LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

VIEDĀS MĀJAS MODULĀRA SISTĒMA UN PROTOTIPA IMPLEMENTĒŠANA

BAKALAURA DARBS

Autors: **Iļja Gubins**
Studenta apliecības Nr.: 11075
Darba vadītājs: Asoc. prof. Dr. dat. Uldis Straujums

RĪGA 2015

Anotācija

Viens no uzdevumiem lietu internētā ir mājas automatizācija. Šajā darbā ir izanalizētas šobrīd eksistējošas gudras mājas sistēmas un, balstoties uz gala lietotāju vajadzībām, tiek izstrādāts prototips oriģinālai viedās mājas sistēmai. Gudra māja ir centralizēta kontroles sistēma, kas ļauj efektīvi pārvaldīt rīcībā esošas ierīces. Sistēma realizēta praksē ar tādu plašu tehnoloģiju spektru kā *Raspberry Pi* un *Arduino Uno* iegultās sistēmas, *nRF24L01* bezvadu komunikācijas modulis, *C++* un *Node.js* programmēšanas valodas, un tās darbība ir pārbaudīta simulācijās un reālos dzīves apstākļos. Sistēmas pārvaldīšana iespējama caur internetu, gan no personālā datora, gan no mobilās ierīces.

Atslēgvārdi: mājas automatizācija, viedā māja, gudrā māja, bezvadu sensoru tīkli, iegultās sistēmas.

Abstract

”Modular smart home system and prototype implementation”

One of the the main problems of the internet of things is home automation. This work presents analysis of currently existing smart home systems and development of original smart home system according to end-users’ needs. Smart home is a centralized control system that allows effective administration of existing devices. System is implemented with wide range of technologies, such as *Raspberry Pi* and *Arduino Uno* embedded devices, *nRF24L01* wireless communication module, *C++* and *Node.js* programming languages, and it has been tested both in simulations and in real life situations. System is controlled via the internet and is accessible from personal computers and mobile devices.

Keywords: home automation, smart home, wireless sensor networks, embedded systems.

SATURS

APZĪMĒJUMU UN JĒDZIENU APRAKSTS

CSRF - *Cross-Site Request Forgery*, starplapu pieprasījumu viltošana.

CSS - *Cascading Style Sheets*, kaskādisku stilu saraksts, tiek izmantots, lai definētu kā atsevišķas birkas tiks attēlotas lapā.

GET - *HTTP* pieprasījuma tips, kad parametri ar vērtībām tiek norādīti, kā vietrāža sastāvdaļa.

HTTP - Hiperteksta transporta protokols. Tīkla Internet standartprotokols, kas nodrošina informācijas apmaiľu globālajā tīmeklī.

IEEE - *Institute of Electrical and Electronics Engineers*, starptautiska nekomerciāla speciālistu asociācija, vadošā organizācija radioeleronikas un elektrotehnikas standartu izstrādē.

IoT - *Internet of Things*, lietu internets.

JS - *JavaScript*, programmēšanas valoda.

JSON - *JavaScript Object Notation*, datu apmaiņas formāts.

MVC - *Model View Controller*, Modelis-Skats-Kontrolieris izstrādes pamatprincips.

ORM - Objektu relāciju kartēšana. Programmēšanas tehnika, kas ļauj pārvērst datus, lai tos varētu lietot kā objektus objektorientētā programmēšanā.

POST - *HTTP* pieprasījuma tips, kad dati, visbiežāk no *HTML* formas, ievietoti pieprasījuma pamattekstā.

REST - *Representational State Transfer*, programmatūras arhitektūra stils, kas sastāv no vadlīnijām lai radītu pielāgojamiem interneta pakalpojumus.

SQL - *Simple Query Language*, visbiežāk tiek izmantota darbā ar datubāzēm, lai veiktu datu atlases atbilstoši nosacījumiem.

WS - *HTTP* pieprasījuma tips, kad dati, tiek ievietoti atsevišķā WebSockets paketā.

ATTĒLU SARAKSTS

TABULU SARAKSTS

IEVADS

Tehnoloģiju laikmetam ejot uz priekšu, informāciju sistēmas ieņem arvien lielāku un lielāku lomu mūsu dzīvēs. Viens no tehnoloģijas evolūcijas posmiem ir internets un tā loģiskā attīstība - lietu internets. Ar katru gadu pieslēgtu pie interneta ierīču ir ar vien vairāk un neizbēgami būs laiks, kad māja būs automatizēta un var būt kontrolēta ar vienu ierīci.

Ideja kontrolēt un pārraudzīt savu māju nav pārāk jauna vai oriģināla. Pirmie mēģinājumi tika dokumentēti un ļoti veiksmīgi izmantoti vēl 70s gados iepriekšējā gadsimtā. Kaut gan tehnoloģijas eksistē jau ilgu laiku, viedas mājas sistēmas ir pārāk dārgas un pagaidām, plašu cilvēku lokam nav pieejamas, bet kaut kādu daļu no tās jau ir iespējams izveidot un izmantot ikdienas dzīvē bez lielām izmaksām.

Šī darba izvirzītie mērķi ir:

1. aplūkot pašreizējo situāciju viedas mājas industrijā,
2. apzināt un analizēt eksistējošus risinājumus,
3. projektēt savu risinājumu,
4. izstrādāt sava risinājuma sistēmas prototipu,
5. ieskicēt tālākos darbus

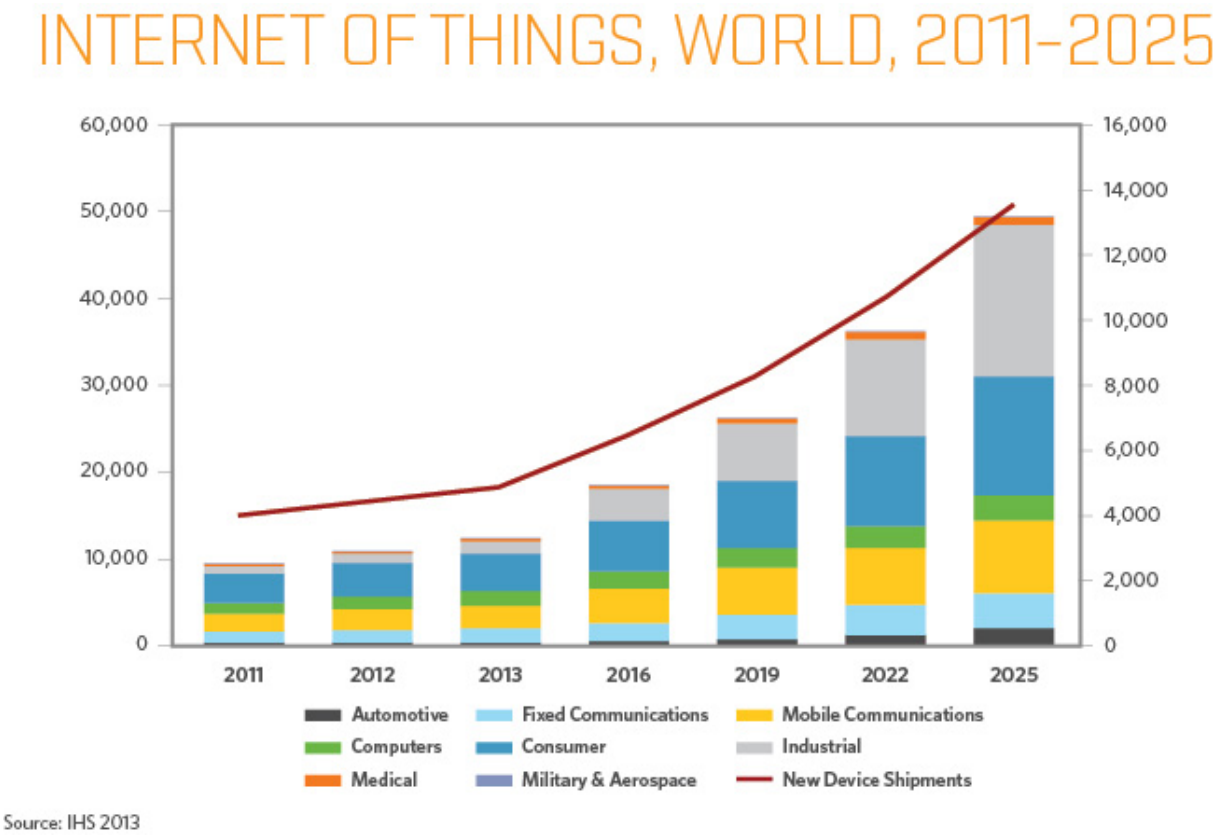
Darbs ir sadalīts vairākās nodaļās. Darba 1. nodaļā tiek aprakstīta esošā situācija un apskatīti eksistējošie risinājumi. 2. nodaļā ir projektēts savs risinājums un izstrādāts risinājuma prototips. 3. nodaļā ir ieskicēts tālākais darbs. Dokumentā arī ir darba secinājumi un divi pielikumi.

1. PAŠREIZĒJĀ SITUĀCIJA

Nodaļā ir apskatīta pašreizējā situācija gudras mājās industrijā.

1.1. Lietu interneta un viedās elektrotehnoloģijas attistība

Interneta attīstība joprojām turpinās: pirms divdesmit pieciem gadiem tajā bija apvienoti aptuveni tūkstoš resursdatoru, un kopš tā laika tas ir nepārtraukti paplašinājies, patlaban ar datoriem un mobilām iekārtām veidojot saikni starp miljardiem cilvēku [1]. Svarīgs nākamais interneta attīstības posms ir tā pakāpeniska attīstīšanās no savstarpēji savienotu datoru tīkla savstarpēji savienotu objektu tīklā, piemēram, grāmatu, automobiļu, elektroiekārtu un pārtikas produktu tīklā, tādējādi veidojot lietisko internetu. Katrā ziņā, kā ir redzams attēlā 1.1, lietisku internetu attīstībai ir prognozēta gaiša nākotne.



1.1. att. **Lietu interneta attīstības prognoze**

IoT ir nevis viengabalains, bet gan visaptverošs risinājums, kurā ietvertas dažādas tehnoloģijas, sistēmas un lietojumi, kas tiek nepārtraukti attīstīti. Pagaidām vēl nav absolūti taustāma realitāte, bet drīzāk perspektīva redzējums par virkni tehnoloģiju, kuras, tās apvienojot, nākamajos 5 līdz 10 gados [2] varētu būtiski mainīt sabiedrības funkcionēšanu.

