

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Markus Visnapuu 206354IACB
Markus Veersoo 222399EARB
Rannar Randmäe 206021IACB
Siim Tišler 213891IACB

TARK LABORIKAPP

Arvutite ja süsteemide projekt

Juhendaja: Raivo Sell
PhD

Tallinn 2023

Autorideklaratsioon

Kinnitame, et oleme koostanud antud aruande iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autorid: Siim Tišler, Markus Visnapuu, Markus Veersoo, Rannar Randmäe

06.11.2023

Annotatsioon

Antud projekti eesmärgiks oli lisada olemasolevale laborikapile elektrooniline lukustuslahendus, mida saab avada NFC (*Near Field Communication*) kaardiga. Sellega kaasnes ülesanne lisada kapile ka mugav süsteem õiguste haldamiseks ning kapi avamise andmete salvestamine logis. Mugavaks kasutajate halduseks ning logile ligipääsemiseks loodi laborikapile veebiliides.

Projekti sooritamiseks olid tarkvara poolelt peamiseks töövahenditeks C ja C++ programmeerimiskeel, VSCode tekstiredaktor, Vue.js raamistik, Firebase andmebaas ning Github veebimajutus versioonihalduseks. Riistvara poolelt kasutati lukustussüsteemi jaoks kahte ESP8266 mikrokontrollerit, puuteekraani, solenoidi, releed, takisteid, sumistit, NFC lugerit ning ühte LED'i. Detailide korpuste modelleerimiseks kasutati SolidWorks'i ning loomiseks 3D-printerit. Lõpptulemusena valmis kahest suuremast osast koosnev süsteem: NFC kaardiga avatav laborikapp ning veebiliides sellele kapile.

Käesolev aruanne toob välja projekti eesmärgid ja nõuded ning tutvustab loodud süsteemi ja selle osasid lähemalt. Lisaks analüüsitakse aruandes projekti töö kulgu, tuues välja esile kerkinud probleemid ning võimalikud edasiarendused. Aruanne on kirjutatud eesti keeles ning sisaldab teksti 63 leheküljel, 11 peatükki, 10 joonist.

Lühendite ja mõistete sõnastik

<i>API</i>	<i>Application Programming Interface</i> , rakendusliides
<i>Backend</i>	Tagaosasüsteem, rakenduse osa, mida kasutaja ei näe; serveripoolsed protsessid, mis töötlevad ja salvestavad ning tihti vahendavad <i>frontend</i> -iga andmeid
<i>Baud</i>	Edastava sümbolikiiruse ühik, sümbolit sekundis
<i>COM</i>	<i>Communication port</i> , järjestikport
<i>CPU</i>	<i>Central Processing Unit</i> , põhiprotsessor
<i>CSS</i>	<i>Cascading Style Sheets</i> , kujunduskeel
<i>DOM</i>	<i>Document Object Model</i> , dokumendi objektimudel on platvormist ja keelest sõltumatu HTML-dokumentidega suhtlemise liides.
<i>EEPROM</i>	<i>Electrically Erasable Programmable Read-Only Memory</i> , elektriliselt kustutatav programmeeritav püsimälu
<i>Frontend</i>	Esiosasüsteem, kasutajaliidese osa, mida kasutaja näeb ning millega omab interaktsioonivõimet
<i>GPIO</i>	<i>General Purpose Input Output</i> , üldkasutatav sisend/väljund väljaviik
<i>HTML</i>	<i>Hypertext Markup Language</i> , tekstipõhine markeerimiskeel, mis defineerib veebilehe struktuuri
<i>IDE</i>	<i>Integrated Development Environment</i> , integreeritud programmeerimiskeskond
<i>IRQ</i>	<i>Interrupt Request</i> , katkestusnõue, kus küsitakse protsessori kohest ressursi, et mingi ülesanne täita
<i>IoT</i>	<i>Internet of Things</i> , asjade internet ehk nutistu
<i>JSON</i>	<i>JavaScript Object Notation</i> , standardne failiformaat, mis koosneb inimloetavast tekstist ja on keelest sõltumatu

<i>LED</i>	<i>Light Emitting Diode</i> , valgust kiirgav diood
<i>Linker</i>	Lihtne arvutiprogramm, mis võtab ühe või rohkem objektifaili ja kombineerib need käivitatavaks failiks või uueks objektiks
<i>Makefile</i>	Viis tarkvara kompileerimisprotseduuri automatiseerimiseks
<i>MQTT</i>	<i>MQ Telemetry Transport</i> , andmeedastusprotokoll
<i>NFC</i>	<i>Near Field Communication</i> , lähisväljaside
<i>NTP</i>	<i>Network Time Protocol</i> , võrgu aja protokoll; ajaserver, mis tagastab konkreetse asukoha täpse kellaaja
<i>Polling</i>	Seade kontrollib pidevalt teise seadme olekut, et teha kindlaks, kas see vajab tähelepanu
<i>SPI</i>	<i>Serial Peripheral Interface</i> , sünkroonse järjestiksuhtluse liidese standard
<i>SQL</i>	<i>Structured Query Language</i> , andmebaasi päringukeel
<i>Toolchain</i>	Arenduse töövoogu võimaldav ja toetav, üksteisega kokkusobivate tööriistade kogum
<i>TX ja RX</i>	<i>Transmit Receive</i> , edasta ja võta vastu
<i>UART</i>	<i>Universal Asynchronous Receiver-Transmitter</i> , järjestikliidese protokoll
<i>USB</i>	<i>Universal Serial Bus</i> , universaalne jadasiin

Sisukord

1 Sissejuhatus	9
2 Projekti eesmärk	10
2.1 Kasutusjuhtude diagramm	10
3 Kasutatav riistvara	14
3.1 Elektriskeem	15
4 Kasutatud tarkvara	17
4.1 IDE.....	17
4.2 PlatformIO	17
4.3 Firebase.....	18
4.4 Versioonihaldus	19
4.5 Teegid	19
4.6 SolidWorks	19
4.7 DevTools	20
4.8 Photoshop	20
5 Andmemudel	21
5.1 Cards	22
5.2 Lockers/locker1	22
5.3 Log.....	23
6 Kapi juhtloogika	24
7 Sardtarkvara.....	26
7.1 Koodi struktuur.....	26
7.1.1 NFC	27
7.1.2 Locker	28
7.1.3 Firebase ja võrguühendused	28
7.1.4 EEPROM	29
7.1.5 SerialDebug	30

7.1.6 Aja abifunktsioonid timeutils	32
8 Veebiliides	33
8.1 Tehnoloogia	33
8.2 Arhitektuur ja komponendid.....	34
8.2.1 Teenused	35
8.2.2 Vaated	37
8.2.3 Ruuter	43
8.2.4 Põhirakendus	43
8.2.5 Lisakomponendid	44
9 Laborikapp	46
9.1 Paigutus	47
9.2 Modifitseerimine ja monteerimine	47
9.3 Kasutatud vahendid	47
10 Projekti analüüs ja võimalikud edasiarendused	49
Kokkuvõte	50
Kasutatud allikad	51
Lisa 1. <i>Main</i> kontrolleri käivitamise plokk skeem.	53
Lisa 2. Pildid kapi komponentidest.	54
Lisa 3. Veebiliidese vealehe vaade.....	59
Lisa 4. Veebiliidese avalehe vaade.....	60
Lisa 5. Veebiliidese sisselogimisvaade.	61
Lisa 6. Veebiliidese kapi staatuse vaade.	62
Lisa 7. Veebiliidese kasutajate vaade.	63
Lisa 8. Veebiliidese logi vaade.....	64

Jooniste loetelu

Joonis 1. Laborikapi kasutusjuhtude diagramm.	11
Joonis 2. Lihtsustatud riistvara struktuurskeem.	14
Joonis 3. Laborikapi elektriskeem.	16
Joonis 4. Lugeri ja ekraani korpuse mudelid	19
Joonis 5. Ukselingi mudel	20
Joonis 6. JSON-andmemudel graafidena.....	21
Joonis 7. Lihtsustatud kapi juhtloogika	24
Joonis 8. UART sõnumite sisse-väljalülitamise makrod.....	31
Joonis 9. Veebiliidese komponentdiagramm.	35
Joonis 10. Valminud tark laborikapp.....	46

1 Sissejuhatus

Projekti teemavalik tulenes sellest, et laborikapile NFC lukustussüsteemi lisamine tundus meie jaoks aktuaalne, kuna järjest enam traditsioonilisi lukke vahetatakse välja tarkade lukuvabade süsteemidega, ning kasulik, kuna sellise ülesande realiseerimine hõlmab väga mitme erineva tööstusharu kombineerimist. Targa laborikapi loomisel on vaja kindlaks teha nõuded, modelleerida detaile, kirjutada tarkvara kapi haldamiseks, leida sobiv riistvara ning see programmeerida, teha vajalikud ühendused ja lisada riistvara kapile ning panna kogu süsteem testimise abil koos toimima.

Töö käigus arendati modelleerimistarkvara oskuseid kapile detailide 3D-printimiseks, õpiti looma kasutajasõbralikku ja turvalist veebiliidest *frontend* ja *backend* osadega Vue.js raamistikku kasutades, saadi tuttavaks Google'i poolt pakutava Firebase andmebaasiga, õpiti riistvara ja tarkvara omavahel liidestama ning saadi käsi valgeks targa süsteemi komplekteerimisel. Et projekti valmimist mõõta, pidi paika panema ka nõuded ja tingimused.

Järgnevad peatükid tutvustavad projekti osasid lähemalt, kirjeldavad disainimist, arendust, haldust ja komplekteerimist ning seatud eesmärkide saavutamist.

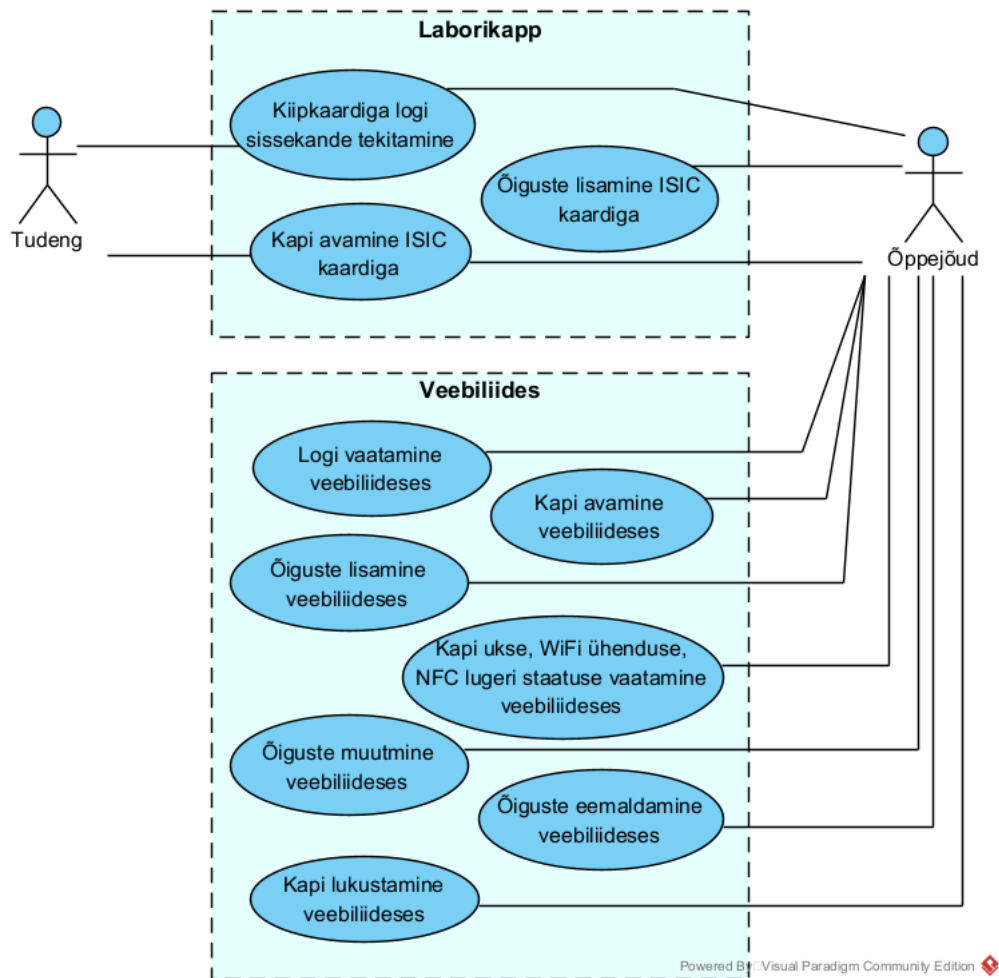
2 Projekti eesmärk

Käesoleva projekti peamine eesmärk oli lisada Tallinna Tehnikaülikooli U05B-208 robotika demokeskuses asuvale laborikapile nutikas lukustussüsteem. Laborikappi kasutatakse robotika kodulabori komplektide hoiustamiseks. Nutika lukustussüsteemi all on mõeldud seda, et tudeng või õppejõud saab kappi avada NFC kiipkaardiga (nt ISIC kaart). Lisaks pidime looma lihtsa süsteemi, mis võimaldaks õppejõul mugavalt hallata, millistel kaartidel on õigus ukse avamiseks, ning näha ajalugu, millal ja kes on kapi avanud.

Lisaeesmärkideks seati puutetundlik ekraan tagasiside jaoks ning kapi avamisvõimalus WiFi signaali või voolu puudumise korral. Projekti käigus oli vaja leida sobiv riistvara, see komplekteerida, disainida elegantsed ja kompaktsed korpused, mis riistvara mahutaks ja kaitseks, ning luua ka seda toetav tarkvara. Tuli koostada ka põhjalik dokumentatsioon, et kapp oleks edaspidi hooldatav ning et tekiks prototüüp, mille põhjal oleks võimalik tulevikus sarnaseid süsteeme ehitada.

2.1 Kasutusjuhtude diagramm

Käesolevas alapeatükis tutvustatakse võimalikke targa laborikapi kasutusjuhte. Et kasutusjuhte paremini illustreerida loodi kasutusjuhtude diagramm (vt Joonis 1).



Joonis 1. Laborikapi kasutusjuhtude diagramm.

Projekti eesmärgist selgub, et laborikapil on kaks peamist kasutajat: tudeng ja õppejõud. Sellega seoses leiti 10 erinevat kasutusjuhtu. Järgnevalt kirjeldatakse neid kasutusjuhte lähemalt.

Kiipkaardiga logi sissekande tekitamine – see kasutusjuht puudutab nii tudengit kui õppejõudu, kuna iga kord, kui kiipkaart NFC lugeri juures skanneeritakse, tekib sellest ajaloolisse logisse sissekanne.

Õiguste lisamine ISIC kaardiga – õppejõul on võimalus laborikapi juures NFC lugeriga skanneerida tudengi kaarti, mis järel saab tudeng õigused kapi avamiseks. Selleks peab õppejõud esmalt skanneerima oma kaarti ning ekraanil avanenud menüüs valima „Add cards“ valiku. Siis on õppejõul aega 10 sekundit, et tudengi kaart skanneerida, mis järel lisatakse tudengi kaardile õigused.

Kapi avamine ISIC kaardiga – nii õppejõul kui ka õiguseid omaval tudengil on võimalik oma ISIC kaardiga kapp avada. Tudeng peab selleks lihtsalt enda kaarti skanneerima ja kapp avatakse. Õppejõud skanneerib kaarti ja seejärel valib ekraanilt avanevast menüüst „Open door“, mille peale lukk avatakse.

Logi vaatamine veebiliideses – õppejõud saab veebiliidese kaudu vaadata kapi avamiste ajaloolist logi. Selleks peab ta kasutama arvutit või mobiili ning avama veebiliidese ja end sisse logima. Seejärel peab valima avalehelt „Vaata logi“, mille peale logi avaneb.

Kapi avamine veebiliideses – õppejõud saab veebiliidese kaudu kappi avada, sealjuures võib viibida, kus iganes, on vaja ainult internetiühendust. Selleks peab õppejõud avama veebiliidese ja end sisse logima. Seejärel peab valima avalehelt „Kapi seaded“, pärast mida avaneb võimalus vajutada nupule „Ava kapp“. Pärast nupule vajutamist peaks kapi lukk avanema, veebiliideses muutuma luku olek avatuks ning vajutatud nupule tekkima tekst „Lukusta kapp“.

