostatai, namų apsaugos sistemos ir kameros bei kiti buitiniai prietaisai). Yra daug rimtų susirūpinimų dėl DI augimo pavojų, ypač privatumo ir saugumo srityse, todėl pramonės ir vyriausybės ėmėsi veiksmų šiems susirūpinimams spręsti, įskaitant tarptautinių standartų kūrimą.

1.1. Istorija

Pagrindinė išmaniųjų įrenginių tinklo koncepcija buvo aptarta jau 1982 m., Kai modifikuotas „Coca-Cola“ pardavimo automatas Carnegie Mellon universitete tapo pirmuoju prie interneto prijungtu prietaisu, galinčiu pranešti apie savo atsargas ir tai, ar naujai pakrauti gėrimai buvo šalti, ar ne. 1991 m. Marko Weiserio straipsnis apie visur esančią kompiuteriją „XXI amžiaus kompiuteris“, taip pat tokios akademinės vietos kaip „UbiComp“ ir „PerCom“ sukūrė šiuolaikinę ”IOT” viziją. 1994 m. Reza Raji apibūdino šią sąvoką kaip „perkelti mažus duomenų paketus į didelį mazgų rinkinį, kad būtų galima integruoti ir automatizuoti viską, pradedant buitine technika ir baigiant visomis gamyklomis“. 1993–1997 m. bendrovės „Microsoft“ ir „Novell“ pasiūlė pirmuosius daiktų interneto sistemų sprendimus. Ši sritis įgavo pagreitį, kai Billas Joy’as numatė, kad ryšys tarp prietaisų yra jo „Šeši tinklai“ sistemos dalis, pristatyta Pasaulio ekonomikos forume Davose 1999 m. Terminą „daiktų internetas“ sugalvojo Kevinas Ashtonas iš „Procter & Gamble“, vėliau MIT „Auto-ID Center“, 1999 m., Nors jis labiau mėgsta frazę „Internetas“. Tuo metu jis laikė radijo dažnio identifikavimą (RDA) būtinu daiktų internetui, kuris leistų kompiuteriams valdyti visus atskirus dalykus. Apibrėždama daiktų internetą kaip „paprasčiausiai laiko momentą, kai prie interneto prijungta daugiau„ daiktų ar objektų “nei žmonių“, „Cisco Systems“ apskaičiavo, kad daiktų internetas „gimė“ 2008–2009 m., Daiktų ir žmonių santykiui augant. nuo 0,08 2003 m. iki 1,84 2010 m. [Bun18]

1.2. Taikymo sritys

1.2.1. Protingi namai

DI prietaisai yra didesnės namų automatikos koncepcijos dalis, kuri gali apimti apšvietimą, šildymą ir oro kondicionavimą, laikmenų ir apsaugos sistemas bei kamerų sistemas. Ilgalaikė nauda galėtų apimti energijos taupymą automatiškai užtikrinant, kad žibintai ir elektronika būtų išjungti,

arba informuodami namų gyventojus apie jų naudojimą. Protingi namai arba automatizuoti namai galėtų būti grindžiami platforma ar mazgais, kurie valdo išmaniuosius įrenginius ir prietaisus. Pavyzdžiui, naudodamiesi „Apple HomeKit“ gamintojai savo namų produktus ir priedus gali valdyti taikydami „iOS“ įrenginių, tokių kaip „iPhone“ ir „Apple Watch“, programas. Tai gali būti skirta programa arba „iOS“ vietinės programos, tokios kaip „Siri“. Tai galima įrodyti „Lenovo Smart Home Essentials“ atveju - tai išmaniųjų namų įrenginių, valdomų per „Apple Home“ programą ar „Siri“, nereikalaujantis „Wi-Fi“ tilto, linija. Taip pat yra specialių išmaniųjų namų centrų, kurie siūlomi kaip atskiros platformos, skirtos sujungti skirtingus išmaniųjų namų produktus. Tai apima „Amazon Echo“, „Google Home“, „Apple HomePod“ ir „Samsung“ „SmartThings Hub“. Be komercinių sistemų, yra daugybė nepatentuotų, atviro kodo ekosistemų; įskaitant „Home Assistant“, „OpenHAB“ ir „Domoticz“. [Bas21]

1.2.2. Medicina ir sveikatos priežiūra

The Medicinos daiktų internetas (IoMT) yra daiktų interneto taikymas medicinos ir sveikatos tikslais, tokais kaip duomenų rinkimas ir analizė medicininiams tyrimams arba pacientų stebėsena. IoMT buvo vadinama „išmaniąja sveikatos priežiūra“, kaip technologija, skirta sukurti skaitmeninę sveikatos priežiūros sistemą, susieti turimus medicinos išteklius ir sveikatos priežiūros paslaugas. DI prietaisai gali būti naudojami nuotolinio sveikatos stebėjimo ir pranešimo apie ekstremalias situacijas sistemoms įgalinti. Šie sveikatos stebėjimo prietaisai gali būti nuo kraujo spaudimo ir širdies ritmo monitorių iki pažangių prietaisų, galinčių stebėti specializuotus implantus, tokius kaip širdies stimuliatoriai, „Fitbit“ elektroninės riešinės ar pažangiosios klausos priemonės. Kai kurios ligoninės pradėjo diegti „išmaniąsias lovas“, kurios galėtų nustatyti, kada jos yra užimtos ir kada pacientas bando keltis. Taip pat esama prietaisų, galinčių prisitaikyti prie paciento poreikių, pavydžiui, užtikrinančių tinkamą spaudimą ir palaikymą pacientui be slaugytojų priežiūros. 2015 m. „Goldman Sachs“ ataskaita nurodė, kad sveikatos priežiūros IoT prietaisai „gali sutaupyti Jungtinėms Valstijoms daugiau nei 300 mlrd. USD metinių sveikatos priežiūros išlaidų, didinant pajamas ir mažinant išlaidas“. Be to, naudojant mobiliuosius prietaisus medicininei veiklai buvo sukurta „m-sveikata“, naudojama sveikatos statistikos analizei. Gyvenamosiose patalpose taip pat gali būti įrengti specializuoti jutikliai, skirti stebėti pagyvenusių žmonių sveikatą ir bendrą savijautą, taip pat užtikrinant, kad būtų skiriamas tinkamas gydymas, taip pat padedant žmonėms atgauti prarastą judrumą gydant. Šie jutikliai sukuria intelektualių jutiklių tinklą, galintį rinkti, apdoroti, perduoti ir analizuoti vertingą informaciją įvairiose aplinkose, pavyzdžiui, prijungti stebėjimo namuose įrenginius prie ligoninėse veikiančių sistemų. Kiti vartojimo prietaisai, skatinantys sveiką gyvenimo būdą, pvz., Prijungtos svarstyklės ar nešiojami širdies monitoriai, taip pat yra IoT galimybė. Priešgimdyminiams ir lėtiniams pacientams taip pat yra prieinamos sveikatos stebėjimo sistemos, užtikrinančios sveikatos būklę ir pasikartojančius vaistų poreikius. Nuo 2018 m IoMT buvo taikomas ne tik klinikinių laboratorijų pramonėje, bet ir sveikatos priežiūros bei sveikatos draudimo pramonėje. IoMT sveikatos priežiūros pramonėje dabar leidžia gydytojams, pacientams ir kitiems asmenims, tokiems kaip pacientų globėjai, slaugytojai, šeimos ir panašūs, būti sistemos dalimi,

kai pacientų įrašai saugomi duomenų bazėje, leidžiančioje gydytojams ir likusiems žmonėms medicinos personalui suteikti prieigą prie pacientų informacijos. Be to, IoT pagrįstos sistemos yra orientuotos į pacientą, o tai reiškia, kad pacientas turi būti lankstus atsižvelgiant į sveikatos būklę. Draudimo srityje esanti IoMT suteikia prieigą prie geresnės ir naujos rūšies dinamiškos informacijos. Tai apima jutikliais pagrįstus sprendimus, tokius kaip biosensoriai, nešiojamieji kompiuteriai, prijungti sveikatos priežiūros prietaisai ir mobiliosios programos klientų elgsenai stebėti. DI taikymas sveikatos priežiūros srityje vaidina pagrindinį vaidmenį valdant lėtines ligas, taip pat prevencijai ir kontrolei. Nuotolinis stebėjimas yra įmanomas sujungus galingus belaidžius sprendimus. Ryšys leidžia sveikatos priežiūros specialistams užfiksuoti paciento duomenis ir taikyti sudėtingus algoritmus sveikatos duomenų analizėje. [Bas21]

1.2.3. Transportas

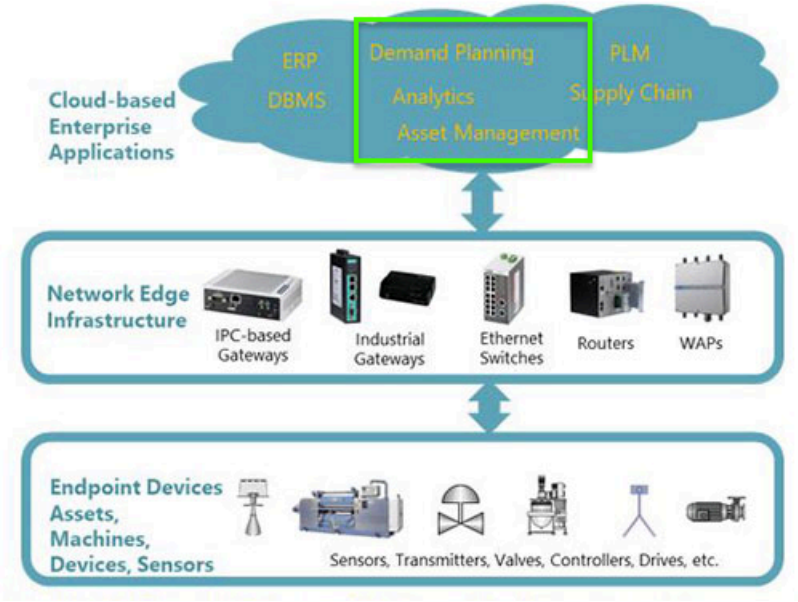
DI gali padėti integruoti ryšius, valdyti ir apdoroti informaciją įvairiose transporto sistemose. DI taikomas visiems transporto sistemų aspektams (t. y. Transporto priemonei, infrastruktūrai ir vairuotojui ar naudotojui). Dinaminė šių transporto sistemos komponentų sąveika suteikia galimybę susisiekti transporto priemonėse, išmaniai valdyti eismą, statyti automobilius. Taip pat daiktų interneto spendimai gali būti taikomi elektroninėse rinkliavų surinkimo sistemose, automobilių parkų valdyme, siekiant pagerinti eismo saugumą ir pagalbos kelyje efektyvumą.

1.3. Daiktų interneto ekosistema

Bendruoju atveju apibrėžiant daiktų internetą, visose jo taikymo srityse galima išskirti šias pagrindines sudedamasias dalis:

- *Prietaisai* - smulkiausias daiktų interneto vienetas. Įrenginys gebantis rinkti duomenis ir/arba vykdyti savo funkciją(-as).
- *Tinklo sistemos(Internetas)* - infrastruktūra skirta įrenginių sujungimui į bendrą tinklą. Norint įgalinti įrengius jiems reikalnga prieiga prie viešo arba vietinio interneto tinklo.
- *Debesis* - Taikomosios stebėjimo ir valdymo programos. Sprendimai renkamų duomenų saugojimui, analizei, įrenginių stebėjimui ir valdymui.