1.2. Esošo sistēmu risinājumu analīze

Šajā apakšnodaļā es gribu apskatīt esošos risinājumus un izveidot salīdzinājumu tabulu, kur būs apvienota visa informācija par protokoliem.

1.2.1. Esošo protokolu analīze

Eksistē daudzi bezvadu komunikācijas protokoli, bet tikai daži ir piemēroti gudras mājas komunikācijai.

1.2.1.1. X10

X10 ir viens no pirmajiem protokoliem un bija izveidots 70os gados. Pirmā protokola versija bija tikai caur mājās elektrolīniju un tikai pēc tam, bezvadu [3] . *X10* ir zināms kā protokols ar vismazāko ātrumu un vismazāk iespējam. *X10* tehnoloģija jau novecoja un ir ļoti ieteicams instalēt kaut ko saderīgāku ar jaunākiem bezvadu standartiem, jo to, *X10*, ar katru gadu ir grūtāk un grūtāk uzstādīt. Bet *X10* ir joprojām de-facto standarts [4] .

1.2.1.2. Z-Wave

Z-Wave ir bezvadu mājas automatizācijas protokols, kas strādā 908.42MHz frekvenču joslā. Tas ir salīdzinoši jauns mājas automatizācijas protokols, bet ir pieaudzis popularitātē diezgan strauji pēdējos gados. Grupa aiz tā protokola, *Z-Wave Alliance*, kas tagad apvieno vairāk nekā 1000 dažādas ierīcēs un 300 uzņēmumus [5] . Viena no galvenajām iezīmēm Z-Wave ir tā, ka tā izmanto mezglutīklu, kas būtībā nozīme, ka viens *Z-Wave* produkts suta informāciju caur citu, un tā tālāk, līdz tas sasniedz paredzēto galamērķi. Tam arī ir ļoti zems enerģijas patēriņš, kas ir ideāli piemērots ierīcēm, kas balstās uz akumulatora enerģiju.

1.2.1.3. ZigBee

ZigBee ir bezvadu sakaru *IEEE* izstrādāts standarts. Tas patērē ļoti mazu elektroenerģijas daudzumu un izmanto mezglutīklu, tāpat kā Z-Wave, bet ir nedaudz lētāks nekā *Z-Wave* [6] .

1.2.1.4. Salidzinājums

Lai būtu vieglāk salīdzināt, bija izveidota salīdzināšanas tabula 1.1.

1.1. tabula. **Esošo sistēmu risinājumu salīdzinājuma tabula**

Īpašība	**X10**	**ZigBee**	**Z-Wave**
Komunikācijas metode	Bezvadu, 433 MHz	Bezvadu, 2.4Ghz	Bezvadu, 868.42 MHz
Gads	1975	2005	2008
Protokols	Izmanto vienkāršu bitu struktūru lai pārsūtītu komandas. Nav pareizības pārbaudes.	Izmanto datu paketes līdzīgas Ethernetam. Katram paketēm ir datu pareizības pārbaude.	Atkarībā no pārraidīšanas režīma.
Ātrums	7kbit/s	250kbit/s	100kbit/s
Maks. ierīces vienā tiklā	256	64000	232
Traucējumu kompensācija	Nav	Ir	Ir
Uzticamība	Pārraidītam komandām nav apstiprinājumu.	Visām komandām ir saņemšanas apstiprinājumi.	Visām komandām ir saņemšanas apstiprinājumi.
Mezglutīkls	Nav	Ir	Ir
Drošība	Nav	Ir	Ir
Iebūvēta enerģijas taupīšana	Nav	Ir	Ir
Ierīces statusa apprasīšana	Nav	Ir	Ir

1.2.2. Esošo mājas sistēmu analīze

Ir iespējams nopirkt jau esošu mājas sistēmu, kur vienā komplektā ir iekļautas dažādas gudras mājas ierīces. Divas no tām tiek apskatītas nākamajās apakšnodaļās. Principā, problēma ar tām ir tikai viena - cena.

1.2.2.1. VeraLite Home Controller

Galvenās funkcijas [7]:

1. Attālinātā kontrolēšana no jebkuras vietas pasaulē ar interneta pieslēgumu
2. Sistēma atbalsta elektroenerģijas skaitītāju reģistrēšanu, kas ļauj labāk apzināt savu elektrības rēķinu
3. Sistēma atbalsta *IP* video novērošanu
4. Bez ikmēneša vai instalācijas maksas
5. Izmanto *Z-Wave* protokolu

Cena ir 175$, bet komplektā ir tikai bāzes stacija, papildus sensori maksā 20-50$ katrs.

1.2.2.2. SmartThings Know and Control Your Home Kit

Galvenās funkcijas [7]:

1. Attālinātā kontrolēšana no jebkuras vietas pasaulē ar interneta pieslēgumu
2. Sistēmā ir iebūvēta paziņojumu sistēma, kas strādā ne tikai ar viedtālruņiem, bet arī ar parastiem tālruņiem, ar SMS palīdzību
3. Paziņojumu sistēma ir viegli modificējamā caur lietotāju saskarni
4. Bez ikmēneša vai instalācijas maksas
5. Atbalsta *Z-Wave* un *ZigBee*

Cena ir 350$, komplektā ir 2 temperatūras sensori, 2 kustību sensori un viena gudrā kontaktligzda.

1.2.2.3. HomeSeer HS3

Galvenās funkcijas [8]:

1. Attālinātā kontrolēšana no jebkuras vietas pasaulē ar interneta pieslēgumu
2. Sistēmā ir iebūvēta paziņojumu sistēma, kas strādā ne tikai ar viedtālruņiem, bet arī ar parastiem tālruņiem, ar SMS palīdzību
3. Paziņojumu sistēma ir viegli modificējamā caur lietotāju saskarni
4. Sistēma atbalsta *IP* video novērošanu

5. Iebūvēta balsa atpazīšanas sistēma
6. Bez ikmēneša vai instalācijas maksas
7. Atbalsta *Z-Wave* un *X10*

Cena ir 250$, bet komplektā ir tikai bāzes stacija, papildus sensori maksā 20-50$ katrs.

2. PIEDĀVĀTAIS RISINĀJUMS

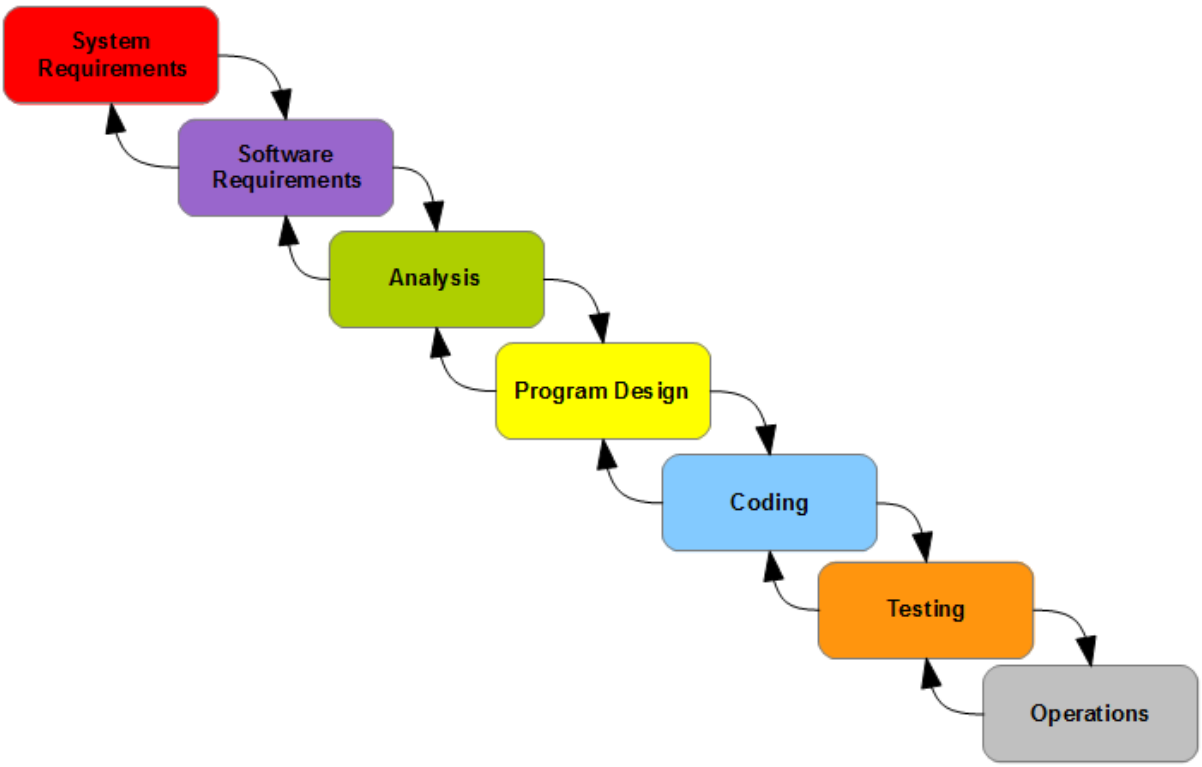
Salīdzinot esošos risinājumus, bija konstatēti trūkumi, tāpēc tika nolemts projektēt savu risinājumu un izstrādāt tā prototipu.

2.1. Ierobežojumi

Lai gan daži no ievadā minētajiem jautājumiem ir aplūkoti šajā pētījumā, autors ir informēts par dažiem ierobežojumiem. Viens no tiem ir datoraparatūras izvēle. Tā kā autoram nav piekļuves pie jau implementētas viedas mājās sistēmas, projekta prototips tiks īstenots uz atklātās pirmkoda aparatūras, ar kuru palīdzību tiek simulētā kontrolēta mājās apkārtne. Otrs svarīgais projekta ierobežojums ir projektam atvēlētais laiks. Projekta tēma tiek izvēlēta pēdējā semestra pirmajā nedēļā, kas atstāj četru mēnešu laika periodu uz projekta realizāciju un šī dokumentu sagatavošanu. Tā kā laiks ir ierobežots un sagatavot projektu pilnā apjomā (ar visiem iespējamiem paplašinājumiem) nav iespējams, bija nolemts strikti noteikt apjomu un termiņus. Lai kontrolētu laicīgu darba izpildīšanu, bija izmantota Ganta diagramma. Šo apstākļu dēļ nebija pietiekami daudz laika lai īstenotu visas idejas. Funkcijas, kas bija palikušas ārpus šī darba tiek iesniegtas kā ierosinājumi turpmākajam darbam 3. nodaļā.