Kapi lukustamine veebiliideses – õppejõud saab veebiliidese kaudu kapi lukustada, kui ta on varem veebiliidese kaudu kapi avanud. Kui avada kapp väljaspool veebiliidest, on lukustamine automaatne. Et kapp lukustada, peab pärast veebiliidese kaudu kapi avamist vajutama nupule „Lukusta kapp“. Pärast vajutust peaks kapp lukustuma, veebiliides kuvama luku olekut lukus ning vajutatud nupu eelneva teksti asemel ilmuma „Ava kapp“.

Õiguste lisamine veebiliideses – õppejõul on võimalus lisada veebiliidese kaudu kasutajale õigusi. Selleks on tal kaks võimalust. Esimene võimalus: tudeng on kohal laborikapi juures ja skanneerib oma ISIC kaardi NFC lugeriga. Õppejõud logib sisse veebiliidesesse ning avab logi, kust on näha, et keegi on hiljuti üritanud kappi avada. Antud logi sissekande paremas ääres on nupp „Lisa kasutajaks“, mille kaudu saab kaardile lisada õigused. Õppejõud vajutab nupule ja sisestab avanenud kasutaja lisamise vormis tudengi nime ning vajutab „Lisa“. Kui see on tehtud, teavitab ta tudengit ja tudeng saab kappi oma kaardiga avada. Teine võimalus: õppejõul on teada tudengi ISIC kaardi UUID. Õppejõud logib sisse veebiliidesesse ning vajutab avalehel nupule „Autoriseeritud kasutajad“. Talle avaneb nimekiri kasutajatest. Õppejõud vajutab lehe vasakul üleval nurgas asuvale nupule „Lisa kasutaja“. Õppejõud täidab avanenud kasutaja lisamise vormi sisestades sinna tudengi nime ning ISIC kaardi UUID ja vajutab „Lisa“. Pärast seda on tudengil võimalus oma ISIC kaardiga kappi avada.

Kapi ukse, WiFi ühenduse, NFC lugeri staatuse vaatamine veebiliideses – õppejõul on veebiliidese kaudu võimalik eemal viibides saada infot kapi staatuse kohta. Selleks peab õppejõud logima veebiliideses sisse ning avalehel vajutama nupule „Kapi seaded“. Avanenud lehelt saab õppejõud näha kapi ukse olekut, WiFi ühenduse olekut, NFC lugeri olekut. Hiirekursoriga WiFi ja NFC oleku kohal hõljudes ilmub õppejõule lisainfo, millal viimati NFC või WiFi ühendust kontrolliti.

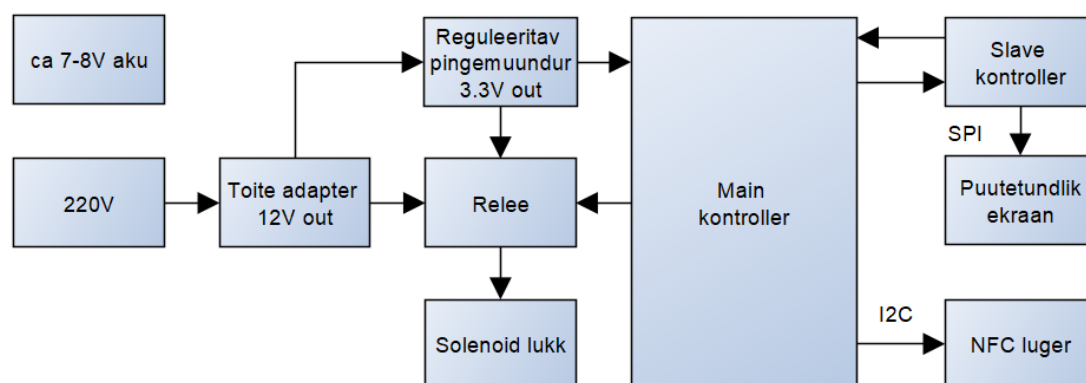
Õiguste muutmine veebiliideses – õppejõul on veebiliidese abil võimalik kaartidele antud õigusi muuta. Seda võib vaja minna, kui mingil tudengi õigustega isikul on vaja kellelegi õigused anda või kui kellegi õigused aeguvad ja on vaja neid pikendada. Selleks peab õppejõud logima end sisse veebiliideses ning seejärel avalehel vajutama nuppu „Autoriseeritud kasutajad“. Avanenud kasutajate nimekirjas peab õppejõud leidma kasutaja, kes soovis kas õiguste pikendamist või muutmist. Kasutaja leidnud, vajutab õppejõud kasutaja kõrval kõige parempoolses veerus asuvat nuppu „Muuda andmeid“. Avanenud vormis saab õppejõud valida kahe õiguste tüübi vahel: student või teacher. Lisaks saab valida uue kuupäeva, milleni õigused kehtivad, juhul kui neid on vaja pikendada. Kui õppejõud on vajalikud muudatused teinud, vajutab ta „Salvesta“ ning sellega on õigused muudetud.

Õiguste eemaldamine veebiliideses – õppejõul on veebiliidese abil võimalik õiguseid eemaldada. Seda võib ta teha juhul, kui on vaja piirata kellegi ligipääsu kapile, või kui kellegi õigused on aegunud. Selleks peab õppejõud logima end sisse veebiliideses ning avalehel vajutama nupule „Autoriseeritud kasutajad“. Avanenud lehel on tal õiguste eemaldamiseks mitu võimalust. Esimene võimalus: õppejõud vajutab vasakul üleval nurgas olevale kastikesele, kuhu tekib linnuke. See tähendab, et valitud on kõik kasutajad. Iga kasutaja juures on samasugune kastike, kuhu saab linnukese lisada või selle eemaldada. Kui valitud on vähemalt üks kasutaja, ilmub kõige ülemise kastikese kõrvale nupp „Eemalda valitud“. Õppejõud vajutab nupule ning ekraanile tekib kinnitusküsimus. Õppejõud vajutab „Jah“, mille peale valitud kasutajate õigused eemaldatakse ning nad ei saa enam kappi avada. Teine võimalus: õppejõud otsib nimekirjast kasutaja, kelle õigused soovib eemaldada. Kasutaja leidnud, vajutab ta samal real kõige parempoolses veerus olevat nuppu „Eemalda kasutaja“, mille peale ilmub ekraanile kinnitusteade. Õppejõud vajutab „Jah“ ning kasutaja õigused on sellega eemaldatud. Vajutades „Ei“, jäävad õigused eemaldamata.

3 Kasutatav riistvara

Toote lõppkasutusse kuuluvad erinevad riistvaralised komponendid. Peamise kapi juhtloogika eest vastutavad kaks ESP8266 mikrokontrollerit, millest üks on peamine kontroller ehk meistri rollis, edaspidi *main* kontroller ja teine selli rollis, edaspidi *slave* kontroller. *Main* ESP8266 on ühendatud PN532 NFC lugeriga. *Slave* kontroller on ühendatud puutetundliku ekraaniga. Samuti on *slave* kontroller ja *main* kontroller omavahel ühendatud kasutades UART (*Universal Asynchronous Receiver-Transmitter*) protokoll.

Toite jaoks on kasutuses vooluvõrku ühendatav 12-voldine toiteplokk, mis läbi reguleeritava pingemuunduri toidab mõlemat ESP8266 kontrollerit 3,3 voldiga. Toiteplokkist tuleb 12 volti ka otse releele. Releed kontrollib GPIO (*General Purpose Input Output*) väljaviiguga *main* kontroller. Releega on võimalik kontrollida solenoid-lukku (vt Joonis 2). Kui vool peaks majas ära minema, või toite adapter katki olema, siis saab toite adapteri välja vahetada aku vastu. Laboris olevad akud on piisavalt võimsad, et anda meie kõikidele seadmetele voolu.



Joonis 2. Lihtsustatud riistvara struktuurskeem.

3.1 Elektriskeem

Targa laborikapi elektriskeemil on detailselt välja toodud kõik riistvaralised komponendid ning nende vahelised ühendused (vt Joonis 3).

Kapi põhijuhtimise eest vastutab *main* ESP8266 D1 mini mikrokontroller, millega on ühendatud PN532 NFC luger, LED, sumisti, solenoid, juhtrelee, magnetandur ning *slave* kontroller. Suhtlus võrguga toimub ESP8266-le sisseehitatud WiFi mooduli abil.

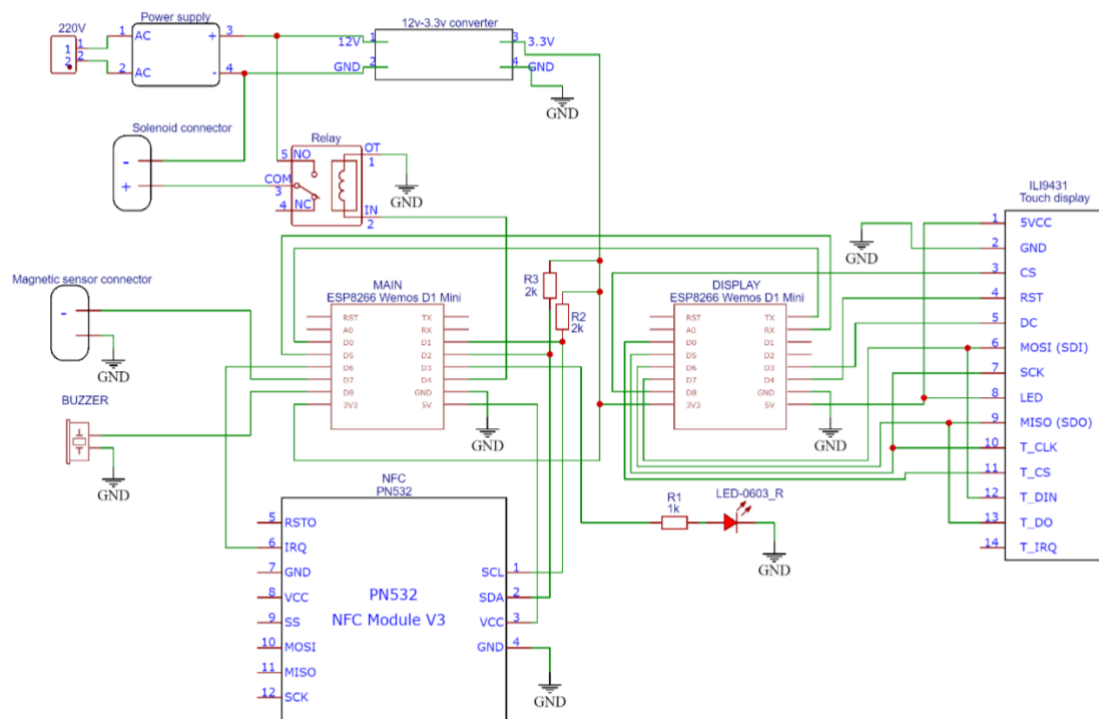
ILI9341 puutetundliku ekraanimooduli juhtimise eest vastutab *slave* ESP8266 D1 mini mikrokontroller. Lisaks sellele suhtleb see *main* kontrolleriga ning vastutab ekraanimooduli toitepinge eest. Suhtlus ekraaniga toimub SPI (*Serial Peripheral Interface*) protokollil alusel. Kasutusel on kaks alamseadet, millest üks on puutetundlikkuse ning teine ekraani andmete edastamiseks. Selle tõttu on ekraanimoodulil SPI sisendeid topelt.

PN532 NFC luger kasutab mikrokontrolleriga suhtlemiseks I2C protokollil ning saab oma toitepinge samuti kontrollerist. Lisaks on kasutusel IRQ (*Interrupt Request*) ühendus lugeja äratamiseks. I2C ühendustele on lisatud ka 2000 Ω tõmbetakistid, mis parandavad töökindlust.

Vooluvõrku ühendatav 12-voldine toiteplokk on kasutusel terve süsteemi energiatarbe rahuldamiseks. Kasutatav toiteplokk muundab 220 V vahelduvvoolu 12 V alalisvooluks ning selle maksimaalne väljundvool on 2 A. Lisaks 12 voldile, mis lülitab solenoidi, vajab ülejäänud süsteemi riistvara 3.3 V toitepinget. Selle saavutamiseks on kasutusel reguleeritav pingemuundur, millel reguleerisime paika väljundpinge ja väljundvoolu.

Laborikappi lukustava solenoidi lülitamiseks on kasutusel Wemos mini relee moodul. Releed juhitakse lihtsa madal/kõrge signaaliga. Valitud solenoidi tööpingeks on 12 V. Ukse staatuse saamiseks on kasutusel magnetandur, mis suletud ukse korral saadab mikrokontrollerile kõrge signaali.

Kasutusmugavuse tõstmiseks võeti riistvaraliselt kasutusele passiivne sumisti ja punane LED, mis annavad kasutajale tagasisidet süsteemi olekutest. LED'i voolu piiramiseks on kasutusel 1000 Ω kaitsetakisti.



Joonis 3. Laborikapi elektriskeem.

4 Kasutatud tarkvara

Järgnevas peatükis kirjeldatakse kogu kasutatud tarkvara. Sinna kuuluvad arenduskeskkonnad, teegid, sardtarkvara loomise raamistikud jne.

4.1 IDE

Riistvarale koodi kirjutamine toimus täielikult kasutades VSCode (*Visual Studio Code*) IDE-d (*Integrated Development Environment*), kuna see keskkond on varasemalt tuttav ja arendajad tunnevad end seal kõige mugavamalt. Otsust toetas ka see, et antud IDE-le on võimalik lisada erinevaid laiendeid.

Veebiliidese loomine toimus samuti VSCode abil, kuna see oli juba varasemalt installeeritud ning selgus, et sobib antud ülesandeks suurepäraselt. See toetas kõiki vajaminevaid laiendusi ning sobis hästi Vue.js raamistikuga arenduseks oma lihtsuse ning rohke abimaterjali tõttu.

4.2 PlatformIO

ESP8266 mikrokontrollerite jaoks kirjutatud koodi kompileerimiseks ning läbi USB virtuaalse COM (*communication port*) pordi koodi mikrokontrolleri mällu laadimiseks kasutatakse PlatformIO tarkvara. PlatformIO on VSCode keskkonda laiendusena installeeritav ning integreerub otse IDE-sse, ilma et peaks eraldiseisvat programmi kasutama. PlatformIO on avatud lähtekoodiga sardtarkvara arenduseks loodud rakendus, mis toetab arendamist paljude erinevate mikrokontrollerite jaoks. Näiteks Arduino, ESP, STM32 jt. Rakendus võimaldab jadaliidest kasutades ESP ja arvuti vahel andmeid saata ja lugeda, siluda programmi, lisada hõlpsalt väliseid teeke ja need *linker*'iga automaatselt kompileeruma panna. Suur eelis on see, et mõne versioonihalduskeskkonna kaudu saab kergelt projekti teiste arendajatega sellisel moel jagada, et teise arendaja käes kõik kohe

töötab, sest kogu *toolchain* on projektiga kaasas. Ei pea kirjutama eraldi *makefile*'e ega CMake faile [1].

4.3 Firebase

Firebase on arendajate jaoks loodud Google'i pilveteenus. Firebase kasutades on võimalik luua NoSQL reaalaja andmebaas. Andmeid hoiustatakse JSON formaadis. Samuti on Firebase'i sissehitatud autentimise funktsionaalsus ning lisaks ka „tasuta“ veebimajutus loodud veebiliidese jaoks. Nii veebimajutus kui ka reaalaja andmebaasis andmete hoiustamine on tasuta, kuni ei ületata teatud allalaetud andmete hulka. Ühe päeva maksimaalne andmeliiklus on 360 MB. Sellele arvule pole kogu arenduse käigus lähedale jõutud. Lõpptoote reaalsel kasutusel ei ületata seda piiri ning lisakulusid ei teki.

Alternatiivselt oleks võinud kasutada mõnda SQL (*Structured Query Language*) varianti, kuid tasuta SQL serveri majutust on varasema kogemuse järgi olnud keeruline leida. Lisaks peaks SQL serveriga eraldi ühenduma näiteks läbi MQTT (*MQ Telemetry Transport*) protokoll mikrokontroller ning arvuti poolt veebiliidese osa. MQTT ja SQL ühenduse vahel on lisakihte oluliselt rohkem, kui otse saata JSON andmeid mikrokontrollerilt Firebase andmebaasi.