Šios trys dalys įvardijamos kaip pagrindiniai daiktų interneto architektūros sluoksniai. [Kav18; Mik19]



1 pav. Daiktų interneto ekosistema

Informacinės sistemos skirtos daiktų interneto įrenginių stebėjimui ir valdymui patenka į trečiąjį sluoksnį(žr. pav.). Kaip jau aprašyta pirmame skyriuje, daiktų internetas turi labai daug taikymo sričių, kuriose skirtingiems tiklsams naudojami skirtingi įrenginiai. Iš to išplaukia vienas esminių informacinės sistemos reikalavimų - sistema turi sugebėti palaikyti įvairius įrenginius, priimti skirtingus duomenis siunčiamus iš skirtingo tipo įrenginių. Taip pat kiekvienas įrenginys turi sau specifinę konfigūraciją, kurios valdymas ir keitimas gali būti vykdomas informacinės sistemos pagalba. Svarbus aspektas yra įrenginių skaičius. Pavydžiui, vien namuose galima įdiegti didelį skaičių daiktų interneto prietaisų. Kiekviename namo kambaryje gali buti atskiri temperatūros davikliai, šviesos jungikliai, langų žaliuzių valdikliai ir taip toliau.

Į grupes sujungti įrenginiai gali veikti kaip vientisa sistema. Pavydžiui, gamybos linijoje esantys sensoriai siunčia informaciją apie žaliavų kiekį ir priklausomai nuo to sistemos administratorius turi gebėti koreguoti kitų gamybos linijoje esančių įrenginių būseną(konfigūraciją).

Taigi, pagrindiniai sistemos kriterijai:

- *Prietaisai* - sistema turi gebėti palaikyti įvairius įrenginius, priimti skirtingus duomenis siunčiamus iš skirtingo tipo įrenginių. Taip turi būti numatyta galimybė dinamiškai apibrėžti įrenginio konfigūraciją.

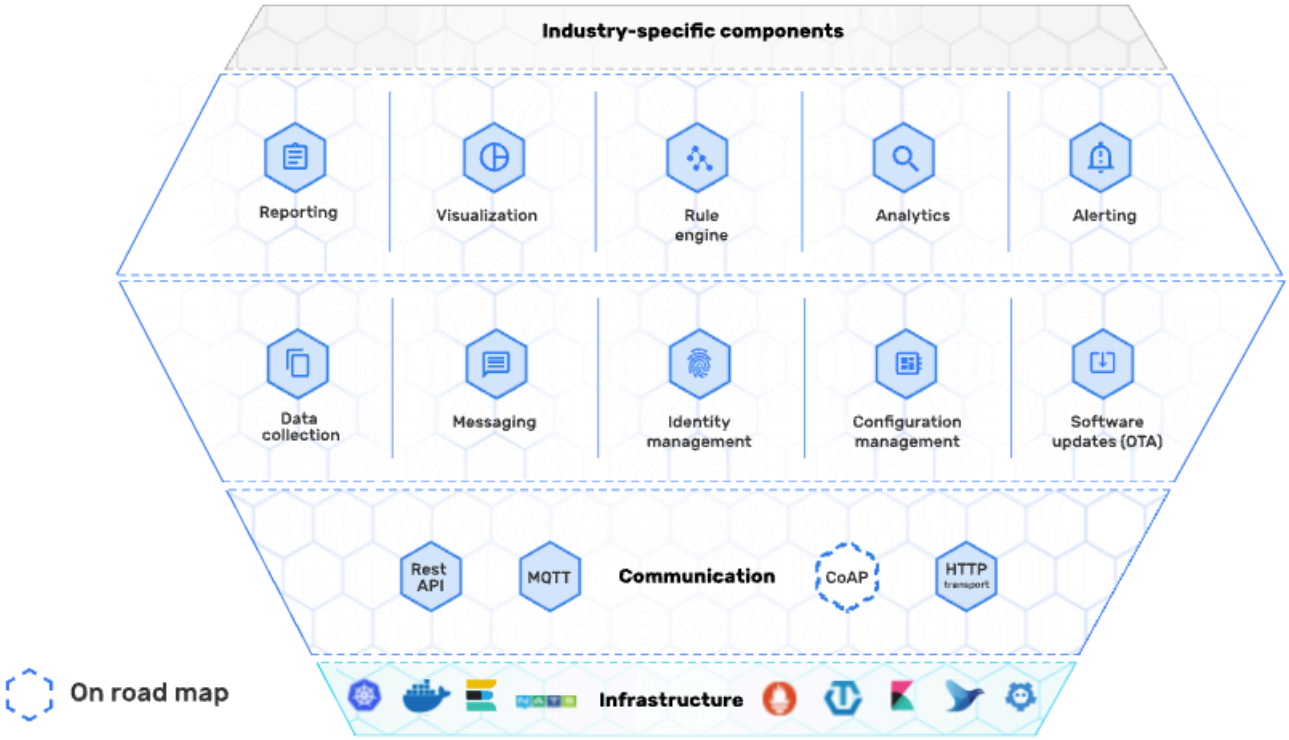
- Įrenginių tinklas gali susideti iš didelio skaičiaus prietaisų. Sistemoje turi būti numatyta galimybė skirstyti prietaisus į grupes pagal jų lokaciją arba paskirtį.

- Kiekvieno įrenginio būsenos stebėjimas ir nustatymų keitimas yra žmogaus įsiterpimo ir laiko reikalaujantis procesas. Reikalinga įgyvendinti sprendimą leidžiantį sukonfigūruoti scenarijus kai sistema automatiškai pakeičia įrenginio(arba kitų įrenginių) konfigūraciją priklausomai nuo surenkamų duomenų.

1.4. Egzistuojančių sprendimų analizė

1.4.1. Kaa IoT

„Kaa“ yra visapusiška daiktų interneto platforma, taikoma bet kokio masto įmonės daiktų interneto projektams. Ji siūlo daugybę funkcijų, leidžiančių kūrėjams kurti pažangias programas išmaniesiems produktams, lanksčiai valdyti savo prijungtus įrenginius per debesį, organizuoti visišką duomenų apdorojimą, analizuoti įrenginio telemetriją ir dar daugiau. Su IoT funkcijomis, kurias „Kaa“ teikia, galima kurti savo IoT programas iki 10 kartų greičiau nei anksčiau. Visos „Kaa“ funkcijos yra įdiegtos naudojant mikro servisus, o „Kaa“ visuma yra pagrįsta lanksčia mikro servisų architektūra. Tai reiškia, kad galima atskirai tinkinti kiekvieną Kaa funkciją, pridėti naujų arba esamas pakeisti kai kuriais trečiųjų šalių įrankiais. „Kaa“ siūlo visas daiktų interneto funkcijas, kurių gali prireikti įprastai IoT programai – nuo duomenų rinkimo ir įrenginių valdymo iki daiktų interneto prietaisų skydelių ir analizės. Taip pat galite pasinaudoti atviromis API, kad integruotumėte Kaa funkcijas į savo modulius ir programas. Šioje diagramoje parodyta bendra funkcinė Kaa architektūra.

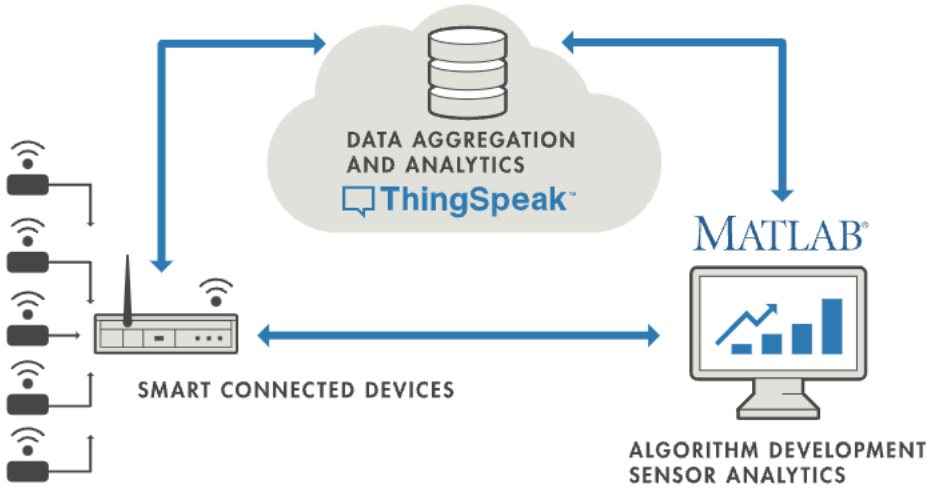


2 pav. Kaa IoT platformos komponentai

1.4.2. ThingSpeak

ThingSpeak™ yra IoT analizės platformos paslauga, leidžianti kaupti, vizualizuoti ir analizuoti tiesioginius duomenų srautus debesyje. „ThingSpeak“ teikia įrenginių duomenų vizualizacijas. Turėdami galimybę vykdyti MATLAB® kodą ThingSpeak, galite atlikti internetinę duomenų analizę ir apdorojimą. „ThingSpeak“ dažnai naudojama prototipams kurti ir IoT sistemoms, kurioms

atliekama renkamų duomenų analizė.



3 pav. ThingSpeak platforma

„ThingSpeak“ leidžia kaupti, vizualizuoti ir analizuoti tiesioginius duomenų srautus debesyje. Kai kurios pagrindinės ThingSpeak galimybės:

- Lengvai konfiūruoti įrenginius ir siųsti duomenis į ThingSpeak naudojantis populiariariais IoT protokolais.
- Vizualizuoti jutiklio duomenis realiuoju laiku.
- Apibendrinti duomenis pagal pareikalavimą iš trečiųjų šalių šaltinių.
- Naudotis MATLAB, atlikti analitikos uždavinius su savo daiktų interneto duomenimis.
- Automatiškai valdyti daiktų interneto prietaisus pagal tvarkaraščius ar įvykius.
- Reakcijos į duomenis – tiek neapdorotus, tiek naujus, apskaičiuojamus, duomenis. Sistemoje apibrėžiamos taisyklės(*Conditions*) ir veiksmai (*Actions*) atliekami, kai taisyklės/sąlygos patenkinamos.

1.4.3. AWS IoT Core

„AWS IoT Core“ leidžia prijungti įrenginius prie AWS debesų kompiuterijos paslaugų ir kitų įrenginių, apsaugoti duomenis ir kurti jų tarpusavio sąveiką, apdoroti įrenginio duomenis ir veikti pagal juos, leidžia programoms sąveikauti su įrenginiais.

AWS IoT Core galimybės:

- *AWS IoT Device SDK* padeda lengvai ir greitai prijungti aparatūros įrenginį arba mobiliąją programą prie AWS IoT Core. AWS IoT įrenginio SDK leidžia įrenginiams prisijungti, autentifikuotis ir keistis pranešimais su AWS IoT Core naudojant MQTT, HTTP arba WebSockets

protokolus. AWS IoT įrenginio SDK palaiko C, JavaScript ir Arduino ir apima klientų bibliotekas, kūrėjo vadovą ir perkėlimo vadovą gamintojams. Taip pat galima naudoti atvirojo kodo alternatyvą arba parašyti savo SDK.