2.2. Prototipa izstrādes metodoloģija

Lai nodrošinātu pastāvīgu progresu jebkurā darbā ir svarīgi noteikt metodoloģiju un piemērotu darba plānu sākotnēja stadijā. Visā šajā prototipa izstrādes laikā bija pieņemta metodika kas balstīta uz vadlīnijām izvirzītiem grāmatā *Engineering Real Time Systems – An Object Oriented Methodology using SDL*. Izmaksas un darba stundas var samazināt ja izstrāde tiek balstīta uz labi pārdomātu modeli. Izmantota metodoloģija, kura ir attēlotā attēlā 2.2, strikti atbalstās uz cieto plānošanu un projektēšanu pirms sistēmas izstrādāšanu [9].



2.2. att. **Agile Waterfall modelis**

Kaut gan sistēmas attīstības process izskatās diezgan līdzīgi ūdenskrituma modelim, ir būtiska atšķirība: iterācijas ar punktētām bultiņām. Iterācijas ļauj rīkoties ar neparedzētam lietām, kas notiks izstrādes procesā. Piemēram:

- Sistēmas prasību maiņa projekta attīstības laikā
- Izstrādātājs iegūst lielāku izpratni par to, kā sistēma darbojas
- Tiek atklāti defekti
- Tiek atklāti dizaina trūkumi
- Ir nepieciešama jau eksistējošo kodu optimizācija

2.2.1. Prasību noteikšana

Kā darba risināmu problēmu es uzskatu tieši profilu vadītu gudras mājas sistēmas piedāvāšanu, tāpēc sākotnējais uzdevums bija definēt kādu gudru mājas sistēmu ir nepieciešams izveidot. Lai to izdarītu bija definēti un aptaujāti gala lietotāji. Aptaujas jautājumus un rezultātus var apskatīt Pielikumā C - Aptaujas jautājumi un rezultāti.

Rezultātā ar iepriekš aptaujāto lietotāju sadarbību, bija modelētas prasību specifikācijas un bija sagatavoti dažādi dokumenti, kuri ir apskatīti nākamajās apakšnodaļās.

2.2.2. Funkcionalitātes analīze

Dokuments kurā ir apskatītas gudras mājas iespējas un definēšana, kuri ir vajadzīgi sistēmai. Funkcionalitātes analīze ir svarīgs plānošanas etaps.

Sensoru rādījumi

Tiem jābūt vērtību neatkarīgiem, t.i. pieņem jebkuras vērtības, jebkādā formātā: temperatūra, gaisa mitrums, apgaismības, utml. sensori.

Aktuatori

Jābūt iespējai atsūtīt signālu kas ieslēgs, izslēgs vai citādāk modificēs ierīces apakšsistēmu – piemēram, ja mezglam būs pievienota releja ar “gudru” lampu, jābūt iespējai ieslēgt, izslēgt, kā arī mainīt gaismas spožumu.

Notikumi

Sistēmai jāģenerē notikumi kas ir izdoti kad kaut kāds iepriekš definēts nosacījums tiek izpildīts. Piemērām, ja temperatūra pārsniedz 26C, ieslēdzas gaismas kondicionieris.

Vēsture

Visiem notikumiem un sensoru rādītajiem ir jābūt saglabātiem, ar visiem laika zīmogiem.

Vairāki lietotāji

Sistēmai jābūt vairāku lietotāju līmeņiem – administrators, kas var mainīt, piemērām, sensoru rādījumu intervālu un administrēt sistēmas mezglus; lietotājs, kas var apskatīt vēsture un arī izmantot mezglus; viesis, kas var apskatīt tiekošo situāciju tikai nekritiskiem elementiem.

Viegla jaunu ierīču instalācija

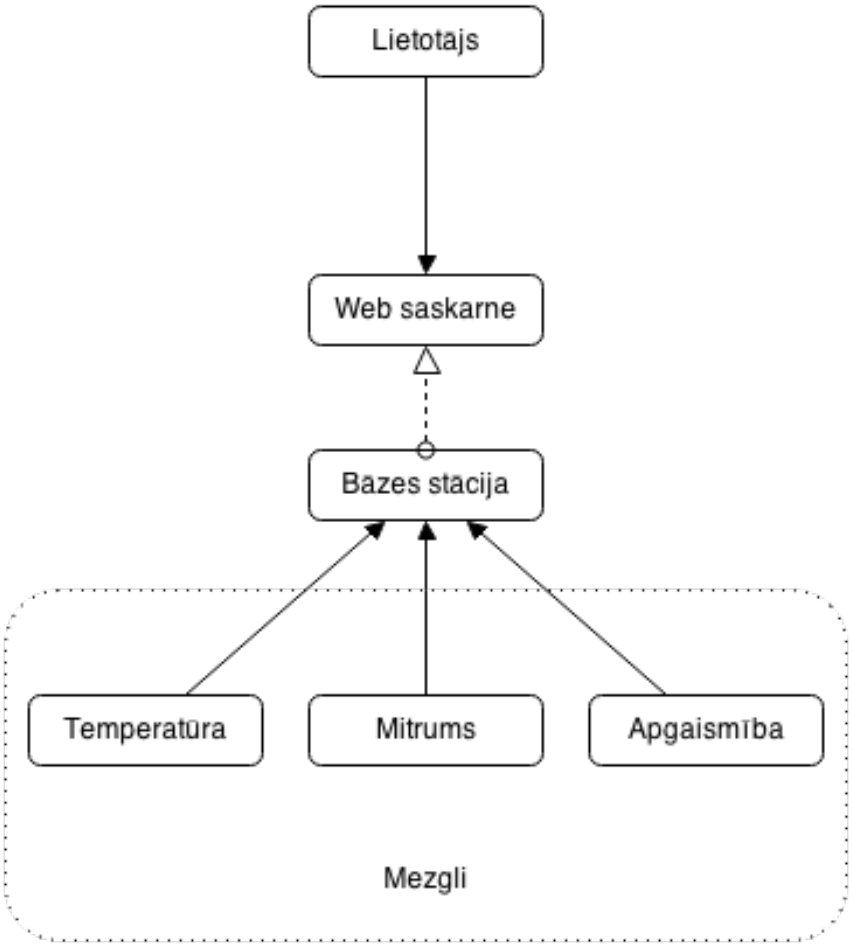
Jaunu ierīču pievienošanai jābūt vieglai un ātrai, optimāli – ieslēgt ierīci, saskarnē apstiprināt, ka tas ir tavs mezgls un sakonfigurēt to pēc vajadzībām.

Sistēma pieejama attālināti

Visām darbībām jābūt pieejamām arī attālināti, ne tikai vienā vietā vai datorā. Piemērām, internētā, privātā serverī, vai uz bāzes stacijām. Sistēmai arī jābūt ērti pieejamai no telefonā – vai nu speciālā lietojumprogrammatūrā, vai ar pārlūkprogrammu.

2.2.3. Augstāka līmeņa sistēmas modelis

Attēlu 2.3 var izmantot lai iegūtu izpratni par sistēmas sfēru.



2.3. att. **Augstāka līmeņa sistēmas modelis**

2.3. Tehnoloģiju salīdzinājums un izvēle

Šajā apakšnodaļā ir apskatītas iespējamās datoraparatūras un programmatūras tehnoloģijas, kas varētu būt izmantotas manā implementācijā, kā arī argumentācija to izvēlē.

2.3.1. Datoraparatūra

Sistēmas datoraparatūras izvēle ir svarīga, bet tomēr šajā gadījumā bija diezgan ierobežota, tāpēc bija izlemts izmantot tieši to, kas bija pieejams darba autoram: *Raspberry Pi* [10] vai *BeagleBoard Black* [11], *Arduino Uno* [12] un *nRF24L01* [13] radio moduļi.

2.3.1.1. Bāzes stacija

Kā jau bija iepriekš definēts, bāzes stacijai jābūt pietiekami jaudīgai, lai atbalstītu interneta pieslēgumu, datubāzi, datubāzes saskarnes, lietotāja saskarni serverī un jābūt iespējai kaut kādā veidā paplašināt šo iekārtu ar radio moduli. Protams, var izmantot tradicionālo personālo datoru,

bet tas ir ļoti neefektīvi, gan no enerģijas patēriņa, gan no resursu viedokļa. Visloģiskāk liekas izmantot vienas plates *ARM* arhitektūras datoru, kur ir pieejama paplašinājuma saskarne, vai arī *USB*, vai *GPIO*.

Potenciāli: *Raspberry Pi* vai *BeagleBoard Black*. Apskatīsīm šos divus tabulā 2.3.

2.3. tabula. **Raspberry Pi un BeagleBoard Black salīdzināšana**

Ipašība	**Raspberry Pi**	**BeagleBoard Black**	**Komentāri**
Procesors	ARM11 @ 700 MHz	Cortex A8 @ 1GHz	A8 ir apmērām 60% jaudīgāks nekā ARM11
GPU	VideoCore IV	PowerVR SGX530	VideoCore IV ir daudz jaudīgāks par PowerVR
RAM	512 MB SDRAM @ 400 Mhz	512 DDR3L @ 400 Mhz	DDR3L izmanto mazāk elektroenerģijas nekā citi
Datu krātuve	SD kartes slots	2GB eMMC + microSD slots	eMMC ir tik pat ātrs kā 10. klases SD
Ethernet	10/100M	10/100M	-
USB	2	2	-
Paplašinājumi	12 GPIO, USART, SPI, I2C, CSI, DSI	65 GPIO, SPI, I2C	Iekš BBB ir daudz vairāk GPIO pinus, bet Pi atbalsta vairākus perifērijas formātus
OS	ARMv6 operētājsistēmas, oficiāli – Debian un Arch Linux ARM	Jebkuras ARM operetājsistēmas, oficiāli - Ubuntu un Angstrom	-
Sabiedrības aktivitāte	Ļoti aktīva sabiedrība, populārākais ARM vienas plates dators	Ne pārāk aktīva sabiedrība, bet pārsvara sastāv no profesionāļiem	-
Cena	35$	200$	Gandrīz x6 atšķirība!

