Latvijas Universitāte

Datorikas fakultāte

“Mājas automatizācijas sistēma”

Kvalifikācijas darbs

2.kurss

Darba vadītājs: B. Dat. Mareks Kalnačs

Autors: Kristaps Kietis

Anotācija

Kvalifikācijas darbs “Mājas automatizācijas sistēma” dod iespēju lietotājam veikt manipulācijas ar sistēmā reģistrētām ierīcēm. Ierīču lietotājam nav jābūt arī ierīces īpašniekam, kas sistēmu padara elastīgu ģimenēm un draugiem.

Kvalifikācijas darba mērķis ir izveidot sistēmu, kas ir viegli lietojama un ierīkojama jebkurā mājoklī. Lai pilnvērtīgi lietotu aplikāciju ir jāveic divas darbības: 1. Jāizveido lietotāja piekļuves konts. 2. Jāreģistrē jaunu ierīci vai jāpieprasa lietošanas tiesības kāda lietotāja ierīcei.

Sistēmā pilnvērtīgi var lietot divu veidu ierīces – slēdzi, kurš strādā balstoties uz binära principa, un signalizācija, kura veic paziņojumus lietotājam, par drošības pārkāpumu.

Sistēmai ir izaugsmes potenciāls nākotnē, ko var realizēt vai nu veidojot specializētas sistēmas ierīces vai ļaujot lietotājam Arduino kodu lietot kā “Open Source” projektu, kas dotu iespēju veidot specializētas lietotāju ierīces.

Atslēgvārdi: Smart House, Gudrā māja, Android, PHP, Java, Arduino, MySql

Abstract

Project " Home Automation System" enables the user to manipulate with system registered devices. Device user does not have to be the owner of the device, which makes the system flexible for families and friends.

The aim of project is to create a system that is easy to use and to be installed in any home. In order to fully use the application requires two steps: 1. create a user account. 2. register a new device or request the access rights to use other user devices.

Fully system works with two types of devices - a switch which works based on a binary basis, and alarm, which notify the user to for security breaches.

The system has the potential for growth in the future, which may be accomplished either through specialized system devices, or allowing the user to use the Arduino code as "Open Source" project, which would enable users to build specialized devices.

Keywords: Smart House, Gudrā māja, Android, PHP, Java, Arduino, MySql

Definīcijas un akronīmi

IT – Informāciju tehnoloģijas

MVC – Modeļu, skatu, kontrolieru programmēšanas veids (*Model View Controller*)

DPD – Datu plūsmu diagrammas

Open Source – atklātā pirmkoda programmatūra

Saturs

1 Programmatūras prasību specifikācija

1.1 Ievads

1.1.1 Nolūks

Programmatūras prasību specifikācija ir izstrādāta, lai precīzi aprakstītu programmatūras "Mājas automatizācijas sistēma" prasības un to realizāciju. Tā sevī ietvers funkcionālās un nefunkcionālās prasības, kuras būs kā ceļvedis kvalitatīvas programmatūras izstrādei.

1.1.2 Darbības sfēra

Programmatūra ir "Gudrās mājas" implementācijas sistēma, kuras mērķis ir veidot kvalitatīvas lietotāja saziņas iespējas lietojot tīmekļa protokolu ar attiecīgi konfigurētām ierīcēm

1.1.3 Saistība ar citiem dokumentiem

Veidojot šo projektu tiek ievērots LVS 68:1996 „Programmatūras prasību specifikācijas ceļvedis" standarts

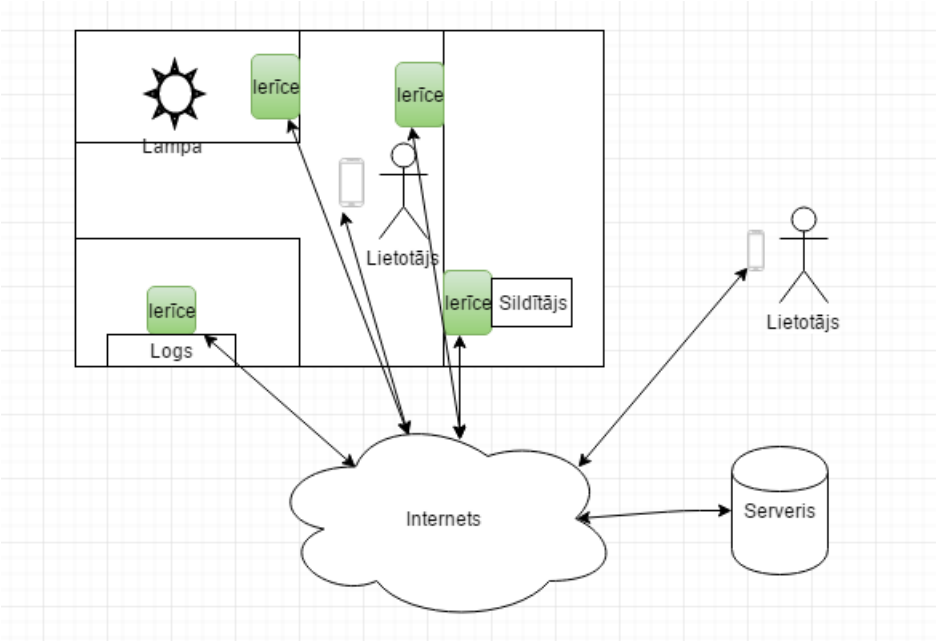
1.2 Vispārējais apraksts

1.2.1 Produkta perspektīvas

Šis produkts ir izstrādāts izmantojot labās prakses programmēšanas raksturiezīmes, kas nozīmē, ka to ir iespēja attīstīt un uzturēt ilglaicīgā laika periodā. Produkta mērķis balstās uz pašsaprotamību no lietotāja puses, kā rezultātā tam ir liels potenciālo klientu skaits, jo tiem nav nepieciešamas tehniskas zināšanas.

1.2.2 Produkta mērķis

Sistēmas galvenais uzdevums ir nodrošināt stabilu mājas kontroli no jebkuras vietas pasaulē. Tā sevī ietver no standarta gaismas ieslēgšanas un izslēgšanas līdz drošības līdzekļiem, kā, piemēram, paziņojumiem par ierobežotas teritorijas sliekšņa pārkāpšanu, piemēram, atvērtu logu brīdī, kad nevienam nebūtu jābūt iespējamai šādai darbībai.



Attēlā ir vizuāli apskatāma sistēmas konceptuālā darbība. Šajā attēlā ir apskatāma vienas mājas funkcionālā darbība, kuru realizē divi neatkarīgi lietotāji. Abiem lietotājiem ir pilnīgi vienādas iespējas kontrolēt māju, ja tam ir dotas tiesības to darīt, par ko atbild centrālais serveris. Atrodoties mājā galvenā lietotāja funkcionalitāte būtu ierīču ieslēgšana un izslēgšana, taču atrodoties ārpus mājas svarīga funkcionalitāte būtu paziņojumu attēlošana mobilā ierīcē brīdī, kad nostrādātu signalizācijas režīms, kā arī ierīču aktivizēšanu, piemēram, sildītāja ieslēgšanos neilgu laiku, pirms klients ierodas mājās.

1.2.3 Risinājuma izvēles pamatojums

Tā kā ierīču programmatūra tiek izstrādāta izmantojot Arduino mikrokontrolieri, ir nepieciešams izstrādāt minimālistisku programmatūru pamata funkcionalitātes realizācijai. Tā rezultātā beidzot ierīces konfigurāciju, tā darbosies nesazinoties ar klientu, bet visas darbības notiks tikai ar servera starpniecību.

1.2.4 Produkta funkcijas

Sistēmas realizācijai ir nepieciešami trīs moduļi:

- Lietotāju modulis – Dod iespēju veidot personalizētu un kontrolējamu lietotāju darbību. Tas realizē iespēju vairākiem lietotājiem, piemēram, ģimenes locekļiem brīvi kontrolēt vienu vai vairākas kopīgas ierīces, ja konkrētam lietotājam ir dotas tiesības tās lietot.
- Ierīces modulis – Nodrošina kvalitatīvu ierīču komandu saņemšanu, sūtīšanu un izpildi. Dod iespēju reģistrēt neatkarīgu ierīci un brīvi veikt manipulācijas šīs ierīces ietvaros.
- Vēstures modulis – Nodrošina iespēju sekot līdzi ierīces un lietotāja statusa maiņai. Tas dod iespēju ierīces īpašniekam noteikt neadekvātu darbību ar ierīci, kā rezultātā noteikt lietotāja lietotājvārdu, lai atņemtu ierīces tiesības.

1.2.5 Lietotāja raksturiezīmes

Potenciālais lietotās ir cilvēks, kuram ir interese tiekties uz nākotni, un pētīt jaunas iespējas, ko ir iespējams sastapt IT pasaulē. Lietotājam ir jābūt velmei atvieglot savu ikdienas dzīvi un palielināt drošības sajūtu, kamēr tas neatrodas savā dzīves vietā.

1.2.6 Vispārējie ierobežojumi

Sistēmas lietošanai interneta pieslēgumam ir jāatbilst interaktīvas lietotāja saskarnes nodrošināšanai, kā rezultātā reakcijas laiks nedrīkst būt ilgāks par 3 sekundēm. Lietotājam ir nepieciešams Android telefons ar operētājsistēmu Android 4.2 vai augstāku.

1.3 Funkcionālās prasības

1.3.1 Lietotāju modulis

1.3.1.1 Reģistrācija

IDENTIFICĒJUMS	FP.LM.01
MĒRĶIS	Nodrošināt personalizētu sistēmas vadības sistēmu
IEVADDATI	"Firstname" – Teksta lauks robežās no 3 līdz 30 simboliem "Lastname" - Teksta lauks robežās no 3 līdz 30 simboliem "Username" - Teksta lauks robežās no 3 līdz 20 simboliem "Password" - Teksta lauks robežās no 6 līdz 255 simboliem, maskēts ar apļa veida simboliem "Register" – Poga autorizācijas inicializēšanai "Back" – Poga autorizācijas atcelšanai
APSTRĀDE	Visi teksta lauki tiek pārbaudīti uz to norādīto garumu un tiek pārveidoti uz atļauto kodējumu novēršot datu drošības pārkāpuma iespējas. Lauks "Username" tiek pārbaudīts uz unikalitāti. Sekmīgas reģistrācijas rezultātā lietotājs tiek dota piekļuve pie programmatūras.
IZVADDATI	"Registred" – Sekmīgas reģistrācijas informatīvs paziņojums
ATKARĪBAS	FP.LM.02, FP.VM.02
KĻŪDU PAZIŅOJUMI	"Too long" – ievadīti pārāk daudz simboli "Too short" – ievadīti pārāk maz simboli "Wrong" – nesakritība ievades struktūrā "Something went wrong" – Serveris nespēj veikt darbību

1.3.1.2 Piekļuves pieprasīšana

Identificējums	FP.LM.02
Mērķis	Nodrošināt piekļuvi pie personalizētas vadības sistēmas, un dot iespēju veikt manipulācijas ar programmatūras funkcijām
Ievaddati	"Username" - Teksta lauks robežās no 3 līdz 20 simboliem "Password" - Teksta lauks robežās no 6 līdz 255 simboliem, maskēts ar paroles aizstājējsimboliem.
Apstrāde	Visi teksta lauki tiek pārbaudīti uz to norādīto garumu un tiek pārveidoti uz atļauto kodējumu novēršot datu drošības pārkāpuma iespējas. Sekmīgas autentifikācijas rezultātā lietotājs tiek novirzīts uz standarta lietotnes lapu.
Izvaddati	"Logged in" – sekmīgas piekļuves informatīvs paziņojums
Atkarības	FP.VM.02
Kļūdu paziņojumi	"Too long" – ievadīti pārāk daudz simboli "Too short" – ievadīti pārāk maz simboli "Something went wrong" – Serveris nespēj veikt darbību

1.3.1.3 Lietotāja datu rediģēšana

Identificējums	FP.LM.03
Mērķis	Veikt izmaiņas lietotāja personalizācijas datos.
Ievaddati	"Firstname" – Teksta lauks robežās no 3 līdz 30 simboliem "Lastname" - Teksta lauks robežās no 3 līdz 30 simboliem "Username" - Teksta lauks robežās no 3 līdz 20 simboliem "Password" - Teksta lauks robežās no 6 līdz 255 simboliem, maskēts ar apļa veida simboliem
Apstrāde	Pārbauda vai ir aizpildīta vismaz viena no vērtībām, pretējā gadījumā izmaiņas netiek veiktas. Teksta lauki tiek pārbaudīti uz to norādīto garumu un tiek pārveidoti uz atļauto kodējumu novēršot datu drošības pārkāpuma iespējas.
Izvaddati	"Success!" – Veiksmīgas lietotāja datu maiņas paziņojums

Atkarības	FP.VM.02
Kļūdu paziņojumi	"Too long" – ievadīti pārāk daudz simboli "Too short" – ievadīti pārāk maz simboli "Something went wrong" – serveris nespēj veikt darbību

1.3.1.4 Ierīces aktivizēšana

Identificējums	FP.LM.04
Mērķis	Dod iespēju lietotājam ieslēgt un izslēgt ierīci
Ievaddati	deviceID - maskēts klasifikators sha32 kodējumā, 8 simbolu garumā Status – Klasifikatorvērtība, kura reprezentē "On" vai "Off" "Username" - Teksta lauks robežās no 3 līdz 20 simboliem
Apstrāde	Dati tiek nodoti serverim, kurš veic aprēķinus, vēlāk piegādājot informāciju ierīcei.
Izvaddati	"Success" – pieprasījums ir veiksmīgi nosūtīts
Atkarības	FP.VM.02
Kļūdu paziņojumi	"Something went wrong" – Serveris nespēj veikt darbību

1.3.1.5 Ierīces lietotāju piekļuves pieprasīšana

Identificējums	FP.LM.05
Mērķis	Pesaistīt personu ierīces koplietošanai
Ievaddati	"Username" – pieprasītāja lietotājvārds "deviceID" – ierīces unikāls identifikators
Apstrāde	Tiek veikta sasaiste lietotājam ar klientu
Izvaddati	"Success" – pieprasījums ir veiksmīgi nosūtīts
Atkarības	FP.VM.02, FP.LM.08
Kļūdu paziņojumi	"Something went wrong" – Serveris nespēj veikt darbību