- *Device Gateway* Tai yra IoT įrenginių, besijungiančių prie AWS, įėjimo taškas. *Device Gateway* valdo visus aktyvius įrenginių ryšius ir įgyvendina kelių protokolų semantiką, kad užtikrintų, jog įrenginiai galėtų saugiai ir efektyviai susisiekti su AWS IoT Core.

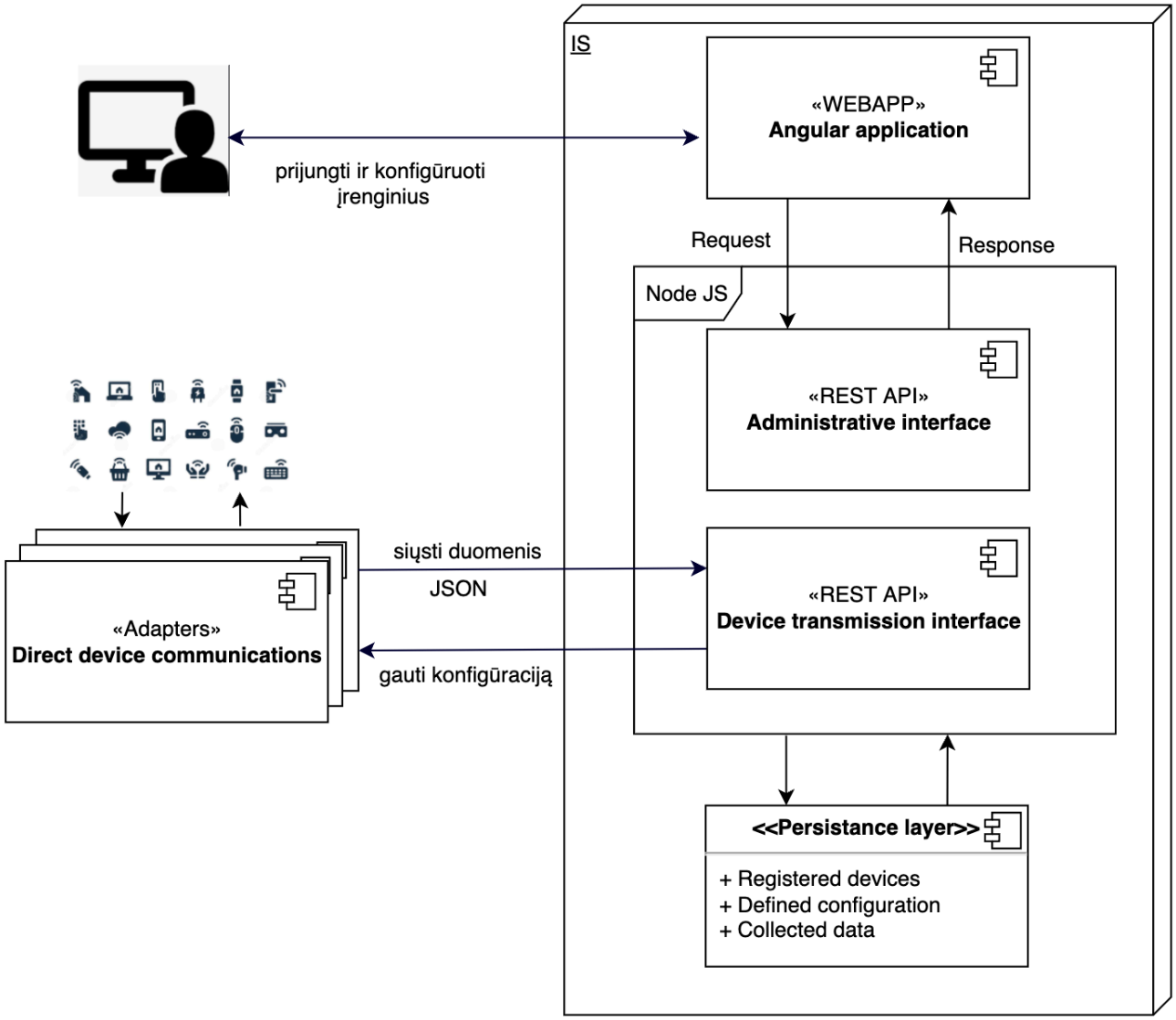
- *Rules Engine* Taisyklių modulis leidžia kurti daiktų interneto programas, kurios renka, apdoroja, analizuoja ir veikia prijungtų įrenginių generuojamus duomenis. Taisyklių modulis įvertina gaunamus pranešimus, paskelbtus AWS IoT Core, ir transformuoja bei pristato juos į kitą įrenginį arba debesies paslaugą pagal apibrėžtas verslo taisykles. Taisyklė gali būti taikoma duomenims iš vieno ar kelių įrenginių ir lygiagrečiai gali atlikti vieną ar kelis veiksmus. Taisyklių modulis taip pat gali nukreipti pranešimus į AWS galinius taškus, įskaitant AWS IoT Analytics, AWS IoT Events, AWS Lambda, Amazon Kinesis, Amazon S3, Amazon DynamoDB, Amazon CloudWatch, Amazon Simple Notification Service (SNS), Amazon Simple Queue Service (SQS), „Amazon Elasticsearch Service“ ir „AWS Step Functions“. Išorinius galutinius taškus galima pasiekti naudojant AWS Lambda, Amazon Kinesis, Amazon SNS ir Rules Engine vietinį HTTP veiksmą. Galima kurti taisykles valdymo konsolėje arba rašyti taisykles naudojant į SQL panašią sintaksę. Pavyzdžiui, jei temperatūros rodmuo viršija tam tikrą slenkstį, tai gali suaktyvinti duomenų perdavimo į AWS lambda taisyklę. Taisyklės taip pat gali būti sukurtos siekiant atsižvelgti į kitus debesyje esančius duomenis, pvz., duomenis iš kitų įrenginių. Taisyklės taip pat gali suaktyvinti „Java“, „Node.js“ arba „Python“ kodo vykdymą „AWS Lambda“, suteikdamos jums maksimalų lankstumą ir galią apdoroti įrenginio duomenis.

Palyginti su kitomis rinkoje esančiomis įmonių debesų platformomis, „Kaa“ leidžia įmonėms daug greičiau ir su mažiau pastangų pradėti naudotis daiktų internetu. Kadangi visos pagrindinės daiktų interneto funkcijos teikiamos intuityvioje vartotojo sąsajoje, „Kaa“ sumažina laiką ir įgūdžius, reikalingus jūsų įrenginiams prijungti ir duomenims vizualizuoti. ”ThingSpeak” išskirtinumas - tiesioginė sąsaja su MatLab, suteikia daug galimybių atlikti duomenų analiką. Didesnių debesų kompiuterijos organizacijų siūlomų IoT sprendimų privalumas - lengvai atlikemos integracijos į kitus to paties debesų kompiuterijos gamintojo sprendimus. Taip pat esama jau naudojimui paruoštų papildomų įrankių skirtų duomenų analitikos(*AWS IoT Analytics*), vizualizacijos(*AWS IoT SiteWise*) uždaviniams spręsti. AWS IoT Core pasižymi sudėtingesnėmis darbo eigomis su didesniu veiksmų skaičiumi panašioms užduotims atlikti, sąrankai reikia naudoti konsolę ir tai užtrunka ilgiau.

2. Projektavimas

2.1. Vizija

Reikalinga sistema, priimanti daiktų interneto įrenginių siunčiamus duomenis. Sistemos vartotojai turi turėti galimybę pridėti įrenginius ir dinamiškai sukonfigūruoti kaip bus interpretuojama kiekvieno įrenginio atsiųsta informacija. Taip pat turi būti numatytas mechanizmas apibrėžti įrenginio konfigūracijos schemą ir nustatyi konfigūracijos reikšmes. Pateiktoje schemoje *Adapters* pažymėtas komponentas(-ai) iliustruoja tarpines sistemas tarp daiktų interneto prietaisų ir kuriamos IS. Šio darbo eigoje apsiribojame susitarimu, kad šios sistemos koordinuoja bendravimą su skirtingų charakteristikų įrenginiais. Tokiu pagrindu, visi įrenginiai prie sistemos jungsis HTTP protokolu ir regurialiai siųs dviejų tipų REST užklausas: konfigūracijos atnaujinimo ir duomenų perdavimo.