Neieskaitot to, ka priekš *Raspberry Pi* vēl ir vajadzīga ātrā SD kārte (10. klase), joprojām sanāk ļoti liela starpība cenā. Bija nolemts, ka pat *BeagleBoard Black* ir nedaudz labāks atsevišķos gadījumos, *Pi* vairāk atbilst iespējam un bakalaura darba kompetencei.

Izvēle: Raspberry Pi

2.3.1.2. Mezgli

Mezglam jābūt pietiekami energoefektīvam, pietiekami jaudīgam lai izmantotu radio moduļus, ar pietiekami daudziem piniem, lai pieslēgtu sensorus vai aktuātorus. Ērtākais veids būtu izmantot *Arduino* vai kaut kādu *Arduino* klonu. Par laimi man jau ir *Arduino Uno*, kas labi atbilst visam prasībām.

Izvēle: Arduino Uno

2.3.1.3. Radio moduļi

Radio modulim ir jāizmanto brīva frekvence, kas ir pieejama visiem; jābūt pievienotām pie izvēlētajām ierīcēm – *Arduino Uno* un *Raspberry Pi*. Sanāk tā ka man jau bija pieejami moduļi – *nRF24L01*. Par laimi modulim arī ir gatava bibliotēka.

Izvēle: nRF24L01

2.3.2. Programmatūra

Programmatūras izvēle arī ir svarīgā. Kaut gan jebkurā valodā ir iespējams uzprogrammēt gandrīz jebko, katrai valodai ir sava ”specializācija”, sava niša, ar kuru valoda strādā optimālāk. Tagad, kad pat iegultās ierīcēs operētājatmiņa ir skaitāma simtos megabaitos, nav vērts mikrooptimizēt prototipēšanas stadijā. Tāpēc katra izvēle tiek izdarīta ņemot vērā arī autora priekšroku.

2.3.2.1. Datubāze

Datubāze ir ļoti svarīga daļa, kas ir pamatelements gudras mājas sistēmai. Tā kā ir ļoti daudz datubāzes un gandrīz visas var būt izmantotas, bija nolemts vispirms izveidot prasības: Publish – Subscribe modeļa atbalsts, priekš reālā laika notikumiem un izmaiņām, un Raspberry Pi (ARMv6) atbalsts.

Pēc apspriedumiem, bija izvelēti sekojošie datubāzes varianti: *Redis* [14], *MongoDB* [15] un *PostgreSQL* [16]. Tie ir salīdzināti tabulā 2.3.

2.3. tabula. **Redis, MongoDB un PostgreSQL salīdzināšana**

Īpašība	**Redis**	**MongoDB**	**PostgreSQL**
Meta modelis	NoSQL	NoSQL	SQL
GPU	Ir	Manuāli kompilēt	Ir
ARMv6 atbalsts	Ir	Ir	Ir
Rezerves kopēšana	Ir	Ir	Ir
ACID atbalsts	Daļēji	Ir	Ir
Pub/Sub pattern	Ir	Ir	Ir
Oficiāli adapteri	Ir	Ir	Ir
Komentāri	Redis nav īsti datubāze, bet drīzāk key-value krātuve.	Katrs MongoDB ieraksts ir JSON objekts. Ir viegli mainīt iekšējo struktūru, bet ir vajadzīga papildus datu validācija.	Viena no progresīvākajām datubāžu sistēmām. Nav tik populāra salīdzinot ar citām SQL.

Izvēle: PostgreSQL

2.3.2.2. Lietojumprogrammas saskarne

Lietojumprogrammas saskarne būs "līme" starp visām darbībām un datubāzi. Ir ļoti daudz iespējas kā to realizēt, bet vajag arī domāt par resursu patēriņu, jo *Raspberry Pi* nevar, piemērām, izmantot pilnu *Java* virtuālo mašīnu, tikai *Embedded* versiju kā arī uz *Linux* nevar izmantot *.NET*. Potenciālas tehnoloģijas: *PHP*, *Go*, *Node.js*.

Go ir jauna programmēšanas valoda, kura mani interesē, bet tomēr tā ir pārāk jauna un nestabila. Daudz kas būs jāraksta no jauna, jo ne visas bibliotēkas ir pārnestas uz *Go*.

PHP ir skriptēšanas valoda, bet viņai ir mīnusi, kurus būs pārāk problemātiski atrisināt, piemērām, lai izmantotu *Websockets* uz *PHP*, vajag izveidot atsevišķu procesu kas būs atbildīgs par Websocket savienojumiem. *PHP* nevar strādāt ar savienojumiem pa taisno, vajag vēl vienu procesu, kas būs atbildīgs par to, piemērām, *Apache* vai *Nginx*.

Node.js ir pietiekami jauna un moderna, lai natīvi atbalstītu jaunākās tehnoloģijas (piemērām, *Websockets*, kas sistēmā ir vajadzīgi); ir ļoti ērta pakešu sistēma - *NPM*, kas atvieglo dzīvi un atļauj viegli un ātri instalēt jaunus apakšmoduļus; *Node* sabiedrība ir ļoti aktīva un ļoti daudz

entuziastu, kas veido daudz jaunus moduļus un ietvarus; *Node* izmanto tikai vienu pavedienu savam procesam un izmanto ļoti maz resursu (kas arī ir ļoti lietderīgs uz *Raspberry PI*) [17]. Tāpēc bija nolemts izmantot *Node*.

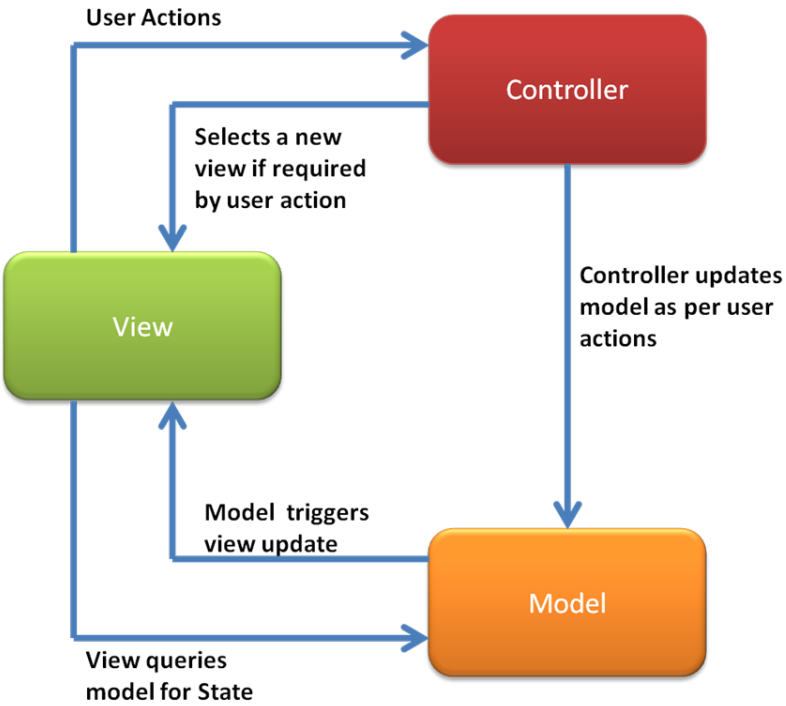
Nākamais solis bija izvelēties ietvaru priekš *Node.js*. Industrijas standarts ir *ExpressJS* kas ir diezgan elastīgs, bet minimālistisks [18]. Tas nav slikti, bet tomēr tāpēc, ka laiks ir ierobežots, bija nolemts izmantot *SailsJS* [19]. Ietvars ir bāzēts uz iepriekš minēto *Express*, bet ir iepakots ar daudzām iespējam priekš reālā laika komunikāciju.

Izvēle: *Node.js* kopā ar *SailsJS* ietvaru ar kuru palīdzību būs izveidota *REST* lietojumprogrammas saskarne.

2.3.2.3. Lietotāju saskarne

Tā kā bija nolemts izmantot *Node.js* un *SailsJS* ietvaru, visātrākais veids izveidot tīmekļa lapu bija izmantot *SailsJS* iebūvēto *MVC* sistēmu. Tā satur 3 daļas (skatīt vairāk attēlā 2.4) [20]:

1. Model – satur aplikācijas datus un likumsakarības starp aplikācijas stāvokļiem
2. View – attēlo lietotāja interfeisu un aplikācijas stāvokli
3. Controler – pārrauga ieejas datus, lai mainītu aplikācijas stāvokli



2.4. att. **MVC modelis**

Izmantojot iebūvētu sastatnes komandu *sails generate* var ļoti ātri izstrādāt sistēmas daļas [21].