1.3.1.6 Jaunas ierīces reģistrēšanas inicializācija

Identificējums	FP.LM.06
Mērķis	Dot lietotājam iespēju papildināt savu lietojamo ierīču skaitu
Ievaddati	"deviceName" - Teksta lauks robežās no 3 līdz 30 simboliem "SSID" – Teksta lauks robežās no 1 līdz 32 simboliem, "wiFiPassword" - Teksta lauks robežās no 6 līdz 255 simboliem, maskēts ar apļa veida simboliem "deviceType" – Ierīces tipu raksturojošs klasifikators "userID" – lietotāju identificējams identifikators
Apstrāde	Visi teksta lauki tiek pārbaudīti uz to norādīto garumu. Tiek atvērts bluetooth savienojums ar ierīci, un tai tiek nosūtīti formas lauki, kurus ierīce apstrādās un nosūtīs informāciju serverim.
Izvaddati	"Device registred" – Ierīcei sekmīgi izdevies piereģistrēties
Atkarības	FP.IM.01, FP.VM.02
Kļūdu paziņojumi	"Connection Failed!" – Nav sagaidīta atbilde no ierīces "Wrong network parameters!" – nekorekti tīkla dati "Device in wrong state" – Ierīce neatrodas reģistrēšanas stāvoklī "Bluetooth Device Not Available" – Aplikācijai nav izdevies izveidot jebkāda veida komunikāciju ar bluetooth adapteri

1.3.1.7 Piederošo ierīču iegūšana

Identificējums	FP.LM.07
Mērķis	Iegūt lietotājam piesaistīto ierīču datus pie datu inicializācijas pieprasījuma.
Ievaddati	"Username" – Autentificēta lietotāja lietotājvārds, kur tiek iegūts no tekošās sesijas.
Apstrāde	Tiek pārbaudītas lietotāja piekļuves tiesības.
Izvaddati	Masīvs ar lietotājam piederošo ierīču datiem "deviceName" - Teksta lauks robežās no 3 līdz 30 simboliem

	“Status” – On/Off
Atkarības	FP.LM.02
Kļūdu paziņojumi	“Something went wrong” – Serveris nespēj veikt darbību

1.3.1.8 Lietotāju meklēšana

Identificējums	FP.LM.08
Mērķis	Atrast lietotāju pēc tā vārda
Ievaddati	“searchName” – Meklēšanas lauks garumā no 3 līdz 30 simboliem, kurš attiecas uz “User” tabulas lauku “FullName”
Apstrāde	Tiek atlasīti visi lietotāji, kuru vārdā iekļaujas meklētais vārds, un atrastā lietotāja ierīces, kuras lietotājam ir iespēja pieprasīt lietošanai.
Izvaddati	Masīvs ar lietotājiem un to ierīcēm.
Atkarības	Nav
Kļūdu paziņojumi	“Something went wrong” – Serveris nespēj veikt darbību

1.3.1.9 Notifikācijas

Identificējums	FP.LM.09
Mērķis	Nekavējoties piegādāt informāciju par jaunāko informāciju
Ievaddati	“userID” – lietotāju identificējams identifikators
Apstrāde	Sistēmas līmeņa process noskaidro vai lietotājam ir kāds aktuāls jaunums, apstrādā un attēlo to. Pēc attēlošanas konkrētai notifikācijai tiek uzstādīts statuss “lasīts”, lai tā tiktu piegādāta vienreiz.
Izvaddati	Masīvs ar notifikācijām.
Atkarības	Nav
Kļūdu paziņojumi	Nav

1.3.2 Ierīces modulis

1.3.2.1 Ierīces reģistrēšana

IDENTIFICĒJUMS	FP.IM.01
MĒRĶIS	Izveidot unikālu ierīces savienojumu ar programmatūras serveri
IEVADDATI	"deviceName" - Teksta lauks robežās no 3 līdz 30 simboliem "SSID" – Teksta lauks robežās no 1 līdz 32 simboliem, "wiFiPassword" - Teksta lauks robežās no 6 līdz 255 simboliem, maskēts ar apļa veida simboliem "deviceType" – Ierīces tipu raksturojošs klasifikators "userID" – lietotāju identificējams identifikators
APSTRĀDE	Tiek veidots savienojums ar tīklu izmantojot iegūtos bezvada tīkla piekļuves datus. Veiksmīga savienojuma rezultātā ierīce veic reģistrāciju kopējai applikācijas lietošanai
IZVADDATI	"Success" – Tiek nosūtīta informācija android ierīcei
ATKARĪBAS	FP.LM.06
KĻŪDU PAZIŅOJUMI	"ERROR" – tehnisks lauks, kura rezultātā klients varēs noteikt reģistrācijas nesekmību

1.3.2.2 Ierīces statusa nosūtīšana

IDENTIFICĒJUMS	FP.IM.02
MĒRĶIS	Nodrošināt signalizācijas funkcionalitāti ierīcēs
IEVADDATI	"deviceID" - maskēts klasifikators sha32 kodējumā, 8 simbolu garumā
APSTRĀDE	Ierīce, kuras tips ir "Alarm" tiek aktivizēta, kā rezultātā par to tiek paziņots serverim, kurš izsūta paziņojumu ierīces īpašniekam
IZVADDATI	Nav
ATKARĪBAS	FP.VM.02
KĻŪDU PAZIŅOJUMI	Nav

1.3.2.3 Ierīces statusa pieprasīšana

Identificējums	FP.IM.03
Mērķis	Noskaidrot aktuālo ierīces stāvokli
Ievaddati	"deviceID" - maskēts klasifikators sha32 kodējumā, 8 simbolu garumā
Apstrāde	Tiek saņemta atbilde no servera, kuru apstrādā, un tiek noteikts nepieciešamais ierīces stāvoklis, nesakritību gadījumā tas tiek nomainīts uz pēdējo aktuālo
Izvaddati	Tiek mainīts ierīces statuss
Atkarības	FP.LM.04
Kļūdu paziņojumi	Nav

1.3.3 Vēstures modulis

1.3.3.1 Vēstures pieprasījums

Identificējums	FP.VM.01
Mērķis	Sekot līdzi darbībām, kas ir veiktas ar konkrēto ierīci
Ievaddati	"deviceID" - maskēts klasifikators sha32 kodējumā, 8 simbolu garumā
Apstrāde	Tiek nolasīti ieraksti no "History" tabulas dilstošā secībā pēc datuma un laika, un attēloti klienta ierīcē
Izvaddati	"description" - Datubāzes līmenī aprēķināts vēstures darbību raksturojošs lauks
Atkarības	FP.VM.02
Kļūdu paziņojumi	"Something went wrong" – Serveris nespēj veikt darbību

1.3.3.2 Vēstures ierakstīšana

Identificējums	FP.VM.02
Mērķis	Veidot ierakstus ierīces vēstures uzskaitē
Ievaddati	"deviceID" - maskēts klasifikators sha32 kodējumā, 8 simbolu garumā "Username" - Teksta lauks robežās no 3 līdz 20 simboliem "Description" – Teksta lauks robežās no 0 līdz 1024 simboliem
Apstrāde	Veicot kādu darbību tiek veikts ierakts "History" tabulā
Izvaddati	Nav
Atkarības	Nav
Kļūdu paziņojumi	Nav

1.4 Ārējās saskarnes prasības

1.4.1 Lietotāja saskarne

- Saskarnei ir jābūt angļu valodā
- Lietotājam ir jāspēj identificēt katru sev piesaistīto ierīci
- Lietotāju ir jāinformē par katru sekmīgi vai nesekmīgi veikto saziņu ar ierīci vai serveri
- Kļūdām ir jābūt apstrādātām
- Lietotājam ir jāspēj lietot programmatūru neveicot nekādu papildus uzstādījumus operētājsistēmas līmenī
- Ja servera atbildes laiks ir lielāks par 1 sekundi, ir vizuāli jāatēlo procesa izpildes logu

1.4.2 Aparatūras saskarne

Saskarnei ir jābūt attēlojamai uz jebkura standartizēta Android telefona ekrāna izmēra

1.5 Nefunkcionālās prasības

1.5.1 Veiktspējas prasības

Programmatūrai ir jāspēj realizēt darbību ne vēlāk kā 3 sekundes pēc darbības pieprasījuma.

1.5.2 Drošība

Lietotāju parolēm ir jābūt šifrētām MD5 kodējumā. Ierīces dati un komunikācija tiek šifrēta CRC32 kodējumā. Tiek lietots HTTPS protokols drošai komunikācijai starp klientu un serveri.

1.5.3 Pārnesamība

Ierīcei ir jābūt viegli integrējamai jebkurā mājoklī. Reģistrētai ierīcei ir jābūt brīvi pārvietojamai, ja ir nepieciešams mainīt tās novietojumu. Programmatūrai ir jābūt viegli pieejamai un ērti uzstādāmai uz atbalstītām Android ierīcēm. Programmatūrai ir jāparedz n neatkarīgu lietotāju skaitu, paredzot, ka nepieciešamības gadījumā jābūt iespējai programmatūru migrēt uz atbilstošu vidi.

2 Programmatūras projektējuma apraksts

2.1 Ievads

Šī programmatūras projektējuma apraksta nolūks ir precīzi aprakstīt projekta "Mājas automatizācijas sistēma" uzbūvi un projektējumu. Sistēma sastāvēs no četrām daļām: Android aplikācija – izstrādāta izmantojot JAVA objektorientēto pieeju, arduino programmatūra apvienota ar pašizstrādātu ierīci – izstrādāta arduino valodā, kura ir bāzēta uz C++, Servera programmatūra – izstrādāta balstoties uz MVC pieejas, lietojot PHP valodu, Datubāze – Izstrādāta mysql datubāzes pārvaldības sistēmā.

2.2 Moduļu dekompozīcija

FP.LM Lietotāju modulis

Funkcijas:

- FP.LM.01 Reģistrācija
- FP.LM.02 Piekļuves pieprasīšana
- FP.LM.03 Lietotāja datu rediģēšana
- FP.LM.04 Ierīces darbības uzstādīšana
- FP.LM.05 Koplietotāju uzstādīšana
- FP.LM.06 Jaunas ierīces reģistrēšanas inicializācija
- FP.LM.07 Piederošo ierīču iegūšana
- FP.LM.08 Lietotāju meklēšana
- FP.LM.09 Notifikācijas

Atkarības: FP.IM, FP.VM

FP.IM Ierīces modulis

Funkcijas:

- FP.IM.01 Ierīces reģistrēšana,
- FP.IM.02 Ierīces stāvokļa nosūtīšana,
- FP.IM.03 Ierīces stāvokļa pieprasīšana

Atkarības: FP.LM, FP.IM

FM.VM Vēstures modulis

Funkcijas:

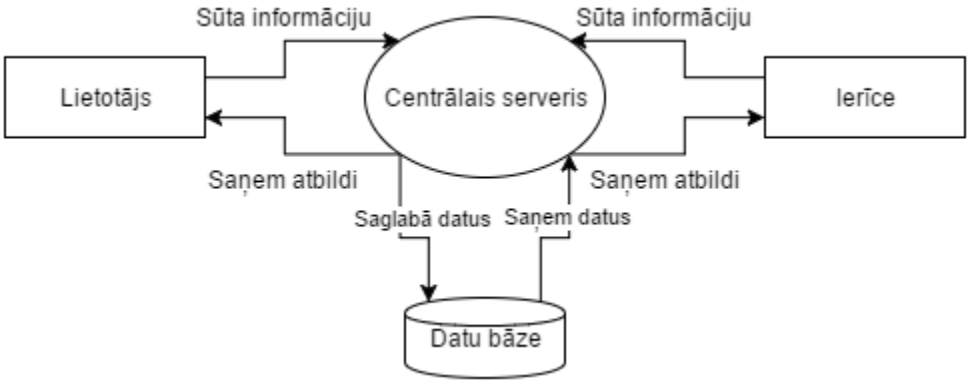
- FM.VM.01 Vēstures pieprasījums
- FM.VM.02 Vēstures ierakstīšana

Atkarības: Nav

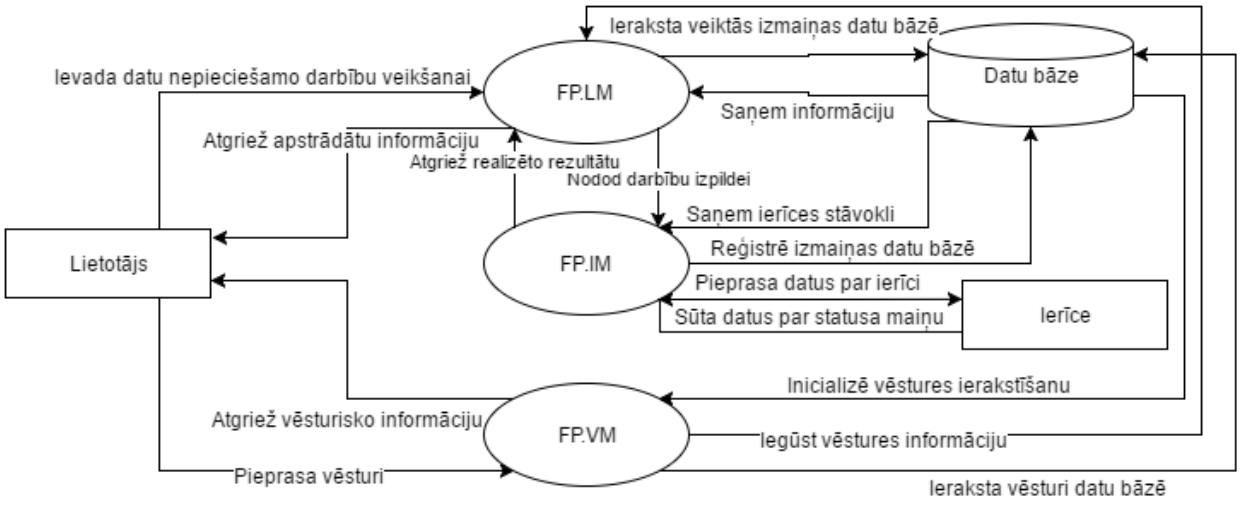
2.3 Datu plūsmu diagrammas

Šajā nodaļā ir vizuāli attēlota moduļu savstarpējā darbība.