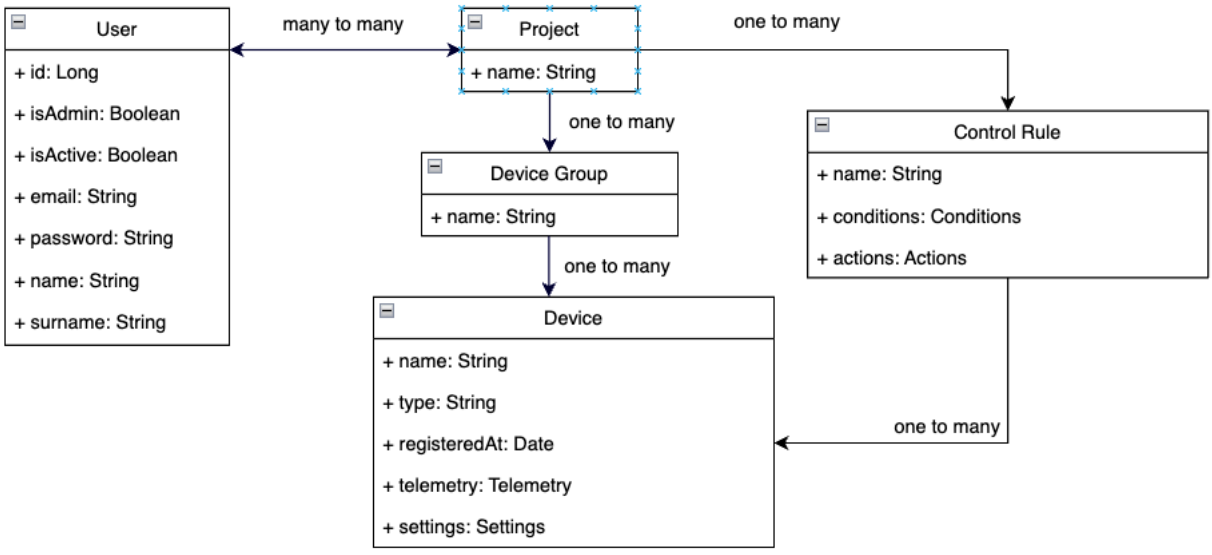


4 pav. Sistemos paskirtys

2.2. Esybės

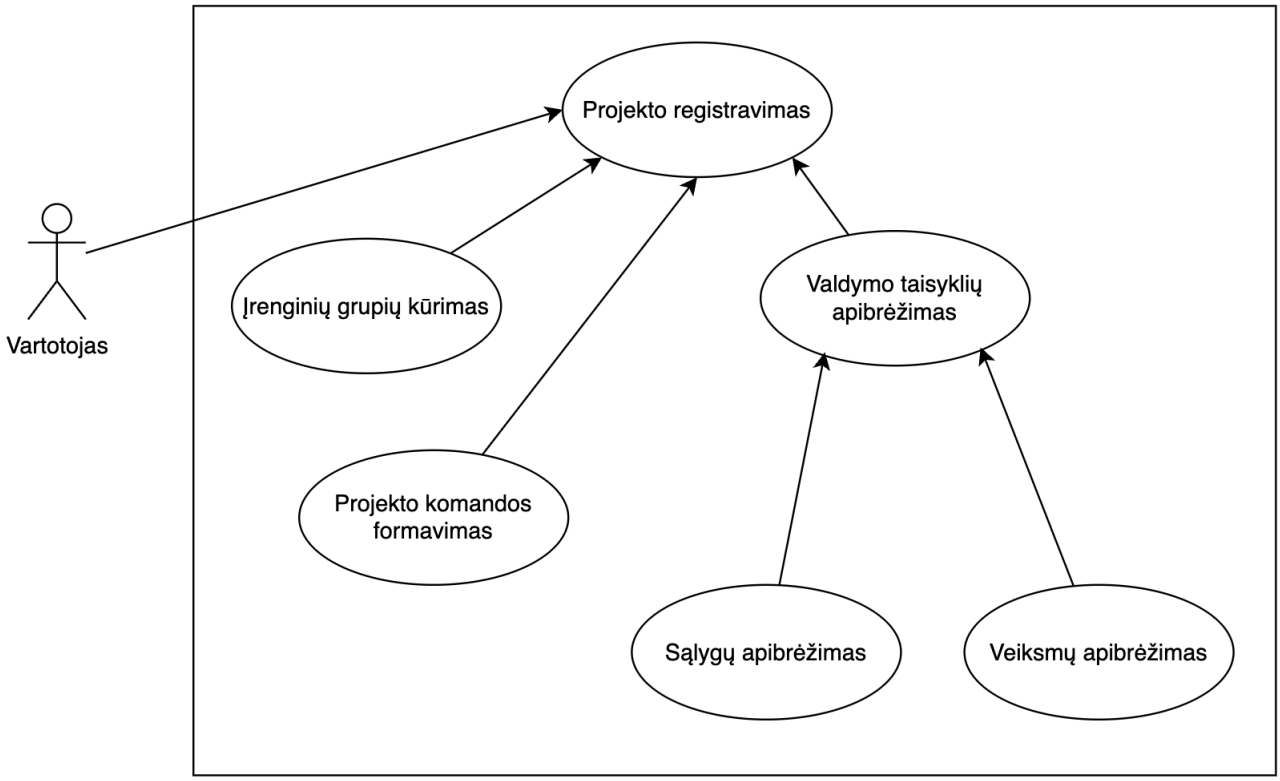
- *Vartotojas(User)* - registruotas sistemos vartotojas. Vartotojas sistemoje turi gebėti pridėti ir konfigūruoti įvairaus tipo įrenginius, peržiūrėti jų siunčiamus duomenis.
- *Projektas(Project)* - Įrenginių, jų grupių ir juos administruojančių vartotojų visuma. Vienas vartotojas gali dalyvauti daugiau nei viename projekte. Vienas įrenginys(ir jų grupė) egzistuoja tik vieno projekto ribose.
- *Grupė(Group)* - įrenginių organizavimas į mažesnes logines grupes(vieno projekto kontekste) pagal projekto komandos(vartotojų) apibrėžtą tvarką.
- *Įrenginys(Device)* - vieną fizinį įrenginį atitinkanti esybė. Kaip savo atributus turi dinaminę įrenginio siunčiamų duomenų schemą ir vartotojo valdomą prietaiso konfigūraciją.
- *Valydmo Taisyklė(Control Rule)* - automatizuoto įrenginių valdymo instrukcija. Kaip savo atributus turi sąrašą sąlygų - sąlygas apibrėžia administratorius, jos tikrina iš nurodytų įrenginių atsiųstus duomenis. Veiksmai(*Actions*) nurodo kaip ir kokiems įrenginiams pakeičiama konfigūraciją, jei joje apibrėžtos sąlygos yra patenkintos.

2.3. Klasių diagrama



5 pav. Sistemos esybių sąryšiai

2.4. Panaudojimo atvejai



6 pav. Projekto administravimas

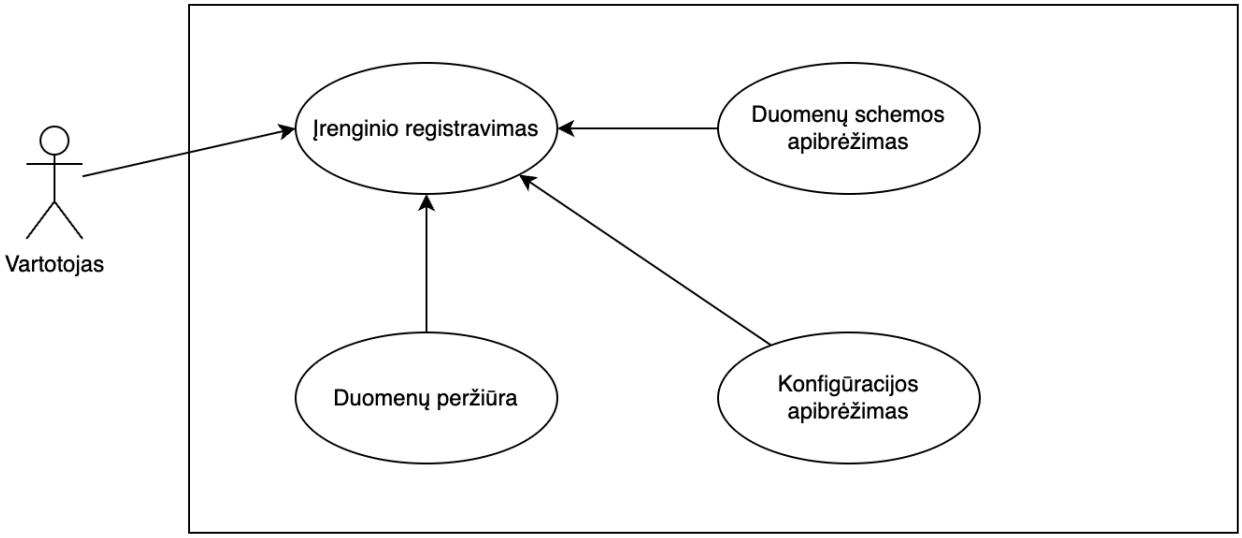
Pavadinimas	**Projekto registravimas**
Aprašymas	Sukuriamas naujas, tuščias projektas įrenginių registravimui, stebėjimui ir valdymui.
Aktorius	Vartotojas
Prieš sąlyga	-
Vykdymo eiga	1. Įvedama projekto informacija
Po sąlyga	Vartotojas automatiškai tampa projekto administratoriumi

Pavadinimas	**Valdymo taisyklių apibrėžimas**
Aprašymas	Valdymo taisyklių apibrėžimas
Aktorius	Vartotojas
Prieš sąlyga	1. Vartotojas dalyvauja projekte ir suteiktos teisės.
Vykdymo eiga	1. Apibrėžiamos sąlygos 2. Apibrėžiami veiksmai

Pavadinimas	**Sąlygų apibrėžimas**
Aprašymas	Valdymo taisyklės sąlygų apibrėžimas. Valdymo taisyklė fiksuoja nurodytų įrenginių duomenis ir tikrina ar jų reikšmės atitinka nurodytas sąlygas. Jei sąlygos tenkinamos - taisyklė pritaikoma
Aktorius	Vartotojas
Prieš sąlyga	1. Vartotojas dalyvauja projekte ir suteiktos teisės 2. Projekte egzistuoja įrenginys(-iai) su apibrėžta duomenų schema ir konfigūracija
Vykdymo eiga	1. Pasirenkamas projekte regisruotas įrenginys 2. Pasirenkama įrenginio duomenų schemos nuoroda 3. Pasirenkamas sąlygos operatorius 4. Įvedama skaitinė reikšmė, kuri yra lyginiama su įrenginio duomenų reikšme
Alternatyvūs veiksmai	1. Vartotojas gali apibrėžti norimą skaičių sąlygų vienoje taisyklėje 2. Vartotojas gali panaikinti pasirinką sąlygą iš taisyklės

Pavadinimas	**Valdymo taisyklės veiksmų apibrėžimas**
Aprašymas	Valdymo taisyklės veiksmų apibrėžimas. Jei visos taisyklėje apibrėžtos sąlygos yra patenkintos, sistema atlieka veiksmus - keičia nurodytų įrenginių konfigūraciją pagal taisyklėje nurodytas reikšmes
Aktorius	Vartotojas
Prieš sąlyga	1. Vartotojas dalyvauja projekte ir suteiktos teisės 2. Projekte egzistuoja įrenginys(-iai) su apibrėžta duomenų schema ir konfigūracija
Vykdymo eiga	1. Pasirenkamas projekte regisruotas įrenginys 2. Pasirenkama įrenginio konfigūracijos schemos nuoroda 3. Įvedama nauja konfigūracijos parametro reikšmė
Alternatyvūs veiksmai	1. Vartotojas gali apibrėžti norimą skaičių veiksmų vienoje taisyklėje 2. Vartotojas gali panaikinti pasirinką veiksmą iš taisyklės.

Pavadinimas	**Projekto komandos formavimas**
Aprašymas	Vartotojų pridėjimas ir šalinimas iš projekto
Aktorius	Vartotojas
Prieš sąlyga	1. Vartotojas dalyvauja projekte ir suteiktos teisės.
Vykdymo eiga	1. Į projektą pridedamas kitas egzistuojantis sistemos vartotojas. arba 1. Vartotojas pašalinamas iš projekto arba 1. Vartotojas palieka projektą



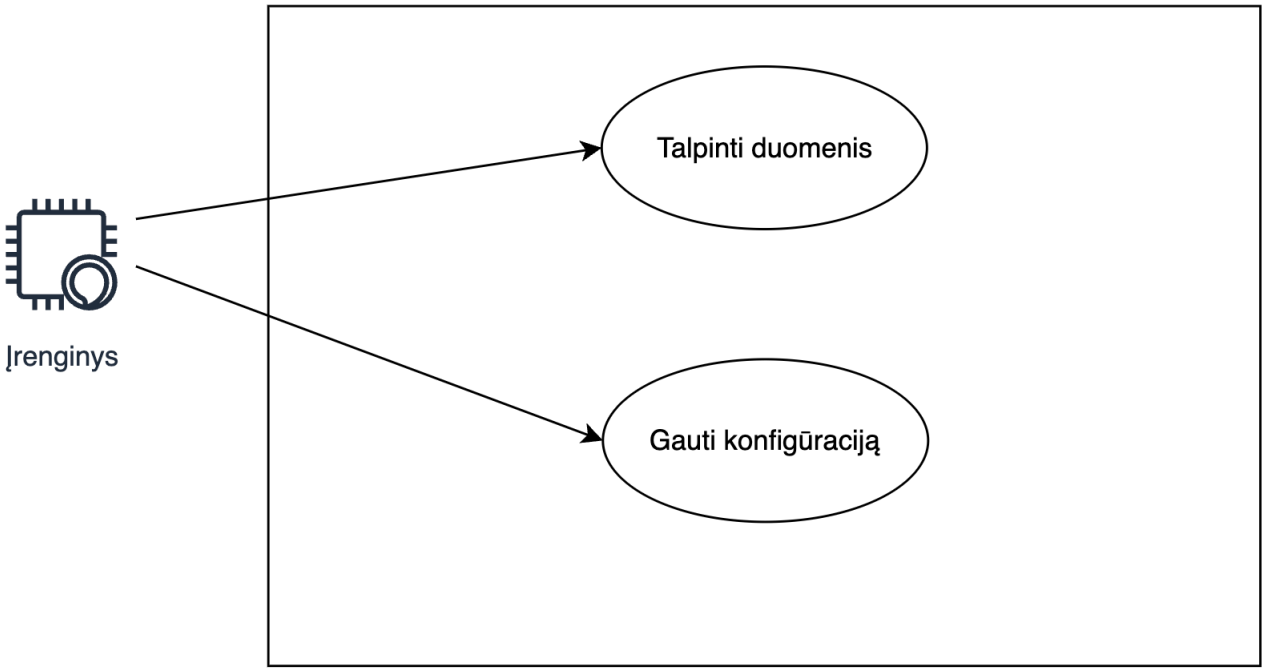
7 pav. Įrenginio administravimas

Pavadinimas	**Įrenginio registravimas**
Aprašymas	Naujo įrenginio sistemoje registravimas
Aktorius	Vartotojas
Prieš sąlyga	1. Vartotojas dalyvauja projekte ir suteiktos teisės. 2. Projekte sukurta bent viena įrenginių grupė.
Vykdymo eiga	1. Nurodoma įrenginio informacija 2. Aprašoma duomenų schema 3. Aprašoma konfigūracija