2.3.2.4. Radio moduļi

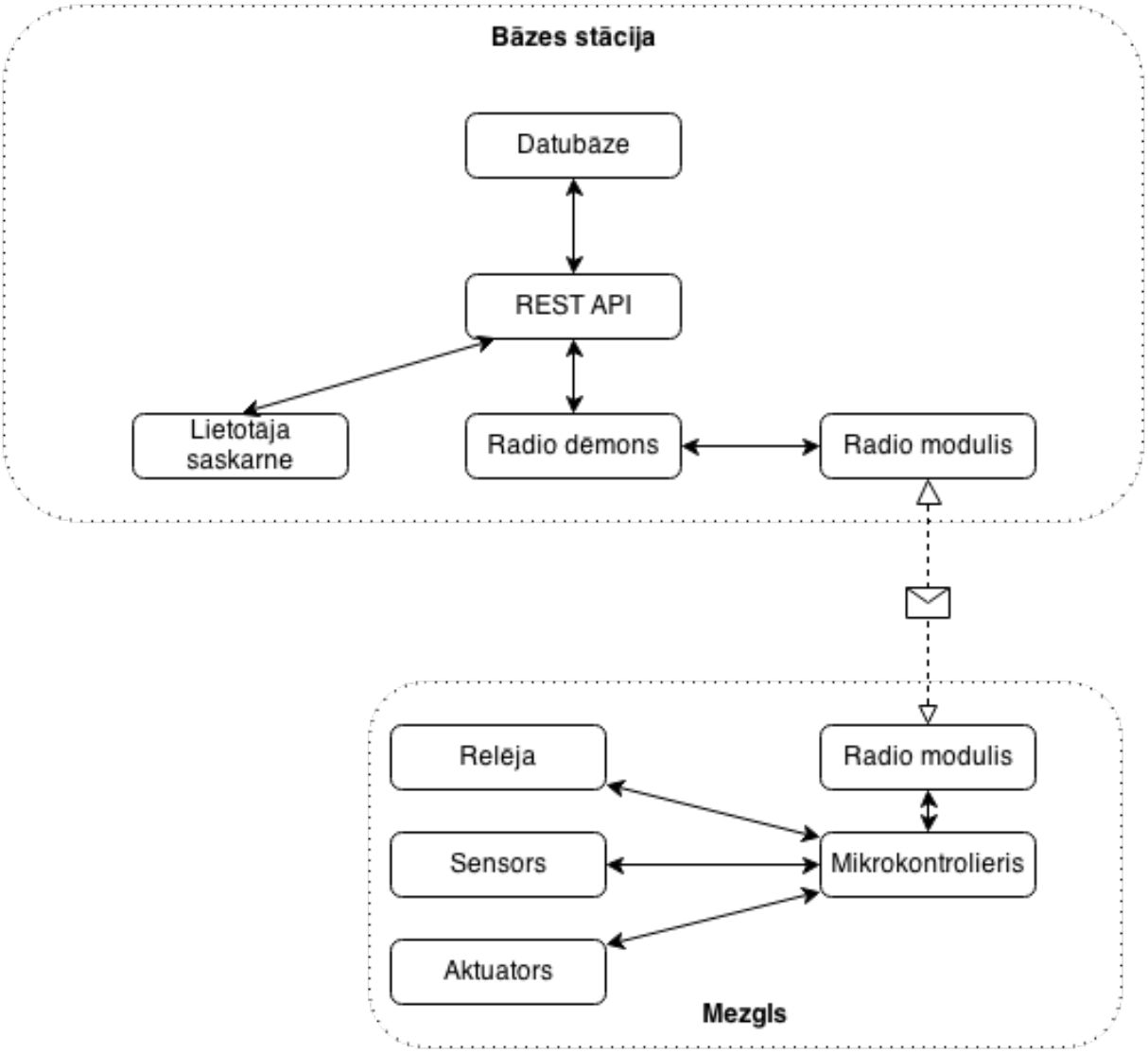
Par laimi, izvelētajam radio modulim *nRF24L01* jau ir izveidotas stabilas bibliotēkas – *RF24* [22] un *RF24Network* [23]. Tās atļauj izveidot stabilu un izturīgu mezglu tīklu.

2.3.2.5. Radio dēmons

Principā, visām programmēšanas valodām ir piekļuve pie zemā līmeņa procesiem (t.sk. *GPIO I/O*), tāpēc šeit izvēle ir pilnīgi personīga. *RF24* un *RF24Network* sākotnēji ir uzrakstīti uz *C++*, tāpēc bija nolemts, ka visērtāk būtu izmantot *C++* arī dēmonu izstrādei.

2.4. Komunikācija starp sensoru mezglu un bāzes stacijas

Viens no uzdevumiem bija izstrādāt komunikācijas protokolu, kas būs pietiekami ērts izmantošanai un tajā pašā laikā pietiekami kompakts. Iespējamus sistēmas galapunktus var atrast 1. pielikumā, ”Pielikums A. Lietojumprogrammas saskarnes galapunkti”. Savukārt, diagrammā 2.5 ir parādīts kā tieši notiek komunikācija starp sensoru mezglu un bāzes staciju.



2.5. att. **Komunikācija starp sensoru mezglu un bāzes staciju**

2.5. Tīmekļa lietojumprogramma

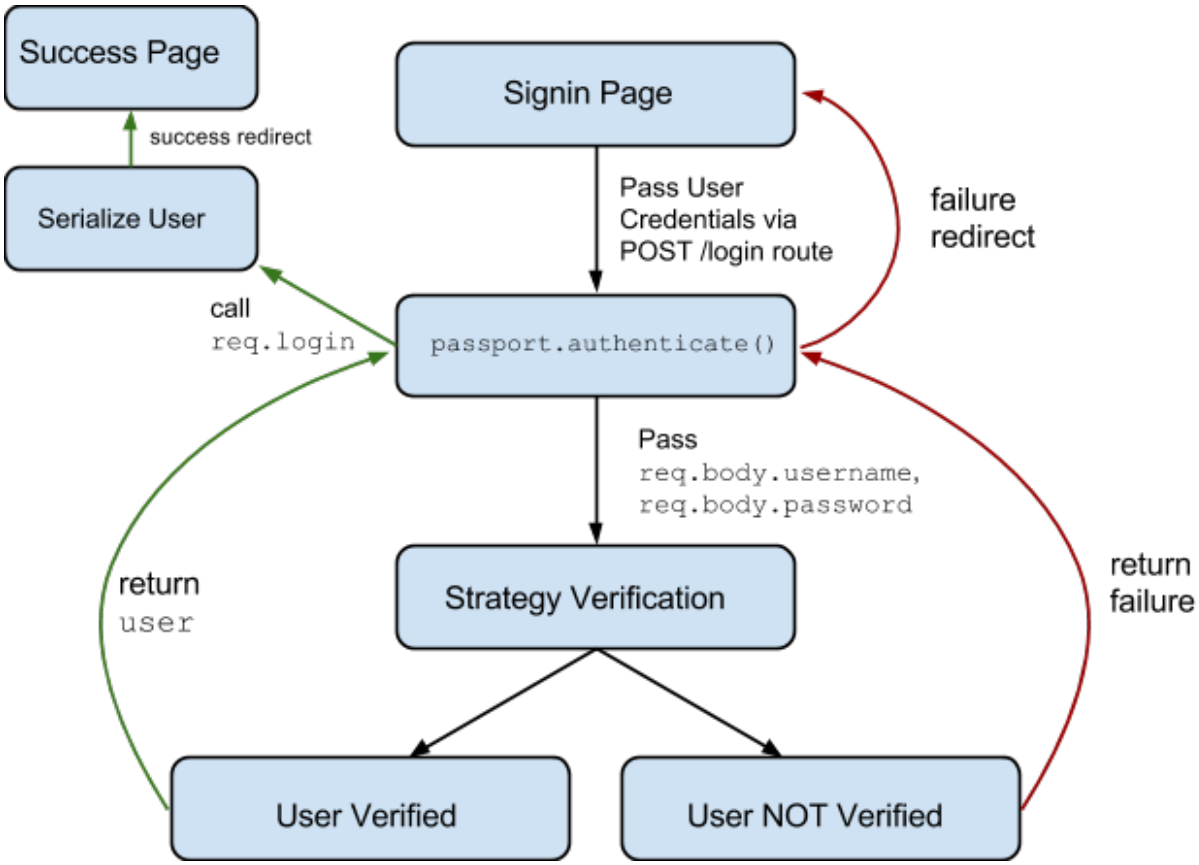
Šajā apakšnodaļā ir apspriesta tīmekļa lietojumprogrammas funkcionalitāte un izskats. Izmantojot GUI un funkcionālas sistēmas modeļus, bija uzsākta izstrāde.

2.5.1. Klienta daļas programmatūra

Tā kā bija nolēmts izveidot tīmekļa lietojumprogrammu, klienta daļas tehnoloģijas izvēle bija tikai starp *CSS* un *JS* ietvariem. Tīmekļa lietojumprogrammai ir jāstrādā tāpat labi gan uz mobiliem ierīcēm, gan uz personāla datoriem, tāpēc ir jāizmanto *CSS* ietvaru kas atbalsta reaģējošu tīmekļa lapas dizainu. Visvairāk pieredzes darba autoram ir ar *Bootstrap CSS* [24] un tā *JS* atkarības - *Bootstrap JS* [25] un *jQuery* [26].

2.5.2. Autentifikācija

Gudrā māja svešās rokās var izraist pārāk daudz problēmas, tieši tāpēc autentifikācija ir ļoti svarīga. Bija nolemts izmantot pārbaudītu autentifikācijas *middleware* (starpprogrammatūra) priekš *ExpressJS* - *Passport* [27]. Tā strādāšanas princips ir paradīts attēlā 2.6.

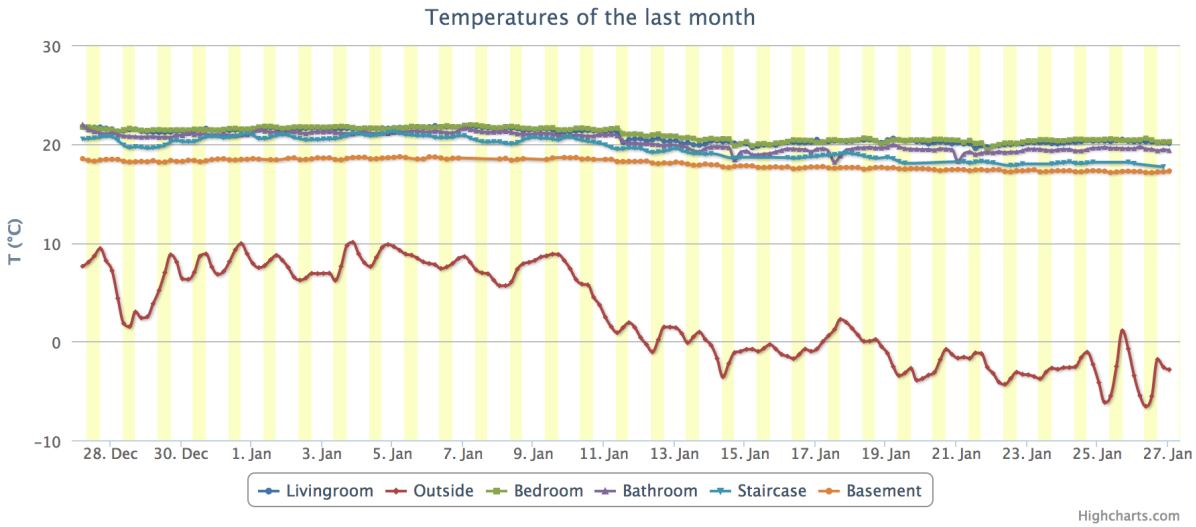


2.6. att. **Izmantots autentifikācijas modelis**

Pēc nepieciešamības, tas dod iespēju pieslēgt sistēma arī citas autentifikācijas stratēģijas, piemērām lai pieslēgtu *Facebook* sociālu tiklu autentifikāciju ir vajadzīgi tikai pievienot passport-facebook moduli (*npm install passport-facebook*) un ievadīt pareizus API atslēgas [28].