MQTT oleks parem variant, kui on mitmeid erinevaid seadmeid, mis on omavahel ühenduses ning üsna tihedalt üksteisega sõnumeid vahendavad, näiteks mõni nutikodu lahendus. Firebase aga vahendab andmeid kasutades HTTP POST ja GET päringuid, nagu tavaliselt klient-server rakendused. Meie projekt on lähedasem klient-server rakendusele.

HTTP päringud on jõudluse poolest küll nõudvamad kui MQTT, kuid siiski on Firebase hea lahendus meie lõppeesmärke arvestades, sest saame ühest kohast väga suure osa vajalikest teenustest. Veebiliides vajab niikuinii majutust, samuti on vaja hoiustada kuskil logi ning ka autoriseeritud kasutajate andmeid. Lisaks võivad veebiliidesele ligi pääseda vaid teatud isikud. Veel on andmebaas reaalajaline, ehk kohe kui arvuti või mikrokontrolleri kaudu andmeid lisatakse, muudetakse või eemaldatakse, kajastuvad muutused ka andmebaasis. Kuna kõike seda pakub Firebase üksinda, otsustati selle kasuks.

4.4 Versioonihaldus

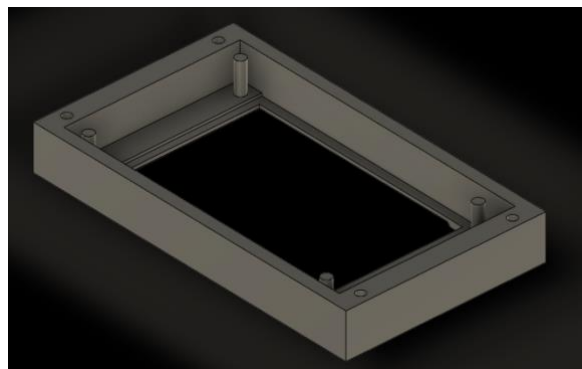
Versioonihalduseks kasutati Git'i ja GitHub'i. Git ja GitHub on jällegi kujunenud heaks praktikaks ka eelnevate projektide puhul ja autorid tunnevad end nendega mugavalt ning lisaks on Git integreeritav VSCode-ga. Projekti lähtekoodi hoiustasime GitHub'i repositooriumis.

4.5 Teegid

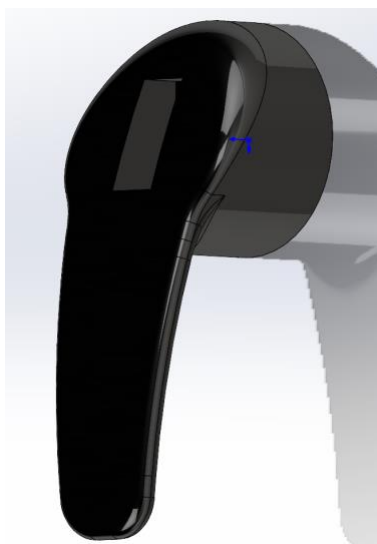
Main kontrolleri poolt kasutatavad kolmandate osapoolte poolt kirjutatud teegid olid: "Arduino.h", "EEPROM.h", "Wire.h", "ESP8266WiFi.h", "ArduinoJson.h", "Firebase_ESP_Client.h", "Adafruit_PN532.h". Ise kirjutatud teegid, mis haldasid või vastutasid konkreetsete ülesannete poolt, on välja toodud peatükis 7.1 Koodi struktuur.

4.6 SolidWorks

NFC lugeri, toitemuunduri ja ekraani korpuse mudel ning ukse link ja tekst korpuse peale modelleeriti kasutades SolidWorks 2022 tarkvara. SolidWorks'i mudeli saab eksportida STL-failiks, mille saab siis ette anda Ultimaker Cura tarkvarale. Detailid on prinditud PLA materjalist Ultimaker 2+ printeriga. Korpust prinditi kahes iteratsioonis.



Joonis 4. Lugeri ja ekraani korpuse mudelid



Joonis 5. Ukselingi mudel

4.7 DevTools

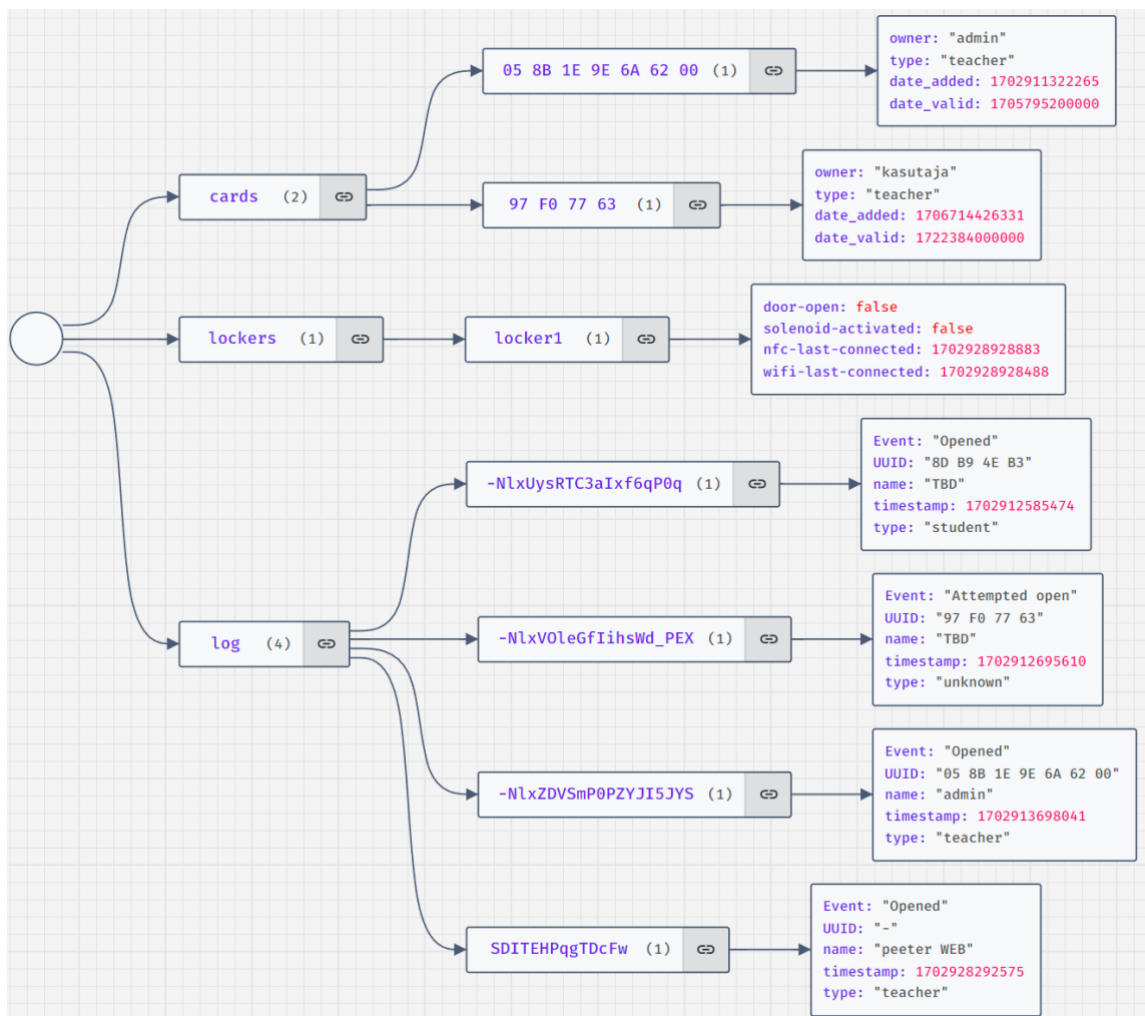
Microsoft Edge veebibrauser tuleb kaasa komplekti arendustööriistadega, mida tuntakse DevTools nime all. Need tööriistad ilmuvad veebilehel, kui seda parema hiireklahvi menüüst inspekteerida, või need avada arendaja menüüst [2]. DevTools'e kasutati laborikapi veebiliidese arendamiseks, et luua reageeriv ja kaasaegne veebileht, mis muutub vastavalt kasutaja ekraani suurusele. Selleks kasutati DevTools'i võimalust erinevate ekraanide ja resolutsioonide emuleerimiseks. Lisaks sai tänu nendele tööriistadele kuvada veebikonsooli, kuhu oli mugav funktsionaalsuse silumiseks erinevaid sõnumeid saata.

4.8 Photoshop

Laborikapi veebiliidesele logode ning taustade loomiseks kasutati Adobe Photoshop fototöötlustarkvara. Antud töövahendi valik tulenes sellest, et projekti liikmetel oli sellega varasem kogemus olemas ning see oli ka arendusmasinasse installeeritud. Antud programmiga oli väga mugav salvestada juba varasemalt valmistatud logosid erinevate resolutsioonidega ning vajadusel värvitoone muuta.

5 Andmemudel

Peatükis kirjeldatakse ning illustreeritakse, milline on kasutatav andmemudel ning mis on erinevate JSON objektide eesmärgid, andmetüübid jne. „Tükk“ andmemudelid on nähtav jooniselt 6.



Joonis 6. JSON-andmemudel graafidena.

Andmemudel koosneb kolmest põhilisest JSON objektist, milleks on „cards“, „lockers“ ja „log“. Järgnevalt on lahti seletatud iga eelnimetatud objekti sisu.

5.1 Cards

Objekti „cards“ alla kuuluvad kõik kasutajad andmebaasis, kaasaarvatud ka kasutajad, kelle õigused ei ole aegunud. Näitena toodud Joonis 6 pealt on näha, et parasjagu eksisteerib 2 kasutajat. Kasutaja objektide võtmeteks/nimedeks on NFC kaardi ID koodid. Antud võtmed on valitud selliselt, et teha otsimine andmebaasist võimalikult optimaalseks. Alternatiivselt, kui kasutaja objekti nimeks oleks näiteks kasutaja eesnimi ning NFC kaardi ID oleks omakorda selle objekti sees, siis tekiks otsimisel üks lisasamm ning peaks minema eraldi iga objekti sisse ja otsima sealt kaardi ID-sid. Sõne andmetüübina on iga kasutaja objekti sees „owner“, mis tähistab kasutaja nime (kaardi omanikku) ja kaardi tüüp „type“, mis saab olla kas „teacher“ või „student“. Numbrilise andmetüübina on kasutaja lisamise ning aegumise ajad „date_added“ ja „date_valid“ Unix’i ajatempli formaadis.

5.2 Lockers/locker1

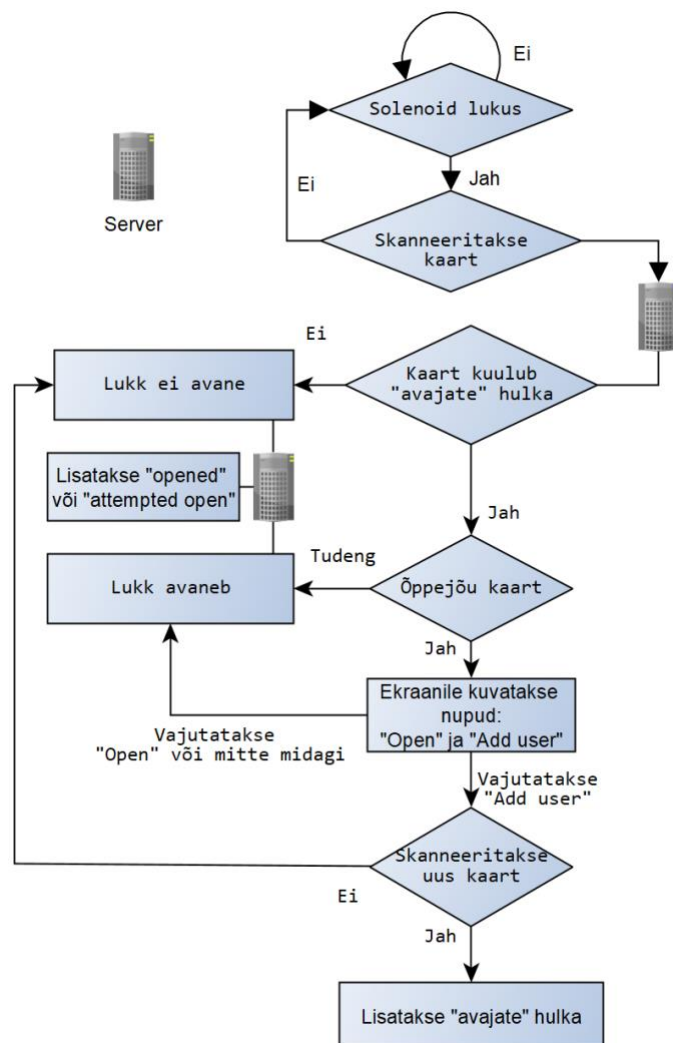
Objekti „locker1“ alla kuuluvad kapiga seonduvad staatused. Tõeväärtuse andmetüüpi „door-open“ staatust muudetakse andmebaasis mikrokontrolleri poolt iga kord, kui on tuvastatud laborikapi ukse oleku muutumine. Tõeväärtuse andmetüüpi „solenoid-activated“ staatust muudetakse andmebaasis, kui veebiliidesest on seda muudetud. Seda muutajat kontrollib mikrokontroller ning vastavalt loetud olekule suudab solenoid-luku avada või sulgeda. Numbrilise andmetüübina, samuti Unix’i ajatempli formaadis, on ka „nfc-last-connected“ ja „wifi-last-connected“. Neid muutujaid uuendab mikrokontroller iga teatud aja tagant ning neid väärtuseid kontrollitakse veebiliideses, et kuvada, kas NFC ja WiFi on ühendatud.

5.3 Log

Objekti „log“ sees on kõik logisse kuuluvad kirjed. Iga logi kirje objekti nimi on Firebase poolt genereeritud unikaalne kood. Iga kirje objekt sisaldab sõne tüübiga sündmust „Event“, mis saab olla väärtusega „Opened“, kui kapp avatakse, või „Attempted open“, kui õigusi ei ole, aga prooviti avada. Sõne andmetüübiga on veel „UUID“, „name“ ja „type“. „UUID“ on NFC kaardi ID kood. „Name“ on samaväärne kasutaja objekti „owner“ väljaga. Kui kasutajale nime määratud pole, või skanneritakse kaart, mille ID-d kasutajate „cards“ hulgas pole, siis antakse „name“ väljale väärtus „TBD“ (*To Be Determined*). Väli „type“ saab olla kas „student“, „teacher“ või „unknown“. „Unknown“ tüüp on logi kirjel siis, kui skanneritud kaardi ID-d ei eksisteeri kasutajate „cards“ hulgas.

6 Kapi juhtloogika

Siin peatükis seletatakse ära kapi üldine juhtloogika. Joonisel pole kogu funktsionaalsust välja toodud ning peatükis ei seletata ära lisafunktsionaalsusi, näiteks seadmete ühendusstaatuse saatmist, EEPROM osa jm. Seletatakse vaid olulisemad ülesanded ning mõnes kohas on lähemalt kirjeldatud mõned sammud, mida Joonis 7 peal kirjas ei ole. Alamülesanded ja lisafunktsionaalsused selgitatakse täpsemalt lahti peatükkides 7.1.1 – 7.1.6. Joonis 7 on lisatud illustatsiooniks, et oleks lihtsam aru saada ning jälgida kapi tähtsamat juhtloogikat.



Joonis 7. Lihtsustatud kapi juhtloogika

Esiteks peab olema *main* kontrollerrakendus käivitatud ning käivitamisega seotud alglaadimised ja konfiguratsioonid tehtud (vt Lisa 1).

Solenoid peab olema lukustatud olekus, et oleks võimalik NFC kaarti skanneerida.

Kui skanneeritakse kaart, kostub sumistist kõrgsageduslik noot ning LED vilgub ühe korra, misjärel päritakse Firebase andmebaasist, kas skanneeritud ID koodiga kasutaja asub avajate hulgas.

Kui kasutajat selle ID koodiga avajate hulgas pole, siis solenoid-lukku ei avata, ekraanile kuvatakse kiri „No access“ ning Firebase andmebaasi logisse lisatakse märge „Attempted open“ koos sündmuse toimumise ajaga, kaardi ID koodiga, nimega „TBD“ ja tüübiga „unknown“.

