TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Otto Kaarel Altroff 193314IADB

eSIM toe arendamine Super kõnekaardi mobiilirakendusele

Bakalaureusetöö

Juhendaja: German Mumma
Magistrikraad
Kaasjuhendaja: Meelis Luiks
Magistrikraad

Tallinn 2023

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö 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.

Autor: Otto Kaarel Altroff

15.05.2023

Annotatsioon

Käesoleva bakalaurusetöö peamine eesmärk on täiendada Super kõnekaardi mobiilirakendust, et see toetaks seadmetesse eSIM'ide paigaldamist. Lahendus võimaldab kliendil liituda Super kõnekaardi teenusega ilma füüsilist SIM-kaarti soetamata.

Antud lõputööst leiab ülevaate eSIM'i arhitektuurist, lahendusmudelist ning alternatiivsetest tehnoloogiatest. Samuti on määratletud loodava rakenduse mooduli nõuded ning kirjeldatud kasutatavate tehnoloogiate valikuid ja rakenduse arenduskäiku.

Töö käigus valmis mitmekihilisel klient-server arhitektuuril põhinev rakendus, mille klientrakendus on arendatud React Native ja TypeScript tööriistadega ning serverrakendus Javal põhineval Spring Boot raamistikul. Rakenduse kaudu saab klient läbida teenusega liitumise teekonna ning selle tulemusena laadida enda seadmesse alla Super kõnekaardi eSIM.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 44 leheküljel, 6 peatükki, 10 joonist, 1 tabelit.

Abstract

Development of eSIM Support for Super Calling Card Mobile Application

The primary objective of this bachelor's thesis is to enhance the existing Super Calling Card mobile application by enabling support for eSIM installation on devices. The solution provides customers with the opportunity to join the Super Calling Card service without the need to purchase a physical SIM card.

Currently, in order to use the Super services, a physical SIM card is required. However, an alternative system that allows for use of these services without the need for a SIM card already exists.

This thesis provides an overview of eSIM architecture, solution models and alternative options to eSIM. It also defines the user requirements for the developed application module and describes the methodology, selected technologies and patterns used in the development process.

The outcome of the thesis is a three-tier client-server application. The client application is a mobile application developed using React Native and TypeScript, while the server application is based on the Spring Boot framework in Java. Through the mobile application, users can sign up and subsequently download the Super Calling Card eSIM to their device.

The thesis is in Estonian and contains 44 pages of text, 6 chapters, 10 figures, 1 table.

Lühendite ja mõistete sõnastik

acceptance criteria	Nõuetele vastavuse kriteerium
AES	*Advanced Encryption Standard*, krüpteerimisstandard
API	*Application Programming Interface*, rakendusliides
BLL	*Business Logic Layer*, rakenduse äriloogikakiht
build	Rakenduse kooste – tarkvara kokkupanek ja pakendamine, et luua kasutatav rakendusfail
CI	*Certificate Issuer*, sertifikaatide väljastaja roll, mida GSMA on väljastanud loetud ettevõtetele
consumer solution	Tavatarbija mudel
CSV	*Comma-separated values*, komaga eraldatud väärtustega tekstifail
DAL	*Data Access Layer*, rakenduse andmetele kättesaadavuse kiht
DI	*Dependency Injection*, sõltuvuste süstimine
DOM	*Document Object Model*, dokumentidega suhtlemise liides
DTO	*Data Transfer Object*, andmeedastusobjekt
ECC	*Elliptic curve*, krüptograafia meetod, mis põhineb piiratud väljade kohal olevate elliptiliste kõverate algebralisel struktuuril.
eesrakendus	Kliendi rakenduse osa
EID	*Embedded Identity Document*, 32-kohaline unikaalne identifikaator seadme eUICC kiibi jaoks
entitlement	volitus
epic	suur kasutajalugu, mis kirjeldab nõuet, mida hakatakse arendama
eSIM	*embedded-SIM*, sängitatud SIM
ETSI	European Telecommunications Standards Institute
eUICC	*Embedded Universal Integrated Circuit Card*, sängitatud kiipkaart
EUM	*eUICC Manufacturer*, eUICC tootja
exception	erind – eriolukord, mis programmi töö jooksul ilmneb
FF	*Form Factor*, mõõt

GSMA	*Global System for Mobile Communications Association*
header	HTTP päringu päis
hook	React uuendus, mis võimaldab kasutada Reacti funktsioone klasside kirjutamiseta
HTTP	*Hypertext Transfer Protocol*, hüperteksti edastusprotokoll
ICCID	*Integrated Circuit Card Identification Number*, unikaalne tunnusnumber UICC kiipkaardi jaoks
IMEI	*International Mobile Equipment Identity*, seadme unikaalne identifikaator
IMSI	*International Mobile Subscriber Identity*, unikaalne tunnusnumber operaatori profiili jaoks
Inside Out	Seestpoolt väljapoole – TDD metoodika lähenemisvõte
IoC	*Inversion of Control*, tarkvaraarenduse muster, mille eesmärk on luua lõdvalt ühendatud klassid, mida on võimalik kergevaevaliselt asendada, laiendada ning testida
IoT	*The Internet of Things*, asjade internet - erinevate seadmete kogum, mis on võrku ühendatud ning mis täidavad seal kindlat ülesannet
iSE	*Integrated Secure Element*, integreeritud turvaelement
iSIM	*Integrated-SIM*, integreeritud SIM
JSON	*JavaScript Object Notation*, andmevahetusformaat
JSX	*JavaScript XML*, JavaScripti süntaksilaiend XML kasutamiseks
ligipääsutunnus	*access token* – pikk stohhastilisel viisil koostatud praktiliselt võltsimatu sõne, mida kasutatakse veebiressursile ligi pääsemiseks lubatud isikutele
LPA	*Local Profile Assistant*, moodul, mis vastutab krüpteeritud profiilide allalaadimise eest eUICC-le
LPWAN	*Low-power wide-area network*, vähese energiatarbe ja laia ulatusega traadita sidevõrk, mis võimaldab seadmetel andmeid edastada vähese energiakuluga
LTE-M	*Long-Term Evolution Machine Type Communication*, LPWAN sidevõrk M2M ja IoT tarbeks
M2M	*machine-to-machine*, masinatevaheline suhtlus
MCC	*Mobile Country Code*, riigi tunnuskood
MFF2	*machine-to-machine form factor*, SIM formaat M2M kiibi jaoks
MNC	*Mobile Network Code*, võrgukood
mocking	testimise simuleerimine

MSIN	*Mobile Subscriber Identification Number*, unikaalne seerianumber abonendi jaoks
n-tier architecture pattern	Mitmekihiline arhitektuur - tarkvaraarenduse arhitektuurimuster
NB-IoT	*Narrowband-Internet of Things*, LPWAN sidevõrk asjade interneti seadmete jaoks
NIST SP800-57	*National Institute of Standards and Technology Special Publication 800-57*
nuSIM	„*Nu*“ – „uus“ (saksa) SIM, mis on loodud spetsiaalselt LPWAN võrgutehnoloogiate jaoks
OTA	*Over the Air*, operaatorprofiilide traadita valmendamise kontseptsioon
Outside In	Väljastpoolt sissepoole – TDD metoodika lähenemisvõte
PKI	*Public Key Infrastructure*, avaliku võtme taristu – süsteem, võtab kokku erinevad turvaaspektid, mis seotud sõnumite turvalise ülekandmisega
provisioning profile	rakenduse valmendamise profiilifail
PSK	*Pre-Shared Key*, eeljagatud võti – krüptimismeetod
pull model	Mudel, mille kohaselt algatab lõppkasutaja operaatoriprofiili eUICC-le paigaldamise
push model	Mudel, mille kohaselt algatab server operaatoriprofiili eUICC-le paigaldamise
QR-kood	*Quick Response*, kahemõõtmeline ribakood
Red-Green-Refactor	„Punane-Roheline-Refaktoreerimine“ – TDD metoodika tsükkel
Remote SIM Provisioning	SIM valmendamise kaughaldus
REST	*Representational state transfer*, veebiteenusega suhtlemise liidese arhitektuur
RNSCM	*react-native-sim-cards-manager*, React Native kolmanda osapoole teek, mis pakub SIM kaartide haldust
Root Certificate	Põhisertifikaat
RSA	*Rivest-Shamir-Adleman* – avaliku võtmega krüpteerimise meetod, mis võimaldab nii andmete krüpteerimist kui ka autentimist
SCP	*Secure Channel Protocol*, turbekanali reeglistik
Security Domain	turbedomeen
separation of concerns	tarkvaraarenduse muster, mille kohaselt on igal kihil on rakenduses täita oma kindel piiritletud roll ja vastutus
SHA	*Secure Hash Algorithms*, turvalised räsialgoritmid – räsialgoritmide seeria, mida kasutatakse sõnumite töötlemiseks

	pseudosuvalise iseloomuga ühepikkuste väljundite koostamiseks, kusjuures samale sisendile vastab alati sama väljund
SIM	*Subscriber Identity Module*, abonendi identiteedi moodul
single responsibility principle	Üksikvastutuse printsiip - tarkvaraarenduse muster, mille kohaselt igal kooditükil peab olema vaid üks konkreetne ülesanne, mida see täidab.
SM-DP+	*Subscription Manager Data Preparation* Plus, komponent, mis valmistab ette, hoiustab ja turvab operaatori profiili andmeid.
SM-DS	*Subscription Management Discovery Service*, komponent, mida kasutatakse LPA teavitamiseks, kui operaatori profiil on eUICC-sse allalaadimiseks saadaval
SM-SR	*Subscription Manager Secure Routing*, komponent, mis vastutab profiilide staatuse haldamise eest
SMSC	*Short Message Service Center*, lühisõnumite võrgukeskus
SOAP	*Simple Object Access Protocol*, arvutivõrkudes kasutatav protokoll, millega veebiteenused vahetavad omavahel struktuurseid andmeid
stubbing	simuleeritud funktsionaalsuse loomine testimise jaoks
subscription	abonement
tagarakendus	serveripoolne loogika, mis juhib erinevaid rakendusi nende taustal. Enamasti hõlmab tagarakendus andmebaasi, serverit ja rakenduse koodi.
TDD	*Test Driven Development*, testidel põhinev arendus
TSX	*TypeScript XML*, TypeScripti süntaksilaiend XML kasutamiseks
UICC	*Universal Integrated Circuit Card*, kiipkaart
URL	*Uniform Resource Locator*, veebiaadress
valmendamine	Andmete töötlus edasiseks kasutamiseks
XML	*Extensible Mark-Up Language*, laiendatav märgistuskeel – suvaliste andmete nii inim- kui ka masinloetaval kujul struktureerimiseks mõeldud märgistuskeel
YAGNI	*„You Aren't Gonna Need It“*, liiasuse vältimise printsiip

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Kuna traditsiooniline SIM-kaart pole oma mitmekümne aastase eluea jooksul tehniliselt ega vormiliselt märkimisväärselt muutunud, seab see piirangu kiiresti arenevale IoT (ingl *Internet of Things*) ehk asjade interneti valdkonnale. Seetõttu on aktuaalseks muutunud eSIM tehnoloogia, mis võimaldab valmendada mobiilsideoperaatorite SIM-profiile ilma otsese füüsilise ligipääsuta SIM-kaartidele. Sealjuures on märgata trendi, kus traditsiooniline füüsiline SIM-kaart on asendumas eSIM'iga. Sellele viitab näiteks 2022.aastal turule tulnud iPhone 14 seeria nutitelefon, mida on mudeli tootja Apple Põhja-Ameerikas turustanud ilma füüsilise SIM-kaardi pesata, suunates teed eSIM'i laiemale levikule.

Super on Telia lepinguvaba kõnekaarditeenus. 2017.aastal valminud iseteenindusveebi kui ka mobiilirakenduse abil saavad kasutajad hallata enda numbreid, laadida kõnekaardile raha, osta pakette ning kontrollida kõne, sõnumite ja internetimahu jääki.

Super kõnekaardi teenusel puudub eSIM tugi. Käesolevaga peab klient teenusega liitumiseks ostma müügipunktist füüsilise SIM-kaardi, kuigi ettevõttel on juba olemas tarkvaraline lahendus, kuidas klient saaks hakata teenuseid kasutama ka füüsilist SIM-kaarti omamata.

Käesoleva töö eesmärgiks on analüüsida eSIM tehnoloogiat ning lisada olemasolevale mobiilirakendusele Super eSIM paigaldamise funktsionaalsus. Lahendus võimaldaks kliendil liituda Super kõnekaardi teenusega ilma füüsilist SIM-kaarti soetamata. Lahendus oleks kasutatav kõikidele aktiivsetele Super kõnekaardi mobiilirakenduse klientidele Eestis, kelle mobiilsel seadmel on eSIM tugi.

Töö on jaotatud viieks peatükiks. Esimeses peatükis antakse ülevaade tööst ja püstitatakse selle eesmärk. Teises peatükis on antud ülevaade eSIM tehnoloogiast, eSIM lahendusmudelist, kirjeldatud eSIM turvalisuse aspekte ning võrreldud eSIM'i alternatiivsete tehnoloogiatega. Lisaks on lahti seletatud probleemi olemus ning välja on toodud ettevõtte põhjused arenduse vajalikkusele. Kolmandas peatükis analüüsib autor

planeeritavat rakenduse moodulit. Kirjeldatud on kasutajanõuded ja arenduses kasutatavad tehnoloogiad ja printsiibid. Neljandas peatükis kirjeldab autor töö raames valminud rakenduse mooduli teostust. Viiendas peatükis analüüsib autor tehtud töö tulemusi. Samuti on loetletud edasiarenduse võimalusi, mida rakendusele tulevikus teha.

2 Taust

2.1 Lähtetingimused

Töö kirjutamiseks ja projekti teostamiseks leiduvad mõned tehnilised ja õiguslikud eeltingimused. Käesoleva töö autor töötab töö kirjutamise hetkel arendajana tarkvaraarenduse ettevõttes Datanor OÜ, kes on Telia AS arenduspartner.

Autor arendab kavandatavat funktsionaalsust juba olemasolevale mobiilirakendusele juurde. Seega on arenduseks võimalik kasutada olemasoleva rakenduse kihte (andmebaas, teenuskiht, esitluskiht) ja seadistust. Lisaks on arenduseks ette määratud ka kasutatavad tehnoloogiad ja raamistikud, mille abil on ülejäänud rakendus juba varasemalt üles ehitatud.

Autoril on võimalik rakenduse analüüsiks kasutada klientettevõtte püstitatud kasutajanõudeid. Autoril on võimalik rakenduse arenduseks kasutada klientettevõtte partnerite varasemalt arendatud rakendusi ja tugiteenuseid, millega loodav rakenduse moodul hakkaks suhtlema. Samuti on arenduse eelduseks ligipääs klientettevõtte loodud disainijoonistele Super mobiilirakenduse jaoks, mille abil mobiilirakendust kujundada. Konfidentsiaalsuse huvides on käesolevas töös kujutatud programmikoodi näidetes ja süsteemikirjeldustes ümber nimetatud kõikide API'de, klasside, muutujate ja ligipääsupunktide nimed.

Arenduse teostamise eelduseks on ka ligipääs Apple'i rakendusliidese dokumentatsioonile, mis kirjeldab eSIM'i seadistust iOS platvormil. Autorile on väljastatud vastava dokumentatsiooni lugemisõigus konfidentsiaalsuslepingu alusel.

2.2 Metoodika

Käesoleva töö peamiseks tulemiks on olemasolevale Super mobiilirakendusele arendatav lisafunktsionaalsus. Valminud rakenduse moodulil on olemas teenuskiht ja esitluskiht. Tööde hulka kuuluvad rakenduse planeerimine, analüüs, teostamine ja testimine.