2.3.1 DPD 0. līmenis

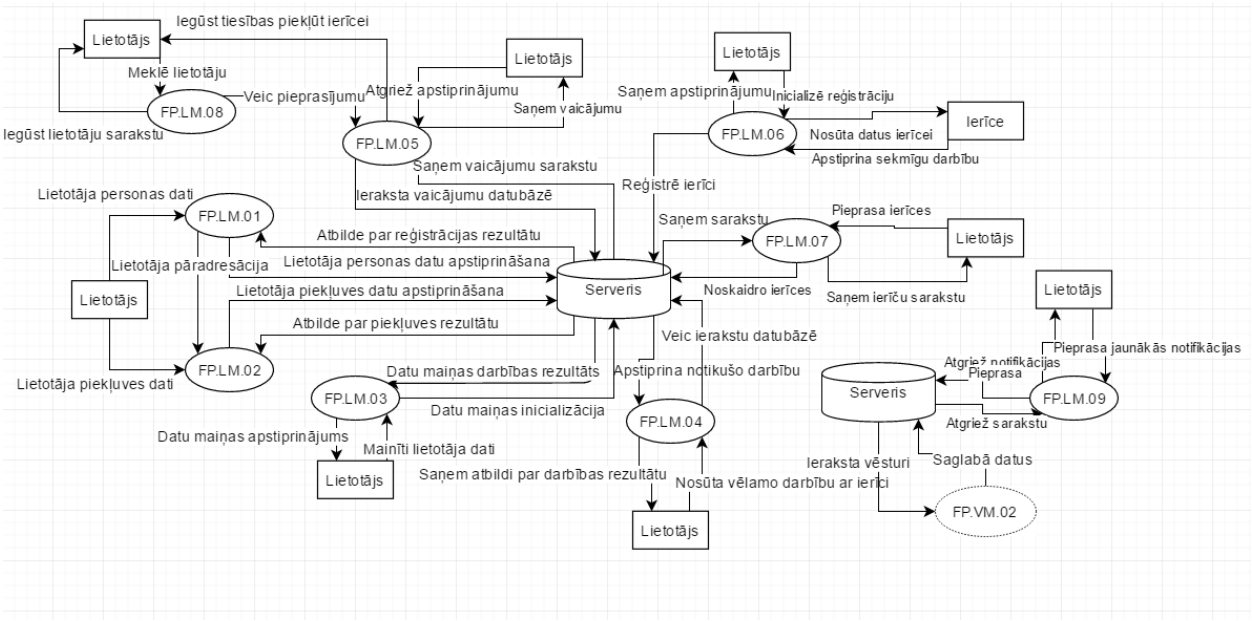


2.3.2 DPD 1. līmenis

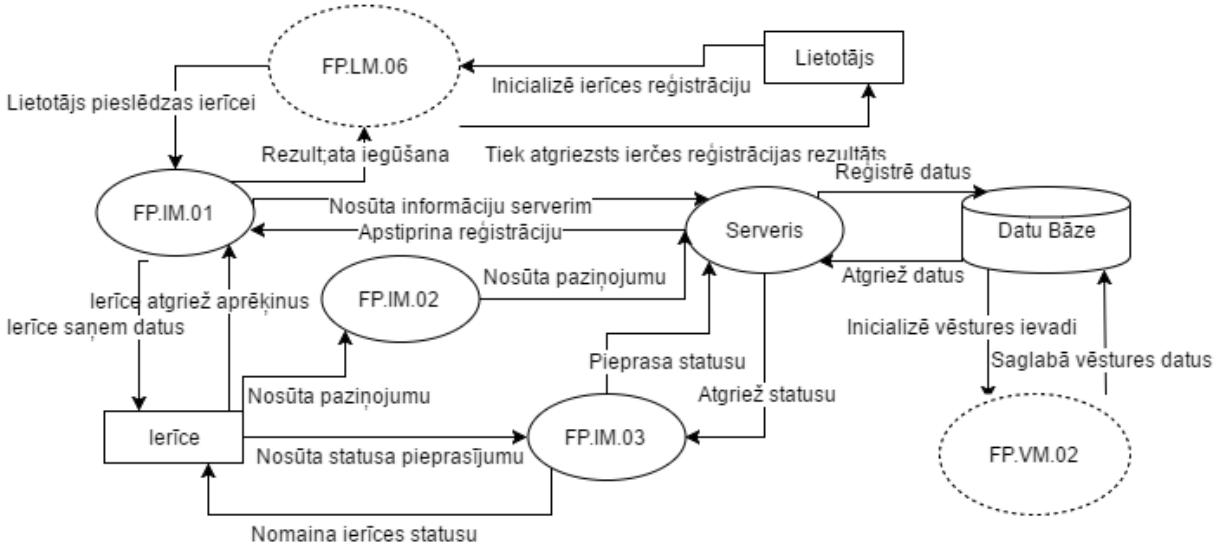


2.3.3 DPD 2. līmenis

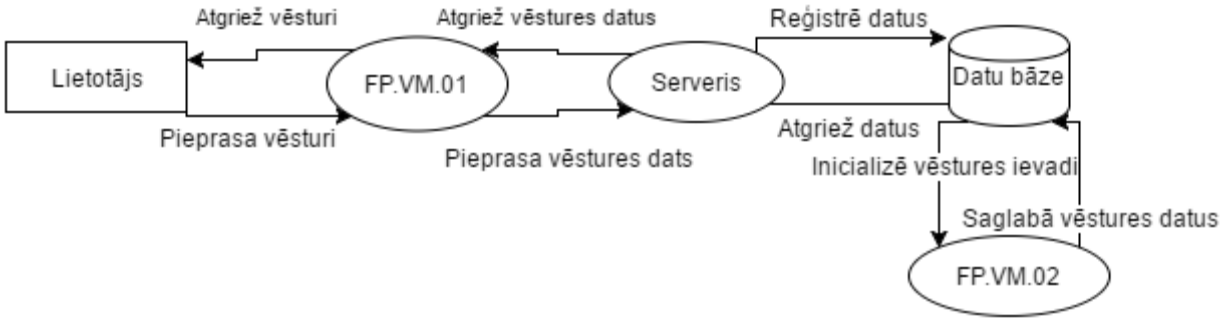
2.3.3.1 FP.LM Lietotāju modulis 2. līmeņa dekompozīcija



2.3.3.2 FP.IM Ierīces modulis 2. līmeņa dekompozīcija

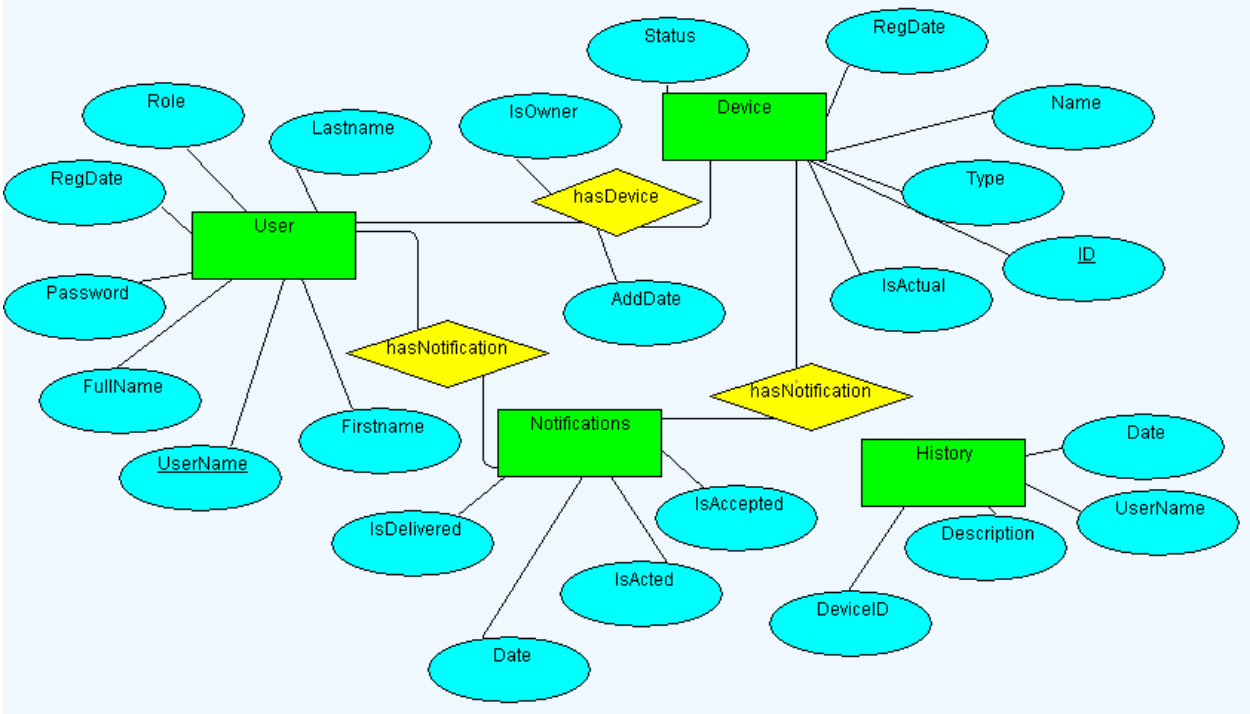


2.3.3.3 FP.VM Vēstures modulis 2. līmeņa dekompozīcija

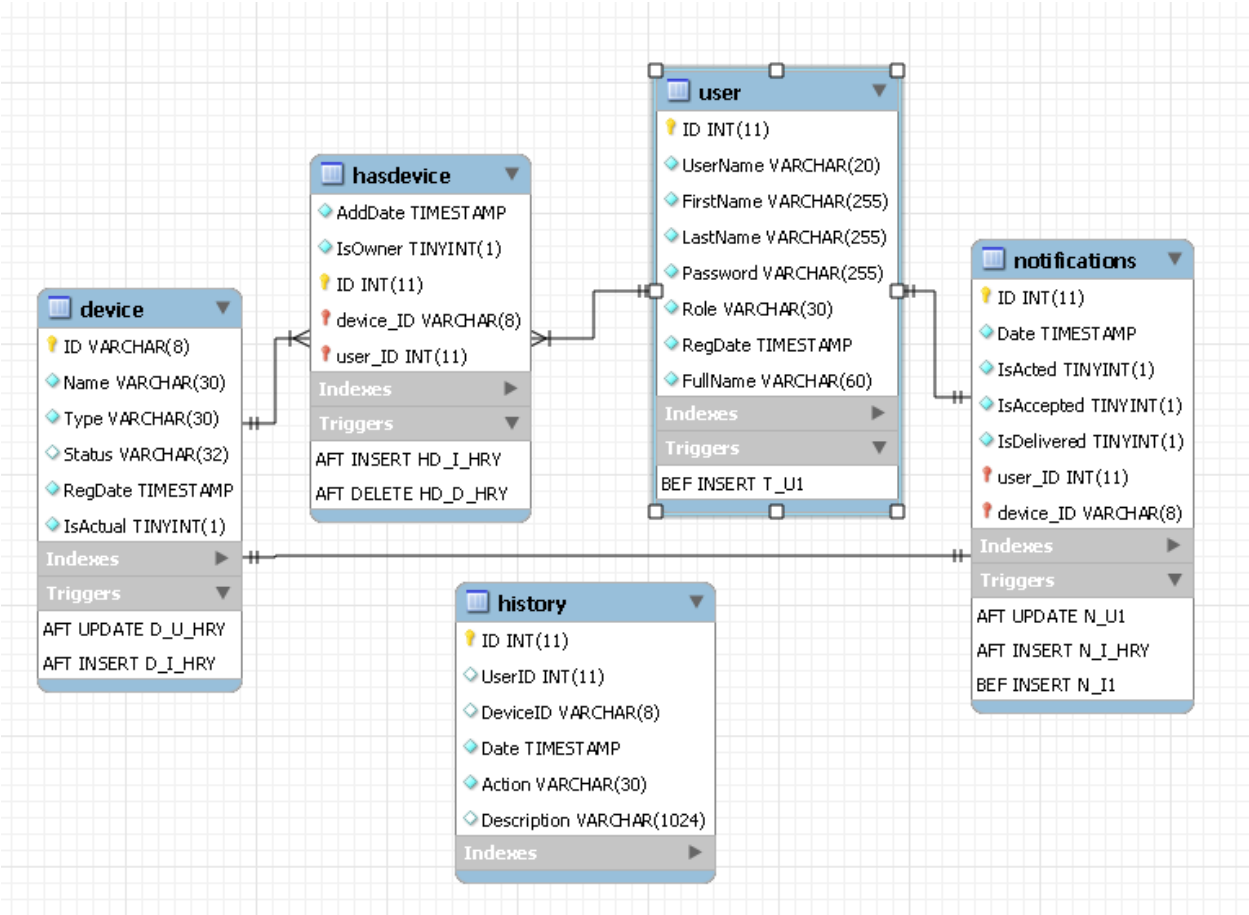


2.4 Datu bāzes projektējums

2.4.1 Konceptuālais ER modelis



2.4.2 Fiziskais datu bāzes modelis



2.4.3 Datubāzes detalizēts projektējums

Tabula: “User”

Lauks	Datu tips	Obligāts	Noklusētā vērtība	Atslēga	Apraksts
ID	INT	Jā		PK	Unikāls identifikators, inkriminējas par viens pie jauna ieraksta izveidošanas
UserName	VARCHAR(20)	Jā			Unikāla simbolu virkne garumā no 3 līdz 20 simboliem
FullName	VARCHAR(61)	Jā			Lauku ‘FirstName’ un ‘Lastname’ salikums, atdalot vienu no otra ar atstarpi
FirstName	VARCHAR(30)	Jā			Unikāla simbolu virkne garumā no 3 līdz 30 simboliem
LastName	VARCHAR(30)	Jā			Unikāla simbolu virkne garumā no 3 līdz 30 simboliem
Password	VARCHAR(32)	Jā			Simbolu virkne MD5 kodējumā
Role	VARCHAR(30)	Jā	‘User’		Lietotāja tiesību raksturojošs lauks ‘User’ vai ‘Admin’
RegDate	TIMESTAMP	Jā	CURRENT_TIMESTAMP		Reģistrācijas brīža datums, iegūts no šī brīža servera laika

Tabula: “Device”

Lauks	Datu tips	Obligāts	Noklusētā vērtība	Atslēga	Apraksts
ID	VARCHAR(8)	Jā		PK	Unikās identifikators, SHA32 kodējumā ģenerēta unikāla simbolu virkne
Name	VARCHAR(30)	Jā			Ierīces nosaukumu aprastošs lauks garumā no 3 līdz 30 simboliem
Type	Varchar(30)	Jā			Ierīces tipu raksturojoša vērtība: ‘Alarm’ vai ‘Switch’
Status	VARCHAR(32)	Jā	'90651ebea9a35ec4e018c8157492e17c'		Maskēts, darvību raksturojošs klasifikators MD5 kodējumā.
RegDate	TIMESTAMP	Jā	CURRENT_TIMESTAMP		Reģistrācijas brīža datums, iegūts no šī brīža servera laika
IsActual	THINYINT(1)	Jā	1		Ierīces aktualitātes raksturotājs: 0 – neaktuāla, 1 - aktuāla

Tabula: “History”

Lauks	Datu tips	Obligāts	Noklusētā vērtība	Atslēga	Apraksts
ID	INT	Jā		PK	Unikāls identifikators, inkriminējas par viens pie jauna ieraksta izveidošanas
UserID	INT	Nē		FK	Ārējā atslēga, kas norāda uz tabulas “User” lietotāju
DeviceID	VARCHAR(8)	Nē			Ierīces veiktās darbības raksturotājs, iegūts no ierīces servera pieprasījuma mainīgā
Date	TIMESTAMP	Jā	CURRENT_TIMESTAMP		Ieraksta veikšanas brīža servera datums
Description	VARCHAR(1024)	Jā			Darbību raksturojoša simbolu virkne

Tabula: “hasDevice”