2.5.3. Grafiku zīmēšana

Sistēmā pamatā ir lielu datu apjomi sakārtoti pēc laiku un vairākums no tiem labāk uztverti ja tie ir grafikā veidā. Tāpēc bija nolemts izmantot ļoti plašu grafiku zīmēšanas *JavaScript* bibliotēku - *Highcharts* [29]. Piemēru, kā izskatās temperatūras grafiks zīmēts ar *Highcharts* var apskatīt 2.7.



2.7. att. **Temperatūras grafiks zīmēts ar Highcharts bibliotēku**

Bibliotēka arī ir iekļauta reāla laika atbalsts un tas ļauj automātiski pievienot jaunus datu punktus grafikā, kas ir ļoti noderīgi tādās sistēmās.

2.6. Testēšana un optimizācija

Sistēma ir paredzēta lai apkalpotu plašu lietotāju grupu daudzumu, kurām ir jābūt labi plānotam, lai arī nodrošināt nākotnes paplašināšanu. Šajā sadaļā ir apskatīti veiktspējas testēšanas rezultāti. Tiek identificēti vājās vietas un ir apskatīts kā tie ir laboti un optimizēti.

2.6.1. Vājās vietas identifikācija

Kā bija teikts nodaļa par projekta metodoloģiju, sistēmas izstrādes procesā bija vairākas iterācijas un katras iterācijas daļa bija nepārtrauktā testēšanā, lai agri atrastu kļūdas un vājās vietas.

Bija izmantota gan manuāla testēšana, analizējot tīmekļa trafiku, gan *Google Chrome* iebūvētais revīzijas instruments [30]. Ar to palīdzību bija atrastas 3 lielākas problēmas:

1. Pārāk daudz *JS*, *CSS* un *HTML* faili ir lejuplādēti
2. Pārāk lēna datubāzes pieprasījumu izpildīšana

2.6.2. Lietojumprogrammas optimizācija

Koda pārskatā no iepriekšējas apakšnodaļās bija atrastas divas lielākas problēmas.

2.6.2.1. *Javascript* un *CSS* failu saspiešana

SailsJS ietvarā ir iebūvēts kompilēšanas process, kuru vajag pareizi nokonfigurēt un ieslēgt. Galvenokārt, saspiešanas procesā noņemt nevajadzīgos bitus (nevajadzīgi semikoli utml.) Saspiests kods nav lasāms cilvēkam. Salīdzinot oriģinālos failus un saspiešanas rezultātus, bija iegūti jūtami rezultāti, jaunie faili ir apmēram 70% mazāki nekā bija.

Ja es nebūtu izmantojis *SailsJS*, es izmantotu brīvus tiešsaistes servisus: *jscompress.com* un *csscompressor.com*.

2.6.2.2. Datubāzes pieprasījumu optimizācija

Līdz optimizācijai bija izmantots *node-postgres* datubāzes adapteris. Bija nolemts izmēģināt gatavu *ORM* risinājumu, kas bija jau iebūvēts *SailsJS - Waterline*. Izdevās samazināt pieprasījumu skaitu par apmērām 50% un samazināt ielādēšanas laiku par 40%.

2.6.3. Sistēmas veiktspējas testēšana

Veiktspējas testēšanai bija izmantots testēšanas rīks *Apache AB* [31], ar kuru bija izveidots stress-tests priekš ”smagākas” lapas: vēstures dati. Tas atļāva atrast maksimālo vienlaicīgu sistēmas lietotāju skaitu ar komfortablu ielādēšanas laiku (<2000ms), kas ir apmērām 400 vienlaicīgu lietotāju. Ar tādu vienlaicīgu pieslēgumu skaitu ir nodrošināti apmēram 200 pieprasījumi sekundē. Cipars ir ļoti iespaidīgs un, protams, daudz lielāks nekā nepieciešams.

2.7. Drošības novērtēšana

Šajā nodaļā uzsvars tiek likts uz dažādām kopīgām tīmekļu lietojumprogrammas drošības draudiem. Šāds novērtējums ir balstīts uz *Open Web Application Security Project (OWASP)*.

Kā redzams tabulā 2.7, sanāca ka ir 4 lielu risku uzbrukumu vektori.

2.7. tabula. **Drošibas risku matrica**

TR 1.1 - Nedroša tiešā objektu norāde	Liels
TR 1.2 - Nedroša kriptogrāfiskā glabātuve	Zems
TR 1.3 - Pārlases uzbrukums	Zems
TR 1.4 - Nekorekta sesiju pārvaldība	Vidējs
TR 1.5 - Nepietiekama transporta slāņa aizsardzība	Liels
TR 1.6 - SQL injicēšana	Liels
TR 1.7 - Starpvietņu skriptēšana	Liels
TR 1.8 - Lietotāju uzskaitīšana	Zems
TR 1.9 - Ceļu šķērsošanu uzbrukums	Vidējs
TR 2.1 - Starplapu pieprasījumu viltošana	Vidējs
TR 2.2 - Pārtveršana	Zems
TR 3.1 - Pakalpojuma atteikuma uzbrukums	Vidējs

Nākamās apakšnodaļās viņi tiek apskatīti un ir arī apraksts kā sistēma no tiem ir aizstāvēta.

2.7.1. TR 1.1 - Nedroša tiešā objektu norāde

Tīmekļa lietotnes bieži izmanto faktisko objekta vārdu vai atslēgu, kad tiek ģenerētas tīmekļa lapas. Lietotne ne vienmēr pārbauda, vai lietotājs ir autorizēts uz šo objektu. Tā rezultātā rodas nedroša tiešā objekta norāde (*Insecure Direct Object Reference*). Testētāji, vienkārši mainot parametru vērtības, var noteikt, vai šāda ievainojamība eksistē, kā arī to var noteikt analizējot pašu programmas kodu. [32]

2.7.2. TR 1.6 - SQL injicēšana

SQL injicēšana ir datorsistēmu ielaušanās veids, kurā ļaundaris ar klienta tiesībām mēģina piekļūt datubāzei, kas ir dotās tīmekļa lapas vai lietotnes pamatā [33].

SQL injicēšanas uzbrukumi ir populārākais no *OWASP* saraksta drošības riskiem. Tas iemesls ir ka tas ir triviāls izmantojamākas risks un tajā pašā laikā potenciāli var sniegt masīvu ietekmi. *SQL* injekcijas uzbrukums izmanto vāji īstenotu ievades apstiprināšanu serveru pusē. Pateicoties *SailsJS* par to nav īsti jādomā, ja izmanto iebūvētus rīkus - *Waterline ORM* un pareizi konfigurētu veidnes dzinēju (*Handlebars*, šajā projektā) [34]. Tas viss dod ”bulletproof” aizsardzību pret *SQL* injekcijām, bet tomēr tas nav ļoti ērti galu lietotājam, tāpēc arī aktīvi jārāda

to viņiem ar klientu pūses validāciju. Tāpēc no klienta puses bija izmantota *Bootstrap.js validate* funkcija, kopā ar *Bootstrap CSS* klasiem kā *success* un *error*, lai dotu klientiem reaģējošu atsauksmi.

2.7.3. TR 1.7 - Starpvietņu skriptēšana

Ļaundaris izmanto *XSS*, lai iemānītu sevis izveidotu skriptu gala lietotājam. Tā kā lietotāja pārlūkprogramma uzskata, ka skripts ir saņemts no uzticama informācijas avota un tai nav nekādas iespējas uzzināt, to ka skriptam nedrīkst uzticēties, programma palaidīs skriptu, tādējādi nodrošinot tam piekļuvi jebkurām lietotāja sīkdatnēm (*cookies*), sesiju datiem un citai informācijai, kuru lietotāja pārlūkprogrammā izmanto tīmekļa lapā. Ar šādu skriptu palīdzību iespējams pārrakstīt *HTML* lapas saturu [35].

Uzbrucējs pievieno *JavaScript* kodu lietojumam vai nosuta upurim saiti ar kodu tajā, bet upura pārlūks vai mobilais lietojums izpilda nosūtīto kodu. Lai no tā pasargāties ir jāvalidē visi ievaddati, ņemot vērā, kā tie tiks apstrādāti un kur glabāti, starpvietņu skriptēšanā jāvalidē visi izvaddati, ņemot vērā, kur tie tiks attēloti.

2.7.4. TR 2.1 - Starplapu pieprasījumu viltošana

CSRF ir īpaši veiksmīgi pielietojams tīmekļa lietotnēs, kas ļauj uzbrucējam paredzēt visas detaļas konkrētām darbībām. Tā kā tīmekļa pārlūki suta autorizācijas datus tāpat kā sesijas sīkdatnes automātiski, uzbrucēji var izveidot ļaunprātīgas tīmekļa vietnes, kas ģenerē viltotus pieprasījumus, kas nav atšķirami no īstajiem. *CSRF* ir viegli noteikt testēšanas un programmas koda analīzes ceļā [32].