Autor on valinud tagarakenduse arendamiseks testimisel põhineva arenduse (ingl *Test-Driven Development*, lüh TDD) mustri. TDD on iteratiivne tarkvaraarenduse metoodika, mille põhimõte näeb ette tarkvara nõuete põhjal testide kava koostamist enne vastavate omaduste tarbeks programmikoodi valmis kirjutamist. Uue funktsionaalsuse ehitamisel näeb TDD metoodika ette tsüklilise kava järgimist, mida kutsutakse ka „Punane-Roheline-Refaktoreerimine" tsükliks (ingl *„Red-Green-Refactor" cycle*) [1]:

1. Kirjutada test
2. Käitada test ja veenduda, et test kukub läbi („punane")
3. Kirjutada täpselt nii palju koodi, et see läbiks testi („roheline")
4. Refaktoreerida esialgu kirjutatud koodi jälgides jooksvalt, et testid endiselt läbiksid („refaktoreerimine")
5. Korrata tsüklit

TDD metoodika aitab lisaks kirjutatava koodi korrektsuse jälgimisele kaasa ka koodi disaini silumise juures. Kuna funktsionaalsust kattev test kirjutatakse enne funktsionaalset koodi ennast, dikteerib koodi disaini justnimelt test. Seetõttu soosib TDD kergesti testitavate ja lõdvalt seotud meetodite koostamist, mida on kerge hallata ning mille funktsionaalsusest on kerge aru saada. TDD metoodika puhul eristatakse kahte peamist lähemist: „seestpoolt väljapoole" ja „väljaspoolt sissepoole" [1]:

- „Seestpoolt väljapoole" (ingl *Inside Out*) lähenemine keskendub tulemustele. Testimine algab väikseimatest ühikutest ning koodi üldine arhitektuur loob end loomulikul moel. Seda lähenemist on algajal kergem õppida ja omandada, sest testimise simuleerimise (ingl *mocking*) vajadus on väike. Koodi lõplik disain ilmneb sammus „refaktoreerimine", mis paraku võib omaette protseduurina paisuda mahukaks ülesandeks.
- „Väljaspoolt sissepoole" (ingl *Outside In*) lähenemine eeldab lähtumist kasutaja käitumisest. Testimine algab koodi „kõige välimisest" kihist ning koodi detailne käitumine ilmneb töö käigus. See lähenemine eeldab rohket väliste sõltuvuste simuleeritud funktsionaalsuse loomist (ingl *stubbing*), mistõttu eeldab selle omandamine pikaajalisemat programmeerimise kogemust. Seevastu aitab see lähenemine järgida üldist äriloogikat väga täpsel tasemel. Koodi disainimine toimub erinevalt Inside Out lähenemisest juba „punases" sammus.

Autor valis käesoleva arenduse jaoks „väljastpoolt sissepoole“ testidel põhineva arenduse metoodika ning järgis seda programmikoodi kirjutamises. Peatükis 4.3 on toodud näide TDD metoodika kasutamise kohta käesoleva arenduse protsessis.

2.3 SIM kaartide ülevaade

2.3.1 SIM programmiüksus

SIM (ingl *Subscriber Identity Module*) on programmiüksus, mis on vajalik seadme ja operaatori vaheliseks suhtluseks. See sisaldab võrgusuhtluse jaoks vajalikke abonendi kasutajatunnuseid, nagu ICCID, IMSI, ligipääsutunnust, kuid ka operaatori tunnuseid, näiteks SMSC (ingl *Short Message Service Center*) ehk lühisõnumite võrgukeskuse aadresse [2].

ICCID (ingl *Integrated Circuit Card Identification Number*) on unikaalne tunnusnumber UICC (ingl *Universal Integrated Circuit Card*) kiipkaardi jaoks. Mobiilside operaatori iga väljastatud profiil on seotud kindla ICCID ja IMSI numbriga [2].

IMSI (ingl *International Mobile Subscriber Identity*) on unikaalne tunnusnumber operaatori profiili jaoks. Levinum IMSI koosneb 15 numbrist, kus 3 esimest tähistavad riigi tunnuskoodi (ingl *Mobile Country Code*, lüh MCC), järgmised 2 võrgukoodi (ingl *Mobile Network Code*, lüh MNC) ning kõik ülejäänud abonendi identifikaatorit (ingl *Mobile Subscriber Identification Number*, MSIN) [2].

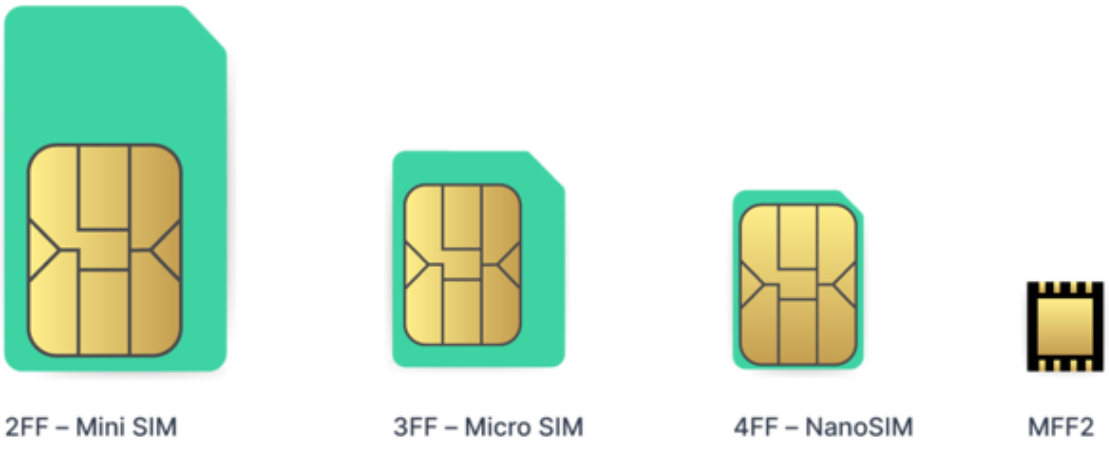
Seoses uue operaatorprofiilide traadita (ingl *Over the Air*, lüh OTA) valmendamise kontseptsiooniga eSIM jaoks tekkis vajadus uue tunnuse – **EID** (ingl *Embedded Identity Document*) – järele. EID on unikaalne identifikaator, mille väärtus koosneb 32 numbrist, mis sisaldab tootja-, riigi- ja väljastaja tunnuseid ning kontrollnumbreid. Kui ICCID ja IMSI väärtused on seotud operaatoriprofiiliga, siis EID identifitseerib füüsilist eSIM kiipi ennast. [2]

2.3.2 SIM-kaartide formaadid

SIM-kaardi formaat on aegade jooksul järk-järgult muutunud. Joonis 1 kujutab SIM-kaardi suuruse võrdlust. Esimesed täissuuruses 1FF tüüpi SIM-kaardid olid krediitkaardi mõõtu (85.60 mm × 53.98 mm × 0.76 mm). Alates 1996.aastast võeti kaustusele Mini-SIM (2FF), mis oli tagasiühilduv 1FF tüüpi SIM-kaardi pesadega, sest koos plastikust

lahtimurtava ümbrisega oli sellel samasugune kiibi paigutus. ETSI tutvustas 2003.aastal Micro-SIM (3FF) versiooni väiksema kiibi ja plastikümbrisega, et mahutada SIM-kaarti veelgi väiksematesse seadmetesse. 2012.aastast alates hakati kasutama veelgi kahandatud plastikümbrisega Nano-SIM (4FF). [3]

Sängitatud SIM formaadis **MFF2** (ingl *machine-to-machine form factor*) tüüpi SIM pakub sama funktsionaalsust, mis 1FF, 2FF, 3FF ja 4FF, kuid erineb oluliselt paigaldamise poolest. Kuna MFF2 SIM koosnebki vaid kiibist, pole selle nimetamine „SIM-kaardiks“ enam kuigi täpne. MFF2 tüüpi eSIM on tootmisprotsessi käigus otse seadme loogikaplaadile joodetud ning pole mõeldud teisaldatavaks. See sobib muuhulgas kasutamiseks seadmes, mis asub välitingimustes, sest kiip on vaakumi abil suletud. Seetõttu on kiip paremini kaitstud korrosiooni ja vibratsiooni eest, pikendades selle eluiga. Lisaks peetakse MFF2 kiipe turvalisemaks, sest neid ei saa seadmest eemaldada ning väärkasutada. On oluline märkida, et operaatorprofiil on MFF2 kiibile alati eellaetud ja muutmatu, välja arvatud juhul, kui vastaval kiibil on eSIM tugi. Seejuures on eksitav eeldada, et MFF2 tähendabki eSIM’i, sest tegelikult võib eSIM tehnoloogia esineda kõigis formaatides alustades 1FF kuni 4FF tüüpi SIM-kaartidena. Siiski on eSIM kõige levinum just MFF2 formaadis. [4]



Joonis 1. Levinumate SIM-kaartide formaadid. [4]

2.3.3 SIM-kaartide haldus

Traditsioonilise SIM-kaardi väljastab kasutajale mobiilsideoperaator. Selleks sõlmib kasutaja valitud teenusepakkujaga lepingu ning saab vastu SIM-kaardi, mis tuleb tal võrku registreerimise jaoks seadmesse sisestada. Joonisel 2 on see SIM-kaart on sammul

„1“ märgistatud punase täpiga, tähistamaks, et selle profiili volitused on vastava operaatori väljastatud [5].

Kui kasutaja peaks soovima operaatorit vahetada, tuleb tal sõlmida teine leping uue operaatoriga ning saab samuti uue füüsilise SIM-kaardi (tähistatud sinise täpiga Joonisel 2 sammul „2“). Kuigi kasutaja valduses on nüüd ka uus SIM-kaart, on seade ühendatud endiselt esimese operaatori mobiilivõrku. Operaatori vahetamiseks tuleb kasutajal füüsiliselt SIM-kaardid seadmes ümber vahetada (Joonis 2 samm „3“). Seadmes erinevate operaatori profiilide vahel lülitumine pole võimalik olukorras, kus operaator on oma ärireeglites sätestanud operaatorivahetuse piirangu kasutajale müüdud seadme osas [5].

Joonis 2. SIM-kaartide haldus. [5]

2.4 eSIM arhitektuur

2.4.1 eSIM profiil

SIM profiil on programmiüksus, mis hõlmab endas mobiilsideoperaatori infot: abonemendi (ingl *subscription*) teavet, operaatori volitusi ning potentsiaalselt ka operaatori või kolmanda osapoole SIM-põhiseid rakendusi [2].

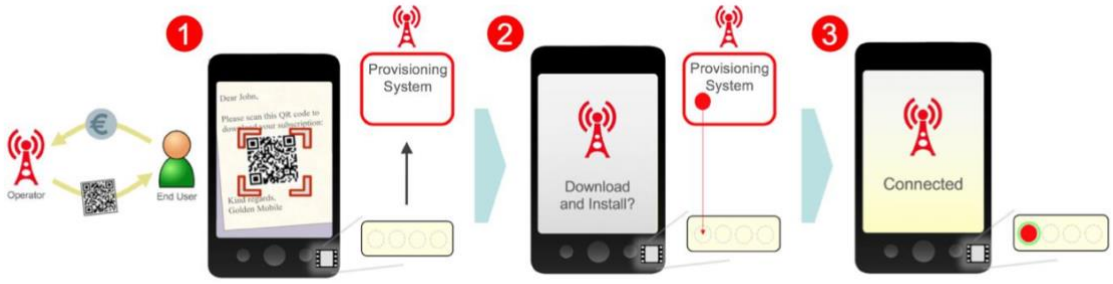
eSIM profiilide sisu ja struktuur on sarnane füüsiliste SIM-kaartide profiilidele, kuid ühe olulise erinevusega: eSIM puhul kasutatakse **eUICC** (ingl *Embedded Universal Integrated Circuit Card*) turvaelementi, mis võimaldab endas hoiustada operaatorite profiile. eUICC-le saab „üle õhu“ laadida alla mitme erineva operaatori profiile. eUICC koostalitlusvõimeliste profiilide tehnilise arhitektuuri on sätestanud Trusted Connectivity Alliance [6].

Kuigi võib öelda, et eUICC on seadme lahutamatu osa, siis SIM profiil jääb operaatori omandiks, sest see sisaldab operaatori omatud intellektuaalset vara (IMSI, ICCID, turvaalgoritmid jne), mis on tarnitud litsentside alusel [2].

2.4.2 eSIM'ide valmendamise kaughaldus

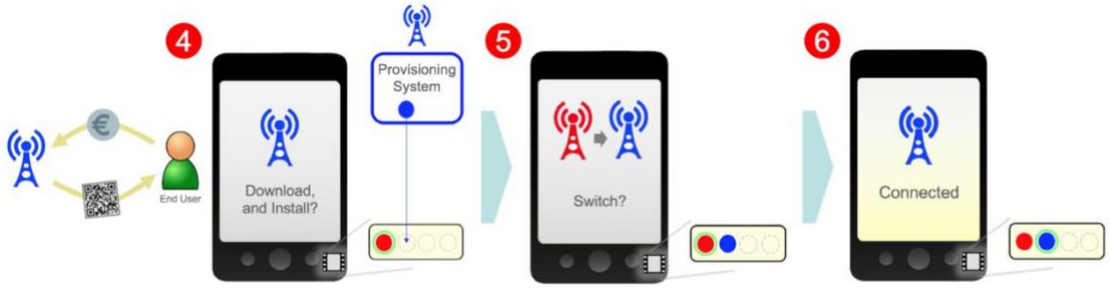
SIM valmendamise kaughalduse (ingl *Remote SIM Provisioning*) klassikalise mudeli kohaselt puuduvad füüsilised SIM-kaardid. Selle asemel on üksainus sängitatud SIM-kiip nimega eUICC, mis harilikult on joodetud seadme loogikaplaadile. eUICC võimaldab seadmes hoiustada mitut SIM profiili, kus iga profiil sisaldab erineva operaatori infot, mis füüsiliste SIM-kaartide puhul olid Joonisel 2 märgitud punase ja sinise täpiga [5].

Kasutaja sõlmib samamoodi operaatoriga lepingu, kuid füüsilise SIM-kaardi asemel saab ta operaatorilt juhised, kuidas ühendada seadet vastava operaatori valmendamise kaughaldussüsteemiga. Joonise 3 sammul „1“ on kujutatud seadet, millega kasutaja skaneerib operaatori väljastatud QR-koodi. See QR-kood sisaldab valmendamise kaughalduse SM-DP+ (ingl *Subscription Manager Data Preparation Plus*) serveri aadressi, mis võimaldab turvaliselt operaatori süsteemiga ühenduda ja SIM profiil seadmesse alla laadida. Kui kasutaja on profiili seadmesse paigaldanud ja aktiveerinud, saab seade ühenduda operaatori võrku. On oluline märkida, et QR-koodi skaneerimine on vaid üks mitmest profiili allalaadimise võimalusest. Nende seas leidub ka võimalus profiili aktiveerida juba eelseadistatud seadmes [5].



Joonis 3. eSIM profiili paigaldamine. [5]

Kui kasutaja peaks soovima operaatorit vahetada, tuleb tal sõlmida teine leping uue operaatoriga ning taaskord saab kasutaja vastu näiteks QR-koodi, mille abil teise operaatori profiil alla laadida (samm „4“ joonisel 4). Kasutaja saab seejärel seadmes erinevate operaatorite profiilide vahel valida, millise operaatori võrgu külge seadmega parasjagu ühenduda (samm „5“ ja „6“ joonisel 4) [5].



Joonis 4. eSIM profiili vahetamine. [5]

2.5 eSIM lahendusmudel

SIM profiilide valmendamise kaughalduseks on levinud kaks standardiseeritud mudelit: M2M mudel ja tavatarbija mudel. GSMA on eSIM'i kasutamisvaldkonnad jaganud kaheks vastavalt kummagi kanali ärivajadustele [5].

M2M mudel (ingl *Machine-to-Machine solution*) oli esimene, mis teostas SIM valmendamise kaughaldust. M2M mudel on kasutuses ärilt ärile ja ärilt klindile (ingl *business-to-business-to-consumer*, lüh B2B2C) ärimudeli puhul. Valmendamise kaughalduseks kasutab M2M serveripoolset profiili paigaldamise algatamist (ingl *push model*). M2M lahenduses toimub profiilide haldamine kaughalduse abil ilma lõppkasutaja sekkumiseta [5].