Pavadinimas	**Duomenų peržiūra**
Aprašymas	Įrenginio siunčiamų duomenų peržiūra
Aktorius	Vartotojas
Prieš sąlyga	1. Vartotojas dalyvauja projekte ir suteiktos teisės. 2. Esama įrenginių, kurie siunčia savo duomenis į sistemą.
Vykdymo eiga	1. Įrenginio duomenys peržiūrimi laiko/duomenų grafike

Pavadinimas	**Duomenų schemos apibrėžimas**
Aprašymas	Duomenų schema priskirta įrenginiui aprašo kaip sistema nuskaitys įrenginio siučiamus duomenis.
Aktorius	Vartotojas
Prieš sąlyga	1. Vartotojas dalyvauja projekte ir suteiktos teisės. 2. Projekte egzistuoja įrenginys(-iai)
Vykdymo eiga	1. Nurodomas duomenų pavadinimas sistemos aplinkoje. 2. Įvedama nuoroda nusakanti kelią iš kurio bus nuskaityta reikšmė.
Alternatyvūs veiksmai	1. Vartotojas gali apibrėžti pasirinktą skaičių nuorodų

Pavadinimas	**Sąlygų apibrėžimas**
Aprašymas	Įrenginio konfigūracijos apibrėžimas
Aktorius	Vartotojas
Prieš sąlyga	1. Vartotojas dalyvauja projekte ir suteiktos teisės. 2. Projekte egzistuoja įrenginys(-iai)
Vykdymo eiga	1. Suteikiamas konfigūracijos parametro pavadinimas sistemos viduje 2. Pasirenkama konfigūracijos parametro tipas(tekstas, skaičius arba loginis) 3. Įvedama nuoroda nusakanti kelią į kur bus įrašyta reikšmė. 4. Įvedama/nustatoma parametro reikšmė
Alternatyvūs veiksmai	1. Vartotojas gali apibrėžti pasirinktą skaičių konfigūracijos parametrų



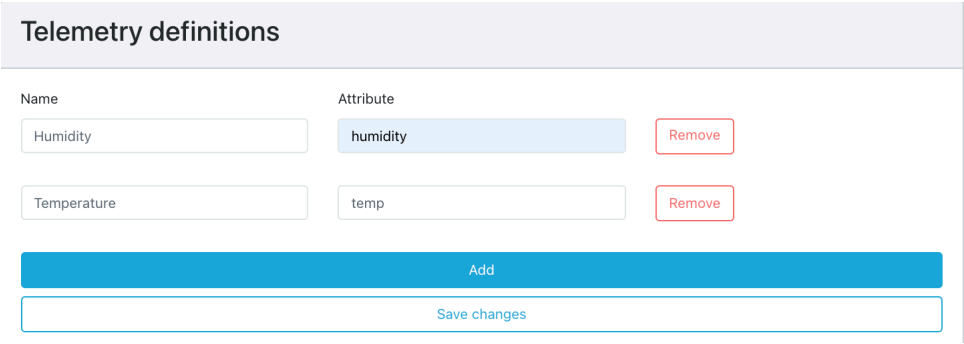
8 pav. Įrenginio ir sistemos komunikacija

Pavadinimas	**Duomenų talpinimas**
Aprašymas	Sistemoje registruotas ir atitinkamai sukonfigūruotas įrenginys siunčia duomenis į sistemą
Aktorius	Įrenginys
Prieš sąlyga	1. Įrenginys registruotas sistemoje 2. Įrenginys turi apibrėžta duomenų schemą
Vykdymo eiga	1. Iškviečiamas duomenų fiksavimo WEB servisas 2. Atsiustas duomenų paketas filtruojamas pagal įrenginiui apibrėžtą duomenų schemą, saugojami tik nurodyti laukai

Pavadinimas	**Konfigūracijos atnaujinimas**
Aprašymas	Konfigūracija leidžia koreguoti/valdyti įrenginio būseną, atliekamą funkciją ir kitas galimas konkretaus įrenginio parinktis
Aktorius	Įrenginys
Prieš sąlyga	1. Įrenginys registruotas sistemoje
Vykdymo eiga	1. Iškviečiamas konfigūracijos naujinimo WEB servisas 2. Tikrinimos valdymo taisyklės, galinčios koreguoti besikreipiančio įrenginio konfigūraciją 3. Į įrenginį išsiunčiama naujausia konfigūracija
Schema	

2.5. Įrenginio konfigūracijos formos

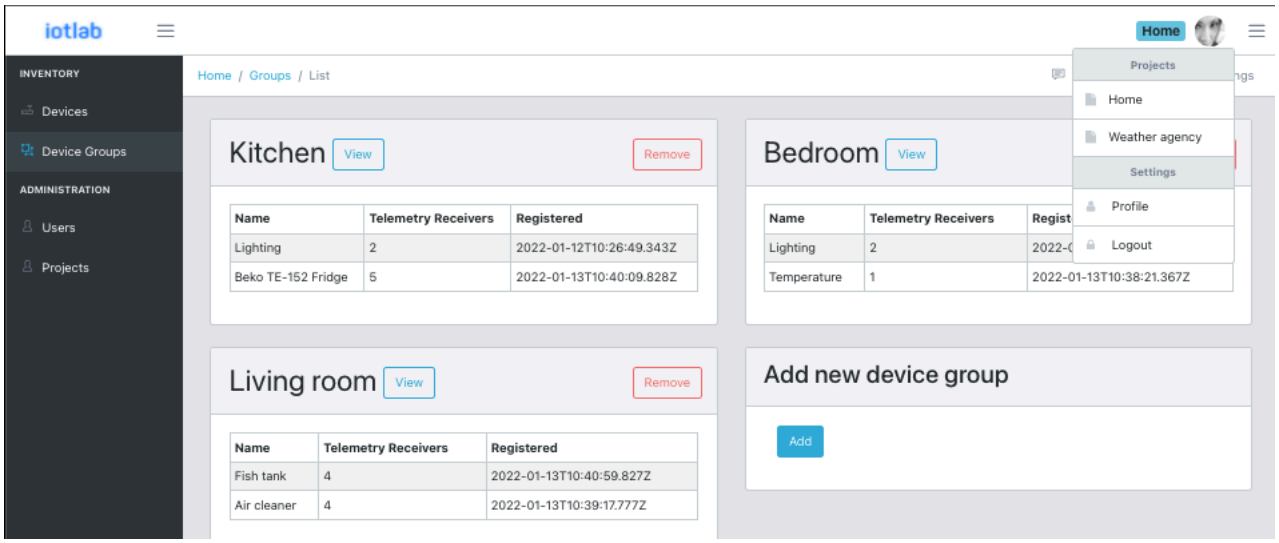
Iš skritingų įrenginių siunčiamos informacijos kiekis ir struktūra gali būti skirtinga, reikalingas būdas priklausomai nuo įrenginio fiksuoti perdavimo metu gautus duomenis. Lygiai tas pat principas galioja ir konfigūracijos parametrams. Įrenginio peržiūros ir registravimo languose pridedame dinamines formas kuriose vartotojas apibrėžia kaip iš įrenginio gauti duomenys bus fiksuojami ir kokios struktūros konfigūracija bus siunčiama į prietaisą, šiam paprašius atnaujinti konfigūraciją. Šie apibrėžimai nėra vartotoja pasirinkimo laisvė - jie priklauso nuo atitinkamo įrenginio specifikacijos. Tokiu būdu, varotojas atlieka sistemos ir įrenginio derinimą.



9 pav. Vartotojas apibrėžia kaip nuskaityti įrenginio siunčiamus duomenis

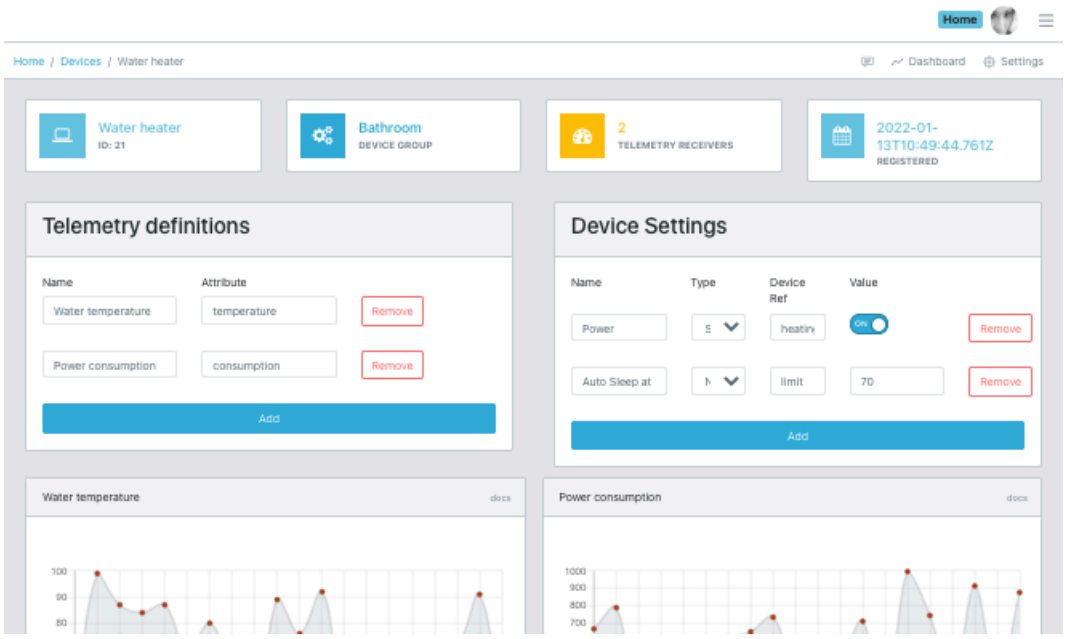
2.6. Įrenginio peržiūros ir grupavimo langai

Pavyzdyje matome įrenginių grupių sąrašo langą. Viršutiniame kampe nurodoma kurio projekto kontekste vartotojas šiuo metu yra. Šalia yra išsiskleidžiantis sąrašas kurio elementai - visi projektai kuriuose prisijungęs vartotojas dalyvauja. Grupių pavadinimai projekte yra pasirenkamaimi laisvai, taigi projekto komanda gali nuspresti kaip bus atliekamas skirstymas. Kiekvienos grupė savyje turi jai priklausančių įrenginių sąrašą.