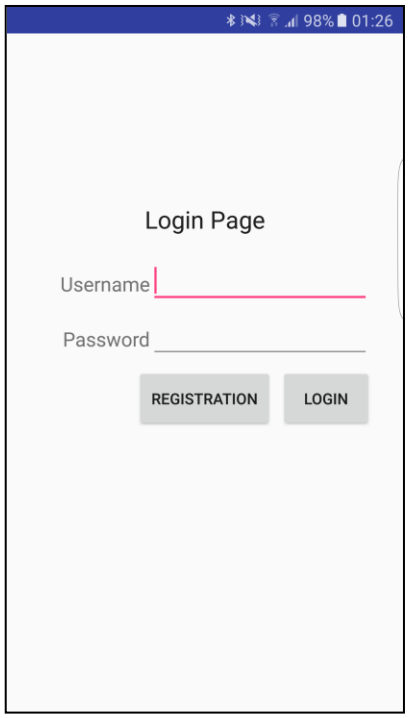
Lauks	Datu tips	Obligāts	Noklusētā vērtība	Atslēga	Apraksts
ID	INT	Jā		PK	Unikāls identifikators, inkriminējas par viens pie jauna ieraksta izveidošanas
UserID	INT	Jā		FK	Ārējā atslēga, norāda uz tabulas “User” lauku “Username”
DeviceID	VARCHAR(8)	Jā		FK	Ārējā atslēga, norāda uz tabulas “Device” lauku “ID”
AddDate	TIMESTAMP	Jā	CURRENT_TIMESTAMP		Saites izveides brīža servera datums
IsOwner	TINYINT(1)	Jā	0		Norāda uz lietotāja ierīces īpašumtiesībām: 0 – ierīce nepieder lietotājam, 1 – ierīce pieder lietotājam

Tabula: “Notifications”

Lauks	Datu tips	Obligāts	Noklusētā vērtība	Atslēga	Apraksts
ID	INT	Jā			Unikāls identifikators, inkriminējas par viens pie jauna ieraksta izveidošanas
user_ID	INT	Nē			Ārējā atslēga, norāda uz tabulas “User” lauku “Username”
device_ID	VARCHAR(8)	Jā			Ārējā atslēga, norāda uz tabulas “Device” lauku “ID”
Date	TIMESTAMP	Jā	CURRENT_TIMESTAMP		Ieraksta veikšanas brīža servera datums
IsActed	TINYINT(1)	Jā	0		Norāda vai lietotājs ir veicis darbību ar notifikāciju: ‘1’ – ir veicis, ‘0’ – nav veicis
IsAccepted	TINYINT(1)	Jā	0		Norāda vai notifikācija ir akceptēta: ‘1’ – ir akceptēta, ‘0’ – noraidīta. Nauks stājas spēkā, kad IsActed vērtība ir ‘1’
IsDelivered	TINYINT(1)	Jā	0		Norāda vai notifikācija ir piegādāta: ‘1’ – piegādāta, ‘0’ – nav piegādāta, jāsūta paziņojumu

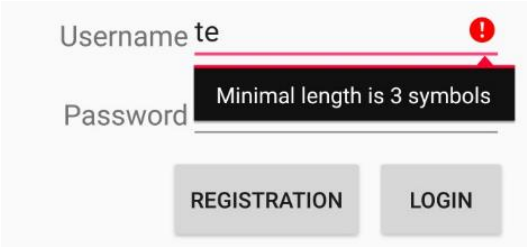
2.5 Ekrānformu projektējums

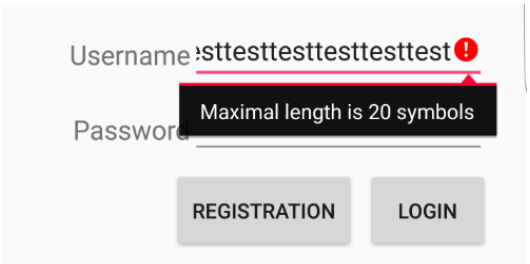
2.5.1 Lietotāja piekļuve

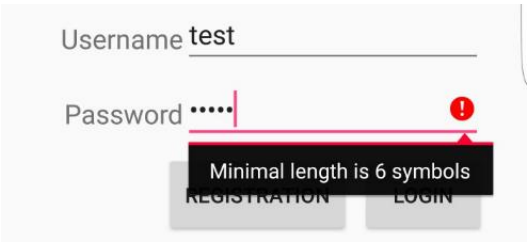


Skats paredzēts, lai dotu lietotājam iespēju pieslēgties aplikācijai un ļautu veikt manipulācijas ar ierīcēm. Ierakstot lietotājvārdu un paroli, un nospiežot pogu login tiek validēti lauku garumi. Lauks ‘Uername’ tiek pārbaudīts garumā no 3 līdz 20 simboliem, lauks ‘Password’ tiek pārbaudīts garumā no 6 līdz 255 simboliem. Nospiežot pogu “Registration” lietotājs tiek novirzīts uz reģistrācijas skatu. Parole tiek maskēta ar paroles aizstājējsiboliem.

Kļūdu paziņojumi:





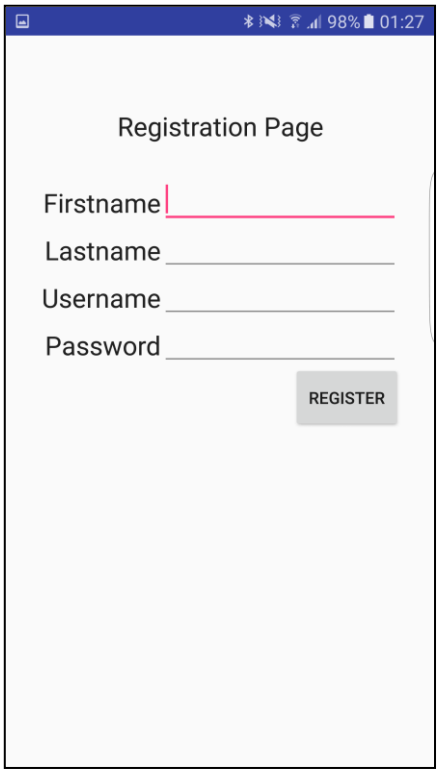


Tekstveida kļūdu paziņojumi:

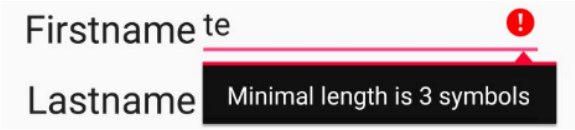
- Something went wrong

2.5.2 Reģistrācija

Skats paredzēts, lai dotu personalizētu iespēju lietotājam pieslēgties pie aplikācijas. Laukam “Username” ir jābūt unikālam. Veicot veiksmīgu reģistrācijas darbību nospiežot pogu “REGISTER” lietotājs tiek novirzīts uz “Ierīču skatu”. Visi lauki tiek validēti uz tam paredzēto garumu – Lauki “Firstname”, “Lastname” ir garumā no 3 līdz 30 simboliem, lauks “Username” ir garumā no 3 līdz 20 simboliem, lauks “Parole” ir maskēts ar paroles aizstājējsimboliem garumā no 6 līdz 255 simboliem.



Kļūdu paziņojumi:



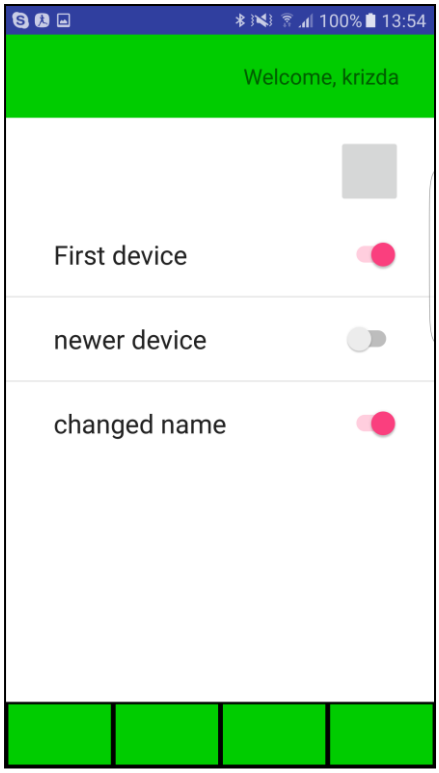
Tekstveida kļūdu paziņojumi:

- Wrong
- Something went wrong
- Username taken

2.5.3 Ierīču skats

2.5.3.1 Ierīču kontroles skats

Mērķfunkcionalitātes skats, kurš dod iespēju kontrolēt un iegūt informāciju par lietotājam piederošām un tiesību piešķirtām ierīcēm. Nospiežot uz ierīces slēdža pogas tiek nomainīts ierīces stāvoklis uz ieslēgts/izslēgts. Sekmīgas darbības rezultātā tiek attēlots tekstveida paziņojums "Success". Nospiežot virsū uz ierīces tiek atvērts ierīces informāciju raksturojošs logs par konkrēto ierīci.



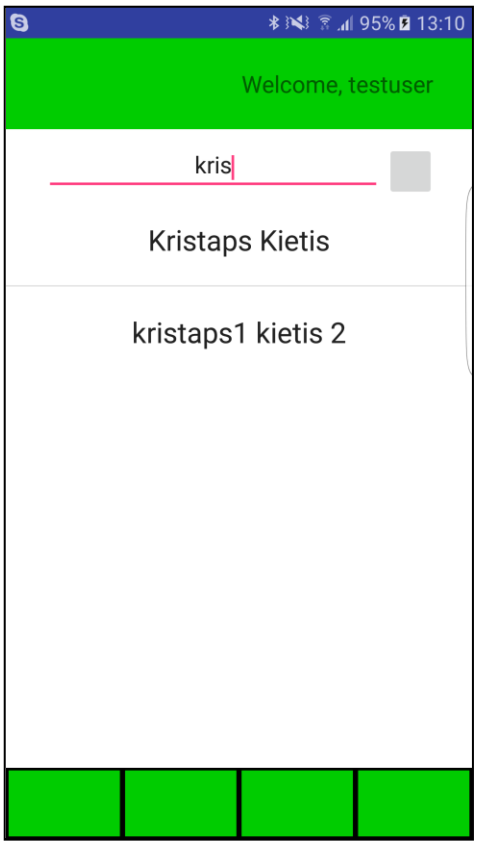
Tekstveida kļūdu paziņojumi:

- Something went wrong

2.5.3.2 Ierīces informācija

Ierīces informācijas skats, kurš attēlo informāciju par lietotāja izvēlētu ierīci. Sarakstā "Assigned users" ir attēloti visi lietotāji, kuriem ir piekļuve lietot ierīci. Nospiežot pogu, kas novietota tieši blakus lietotājam, tas tiek atsaistīts no ierīces. Šī funkcionalitāte ir iespējama tikai lietotājam, kurš ir reģistrējis ierīci, pārējiem šis saraksts ir tikai informatīvos nolūkos. Veiksmīgas atsaistes rezultātā tiek attēlots tekstveida paziņojums "Success", un poga tiek bloķēta. Saraksts "Device history" ir informatīvs saraksts, kurš glabā visus notikumus, kas ir saistīti ar konkrēto ierīci. Saraksts kārtots pēc notikuma datuma dilstošā secībā.

2.5.4 Meklēšanas skats



Skats paredzēts lai dotu iespēju lietotājiem pieprasīt ierīces lietošanas tiesības. Lietotājam ir jāievada vismaz 3 simboli, kā rezultātā tiks atlasīti visi lietotāji, kuru vārda un uzvārda salikumā būs šī simbolu virkne. Nospiežot uz lietotāja virsū tiek attēlotas visas ierīces, kuras šobrīd ir piedāvātas tiesību iegūšanai. Ja lietotājam ir ierīces, un, piemēram, visas pieder vai šobrīd jau tiek pieprasītas tiesības, tad šādām ierīcēm nebūs iespējams to izdarīt atkārtoti, līdz nebūs veikta darbība ar notifikāciju.

Nosiežot uz lietotāja tiek attēlots logs:

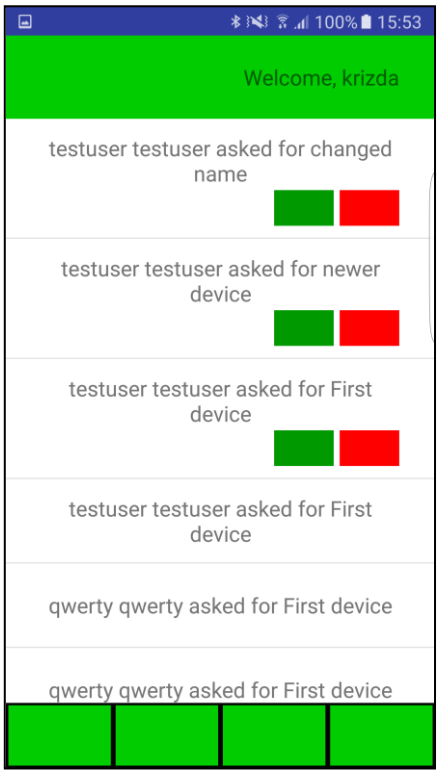
Šajā logā nospiežot uz pelēkās pogas tiek izsūtīts piekrasījums lietot ierīci. Sekmīga rezultāta gadījumā tiek attēlots tekstveida paziņojums “Success”.