Kui kasutaja selle ID koodiga on avajate hulgas olemas, kaardi tüübiks on „student“ ning kasutaja õigused on parasjagu kehtivad, siis solenoid-lukk avatakse, sumistist kostub uus kõrgsageduslik noot ja LED vilgub korra ning ekraanile kuvatakse kiri „Lock opened“, pärast mida lisatakse Firebase andmebaasi logisse märge „Opened“ koos avamise ajaga, kaardi ID koodiga ning kasutaja nimega. Kui kaardi ID kood on avajate hulgas olemas ning kaardi tüübiks on „teacher“ ja õigused on parasjagu kehtivad, siis kuvatakse ekraanile kaks nuppu: „Open door“ ja „Add student“.

Vajutades „Open door“ avatakse kapp tavapäraselt ning logisse lisatakse jällegi vastava kasutaja andmed. Vajutades „Add student“ kuvatakse ekraanile sõnum „Scan card to add user“. Sõnumi alla kuvatakse taimer, mis loeb sekundites 10-st 0-ni. Kui 10 sekundi jooksul skanneeritakse edukalt uus kaart, kostub sumistist noot, ekraanile kuvatakse sõnum „Successfully added new user“ ning kasutaja lisatakse Firebase andmebaasis avajate hulka ja lisatud kaardiga on võimalik koheselt kappi avada.

Kui kasutaja lisatakse nupuga „Add student“, siis vaikimisi seatakse kasutaja tüübiks „student“, kehtivusaegaks kuni käesoleva semestri lõpp, ning nimeks „TBD“. Kui 10 sekundi jooksul uut kaarti ei skanneerita ja taimer 0-ni jõuab, ei juhtu mitte midagi. Kui nuppudest „Open door“ või „Add student“ kumbagi 5 sekundi jooksul ei vajutata, siis avatakse solenoid-lukk.

7 Sardtarkvara

Antud peatükis kirjeldatakse sardtarkvara koodi kirjutamiseks kasutatud tööriistu, sh kasutatud tarkvara, teeke, koodi struktureerimisel rakendatud põhimõtteid, ülesehitust. Samuti on kirjeldatud üldiselt, mis on iga koodi komponendi nõuded ja ülesanded.

7.1 Koodi struktuur

Kood on GitHubis jagatud kahte harusse, “master” ja “display”. Kui üldiselt ei kasutata „master“ haru, et arendusega tegeleda, vaid sinna pannakse teisest harust valmis saanud lõpp-toode, siis meie projekti lahenduses ei olnud otsest vahet, kuna “display” ja “master” haru ei liideta kunagi kokku ja nad jäävad lõplikult lahku. Põhjuseks on see, et “display” haru kasutab ainult *slave* kontrolleri ja “masterit” ainult *main* kontrolleri. Mugavam oli hoida kogu kood ühe repositooriumi erinevates harudes, kui kahe erineva repositooriumi kasutamine.

Lisaks tegelesid “master” ja “display” harude arendamisega ka kaks erinevat isikut, seega oli versioonihalduskonflikte kerge vältida just eraldatuse tõttu.

“Master” haru siseselt on kood jaotatud nii-öelda komponentideks, kus iga komponendi kohta on loodud teek ja iga komponent vastutab üldiselt mingi konkreetse ülesande eest. Näiteks teek nimega “nfc.h” koos “nfc.cpp”-ga sisaldab kogu koodi, mis puudutab PN532 NFC lugerit ning sellega seonduvaid ülesandeid. Selline struktuur võimaldab eraldada erinevaid koodi osi ja nende ülesandeid. Sisuliselt tekitatakse API (*Application Programming Interface*) teek konkreetse komponendi jaoks, mida peaprogrammis “main.cpp” kasutatakse. PlatformIO-siseselt peavad olema vastavad teegid projekti lib kaustas, siis käib *toolchain* rekursiivselt kaustad läbi ning aitab *linker*’il kõik kaustades olevad .h ja .cpp laienditega failid üles leida. Põhikood, kus asub „main.cpp“, paikneb projekti src kaustas.

Meie poolt kirjutatud teegid asuvad oma funktsionaalsusega sarnase nimega olevas kaustas: “locker.h”, “nfc.h”, “timeutils.h”, “user_eeprom.h”, “FBServer.h” ja “SerialDebug.h”. Järgnevates alapeatükkides kirjeldatakse nende loogikat.

Main kontrolleri koodis välditakse viitefunktsioonide, ehk *delay* kasutamist, et programm oleks võimalikult kiire. Kuna andmebaasi päringud läbi interneti on juba üldiselt blokeerivad, mis tähendab, et mingite andmete pärimisel oodatakse teatud aeg, kuni vastus jõuab. Sel ajal kontrollir muude asjadega tegeleda ei saa. Lisaks on ESP8266 protsessor ühetuumaline ning mitme-lõimulisus on võimatu. Viitefunktsioonide asemel kasutatakse hulganisti ära *polling* põhimõtet. Alati ei ole *polling* mõne teise seadme staatuse kontrollimiseks, vaid vaadatakse, kas teatud aeg on möödunud, et mõnda funktsiooni välja kutsuda. Näiteks iga 10 minuti tagant saadetakse andmebaasi NFC lugeri staatus, kas seade on ühendatud või mitte. Viitefunktsioonid on aktsepteeritavad ainult programmi käivitamise ajal, kui neid kutsutakse välja vaid korra.

7.1.1 NFC

Meie loodud API teek nimega nfc.h sõltub veel kolmandatest teekidest, nagu “Wire.h” ja “Adafruit_PN532.h”. Kolmandad teegid on vajalikud I2C liidese ja PN532 draiverite jaoks.

Teegi peamisteks ülesanneteks on NFC lugeri käivitamine ja ühenduse loomine, lugeri konfigureerimine ning lõpuks NFC kaartide lugemine. Luger konfigureeritakse kasutama IRQ väljaviiku ja kasutatakse passiivset lugemist, mis tähendab, et PN532 on kogu aeg lugemise režiimis [3]. See võimaldab sujuva kasutuse lõpptarbija jaoks. Kui *main* kontrollir parasjagu täidab mõnda teist ülesannet ning samal ajal skanneeritakse NFC kaart, siis jätab luger skanneerimise meelde, salvestab kaardiandmed ära ning hiljem, kui protsessori ressursid on PN532-le kättesaadavad, saab salvestatud kaardiandmed kätte.

Lisaturvalisusena taaskäivitab NFC luger ennast peale igat viit minutit. Testides makettplaadil PN532 seadet, siis vahepeal kadus ära ühendus ESP ja PN532 NFC lugeri vahel. Aitas manuaalne VCC või GND uuesti ühendamise või tarkvaraline taaskäivitus NFC seadmele. Kui ESP aga PN532 külge kinni jootsime, siis ühenduse viga enam ei ilmnenud. Siiski on jäetud lõppkoodi sisse lugeri tarkvaraline taaskäivitus juhuks, kui peaks tekkima näiteks olukord, kus kapi uste paugutamisel ühendused halvenevad, või mingil muul põhjusel I2C siinil ühendus katkeb.

7.1.2 Locker

Locker API teegi ülesanneteks on kõik, mis on seotud kapi ukse ja luku avamise või sulgemisega. Lisaks ka sulgemisest ja avamisest tagasiside andmisega. Locker kontrollib triviaalsemaid seadmeid läbi GPIO väljaviikude. Nendeks seadmeteks on magnetandur, solenoid-lukk (läbi relee), passiivne sumisti ning indikaator LED.

Alguses teostatakse kõikide GPIO väljaviikude alglaadimine, et seadmeid kontrollida. Magnetanduriga ühendatav väljaviik konfigureeritakse sisend-režiimis, kasutades lisaks ka ESP sisemist *pull-up* takistit. Ülejäänud seadmetega ühendatud väljaviigud konfigureeritakse väljund-režiimis.

Magnetanduri väljaviigule lisatakse ka *interrupt* funktsionaalsus. *Interrupt* toimub digitaalsignaali loogilisel muutumisel ehk kui signaal muutub kas loogilise 0 tasemelt loogilise 1 tasemele, või loogilise 1 tasemelt loogilise 0 tasemele. See võimaldab kutsuda *interrupt* funktsiooni mõlemal korral, nii ukse avamisel kui sulgemisel. *Interrupt* funktsiooni käigus kuvatakse ukse olekut ka ekraanil. Kui ukse staatus peaks jääma millegipärast valesse olekusse, siis kontrollitakse ka seda ja vajadusel parandatakse ära.

Kui solenoid-lukk avatakse, siis kontrollitakse, kas see on olnud avatud olekus teatud aja, enne kui lukk jälle suletakse.

Tagasiside kapi olekutest saab kasutaja sumisti ja punase LED-i kaudu. Esiteks kostub sumistist kõrgsageduslik noot ja LED teeb ühe vilkumise, kui kaardi skanneerimine ning lugemine on edukas. Teine kord vilgatab LED ja kostub noot siis, kui solenoid-lukk avatakse.

7.1.3 Firebase ja võrguühendused

Kõik võrguga/WiFi-ga seotud on failides FBServer.cpp ja FBServer.h ning nad sõltuvad kolmandatest teekidest: “ESP8266WiFi.h”, “Firebase_ESP_Client.h” ja “ArduinoJson.h”. API funktsioonid võimaldavad alguses luua ühenduse TalTech’i WiFi’ga ning seejärel ühendada meie Firebase reaalse andmebaasiga.

Peamine ülesanne on kontrollida Firebase andmebaasist, kas skanneeritud kaardi ID asub avajate nimekirjas. Kapi avamisest ja ka nurjunud avamisest lisatakse märke logisse. Kasutajad kaarditüübiga „teacher“ saavad kappi avada ja avajate hulka uusi õpilasi lisada.

Kui WiFi ühendus puudub, vaadatakse kas skanneeritud kaardi ID asub ESP EEPROM mälus.

Õppejõule on lisatud võimalus avada kapi lukk ka läbi veebiliidese. Selle jaoks *poll*-itakse Firebase andmebaasis olevat solenoidi muutuva olekut iga teatud aja tagant, et tuvastada, kas õppejõud soovib kappi avada. Vastavalt solenoidi muutuva olekule saab *main* kontrollida solenoidi avada ja lukustada.

Ukse avatuse olekut on samuti võimalik läbi veebiliidese vaadata. Ukse olekut muudetakse veebiliideses iga kord, kui muutub ukse oleku staatus programmi töö käigus. Kui ust avatakse või suletakse, saadetakse andmebaasi uus ukse olek ja seda kuvab vastavalt veebiliides.

Lisaks saadetakse iga teatud aja tagant andmebaasi ka WiFi ning NFC lugeri staatused. Juhuks kui WiFi ühendus peaks katkema, või NFC luger ajapikku lahti ühenduma või katki minema, siis saab õppejõud sellest kohe teada läbi veebiliidese. WiFi ühenduse katkemise korral proovib kontrollida WiFi'ga taas ühendust luua iga teatud aja tagant.

7.1.4 EEPROM

ESP8266 puhul tegelikult tõelist EEPROM mälu ei ole, vaid kasutatakse CPU *flash* mäluosa ning emuleeritakse EEPROM-i (edaspidi kutsutakse aruandes siiski seda mäluosa EEPROM mäluks). Flash mälu suurus on maksimaalselt 4MB, kuid sellest on EEPROM mälule eraldatud maksimaalselt 4 kB ehk 4096 baiti [4]. Lisaks on tavalisel EEPROM mälul kustutamise-kirjutamistsüklite arv ca 100 000 korda, kuid EEPROM'i emuleerival mälul umbes 10 korda väiksem, ca 10 000 korda. Seega tuleb jälgida kirjutamiste kordust ja teha seda võimalikult vähe. Lugemisega seonduvat maksimaalset tsüklite arvu ei arvestata, sest see on piisavalt suur, et lugemisega probleeme ei kaasne.

EEPROM mälu kasutamine on lisatöökindluse tagamiseks, kui labori ruumis või TalTech'is peaks WiFi ja ESP vahel ühendus katkema. Siis on võimalus kasutada Firebase andmebaasi asemel ESP-s olevat muudetavat püsimälu. Püsimälu on

salvestatud avajate ID-d ning kui skanneeritava kaardiga identne ID leidub, saab kapi avada ka ilma WiFi'ta.

Programmi alguses algseadistatakse EEPROM mälu suurusega 2048 baiti ehk 50% kogu kasutatavast EEPROM mälust. EEPROM mällu lisatakse mälu kokkuhoidmise eesmärgil ainult NFC kaartide ID koodid, maksimaalne ID pikkus on 21 baiti. Algseadistatud mälu suurus võimaldab lisada EEPROM mällu kuni:

$$\frac{2048 \text{ baiti}}{21 \text{ baiti}} = 97,52$$

Seega on võimalik lisada kuni 97 kasutajat ning 11 baiti jääb 2048-st kasutamata. Kuna juhendaja sõnul nii palju inimesi kappi arvatavasti kunagi kasutama ei hakka, siis sellist aktiivsete kasutajate arvu kunagi täis ei saa. Aktiivsed kasutajad on need, kelle kaardil on ajaliselt kehtivad õigused.

ESP käivitamise ajal kui ka iga 12 tunni tagant, kui on olemas WiFi ühendus, päritakse kõik avajate kaartide ID-d ning kontrollitakse eelnevalt EEPROM mälus olemasolevaid ID-sid. Kui need ID-d ei ole samad, siis uuendatakse EEPROM mälus olevaid ID-sid. Isegi kui peaks juhtuma, et EEPROM mälu imekombel uuendatakse iga 12 tunni tagant ehk 2 korda päevas, kulub väga palju aega, enne kui kustutamise-kirjutamistsükli arv täis saab.

$$\frac{5000}{2 \times 365} = 6,85 \text{ aastat}$$

Juba 5000 kustutamise-kirjutamistsükli täis saamiseks kulub 6,85 aastat. Saab arvestada, et EEPROM mälu tõttu kapp töötamast ei lakka.

7.1.5 SerialDebug

Selles API teegis kasutatakse UART jadaliidesega seotud ülesandeid. Esiteks on silumiseks mõeldud peamine jadaliides, mis on ühendatud USB kaudu. Jadaliidesega on võimalik arenduse käigus konsoolist lugeda programmi muutujate väärtuseid, olekuid, silumiseks koostatud sõnumeid jne. Kuna arenduse lõppkäigus enam neid sõnumeid vaja ei lähe ja keegi neid ei loe, siis ressursi kokkuhoidmise jaoks võib need sõnumid välja lülitada.

```

#ifdef DEBUG
    #define DBGL(x) Serial.println(x)
    #define DBG(x) Serial.print(x)
#else
    #define DBGL(x) ((void)0)
    #define DBG(x) ((void)0)
#endif

```

Joonis 8. UART sõnumite sisse-väljalülitamise makrod.

Kui “DEBUG” pole defineeritud, siis sõnumeid ei kuvata. See võimaldab silumise teha mugavamaks, näiteks kui soovitakse siluda ainult NFC API-t, siis võib lihtsalt NFC kaustas “DEBUG”-i ära defineerida, selle asemel, et pidevalt “Serial.print()” funktsiooni välja kommenteerida.

Lisaks on SerialDebug kaustas ka tarkvara jadaliidese ülesanded. *Master* kontrolleri tarkvara jadaliides on ühendatud *slave* kontrolleri riistvaralise jadaliidesega. Mõlemad jadaliidesed peavad kasutama sama kiirust ning nad on konfigureeritud edastuskiiruse peale 9600 baudi, maksimaalne on 115200 baudi [5]. Valik tulenes sellest, et väiksemale edastuskiirusele on omane parem müra- ja veakindlus. Samuti mida pikemaks lähevad andmeedastusjuhtmed, seda väiksem on soovituslik edastuskiirus. Meil väga pikkasid juhtmeid ei ole, kuid veakindlust eelistades valisime piisavalt kiire ja hea veakindlusega edastuskiiruse [6], [7]. Nii vaikimisi kui ka meie projekti puhul on jadaliidesed “Arduino.h” teegis konfigureeritud 8-N-1 edastuse peale [8]. See tähendab, et ühes jadaliidese nii-öelda saadetud sõnumi koosseisus on 1 stardi bitt, 8 andmebiti, 0 paarsusbiti ja 1 stopp bitt. Seega kokku on sõnumis 10 bitti.