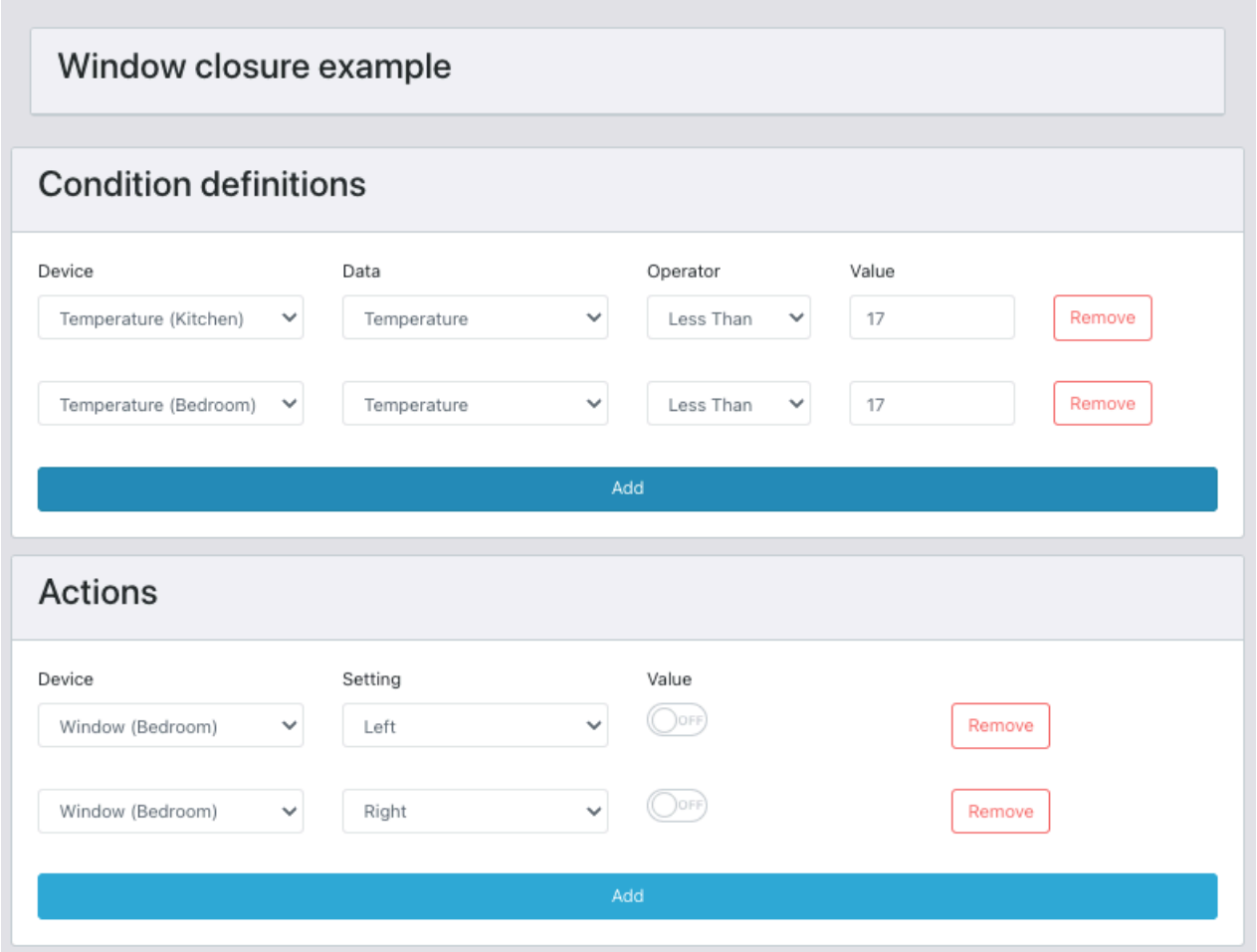
10 pav. Įrenginių grupės namo kambariams

Įrenginio peržiūros ir konfigūracijos lange galima matyti iš įrenginio atsiūstų duomenų grafikus. Konfigūruoti kaip gauti duomenys yra nuskaitomi ir kokie nustatymai galioja užregistruotam įrenginiui.



11 pav. Vieno įrenginio konfigūracija ir duomenų peržiūra

2.7. Automatizuoto valdymo konfigūracijos langas

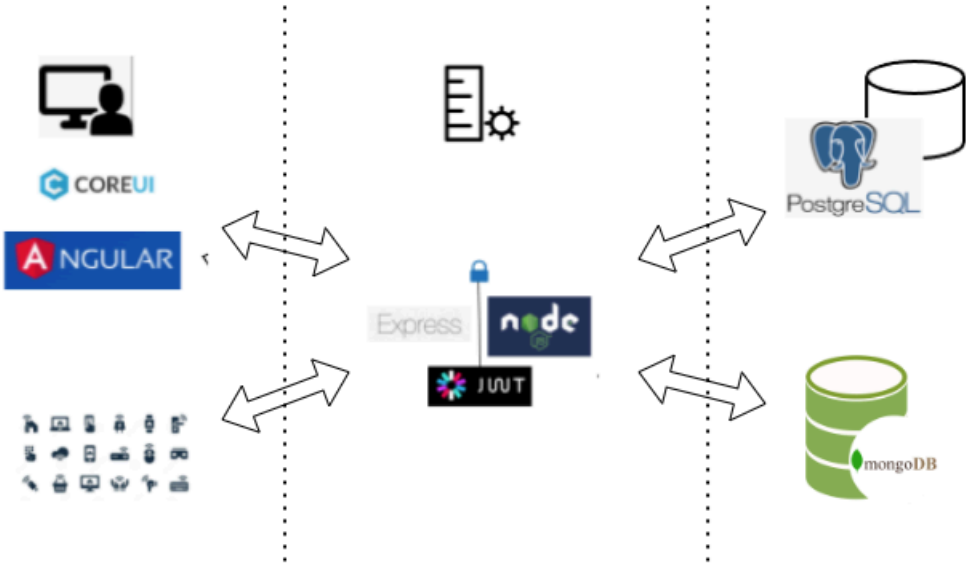


12 pav. Valdymo taisyklės aprašas: Sąlygos ir veiksmai

Valydmo taisyklės pavyzdyje pateikiamas scenarijus - dviejuose kambariuose fiksuojama temperatūra. Jei abu sensoriai praneša, kad temperatūra žemesnė, nei nurodyta sąlygoje, vykdomi taisyklėje apibrėžti veiksmai. Languose esantys valdikliai uždaro langus. *Sąlygos* aprašas nurodo įrenginį, jo siunčiamų duomenų nuorodą, loginį operatorių palyginimui atlikti ir reikšmę, su kuria yra lyginimas paskutinis įrenginio atsiųstas duomenų vienetas. *Veiksmas* aprašomas nurodant įrenginį, jo konfigūracijos lauką ir naują jo reikšmę. Vienoje taisyklėje galima apibrėžti daugiau nei vieną sąlygą ir daugiau nei vieną veiksmą. Taisyklės veiksmai vykdomi jei visos apibrėžtos sąlygos yra patenkintos.

3. Technologijos ir įgyvendinimas

3.1. Technologijos



13 pav. Sistemos komponentai

- *Angular* - TypeScript programavimo kalbos karkasas WEB vartotojo sąsajos kūrimui.
- *Core UI* - naudojimui paruoštų Angular komponentų ir CSS stilių rinkinys.
- *Node JS* - serverio programinės įrangos vykdymo aplinka(programavimo kalba - Javscript).
- *Express* - biblioteka skirta REST tipo Web servisams kurti.
- *JWT* - autentifikacijos biblioteka.
- *Postgres SQL* - registruoti įrenginiai, sistemos vartotojai ir įrenginių konfigūracijos saugomos reliacinėje duomenų bazėje.
- *Mongo DB* - įrenginių siunčiami duomenys saugojami dokumentinėje duomenų bazėje. [Hem16; Nad17]

3.2. Vartotojo sąsaja

3.2.1. Angular

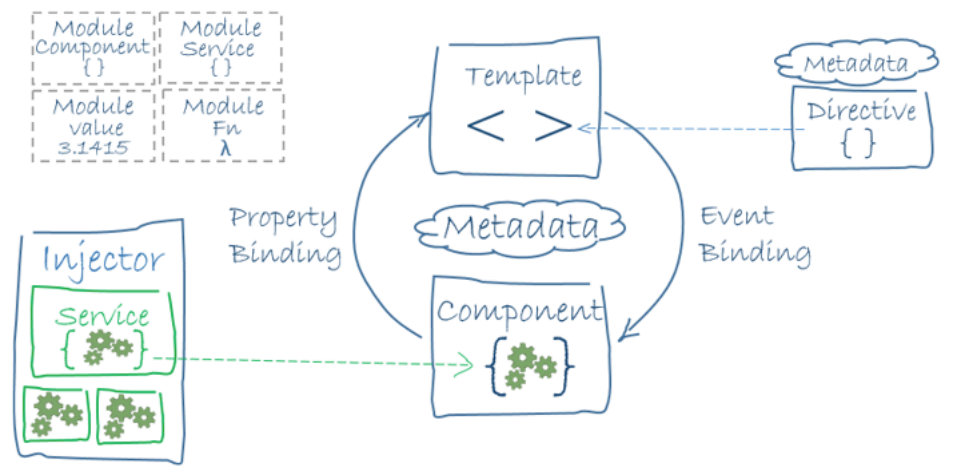
Angular yra platforma ir sistema, skirta kurti vieno puslapio kliento programas naudojant HTML ir TypeScript. Angular parašyta TypeScript programavimo kalba. Jis įgyvendina pagrindines ir pasirenkamas funkcijas kaip „TypeScript“ bibliotekų rinkinį. Karkaso architektūra remiasi tam tikromis pagrindinėmis sąvokomis. „TypeScript“ bibliotekos sudarančios Angular karkasą vadinamos

Moduliais(Module). Moduliai surenka susijusį kodą į funkcinius rinkinius; programa apibrėžiama modulių rinkiniu.

Programa visada turi bent vieną(šakninį) modulį, kuris įgalina įkrovą, o dažnu atveju, Angular aplikacijos turi daug daugiau funkcijų modulių. Jie leidžia programuotojui rašyti modulines programas ir pernaudoti kodą vietoje dublikavimo. *Komponentai(Component)* apibrėžia rodinius, kurie yra ekrano elementų rinkiniai, kuriuos Angular gali pasirinkti ir modifikuoti pagal kuriamos programos logiką ir duomenis.

Komponentai naudoja *Servisus(Service)*, kurie teikia specifines funkcijas, tiesiogiai nesusijusias su vaizdu, kurį mato vartotojas. Servisai gali būti įtraukti į komponentus kaip priklausomybės, todėl kodas tampa modulinis, pakartotinai naudojamas ir efektyvus. Kiekvienas komponentas turi sau priskirtą dali HTML teksto dalį kuri atvaizduojama vartotojo sąsajoje, jie vadinami *Šablonais(Template)*. Šablonas sujungia įprastą HTML su Angular direktyvomis, leidžiančiu modifikuoti HTML prieš pateikiant jį rodyti. Arba keisti matomus duomenis aplikacijos naudojimo metu.