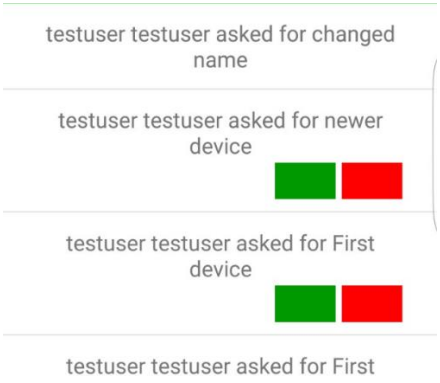
Tekstveida kļūdu paziņojumi:

- Something went wrong

2.5.5 Notifikāciju skats

Notifikāciju skats paredzēts, lai ierīces īpašnieks spētu piešķirt tiesības lietotājam pēc tiesību pieprasījuma. Zaļā poga nozīmē – piešķirt tiesības, sarkanā – noraidīt. Pogas attēlojas tikai neapstrādātiem pieprasījumiem. Apstiprinot vai noraidot pieprasījumu, pogas tiek maskētas. Teksts tiek attēlots formā ‘Fullname’ + ‘asked for’ + ‘DeviceName’. Veiksmīgas darbības rezultātā tiek attēlots tekstveida paziņojums “Success”



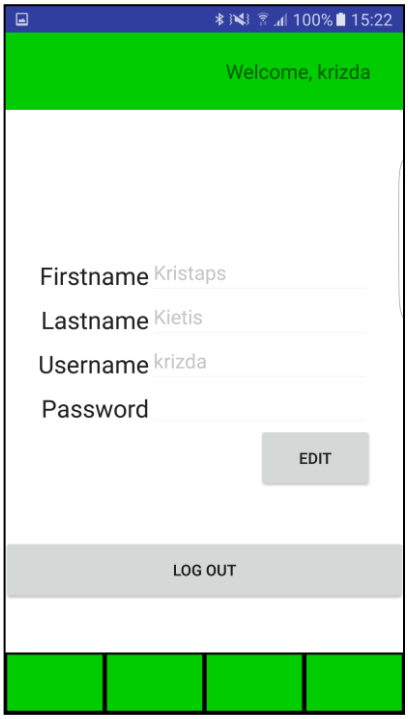


Klūdu paziņojumi:

- Something went wrong

2.5.6 Lietotāja rediģēšana

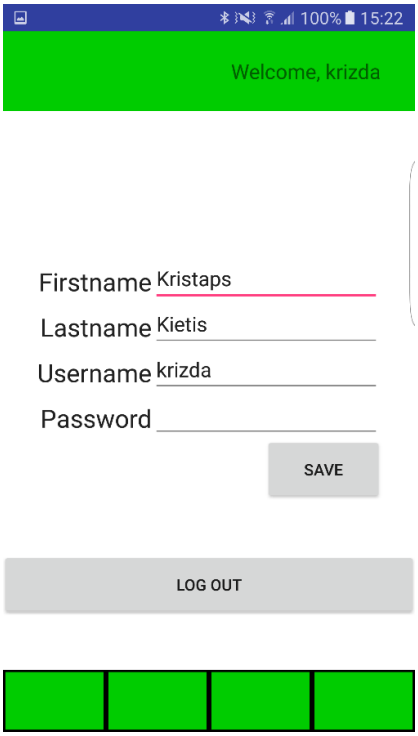
Lietotāju rediģēšanas skats paredzēts, lai lietotājam būtu iespēja mainīt sev personalizētu informāciju. Visi lauki pēc noklusējuma ir atspējoti. Nospiežot pogu "EDIT", parādās iespēja labot laukus, un poga pārtop par "SAVE" pogu. Visi lauki tiek automātiski aizpildīti balstoties uz šī brīža aktuālo informāciju. Paroles lauks ir tukšs līdz lietotājs veic tā satura maiņu. Paroles lauks tiek validēts tikai, ja lietotājs ir mainījis tā saturu. Tukšs – parole netiek mainīta. Poga "LOG OUT" aptur tekošo lietotāja sesiju, un dzēš piekļuves datus. Nospiežot šo pogu, klients tiek novirzīts uz "Lietotāja piekļuves" skatu.



Kļūdu paziņojumi:

Tekstveida kļūdu paziņojumi:

- Something went wrong

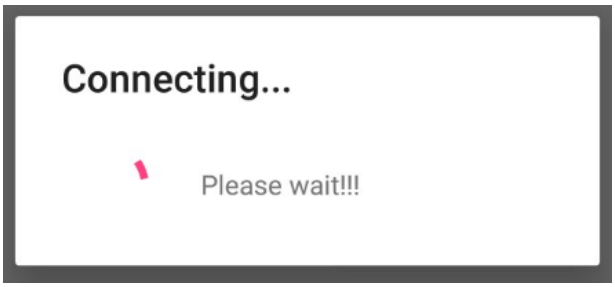


2.5.7 Ierīces reģistrācijas skats

2.5.7.1 Savienošanās ar ierīci

Skats paredzēts, lai izveidotu unikālu savienojumu ar ierīci, pirms tā ir piesaistīta bezvadu tīklam. Atverot skatu tiek attēlotas tuvākās Bluetooth ierīces. Nospiežot uz ierīces identificējama nosaukuma, tiek veidots savienojums ar to. Nospiežot pogu "Show devices" tiek atjaunots ierīču saraksts.

Savienošanās skats.

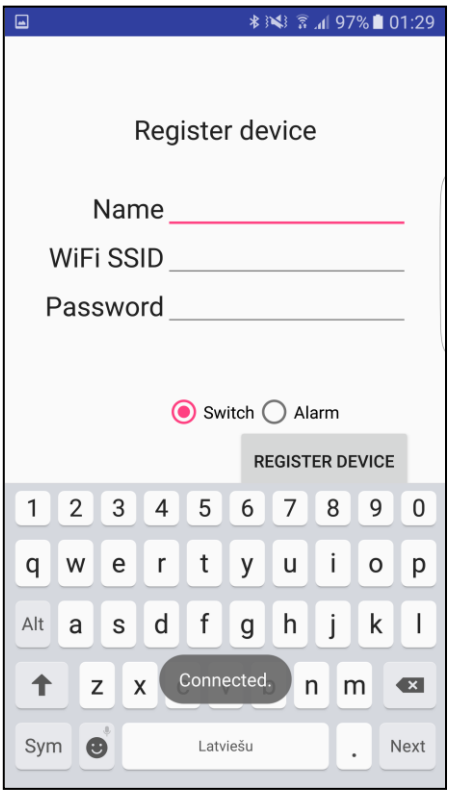


Sekmīga savienojuma rezultājā lietotājs tiek novirzīts uz Reģistrācijas datu ievadi.

Tekstveida kļūdu paziņojumi:

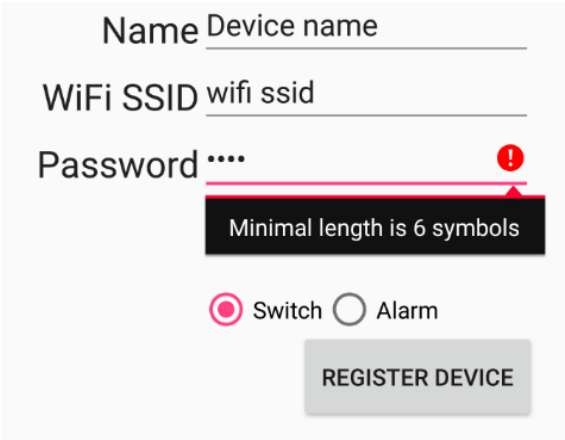
- Connection failed

2.5.7.2 Ierīces reģistrācijas datu ievade



Skats paredzēts, lai lietotājs varētu reģistrēt jaunu ierīci sistēmā un dot iespēju tai savienoties ar bezvadu tīklu. Laukā "Name" tiek ievadīts lietotāja izvēlēts ierīces nosaukums, garumā no 3 līdz 30 simboliem. Laukā "WiFi SSID" tiek ievadīts lietotājam piederoša bezvadu tīkla nosaukums, garumā no 3 līdz 32 simboliem. Laukā "Password" tiek ievadīta wifi tīkla parole, garumā no 6 līdz 255 simboliem. Lietotājam tiek dota iespēja izvēlēties reģistrējamās ierīces veidu, noklusētā vētība ir "Switch". Nospiežot pogu "REGISTER DEVICE" tiek inicializēta ierīces reģistrācija. Sekmīgas reģistrācijas gadījumā tiek attēlots tekstveida paziņojums "Device registred" un lietotājs tiek novirzīts uz ierīču skatu.

Kļūdu paziņojumi:



Tekstveida kļūdu paziņojumi:

- Error
- Check your network connection
- Wrong network parameters
- Device in wrong state

3 Testēšanas dokumentācija

3.1 Ievads

Veicot programmēšanas darbus tika mēģināts maksimāli izvairīties no jebkādām kļūdām, kas varētu ietekmēt programmatūras kopējo darbību, pārbaudot vai tās strādā atbilstoši PPS prasībām. Programmatūras testēšana tika iesākta brīdī, kad lietotājs varēja pilnīgi lietot programmatūras pamatfunkcionalitāti.

3.2 Testēšanas žurnāls

3.2.1 Lietotāju modulis

3.2.1.1 Reģistrācija

Nr.	Testpiemēra apraksts	Vēlamais rezultāts	Rezultāts
1.	Lietotājs var atvērt reģistrācijas formu, un to var izdarīt tikai no "Lietotāja piekļuves" formas	Lietotājam tiek attēlota reģistrācijas forma	Izpildās
2.	Lietotāja ievadītie dati tiek validēti uz programmatūras aprastā noteiktā lauku garuma	Nepareizi ievadītiem laukiem tiek attēlots kļūdas paziņojums	Izpildās
3.	Parole tiek maskēta	Ievadot paroli nav iespējams noteikt, kas tur ir rakstīts	Izpildās
4.	Tiek pārbaudīts vai lietotājs spēj reģistrēties sistēmā	Lietotājs spēj piekļūt sistēmai, un datubāzē tiek glabāti ievadītie dati	Izpildās
5.	Interneta savienojuma problēmu apstrāde	Lietotājam tiek attēlots paziņojums par neizdevušos darbību	16.05.2016 Izpildās daļēji Ja ir ievērojama noildze, netiek vizuāli attēlots process 19.05.2016

			Izpildās
6.	Lietotājs saņem paziņojumu par aizņemtu lietotājvārdu	Lietotājs tiek informēts	16.05.2016 Neizpildās Nav iespējams noteikt par ko ir attēlotā kļūda 19.05.2016 Izpildās

3.2.1.2 Piekļuves pieprasīšana

Nr.	Testpiemēra apraksts	Vēlamais rezultāts	Rezultāts
1.	Atverot aplikāciju lietotāja piekļuves forma atveras kā pirmā.	Izslēdzot aplikāciju tiek attēlota tieši lietotāja piekļuves skats, ja nav saglabāta sesija	Izpildās
2.	Lietotāja ievadītie dati tiek validēti uz programmatūras aprastā noteiktā lauku garuma	Nepareizi ievadītiem laukiem tiek attēlots kļūdas paziņojums	Izpildās
3.	Parole tiek maskēta	Ievadot paroli nav iespējams noteikt, kas tur ir rakstīts	Izpildās
4.	Aplikācijai ir jāatveras ierīču kontroles skatā, ja ir saglabājusies piekļuve	Lietotājam pieslēdzoties sistēmai, to darot atkātoti, nav jāievada piekļuves dati.	Izpildās
5.	Aplikācijai ir jāpārtrauc sesiju, nospiežot pogu “Log out”	Ieslēdzot aplikāciju atkārtoti nav saglabājusies iepriekšējā sesija	Izpildās
6.	Tiek attēlots paziņojums pie nesekmīgas pieslēgšanās	Lietotājs saņem paziņojumu	Izpildās

7.	Interneta savienojuma problēmu apstrāde	Lietotājam tiek attēlots paziņojums par neizdevušos darbību	16.05.2016 Izpildās daļēji Ir novērojama liela noildze līdz paziņojuma saņemšanai. 19.05.2016 Izpildās
8.	Sekmīgas piekļuves rezultātā lietotāju ir jānovirza uz ierīču skatu	Lietotājs piekļūst ierīču skatam	Izpildās

3.2.1.3 Lietotāju datu rediģēšana

Nr.	Testpiemēra apraksts	Vēlamais rezultāts	Rezultāts
1.	Lietotājam ir iespēja piekļūt pie formas	Lietotājs redz formu	Izpildās
2.	Lietotājam nav iespējams mainīt lauku saturu nenospiežot pogu “EDIT”	Nav mainīti lietotāja dati	Izpildās
3.	Visi lauki, izņemot lauku “Password” ir automātiski aizpildīti	Lietotājs redz savus datus, izņemot paroli	Izpildās
4.	Nospiežot pogu “Edit” parādās iespēja labot laukus, un poga “Edit” pārtop par “Save” pogu.	Lietotājs spēj rediģēt datus	Izpildās

5.	Paroles lauks kļūst aktuāls, ja tajā ir ievadīts vismaz 1 elements	Lietotājs spēj rediģēt datus bez paroles maiņas, un ar paroles maiņu	Izpildās
6.	Nospiežot pogu "Save" lietotāja dati tiek nomainīti	Lietotājs redz jauno vērtību, datubāzē tiek saglabāti dati, attēlojas paziņojums par datu maiņu	16.05.2016 Izpildās daļēji Dati veiksmīgi nomainīti, taču nenomainās lietotājvārds augšējā labajā stūrī, paziņojums attēlojas veiksmīgi 19.05.2016 Izpildās

3.2.1.4 Ierīces aktivizēšana

Nr.	Testpiemēra apraksts	Vēlamais rezultāts	Rezultāts
1.	Ierīces skatā nospiežot slēdža pogu, ja ierīce ir ar tipu "Switch", tai ir jāieslēdzas vai jāizslēdzas	Nospiežot pogu, ierīce maina stāvokli mazāk kā 3 sekundēs	Izpildās, vidējais laiks <1 sekundi, maksimālais laiks < 3 sekundēm
2.	Ierīces skatā nospiežot slēdža pogu, ja ierīce ir ar tipu "Alarm", tai nav jāsūta lietotājam paziņojumu, ja tā ir izslēgta, ir jāsūta, ja ir ieslēgta	Lietotājs saņem paziņojumu, ja ir ieslēgta, nesaņem notifikāciju, ja ir izslēgta	Izpildās

3.2.1.5 Ierīces lietotāju piekļuves pieprasījums

Nr.	Testpiemēra apraksts	Vēlamais rezultāts	Rezultāts
1.	Atrodot lietotāju(FP.LM.08) ir iespēja izsūtīt vaicājumu pēc ierīces	Ierīces īpašnieks saņem paziņojumu par vaicājumu	Izpildās
2.	Ierīces īpašniekam ir iespējams apstiprināt vaicājumu, kā rezultātā tiek dotas tiesības lietot ierīci	Apstiprinot – vaicātājam ir iespēja vadīt ierīci Noraidot – vaicātājam nav iespējas vadīt ierīci	Izpildās
3.	Tiek attēlots paziņojums par darbības izpildi	Lietotājs redz rezultātu	Izpildās

3.2.1.6 Jaunas ierīces reģistrēšanas inicializācija

Nr.	Testpiemēra apraksts	Vēlamais rezultāts	Rezultāts
1.	Nospiežot pogu “Reģistrēt jaunu ierīci” tiek attēlots saraksts ar bluetooth ierīcēm	Lietotājam ir iespēja izvēlēties ierīci, un tai pieslēgties, rezultātā tiek atvērta reģistrācijas forma	16.05.2016 Izpildās daļēji Ierīces pieslēgumam ir jābūt iepriekš saglabātam caur standartizētiem bluetooth pieslēgumiem
2.	Lietotāja ievadītie dati tiek validēti uz programmatūras aprastā noteiktā lauku garuma	Nepareizi ievadītiem laukiem tiek attēlots kļūdas paziņojums	Izpildās
3.	Ierīce paziņo par nespēju pieslēgties bezvadu tīklam	Lietotājs tiek informēts par nespēju pieslēgties tīklam	16.05.2016 Neizpildās 20.05.2016 Izpildās

4.	Lietotājs saņem paziņojumu par veiksmīgu ierīces reģistrāciju	Lietotājs saņem paziņojumu, un forma aizveras	16.05.2016 Neizpildās 20.05.2016 Izpildās
5.	Ierīces dati saglabājas datubāzē un ir iespēja veikt darbības ar ierīci	Saglabājas dati un lietotājam parādās iespēja lietot ierīci	Izpildās