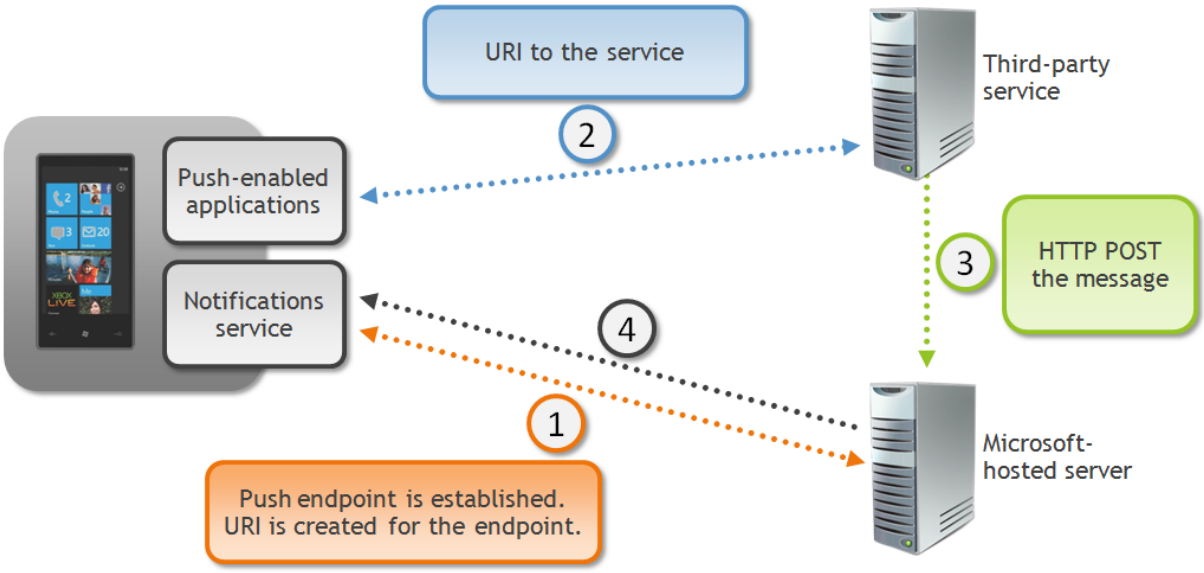
Izmantojot *SailsJS* par *CSRF* daudz nav jāuztraucas, tikai pievienot vēl vienu ievades lauku katrā formā ar vārdu *_csrf* kura vērtība ir vienāda ar *_csrf* mainīgo [36].

3. TURPMĀKAIS DARBS

Kaut gan sistēmas prototips ir izstrādāts un visi sākotnējie uzdevumi ir izpildīti, tādai sarežģītai sistēmai vienmēr ir kaut kas, ko uzlabot vai pievienot, it īpaši tāpēc, ka sistēma ir izveidota ar modulāru pieeju un ļauj viegli pievienot jaunas funkcijas.

3.1. Paziņojumu uzlabošana

Viens no vājajiem punktiem ir sistēmas paziņojumi. Pašlaik ir tikai viens veids kā tos saņemt, tas ir apmeklējot tīmekļa lapu. Tas nav optimāli un var būt izlabots pievienojot sistēmā *push* paziņojumus. Vispārīgi šāda sistēma strādā kā apskatīts attēlā 3.8.



3.8. att. **Push paziņojumi**

Principā, ir iespējams aizvietot *push* ar *SMS*.

3.2. Pielāgojams informācijas panelis

Prototipa galvenajā lapā tiek rādīta noteiktā profila svarīgākā informācija. Ir ļoti iespējams, ka dažādiem cilvēkiem būs dažādās domas par svarīgāko informāciju, tāpēc būtu labi, ja būtu iespēja pielāgot informācijas paneli.

3.3. Sistēmas mērogošana

Balstoties uz noslodzes testiem, kuri bija veikti nodaļā 2.6.3, sistēmas prototips reaģē ļoti labi un atsaucīgi pat ar 400 lietotajiem vienlaicīgi, vienīgi tas viss ir novērots ”laboratorijas ap-

stākļos”. Reālos apstākļos, visticamāk, būs negaidītas problēmas (kā vienmēr), tāpēc jau uzreiz ir jādomā kas būs jādara, ja sistēmas optimizācijas nepalīdz vai ne attaisno savu izstrādāšanas ”cenu” - laiku.

Node.js un *Sails.js* mērogo horizontāli - ja sistēmu apkalpo divi serveri, tad sistēmas maksimāla iekraušana ir apmēram divreiz lielāk nekā vienām serverim [37]. Tas viss nozīme, ka pietiek tikai uzstādīt slodzes stabilizatoru (*load balancer*), uzstādīt *Redis* priekš *session* un *websockets* pareizo sadalīšanu, un uzstādīt vairāk lietojumprogrammas serverus. Savukārt, datubāze arī var būt iedalīta uz klasteriem [38].

4. REZULTĀTI

Šī darba ietvaros, autors ir izpētījis viedās mājas tehnoloģijas un tās funkcionālas prasības. Darba rezultātā tika izveidota gudras mājas sistēma, kura tagad ir veiksmīgi uzstādītā darba autora mājās. Šobrīd, prototipa kopā ir:

1. divas multi-relejas, kas kontrolē apgaismojamību divās istabās,
2. divi temperatūras sensori, kas reģistrē mājas iekšējo temperatūru un arī temperatūru aiz logiem,
3. bāzes stacija, kas nodrošina datu saglabāšanu un piekļuvi sistēmai tiešsaitē,
4. tīmekļa lietojumprogramma, kas atļauj kontrolēt multi-relejas, apskatīt temperatūras datus un konfigurēt sistēmas elementus.

SECINĀJUMI

Šī darba ietvaros, autors ir izpētījis viedās mājas tehnoloģijas un tās funkcionālas prasības. Darba rezultātā ir izstrādāts strādājošs, ar koncepciju pierādīts prototips, kurš izgājis gan veiktspējas testus, gan funkcionalitātes testēšanu un tagad tiek veiksmīgi izmantots darba autora mājās. Darbā bija izmantots plašs tehnoloģiju spektrs: *C++* un *Javascript* programmēšanas valodas, *PostgreSQL* datubāze un daudz palīdzbibliotēkas.

Autors konkrēto sistēmu novērtē kā drošu un domā ka pēc nelieliem papildinājumiem, pilot-testēšanu un pareizo uzrādīšanu, tādu sistēmu noteikti var izmantot cilvēki bez tehniskām prasmēm, kā arī izmantot to sistēmu komerciāli.

Kopumā var teikt, ka darba mērķi ir sasniegti – ir veiksmīgi izpētīti eksistējošie risinājumi, apzināti to trūkumi un priekšrocības, projektēts un izstrādāts savs gudras mājas prototips, kā arī izdarīta analīze tālākajiem darbiem.

IZMANTOTĀ LITERATŪRA UN AVOTI

[1] Inc. Gartner. *Press Release - 4.9 Billion Connected "Things" Will Be in Use in 2015*. www.gartner.com/newsroom/id/2905717/, 2014.

[2] Inc. Gartner. *Press Release - Gartner's 2014 Hype Cycle for Emerging Technologies*. www.gartner.com/newsroom/id/2819918/, 2014.

[3] Jr. Edward B.Driscoll. *The History of X10*. home.planet.nl/˜lhendrix/x10_history.htm, 2007.

[4] Dave Rye. *Dave Rye of X10*. www.hometoys.com/content.php?post_type=394, 2004.

[5] Z-Wave Alliance. *Our history*. www.z-wavealliance.org/z-wave_alliance_history/, 2008.

[6] Ata Elahi & Adam Gschwender. *Introduction to the ZigBee Wireless Sensor and Control Network*. www.informit.com/articles/article.aspx?p=1409785, 2009.

[7] SafeSoundFamily.com. *The Best Home Automation Software & Systems*. www.safesoundfamily.com/blog/best-home-automation-software-products/, 2015.

[8] HomeSeer. *What Can You Do With HomeSeer?* store.homeseer.com/store/HomeSeer-Application-Guide-W29.aspx, 2015.

[9] Ystein Haugen Rolv Braek, Oystein Haugen. *Engineering Real Time Systems: An Object-Oriented Methodology Using SDL*. Prentice Hall, 1 edition, 1993.

[10] Raspberry Pi Foundation. *Raspberry Pi 1 Model B+*. www.raspberrypi.org/products/model-b-plus/, 2014.

[11] BeagleBoard.org. *What is BeagleBone Black?* www.beagleboard.org/black, 2014.

[12] Arduino. *Arduino UNO Rev3*. store.arduino.cc/product/A000066/, 2013.

[13] Nordic Semiconductor. *nRF24L01 - Overview*. www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01/, 2013.

[14] Redis. *Redis*. www.redis.io, 2015.

[15] Inc. MongoDB. *MongoDB*. www.mongodb.org, 2015.

[16] The PostgreSQL Global Development Group. *PostgreSQL*. www.postgresql.org, 2015.

[17] LLC Toptal. *Why The Hell Would I Use Node.js?* www.toptal.com/nodejs/why-the-hell-would-i-use-node-js/, 2015.

[18] Azat Mardan. *Express.js Guide*. www.expressjsguide.com/, 2013.

[19] SailsJS Team. *Sails.js - Get Started*. http://sailsjs.org/#!/getStarted, 2013.

[20] Nick Chaves. *What is MVC, really?* programmers.stackexchange.com/a/127632/, 2013.

[21] SailsJS Team. *Sails.js Command Line Interface*. www.sailsjs.org/#!/documentation/reference/cli/sailsgenerate.html, 2013.

[22] Stanley Seow. *Arduino & Raspberry Pi driver for nRF24L01*. www.github.com/stanleyseow/RF24/, 2014.

[23] TMRh20. *Optimized Network Layer for nRF24L01*. www.github.com/TMRh20/RF24Network/, 2014.

[24] Twitter Inc. *Bootstrap CSS*. www.getbootstrap.com/css/, 2015.

[25] Twitter Inc. *Bootstrap JavaScript*. www.getbootstrap.com/javascript/, 2015.

[26] The jQuery Foundation. *jQuery*. www.jquery.com, 2015.

[27] Jared Hanson. *Passport - Simple, unobtrusive authentication for Node.js*. www.passportjs.org, 2015.

[28] Jared Hanson. *Passport - Facebook authentication strategy*. www.passportjs.org/docs/facebook/, 2015.

[29] Highcharts LLC. *Highcharts - Interactive JavaScript charts*. www.highcharts.com, 2015.

[30] Google Inc. *Chrome DevTools Overview*. developer.chrome.com/devtools/, 2015.

[31] The Apache Software Foundation. *ab - Apache HTTP server benchmarking tool*. httpd.apache.org/docs/2.2/programs/ab.html, 2015.

[32] Mārtiņš Briedis. Tīmekļa vietnes drošības novērtēšana un uzlabošana. Master's thesis, Latvijas Universitāte, 2011.

[33] pentest.lv. *SQL injicēšana*. www.pentest.lv/lv/attacks/sqli/, 2014.