Programos komponentai paprastai apibrėžia daugybę rodinių(*Views*), išdėstytų hierarchiškai. „Angular“ teikia maršrutizatoriaus(*Router*) paslaugą, kuri padės nustatyti naršymo kelius tarp vaizdų. Maršrutizatorius suteikia sudėtingas naršymo naršyklėje galimybes.



14 pav. Angular konceptas išsidėstymas

Angular aplikacijos kodas yra vykdomas vartotojo kompiuteryje, naudojamos naršyklės Javascript interpretatoriuje. Taigi galutinis HTML išeities tekstas yra sukuriamas dinamiškai, kliento mašinoje. Vartotojo sąsajos prasme, serverio dalis sistemoje yra atsakinga tik už Angular aplikacijos kodo pateikimą statinių failų pavidalu. Angular *Service* klasės atlieka HTTP kreipinius į sistemos WEB servisą siekiant atlikti prisijungimą prie sistemos arba gauti vartotojo pageidaujamus duomenis.

3.2.2. Angular Reactive forms

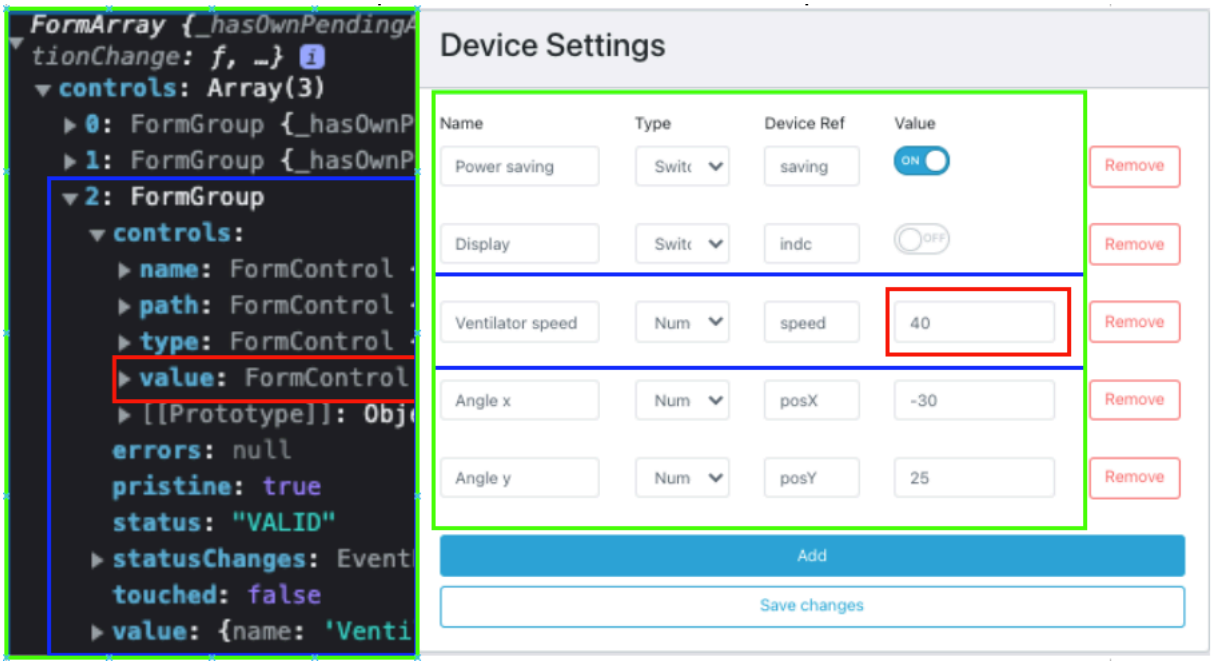
Tai vienas iš Angular karkaso modulių leidžiančių dirbti su HTML formomis. Angular reaktyviosios formos laikosi modeliu pagrįsto metodo formų įvesties apdorojimui, kurios vertės laikui bėgant gali būti keičiamos. Tai taip pat žinoma kaip modeliu pagrįstos formos. Reaktyviosiose formose galime sukurti ir atnaujinti savo paprastą formos įvedimo lauką, naudoti kelis įvedimo elementus grupėje, atlikti formos reikšmių validavimą, įdiegti sudėtingesnes formas, dinamiškai pridėti ir šalinti formos įvedimo laukus.

Reaktyviosiose formose naudojamas aiškus ir nekintamas būdas valdyti formos būseną tam tikru momentu. Kai pakeičiame formos būseną, ji grąžina naują būseną, kuri valdo modelių vientisumą tarp pakeitimų. Naudojant ši modulį, komponento klasėje sukuriame formą atspindintį kintamąjį, kurio pagalbą galima sekti formos būseną.

Modulio interfeisas susideda iš keleto klasių: [Doc21]

Klasė	**Paskirtis**
AbstractControl	Abstrakčioji bazinė klasė formų valdymo klasėms FormControl, FormGroup ir FormArray. Tai suteikia jiems bendrą elgesį ir savybes.
Form Control	Tvarko individualaus formos įvedimo lauko reikšmę ir validumą. Tai atitinka HTML formos įvedimo lauką, pvz., <input> arba <select>
Form Group	Tvarko „AbstractControl“ egzempliorių grupės reikšmę ir įvesties validumą. Grupės ypatybės apima jos antrinius valdiklius(esančius grupėje). Komponento aukščiausio lygio forma yra FormGroup klasės egzempliorius.
Form Array	Tvarko skaitiniu būdu indeksuoto AbstractControl egzempliorių masyvą. Ši klasė leidžia vykdomo metu į formą pridėti naujus įvedimo laukus, arba įvedimo laukų grupes. Naudojantis šia klase galima sukurti dinaminę įvedimo formą, kurios struktūrą iš dalies gali apibrėžti pats vartotojas.
FormBuilder	Injekcinė paslauga, teikianti gamyklinius valdymo egzempliorių kūrimo metodus.

Pateiktame pavyzdyje(7 pav.) matome TypeScript objekto reikšmę ir jį atitinkančą HTML formą. Pasikartojančios eilutės ir saugomos kaip *FormArray* sąrašo nariai. Vartotojas pasirintinai gali pridėti naują arba pašalintį norimą įvesties eilutę. Kiekviena eilutė savaime priklauso klasei *FormGroup*, kadangi savyje turi keletą įvedimo laukų. Kiekvienas įvedimo laukas atspindimas *FormControl* klasės objektu.



15 pav. Dinaminės formos pav.

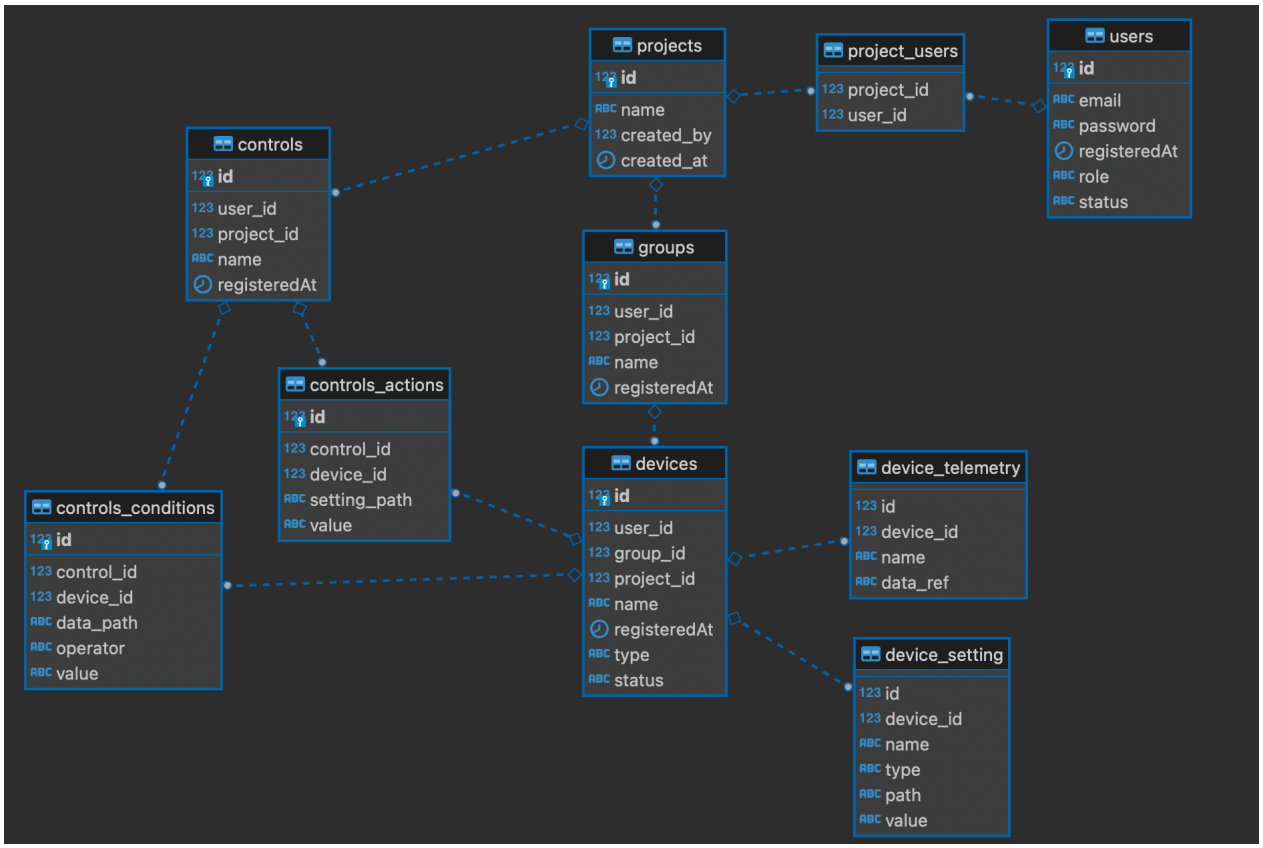
3.2.3. Core UI

Naudojimui paruoštų Angular komponentų ir CSS stilių rinkinys. Tai leidžia pagreintinti grafinės aplikacijos dalies kūrimą. Kadangi šios sistemos kūrimo metu nėra griežtai aprašytų dizaino reikalavimų, taip pat siekiant supaprastinti vartotojo sąsajos grafinės dalies uždavinius naudojame Core UI siūlomus HTML ir CSS pavydžius:

- HTML elementų pozicionavimas ir išdėstymas naršyklės lange.
- Spalvų deriniai.
- Formų ivedimo laukai, mygtukai, šoninė navigacijos juosta.
- Prisijungimo ir navigacijos langai.