3.2.1.7 Piederošo ierīču iegūšana

Nr.	Testpiemēra apraksts	Vēlamais rezultāts	Rezultāts
1.	Atverot ierīču sarakstu lietotājam tiek attēlotas visas ierīces ar kurām tas drīkst veikt manipulācijas	Dati tiek attēloti pareizi, un ir iespējama funkcionalitāte	Izpildās
2.	Nav iespējams rediģēt ierīces, kurām lietotājs nav īpašnieks	Ir apslēpta īpašnieka funkcionalitāte	Izpildās

3.2.1.8 Lietotāju meklēšana

Nr.	Testpiemēra apraksts	Vēlamais rezultāts	Rezultāts
1.	Tiek pārbaudīts vai meklējamais vārds ir garumā vismaz 3 simboli	Attēlots kļūdas paziņojums, ja garums mazāks par 3 simboliem, tiek atlasīti lietotāji, ja garums ir vismaz 3 simboli	Izpildās
2.	Tiek attēlots paziņojums, ja nav atrasts neviens lietotājs	Lietotājs saņem paziņojumu	16.05.2016 Neizpildās 19.05.2016 Izpildās

3.	Tiek atlasīts jebkurš aplikācijas lietotājs, un tikai tam piederošās ierīces, kurām ir iespēja pieprasīt lietošanas tiesības	Lietotājam attēlojas pareizs meklēšanas rezultāts	16.05.2016 Izpildās Tiks mainīta atlase, nepieciešams pārtestēt pēc labojumiem 21.05.2016 Izpildās

3.2.1.9 Notifikācijas

Nr.	Testpiemēra apraksts	Vēlamais rezultāts	Rezultāts
1.	Lietotājs saņem paziņojumu par aktivizētu signalizāciju	Lietotājs ir informēts par signalizācijas aktivizēšanu arī, ja aplikācija nav atvērta. Nospiežot virsū tiek atvērta aplikācija	Izpildās
2.	Lietotājs saņem paziņojumu, ja kāds pieprasa lietošanas tiesības viņa ierīcei	Lietotājs ir informēts par tiesību pieprasījumu. Nospiežot virsū tiek atvērta aplikācija	16.05.2016 Neizpildās 19.05.2016 Izpildās
3	Lietotājs nesaņem notifikāciju, ja ir nospiedis pogu “Log Out”(Nav salabājis sesiju)	Lietotājs manuāli atslēdzas no sistēmas, un nesaņem šajā laikā notifikācijas	Izpildās
4	Paziņojumam ir jābūt saklausāmam un sajūtamam, ja paziņojums ir par signalizācijas iedarbošanos.	Lietotājs dzird un jūt paziņojumu	Izpildās

5	Paziņojumam ir jābūt redzamam, ja tā nolūks ir attēlot, ka kāds ir pieprasījis ierīces lietošanas tiesības	Atverot aplikāciju lietotājs vizuāli redz paziņojumu, līdz aplikācijas atvēršanai mirgo sarkana lampiņa(Ierobežojums: Ierīcei ir jābūt ar tādu aprīkotai)	Izpildās

3.2.2 Ierīces modulis

Ierīces reģistrēšana aprakstīta nodaļā – 3.2.1.6

3.2.2.1 Ierīces statusa nosūtīšana

Nr.	Testpiemēra apraksts	Vēlamais rezultāts	Rezultāts
1.	Aktivizējot signalizāciju ierīce par to paziņo serverim	Izveidojas ieraksts datubāzē	Izpildās
2.	Signalizācija ieciklojas stāvoklī “Aktīvs”	Ieraksts datubāzē tiek veikts vienreiz stundā	Izpildās

3.2.2.2 Ierīces statusa pieprasīšana

Nr.	Testpiemēra apraksts	Vēlamais rezultāts	Rezultāts
1.	Ierīce spēj mainīt stāvokli starp “Ieslēgts/Izslēgts”	Ierīce nomaina stāvokli	Izpildās
2.	Signalizācijas tipa ierīce uzstāda normālpozīciju pie tās ieslēgšanas	Ierīce uzstāda attālumu centimetros(Normālstāvokli)	Izpildās

3.2.3 Vēstures modulis

3.2.3.1 Vēstures pieprasījums

Nr.	Testpiemēra apraksts	Vēlamais rezultāts	Rezultāts
1.	Lietotājs vizuāli spēj apskatīt ierīces vēsturi	Lietotājs redz konkrētas ierīces darbību vēsturi	Izpildās

3.2.3.2 Vēstures ierakstīšana

Nr.	Testpiemēra apraksts	Vēlamais rezultāts	Rezultāts
1.	Notiekot iepriekš definētām darbībām tiek saglabāti strukturēti pareizi dati	Datu bāzē ir vizuāli redzami saglabātie dati	Izpildās

4 Projekta organizācija

Šo tēmu kvalifikācijas darba autors izvēlējās, jo redz lielu potenciālu gudro māju attīstībā, un jau agrāk nelielā līmenī ir saskāries ar zema līmeņa mājas automatizēšanu, un vēlējās izveidot vienotu mājas automatizācijas sistēmu, kas ļauj to lietot no "Sociālā tīkla" puses, kur pastāv iespēja lietot arī draugu ierīces.

Projekts tika izstrādāts pēc tuvināta ūdenskrituma modeļa. Sākumā tika veikta vispārēja sistēmas un iespēju analīze, kurai paralēli tapa Programmatūras Prasību Specifikācija. Pēc tam tika veikti programmēšanas darbi, kas balstījās uz PPS, tika veikti arī vairāki labojumi, kuri nebija iepriekš paredzēti. Paralēli programmēšanai notika arī vairāki vienībtesti, kas netika reģistrēti testēšanas žurnālā. Brīdī, kad programmatūra sasniedza stabilu dzīves ciklu tika uzsākta Programmatūras Prasību Specifikācijas rakstīšana, kā arī uzsākta un reģistrēta testēšana. Programmatūra ir uzstādīta uz publiski pieejama servera, lai varētu pārliecināties par tās stabilitāti.

5 Kvalitātes nodrošināšana

Lai nodrošinātu kvalitatīvu programmatūras attīstību, tika sastādīta dokumentācija atbilstoši standartam LVS 68:1996 „Programmatūras prasību specifikācijas ceļvedis". Programmējot tika ievērots “Labais programmēšanas stils”. Tā kā es biju vienīgais programmētājs, un man nebija jāpaļaujas uz citu darbu, tāpēc visi programmatūras moduļi tika programmēti paralēli, lai laicīgi identificētu problēmas un nodrošinātu korektu moduļu savstarpēju sadarbību.

Programmatūras izstrādes laikā tika veikti regresa testi, taču iestājoties stabilam programmatūras dzīves ciklam, tika uzsākta arī testēšanas dokumentēšana.

6 Konfigurācijas pārvaldība

Konfigurācijas pārvaldībai tika izmantos Latvijas Universitātes piedāvātā GIT koda versiju pārvaldības sistēma, kas glabā visas koda izmaiņas, kas veiktas koda izstrādē. Papildus programmatūras kodam tika versionēta (glabāta) arī kvalifikācijas darba rakstiskā daļa.

Faili tika glabāti pēc struktūras:

- Kvalifikācijas darbs
    - Android izstrade
        - HomeAutomation
    - Arduino
        - 1.ierīce.ino
    - Datubāzes izstrāde
        - ER Modelis
        - Triggeri
        - Datubāzes veidošanas skripts.sql
    - Kvalifikācijas darba izstrāde
        - DPD(Datu pūsmu diagrammas)
        - Klasifikatori.xlsx
        - Moduļi un to dekompozīcija.docx
    - Servera izstrāde
        - HomeAutomation
    - Kvalifikācijas darbs.docx

Tiek glabāts arī datubāzes izstrādes skripts, lai būtu ērti integrēt to uz jaunas vides.

Vizuāli ir apskatāms kvalifikācijas darba izstrādes process:

7 Darbietilpības novērtējums

Darbietilpības novērtējumam tika izmantota COCOMO darbietilpības metode., kura tika ņemta par pamatu gala rezultāta novērtēšanai.

Kopējs gala produkts, izstrādāts uz vairākām vidēm, un ar pilnu funkcionalitāti neiekļautos kvalifikācijas darbam atvēlētajam laika posmam, tāpēc kvalifikācijas darba ietvaros funkcionalitāte tika samazināta līdz stabilas un vienkāršas programmatūras realizējumam, un lietotāja saskarne tika ierobežota līdz "Android" operētājsistēmas klientiem, taču viss programmatūras kods tika veidots struktūrā, kas ļautu viegli integrēties arī uz citām vidēm – Web, IOS, u.c.

Kvalifikācijas darba loģistikas posmā, apspriežoties ar kvalifikācijas darba vadītāju tika izveidots darba plāns priekš aptuveni 3 cilvēkmēnešiem.

Izmantojot "IntelliJ IDEA" rīku "Statistic" tika aprēķināts tīrais koda rindiņu skaits.

- Java(Android) – 1770 rindiņas
- XML(Android) – 1087 rindiņas
- PHP -1030 rindiņas
- Arduino – 385 rindiņas
- SQL – 128 rindiņas

Kopā tika aprēķinātas – 4400 rindiņu koda.

Izmantojot "Basic Cocomo" metodi, kas balstās uz formulu :

"Cilvēkmēneši = A*KLOC^B = 2.4 * (4400/1000)^1.05 = 11.37"

"Pavadītais laiks= C * (A*KLOC^B)^D = 2.5 * 11.37^0.38 = 6,3"

, kur A ir 2.4, B ir 1.05, C = 2.5, D = 0.38

Tika aprēķināts, ka kvalifikācijas darbā ieguldītais laika posms ir aptuveni 6.3 mēneši un 11,37 cilvēkmēneši.

Secinājumi:

Es uzskatu, ka nav adekvāti iespējams salīdzināt COCOMO aprēķināto darbietilpību ar iepriekš novērtēto darbietilpību. Manuprāt, reālais pavadītais laiks, kas tika lietots izstrādājot kvalifikācijas darbu ir tuvāks iepriekš secinātajam. Darba gaitā, man izdevās rakstīt kodu bez

lielām aizķeršanām, kas nozīmē, ka viennozīmīgi bija vairākas dienas, kurās tika padarīts daudz vairāk par atvēlēto laiku šim projektam, kas neizslēdz arī COCOMO patiesumu.

8. Izmantotā literatūra un rīki

1. LVS 68:1996 „Programmatūras prasību specifikācijas ceļvedis"
2. LVS 72:1996 “Ieteicamā prakse programmatūras projektējuma aprakstīšanai
3. http://groups.engin.umd.umich.edu/CIS/course.des/cis525/js/f00/kutcher/kutcher.html
4. http://git.df.lu.lv/kk14056/kvalifikacijas-darbs

9 Programmatūras kods

9.1 Android kods

Kvalifikācijas darbam tiek pievienota daļa koda no izstrādes pakas "com.kkietis.kristapskietis.homeautomation.DevicePackage"

9.1.1 Klase Device

```
package com.kkietis.kristapskietis.homeautomation.DevicePackage;

import android.content.Context;
import android.widget.Switch;
import android.widget.Toast;

import com.kkietis.kristapskietis.homeautomation.Core.BackgroundTasks;
import com.kkietis.kristapskietis.homeautomation.Core.CallBack;
import com.kkietis.kristapskietis.homeautomation.Core.ValueStore;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

/**
 * Veidoja Kristaps Kietis 4/3/2016 datumā.
 * Klase paredzēta ierīču objektu apstrādei
 */

public class Device {
    final public static String STATUS_ON = "90651ebea9a35ec4e018c8157492e17c";
    private final static String STATUS_OFF = "88559a0cfd8250c9d65970cc145c92d4";
    public String deviceID; // Ierīces unikāls identifikators
    public String deviceName; // Ierīces nosaukums
    public String deviceStatus; //Ierīces šī brīža status
    public String deviceType; // Ierīces tips
    public Boolean isActual; // Ierīces aktualitātes patiesuma mainīgais
    public Boolean isOwner; // Norāda vai piesaistitais lietotājs ir ierīces īpašnieks

    public Device(JSONObject device, Boolean isSearch) throws JSONException {
        deviceID = device.get("id").toString();
        deviceName = device.get("name").toString();
        if(!isSearch){//Ierīču objektam, kurš nepieder īpašniekam, nedrīkt lietot šos
mainīgos
            deviceStatus = device.get("status").toString();
            deviceType = device.get("type").toString();
            isActual = "1".equalsIgnoreCase(device.get("isactual").toString());
            isOwner = "1".equalsIgnoreCase(device.get("isowner").toString());
        }
    }

    /**
     * @param clickedSwitch
     * @param isChecked
     * @param context
     * Metode, kura maina ierīces stāvokli, kurš tiek nolasīts no android objekta
"Switch"
     */
    public void chageDeviceState(final Switch clickedSwitch, final Boolean isChecked,
final Context context){
```

```
        clickedSwitch.setClickable(false);//Apturam iespēju spiest pogu, pirms
serveris ir izdarījis aprēķinus un darbību
        String status = deviceState(isChecked); // Iegūst objekta "Switch" šī brīža
stāvokli
        String method = "setDeviceState"; // Izsaucamās metodes nosaukums
        BackgroundTasks backgroundTask = new BackgroundTasks(new CallBack() {
            @Override
            public void done(String result) {
                if (result == null || !result.equals("Success")) {
                    deviceStatus=deviceState(!isChecked); // Tiek uzstādīts ierīces
objekta stāvoklis
                    Toast.makeText(context, "Something went wrong",
Toast.LENGTH_SHORT).show();//paziņojums par izpildes rezultātu
                }else{
                    deviceStatus=deviceState(isChecked);
                    Toast.makeText(context, result,
Toast.LENGTH_SHORT).show();//paziņojums par izpildes rezultātu
                }

                clickedSwitch.setClickable(true);//Atļaujam atkārtoti pārslēkt ierīces
stāvokli
            }
        }, context);
        ValueStore vs = (ValueStore)context;//Iegūstam aplikācijas saglabātos mainīgos
        backgroundTask.execute(method, deviceID, vs.userName,status); // Tiek
izpildīts asinhrons uzdevums

    }

    /**
     * @param isOn
     * @return
     * Metode, kas patiesuma vērtību pārvērš klasifikatorā
     */
    private String deviceState(Boolean isOn){
        if(isOn){
            return STATUS_ON;
        }else{
            return STATUS_OFF;
        }
    }

    /**
     * @param stringJSON
     * @return
     * Metode pārveido JSON objektu par ierīču masīvu
     */
    public static Device[] createDeviceArray(String stringJSON){
        Device deviceArray[] = null;

        try {
            JSONArray deviceArrayJSON = new JSONArray(stringJSON);
            deviceArray = new Device[deviceArrayJSON.length()];
            for(int i=0;i<deviceArrayJSON.length();i++){
                JSONObject singleDeviceJSON = deviceArrayJSON.optJSONObject(i);
                Device singleDevice = new Device(singleDeviceJSON, false);
                deviceArray[i]=singleDevice;
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return deviceArray;
```

```
    }

}
```