[34] Cody Stoltman. *Waterline ORM: Query Language*. www.github.com/balderdashy/waterline-docs/blob/master/query-language.md, 2015.

[35] pentest.lv. *Starpvietņu skriptēšana*. www.pentest.lv/lv/attacks/xss/, 2014.

[36] SailsJS Team. *SailsJS: CSRF*. sailsjs-documentation.readthedocs.org/en/latest/concepts/CSRF/, 2015.

[37] SailsJS Team. *Sails.js - Scaling*. www.sailsjs.org/#!/documentation/concepts/Deployment/Scaling.html 2015.

[38] PostgreSQL Documentation. *Replication, Clustering, and Connection Pooling*. wiki.postgresql.org/wiki/Replication,_Clustering,_and_Connection_Pooling/, 2015.

PIELIKUMS A. LIETOJUMPROGRAMMAS SASKARNES GALAPUNKTI

Sistēmā ir definēti dažādi galapunkti, gan priekš klientiem, gan sistēmas ierīcēm.

GET /devices

Atgriež HTML, kur var apskatīt visas ierīces sistēmā.

GET /devices/id

Atgriež HTML, kur var apskatīt noteiktu ierīci.

POST /devices/found

Jaunas, sistēmai nepievienotas ierīces šeit suta informāciju par sevi. Atgriež statusa kodu, ja bija pievienota jauna ierīce, tad veiksmīgu, ja jauna ierīce nebija pievienota, tad neveiksmīgu.

POST /devices/pairing

Atgriež statusa kodu, ja slepenais ID, kas bija ievadīts ir pareizs, tad veiksmīgu, pretējā gadījumā neveiksmīgu.

GET /devices/setup/id

Atgriež HTML, kur var iestādīt jauno ierīci sistēma. Tai vispirms jābūt pievienotai ar /devices/found.

POST /devices/setup/id

Novirza atpakaļ uz tādu pašu GET.

GET /devices/edit/id

Atgriež HTML, kur var rediģēt ierīces konfigurāciju.

GET /devices/delete/id

Izdzēš ierīci no sistēmas. Novirza uz GET /devices.

WS /subscribe/devices

Abonē websocket notikumus, kas ir saistīti ar ierīcēm.

WS /subscribe/incoming

Abonē websocket notikumus, kas ir saistīti ar jaunām ienākošām atskaitēm no ierīcēm.

POST /report

Galapunkts priekš radio dēmona. Šeit tas atsuta visu saņemto informāciju no mezgliem, piem. Sensora vērtības. Atgriež statusa kodu, ja vērtība bija pievienota, tad veiksmīgu, pretējā gadījumā neveiksmīgu.

GET /history

Atgriež HTML, kur var apskatīt visus pēdējos notikumus sistēmā.

GET /history/id

Atgriež HTML, kur var apskatīt kādu noteiktu notikumu sistēmā.

GET /triggers

Atgriež HTML, kur var apskatīt visus pievienotus trigerus sistēmā.

GET /triggers/id

Atgriež HTML, kur var apskatīt kādu noteiktu trigeru sistēmā.

POST /triggers/id

Novirza atpakaļ uz tādu pašu GET.

POST /triggers/add

Novirza uz jaunizveidotu trigeru - GET /triggers/id.

POST /triggers/delete/id

Izdzēš trigeru no sistēmas. Novirza uz GET /triggers.

GET /dashboard

Atgriež HTML, kur var apskatīties tiekošo kopējo sistēmas statusu un arī ātri izmantot populārākus aktuātorus, piem. ieslēgt gaismu.

POST /dashboard/use

Šeit ir atsūtīta informācija par kuru aktuatoru iet runa un ko tieši tas var izdarīt. Atgriež operācijas statusu kodu.

PIELIKUMS B. KODA FRAGMENTI

Attēlā 4.9. ir redzāms koda fragments kas ir atbildīgs par jauno ziņojumu apstrādi uz bāzes stacijas.

```
12    incoming: function(req, res) {
13      if (req.body) {
14        IncomingMessage.create({
15          fromNode: req.body.fromNode,
16          messageReason: req.body.messageReason,
17          messageBody: req.body.messageBody
18        }).done(function(err, message) {
19          if (err) {
20            console.error('IncomingMessage error', err);
21            return res.send(302);
22          } else {
23
24            IncomingMessage.publishCreate({
25              id: message.id,
26              nodeId: req.body.fromNode,
27              messageReason: message.messageReason,
28              messageBody: message.messageBody,
29              createdAt: message.createdAt
30
31            });
32            return res.send(200);
33          }
34        });
35      } else {
36        console.error('Error. Code 77');
37        return res.send(302);
38      }
39
40    }
```

4.9. att. **Koda fragments 1 - Ziņojumu apstrāde**

Attēlā 4.10. ir redzāms koda fragments kas ir atbildīgs par klientu abonēšānu uz izmaiņu mezglās.

```
14    subscribeDiscoveredDevices: function(req, res) {
15      if (req.isSocket) {
16        async.parallel({
17          nodes: function(cb) {
18            Node.find().sort('added DESC')
19              .exec(function(err, nodes) {
20                cb(err, nodes);
21              });
22          },
23          discoveredNodes: function(cb) {
24            DiscoveredNode.find().sort('added DESC')
25              .exec(function(err, discNodes) {
26                cb(err, discNodes);
27              });
28          }
29        }, function(err, results) {
30          if (err) {
31            console.log(err);
32            return res.send(500);
33          } else {
34            Node.watch(req.socket);
35            DiscoveredNode.watch(req.socket);
36
37            return res.send(200);
38          }
39        });
40      } else {
41        return res.send(403);
42      }
43    },
```

4.10. att. **Koda fragments 2 - Websocket mezglu izmaiņu abonēšana**

Attēlā 4.11. ir redzāms koda fragments kas ir atbildīgs par lapu atveidošanu ar visiem aktiviem ierīcēm sistēmā un visiem atrastiem ierīcēm apkārt.

```
13    index: function (req, res) {
14
15      async.parallel({
16        nodes: function(cb) {
17          Node.find().sort('added DESC')
18            .exec(function(err, nodes) {
19              Node.subscribe(req.socket, nodes);
20              cb(err, nodes);
21            });
22        },
23        discoveredNodes: function(cb) {
24          DiscoveredNode.find().sort('added ASC')
25            .exec(function(err, discNodes) {
26              cb(err, discNodes);
27            });
28        }
29      }, function(err, results) {
30        var sessionError = req.session.error;
31        req.session.error = null;
32        if (err) {
33          console.log(err);
34        }
35        return res.view('devices/index', {
36          nodes: results.nodes,
37          discoveredNodes: results.discoveredNodes,
38          errorMessage: sessionError
39        });
40      });
41
42    },
```

4.11. att. **Koda fragments 3 - Lapu atveidošana ar visiem aktiviem un visiem apkārt atrastiem ierīcēm**

PIELIKUMS C. APTAUJAS JAUTĀJUMI UN REZULTĀTI

Kā darba risināmu problēmu es uzskatu tieši profilu vadītu gudras mājas sistēmas piedāvāšanu, tāpēc sākotnējais uzdevums bija definēt kādu gudru mājas sistēmu ir nepieciešams izveidot. Lai to izdarītu bija aptaujāti gala lietotāji - ģimene kura dzīvo dzīvoklī kur ir plānots ievest sistēmu.

Jautājums un atbilžu varianti:

Kādas iespējas jūs interesē gudrās mājās sistēmā?

(1) Dzīvokļa temperatūra

(2) Ārējā temperatūra

(3) Gaisa mitrums

(4) Kustības detektēšana

(5) Tīmekļa kameras apskate

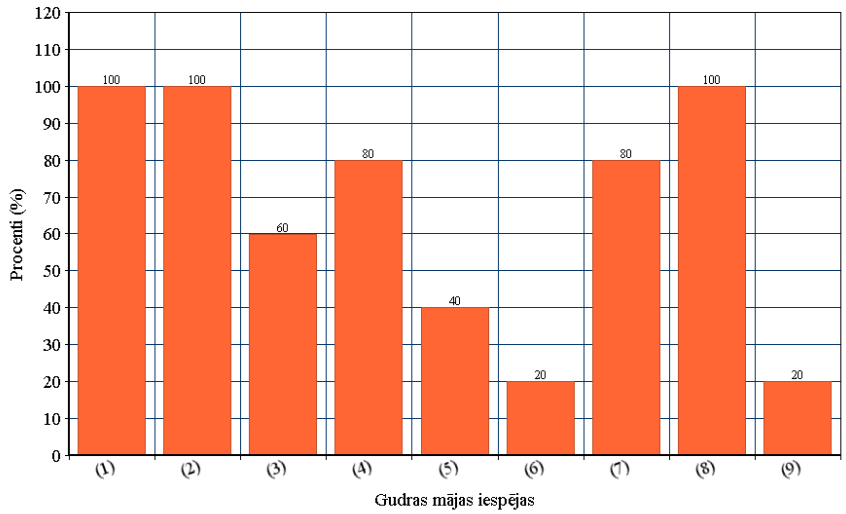
(6) Datu vēsture

(7) Izslēgt un ieslēgt elektroierīces attālināti

(8) Paziņojumu paziņošana

(9) Iespēja to viss apskatīt un pārvaldīt no viedtālruni

Rezultātā, kā ir redzams attēla 4.12, sanāca ka gandrīz visas atbildes ir pietiekami svarīgi un vajadzīgi.



4.12. att. **Aptaujas rezultāti**

Bakalaura darbs «Viedās mājās modulāra sistēma un prototipa implementēšana» izstrādāts Latvijas Universitātes Datorikas fakultātē.

Ar savu parakstu apliecinu, ka darbs izstrādāts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors:

Ilja Gubins _____ __.06.2015.

Rekomendēju/nerekomendēju darbu aizstāvēšanai (nevajadzīgo izsvītrot).

Darba vadītājs: LU asoc. prof., Dr. dat.

Uldis Straujums _____ __.06.2015.

Recenzents: LU docents, Dr.sc.comp.

Lelde Lāce _____ __.06.2015.

Darbs iesniegts Datorikas fakultātē __.06.2015.

Dekāna pilnvarotā persona:

_____ __.06.2015.

Darbs aizstāvēts bakalaura darbu komisijas sēdē
__.06.2015. prot. Nr. ____

Komisija: _____