Biti kestvus t , baidi saatmise kiirus k ja stopp ja start bitt v , efektiivne saatmise kiirus v_{ef} .

$$t = \frac{1}{9600} = 104\mu s$$

$$v = t \times 10 = 1,04ms$$

$$v_{ef} = \frac{8(bitti)}{v(s)} = 7680 \frac{bitti}{s}$$

Kuna meil on seadistatud ühe käsu suuruseks täpselt üks bait, siis edastuskiirus on selle jaoks piisav, lisaks kindluse huvides piisavalt väike, et signaali ei häiriks müra ning andmeedastus oleks efektiivne.

Master kontrolleri saadab üle jadaliidese kärke *slave* kontrolleri, et *slave* teaks, mida parasjagu tuleb ekraani peal kuvada. Samuti saab *slave* kontrolleri saata ka *master* kontrolleri andmeid.

Kui peamine jadaliides kasutab andmeedastuseks TX ja RX väljaviike, siis need on riistvaraliselt juba loodud jadaliidese kasutamiseks. Tarkvaraline jadaliides püüab seda sama emuleerida kasutades kahte GPIO väljaviiku, mis on konfigureeritud tarkvaralise jadaliidese TX ja RX väljaviikudeks. Tarkvaralise jadaliidese kasutamine oli vajalik, et riistvara TX ja RX väljaviigud jääks arenduse käigus vabaks, et saaks programmi USB ühendusega arvutis siluda ning et *main* kontrolleri ei liiguks ebavajalikke andmeid *slave* kontrolleri.

7.1.6 Aja abifunktsioonid timeutils

Abifunktsioone kasutab vaid Firebase ja võrguühendusega seonduv. Eesmärgiks on kontrollida, et tudengite kaartide õigused pole aegunud.

Aja abifunktsioonid võimaldavad ühendada NTP serveriga, et konfigureerida kontrolleri kasutama Eesti reaalaega. Selle abil on võimalik võrrelda Firebase andmebaasis olevate kasutajate õiguste aegumise aega praeguse ajaga ja selle põhjal otsustada, kas kasutaja peaks kapile ligi pääsema. Teiseks, kui lisatakse kasutajaid, siis arvutatakse praeguse aja järgi uute kasutajate aegumise aeg. NFC lugeriga skanneeritud kaardi õiguste lisamisel antakse õigused praeguse semestri lõpuni. Semestri lõpu ajad on võetud TalTech'i akadeemilise kalendri järgi, TalTech'i kodulehelt [9].

Lisaks on võimalik konverteerida abifunktsioonide Unix süsteemidele omane ajatempel sellele vastavaks loetavaks formaadiks ning ka vastupidi. Näide: ajatempel 1702901868 on Eesti ajavööndis 18.12.2023 kell 14:17:48.

8 Veebiliides

Veebiliides on laborikapi süsteemi osa, mille abil kapi haldaja (õppejõud) saab kontrollida kapi olekut, vaadata kapi kasutamise ajalugu, lisada ja eemaldada kasutajaid, kes kapi sisule ligi pääsevad. Veebiliidese loomine tulenes projekti nõudest, et kapil peab olema jälgitav ajalugu ja võimalus lisada või eemaldada kapi kasutamisoigusi. Muud veebiliidese funktsioonid on loodud vastavalt tellija soovile. Veebiliides on külastatav Firebase pakutavas tasuta pilvepõhises veebimajutuses aadressil

<https://nutikapp.web.app/>.

8.1 Tehnoloogia

Veebiliidese loomiseks otsustati kasutada Vue.js *frontend* raamistikku, kuna taheti luua kaasaegne ning kasutajasõbralik muutustele reageeriv lehekülg. Võrreldes teiste suuremate ja aktuaalsete raamistikega, nagu Angular ja React, langes valik just Vue kasuks järgnevatel põhjustel.

Angular on Google raamistik ning sellega tuleb kaasa väga suurel hulgal sisseehitatud laiendusi ning funktsioone, mis võib olla *overkill* sellisele minimaalsele veebiliidesele, nagu antud projekti raames vaja on. React on Meta hallatav raamistik ning sellega tuleb kaasa väga vähe pakette, mis on teine äärmus. Enamik React'i laiendusi ei ole loodud Meta poolt, vaid tuleb leida kolmandate isikute poolt loodud teekidest. Vue.js on aga täpselt vahepealne. See on loodud eraldiseisva arendajate tiimi poolt, mitte suurkorporatsiooni, ning omab rohkem sisseehitatud funktsionaalsusi kui React ja vähem kui Angular, ehk siis on kuldne kesktee [10].

Teine mõjuv põhjus Vue.js valimiseks oli projekti jaoks antud aeg. Kuna meil polnud kellelgi veebiarendusega varasemat kogemust ja ei olnud pikka aega keeruka raamistiku õppimiseks, oli vaja leida midagi, mida oleks lihtne kasutama hakata. Erinevate

raamistike kohta uurides selgus, et Vue.js õppimiskõver on võrreldes teiste raamistikega palju lineaarsem ning projekt on võimalik kerge vaevaga üles seada [10].

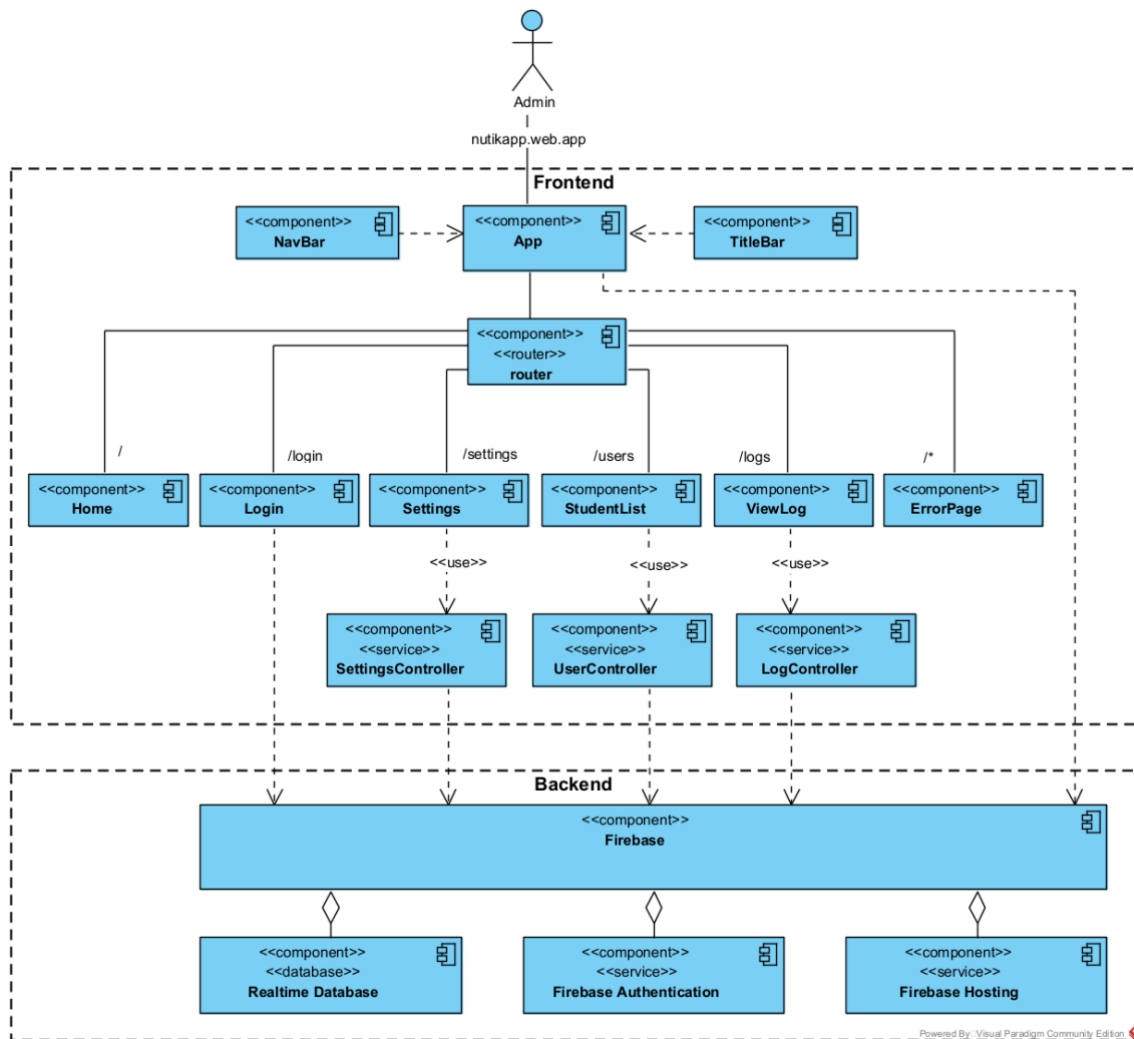
Veebiliidese loomisel kasutatakse Vue.js kolmandat versiooni, mis on projekti arendamise ajal kõige uuem versioon. Tüüpiline .vue laiendusega fail koosneb ühest kuni kolmest osast. Üks osa on *template* või DOM (*Document Object Model*), mis on sisuliselt HTML'is kirjutatud veebilehe struktuuri kood, kuhu on põimitud Vue jaoks teatud märksõnad. See osa algab. Teises osas on koodi funktsionaalsus, mis on kirjutatud JavaScriptis või TypeScriptis. Kolmandas osas on kõik, mis puudutab veebiliidese kujundust. See osa on kirjutatud CSS kujunduskeeles.

8.2 Arhitektuur ja komponendid

Veebiliidese arhitektuuris otsustati kasutada kapseldamise põhimõtet, kuna sarnasel põhimõttel ülesehitatud eelsätte pakkus ka arenduseks kasutatav Vue.js raamistik. Kapseldamise eelis on see, et eraldiseisvaid komponente, mis täidavad oma kindlaid ülesandeid, on lihtsam hallata ja välja vahetada, kui üht suurt monoliiti. Lisaks on programmist parem ülevaade ja ka lihtsam dokumenteerida.

Veebiliides koosneb meie poolt loodud *frontend*'ist ning Firebase poolt pakutavast *backend*'ist, mis hõlmab endas andmebaasi, autentimist ning veebilehe majutust pilves. Kokku on süsteemil 17 komponenti, 13 meie poolt loodud *frontend*'is ning 4 Firebase *backend*'is. *Frontend*'i komponendid jaotuvad omakorda järgnevalt: teenused, vaated, ruuter, põhirakendus ning lisakomponendid (vt Joonis 9).

Järgnevalt kirjeldatakse loodud *frontend* komponente. *Backend* komponentide kohta on täpsem info leitav Firebase dokumentatsioonist [11].



Joonis 9. Veebiliidese komponentdiagramm.

8.2.1 Teenused

Teenused on komponendid, mis toimivad liidestena Firebase ning vaadete vahel, nende faililaiend on .ts ehk need on kirjutatud TypeScript'is. Vaates kutsub teatud nupu vajutus välja teenuse, mis pärib Firebase andmebaasist vajalikud andmed. Iga teenus vastab kindlale vaatele ja omab funktsionaalsust vaid selle vaate jaoks.

- **UserController.ts** vastab vaatele StudentList.vue ning omab järgnevaid meetodeid:
 getUsersAndSort(sortField, sortOrder) - see meetod küsib Firebase andmebaasist kõik 'cards' JSON objektis olevad kasutajad ning tagastab õige vorminguga ning meetodile antud parameetritele vastavalt sorteeritud kasutajate nimekirja, kasutab omakorda meetodeid formatUserData ning sortTable;

`formatUserData(users)` – see meetod vormindab parameetrina saadud kasutajate nimekirja tabelile sobivaks formaadiks ja tagastab vormindatud nimekirja;

`sortTable(data, sortField, sortOrder)` – see meetod sorteerib data parameetri ehk kasutajate nimekirja vastavalt `sortField` ja `sortOrder` parameetritele ja tagastab sorteeritud nimekirja, parameetri `sortField` väärtuseks on välja tüüp (nimi, tüüp, lisatud, aegub), `sortOrder` väärtuseks on sorteerimise järjekord, kas 0 (kahanev) või 1 (kasvav);

`addNewUser(owner, UUID, type, date_valid)` – selle meetodi parameetriteks on lisatava kasutaja andmed (omanik, identifikaator, tüüp, kaua kehtib), meetod lisab kasutaja Firebase andmebaasi 'cards' objekti;

`updateUser(userId, updatedUserData)` – see meetod uuendab `userId` parameetriga määratud kasutaja andmeid Firebase andmebaasis, uued andmed on parameetris `updatedUserData`;

`removeUser(UUID)` – see meetod eemaldab parameetrina antud identifikaatori põhjal Firebase andmebaasist 'cards' objektist ühe kasutaja.

- **SettingsController.ts** vastab vaatele `Settings.vue` ning omab järgnevaid meetodeid:

`stringGen(length)` – see meetod tagastab suvalise suur ja väiketähtede jada, mida hiljem kasutatakse logi sissekande identifikaatorina, *length* parameeter määrab jada pikkuse;

`getLockerStatus(property)` – see meetod tagastab `property` parameetri põhjal Firebase andmebaasist parameetri väärtuse;

`checkWifiStatus()` – see meetod kontrollib Firebases andmebaasis oleva 'wifi-last-connected' parameetri olekut ning muudab vastavalt sellele 'wifi-connected' parameetri olekut, kui WiFi oli viimati ühendatud rohkem kui 10 minutit tagasi, muutub WiFi olek *false*'ks;

`checkNfcStatus()` – see meetod toimib samamoodi, nagu eelmine, kuid kontrollitakse WiFi asemel NFC olekut;

toggleLockerStatus() – see meetod muudab Firebase andmebaasis 'solenoid-activated' välja staatust, mis põhjustab kapi avamise või lukustamise;

addLogEntry(name) – see meetod lisab logisse sissekande parameetrina antud nimega (milleks on veebiliidesesse logitud kasutaja nimi), kui kapp avatakse veebiliidese kaudu.

- **LogController.ts** vastab vaatele ViewLog.vue ning omab järgnevaid meetodeid:

getLogsAndSort(sortField, sortOrder) – see meetod toimib sarnaselt getUsersAndSort meetodiga UserController teenuses, küsib Firebase andmebaasist 'log' JSON objektis olevad logi sissekanded, vormindab need ning tagastab sorteeritud loetelu;

checkUuidType(uuid) – see meetod kontrollib, kas parameetrina antud identifikaatoriga kasutaja juba eksisteerib Firebaas andmebaasis 'cards' objektis, tagastab tõeväärtuse;

formatLogsData(logs) – see meetod vormindab parameetrina antud nimekirja tabelile sobivasse formaati ning tagastab vormindatud nimekirja;

sortTable(data, sortField, sortOrder) – see meetod sorteerib data parameetri ehk logi sissekannete nimekirja vastavalt sortField ja sortOrder parameetritega, tööpõhimõte on sama, mis sortTable meetodil UserController teenuses.

8.2.2 Vaated

Vaated on komponendid, mis sisaldavad veebilehe ühe konkreetse alamlehe funktsionaalsust ja kujundust. Iga vaade vastab kindlale marsruudile aadressiribal. Vaadete faililaiend on .vue ning nad on kirjutatud kasutades Vue.js raamistiku tüüpilist kolme osalist süntaksit.

- **ErrorPage.vue** vastab marsruudile '/*', mis tähendab, et iga marsruut, mida pole ruuteris defineeritud, juhatab selle vaateni (vt Lisa 3). Komponent meetodeid ei sisalda.

- **Home.vue** vastab marsruudile '/' ehk lihtsalt veebilehe aadress. Antud komponendi puhul on tegu veebiliidese avalehega (vt Lisa 4). See vaade avaneb, kui veebilehele sisse logida. Vaate keskel on pilt laborikapist ning selle all kolm nuppu, mis juhatavad kasutajate, seadete ning logide alamlehtedele. Komponendis on järgnevad meetodid:

`onMounted()` – see on Vue.js üks elutsükli *hook* ehk konks, mis tähendab seda, et kõik mis selle sees on käivitatakse, kui antud komponent ühendatakse vastava vaate komponent avatakse, selle komponendi puhul sisaldab see endas veebilehe keha klassi muutmist, et kohandada lehte erineva suurusega ekraanide jaoks; `navigateTo(path)` – see meetod kasutab ruuterit, et lisada veebiaadressile parameetrina antud teekond, seda meetodit kasutavad komponendi nupud, et liikuda teistele alamlehtedele.