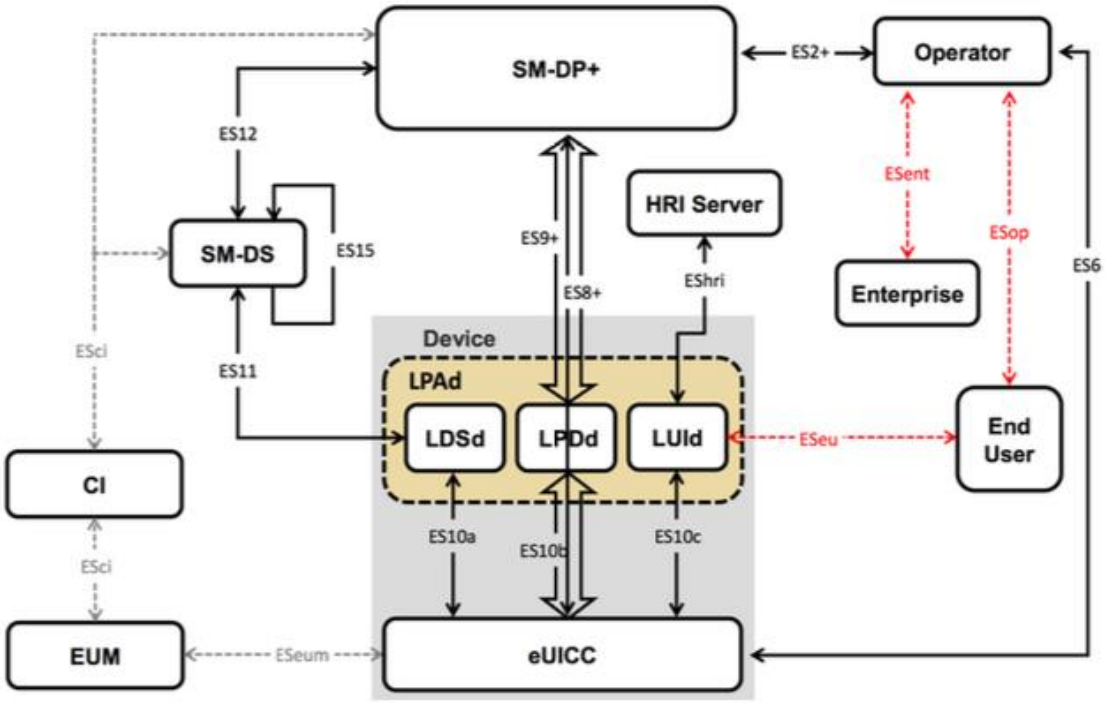
M2M mudel on loodud B2B2C ärimudeli nõuete rahuldamiseks. Käesolev lõputöö keskendub tavatarbija mudelile, sest käsitletav probleem puudutab ettevõtte loodud laiatarbetarkvara, mida kliendid kasutavad nutiseadmetes. Probleemi kõrvaldamine eeldab seega tavatarbija mudeli jaoks disainitud lahendust.

Tavatarbija mudel (ingl *consumer solution*), teisisõnu teenusepakkuja-kliendi kanal, kirjeldab kasutusvaldkonda, kus lõppkasutajal on valida erinevate teenusepakkujate vahel. Tavatarbija lahendus nõuab lõppkasutajalt teadmisi ja interaktsiooni ning eelduseks on, et kasutaja on tuttav teenusepakkujate kasutajaliideste ja lahendustega. Tavatarbija mudel on ehitatud eelnevalt väljatöötatud M2M lahenduse peale, lisades vajalikku funktsionaalsust [5].

Tavatarbija mudel eeldab, et kontroll abonemendi profiili sätestamise ja haldamise üle on täielikult lõppkasutaja käes läbi seadme kasutajaliidese. Mudeli järgselt algatab

lõppkasutaja eUICC-le profiili paigaldamise (ingl *pull model*). Ka teatud juhtudel, näiteks eSIM paigaldamisel nutikellale, kuvatakse profiili aktiveerimise liidest kasutaja peamises seadmes, näiteks nutitelefonis. See tagab lõppkasutaja jaoks sujuvama kasutuskogemuse. Lahendus põhineb mitmete erinevate süsteemikomponentide koostööl, mida kujutab joonis 5. Joonisel on välja toodud süsteemikomponentide seosed, infoedastuse suunad kuid ka välised üksused, näiteks operaator, lõppkasutaja, EUM ja CI. Olulisemad eUICC-ga seotud süsteemikomponentid on järgmised [5]:

- eUICC on turvaelement, mis võimaldab hoiustada erinevate operaatorite profiile. See võib olla sängitatud seadme loogikaplokile kui ka esineda füüsiliselt eemaldataval SIM-kaardil.
- SM-DP+ valmistab ette, hoiustab ja turvab operaatori profiili andmeid. Lisaks sellele on SM-DP+ komponenti kapseldatud ka SM-SR (ingl *Subscription Manager Secure Routing*) komponent, mis vastutab profiilide staatuse haldamise eest
- SM-DS (ingl *Subscription Management Discovery Service*) ja LPA (ingl *Local Profile Assistant*) võimaldab SM-DP+ komponendil ühenduda eUICC-ga ilma teadmiseta, millisesse võrku seade parasjagu ühendatud on. See eriomadus on oluline, sest seade võib olla ühendatud erinevatesse pääsuvõrkudesse erinevatelt ligipääsuaadressidelt. SM-DS komponenti kasutatakse LPA teavitamiseks, kui operaatori profiil on eUICC-sse allalaadimiseks saadaval.
- LPA on moodul, mis vastutab krüpteeritud profiilide allalaadimise eest eUICC-le. LPA skaneerib eUICC-s seadistatud sageduse alusel aktiivselt SM-DS olekut. Lisaks pakub LPA lõppkasutaja jaoks kasutajaliidest, mille abil saab kasutaja profiile hallata. LPA põhiline funktsionaalsus võib olla sisseehitatud ka otse eUICC sisse.



Joonis 5. Tavatarbija mudeli arhitektuur. [5]

Tavatarbija mudeli süsteemikomponentide koostöö rahuldab mudeli ärinõudeid. Tavatarbija mudeli arhitektuur sisaldab võrreldes M2M mudeliga lisakeerukust, mis on vajalik enamate kasutusjuhtude jaoks. Sellest johtuvalt peab tavatarbija mudel vastama ka enamatele tehnilistele- ja turvanõuetele [5]. Järgmises peatükis on kirjeldatud tavatarbija mudelile vastava eSIM arhitektuuri turvalisuse aspekte.

2.6 eSIM turvalisuse aspektid

GSMA (Global System for Mobile Communications Association) on eSIM'i standardi väljatöötamise algusest saadik seadnud eesmärgiks pakkuda eSIM lahendustele vähemalt sama turvalisuse taset kui füüsilistele SIM'idele. Valmendamise kaughaldus seab turbele mitmeid väljakutseid. GSMA on välja toonud järgmised riskid [2]:

- kaughaldusega seotud riskid;
- profiilid pole eSIM'i puhul fikseeritud ühele füüsilisele kaardile, vaid on väljavahetatavad;
- eSIM saab kanda samaaegselt mitut profiili;
- eSIM profiilide haldamises on seotud rohkem osapooli, sealhulgas lõppkasutaja.

eSIM arhitektuuri jaotatakse mitmeks alaosaks, kus igal osal on turbes oma roll. Võimalikuks jaotuseks oleks rollide kobar, mis koosneb järgmistest alamosadeks: operaator, eUICC, eUICC tootja, SM-DP+, SM-DS. eSIM spetsifikatsioon seab lahendusele erinevad turbenõuded, mida saab jaotada üldisteks turbenõueteks ning iga rolliga seotud turbenõueteks. Sertifikaatide väljastaja (ingl *Certificate Issuer*, lüh CI) on roll, mida GSMA on väljastanud loetud ettevõtetele. Sertifikaadi väljastaja käitub eUICC kaughalduse süsteemis usaldatud kolmanda osapoolena, kes pakub eUICC arhitektuuri komponentide vahelist autentimise võimekust [2].

Tavatarbija mudelile vastav eSIM kasutab turbe tagamiseks kombinatsiooni erinevatest krüptoalgoritme ja räsifunktsioone. Kasutusel on PSK (ingl Pre-Shared Key) ja PKI (ingl Public Key Infrastructure) põhised krüptograafilised võtted. Kõik krüptograafilised võtted peavad vastama vähemalt dokumendis NIST SP800-57 (ingl *National Institute of Standards and Technology Special Publication 800-57*) kirjeldatud soovitustele. Kasutatud algoritmid, räsifunktsioonid ja minimaalsed võtmepikkused on järgnevad [8]:

- Advanced Encryption Standard (ingl *Advanced Encryption Standard*, lüh AES) - 128 bitti, bloki suurus 128 bitti;
- Rivest–Shamir–Adleman (ingl *Rivest-Shamir-Adleman*, lüh RSA) – 3070 bitti;
- Elliptic curve (ingl *Elliptic curve*, lüh ECC) - 256 bitti;
- SHA-256 (ingl *Secure Hash Algorithm 256-bit)* räsialgoritm - 256 bitti.

SCP (ingl *Secure Channel Protocol*) ehk turbekanali reeglistik pakub turvalist suhtluskanalit eUICC ja eSIM infrastruktuuri vahel. SCP tagab sõnumivahetuses turbe kolm omadust: terviklikkuse, käideldavuse ja konfidentsiaalsuse. Tavatarbija eUICC spetsifikatsioon põhineb SCP-SGP22, SCP80 ja SCP81 protokollidel [9].

eUICC arhitektuuris ja profiilide halduses kasutatakse turbedomeene (ingl *Security Domain*). Turbedomeen on olemuselt rakendus, mis omab kindlaid õiguseid ning sisaldab kindlaid võtmeid krüptoalgoritmide jaoks, viimaks läbi kindlaid ülesandeid. Turbedomeenide kihilise jaotuse eesmärk on hajutada vastutust ja andmeid [7].

Käesoleva hetkeni pole eSIM spetsifikatsiooni puhul teada märkimisväärseid turvanõrkusi. Kasutatavate turbedomeene ja krüptoalgoritmide peal on juba aastaid ulatuslikult teste läbi viidud. Rollipõhine rühmitamine ja turbekanali reeglistik annavad aluse pidada eSIM põhistruktuuri turvaliseks [2].

2.7 Alternatiivsed tehnoloogiad

Peatükis 2.3.2 loetletud SIM-kaartide formaadid erinevad pealtnäha küll füüsilistelt mõõtmetelt, aga need võivad endas kätkeda ka erinevaid SIM tehnoloogiaid. Kuigi eSIM tehnoloogia on veel noor, leidub sellele juba märkimisväärseid alternatiive edasiarenduste näol.

iSIM (ingl *Integrated-SIM*) ehk integreeritud SIM'i võib pidada eSIM'i alamliigituseks või edasiarenduseks. iSIM jagab hariliku eSIM'iga samaväärset funktsionaalsust, kuid erineb paigaldusviisi poolest. Kuna eSIM on küll seadme loogikaplokile sängitatud (joodetud) ning sellest hiljem eemaldamatu, on eSIM olemuselt siiski väline moodul. Seevastu on iSIM seadme mobiilse andmeside moodulisse sisseehitatud, st riistvaraga täielikult integreeritud. iSIM eelisteks peetakse järgmiseid omadusi [10]:

1. Seadme tootmine on soodsam, sest puudub vajadus väliste riistvarakomponentide (eraldi eUICC kiibi) järgi.
2. Seade on potentsiaalselt töökindlam ja pikema elueaga, sest SIM on integreeritud seadme protsessoriga. iSIM maandab tehniliste rikete riske, mis on tingitud moodulite vahelisest suhtlusest.
3. Seade on energiaefektiivsem, sest erinevalt eSIM'ist ei vaja protsessoriga integreeritud iSIM eraldi voolu kulutavat mikroprotsessorit.
4. Seadmesse jääb alles rohkem füüsilist ruumi, sest iSIM ei võta seadme loogikaplaadil väärtuslikku ruumi ning mahub väiksemale alale kui 1 ruutmillimeeter.
5. iSIM on potentsiaalselt turvalisem, sest kaasab endaga lisanduva autentimise kihi.

Hoolimata loetletud eelistest eSIM'i ees, ei ole iSIM veel tänaseni standardiseeritud lahendus, mida pole ametlikult tunnustanud ka GSMA [2].

nuSIM'i (saksa „*Nu*" – „uus") võib pidada iSIM'i minimeeritud versiooniks. nuSIM on loodud spetsiaalselt M2M kasutusmudeli ning LPWAN (ingl *Low-power wide-area network*) võrgutehnoloogiate jaoks, näiteks NB-IoT (ingl *Narrowband-Internet of Things*) ja LTE-M (ingl *Long-Term Evolution Machine Type Communication*) [12]. nuSIM'ist on eemaldatud võimalikult palju kasutusvaldkonna jaoks ebavajalikke osasid ja funktsionaalsust, mida harilikult kannavad füüsiline SIM-kaart, eSIM või iSIM. See võimaldab oluliselt hoida kokku mälumahtu. nuSIM'ile paigaldatakse operaatoriprofiil tootmisprotsessi käigus, mida hiljem ei saa muuta. Lisaks puudub nuSIM'il

valmendamise kaughalduse ning profiilide vahetamise võimekus. NB-IoT valdkonda kuuluva riistvara näideteks on suitsuandurid, targad lülitid, mõõtmisandurid. Puuduolevat funktsionaalsust tasub vaadelda kui mõistlikku optimeerimist, sest neis seadmetes hoitavaid profiile polegi kasutajal üldiselt kunagi vaja muuta [12]. GSMA pole nuSIM'i ametlikult tunnustanud ning sellel pole ka koostalitlusvõimeliste profiilide arhitektuuri tuge, mille standardid on sätestanud Trusted Connectivity Alliance [6].

Käesolev lõputöö keskendub eSIM tehnoloogia uurimise ja paigaldamise teostamisele Android ja iOS mobiilirakendustes, sest need operatsioonisüsteemid toetavad töö kirjutamise ajal veel vaid eSIM'i. Lisaks on eSIM erinevalt tema alternatiividest ainsana GSMA tõendatud ja nõuetele vastav standardiseeritud lahendus. Kuigi eSIM pole kaugeltki veel nii mahuliselt käibel kui traditsioonilised SIM-kaardid, on eSIM kasutusvaldkond siiski palju suurem järgnevalt kirjeldatud iSIM ja nuSIM lahenduste omast.

2.8 Ettevõtte põhjused uue funktsionaalsuse vajalikkusele

Telia tuli 2018.aastal turule enda eSIM toega, pakkudes lepingulisi mobiiliteenuseid eSIM kujul. Lisandunud on ka võimekus tõsta kliendi füüsilise SIM-kaardiga seotud mobiilinumber eSIM'ile ja vastupidi, muutes teenuse kliendi jaoks paindlikuks. Telia toob Superi kui enda toote eSIM arenduse vajalikkuse põhjenduseks järgmised punktid:

- Kontaktivaba teenusega liitumine – klient ei pea tänu eSIM lahendusele enam SIM-stardikomplekti ostmiseks minema müügipunkti, vaid saab ostu sooritada digitaalselt. Muuhulgas on kontaktide vähendamine nakkusohutuse säilitamiseks eriti oluline näiteks haiguspandeemiate vältel.
- Keskkonnasõbralikkus – seoses üleminekuga füüsilistelt SIM-kaardi lahendustelt digitaalsetele eSIM'idele, pole enam tarvis toota ega käidelda SIM-kaardi kiipe, neid ümbritsevat plastikaarte ega papist tootepakendeid, säästes sellega loodusressursse.
- Teenusega liitumise kiirus – klient saab eSIM teenusega liituda loetud minutite jooksul kodust väljumata kasutades vaid oma nutitelefoni.
- Mitme SIM-kaardi haldus – eSIM lahenduse abil kõrvaldatakse seadme SIM-kaardi füüsiliste pesade arvu piirang ning klient saab seadmes hakata paralleelselt kasutama rohkem SIM-kaarte kui varem.

Tuginedes klientettevõtte loetletud füüsilise SIM-kaardiga seotud puudujääkidele ja eSIM eelistele füüsilise SIM-kaardi ees, saab teenusega liitumist eSIM abil pidada sobivaks lahenduseks olemasolevate probleemide kõrvaldamiseks.

3 Rakenduse mooduli analüüs

3.1 Rakenduse mooduli kasutajanõuded

Kasutajanõuded on dokument, mis kirjeldab rakenduse kasutajate ootusi, vajadusi ja eesmärke ning nõudeid, mida rakendus peab täitma. Need võivad hõlmata funktsionaalsuse, kasutatavuse, jõudluse, turvalisuse ja muude aspektide nõudeid. Kasutajanõuded on olulised, sest need aitavad tagada, et rakendus vastab kasutajate vajadustele ja eelistustele. Lisaks võivad need aidata vältida tulevasi probleeme ja kulukaid muudatusi, sest nad aitavad selgitada rakenduse arendajatele, millised on kasutajate ootused ja mida nad tegelikult vajavad [12].