3.3. Serverio programinė įranga

3.3.1. ER Diagrama



16 pav. RDBMS schema

3.3.2. Express

„Express“ yra populiariausia „Node JS“ biblioteka skirta API ir WEB servisų kūrimui. Express naudojami ir daugelio kitų NPM bibliotekų kaip sudedamoji dalis. Jame numatyti mechanizmai:

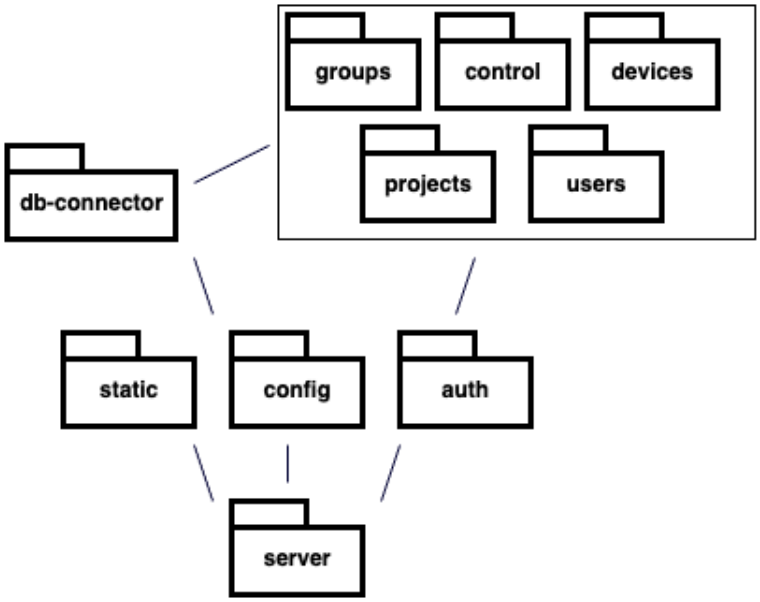
- Nesudėtingai aprašyti serverio maršrutus(*Routes, Endpoints*) ir su jais susieti atitinkamas apdorojimo funkcijas.
- Pridėti papildomas funkcijas prie vieno resurso arba jų grupės. Tai įgalina moduliarumą, visišką laisvę pasirinti kaip bus vykdoma užklausų autentifikacija, kokiu formatu į WEB servisą atkeliaus duomenys ir kaip bus suformuotas atsakymas.

```
const express = require('express');
const app = express();
app.get('/', (req,res)=>{
  res.send('Hi, your request has been received')
});
```

```
app.listen(2000, ()=>{
  console.log('listening at http://localhost:2000')
})
```

3.3.3. Serverio programos stuktūra

- *server* - programos įeities taškas.
- *static* - vartotojo sąsajos statinių failų pateikimas į kliento kompiuterį.
- *auth* - vartotojo prisijungimo, atsijungimo ir authentifikacijos mechanizmas, paremtas JWT.
- *config* - programos konfigūracija - duomenų bazės prisijungimo slaptažodžiai. JWT privataus ir viešo rakto reikšmės, WEB serverio portas.
- *devices, groups, projects, users* - sistemos esybių logika - kiekvienas paketas savyje turi HTTP užklausas apdorančias funkcijas, taip pat generuoja esybei aktualias užklausas į paketą db-connector.
- *control* - Įrenginių siunčiamų signalų ir duomenų apdorojimas.
- *db-connector* - Prisijungimas prie Postgres duomenų bazių valdymo sistemos. [Kup19]



17 pav. Serverio dalies Node JS paketai

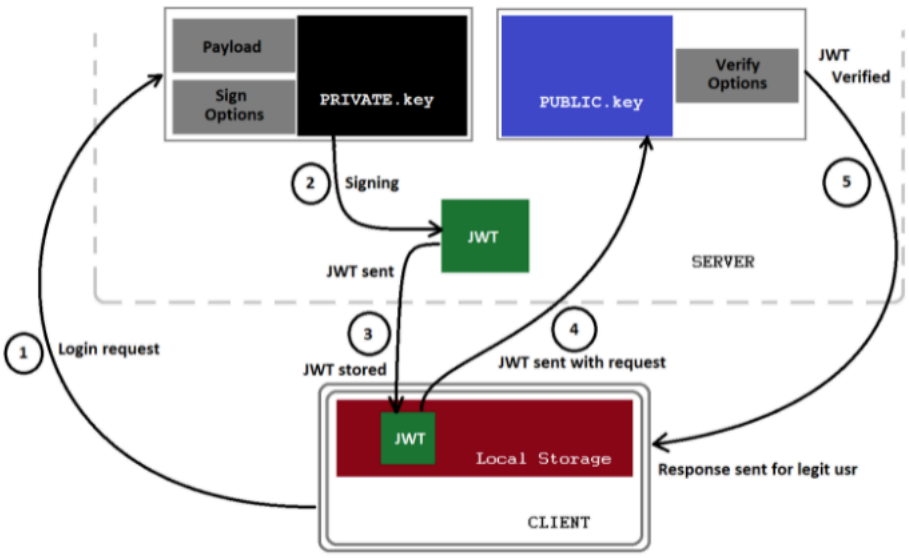
Pasirinkimas serverio dalį igyvendinti Node JS technologijomis suteikė galimybę rinkis iš daug NPM repositorijose viešai prieinamų atviro kodo bibliotekų. Node JS ekosistemoje esama daugybė

kiekvienai sistemos daliai įgyvendinti reikalingų bibliotekų. (Web servisai, autentifikacija, duomenų bazės prieiga). Kita vertus, kai pasirinkimas platus, prieš įtraukiant bibliotekas į projektą svarbu įsitikinti, kad biblioteka atliks tai ko tikimasi. Kadangi Node JS bibliotekų repositorijos yra viešos jose galima rasti ir įrankių kurie nebūtinai veiks korektiškai arba yra pritaikyti tik bendriniams panaudojimo atvejams, nesuteikdami mūsų projektui reikalingo funkcionalumo.

3.3.4. Autentifikacija naudojant JWT

JWT(angl. Json web token) yra standartas, apibrėžiantis kompaktišką ir savarankišką būdą, kaip saugiai perduoti informaciją tarp kliento ir serverio kaip JSON objektą. Dėl kompaktiško dydžio žetonus lengva perduoti per URL, POST parametrą arba HTTP antraštėje. Be to, kadangi jie yra savarankiški, juose yra visa reikalinga informacija apie vartotoją, todėl duomenų bazės nereikia užduoti daugiau nei vieną kartą. JWT esančia informacija galima pasitikėti, nes ji yra skaitmeniniu būdu pasirašyta naudojant slaptą arba viešųjų / privačių raktų porą. [Cho18]

1. Registruotas vartotojas pateikia prisijungimo užklausą.
2. Jei vartotojas registruotas sistemoje ir paiekia teisingą slaptažodį, jam sugeneruojamas naujas JWT.
3. JWT siunčiamas į vartotojo kompiuterį ir išsaugomas naršyklės sesijos atmintyje.
4. Kiekviena duomenų užklausą į WEB servisą siunčiama kartu su šiuo JWT. Serveris, prieš įvykdydamas užklausą patikrina žetono validumą ir galiojimo laiką.
5. Jei pirmieji žingsinai įvykdomi sėkmingai, WEB serviso metodai vartotojui prieinami.



18 pav. JWT autentifikacijos schema

Išvados

1. Informacinė sistema yra tinkama skirtingų tipų ir paskirties įrenginių administravimui ir valdymui. Suprojektuotas ir realizuotas sprendimas leidžia vartotojams apibrėžti, kokie duomenys ir konfigūracija priskiriami kiekvienam prietaisui.

2. Angular karkaso ir Core UI bibliotekos kombinacija pagreitino ir supaprastino vartotojo sąsajos programavimo darbus. *Angular Reactive Forms* taikymas palengvino dinaminių įvedimo formų kūrimą, o *Core UI* elementai leido greitai įgyvendinti patrauklią vartotojo sąsają.

3. Pasirinkimas naudoti *Node JS* serverio pusėje suteikė galimybę naudotis atviro kodo *NPM* bibliotekomis. **postgres**, **mongodb** - prisijungimui į duomenų bazes, **express** - WEB serviso realizacijai, **passport-jwt** - authentifikacijai. *Node JS* bendruomenė yra didelė, bibliotekų skaičius lenkia kitas programavimo kalbas, gausu informacijos internete. Dėl šių priežasčių, buvo galima daugiau dėmesio sutelkti į informacinės sistemos algoritmų kūrimą, o ne į techninius uždavinius.

4. Documentinė *MongoDB* duomenų bazė skirta prietaisų duomenims laikyti užtikrina greitą duomenų saugojimą, kadangi, skirtingai nei reliacinėse duomenų bazėse nevykdomi išorinių raktų ir duomenų schemos tikrinimai.

5. Ateityje atsiradus poreikui, *MongoDB* gali būti horizontaliai plečiama į keletą serverių naudojantis *Sharding* galimybėmis.

6. Tolimesniam vystymui galima svarstyti duomenų analitikos, anomalijų atpažinimo, vizualizacijos, valdymo taisyklių validavimo, papildomų protokolų palaikymo sprendimus.

Conclusions

1. Informational system is capable of administating and managaing IoT devices of a different kinds and purposes. Planned and implemented solution allows users to define themselves what data schema and configuration values are applicable for every device.

2. Angular framework and Core UI template collection has reduced frontend delivery time for a good measure. Angular Reactive Forms allowed to easily develop dynamic user forms. Core UI elements gave interface attractive appearance out of the box.

3. Node JS as a backend implementation language gave access to large number of open source *NPM* packages. **postgres**, **mongodb** - connections to databases, **express** - http web server, **passport-jwt** - authentication. Node JS has large and active community, available package and library counts exceeds any other modern programming language. It is easy to find documentation and information on the net. These reasons made technical objectives a bit easier and let developer focus more on informational system features.

4. Document type Mongo DB responsible for collected device data storage ensures high data throughput because of the swift nature of Mongo DB technology. As opposed to relational databases, Mongodb queries does not perform any foreign key checks, neither data schema validation.

5. Usage of Mongo DB leaves horizontal scaling optional available through the feature of Sharding.

6. Further development considerations include data analytics, detection of anomalies, data visualisation, rule contradiction validation, and additional device protocol support.

Literatūra

[Bas21] Shared Knowledge Base. Json web token (jwt) — the right way of implementing, with node.js. https://lt.nomuwiki.com/104673-internet-of-things-WMHYJU, 2021.

[Bun18] Jonas Bunevičius. Kas tai yra daiktų internetas iot ir ką reikia žinoti norint juo saugiai naudotis. http://www.technologijos.lt/n/technologijos/it/S-67862/straipsnis/Kas-tai-yra-daiktu-internetas-IoT-ir-ka-reikia-zinoti-norint-juo-saugiai-naudotis, 2018.

[Cho18] Siddhartha Chowdhury. Json web token (jwt) — the right way of implementing, with node.js. https://siddharthac6.medium.com/json-web-token-jwt-the-right-way-of-implementing-with-node-js-65b8915d550e, 2018.

[Doc21] Angular Documentation. Angular reactive forms. https://angular.io/guide/reactive-forms, 2021.

[Hem16] T. Santhanakrishnan Hema Krishnan M.Sudheep Elayidom. Mongodb – a comparison with nosql databases. https://www.researchgate.net/publication/327120267_MongoDB_-_a_comparison_with_NoSQL_databases, 2016.

[Kav18] Rishika Mehtaa KavitaKhannac Jyoti Sahnib. Internet of things: vision, applications and challenges. https://www.sciencedirect.com/science/article/pii/S1877050918307749, 2018.

[Kup19] Walter Kuppens. Connecting to postgresql with node.js. https://www.thisdot.co/blog/connecting-to-postgresql-with-node-js, 2019.

[Mik19] Sachin Kumar Mikhail Zymbler Prayag Tiwari. Internet of things is a revolutionary approach for future technology enhancement: a review. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0268-2, 2019.

[Nad17] Leonardo Mostarda Nadeem Qaisar Mehmood Rosario Culmone. Modeling temporal aspects of sensor data for mongodb nosql database. Journal of Big Data, 2017.