- **Login.vue** vastab marsruudile '/login' ning kujutab endas sisselogimise alamlehte. Kui laborikapi veebiliides avada, on see esimene leht, mida kasutaja näeb. Antud vaate taustapildiks on joonis laborikapist ning selle keskel on sisselogimisvorm (vt Lisa 5). Et antud vaatest koduleheni edasi pääseda, on vaja sisestada sisestada sobiv kasutajanimi ja parool, mis on määratud Firebase konsooli lehel. Komponendis on järgnevad meetodid:

`login()` – see meetod kasutab Firebase autentimist, et kasutaja veebilehele sisse logida, kui õige kasutaja nimi ja parool on sisestatud;

`handleLoginError(error)` – see meetod muudab sisselogimisel tekkinud veateate sõnumi eestikeelset väärtust vastavalt veateate tüübile;

`onBeforeMount()` – see on Vue.js elutsükli *hook*, mis käivitatakse enne komponendi lisamist ehk enne, kui antud alamleht avaneb, sisselogimislehe puhul sisaldab see endas `setTimeout()` funktsiooni, mida kasutatakse veebilehe laadimise näitamiseks, et taustal saaks toimuda autentimise kontroll, lisaks on seal rakenduse keha klassi muutmine, et kohandada erinevate ekraani suurustega.

- **Settings.vue** vastab marsruudile '/settings' ning kujutab endas laborikapi staatuse alamlehte. Selle vaate avamiseks saab vajutada vastavat nuppu kodulehel või navigatsiooniriba komponendil. Antud vaates on ekraani keskel kapi staatuse

aken koos järgnevate väljadega: WiFi staatus, NFC staatus, luku olek, ukse olek (vt Lisa 6). Iga välja väärtus on otseselt seotud Firebase andmebaasis 'locker' JSON objektis asuva 'locker1' väljadega. WiFi staatusele vastab väli 'wifi-connected', NFC staatusele 'nfc-connected', luku olekule 'solenoid-activated' ning ukse olekule 'door-open'. Oleku väljade all on nupp kapi lukustamiseks ja avamiseks. Komponendis on kasutusel järgnevad meetodid:

`checkLockerStatus()` – see meetod kontrollib Firebase andmebaasist kapi staatust, kasutades selleks `SettingsController` teenust, ning muudab ka ekraanil olevate väljade väärtust kasutades meetodit `updateComputedProperties()`;

`updateComputedProperties()` – see meetod muudab ekraanil olevate väljade väärtusi vastavalt kapi staatuse väärtustele;

`handleToggleLock()` – see meetod lukustab või avab kapiukse kasutades `SettingsController`'i teenust, mis suhtleb Firebase andmebaasiga;

`onMounted()` – see on Vue.js elutsükli *hook*, mis käivitab kõik enda sees oleva, kui antud komponendi vaade avatakse, staatuse komponendi puhul sisaldab see dokumendi keha klassi muutmist, et kohanduda erinevate ekraani suurustega, meetodit `checkLockerStatus()`, et vaate avamisel kuvatakse värske kapi olek, meetodit `setInterval()`, milles kontrollitakse kapi staatust iga 10 sekundi tagant ning Vue.js elutsükli *hook*'i `onBeforeUnmount()`, kus sees komponendist väljumisel tühistatakse intervall, et kapi oleku kontroll teiste vaadete kuvamisel peatada.

- **StudentList.vue** vastab marsruudile '/users' ning on vaade, mis koosneb tabelist kus on kirjas kõik kapi autoriseeritud kasutajad, kellel on õigus kapi avamiseks (vt Lisa 7). Tabelis on pealkirjad UUID (kiipkaardi identifikaator), nimi (kasutajale määratud nimi), tüüp (teacher/student), lisatud (õiguste lisamise kuupäev), aegub (õiguste aegumise kuupäev), tegevused (valik kasutaja kustutada või andmeid muuta). Tabeli vasakul üleval nurgas on nupp uue kasutaja andmebaasi lisamiseks ning selle all on veerg, et kasutajaid valida (valiku abil saab eemaldada mitu kasutajat korraga). Tabeli veeru peal hiirega hõljudes muudab see värvi ning tabeli pealkirjadele vajutades on võimalik tabel sorteerida

vajutatud veeru pealkirja järgi. Kui kasutaja õigused on aegunud, kuvatakse aegub veerus kuupäev punaselt.

Komponendis on lisaks tabelile ka kolm vaikumisi peidetud lisaakent (addDialog, modifyDialog, removeDialog), mis avanevad samal lehel modaalsena. Lisaaken addDialog kuvatakse, kui kasutaja vajutab kasutaja lisamise nupule, modifyDialog kuvatakse kasutaja muutmise nupule vajutades ning removeDialog ilmub kinnitusena, kui tahetakse mõnda kasutajat kustutada. Kasutaja lisamise lisaaken sisaldab endas vormi kasutaja andmete sisestamiseks. Et kasutajat sisestada tuleb täita järgnevad lahtrid: täisnimi, UUID, kasutaja tüüp (vaikumisi määratud õpilane), kehtivuse lõpp (vaikumisi määratud semestri lõpu kuupäev). Igal lahtril on ka sisendi kontroll, et pahatahtlik kasutaja ei saaks süsteemi sobimatuid andmeid sisestada. Kasutaja muutmise lisaaknas on võimalik muuta kasutaja nime, tüüpi ja kehtivuse lõppu. Antud komponent sisaldab endas järgnevaid meetodeid:

openAddDialog(log) – see meetod avab kasutaja lisamise lisaakna. Kui parameeter log eksisteerib, siis tähendab see, et lisaaken on avatud teisest vaatest ViewLog ning kasutaja andmed eeltäidetakse log parameetris sisalduva UUID'ga, kui parameetrit ei eksisteeri, avatakse lisaaken ning eeltäidetud on ainult tüüp (student) ning kehtivuse lõpu kuupäev (semestri lõpp);

closeAddDialog() – see meetod sulgeb kasutaja lisamise lisaakna;

getNextDate() – see meetod leiab hetke kuupäeva arvestades käesoleva semestri lõpu kuupäeva ja tagastab selle;

openRemoveDialog(actionType, user) – see meetod avab kasutaja eemaldamise kinnitusakna, parameetriteks on actionType, mis viitab sellele kas tegemist on ühe kasutaja kustutamisega või mitme, ning user, kus on hetkel valitud kasutaja andmed, mis salvestatakse ümber muutujasse selectedUser;

closeRemoveDialog() – see meetod sulgeb kasutajate eemaldamise kinnitusakna;

handleRemoval(selectedActionType) – see meetod vaatab parameetrit selectedActionType, mille väärtus võib olla kas 'deleteSelected' või

'handleRemoveUser' ning selle põhjal otsustab kumbat funktsiooni kutsuda, kas ühe kasutaja eemaldamine või mitme;

openModifyDialog(user) – see meetod avab kasutaja muutmise lisaakna ning parameetri user abil eeltäidab mõned väljad muudetava kasutaja andmetega;

closeModifyDialog() – see meetod suleb kasutaja muutmise lisaakna;

formatTimestamp(timestamp) – see meetod vormindab ajatempli Unix formaadist Eestis kasutatavasse kuupäeva formaati (pp.kk.aaaa) ning tagastab vormindatud kuupäeva;

handleAddUser() – see meetod suhtleb UserController teenusega, et lisada Firebase andmebaasi uus kasutaja;

handleModifyUser() – see meetod kasutab UserController teenust, et muuta Firebase andmebaasis juba olemasoleva kasutaja andmeid;

getUsersAndSortData() – see meetod kasutab UserController teenust, et saada Firebase andmebaasist kasutajate nimekiri ning see sorteerida ning vormindada tabelis kuvamiseks;

onMounted() – Vue.js elutsükli *hook*, mis antud komponendi puhul sisaldab endas dokumendi keha klassi muutmist, et kohanduda erinevate ekraani suurustega, getUsersAndSortData meetodit, et vaate avamisel oleks tabel andmetega täidetud, kontrolli, kas päringu parameetrite hulgas oli 'log', et avada sellel puhul kasutajate lisamise lisaaken;

selectAllRows() – see meetod teeb kõik tabeli read aktiivseks, kui on tehtud linnuke vastavasse kasti;

sortTable() – see meetod kutsub välja tabeli sorteerimiseks getUsersAndSortData meetodi ning muudab tabeli pealkirjade vajutamisel sortField ja sortOrder parameetrite väärtusi, et tabelit nende järgi sorteerida;

deleteSelectedRows() – see meetod kasutab UserController teenust, et kustutada andmebaasist kasutajad, mis on valitud, meetod eemaldab kustutatud kasutajad ka tabelist;

`removeUser(id)` – see meetod kasutab `UserController` teenust, et kustutada andmebaasist üks kasutaja, kelle kiipkaardi identifikaator on parameetrik; `handleRemoveUser (idToRemove)` – see meetod kutsub välja `removeUser` meetodi valitud kasutaja eemaldamiseks ning tabelist kustutamiseks.

- **ViewLog.vue** vastab marsruudile `'/logs'` ning kuvab ekraanil laborikapi ajaloolise logi, kus on näha kes ja millal on kappi avanud (vt Lisa 8). Tabelis on järgnevad veerud: järjekorranumber (tabeli kirjete arvu loetlemiseks), kuupäev ja kellaaeg (millal kapp avati), sündmus (`Opened/Attempted open`), nimi (kes kapi avas; kui nimi puudub, siis vaikeväärtus „TBD“; kui avatakse veebiliidesest, on nimel järelliide „WEB“), tüüp (`student/teacher/unknown`), UUID. Tabel on sorteeritav sündmuse, kuupäeva ja kellaaaja, nime ja tüübi järgi vastavatele veerupealkirjadele vajutades. Tabeli andmeid uuendatakse iga 5 sekundi tagant. Antud komponent sisaldab järgmiseid meetodeid:

`showAddButton(log)` – see meetod vaatab saadud parameetri põhjal, kas antud logi sissekandes leitav UUID (kiipkaardi identifikaator) on juba Firebase andmebaasis `'cards'` ehk kasutajate nimekirjas olemas, kui seda UUID-d seal ei ole, siis lisab meetod tabeli viimasesse veergu vastavasse ritta nupu, mis võimaldab logist otse kasutajale õigusi lisada;

`openAddDialog(log)` – see meetod muudab enda parameetri JSON objektiks ning lisab selle päringule ning pärast seda suunab kasutaja ruuteri abil `'/users'` marsruudile, kus avaneb kasutaja lisamise lisaaken, selle meetodi kutsub välja logist kasutaja lisamise nupp;

`getLogsAndSortData()` – see meetod kasutab `LogController` teenust, et saada Firebase andmebaasist logi sissekanded ning need vormindada ja sorteerida tabelisse paigutamiseks;

`sortTable(field)` – see meetod muudab vajutatud veerupealkirja järgi muutujate `sortField` ning `sortOrder` väärtust ning kutsub meetodi `getLogsAndSortData`, mis nende parameetrite põhjal sorteerib tabeli;

`formatTimestamp(timestamp)` – see meetod vormindab Unix formaadis ajatempli meie regioonis loetavaks kuupäevaks ja kellaajaks (pp.kk.aaaa, tt:mm:ss) ja tagastab selle;

`onMounted()` – Vue.js elutsükli *hook*, mis selles komponendis sisaldab dokumendi keha klassi vahetust, et tagada sujuv kasutuskogemus erineva ekraani suurusega seadmetes, lisaks kutsutakse välja `getLogsAndSortData`, et tabel oleks vaate avamisel andmetega täidetud, sisaldab veel `setInterval` meetodit, mis kutsub `getLogsAndSortData` välja iga viie sekundi tagant, et logi andmed püsiksid värsked, viimasena on kasutatud teist elutsükli konksu `onBeforeUnmount`, kus kasutatakse meetodit `clearInterval`, et vaatest väljumisel tabeli värskete logiandmete hankimine lõpetada.

8.2.3 Ruuter

Ruuter on Vue.js sisseehitatud funktsionaalsus, mis kontrollib kõiki veebilehe marsruute ning nende vahel liikumist. Näiteks vajutades kodulehel nupule „Vaata logi“, navigeerib ruuter marsruudile `/logs` ning kasutajale avaneb logide vaade. Et seda Vue poolt pakutavat funktsionaalsust kasutada, pidi failis `router.ts` looma nimekirja soovitatavatest marsruutidest ning lisama meetodid, mis tagaks navigeerimise ainult siis, kui kasutaja on sisselogitud. Marsruutideks olid JSON objektid järgneva sisuga: `path` (marsruut), `name` (marsruudi nimi), `component` (marsruudiga seotud vaate komponent), `props`, `meta` (muu marsruudiga seonduv, näiteks brauseris kuvatav pealkiri ning kontroll, kas antud marsruut nõuab sisselogimist). Antud komponent sisaldab ühte meetodit:

`beforeEach()` – see meetod kontrollib enne igat korda, kui ruuter välja kutsutakse, kas kasutaja on sisselogitud ja omab luba sinna marsruudile minemiseks, kui kasutaja pole sisselogitud, suunatakse ta `/login` marsruudile mis avab sisselogimise vaate, kui kasutaja on sisselogitud, siis `/login` marsruudile ei pääse ja ta suunatakse kodulehele.

8.2.4 Põhirakendus

Põhirakenduse komponent `App.vue` on Vue.js puhul standardne konteiner, mille sees saavad kõik teised vaated ja komponendid toimida. See komponent sisaldab endas ka lisakomponente `NavBar` (navigatsiooniriba) ning `TitleBar` (tiitliriba). `NavBar` komponenti näidatakse põhirakenduses, kui kasutaja on sisselogitud ning `TitleBar`

komponenti, kui kasutaja on sisselogimise lehel. Põhirakenduses on ka router-view, mis võimaldab alamlehtede vahelist navigeerimist. Antud komponent sisaldab endas järgnevaid meetodeid:

`startTimer()` – see meetod seab taimeri üheks tunniks alates sellest, kui veebilehe kasutaja muutub mitteaktiivseks, kui taimer täis saab, logitakse kasutaja automaatselt veebiliidestest välja, see meetod on loodud turvalisuse kaalutlustel, et vältida lahti jäänud veebilehitseja akna kuritarvitamist, mitteaktiivsus tuvastatakse siis, kui pole tuvastatud `mousemove` (hiire kursori liikumine ekraanil) sündmust;

`resetTimer()` – see meetod taaskäivitab taimeri, taimer taaskäivitatakse `mousemove` sündmuse puhul;

`onBeforeMount` – Vue.js elutsükli *hook*, mis aktiveerub enne, kui komponent monteeritakse, ehk enne kui veebiliides kasutajale vaatevälja ilmub, antud komponendi puhul sisaldab see endas meetodit `setPersistence`, mis muudab sisselogimise veebilehitseja sessioonipõhiseks, mis tähendab seda, et teisele vaheaknale liikudes peab uuesti sisse logima, meetodit `onAuthStateChanged`, mis kontrollib Firebase autentimist kasutades kasutaja sisselogimise oleku muutumist ja vastavalt otsustab ka, kas kuvada navigatsiooniriba või tiitliriba, meetodit `startTimer`, mis käivitab mitteaktiivsuse taimeri.

8.2.5 Lisakomponendid

Lisakomponendid on antud veebiliidese puhul navigatsiooniriba (`NavBar.vue`) ning tiitliriba (`TitleBar.vue`). Navigatsiooniriba asub kodulehel ja igal alamlehel, v.a sisselogimise alamleht. Tiitliriba asub sisselogimise lehel.