Funktsionaalsed kasutajanõuded kirjeldavad rakenduse funktsioone ja nende toimimist. Need hõlmavad nõudeid selle kohta, mida rakendus peab suutma teha. Näiteks võivad funktsionaalsed kasutajanõuded kirjeldada, milliseid toiminguid kasutaja saab rakenduses teha, milliseid sisendeid kasutaja saab anda ja milliseid väljundeid rakendus annab [12]. Autor määratles koos kliendiga rakendusele järgmised funktsionaalsed nõuded:

1. Kasutajana soovin, et mobiilirakendus kontrolliks ise, kas minu seade toetab eSIM'i, et ei peaks ise käsitsi seadme IMEI (ingl International Mobile Equipment Identity) koodi sisestama.
2. Kasutajana soovin näha, mida valikus olevad eSIM stardipaketid sisaldavad, et leida enda jaoks sobivaim.
3. Kasutajana soovin saada valida mitme eSIM stardipaketi vahel, et ei peaks pärast liitumist kohe lisasammuna raha juurde laadima.
4. Kasutajana soovin eSIM'i soetades valida mitme erineva mobiilinumbri vahel, et leida endale meelepärane ja meeldejääv number.
5. Kasutajana soovin enne ostu sooritamist tutvuda hinnakirja ja üldtingimustega, et viia end teenuse tingimustega kurssi.
6. Kasutajana soovin, et saaksin eSIM ostu sooritamiseks valida krediitkaardi ja pangalingi vahel, et saaksin tasuda endale sobivaima maksemeetodiga.

7. Kasutajana soovin, et eSIM paigaldamise juhend saadetakse pärast ostu sooritamist e-postile, et vajadusel abi leida.
8. Kasutajana soovin, et saaksin kohe pärast eSIM ostu sooritamist ja eSIM seadmesse paigaldamist mobiilirakendusse sisse logida, et alustada teenuse kasutamist.

Mittefunktsionaalsed kasutajanõuded on kasutajanõuded, mis kirjeldavad rakenduse kvaliteedi ja jõudlusega seotud omadusi. Need hõlmavad nõudeid, mis käsitlevad rakenduse kasutatavust, jõudlust, turvalisust ning kättesaadavust. Näiteks võivad mittefunktsionaalsed kasutajanõuded hõlmata nõudeid seoses rakenduse reageerimiskiirusega, turvalisusega, kasutajaliidese kujundusega, andmekaitse ja privaatsusega. Need nõuded on samavõrd tähtsad kui funktsionaalsed kasutajanõuded, sest nad mõjutavad kasutajate kogemust ja rahulolu [12]. Autor määratles koos kliendiga rakendusele järgmised mittefunktsionaalsed nõuded:

1. Kasutajana soovin, et eSIM'iga liitumine ja seadistus oleks lihtne, et saaksin iseseisvalt protsessiga hakkama.
2. Kasutajana soovin, et rakendus töötaks eSIM'iga liitumisteekonna jooksul sujuvalt ning reageeriks sisenditele kiiresti, et saaksin kiiremini asuda teenust kasutama.
3. Kasutajana soovin, et eSIM liitumisteekond sisaldaks võimalikult vähe samme, et saaksin kiiremini asuda teenust kasutama.
4. Kasutajana soovin, et eSIM liitumisteekonna jooksul näeksin progressiriba, et aru saada, kui palju samme veel läbida on vaja.
5. Kasutajana soovin, et peaksin eSIM ostuteekonnal võimalikult vähe andmeid käsitsi sisestama, et saaksin kiiremini asuda teenust kasutama.
6. Kasutajana soovin, et mobiilirakenduse disain oleks kujunduselt kena ja loogiline, et liitumisteekond oleks arusaadav.
7. Kasutajana soovin, et eSIM liitumisteekond oleks turvaline ja privaatsust arvestav, et minu andmed oleksid kaitstud.

Rakenduse kasutajanõuded on olulised, kuna need aitavad tagada, et rakendus täidab kasutajate ootusi ja vajadusi. Lisaks aitavad täpselt määratletud kasutajanõuded arendajatel paremini mõista, mida rakenduselt oodatakse ja kuidas seda ehitada. Käesoleva projekti arenduse tööde planeerimisel on arvestatud kõikide funktsionaalsete ja mittefunktsionaalsete kasutajanõuetega. Peatükis 4.1 on toodud näide, kuidas võeti tööde planeerimisel arvesse ühte määratletud funktsionaalset kasutajanõuet. Järgnevas

peatükis 3.2 on selgitatud, kuidas kasutatakse määratletud kasutajanõudeid rakenduse valideerimise protsessis.

3.2 Rakenduse valideerimine

Rakenduse valideerimine on oluline protsess, mis aitab tagada, et rakendus vastab äri nõuetele ja kasutajate vajadustele ning on kvaliteetne, kasutajasõbralik ja töökindel. Arendatud mooduli valideerimiseks on kavas hinnata järgmiseid aspekte:

- Nõuete hindamine – peatükis 3.1 loetletud rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded tuleb üle vaadata ja hinnata, et tagada nende vastavus kasutajate vajadustele ja eesmärkidele. Selle käigus hinnatakse ka nõuete selgust, täielikkust ja rakendatavust.
- Funktsionaalsete testide läbiviimine - rakenduse funktsionaalsust tuleb testida, et tagada, et see täidab oma eesmärki ja töötab õigesti. Selle käigus hinnatakse ka rakenduse jõudlust ja töökindlust. Mobiilirakendust testib arendusmeeskonna testija käsitsi E2E testidega.
- Turvalisuse hindamine - rakenduse turvalisust tuleb hinnata, et tagada, et see vastab turvalisusstandarditele ja kasutajate andmed on kaitstud. Selle käigus hinnatakse ka rakenduse haavatavust ja turvalisuse puudujääke. Turbeanalüüsi viib läbi arendusmeeskond.
- Disaini hindamine - rakenduse disaini tuleb hinnata, et tagada vastavus disaineri ettejoonistatud prototüübijoonistele, aga ühtlasi ka kasutajaliidese disaini ja kasutusmugavuse parimatele tavadele. Mobiilirakenduse disaini hindamise viib läbi klientettevõtte tooteomanik koos disaineriga.
- Kasutajakogemuse hindamine – klientettevõte annab valminud mobiilirakenduse lahenduse testida kindlale hulgale beetakasutajatele. Rakenduse kasutajakogemust tuleb hinnata, et tagada rakenduse kasutajasõbralikkus ja vastavus kasutajate ootustele. Selle käigus hinnatakse kasutajaliidese navigeerimist, selle kasutamise lihtsust, vigade arvu ja muud tagasisidet.
- Lõpliku hindamisaruande koostamine - rakenduse valideerimise protsessi tulemused tuleb kokku võtta ja koostada lõplik hindamisaruanne, kus on esitatud rakenduse kvaliteedi, tõhususe, kasutusmugavuse ja usaldusväärsuse hinnang.

Aruandes võib esineda soovitusi ja parandusettepanekuid, et rakendus vastaks paremini nõuetele ja kasutajate ootustele.

Rakenduse valideerimisel võetakse arvesse kõiki loetletud aspekte. Täpselt määratletud kriteeriumid on olulised, et rakendus saaks enne turustamist terviklikult hinnatud.

3.3 Rakenduses kasutatavad tehnoloogiad ja raamistikud

Käesoleva arenduse jaoks on valitud samad peamised tehnoloogiad ja raamistikud, mille abil senigi rakendus varasemalt kirjutatud on. See tagab koodi ühtluse ning võimaldab antud projektis juurde arendada vaid nii palju, kui vajalik on. Sellegipoolest on autor käesoleva arenduse klientrakenduse puhul võtnud kasutusele nii mõnegi uuema teegi, et täita rakenduse funktsionaalseid vajadusi. Järgnevalt on kirjeldatud, mille jaoks valitud tööriistu kasutatakse nii taga- kui ka eesrakenduse puhul.

3.3.1 Tagarakenduses kasutatavad tehnoloogiad ja raamistikud

Java

Tagarakendus on realiseeritud programmeerimiskeeles Java. Tegu on tüübikindla klassipõhise objektorienteeritud keelega. Java lähtekood kompileeritakse baitkoodi. Kompileeritud Java lähtekoodi käivitamine on võimalik Java virtuaalmasina JVM abil sõltumata platvormist. Seega on Java sõltumatu riistvarast ja operatsioonisüsteemist, millel seda kasutatakse [13]. Antud projektis on kasutusel 2018.aastal avaldatud pikaajalise toega Java LTS 11 versiooni.

Spring Boot

Spring on Java avatud lähtekoodiga annotatsioonidel põhinev veebiteenuse raamistik. Spring Boot on Spring raamistiku edasiarendus, mis võimaldab arendaja jaoks mugavalt luua Spring veebirakendusi. Spring Boot üritab minimeerida arendaja kirjutatavat seadistuse hulka, pakkudes *starter* sõltuvusi ning aidates kolmanda osapoole teekide automaatse seadistusega. Sõltuvustena on antud projektis kasutusel Spring Data JPA, Spring Security. Spring Boot sisaldab HTTP (ingl *Hypertext Transfer Protocol*) serverit, näiteks Tomcat, Jetti või Undertow, mis võimaldab arendajal veebirakendust lokaalselt käivitada [14].

Gradle

Gradle on rakenduse kooste (ingl *build*) automatiseerimise tööriist. Gradle võimaldab mugavalt hallata Java projekti sõltuvusi, käivitada teste ja teisi käivitamisega seotud protsesse ning koostada uusi versioone [15].

Lombok

Lombok teek on projektis kasutusel trafarettkoodi kirjutamise vajaduse vähendamise tarbeks. Lombok koostab Java annotatsioonide põhjal klassidele *getter*-, *setter*- ja konstruktormeetodid automaatselt kompileerimise hetkel. Sellevõrra vähem tuleb arendajal lähtekoodi käsitsi kirjutada. Arendusvahendisse on võimalik paigaldada Lomboki pistikprogramm, et töövahend oskaks pakkuda koodieete tuge [16].

WireMock

WireMock on avatud lähtekoodiga paindlik API (ingl *Application Programming Interface*) ehk rakendusliidese simuleerimise tööriist, mis võimaldab arendamisel ja testimisel simuleerida HTTP päringute vastuseid. WireMock käivitab rakenduse jooksutamiseks enda HTTP serveri. WireMock toetab mitmeid simuleeritud API'de loomise viise: läbi koodi, WireMock REST (ingl *Representational state transfer*) API või JSON failide alusel [17]. Antud arenduses on WireMock kasutusel integratsioonitestide loomisel, simuleerides Superi tagarakendusele eSIM'ide pärimise vastuseid.

3.3.1 Eesrakenduses kasutatavad tehnoloogiad ja raamistikud

JSON

JSON (ingl *JavaScript Object Notation*) on liiasuseta andmevahetusformaat, mis põhineb programmeerimiskeele JavaScript alamosal. JSON formaati on mugav kasutada koos Java programmeerimiskeelega teekide abil, mis teisendavad JSON formaati Java objektideks ja vastupidi. JavaScript rakenduses on JSON struktuuriga objekti kerge itereerida [18].

JavaScript

JavaScript on dünaamiliselt tüpiseeritud programmeerimiskeel ehk selle muutujate andmetüübid tuvastatakse interpretaatori poolt käitusfaasis. JavaScript võimaldab luua dünaamilist ehk kasutaja sisendile reageerivat veebirakendust. Kuna JavaScriptil puudub tüübivigade käitamiseelne tuvastus, on suur võimalus vigase koodi kirjutamiseks [19].

TypeScript

TypeScript on JavaScripti ülemhulk, mis käitusfaasis kompileeritakse JavaScriptiks. TypeScript pakub JavaScriptile täiendavat süntaksit koos valikuliste tüübidefinitsioonidega, mille olemasolul käsitletakse muutujaid staatiliselt tüpiseeritud muutujatena. TypeScripti tüübidefinitsioonid aitavad arendajal tüübivigu avastada juba kompileerimisfaasis. Käesoleva töö klientrakenduse kirjutamisel on kasutatud TypeScripti, kus tüübideklaratsioonid on iga muutuja juures kohustuslikud [20].

React

React on JavaScripti raamistik kasutajaliideste arendamiseks. React on deklaratiivne, mis tähendab seda, et arendaja saab koodis kirjeldada, millisena ta soovib, et DOM (ingl *Document Object Model*) ehk dokumendiobjekti mudel välja näeks. React hoolitseb järgneva eest ise. React rakendus koosneb kapseldatud komponentidest. Iga komponent võib omada oma olekut ja funktsioone. See võimaldab hoida rakenduse olekut väljaspool DOM'i. React rakenduse puhul on võimalik struktuuri ja funktsionaalsust omavahel kombineerida samas failis, kasutades JavaScript süntaksit laiendavat keelt JSX [21]. JSX on JavaScripti süntaksit laiendav keel, mis võimaldab muutujate defineerimiseks kasutada HTML'ile või XML'ile sarnanevat süntaksit. Reactis nimetatakse neid muutujaid elementideks. JSX kompileeritakse rakenduse käitusfaasis JavaScriptiks [22]. TSX (TypeScript XML) fail on TypeScripti abil kirjutatud fail, mis kasutab JSX (JavaScript XML) süntaksit.

React Native

React Native on React'il põhinev JavaScripti raamistik ning sellega on võimalik arendada hübriidmobiilirakendusi. React on loodud veebilehitsejas kasutatavate, React Native mobiilirakenduse kasutajaliideste arendamiseks. React Native toetab iOS ja Android platvorme. React Native abil on võimalik kirjutada mobiilirakenduse kasutajaliidest ühe korra, sest React Native „tõlgib" kasutajaliidese elemendid platvormipõhisteks elementideks [23].

React Query

React ei sisalda ega näe arendajatele vaikimisi ette ühtegi kindlat viisi andmete pärimise ega komponentide sees andmete uuendamise osas. Seetõttu peavad arendajad ise valima sobilikud kolmanda osapoole teegid, mille abil serverrakendusest andmeid pärida ning seejärel klientrakenduse poolel rakenduse olekut hallata. Oleku haldamise teegid on küll

sobilikud klientrakenduse oleku haldamiseks, kuid need ei sobi otseselt kasutamiseks serveri oleku jälgimiseks. Serveri olek erineb oluliselt klientrakenduse olekust. Need kaks ülesannet – andmete pärimine ja oleku haldamine - toovad klientrakenduses kaasa mitmeid väljakutseid [24]:

- Kuidas hallata vahemälu kasutamist andmete ajutiseks salvestamiseks?
- Kuidas värskendada andmeid „taustal“?
- Kuidas teada, et andmed on „aegunud“ ja vajavad jooksvalt värskendamist?
- Kuidas värskendada andmeid võimalikult aegsasti?
- Kuidas optimiseerida päringute haldamise jõudlust?
- Kuidas teostada andmete pärimist vaid vajaduspõhiselt „laisa laadimise“ põhimõttel?

React Query püüab lahendada kõiki neid muresid, paketeerides kogu funktsionaalsuse ühteainsasse teeki. Pealtnäha võib React Query’t pidada lihtsalt andmete pärimise teegiks, kuid tegelikult sisaldab see ka oskust optimeerida vahemälu ning sünkroniseerida serveri olekut klientrakenduses [24].

React Query on loodud „konventsioon konfiguratsiooni üle“ põhimõttel. Konventsioon konfiguratsiooni üle on tarkvara raamistikes kasutatav tarkvarakujunduse paradigma, mis püüab vähendada otsuste arvu, mida raamistikku kasutav arendaja peab tegema, ilma, et raamistik kaotaks paindlikkust. React Query võimaldab arendust lihtsustada ning ressursse kokku hoida, kõrvaldades vajaduse käsitsi kirjutada suurtes kogustes andmete pärimise ja oleku haldamisega seotud koodi. Samas on teeki võimalik hulgaliselt seadistada vastavalt rakenduse vajadustele, mistõttu on see sobilik kasutamiseks ka väga suurtes rakendustes. Lisaks on raamistikul potentsiaal suurendada rakenduse jõudlust, hoides kokku mälukasutust. [25].