9.1.2 Klase DeviceControl

```
package com.kkietis.kristapskietis.homeautomation.DevicePackage;

import android.app.Dialog;
import android.app.ListFragment;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Toast;

import com.kkietis.kristapskietis.homeautomation.Core.BackgroundTasks;
import com.kkietis.kristapskietis.homeautomation.Core.CallBack;
import com.kkietis.kristapskietis.homeautomation.R;
import com.kkietis.kristapskietis.homeautomation.Core.ValueStore;


/**
 * Veidoja Kristaps Kietis 3/1/2016 datumā.
 * Galvenā ierīču skata klase
 */
public class DeviceControl extends ListFragment{
    ValueStore vs;
    /**
     * @param inflater
     * @param container
     * @param savedInstanceState
     * @return
     */
    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        return inflater.inflate(R.layout.device_control, container,false);

    }

    /**
     * @param view
     * @param savedInstanceState
     * Tiek inicializēta ierīču saraksta aizpilde, pie skata izveidošanās
     */
    @Override
    public void onViewCreated(final View view, Bundle savedInstanceState){
        super.onViewCreated(view, savedInstanceState);
        String method = "getDeviceList"; //Metodes nosaukums
        BackgroundTasks backgroundTask = new BackgroundTasks(new CallBack() {
            @Override
            public void done(String result) throws Exception {
                if (result != null && !result.isEmpty()) {
                    createDeviceList(view, Device.createDeviceArray(result));//Tiek
aizpildīts saraksts ar ierīcēm
```

```
                }else{
                    Toast.makeText(view.getContext(),"Cannot connect!",
Toast.LENGTH_SHORT).show(); // Attēlots kļūdas paziņojums
                }

            }
        },view.getContext());
        backgroundTask.execute(method); // Tiek izpildīts asinhronais uzdevums

    }

    /**
     * @param view
     * @param deviceList
     * Tiek izveidots saraksts
     */
    private void createDeviceList(View view, Device[] deviceList) throws  Exception{
        ArrayAdapter myDeviceAdapter = new
DeviceListAdapter(view.getContext(),deviceList); // Izveidots ierīču saraksta
adapteris
        setListAdapter(myDeviceAdapter); //Adapteris tiek uzstādīts sarakstā
        getListView().setOnItemClickListener(new AdapterView.OnItemClickListener()
{//Izveidots klikšķa klausītājs, kurš atvērs ierīces info logu
            /**
             *
             * @param parent
             * @param view
             * @param position
             * @param id
             * Uznirstošā loga inicializācija konkrētai ierīcei
             */
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
                Device singleDevice = (Device) parent.getItemAtPosition(position);
                Dialog userDialog = new DeviceInfoDialog(view.getContext(),
singleDevice);
                userDialog.show();//Tiek attēlots ierīces uznirstošais logs
            }
        });
    }

}
```

9.1.3 Klase DeviceListAdapter

```
package com.kkietis.kristapskietis.homeautomation.DevicePackage;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.CompoundButton;
import android.widget.Switch;
import android.widget.TextView;
```

```
import com.kkietis.kristapskietis.homeautomation.R;

/**
 * Veidoja Kristaps Kietis 4/3/2016 datumā.
 * Ierīču saraksta adapteris
 */
class DeviceListAdapter extends ArrayAdapter<Device> {
    private final Context ctx;

    /**
     * @param context
     * @param resource
     */
    public DeviceListAdapter(Context context, Device[] resource) {
        super(context, R.layout.list_device, resource);
        ctx=context.getApplicationContext();
    }

    /**
     * @param position
     * @param convertView
     * @param parent
     * @return
     * Metode uzstāda funkcionalitāti konkrētam saraksta elementam
     */
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflater = LayoutInflater.from(parent.getContext());
            convertView = inflater.inflate(R.layout.list_device, parent, false); //
Tiek uzstādīts ierīces elementa skats
            final Device singleItem = getItem(position);//iegūst konkrēto saraksta
ierīces elementu
            TextView text = (TextView)convertView.findViewById(R.id.itemNameTW);
            final Switch statusBtn =
(Switch)convertView.findViewById(R.id.sBtnDeviceState);

statusBtn.setChecked(singleItem.deviceStatus.equals(Device.STATUS_ON));//Uzstāda šī
brīža ierīces stāvokli slēdža attēlošanai
            text.setText(singleItem.deviceName);//uzstāda ierīces nosaukumu
            statusBtn.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {//Pogai tiek uzstādīta klikšķa
funkcionalitāte

                @Override
                public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                    singleItem.chageDeviceState(statusBtn,isChecked,ctx);//Mainot
slēdža stāvokli, tiek inicializēta ierīces statusa maiņa
                }
            });
        return convertView;
    }
}
```

9.1.4 Klase DeviceInfoDialog

```
package com.kkietis.kristapskietis.homeautomation.DevicePackage;

import android.app.Dialog;
import android.content.Context;
```

```
import android.graphics.Paint;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import com.kkietis.kristapskietis.homeautomation.Core.BackgroundTasks;
import com.kkietis.kristapskietis.homeautomation.Core.CallBack;
import com.kkietis.kristapskietis.homeautomation.HistoryPackage.History;
import com.kkietis.kristapskietis.homeautomation.HistoryPackage.HistoryListAdapter;
import com.kkietis.kristapskietis.homeautomation.R;
import com.kkietis.kristapskietis.homeautomation.UserPackage.User;


import java.util.ArrayList;
import java.util.Arrays;

/**
 * Veidoja Kristaps Kietis 4/19/2016 datumā.
 * Klase veic darbības ar uznirstošo logu, kurš tiek attēlots brīdī, kad nospiests
virsū konkrētai ierīcei
 */
class DeviceInfoDialog extends Dialog {
    private String deviceID;
    Boolean isOwner;
    TextView deviceNameTW;
    String deviceName;

    /**
     * @param context
     * @param device
     */
    public DeviceInfoDialog(Context context, Device device) {
        super(context);
        deviceID = device.deviceID;
        isOwner = device.isOwner;
        deviceName = device.deviceName;
    }

    /**
     * @param savedInstanceState
     * Metode tiek izsaukta pie skata izveides
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.my_device_dialog);
        //Inicializējam sarakstus, kuros glabāsies ierīces piesaistītie lietotāji un
ierīces vēsture
        final ListView assignedUsersLW = (ListView)findViewById(R.id.assignedUsersLW);
        final ListView deviceHistoryLW = (ListView)findViewById(R.id.deviceHistoryLW);
        //Tiek uzstādīts ierīces nosaukums
        deviceNameTW = (TextView)findViewById(R.id.dialogDeviceNameTW);
        deviceNameTW.setPaintFlags(deviceNameTW.getPaintFlags() |
Paint.UNDERLINE_TEXT_FLAG); // Teksts tiek uzvilkta svītra apakšējā pozīcijā
        deviceNameTW.setText(deviceName);

        //Saraksti tiek aizpildīti asinhroni
        String method = "getDeviceAssignedUsers";
        BackgroundTasks backgroundTask = new BackgroundTasks(new CallBack() {
            @Override
```

```
            public void done(String result) {
                if (!result.isEmpty()) {
                    ArrayAdapter myUserAdapter = new
DeviceSubscriptionListAdapter(getContext(),new
ArrayList<>(Arrays.asList(User.getUserAttributes(result))),deviceID, isOwner);
                    assignedUsersLW.setAdapter(myUserAdapter);
                }else{
                    Toast.makeText(getContext(), "Cannot connect!",
Toast.LENGTH_SHORT).show();
                }

            }
        }, getContext());
        backgroundTask.execute(method, deviceID);
        method = "getDeviceHistory";
        BackgroundTasks backgroundTask2 = new BackgroundTasks(new CallBack() {
            @Override
            public void done(String result) {
                if (result != null && !result.isEmpty()) {
                    ArrayAdapter myUserAdapter = new HistoryListAdapter(getContext(),
History.getDeviceHistory(result));
                    deviceHistoryLW.setAdapter(myUserAdapter);
                }else{
                    Toast.makeText(getContext(), "Cannot connect!",
Toast.LENGTH_SHORT).show();
                }

            }
        }, getContext());
        backgroundTask2.execute(method, deviceID);
    }


}
```

9.1.5 Klase DeviceSubscriptionListAdapter

```
package com.kkietis.kristapskietis.homeautomation.DevicePackage;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;

import com.kkietis.kristapskietis.homeautomation.Core.BackgroundTasks;
import com.kkietis.kristapskietis.homeautomation.Core.CallBack;
import com.kkietis.kristapskietis.homeautomation.R;
import com.kkietis.kristapskietis.homeautomation.UserPackage.User;
import com.kkietis.kristapskietis.homeautomation.Core.ValueStore;

import java.util.ArrayList;

/**
```

```
 * Veidoja Kristaps Kietis 4/19/2016 datumā.
 */
class DeviceSubscriptionListAdapter extends ArrayAdapter {
    private final String deviceID;//Uzspiestās ierīces id
    private final ArrayList userList;//Piesaistīto lietotāju saraksts
    private boolean isowner;
    ValueStore vs;

    /**
     * @param context
     * @param userList
     * @param deviceID
     * @param isOwner
     */
    public DeviceSubscriptionListAdapter(Context context, ArrayList userList,String
deviceID, boolean isOwner) {
        super(context, R.layout.list_assigned_users, userList);
        this.deviceID = deviceID;
        this.userList = userList;
        this.isowner = isOwner;
        this.vs = (ValueStore)context.getApplicationContext();
    }

    /**
     * @param position
     * @param convertView
     * @param parent
     * @return
     * Tiek uzstādīta funkcionalitāte konkrētam ierīces elementam
     */
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflater = LayoutInflater.from(parent.getContext());
            convertView = inflater.inflate(R.layout.list_assigned_users, parent,
false);
            final User singleItem = (User)getItem(position);//Iegūst konkrētā
lietotāja elementu
            final ImageButton unsubscribeBTN =
(ImageButton)convertView.findViewById(R.id.unsignBTN);//Lietotāja atsaistes poga, lai
noņemtu lietotāju no īpašniekam piederošas ierīces
            TextView fullName =
(TextView)convertView.findViewById(R.id.assignedNameTW);
            fullName.setText(singleItem.fullName);
            if(!isowner && !vs.userID.equals(singleItem.ID)){//Ja lietotājam nepieder
ierīce, poga tiek paslēpta
                unsubscribeBTN.setVisibility(View.GONE);
            }
            unsubscribeBTN.setOnClickListener(new View.OnClickListener() {//Atsaistes
pogas klikšķa klausītājs
                @Override
                public void onClick(View v) {
                    final String method = "removeUserFromDevice";
                    BackgroundTasks backgroundTask = new BackgroundTasks(new
CallBack() {
                        @Override
                        public void done(String result) {
                            if (result != null && !result.isEmpty()) {
                                Toast.makeText(getContext(), result,
Toast.LENGTH_SHORT).show();
                                unsubscribeBTN.setClickable(false); // Poga tiek
atietatīta, lai nevarētu to nospiest atkārtoti
                                userList.remove(singleItem);
```

```
                            }else{
                                Toast.makeText(getContext(), "Cannot connect!",
Toast.LENGTH_SHORT).show();
                            }

                        }
                    }, getContext());
                    backgroundTask.execute(method, singleItem.ID,deviceID);//Vadoties
pēc lietotāja un ierīces ID lietotājs tiek atsaistīts no ierīces
                }
            });
        return convertView;
    }
}
```

9.1.6 Klase RegisterDeviceList

```
package com.kkietis.kristapskietis.homeautomation.DevicePackage;

import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.widget.TextView;
import android.widget.Toast;

import com.kkietis.kristapskietis.homeautomation.R;

import java.util.ArrayList;
import java.util.Set;


/**
 * Veidoja Kristaps Kietis 5/15/2016 datumā.
 * Klase paredzēta bluetooth ierīču attēlosanai
 */
public class RegisterDeviceList extends Activity
{
    Button btnPaired;
    ListView devicelist;
    private BluetoothAdapter myBluetooth = null;
    private Set<BluetoothDevice> pairedDevices;
    public static String EXTRA_ADDRESS = "";
    ArrayList<String> list;
    BroadcastReceiver mReceiver;
    /**
     * @param savedInstanceState
     * Metode tiek izsaukta pie skata izveides, un uzstāda elementu funkcionalitāti
     */
    @Override
    protected void onCreate(Bundle savedInstanceState)
```

```
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.list_activity_device);
        btnPaired = (Button)findViewById(R.id.BTshowBTN);
        devicelist = (ListView)findViewById(R.id.listView);
        myBluetooth = BluetoothAdapter.getDefaultAdapter();
        list = new ArrayList<>();
        if(myBluetooth == null) {
            Toast.makeText(getApplicationContext(), "Bluetooth Device Not Available",
Toast.LENGTH_LONG).show();
            finish();
        }
        else if(!myBluetooth.isEnabled()) {
                Intent turnBTon = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);//pieprasa lietotājam ieslēgt Bluetooth
ierīci
                startActivityForResult(turnBTon,1);
        }
        pairedDevicesList();
        btnPaired.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                pairedDevicesList(); //Tiek atjaunots ierīču saraksts
            }
        });

    }

    /**
     *Metode iegūst un uzstāda sapāroto ierīču sarakstu
     */
    private void pairedDevicesList() {
        pairedDevices = myBluetooth.getBondedDevices(); // Iegūst ier;ices, ar kurām
ir izveidots savienojums
        list = new ArrayList<>();

        if (pairedDevices.size()>0) {
            for(BluetoothDevice bt : pairedDevices) {
                list.add(bt.getName() + "\n" + bt.getAddress()); //Iegūst ierīces
nosaukumu un adresi
            }
        }
        else {
            Toast.makeText(getApplicationContext(), "No Paired Bluetooth Devices
Found.", Toast.LENGTH_LONG).show();
        }

        final ArrayAdapter<String> adapter = new
ArrayAdapter<>(this,android.R.layout.simple_list_item_1, list);
        devicelist.setAdapter(adapter); // Ier;ices tiek uzstādītas sarakstā
        devicelist.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            /**
             * Nospižot uz ierīces elementa tiek uzsāks savienojums ar ierīci un
uzstādīta RegisterDevice aktivitāte
             * @param av
             * @param v
             * @param position
             * @param id
             */
            public void onItemClick(AdapterView<?> av, View v, int position, long id)
{
                String info = ((TextView) v).getText().toString();
```

```
                String address = info.substring(info.length() - 17);
                Intent i = new Intent(RegisterDeviceList.this, RegisterDevice.class);
                i.putExtra(EXTRA_ADDRESS, address);
                startActivity(i); // Lietotājs tiek novirzīts uz RegisterDevice klasi
                finish();
            }
        });

    }

}
```