- **NavBar.vue** on komponent, mis asub lehe ülemises ääres. Selle eesmärk on võimaldada kasutajal mugavalt navigeerida erinevate vaadete vahel sõltumata vaatest, kus ta parasjagu asub. Komponendi vasakus servas on laborikapi projekti jaoks loodud logo, mis kujutab laborikappi ning NFC kaarti, ning selle kõrval tekst „TARK LABORIKAPP“. Keskelt kuni parema servani kulgeb menüü, kus on toodud vaadete nimed koos kirjeldava ikooniga: Kodu (avaleht), Kasutajad (marsruut `’/users’`), Logi (marsruut `’/logs’`), Kapi seaded (marsruut `’/settings’`). Kõige parempoolsem on väljalogimise nupp, millel vajutades logitakse kasutaja

veebiliidesest välja ning suunatakse sisselogimise vaatele. Komponenti luues on arvestatud ka mobiilseadmete kasutajatega ning mobiilivaates asendub komponendi keskelt kuni paremale kulgev menüü ikooniga, kus on kolm üksteise kohal olevat kriipsu. Sellele ikoonile vajutades avaneb navigatsiooniribast allapoole rippmenüü, kus on varasemalt peidetud tavamenüü valikud. Antud komponent sisaldab järgmiseid meetodeid:

`toggleMenu()` – see meetod muudab muutuja `isMenuOpen` väärtuse vastupidiseks, et mobiilivaates menüü ikoonile vajutades rippmenüü avaneks;

`closeMenu()` – see meetod sulgeb mobiilivaates oleva rippmenüü;

`logout ()` – see meetod logib kasutaja veebiliidesest välja, meetod kutsutakse väljalogimise nupu vajutamisel.

- **TitleBar.vue** on komponent, mis on kasutusel sisselogimise lehel. Selle komponendi vajalikkus seisneb selles, et pahatahtlik kasutaja ei saaks sisse logimata veebiliidest kasutama hakata. Komponent on põhimõtteliselt koopia navigatsiooniriba komponendist, kuid ilma ühegi menüüta, et ei oleks võimalik sisselogimise lehelt muid vaateid näha. Antud komponent endas meetodeid ei sisalda.

9 Laborikapp

Käesolev peatükk kirjeldab valminud laborikappi (vt Joonis 10), sellele riistvara kinnitamist ning tööks kasutatud vahendeid. Projektis kasutatud laborikapiks on tootja AJ riiulitega laokapp, mille andmed ja mõõtmed on kättesaadavad tootja kodulehel [12].



Joonis 10. Valminud tark laborikapp.

9.1 Paigutus

Kogu valminud süsteem otsustati paigaldada kapi uksele, seda kahel põhjusel. Esiteks suudeti luua piisavalt kompaktsed ning viisakad korpused, kuhu riistvara paigutada. Seega saadi kindel olla, et kõik komponendid on kaitstud, püsivad omal kohal ning ei sega kapi tavapärasest kasutamist. Teiseks oli soov, et võimalikult vähe juhtmeid oleks veetud kapi ja ukse vahelt, kuna ukse pideva avamise ja sulgemise tõttu on see juhtmete jaoks kriitiline stressikoht. Valitud lahendusega suudetigi antud tingimus saavutada, sest kapi ja ukse vahel on vaid üks kahesoone line toitekaabel.

Süsteem koosneb viiest põhiosast (vt Lisa 2), millest esimene on toiteplokk, mis jääb kapist välja. Sealt suundub vool releesse ja toitemuundurisse, kust see liigub omakorda edasi ekraani ja lugeri korpustesse. Viimaseks põhiosaks on solenoid, mis paikneb ukse ülemisel serval ning mida juhitakse releega. Osade paigutus uksele valiti eelkõige kasutajamugavuse ja töökindluse järgi. Ekraan ja luger on paigutatud enamiku kasutajate jaoks mugavale kõrgusele ning ukse kergemaks avamiseks lisati ka ise disainitud link. Ukse sisemised kaablid otsustati vedada ukse serva tagant, et need võimalikult kaitstud ja varjatud oleks.

9.2 Modifitseerimine ja monteerimine

Kõigi komponentide kappi paigaldamiseks pidi kappi veidi modifitseerima. Esmalt pidi puurima augud ekraani korpusele, magnetandurile, lingile ning solenoidile. Solenoid, ukse link, magnetandur ning ekraani korpus kinnitati kapi külge kruvide, poltide ja mutritega. Ülejäänud komponendid kinnitati liimi või teibiga, sest nende puhul polnud kinnitamise tugevus kriitiline. Sellega ennetati rohkemate aukude puurimisega kaasnevaid tagajärgi. Kui komponendid olid paigaldatud, sai keskenduda kaablitele, millest enamus joodeti kokku ning isoleeriti, kuid strateegilistesse kohtadesse lisati ka pistikud, et komponente saaks ühe kaupa ilma jootmata eemaldada. Kaablid kinnitati teibi, kaablivitsade ning kuuma liimiga.

9.3 Kasutatud vahendid

Süsteemi edukaks paigaldamiseks kasutati järgnevaid tööriistu ja vahendeid: kruvikeerajate komplekt, näpitsad, lõikajad, viil, jootekolb koos tinaga, akutrell koos

puuridega, isekleepuvad kaablifiksaatorid, kuum liim, universaalliim, isoleerteip, 5 meetrit 2-soonelist toitekaablit, 5 meetrit signaalijuhet, poldid, mutrid, muud kinnitustarvikud. Osad vahendid hangiti ise, kuid suurem osa vahenditest saadi juhendaja kaudu.

10 Projekti analüüs ja võimalikud edasiarendused

Autoritel on veendumusel, et käesolev projekt osutus edukaks, sest täideti kõik seatud eesmärgid ning lõpptulemuseks saadi töötav süsteem, mis on funktsionaalne, töökindel ja mugav kasutada. Esimesel sprindil tegeleti ülesande püstituse, eeltöö ja ajaplaneerimisega. Järgneval sprindil telliti riistvara, seati üles arenduskeskkonnad, loodi esialgsed NFC lugeri ning solenoid-luku vahelised seosed ning konsulteeriti juhendajaga täpsemate nõuete osas.

Neile järgneva kolme sprindi jooksul keskenduti riistvara komplekteerimisele, 3D modelleerimisele ja tarkvara arendusele ning jooksvalt ka dokumenteerimisele. Esialgu oli plaan jõuda arendusega valmis kaheteistkümnendaks nädalaks, kuid tegelikkuses lõppes suurem arendustegevus kolmeteistkümnenda nädala lõpus ning sai alustada süsteemi kappi paigaldamisega ja põhjalikumalt aruande kirjutamisega. Viieteistkümnenda nädala lõpuks oli süsteem edukalt valmis ning paigaldatud.

Meeskonnatöö sujus projekti jooksul hästi ning sellega seoses suuremaid tagasilööke ei tekkinud. Ainsaks suuremaks probleemiks oli ühise aja leidmine. Väiksemaks probleemiks oli ka see, et laboris käimised ja kapile ligipääsemine oli vaja eelnevalt kooskõlastada, mis tekitas mitmeid kordi viivitusi. Tööjaotust projektiga alustades ei seatud, kuid arenduse käigus loksus see paika. Aruande ja sprindiesitlused koostati ühiselt ning hinnang on, et kõigi panused olid projekti jooksul võrdsed.

Autorite arvates osutus projekti teema valik positiivseks, sest see oli kõigi jaoks huvitav, mis omakorda motiveeris arendusele aega panustama. Projekti juures oli suureks lisandväärtuseks veel see, et arenduse käigus õpiti palju kasutatud teekide ja C++ programmeerimiskeele, veebiarenduse, andmebaaside ja 3D-modelleerimise kohta ning saadi hea terviklik IoT tootearenduse kogemus. Valminud süsteemil on ka võimalikke edasiarendusi, nagu ukse automaatne lahtilükkamine, ning ka võimalus koostatud dokumentatsiooni ja lahenduse põhjal aina rohkem laborikappe nutikaks teha.

11 Kokkuvõte

Antud töö eesmärgiks oli arvutite ja süsteemide projekti aine raames luua olemasolevale laborikapile Tallinna Tehnikaülikooli laboris nutikas lukustussüsteem, mis hõlbustaks ja muudaks mugavamaks õppejõu ja tudengite jaoks laborivahendite hoiustamist. Meile seatud ülesandeks oli kapi modifitseerimine viisil, et kapp oleks lukustatav ning avatav NFC kaardiga (ISIC kaart) ning et süsteem hõlmaks kaartide õiguste haldamist ja kapi avamise ajalugu.

Projekti käigus valmis toimiv lahendus, mis täidab oma eesmärgi. Tagasiside jaoks on laborikapile lisatud puutetundlik ekraan, LED ja sumisti ning veebiliideses saab hallata õiguseid, vaadata logi, kappi avada ning näha selle staatust. Seega koosneb süsteem kapis asuvast riistvaralisest lahendusest koos tarkvaraga ning veebimajutuses paiknevast veebiliidesest. Lisaks on kapis asuv riistvara kaitstud kompaksete 3D-prinditud korpustega.

Lõpptulemus sai valmis viie kahe nädalase sprindi jooksul, mille käigus saime meeskonnana mitmeid kordi kokku, konsulteerisime juhendajaga ning arendasime süsteemi erinevaid osasid ka iseseisvalt. Arendusel olid töövahenditeks tekstiredaktor VSCode, C ja C++ programmeerimiskeeled ning PlatformIO arenduskeskkond, veebilehe jaoks Vue.js raamistik ja DevTools arendustööriistad, 3D-modelleerimiseks SolidWorks. Rakenduse versioonihalduseks ning sujuvaks arenduseks kasutasime Github'i ning veebimajutuseks ja andmebaasiks Firebase pilveteenust.

Repositoorium: <https://github.com/siimtishler/kapp>

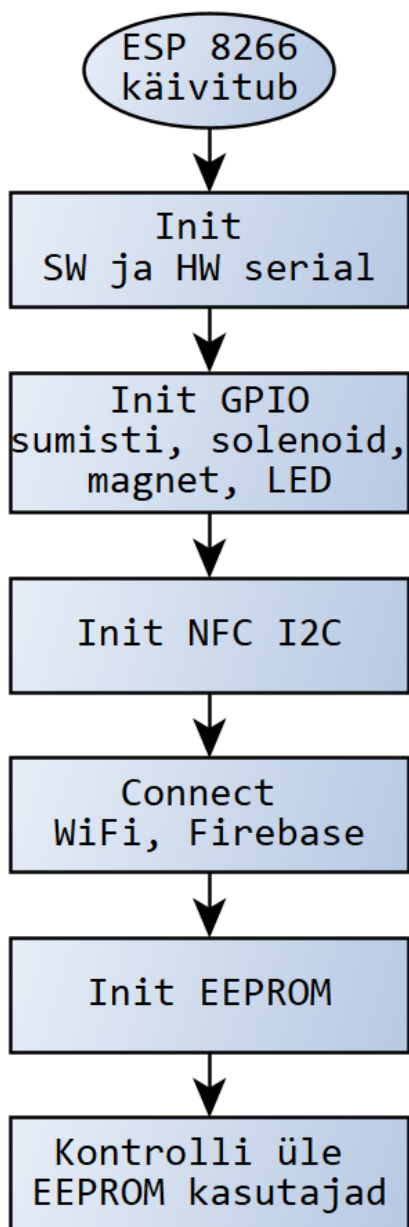
Valminud laborikappi on võimalik näha Tallinna Tehnikaülikooli robotika demokeskuses, ruumis U05B-208.

Kasutatud allikad

- [1] „platformio.ini” (Project Configuration File),“ PlatformIO Labs, [Võrgumaterjal]. Available: <https://docs.platformio.org/en/stable/projectconf/index.html>. [Kasutatud 4. detsember 2023].
- [2] MSEdgeTeam, captainbrosset, pankajparashar, mikehoffms, JasonAndrewWriter ja ryanborMSFT, „Overview of DevTools,” Microsoft, 12. juuli 2023. [Võrgumaterjal]. Available: <https://learn.microsoft.com/en-us/microsoft-edge/devtools-guide-chromium/overview>. [Kasutatud 10. detsember 2023].
- [3] „Adafruit PN532 GitHub repositoorium v1.3.2,” [Võrgumaterjal]. Available: <https://github.com/adafruit/Adafruit-PN532/releases/tag/1.3.2>. [Kasutatud 20. oktoober 2023].
- [4] I. Grokhotkov, „ESP8266 Arduino Core documentation Libraries EEPROM,” [Võrgumaterjal]. Available: <https://arduino-esp8266.readthedocs.io/en/latest/libraries.html#eeprom>. [Kasutatud 28. november 2023].
- [5] „ESP tarkvara jadaliides GitHub repositoorium v8.1.0,” [Võrgumaterjal]. Available: <https://github.com/plerup/espsoftwareserial/releases/tag/8.1.0>. [Kasutatud 19. detsember 2023].
- [6] L. Bies, „RS232 Specifications and standard,” August 2021. [Võrgumaterjal]. Available: <https://www.lammertbies.nl/comm/info/rs-232-specs>. [Kasutatud 17. detsember 2023].
- [7] H. L. D. Z. Yichen Huang, „Analysis of the baud rate of the UART to affects the data,” *Highlights in Science, Engineering and Technology*, kd. 61, pp. 200-205, 2023.
- [8] „Arduino teegi jadaliidese funktsiooni Serial.Begin dokumentatsioon,” [Võrgumaterjal]. Available: <https://www.arduino.cc/reference/en/language/functions/communication/serial/begin/>. [Kasutatud 18. detsember 2023].
- [9] „TalTech akadeemiline kalender,” [Võrgumaterjal]. Available: <https://taltech.ee/akadeemiline-kalender>. [Kasutatud 13. detsember 2023].
- [10] Academind, „Angular vs React vs Vue [2020 Update],“ 19. märts 2020. [Võrgumaterjal]. Available: <https://www.youtube.com/watch?v=1YWYWyX04JI&t=126s>. [Kasutatud 15. detsember 2023].
- [11] Google, „API Reference,” 8. august 2023. [Võrgumaterjal]. Available: <https://firebase.google.com/docs/reference/js>. [Kasutatud 15. detsember 2023].
- [12] AJ Tooted AS, „Laokapid,” [Võrgumaterjal]. Available: <https://www.ajtooted.ee/ladu-ja-toostus/ladustamine/laokapid>. [Kasutatud 15. detsember 2023].
- [13] „Adafruit BusIO GitHub repositoorium v1.14.5,” [Võrgumaterjal]. Available: https://github.com/adafruit/Adafruit_BusIO/releases/tag/1.14.5. [Kasutatud 18. detsember 2023].

- [14] „Arduino JSON GitHub repositoorium v6.21.3,“ [Võrgumaterjal]. Available: <https://github.com/bblanchon/ArduinoJson/releases/tag/v6.21.3>. [Kasutatud 18. detsember 2023].
- [15] „Firebase ESP klient GitHub repositoorium v4.4.8,“ [Võrgumaterjal]. Available: <https://github.com/mobizt/Firebase-ESP-Client/releases/tag/v4.4.8>. [Kasutatud 18. detsember 2023].
- [16] I. Grokhotkov, „ESP8266 Arduino Core documentation ESP8266WiFi library,“ [Võrgumaterjal]. Available: <https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html#esp8266wifi-library>. [Kasutatud 28. november 2023].