React Hooks

React Hooks lisati React’ile 16.8.0 ja React Native’ile 0.59 versioonis. *Hook* on JavaScripti funktsioon, mis võimaldab kasutada komponendis olekut ja teisi React’i elutsükli funktsionaalsusi ilma Javascripti klassi loomata. *Hook*’i saab kasutada vaid funktsionaalses komponendis ning ei saa kasutada klassis. *Hook*’i eesmärk on eraldada korduvkasutatav olekuga seotud osa funktsionaalsest komponendist [26].

4.3.2 Eesrakenduse platvormipõhised moodulid

Olgugi, et Super mobiilirakendus on olemuselt kõnekaardi haldamise rakendus, siis varasemalt puudus igasugune funktsionaalsus SIM-kaartide halduseks operatsioonisüsteemi tasemel. Analüüsi tulemusel valis autor käesoleva projekti teostamiseks Android ja iOS mobiilirakenduses kasutada hübriidse lahendusena React Native teeki **react-native-sim-cards-manager** (lüh RNSCM) järgmistel kaalutlustel:

- autori kogemus platvormipõhiste mobiilirakenduste arendamisel on väike
- RNSCM teek katab täielikult esialgsed funktsionaalsed vajadused

Seadme SIM-kaartide (sealhulgas eSIM'ide) haldamiseks on võimalik arendada ka platvormipõhiseid lahendusi, kuid autor valis siiski hübriidlahenduse. RNSCM teegi meetoditest arusaamiseks ning võimalikuks silumiseks oluline põhjalikult tunda ka platvormipõhiseid mooduleid. Seetõttu on järgnevalt kirjeldatud nii RNSCM teegi kui ka iOS ja Android platvormipõhist loogikat.

react-native-sim-cards-manager

react-native-sim-cards-manager on avalikult kättesaadav ja tasuta kasutatav React Native kolmanda osapoole teek, mis on loodud seadme SIM-profiilide (sealhulgas eSIM'ide) haldamiseks. RNSCM sisaldab kolme olulist funktsionaalsust [27]:

1. `getSimCards` meetod pöördub operatsioonisüsteemi poole ning tagastab kõikide seadmes paigaldatud SIM-kaartide info koos lisainfoga operaatori ja abonendi kohta.
2. `isEsimSupported` meetod pöördub operatsioonisüsteemi poole ja tagastab tõeväärtuse kas antud seaded toetab eSIM'i paigaldamist ja kasutamist.
3. `setupEsim` meetod pöördub operatsioonisüsteemi poole ning algatab platvormipõhise eSIM paigaldamise aknastiku. Lisaks võimaldab meetod ka kuulata paigalduse sündmuseid (kas paigaldus viidi lõpuni ja kas esines katkestusi või vigu) ning sellest mobiilirakendust teavitada.

RNSCM teek sisaldab kogu vajalikku loogikat ja hoolitseb selle eest, et operatsioonisüsteemiga suhelda ning sellele uus eSIM paigaldada. Teek kasutab selleks enda sees pöördumisi platvormipõhiste moodulite Android EuiccManager ja iOS Core Telephony poole [27].

Android EuiccManager

EuiccManager on Android operatsioonisüsteemis rakendusliideste kausta `android.telephony.euicc` kuuluv klass seadme eUICC'ga seotud operatsioonide jaoks. EuiccManager sisaldab järgmiseid meetodeid [28]:

- `getEuiccInfo` meetod tagastab teavet eUICC kiibi kohta.
- `downloadSubscription` meetod algatab etteantud profiili paigaldamist eUICC-le.
- `deleteSubscription` meetod kustutab eUICC-lt etteantud identifikaatorile vastava eSIM'i.

Enne rakendusliidese ülejäänud meetodite poole pöördumist tasub veenduda, et seade üldse hõlmab endas eUICC kiipi. Selleks kasutatakse `isEnabled` meetodit. Meetodilt vastuse saamiseks on tarvis, et Androidi paketihaldurile PackageManager oleks paigaldatud *`FEATURE_TELEPHONY_EUICC`* pakett [28].

iOS Core Telephony

Core Telephony raamistik võimaldab iOS platvormil ligipääsu teabele mobiilsidepakkuja kohta SIM-kaardi alusel. Core Telephony sisaldab muuhulgas ka kahte järgnevat klassi, mis on tarvilikud eSIM'i paigaldamiseks [29]:

- `CTCellularPlanProvisioning` – klass, mille intstantsi saab kasutada eSIM'i paigaldamise ja seadistamise tarbeks
- `CTCellularPlanProvisioningRequest` – klass, mille intstantsi saab kasutada päringu kehana eSIM'i paigaldamise algatamiseks

Klass `CTCellularPlanProvisioning` hõlmab endas kahte olulist meetodit [29]:

- `supportsCellularPlan` meetod tagastab, kas antud seade toetab eSIM'i paigaldamist
- `addPlan` meetod algatab `CTCellularPlanProvisioningRequest` objekti abil kirjeldatava eSIM'i valmendamise protsessi

Klasse CTCellularPlanProvisioning ja CTCellularPlanProvisioningRequest kirjeldav dokumentatsioon on osaliselt avalik, kuid vastav API on kasutamiseks ligipääsetav vaid

mobiilsideoperaatoritele, kes on Apple'ilt taotlenud ja saanud vastava volituse (ingl *entitlement*) nimega "eSIM Development". Muuhulgas on API meetodeid võimalik mobiilirakenduses kasutada vaid juhul, kui rakenduse valmendamise profiilifail (ingl *provisioning profile*) sisaldab seda volitust. Käesoleva arenduse teostamiseks taotles klientettevõte Apple'ilt vastava volituse ning autor sai konfidentsiaalsuslepingu alusel ligipääsu ametlikule API dokumentatsioonile.

3.4 Rakenduses kasutatavad mustrid ja printsiibid

3.4.1 Mitmekihiline arhitektuur

Mitmekihiline arhitektuur (ingl *n-tier architecture pattern*) on üks levinumaid arhitektuurimustreid tarkvaraarenduses. Klient-server arhitektuuri puhul käitub server ressursside ning teenuste pakkujana ning klient ressursside tarbijana. Klient ja server vahetavad andmeid üle võrgu. Kusjuures ühe serveriga võib olla ühendatud mitu klienti, kes omakorda võivad olla ühenduses ka mitme serveriga [30]. Mitmekihilises klient-server arhitektuuris on eraldatud esitlus-, äriloogika- ja andmekiht. Kihiline arhitektuur otseselt ei defineeri kihtide täpset arvu, kuid suur osa vastavat mustrit kujutavatest rakendustest omavad kolme, nelja või enamat kihti [31].

Esitluskihiks peetakse veebibrauseris või veebipõhises rakenduses töötavat graafilist kasutajaliidest. Esitluskihi rolli täidab klientrakendus, kelle peamiseks ülesandeks on andmete esitamine kasutajale arusaadaval kujul. Äriloogikakihiks (ingl *Business Logic Layer*, lüh BLL) võib pidada serverrakendust, mis sisaldab kogu rakenduse äriloogikat. Klientrakendus suhtleb serverrakendusega rakendusliidese ehk API kaudu. BLL suhtleb andmekihiga tavaliselt läbi DAL (ingl *Data Access Layer*) ehk andmetele kättesaadavuse kihi. Andmekihiks on andmebaasisüsteem [31].

Mitmekihilise arhitektuuri üheks suurimaks tugevuseks on otstarbe lahususe printsiip (ingl *separation of concerns*). Igal kihil on rakenduses täita oma kindel roll ja vastutus. Teisisõnu, üks kiht ei pea teadma teise kihi sees toimuvast. Mida piiritletum on iga kihi skoop, seda kergem on iga kihti eraldiseisvalt hallata, täiendada, välja vahetada ja testida ilma teisi kihte mõjutamata [32]. See tunnus muudab rakenduse skaleerimise lihtsamaks, sest skaleerida saab vastavalt vajadusele kas klient- või serverrakendust [31].

Käesolevas töös on esitluskihiks React Native rakendus ja äriloogikakihiks Spring rakendus. Kuna antud projekti skoobi raames ei täiendata olemasolevat Super rakenduse andmekihti, siis pole käesolevas töös andmekihiga seotud loogikat käsitletud.

3.4.2 REST API

REST (ingl *Representational State Transfer*) on süsteemide vahelist kommunikatsiooni standardiseeriv arhitektuurilaad. Selle abil on võimalik arendada teenuseid erinevate rakenduste toimimiseks vajalike ressursside pärimiseks ja lisamiseks. REST on olemuselt olekuta, mis tähendab, et klient ja server on eraldatud ning üks ei ole mõjutatud olekust, milles on parasjagu teine [33].

REST API on rakendusliides, mis kasutab andmete juurdepääsuks standardseid HTTP-päringuid, et tuvastada levinumate verbidega tähistatud GET, POST, PUT ja DELETE meetoditel andmete edastamist [34].

Klient- ja serverrakenduse omavahelise suhtluse tagamiseks tuleb määrata, millisel kujul andmeid edastatakse ja vastu võetakse. REST liidese kasutamiseks määratakse kindlad **URL** (ingl *Uniform Resource Locator*) lõpp-punktid, mille pihta saab klientrakendus serverrakendusele päringuid teha [34]. Andmevahetuseks saadab klient serverile HTTP päringu, mis sisaldab HTTP verbi, päist, ressursi teekonda ning vajadusel ka lasti. Server vastab päringule, märkides lasti sisu tüübi, staatuse koodi ning vajadusel lasti. Mitmele erinevale HTTP verbile võib vastata täpselt sama URL [33].

REST teenuste puhul võib andmevahetus toimuda erinevates andmeformaatides: JSON, XML (ingl *Extensible Mark-Up Language*) või CSV (ingl *Comma-separated values*). REST eelis teise levinud arhitektuurilaadi SOAP (ingl *Simple Object Access Protocol*) ees seisneb selles, et REST suudab andmeid vahendada nii XML, JSON kui ka CSV formaadis, kuid SOAP võimaldab realiseerida ainult XML formaadis andmevahetust [34]. Käesolevas töös kasutatakse JSON formaati.

3.4.3 Kontrolli inversioon ja sõltuvuste süstimine

Käesolevas töös arenduses kasutatav Spring raamistiku alustalade hulka kuulub kontrolli inversiooni (ingl *Inversion of Control*, lüh **IoC**) printsiip. IoC eesmärk on luua lõdvalt ühendatud klassid, mida on võimalik kergevaevaliselt asendada, laiendada ning testida

[35]. Spring IoC konteiner haldab Spring *bean*-nimelisi objekte, millest koosneb Spring rakendus. Java rakenduses on Spring bean objekt tähistatud `@Bean` annotatsiooniga [36].

Sõltuvuste süstimise (ingl *Dependancy Injection*, lüh **DI**) printsiip on tihedalt seotud IoC printsiibiga. Springi `@Autowired` funktsionaalsus võimaldab Springil automaatselt tekitada ühendusi Spring *bean*'ide vahel [35]. Teisisõnu, ühe Spring *bean*'i sõltuvus teisest on lahendatud DI abil.

3.4.4 Koodi stiil

Käesoleva arenduse puhul võttis autor eesmärgiks järgida üldise koodi struktuuri ja disaini põhimõtteid:

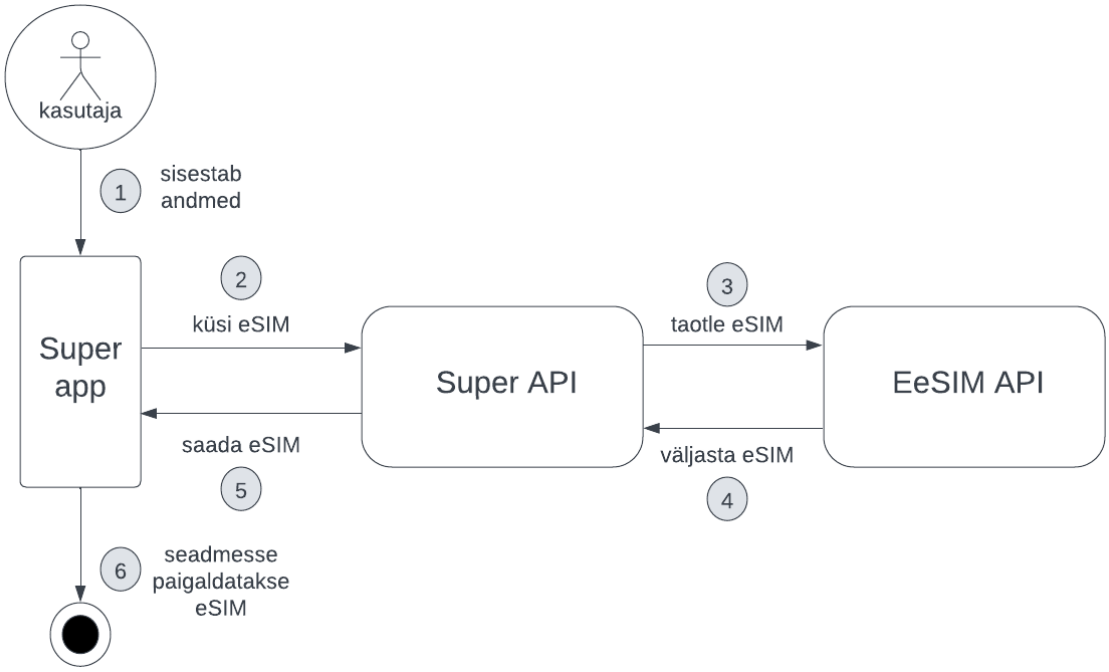
1. Otstarbe lahusus – koodi tükeldamine kindla otstarbega eraldiseisvateks tükkideks, mida on kerge hallata ja välja vahetada [32].
2. Üksikvastutuse printsiip (ingl *single responsibility principle*) – igal kooditükil, näiteks meetodil, peab olema vaid üks konkreetne ülesanne, mida see täidab [37].
3. Koodi taaskasutus – koodi tükeldamine taaskasutatavateks protseduurideks, et vältida duplikaatkoodi kirjutamist. Tuntud ka kui DRY („*Don't Repeat Yourself*“) printsiip [38].
4. Liiasuse vältimine – mitte arendada teatud funktsionaalsust igaks juhuks, kui see otseselt pole antud projekti teostamiseks vajalik. Tuntud ka kui YAGNI (ingl „*You Aren't Gonna Need It*“) printsiip [39].
5. Muutujate ja meetodite selge nimetamine – kogu oluline informatsioon muutuja või meetodi funktsionaalsuse kohta peab ilmnema ainuüksi sellele antud nimest. Seeläbi välistatakse vajadus lisada koodile kommentaaride kujul kirjutatud dokumentatsiooni.
6. „*Tell-Don't-Ask*“ printsiip – objektorienteeritus tähendab muuhulgas seda, andmed ja käitumine on omavahel põimunud. Teisisõnu, vaid objekti enda andmetega opereerivad funktsioonid peaksid olema objekti enda valduses, mitte temast väljaspool. Selmet objekti käest küsida tema andmevälju ja neid töödelda, tuleks hoopis objektile öelda, kuidas käituda. See printsiip julgustab arendajat paigutama objektiga seonduvad käitumist objekti enda külge [40].

Loetletud disaini põhimõtteid ja printsiipe on järgitud nii taga- kui eesrakenduse programmikoodi loomisel. Neid on oluline järgida, et koodil säiliks hea loetavus, hallatavus ning silumise hõlpsus.

4 Rakenduse mooduli arendamine

Eesmärgi saavutamiseks tuleb Super rakendusele lisada nii taga- kui eesrakenduse moodul. Eesrakenduse mooduli arendamine põhineb Super mobiilirakendusele eSIM liitumisteekonna loomisel. Tagarakenduse mooduli arendamine põhineb Super API ja välise EeSIM API vahelise andmevahetuse loomisel. EeSIM API on klientettevõtte eelnevalt arendatud ja hallatud API, mis võimaldab klientettevõtte pakutud eSIM'e hallata (broneerida, alla laadida ja infot pärida). EeSIM API't on antud töös käsitletud kui „musta kasti" süsteemi (ingl *black box system*), mille puhul on oluline teada vaid süsteemi antavaid sisendeid ja saadavaid väljundeid. EeSIM API süsteemi sees täpselt toimuv pole antud kontekstis oluline.