9.2 Servera kods

Kvalifikācijas darbam pievienots lietotāja sistēmas piekļuves kods “Login.php”

```
<?php
namespace App\Model;

use App\core\Database;

/**
 * Klase pārbauda lietotāja piekļuves tiesības, un ļauj pieslēgries sistēmai
 */
class Authorization{

    private $username;
    private $password;
    private $usernameLen = 20;
    function __construct(){
        $this->setVariables();
    }

    /**
     * Tiek uzstādīti modeļa mainīgie drošā formātā
     */
    function setVariables(){
       $this->password = md5($_POST['password']);//Lietotāja ievadītā parele
tiek kārveidota md5 kodējumā, lai varētu salīdzināt ar datubāzē glabāto
vētību
       $this->username = htmlspecialchars($_POST['username']);
    }

    /**
     *
     * @param type $username
     * @param type $password
     * @return boolean
     * Metode pārbauda vai lietotāja nodotie dati ir pareizi
     */
    function validateLogin($username, $password){
       if($this -> usernameLen < strlen($this -> username) || strlen($this -
> username) == 0 ){
           return false;
       }
       $query = "Select password from user where username = lower('".$this -
> username."') and password = '".$this->password."'";
       $passwordSQL = mysqli_query(Database::returnConnection(),$query);
       $row = $passwordSQL->fetch_assoc();
       if($password == $row['password']){//Tiek pārbaudīts vai lietotāja
ievadītie dati ir pareizi
           return true;
       }else{
           return false;
       }
    }

    /**
```

```
     * @return boolean
     * Metode izpilda piekļuves funkcionalitāti
     */
    function doLogin(){
        if($this -> validateLogin($this -> username, $this -> password)){
            $this->setSession($this -> username);
            return true;
        }
        return false;
    }

    /**
     * @return boolean
     * Metode pārbauda vai lietotājs ir pieslēdzies sistēmai
     */
   static function checkUserLoggedIn(){

        if(isset($_SESSION['username'])){
            return true;
        }
        else{
            return false;
        }

    }
    /**
     * Metode, kas pārtrauc lietotāja sesiju ar serveri
     */
    static function logout(){
        session_destroy();
        session_start();
    }
    /**
     *
     * @param type $username
     * @return boolean
     * Metode jaunas sesijas uzstādīšanai
     */
    static function setSession($username){
        if(!isset($_SESSION)){
            session_start();
        }
            $_SESSION['username'] = $username;
            return true;
    }

}
```

9.3 Arduino kods

Kvalifikācijas darbam pievienots arduino funkcionalitātes kods

```
#include <string.h>
#include <SoftwareSerial.h>
//Tiek definēti mainīgie, kuri būs nemainīgi visas ierīces darbības laikā
const char Domain[20] = "kietis.asuscomm.com"; //Servera ip adrese
const char Port[3] = "80"; //Porta numurs
const char Path[38] = "/HomeAutomation/Public/DeviceControl/";//Saite, kas paredzēta
ierīcēm
const char deviceState[12] = "devicestate";//metožu nosaukumi priekš web servera
const char activateAlarm[14] = "activatealarm";
const char registerDevice[15] = "registerdevice";
const char deviceType[14] = "getdevicetype";
const int hashLen = 32;
const char deviceID[9] = "6c300467";// Unikāls ierīces identifikators, tiek iekodēts
kodā, jo Arduino nav iespējams saglabāt datus
const char statusON[hashLen+1] = "90651ebea9a35ec4e018c8157492e17c"; //MD5 kodējuma
statusi
const char statusOFF[hashLen+1] = "88559a0cfd8250c9d65970cc145c92d4";
const char statusSwitch[hashLen+1] = "b36eb6a54154f7301f004e1e61c87ce8";
const char statusAlarm[hashLen+1] = "6486b4e98228b83d4b13d54febe5f170";
const char beginningOfResponse[hashLen+1] = "c8c54668d23390956bbeba4ba300d4b2";
const int trashHold = 7; // I/O pin numuri
const int setupInput = 2;
const int devicePowerPin = 12;
const int trigerPin = 7;
const int echoPin = 8;
const int responseLen = 103;
const char responseEndChar = '`'; // bluetooth datu beigu simbols
const char responseSplitChar = '|'; // bluetooth datu sadalītājsimbols
//Tiek definēti mainīgie, kuri ir nepieciešami programmatūras darbībā
boolean isOn = false; //Ierīces stāvoklis
SoftwareSerial blueToothSerial(5, 6); // Bluetooth I/O datu nesējs
bool isSwitch; // Ierīces tipa mainīgais
int normalDistance; // Signalizācijas normālstāvoklis


/**
 * Metode tiek izsaukta ieslēdzot ierīci
 * Tiek uzstādīti mainīgie, kas nepieciešami, lai uzsāktu programmatūras darbību
 */
void setup() {
  pinMode(devicePowerPin, OUTPUT); //Tiek pateikts, ka devicePowerPin būs strāvu
padodošais I/O pin, kurš būs nepieciešams tipa "swtch" ieslēgšanai un izslēgšanai
  Serial.begin(9600); // Serial komunikācija ar wifi
  blueToothSerial.begin(9600); // Serial komunikācija ar bluetooth
  while(!checkWiFiOn()); // Tiek pārbaudīts, vai ir eksistējoša wifi ierīce, ja nav,
tad nav vērts turpināt programmatūras darbību. To par pieslēgt kāt arī brīdī, kad
ierīce ir nonākusi šajā stāvoklī
  digitalWrite(setupInput, INPUT); // Pogas I/O pin, kas nosaka, ka ieslēdzoties
ierīcei to ir jāreģistrē
  if(digitalRead(setupInput) == HIGH){
   while(setupDevice());
  }
  while(!determineDeviceState(deviceType)); // Tiek noteikts ierīces tips, ar kādu to
ir reģistrējis lietotājs
  pinMode(trigerPin, OUTPUT);
   pinMode(echoPin, INPUT);
  if(!isSwitch){//Nepieciešams uzstādīt signalizāciju, ja tas nav slēdzis
    setupAlarm();
  }
}

/**
```

```
 * Mūžīgs cikls, kurā ir realizēta arduino funkcionalitāte
 */
void loop() {

  determineDeviceState(deviceState); //Tiek noteikts ierīces stāvoklis
  if(!isSwitch && isOn){ // Ja ierīce ir signalizācijas tipa, un lietotājs to ir
uzstādījis aktīvu tiek pārbaudīts vai tā nav nostrādājusi
    checkAlarmTrigger();
  }
  if(digitalRead(setupInput) == HIGH){
    while(setupDevice());
    while(!determineDeviceState(deviceType));
    if(!isSwitch){//Nepieciešams uzstādīt signalizāciju, ja tas nav slēdzis
    setupAlarm();
  }
  }
}

/**
 * Pārbauda vai ir eksistējoša wifi ierīce
 */
boolean checkWiFiOn(){
  Serial.println("AT+RST");
  delay(1000);
  Serial.println("AT\r\n");
  if(Serial.find("OK\r\n")){
    Serial.println("AT+CIPMUX=0");
    return true;
  }
  else{
    return false;
  }
}

/**
 * funkcija, lai izveidotu sasaisti ar lokālo bezvadu tīklu
 */
boolean connectToWiFi(String SSID, String PASSWORD){
  Serial.println("AT+CWQAP");
  delay(100);
  Serial.println("AT+CWMODE=1");
  delay(50);
  String sendData;
  sendData= "AT+CWJAP=\"";
  sendData+=SSID;
  sendData+="\",\"";
  sendData+=PASSWORD;
  sendData+="\"";
  Serial.println(sendData);
  sendData="";
  if(Serial.find("OK")){
    return true;
  }else{
    return false;
  }
}

/**
 * Funkcija, kas nolasa wifi adaptera atbildi no servera
 */
String readResponse(){
    char res[responseLen];
```

```
    Serial.readBytes(res,responseLen);
    //Serial.println(res);
    return determineHashResponse(res,beginningOfResponse);
}

/**
 * Funkcija, kas veic servera izsaukumu,
 * String page - servera metodes nosaukums
 * Boolean isRespone - Vai arduino ir jāapstrādā atbildi no servera
 * String extraPostData - papildus dati, kas tiek pievienoti papildus deviceID
 */

String makeCallToServer(String page, boolean isResponse, String extraPostData){
    serialFlush();
    int length;
    String response;
    String sendData;
    sendData="";
    sendData = "AT+CIPSTART=\"TCP\",\"";
    sendData += Domain;
    sendData += "\",";
    sendData += "80";
    Serial.println(sendData);

    delay(100);
    sendData = "GET ";
    sendData+= Path;
    sendData+= page;
    sendData+= "?deviceid=";
    sendData+= deviceID;
    sendData+= extraPostData;
    sendData += "\r\n\r\n";
    length = sendData.length();
    Serial.print("AT+CIPSEND=");
     Serial.println(sendData.length());
     if(!Serial.find(">")){
         Serial.println("AT+CIPCLOSE");
         delay(100);
         return "ERROR";
       }
    Serial.print(sendData);
    sendData = "";

    if(isResponse){
      response = readResponse();
      return response;
    }else{
      readResponse();
      return "";
    }

}


/**
 * Funkcija, kas apstrādā servera atgrieztos datus
 */

boolean determineDeviceState(String page){
    String responseFromServer = makeCallToServer(page, true, "");
    boolean isAction = false;
    if(responseFromServer.equals(statusON)){
```

```
      if(isSwitch){
        digitalWrite(devicePowerPin, HIGH);
      }else if (!isSwitch && !isOn){
        setupAlarm();
      }
      isOn = true;
      isAction = true;
    }else if(responseFromServer.equals(statusOFF)){
      if(isSwitch){
        digitalWrite(devicePowerPin, LOW);
      }
      isOn = false;
      isAction = true;
    }else if(responseFromServer.equals(statusSwitch)){
      isSwitch = true;
      Serial.println("Device is: Switch");
      isAction = true;
    }else if(responseFromServer.equals(statusAlarm)){
      isSwitch = false;
      Serial.println("Device is: Alarm");
      isAction = true;
      digitalWrite(devicePowerPin, LOW);
    }
  return isAction;
}

/**
 * Funkcija, kas no atgrieztajiem datiem izrēķina klasifikatoru
 */
String determineHashResponse(char* page, String startSequence){
int i = 0;
int j = 0;
char resp[hashLen+1];
while(i<responseLen && page[i] != '\0'){

    if(page[i] == startSequence[j]){
        j++;
    }
    else{
        j=0;
    }
    if(j == hashLen){
      i++;
        for(int k = 0; k<hashLen;k++){
            resp[k] = page[i];
            i++;
        }
        resp[hashLen] = '\0';
        break;
    }

i++;
}

return resp;
}

/**
 * Funkcija, kas secina, vai signalizācija ir nostrādājusi
 */
boolean checkAlarmTrigger(){
  int distance = returnDistance();
```

```
  if(normalDistance - distance > trashHold || distance - normalDistance > trashHold){
    makeCallToServer(activateAlarm, false, "");
    delay(50);
  }

}

/**
 * Funkcija, kas aprēģina tekošo signalizācijas stāvokli
 */
int returnDistance(){
  long duration, distance;
  do{
    duration = distance = 0;
    digitalWrite(trigerPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigerPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigerPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    distance = (duration/2) * 0.0344;
  }while(distance >= 200 || distance <= 0);
    return distance;
}

/**
 * Funkcija, kas uzstāda signalizācijas normālpozīciju
 */
void setupAlarm(){
  int times = 5;
  int distanceSum = 0;
  for(int i = 0; i<times;i++){
    distanceSum = distanceSum + returnDistance();
    Serial.println("Setting up");
    delay(1000);
  }
  normalDistance = distanceSum / times;
  Serial.print("Normal state is: ");
  Serial.println(normalDistance);
}

/**
 * Funkcija, kas attīra Serial krātuvi
 */
void serialFlush(){
  while(Serial.available() > 0) {
    char t = Serial.read();
  }
}

/**
 * Funkcija, kas uzstāda caur Bluetooth saņemtos datus ierīces parametros, un
veiksmīga rezultāta gadījumā reģistrē ierīci
 */
boolean setupDevice(){

  boolean isValidResponse = false;
  String response = "";
  String extraPostData = "";
  while(!isValidResponse){
    while(blueToothSerial.available()){
      char singleChar= blueToothSerial.read();
```

```
      response = response + singleChar;
      if(singleChar == responseEndChar){
        isValidResponse = true;
      }
     }
  }
  boolean isError = false;
  if(isValidResponse){
  int i = 0;
  String SSID = "";
  String PASSWORD = "";
  String userID = "";
  String deviceType;
  String deviceName = "";
  while(response[i] != responseEndChar){
    while(response[i] != responseSplitChar){
      deviceName += response[i];
      i++;
    }
    i++;
    while(response[i] != responseSplitChar){

      SSID += response[i];
      i++;
    }
    i++;
    while(response[i] != responseSplitChar){
     PASSWORD += response[i];
     i++;
    }
    i++;
    while(response[i] != responseSplitChar){
      deviceType += response[i];
      i++;
    }
    i++;
    while(response[i] != responseEndChar){
        userID += response[i];
        i++;
    }
  }
    int k = 0;
    int j = 10; // Tiks 10 reizes mēģināts veiksmīgi veikt darbību
    while(k<j){
      isError = true;
      //Serial.println("Trying to connect to wifi");
      if(connectToWiFi(SSID, PASSWORD)){
        isError = false;
        break;
      }
      k++;
    }
    if(isError){
      blueToothSerial.print("WIFIERROR");
      return isError;
    }
    extraPostData = "&userid="+userID + "&devicename=" + deviceName + "&devicetype=" +
deviceType;
    k=0;
    while(k<j){
      isError = true;
      if(makeCallToServer(registerDevice, false,extraPostData) != "ERROR"){
```

```
        isError = false;
        break;
      }
      k++;
    }
    if(isError){
      blueToothSerial.print("NETWORKERROR");
      return isError;
    }else{
      blueToothSerial.print("SUCCESS");
    }

  }
  return isError;
}
```

Kvalifikācijas darbs „ ***Mājas automatizācijas sistēma***” 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: ***Kristaps Kietis*** _____ **30.05.2016**.

Rekomendēju darbu aizstāvēšanai

Darba vadītājs: ***B.dat Mareks Kalnačs*** _____ **30.05.2016.**

Recenzents: ***M.dat. Raimonds Simanovskis***

Darbs iesniegts 30.05.2016.

Kvalifikācijas darbu pārbaudījumu komisijas sekretārs: Darja Solodovņikova_____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē

___.06.2016. prot. Nr. _____

Komisijas sekretāre: _____