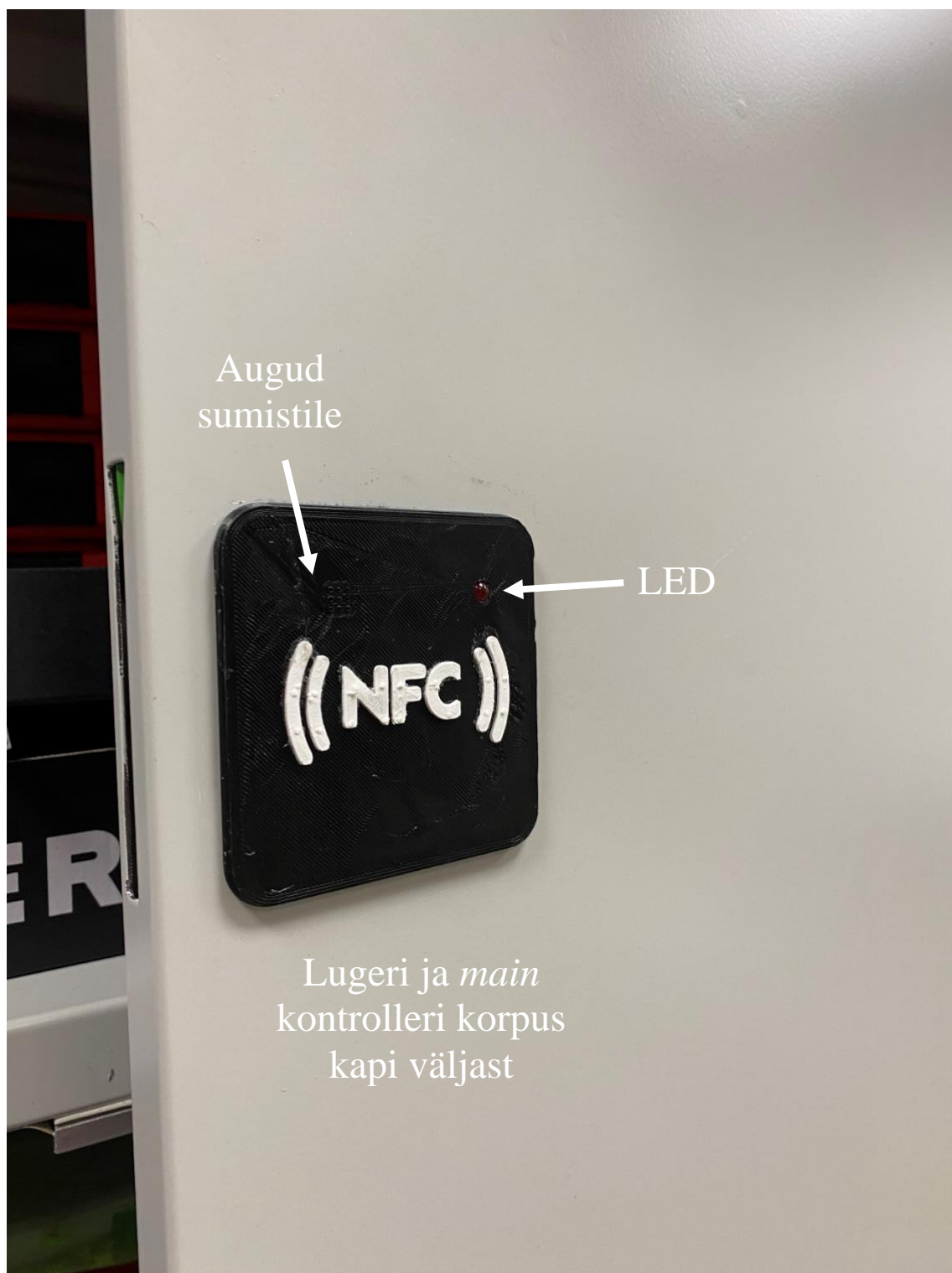
Lisa 1. *Main* kontrolleri käivitamise plokk skeem.



Lisa 2. Pildid kapi komponentidest.





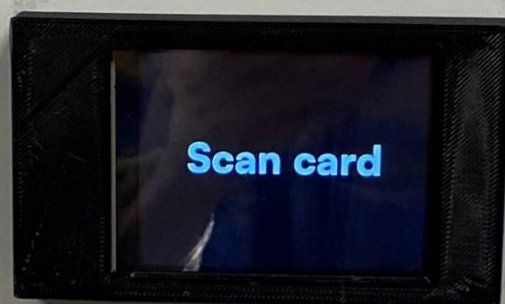




Ekraan korpuses

Link

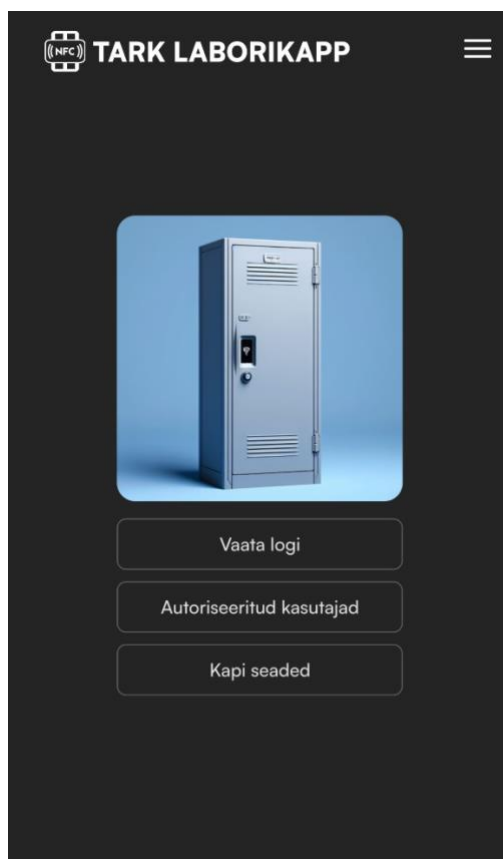
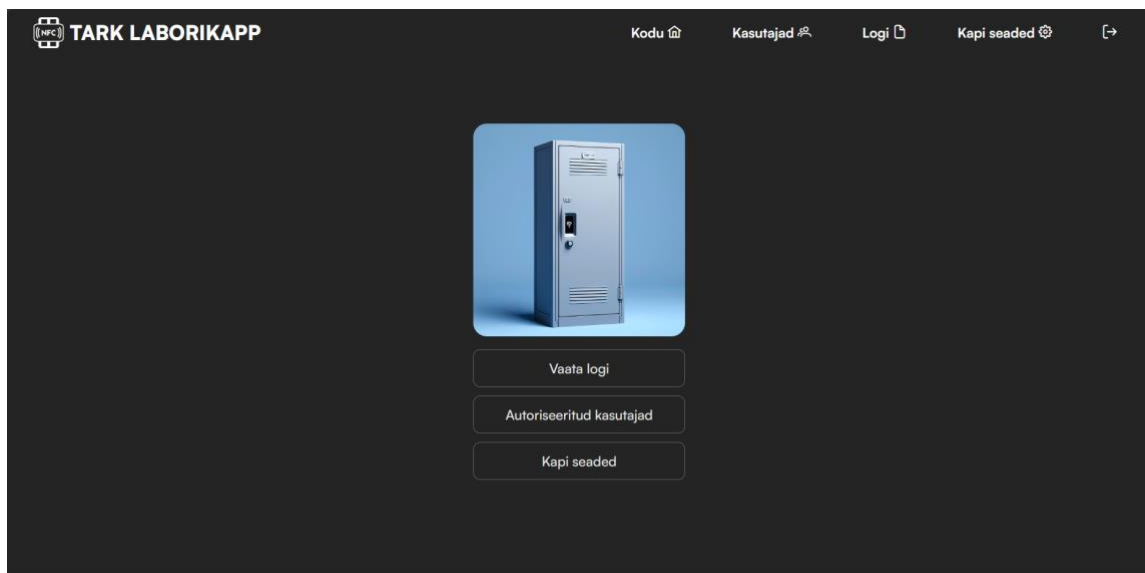
Ekraan korpuses



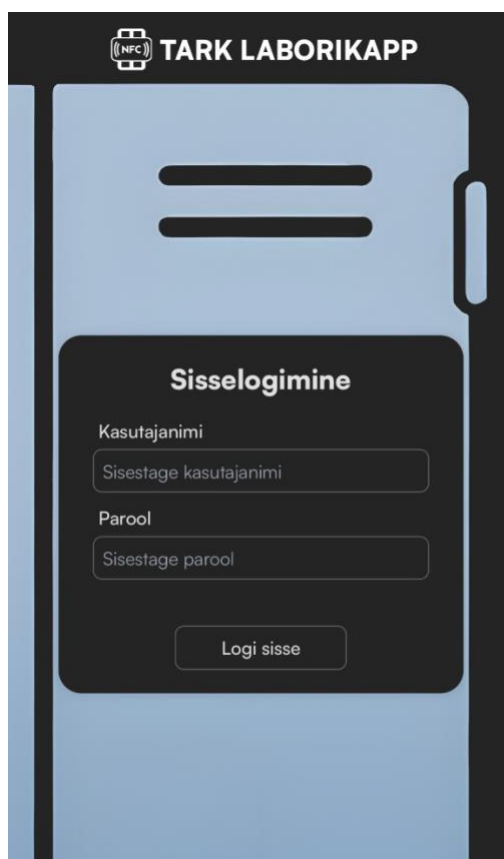
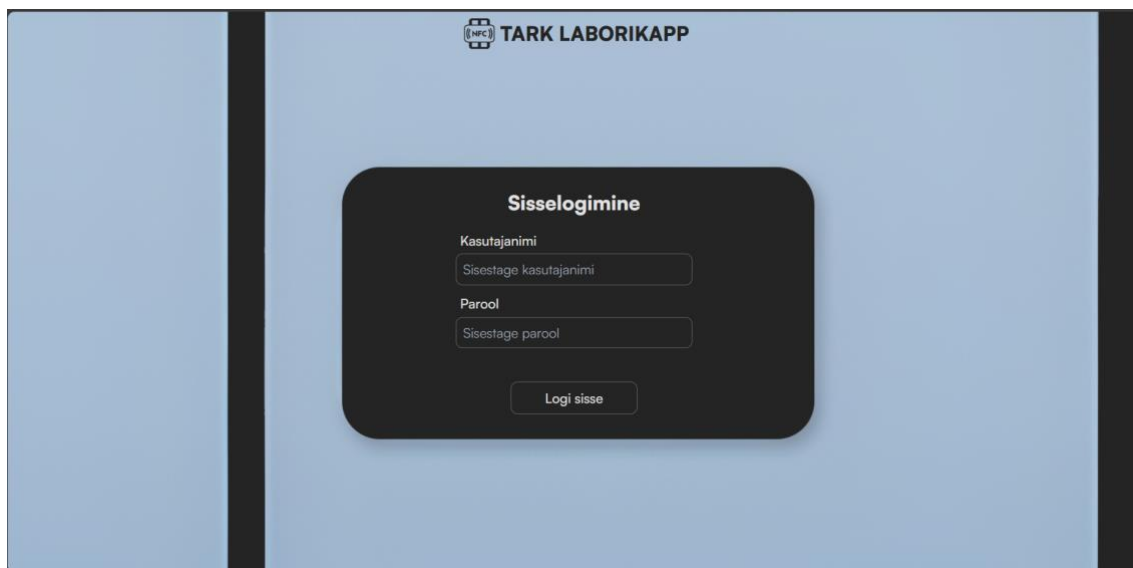
Lisa 3. Veebiliidese vealehe vaade.



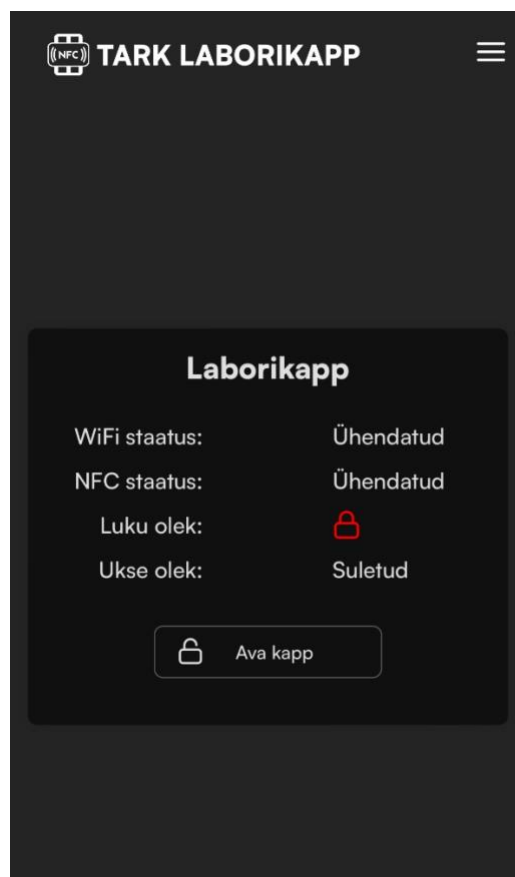
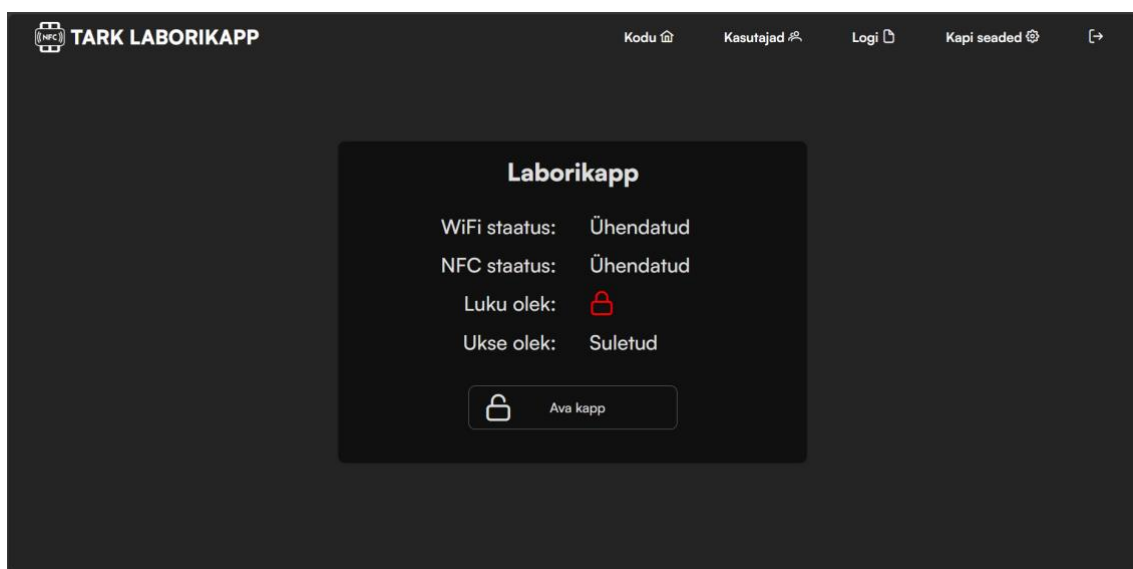
Lisa 4. Veebiliidese avalehe vaade.





Lisa 5. Veebiliidese sisselogimisvaade.




Lisa 6. Veebiliidese kapi staatuse vaade.



Lisa 7. Veebiliidese kasutajate vaade.

TARK LABORIKAPP		Kodu					Kasutajad	Logi	Kapi seaded	[→]
[Avatar]		Autoriseeritud kasutajad								
	UUID	Nimi	Tüüp	Lisatud:	Aegub:	Tegevused				
<input type="checkbox"/>	05 8B 1E 9E 6A 62	admin	teacher	18.12.2023	21.01.2024	 				


Lisa 8. Veebiliidese logi vaade.

 TARK LABORIKAPP

KoduKasutajadLogiKapi seaded

Logi

#	Kuupäev ja kellaaeg	Sündmus	Nimi	Tüüp	UUID
37	18.12.2023, 23:44:31	Opened	peeter WEB	teacher	-
38	18.12.2023, 21:38:12	Opened	peeter WEB	teacher	-
39	18.12.2023, 18:37:52	Opened	kasutaja	teacher	97 F0 77 63
40	18.12.2023, 17:37:42	Attempted open	TBD	unknown	8D B9 4E B3
41	18.12.2023, 17:35:14	Opened	peeter WEB	teacher	-
42	18.12.2023, 17:34:58	Opened	admin	teacher	05 8B 1E 9E 6A 62 00
43	18.12.2023, 17:34:30	Attempted open	TBD	unknown	08 B8 6A BA
44	18.12.2023, 17:25:12	Opened	kasutaja	teacher	97 F0 77 63
45	18.12.2023, 17:24:30	Opened	kasutaja	teacher	97 F0 77 63
46	18.12.2023, 17:24:07	Opened	kasutaja	student	97 F0 77 63
47	18.12.2023, 17:20:55	Opened	kasutaja	student	97 F0 77 63
48	18.12.2023, 17:20:42	Opened	kasutaja	student	97 F0 77 63
49	18.12.2023, 17:20:07	Attempted open	TBD	unknown	08 B8 6A BA

 TARK LABORIKAPP

Logi

#	Kuupäev ja kellaaeg	Sündmus
1	20.12.2023, 18:28:22	Opened
2	20.12.2023, 17:30:04	Opened
3	19.12.2023, 13:35:43	Opened
4	19.12.2023, 13:35:39	Opened
5	19.12.2023, 13:35:34	Opened
6	19.12.2023, 13:35:27	Opened
7	19.12.2023, 12:51:25	Opened
8	19.12.2023, 12:51:16	Opened
9	19.12.2023, 12:51:14	Opened
10	19.12.2023, 12:51:12	Opened
11	19.12.2023, 12:51:11	Opened
12	19.12.2023, 12:51:10	Opened
13	19.12.2023, 12:51:07	Opened
14	19.12.2023, 12:51:00	Opened
15	19.12.2023, 02:36:35	Opened
16	19.12.2023, 02:36:30	Opened
17	19.12.2023, 02:36:24	Opened