Arendatava süsteemi üldine loogika on toodud joonisel 6. Kasutaja, kes soovib liituda Super eSIM teenusega, sisestab liitumisteekonna jooksul vajalikud andmed (joonisel tähistatud sammuna „1"). Eduka ostu puhul pöördub Superi mobiilirakendus Super API poole (joonisel tähistatud sammuna „2"), kes omakorda taotleb eSIM'i väliselt EeSIM API'lt (joonisel tähistatud sammuna „3"). EeSIM API väljastab eSIM'i Super API'le (joonisel tähistatud sammuna „4"), mis teostab vajaliku andmetöötluse ning edastab eSIM'i Super mobiilirakendusele (joonisel tähistatud sammuna „5"). Seejärel toimub mobiilirakenduse vahendusel eSIM'i paigaldamine kasutaja seadmesse (joonisel tähistatud sammuna „6"). Sellega on liitumisteekond lõppenud.



Joonis 6. Rakenduse süsteemi ülesehitus.

4.1 Tööde planeerimine

Suuremahuliste agiilsete arendusprojektide haldamiseks on kasulikuks tööriistaks projektihaldustarkvara. See võimaldab planeerida töid ja aega ning viia läbi analüüse. Töid saavad kaardistada nii projektijuhid, analüütikud, arendajad kui ka testijad. Projektihaldustarkvara võimaldab näiteks luua iga kasutajaloo kohta oma alamülesanne ning seeläbi jagada projekti väiksemateks tükkideks [41]. Klientettevõte on käesoleva arenduse analüüsi ja planeerimise tööriistaks varasemalt valinud **Jira**, mida autor kasutab ka käesoleva projekti haldamiseks.

Käesolev arendus teostati mitmeliikmelise meeskonnana, kelle hulka kuulusid projektijuht, testija, kaks arendajat ning arhitekt. Arendajad jagasid arendusülesanded omavahel vastavalt koormusele. Autor oli üks arendajatest. Arhitekt nõustas arendajaid, aidates valida sobilikke võtteid ning teostades koodi ülevaatusi. Testija teostas manuaalseid teste rakenduse mooduli valmimisjärgus ning viis läbi regressiooniteste. Lisaks osales projekti analüüsis, teostamises ning konsulteerimises veel klientettevõttest mitmeid spetsialiste ja meeskondi.

Arenduseks vajalike tööde kaardistamiseks loodi Jira projektihalduskeskkonda uus *epic*. *Epic* on suur kasutajalugu, mis kirjeldab nõuet, mida hakatakse arendama. Projekti jagamine mitmeks pisemaks tükiks on oluline, et luua arendustööde jaoks struktuuri. Lisaks aitab see saada tööde mahust parema ülevaate ning anda kliendile täpsemaid ajahinnanguid. Tööde järjekorda seadmine on oluline, sest näiteks mõni funktsionaalsus klientrakenduses võib tingimata eeldada serverrakenduse teatud funktsionaalsuse olemasolu. Käesoleva arenduse *epic* pealkirjaga „Super eSIM arendus“ jaotati käesoleva arenduse raames järgmisteks alamülesanneteks:

- Android eSIM võimekuse kontroll;
- iOS eSIM võimekuse kontroll;
- TypeScript lisamine Super mobiilirakendusse;
- Vaadete ja navigatsiooni loomine;
- eSIM book & pull liidestus;
- Mitme eSIM ja mobiilinumbri broneerimine;
- Makselahenduse teostamine;
- Seadmes eSIM paigaldamine;
- Kasutaja e-postile paigaldamise juhendi saatmine;

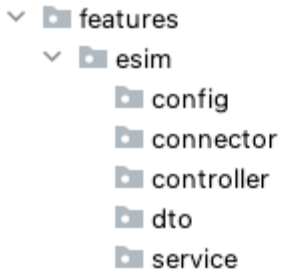
Arendustööde loodi peatükis 3.1 loetletud kasutajanõudeid silmas pidades. Järgnevalt on kirjeldatud, kuidas planeeriti ülesannet „*Mitme eSIM ja mobiilinumbri broneerimine*“ funktsionaalse kasutajanõude „*Kasutajana soovin eSIM'i soetades valida mitme erineva mobiilinumbri vahel, et leida endale meelepärane ja meeldejääv number*“ alusel. Esmalt defineeriti tööle eelduseks olevad ülesanded. Nende hulka kuulub näiteks „*eSIM book & pull liidestus*“. Seejärel analüüsiti, kuidas on tarvis täiendada serverrakenduse teenuse komponenti, toetamaks mitme eSIM broneerimise funktsionaalsust. Viimaks kirjeldati täpne tehniline lahendus, kuidas tulemuseni jõuda.

Iga alamülesande puhul määrati kategooria, millise rakenduse tüübi alla see käib (mobiilirakendus või tagarakendus) ning rollid. Lisaks kirjeldati iga ülesande puhul nõuetele vastavuse kriteeriumid (ingl *acceptance criteria*), mille alusel saab ülesannet lugeda sooritatuks. Projekti liikmed said Jira keskkonnas jälgida ülesannete staatust, ajakulu ja tulemusi. Rakenduse testija sai ülesande juurde anda tagasisidet.

4.2 Tagarakenduse mooduli arhitektuur

Tagarakenduse mooduli loomine põhineb Super API ja välise EeSIM API vahelise andmevahetuse arendamisel. Autorile on antud ligipääs EeSIM API dokumentatsiooniga tutvumiseks. Super tagarakenduse üldine seadistus on antud koodivaramus juba eelnevalt paika pandud ning seda käesolevas töös antud arenduse raames detailselt ei kirjeldata.

Tagarakendus on funktsionaalsuse alusel jaotatud *features* kaustades tükkideks. Käesoleva arenduse jaoks on *features* kausta loodud *esim* nimeline kaust, mis on omakorda jagatud viite alamkausta. Tagarakenduse mooduli failipuu on toodud joonisel 7.



Joonis 7. Tagarakenduse mooduli failipuu ülesehitus.

`Config` kaust sisaldab klassi `EeSimApiParinguVaheleSekkuja`, mis „sekkub vahele“ iga EeSIM API poole pöörduva päringu puhul. See klass hoolitseb selle eest, et iga EeSIM API päring oleks varustatud korrektse päisega (ingl *header*). Iga päringu päises sisaldub ka ligipääsutunnus, mis on tarvilik eduka päringu jaoks. Ligipääsutunnus on kindla kehtivusajaga. Ühe ja sama võtmega sooritatakse nii palju päringuid, kui võtme kehtivusaeg võimaldab. Uue võtme genereerimiseks on vastav mehhanism, mis uuendab võtme väärtust, kui eelmine võti on kehtivusaja kaotanud. `EeSimApiParinguVaheleSekkuja` klass on tähistatud `@Component` annotatsiooniga, et Spring raamistik teaks rakenduse käivitamisel käsitleda klassi kui Springi *bean*’i.

Kõik Super API päringud EeSIM API poole nõuavad identsel kujul päringu päiseid. Siinjuures täidabki `EeSimApiParinguVaheleSekkuja` olulist rolli. Kui taolist päringutele vahele sekkuvat klassi poleks, siis peaksid teised EeSIM API päringuid välja kutsuvad klassid igaüks muretsema päiste ja autentimisvõtme genereerimise eest. Seega

on kasulik see loogika paigutada ühteainsasse klassi, siludes koodi stiili ja vähendades duplikaatkoodi kirjutamise vajadust.

`Connector` nimeline kaust sisaldab `EeSimApiUhenduja` nimelist klassi. **RestTemplate** on Spring raamistiku peamine API, mille abil sünkroonseid HTTP päringuid sooritada. RestTemplate aitab arendajal vältida trafarettkoodi kirjutamist, hoolitsedes ise järgmiste tegevuste eest HTTP päringu sooritamiseks [42]:

1. URL objekti
2. Ühenduse algatamine ja peatamine
3. HTTP päringu seadistamine
4. HTTP päringu sooritamine
5. HTTP päringu vastuse tõlgendamine
6. HTTP päringu vastuse keha
7. Veakäsitlus

Klassis `EeSimApiUhenduja` on defineeritud EeSIM API ligipääsupunktid iga vajaliku päringu sooritamise jaoks. Lisaks on klassisiseselt realiseeritud keskne veahaldus, mida kasutavad kõik klassi meetodid. Klass `EeSimApiUhenduja` on toodud lisas 2.

`Controller` kaust sisaldab kontrolleri klassi `ESimKontroller`, mille ülesandeks on suhelda klientrakendusega. `@RestController` annotatsioon klassi küljes tagab selle, et kontrolleri meetodi tagastatav objekt saadetakse HTTP päringu vastusena.

Rakenduse mooduli arendamise jaoks loodi üks kontroller, mis sisaldab kõiki klientrakenduse jaoks vajalikke ligipääsupunkte. Tabelis 1 on toodud kontrolleris `ESimKontroller` määratletud ligipääsupunktid, mille poole saab klientrakendus HTTP päringuid teha.

Tabel 1. API ligipääsupunktid.

Marsruut	**HTTP meetod**	**Kirjeldus**
/teave/{kood}	GET	ICCID alusel konkreetse väljastatud eSIM'i kohta info pärimine
/broneerimine/{arv}	GET	Kindla arvu eSIM'ide ja mobiilinumbrite broneerimine
/allalaadimine/{kood}	GET	ICCID alusel eSIM'i aktiveerimine ja allalaadimine

`DTO` nimeline kaust sisaldab andmesaateobjekte (ingl *Data Transfer Object*, lüh DTO), mida kasutatakse andmete vahetamiseks kindlal kujul nii serverrakenduse siseselt kui ka klient- ja serverrakenduse vahelises suhtluses. DTO objektid on kasulikud näiteks olukorras, kus klientrakendusele pole tarvis saata kõiki andmevälju, mida kasutab teenuskiht või andmekiht. Seeläbi saab rakenduse andmeliikluses vähendada andmete liiasust.

Autor on valinud teostada DTO objektide valideerimised vastava DTO klassi sees, mitte sellest väljaspool, näiteks *Service* klassis. Kirjeldatud lähenemine aitab koodi stiili puhul järgida „*Tell-Don't Ask*" printsiipi, sidudes objekti funktsionaalsuse objekti endaga. Andmesaateobjekti struktuur ja „*Tell-Don't Ask*" printsiibi kasutamise näide on toodud lisas 3.

Andmesaateobjektide loomisel on autor valinud kasutada **Builder** mustrit. Builder on Lombok teeki kuuluv funktsionaalsus. Builder muster aitab objekti klassi konstruktori mõtteliselt „tõsta klassist välja" eraldiseisvateks builder objektiks. Seeläbi on objekti võimalik ehitada võimalikult loetavalt, paindlikult ja järkjärgult, vältides ühe või mitme erineva konstruktori eraldi defineerimist. Builder funktsionaalsusega klassid märgistatakse `@Builder` annotatsiooniga.

`Service` kaust sisaldab teenuse klassi `ESimTeenus`. `@Service` annotatsioon märgib klassi teenusekihi osaks ning näitab, et antud klass sisaldab äriloogikat. Teenusekiht hõlmab endas kogu rakenduse äriloogikat ning on ainuke kiht, mille kaudu saab antud

rakenduses algatada andmebaasi- või väliste API'de päringuid. Välise EeSIM API päringute jaoks tarvilik `EeSimApiUhenduja` sõltuvus on süstitud teenuse klassi kasutades `@Autowired` annotatsiooni, järgides sellega DI printsiipi. Teenuse klass on toodud lisas 4.

Teenuse klassi meetodite poole pöördutakse vaid kontrollerist. Teenuse klass vahendab välise API päringuid, töötleb nende vastuseid (sorteerimine, filtreerimine), sooritab andmesaateobjektide teisendust ning vigade ilmnemisel erinditest (ingl *exception*) teavitamist. Teenuse klassi meetodid on nimetatud selgelt ja ühemõtteliselt, välistades kommentaaride kujul dokumentatsiooni lisamise vajaduse.

4.3 Tagarakenduse arendamine TDD metoodikaga

Järgnevalt on kirjeldatud ühte näidet, mille jooksul arendatavat rakendust täiendati järgides TDD „Punane-Roheline-Refaktoreerimine" tsükli põhimõtet.

Ülesandepüstitus

Iga EeSIM API poole pöörduva päringu päisele tuleb kaasa anda API ligipääsutunnus (ingl *access token*), mis on tarvilik eduka päringu jaoks. Ligipääsutunnus on kindla kehtivusajaga. Uut ligipääsutunnust on võimalik küsida eraldi päringu abil. Selmet lasta EeSIM API'l iga päringu jaoks uut ligipääsutunnust genereerida, saaks kehtivusaja jooksul kasutada olemasolevat võtit. Ühe ja sama võtmega tuleks sooritada nii palju päringuid, kui võtme kehtivusaeg võimaldab. Uus ligipääsutunnus tuleb küsida vaid juhul, kui see on veel väärtustamata või on muutunud kehtetuks.

Testi kirjutamine

Joonisel 8 on toodud autori kirjutatud test nimega `testVotitParitakseVaidUksKord()`, mis kontrollib, kas kolme järjestikuse API päringu jooksul päritakse EeSIM API'st uut ligipääsutunnust täpselt ühe korra.

```
@Test
public void testVotitParitakseVaidUksKord () {
    eSimTeenus.broneerimine(1);
    eSimTeenus.allalaadimine(iccid);
    eSimTeenus.broneerimine(1);

    verify(exactly(1),
    postRequestedFor(urlEqualTo("/v1/eesim/token")));
}
```

Joonis 8. Ligipääsutunnuse genereerimist kontrolliv test.

Samm „Punane“

Kuna eelnevalt oli programmikood seadistatud nii, et enne igat eSIM'i broneerimise ja allalaadimise päringut küsis rakendus EeSIM API'st uue ligipääsutunnuse, siis see test ebaõnnestus.

Samm „Roheline“

Testi läbimiseks tuli ümber teha klassi `EeSimApiParinguVaheleSekkuja`, mis vastutab ligipääsutunnuse pärimise eest. Selleks lisas autor klassile kaks kohalikku muutujat, mille abil saab rakendamise käitamise jooksul ligipääsutunnuse väärtust ja kehtivusaega mälus meeles hoida ning kasutada.

- `public String voti` – selles muutujas hoitakse ligipääsutunnuse väärtust
- `public Instant kehtivus` – selles muutujas hoitakse ligipääsutunnuse genereerimise hetke ajalist väärtust

Järgnevalt lõi autor uue klassimeetodi `annaVoti()`, mille eesmärgiks on otsustada, kas saab kasutada veel kehtivat ligipääsutunnust või tuleb pärida uus ning viimaks tagastada kehtiv ligipääsutunnus. Meetod on toodud joonisel 9. Koodi täiendati vaid nii palju kui vaja, et test edukalt läbima hakkaks. Käesolevas sammus ei pööratud kuigi palju tähelepanu koodi stiilile. Ainukeseks eesmärgiks oli test läbida.

```
private synchronized String annaVoti() {
    if (voti == null) || voti.equals(“”) || kehtivus == null ||
    Instant.now().isAfter(kehtivus)) {
        VotiDto dto = pariVoti();
        voti = dto.getToken();
         kehtivus = Instant.now().plusSeconds(dto.getExpiry() - 15);
    }
    return voti;
}
```

Joonis 9. Ligipääsutunnuse genereerimine enne refaktoreerimist.

Seejärel lisati klassis `EeSimApiParinguVaheleSekkuja` meetodi `annaVoti()` väljakutse meetodile `intercept()`. Järgmisel korral algset testi käitades test õnnestus. Seega loodi edukalt mehhanism, mis tagastab olemasoleva kehtiva võtme või uuendab võtme väärtust, kui eelmine võti on kehtivusaja kaotanud või ei ole veel väärtustatud. Esialgne nõue sai täidetud.

Samm „Refaktoreerimine“

Kuna eelmises sammus ei pööranud autor tähelepanu kirjutatava koodi stiilile ning ainsaks eesmärgiks oli testi läbimine, siis tuli seejärel hinnata äsja lisatud koodi. Kirjutatud meetod `annaVoti()` osutus lohiseva *if*-klausli tõttu raskesti loetavaks. Meetodi parendamiseks tõstis autor väärtusi kontrollivad operatsioonid eraldi meetoditesse ning neile anti üheselt arusaadavad nimetused. Refaktoreeritud kood on toodud joonisel 10.

```
private synchronized String annaVoti() {
    if (StringUtils.isEmpty(voti) || votiOnAegunud()) {
        VotiDto votiDto = pariVoti();
        uusVoti = votiDto.getToken();
        seaUusKehtivus(uusVoti);
    }
    return voti;
}

private boolean votiOnAegunud() {
    return kehtivus == null || Instant.now().isAfter(kehtivus);
}

private void seaUusKehtivus(VotiDto votiDto) {
    kehtivus = Instant.now().plusSeconds(votiDto.getExpiry() - 15);
}
```

Joonis 10. Ligipääsutunnuse genereerimine pärast refaktoreerimist.

Refaktoreerimise jooksul käivitati mitmel korral algne test ning veenduti, et koodi õige käitumine protsessi käigus tõepoolest säilis.

Järeldus

Kokkuvõttes täideti algne nõue ja kaeti käitumine testidega. Seejuures muudeti ligipääsutunnuse kehtivuse üle otsustamise loogikat loetavamaks, kergemini testitavamaks ja tulevikus paremini hallatavamaks. Need tulemused ühtivad TDD metoodika sisuliste eesmärkidega, tõestades sellega metoodika kasulikkust antud rakenduses ning tarkvaraarenduses üldisemalt.

4.4 Eesrakenduse mooduli arendamine

Eesrakenduse mooduli loomine tähendab React Native platvormil arendatud Superi mobiilirakenduse täiendamist. Mooduliks on liitumisteekond, mille läbides saab kasutaja endale soetada ja seadmesse paigaldada Super kõnekaardi eSIM'i. Mobiilirakenduse üldine ülesehitus on antud koodivaramus juba eelnevalt seadistatud ning seda käesolevas töös detailselt ei kirjeldata. Järgnevalt on kirjeldatud olulisemaid aspekte, mis kuulusid mobiilirakenduse mooduli arenduse sammude hulka.

Disain

Eesrakenduse disainijoonised oli arendusele eelduseks, sest need aitasid planeerida kogu arendatava eesrakenduse mooduli ülesehitust. Mobiilirakenduse prototüübijoonised andis arendusmeeskonnale ette klientettevõte. Klientettevõtte disainer oli Figma tööriistaga joonistanud ette kõik vajalikud ekraanivaated, komponendid, värviskeemid, tekstistiilid ja ikoonid kõikide vajalike parameetritega. Seejärel sai autor alustada mobiilirakenduse mooduli arendamisega.

Komponendid

Moodulis on kasutusele võetud React'i funktsionaalsed komponendid. Need on JavaScripti funktsioonid, mis võtavad sisendparameetreid ning tagastavad React elemendi. Antud arendus moodulis tagastatakse TSX komponente. Komponendid on taaskasutatavad kooditükid üle terve rakenduse. Üks komponent võib sisaldada mitut teist komponenti. Arendatud mooduli sekka loodi juurde kaheksa uut ekraanikomponenti, millest igaüks esindas ühte liitumisteekonna sammu. Need ekraanivaated sisaldasid

omakorda väiksemaid taaskasutatavaid komponente: tekstivälju, sisestusvälju, nuppe, animatsioone.

Navigatsioon

Mobiilirakenduse juurde arendatud mooduliks on liitumisteekond, mille läbides saab kasutaja endale soetada ja seadmesse paigaldada Super kõnekaardi eSIM'i. Liitumisteekonnaks on kaheksast erinevast ekraanist koosnev jada. Selleks võeti kasutusele **react-navigation/stack** teek, mis võimaldab luua navigatsiooni erinevate vaadete vahel. Eesmärgiks oli, et kasutaja saab liigelda vabalt edasi ja tagasi kõikide sammude vahel, mis on vaja läbida teenusega liitumiseks. React-navigation/stack teegi kasutamine antud rakenduses on toodud lisas 5.

React Hooks ja React Query

Arendatavas moodulis hoolitsevad funktsionaalsete komponentide oleku haldamise eest *hook*'id. Neid kasutades sai teisaldada komponendi olekuga seotud loogika React'i elemendist välja omaette funktsiooni. Tänu sellele paranes koodi loetavus ning suurenes koodi taaskasutatavus. Lisas 6 on toodud eSIM broneerimise *hook*'i `useBookEsim()`. See võtab sisendparameetriks arvu, mitu eSIM'i on tarvis broneerida ja tagastada ning tagastab objekti erinevatest muutujatest ja funktsioonidest. Sellel *hook*'il on kaks peamist ülesannet:

1. hoiustada muutuja `selectedNumber` olekut ja
2. vahendada eSIM broneerimisele vastava API päringu tegemist ja selle vastuse tulemusi

React Query roll on vahendada andmete pärimise protseduure ja uuendada komponentide sees andmeid. Lisaks aitab React Query hoiustada ja hallata päringute ja nende tulemuste olekut. Lisas 6 toodud näites on teek kapseldatud funktsiooni `useAppQuery()` sisse. Funktsiooni sisendparameetriks on objekt. Objekti esimene väli *queryFn* on funktsioon, mille väärtuseks on funktsioon `bookESim()`, mis pöördub API poole eSIM broneerimise päringu sooritamiseks. Objekti teiseks väljaks on valikuline parameeter *enabled*, mille tõeväärtus on siinjuhul väär. See tähendab, et funktsiooni `queryFn` väärtust vaikimisi ei käivitata ning käivitatakse alles siis, kui kutsutakse välja React Query funktsiooni `refetch()`. React Query parameeter `data` väärtustatakse `queryFn` funktsiooni päringu vastusega. Kui API päring peaks tagastama erindi, siis

püüab React Query selle kinni ning väärtustab parameetri `isError` tõeseks. Nii data, refetch kui isError sisalduvad *hook*'i `useBookEsim()` tagastavas objektis, muutes need ligipääsetavaks üle terve rakenduse.

react-native-sim-cards-manager

Teegil react-native-sim-cards-manager on arendatavas moodulis kaks peamist ülesannet:

1. teostada seadme eSIM võimekuse kontroll, kasutades teeki kuuluvat funktsiooni `isEsimSupported()`, mis tagastab tõeväärtuse, kas seadmel on eSIM tugi või mitte. Kui tagastatud vastus oli tõene, siis juhatati kasutaja edasi järgmisele ekraanile. Eitava vastuse korral kuvati kasutajale veateadet ning juhatati ta tagasi eelnevale ekraanile. Funktsioon on toodud lisas 7.
2. algatada seadmesse eSIM paigaldamine, kasutades teeki kuuluvat funktsiooni `setupEsim()`. See võtab sisendparameetriteks eSIM allalaetavale profiilile vastava QR-koodi (ingl *Quick Response*, lüh QR) aadressi. Funktsioon tagastab tõeväärtuse, kas eSIM paigaldamine seadmesse lõppes edukalt või mitte. Kui paigaldamine oli edukas, kuvati kasutajale vastavat teadet ning juhatati ta edasi rakendusse sisselogimise ekraanile. Vastasel juhul kuvati kasutajale vastavat veateadet ning kasutajal avanes uus võimalus eSIM paigaldamist algatada.

4.5 Rakenduse testimine

Tagarakenduse moodul sai TDD metoodika kasutamise käigus kaetud ulatuslikult nii ühik- kui integratsioonitestidega. Tagarakenduse teenuse funktsionaalsus kaeti 100% määral testidega. See-eest eesrakendust testiti käsitsi ja vastavad ühiktestid puuduvad. Super rakendusel polnud enne käesoleva projekti alustamist mobiilirakenduse automaattestimise tööriistasid integreeritud ning see ei mahtunud ka projekti skoopi. Arendusmeeskonna testija testis mobiilirakendust käsitsi kogu arenduse vältel, et veenduda kasutajaliidese funktsionaalsuses ning leida erijuhtudel esinevaid vigu. Sellega on täidetud rakenduse testimise minimaalsed nõuded.

5 Tulemuste analüüs

Käesoleva lõputöö skoobiks oli Super mobiilirakendusele eSIM paigaldamise toe loomine. Projekti ja mobiilirakenduse uuenduse turustamise esialgne tähtaeg oli 1.mai 2023. Seoses teiste arendustöödega Super rakenduses 2023.aasta kevade vältel langes antud projekti arendustööde prioriteetsus. Kuna arendusmeeskonna ajalised ressursid olid piiratud, siis otsustas klientettevõte projekti tähtaega edasi lükata 2023.aasta 1.juuliks. Projekt ei ole seega veel oma elutsükli lõpus, kuid arendus on teostatud ning selle põhjal saab analüüsida seniseid tulemusi. Tehtud töö väärtust on järgnevalt kirjeldatud kasutajanõuetele vastavuse, pideva integreerimise kirjelduse ja arenduspraktikatele vastavuse abil.

5.1 Rakenduse vastavus kasutajanõuetele

Käesoleva töö kirjutamise hetkeks on projektis teostatud infosüsteemi (rakenduse mooduli) analüüs ja kõik arendusülesanded. Autor teostas vajalikest arendusülesannete töömahust hinnanguliselt 80% ning oli antud projektis põhiline arendaja. Autor teostas peatükis 4.1 *epic* „Superi eSIM arendus“ all loetletud kõikidest arendusülesannetest järgmised:

- iOS eSIM võimekuse kontroll;
- Vaadete ja navigatsiooni loomine;
- eSIM book & pull liidestus;
- Mitme eSIM ja mobiilinumbri broneerimine;
- Makselahenduse teostamine;
- Seadmes eSIM paigaldamine;
- Kasutaja e-postile paigaldamise juhendi saatmine.

Peatükis 3.1 loetletud kasutajanõuded kirjeldasid rakenduse kasutajate ootusi, vajadusi ja eesmärke ning nõudeid, mida rakendus peab täitma. Arendustööde planeerimiseks võeti aluseks kõik funktsionaalsed ja mittefunktsionaalsed kasutajanõuded. Selle eesmärk oli

täpselt määratleda, milliseid nõuded peab Super mobiilirakenduse eSIM liitumisteekonna moodul täitma.

Rakenduse moodul rahuldab kõiki funktsionaalseid kasutajanõudeid. Rakendus vastab kõigile nõuetele, milliseid toiminguid kasutaja saab rakenduses teha, milliseid sisendeid kasutaja saab anda ja milliseid väljundeid rakendus annab. Rakenduse moodul rahuldab ka kõik mittefunktsionaalsed kasutajanõuded. Rakendus vastab täpselt klientettevõtte disainijoonistele, tehes kasutajaliidesest kasutajasõbraliku mobiilirakenduse. Lisaks on arvestatud nõudeid seoses rakenduse reageerimiskiirusega, jõudlusega ja andmekaitsega.

Kasutajanõuded aitasid tagada, et rakendus vastab kasutajate vajadustele ja eelistustele. Lisaks aitasid need vältida tulevasi probleeme ja kulukaid muudatusi, sest selgitasid autorile juba enne arendusprotsessi algust, millised on kasutajate ootused ja mida nad tegelikult vajavad. Arendatud rakenduse moodul rahuldab autori hinnangul kõik ettenähtud funktsionaalsed ja mittefunktsionaalsed nõuded.

5.2 Rakendus CI süsteemis

Pideva integreerimise protsess (ingl *Continuous Integration*, lüh CI) on protsess, mis hõlmab koodi regulaarset integreerimist ja testimist automaatsete vahendite abil. Selle protsessi tulemusena tekib automaatne koodi kokkupanek ja testimine, mis võimaldab varakult avastada vigu või probleeme. CI protsess sisaldab tavaliselt mitmeid samme, nagu koodihoidla kontrollimine, programmikoodi kooste, testimine ja vajadusel ka koodi käivitamine või paigaldamine erinevatesse keskkondadesse.

Arendatud Superi taga- ja eesrakenduse moodulid on integreeritud **Bamboo** CI süsteemi. Rakenduse testid käitatakse CI süsteemis iga kooste vältel. See omadus võimaldab läbikukkunud testidest kiiresti teada saada ning tekkinud vead parandada. Rakenduse kooste õnnestub CI süsteemis ilma esinevate vigadeta ja hoiatusteta.

Super rakendused on paigaldatud erinevatesse teineteisest eraldatud keskkondadesse. Rakenduse moodul on käesolevaga paigaldatud Super rakenduse testkeskkonda. See võimaldab testijatel Superi testrakenduse kaudu funktsionaalsust testida. Testrakenduses testimine ei mõjuta avalikult kättesaadavat ja Superi klientidele kasutatavat tootmiskeskkonda (ing *live environment*), sest need on teineteisest eraldatud.

Super rakenduse arendatud moodulit saab pidada tarnitavaks lahenduseks. Rakendus läbib edukalt kooste ning on integreeritud CI süsteemis.

5.3 Rakenduse vastavus arenduspraktikatele

Klientettevõte on seadnud uutele Super projekti arendustele **testidega katvuse nõude**. Kogu arendatud tagarakenduse mooduli teenuskiht on kaetud ühik- ja integratsioonitestidega 100% määral. TDD mustri kasutamine tagarakenduse arendamisel on osutunud kasulikuks metoodikaks. Testikava kirjutamine enne programmikoodi kirjutamist aitas koodi disainida võimalikult kergesti arusaadavaks, hallatavaks, silutavaks ja testitavaks. Programmikoodi kõrge testide katvuse määr annab ka kindluse, et tulevased võimalikud muudatused koodis ei tee endist funktsionaalsust katki. Ka edasistes Super projekti arendustes on kavas edaspidi TDD metoodikat kasutada. Seega on autor rakenduse arendamisel täitnud testidega katvuse nõude.

Lisaks on klientettevõte seadnud Super projekti arendustele ka **dokumentatsiooni** nõude. Arendatud tagarakenduse mooduli dokumentatsioon on korrektselt vormistatud ja esitatud klientettevõtte töökeskkonnas **Confluence**. Loodud dokumentatsiooni hulka kuuluvad järgmised komponendid:

1. API kirjeldus: Selgitus selle kohta, millised API päringud on saadaval, kuidas neid päringuid kasutada ning milliseid andmed need päringud tagastavad.
2. Andmemudelid: Selgitus andmemudelite struktuurist, sh millised andmed on saadaval, kuidas need andmed on seotud ja milline on nende vaheline suhe.
3. Serverikeskkonna konfiguratsioon: selgitus selle kohta, millised teenused on kasutusel ja millised konfiguratsioonifailid on muudetud.
4. Tarkvara sõltuvused: Selgitus tarkvara sõltuvustest, sh millised raamistikud, programmeerimiskeeled ja muud tehnoloogiad on kasutusel ning millised versioonid neist on vajalikud rakenduse käivitamiseks.
5. Turvanõuded: Selgitus turvanõuetest, sh millised meetmed on rakendatud turvalisuse tagamiseks ja millised on turvanõuetele vastavuse tagamiseks vajalikud protseduurid.
6. Paigaldus- ja käivitamisjuhised: Selgitus selle kohta, kuidas rakendus tuleks paigaldada ja käivitada, sh millised sammud on vajalikud serveri ettevalmistamiseks, rakenduse konfigureerimiseks ja käivitamiseks.

Arendatud rakenduse moodul vastab levinumate kui ka klientettevõttega kokkulepitud arendustavadele ja -printsiipidele. Rakendus on korrektselt dokumenteeritud ning kaetud testidega piisaval määral. Super rakendus vastab seega parimate tarkvaraarenduse praktikate nõuetele.

5.4 Edasiarenduse võimalused

Tulemuste põhjal saab kirjeldada, millised edasised sammud on ette nähtud projekti elutsüklis. Kõik arendusülesanded läbivad järgmisena süsteemitestimise faasi. Valminud mobiilirakenduse lahendus annab klientettevõte seejärel testida kindlale hulgale beetakasutajatele. Kasutajatestidest laekunud tagasisidet analüüsitakse ning vastavalt vajadusele parendatakse lahendust. Seejärel levitatakse lahendust laiemale kasutajabaasile ning lahendus saab osaks mobiilirakendusest. Eesseisvate tööde hulka kuuluvad seega järgmised ülesanded:

- funktsionaalne E2E testimine;
- kasutajatestimine (beetakasutajad);
- kasutajatestidest laekunud tagasiside analüüsimine;
- rakenduse parendamine vastavalt tagasisidele;
- rakenduse valideerimine vastavalt peatükis 3.2 loetletud kriteeriumitele;
- mobiilirakenduse uue versiooni levitamine tervele kasutajabaasile;
- rakenduse hooldusperiood.

Käesoleva arenduse tulemused on aluseks järgnevatele edasiarendustele. Super mobiilirakenduse eSIM liitumisteekonna loomine on projekti esimeseks etapiks. Projekti järgnevates etappides on klientettevõttel kavas skoopi laiendada, pakkudes Superi kasutajale järgnevaid võimalusi:

- Vahetada füüsiline SIM-kaart eSIM vastu
- Vahetada eSIM füüsilise SIM-kaardi vastu
- Vahetada vana eSIM uue eSIM vastu

Edasiarenduste eesmärgiks on pakkuda Superi kasutajatele erinevaid võimalusi, kuidas teenust kasutada. Kuna eSIM ei hõiva seadmes ühtegi füüsilist SIM-kaardi pesa, võimaldab see funktsionaalsus laiendada Super kõnekaardi kasutajabaasi.

6 Kokkuvõte

Käesoleva bakalaureusetöö põhieesmärgiks oli lisada Super kõnekaardi mobiilirakendusele funktsionaalsus, mis võimaldaks kasutajal enda seadmesse paigaldada eSIM.

Töö analüüsi osas kaardistati rakenduse mooduli funktsionaalsed ja mittefunktsionaalsed nõuded, mille alusel planeeriti vastavad arendusülesanded. Lisaks andis analüüsi osa hea ülevaate kasutatavatest tehnoloogiatest ning tarkvaraarenduse mustritest, mida arenduse käigus järgiti.

Töös püstitatud eesmärgi saavutamiseks loodi olemasolevale Super mobiilirakendusele juurde moodul, mis suhtleb klientettevõtte loodud teiste rakenduste ja tugiteenustega. Moodul on loodud kasutades mitmekihilist klient-server arhitektuuri, kus on eraldatud esitlus-, äriloogika- ja andmekiht. Rakenduse äriloogikakiht realiseeriti Java programmeerimiskeeles, kasutades Spring Boot raamistikku. Rakenduse kasutajaliides arendati React Native raamistikuga. Kasutajal on mobiilirakenduse kaudu võimalik liituda Super kõnekaardi teenusega ja seadmesse paigaldada Super eSIM kasutades selleks loodud liitumisteekonda.

Rakenduse programmikoodi kirjutamisel kasutati testidel põhineva arenduse metoodikat. Testikava kirjutamine enne programmikoodi kirjutamist aitas koodi disainida võimalikult kergesti arusaadavaks, hallatavaks, silutavaks ja testitavaks. Lisaks kaeti arendatud mooduli äriloogikakiht ulatuslikult testidega, mis aitavad järgnevate arenduste käigus võimalikke tekkivaid vigu varakult avastada. Töö tulemusena leiti, et antud metoodikat on sobilik kasutada ka tulevaste arenduste käigus.

Lõpptulemusena loodud rakenduse moodul rahuldab kõiki ettenähtud funktsionaalseid ja mittefunktsionaalseid nõudeid. Loodud moodul on tarnitav lahendus ning on integreeritud rakenduse CI süsteemis. Moodul vastas algselt püstitatud nõuetele ja peamine eesmärk sai saavutatud. Lahendus võimaldab tulevikus kliendil liituda Super kõnekaardi teenusega ilma füüsilist SIM-kaarti soetamata.

Kasutatud kirjandus

[1] TestDriven Labs, „What is Test-Driven Development?,“ [Võrgumaterjal]. Available: https://testdriven.io/test-driven-development/. [Kasutatud 4 mai 2023].

[2] K.-T. Peterson, „M2M embedded subscriber identity module provisioning in networks without SMS service,“ TalTech, Tallinn, 2020.

[3] A. Mustafa, „What is the History of SIM Cards?,“ Hybrid Sim, 12 juuni 2022. [Võrgumaterjal]. Available: https://hybridsim.com/history-of-sim-cards/. [Kasutatud 11 aprill 2022].

[4] M. Kroodo, „Differences between SIM types - which SIM to choose?,“ 5 detsember 2019. [Võrgumaterjal]. Available: https://1ot.mobi/resources/blog/differences-between-sim-types-which-sim-to-choose. [Kasutatud 4 mai 2022].

[5] GSM Association, „eSIM Whitepaper: The what and how of Remote SIM Provisioning,“ Märts 2018. [Võrgumaterjal]. Available: https://www.gsma.com/esim/wp-content/uploads/2018/12/esim-whitepaper.pdf. [Kasutatud 4 Mai 2022].

[6] Trusted Connectivity Alliance, „eUICC Profile Package: Interoperable Format Technical Specification,“ Mai 2022. [Võrgumaterjal]. Available: https://trustedconnectivityalliance.org/technology_library/euicc-profile-package-technical-specifications/. [Kasutatud 11 aprill 2022].

[7] National Institute of Standards and Technology, „Recommendation for Key Management,“ mai 2020. [Võrgumaterjal]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf. [Kasutatud 11 aprill 2022].

[8] GSM Association, „Embedded UICC for Consumer Devices,“ 5 juuni 2018. [Võrgumaterjal]. Available: https://www.commoncriteriaportal.org/files/ppfiles/pp0100b_pdf.pdf. [Kasutatud 20 märts 2022].

[9] GSM Association, „Remote Provisioning Architecture for Embedded UICC Technical Specification,“ 27 mai 2016. [Võrgumaterjal]. Available: https://www.gsma.com/newsroom/wp-content/uploads/SGP.02_v3.1.pdf. [Kasutatud 3 aprill 2022].

[10] Trusted Connectivity Alliance, „Integrated SIM Functionality: Drivers, Approaches to Standardisation and Use Cases,“ mai 2021. [Võrgumaterjal]. Available: https://trustedconnectivityalliance.org/wp-content/uploads/2021/05/TCA-iTech-Intro-Paper_FINAL-FOR-WEB-UPLOAD.pdf. [Kasutatud 13 veebruar 2022].

[11] Deutsche Telekom AG, „nuSIM FAQ,“ 26 veebruar 2020. [Võrgumaterjal]. Available:

https://hardware.iot.telekom.com/LoadDocument/1384/nuSIM%20FAQ%20EN.pdf. [Kasutatud 23 märts 2022].

[12] K. M. Adams, Non-functional Requirements in Systems Analysis and Design, Springer, 2015.

[13] Oracle, „What is Java technology and why do I need it?," 2022. [Võrgumaterjal]. Available: https://www.java.com/en/download/help/whatis_java.html. [Kasutatud 11 mai 2023].

[14] VMware, Inc, „Spring Boot," 2023. [Võrgumaterjal]. Available: https://spring.io/projects/spring-boot. [Kasutatud 4 mai 2023].

[15] Gradle Inc, „What is Gradle?," 2022. [Võrgumaterjal]. Available: https://docs.gradle.org/current/userguide/what_is_gradle.html. [Kasutatud 11 aprill 2023].

[16] Project Lombok, „Project Lombok," 2022. [Võrgumaterjal]. Available: https://projectlombok.org. [Kasutatud 11 aprill 2023].

[17] WireMock, „WireMock - flexible API mocking," 2023. [Võrgumaterjal]. Available: https://wiremock.org. [Kasutatud 01 mai 2023].

[18] Oracle, „JSON Defined," 2017. [Võrgumaterjal]. Available: https://www.oracle.com/database/what-is-json/. [Kasutatud 21 aprill 2023].

[19] Mozilla Corporation, „What is JavaScript?," 5 märts 2023. [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. [Kasutatud 1 mai 2023].

[20] Microsoft, „What is TypeScript?," 2023. [Võrgumaterjal]. Available: https://www.typescriptlang.org. [Kasutatud 4 mai 2023].

[21] Meta, „Getting Started with React," [Võrgumaterjal]. Available: https://legacy.reactjs.org/docs/getting-started.html. [Kasutatud 4 mai 2023].

[22] Meta, „Introducing JSX," [Võrgumaterjal]. Available: https://legacy.reactjs.org/docs/introducing-jsx.html. [Kasutatud 4 mai 2023].

[23] Meta, „React Native: Learn once, write anywhere," [Võrgumaterjal]. Available: https://reactnative.dev. [Kasutatud 11 aprill 2023].

[24] T. Linsley, „Powerful asynchronous state management," [Võrgumaterjal]. Available: https://tanstack.com/query/v3/. [Kasutatud 19 aprill 2023].

[25] Microsoft, „Patterns in Practice - Convention Over Configuration," veebruar 2009. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-in-practice-convention-over-configuration. [Kasutatud 4 mai 2023].

[26] Meta, „Introducing Hooks," [Võrgumaterjal]. Available: https://legacy.reactjs.org/docs/hooks-intro.html. [Kasutatud 19 aprill 2023].

[27] Odemolliens, „react-native-sim-cards-manager," [Võrgumaterjal]. Available: https://github.com/odemolliens/react-native-sim-cards-manager. [Kasutatud 13 veebruar 2023].

[28] Google, „EuiccManager," [Võrgumaterjal]. Available: https://developer.android.com/reference/android/telephony/euicc/EuiccManager. [Kasutatud 29 aprill 2023].

[29] Apple Inc, „Core Telephony,“ [Võrgumaterjal]. Available: https://developer.apple.com/documentation/coretelephony. [Kasutatud 1 mai 2023].

[30] M. Rouse, „Client/Server Architecture,“ 25 august 2020. [Võrgumaterjal]. Available: https://www.techopedia.com/definition/438/clientserver-architecture. [Kasutatud 21 veebruar 2023].

[31] M. Richards, Software Architecture Patterns, Sebastopol: O’Reilly Media, Inc, 2015.

[32] Microsoft, „Three-Layered Services Application,“ 17 märts 2014. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ff648105(v=pandp.10)?redirectedfrom=MSDN. [Kasutatud 11 veebruar 2023].

[33] Codecademy, „What is REST?,“ [Võrgumaterjal]. Available: https://www.codecademy.com/article/what-is-rest. [Kasutatud 23 märts 2023].

[34] R. T. Fielding, „Representational State Transfer (REST),“ 2000. [Võrgumaterjal]. Available: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. [Kasutatud 23 veebruar 2023].

[35] TutorialsTeacher, „Inversion of Control,“ [Võrgumaterjal]. Available: https://www.tutorialsteacher.com/ioc/inversion-of-control. [Kasutatud 11 jaanuar 2023].

[36] VMware, Inc, „The IoC Container,“ 18 aprill 2023. [Võrgumaterjal]. Available: https://docs.spring.io/spring-framework/docs/current/reference/html/core.html#beans-introduction. [Kasutatud 4 mai 2023].

[37] R. C. Martin, „Design Principles and Design Patterns,“ Robert C. Martin, 2000.

[38] M. Fowler, „Avoiding Repetition,“ jaanuar 2001. [Võrgumaterjal]. Available: https://martinfowler.com/ieeeSoftware/repetition.pdf. [Kasutatud 4 mai 2023].

[39] M. Fowler, „martinfowler.com,“ 26 mai 2015. [Võrgumaterjal]. Available: https://martinfowler.com/bliki/Yagni.html. [Kasutatud 11 aprill 2023].

[40] M. Fowler, „Tell-Don't-Ask,“ 5 september 2013. [Võrgumaterjal]. Available: https://martinfowler.com/bliki/TellDontAsk.html. [Kasutatud 11 aprill 2023].

[41] Atlassian, „Jira Software for teams,“ [Võrgumaterjal]. Available: https://www.atlassian.com/software/jira/guides/getting-started/who-uses-jira#for-agile-teams. [Kasutatud 9 veebruar 2023].

[42] D. Wagner, „Using RestTemplate in Spring,“ 30 mai 2019. [Võrgumaterjal]. Available: https://springframework.guru/using-resttemplate-in-spring/. [Kasutatud 4 mai 2023].

[43] GSM Association, „SGP.02 Remote Provisioning Architecture for Embedded UICC Technical Specification,“ 5 juuni 2020. [Võrgumaterjal]. Available: https://www.gsma.com/esim/wp-content/uploads/2020/06/SGP.02-v4.1.pdf. [Kasutatud 11 aprill 2022].

[44] GSM Association, „Brief History of GSM & the GSMA,“ 2017. [Võrgumaterjal]. Available: https://www.gsma.com/aboutus/history. [Kasutatud 11 detsember 2021].

[45] Qualcomm Technologies, „Vodafone, Qualcomm Technologies, and Thales Deliver World-First Smartphone Demonstration of Integrated SIM (iSIM) Technology,“ 17 jaanuar 2022. [Võrgumaterjal]. Available: https://www.qualcomm.com/news/releases/2022/01/vodafone-qualcomm-technologies-and-thales-deliver-world-first-smartphone. [Kasutatud 11 aprill 2022].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Otto Kaarel Altroff

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „eSIM toe arendamine Super kõnekaardi mobiilirakendusele“, mille juhendaja on German Mumma ja kaasjuhendaja Meelis Luiks
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

15.05.2023

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 –RestTemplate klass

```
@Component
public class EeSimApiUhenduja {

    @Autowired
    @Qualifier("EeSimApiUhenduja")
    private RestTemplate restTemplate;

    public ESimVastus laeAllaESim(ESimParing body) {
        return reMapHttpStatusCodeException(() -> {
            HttpEntity<Object> entity = new HttpEntity<>(body);
             ResponseEntity<ESimResponse> entityResponse =
             restTemplate.exchange( "/v1/esim/laeAlla",
                    HttpMethod.POST, entity, ESimVastus.class);
            return entityResponse.getBody();
        }, "lae alla");
    }

    private <T> T reMapHttpStatusCodeException(Supplier<T> callable, String
    message){
        try{
            return callable.get();
        } catch( HttpStatusCodeException sce ){
             throw new GeneralException( "Exception /v1/esim/" + message +
             sce, "500" );
        }
    }
}
```

Lisa 3 – Andmesaateobjekt ja selle validatsioon

```
@Data
@Builder
@AllArgsConstructor
public class ESimNumber {

    private String iccidKood;
    private String mobNr;

    public boolean andmedEksisteerivad() {
        return StringUtils.hasLength(mobNr) &&
             StringUtils.hasLength(iccidKood);
    }

    public void valideeri() {
       if (!andmedEksisteerivad()) {
        throw new IllegalArgumentException("illegaalne eSimNumber");
       }
    }
}
```

Lisa 4 – Teenuse klass

```
@Service
public class ESimTeenus {

    @Autowired
    EeSimApiUhenduja eeSimApiUhenduja;

    public ESimTeave pariTeavet(String iccidKood) {
       ESimTeave teave =
       eeSimApiRestTemplate.pariEsimTeavet(iccidKood);
        if (teave == null) {
            throw new IllegalArgumentException("ESim error:
             eSIM not found with ICCID " + iccidKood);
        }
        return info;
    }
}
```

Lisa 5 – React-Navigation/stack teegi kasutamine

```
const ESimStack = createStackNavigator();

const ESimNav = () => {
    const screenOptions = {
        ...tabsViewHeaderStyle,
        headerBackTitle: I18n.t('navbar.back'),
    };

    return (
        <ESimStack.Navigator initialRouteName={'eSimLanding'}>
            {Router.eSimRoutes.map(({ name, component, options },
index) => (
                <ESimStack.Screen
                    name={name}
                    component={component}
                    options={{ ...screenOptions, ...options }}
                    key={index}
                />
            ))}
        </ESimStack.Navigator>
    );
};
```

Lisa 6 – React-Query ja React Hooks kasutamine

```
export const useBookEsim = (quantity: number) => {
    const [selectedNumber, setSelectedNumber] = useState<ESimNumber |
undefined>();

    const { data, refetch, isError} = useAppQuery(['eSimNumbers'], {
        queryFn: () => api.bookESim(quantity) as Promise<ESimNumber[]>,
        enabled: false,
    });

    return {
        data,
        selectedNumber,
        setSelectedNumber,
        refetch,
        isError,
    };
};
```

Lisa 7 – Teegi react-native-sim-cards-manager kasutamine

```
export default function useSupportEsim() {
    const [supportError, setSupportError] = useState<string |
    undefined>();

    const isEsimSupported = async () => {
        try {
            return await SimCardsManagerModule.isEsimSupported();
        } catch (error: any) {
            if (error.message) {
                setSupportError(error.message);
            }
            return false;
        }
    };

    return {
        isEsimSupported,
        supportError,
    };
}
```