TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Infosüsteemide õppetool

Universaalse krüptovaluuta rahakotiteenuse arendamine Bitplexus näitel

Magistritöö

Üliõpilane: Priidu Neemre

Üliõpilaskood: 132276IABMM

Juhendaja(d): dotsent Enn Õunapuu
lektor Raul Liivrand

Tallinn
2016

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.

…… ……

(kuupäev) *(allkiri)*

Universaalse krüptovaluuta rahakotiteenuse arendamine Bitplexus näitel

Annotatsioon

Käesoleva töö eesmärgiks on luua turvaline veebipõhine keskkond, mis võimaldab kasutajal hallata oma krüptorahasid ilma sellega kaasnevate kohustuste (plokiahelate allalaadimine) ning vastutusteta (salajaste võtmete krüpteerimine ning varundamine). Enim tähelepanu pööratakse seejuures tervikliku multivaluuta toe väljaehitamisele, et süsteemi oleks võimalikult lihtne täiendada uute plokiahelapõhiste krüptovaluutadega.

Eelkirjeldatud eesmärkide saavutamiseks viiakse esmalt läbi vaadeldava infosüsteemi (Bitplexus) lihtsustatud süsteemianalüüs ning pannakse paika selle esialgne süsteemiarhitektuur. Seejärel realiseeritakse rakenduse aluseks olev andmebaas ning analüüsitakse mõningaid krüptoraha teenuste tüüpilisi puuduseid ja probleeme.

Töö põhitulemuseks on universaalne krüptovaluuta rahakotiteenus, mida on võimalik kasutada kliendi Bitcoini ja Litecoini varade haldamiseks. Lisaks sellele pakutakse välja ka üldistatud domeenimudel plokiahelapõhiste krüptovaluutadega liidestumiseks. Bitcoini ja Litecoini võrgustikega suhtlemiseks arendatakse valmis Java konnektorteegid nimega *btcd-cli4j* ja *ltcd-cli4j*, mida on kasutatud ka juba teiste autorite krüptoraha teemalistes projektides.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 80-l leheküljel, 3 peatükki, 16 joonist ja 3 tabelit.

Development of Universal Cryptocurrency Wallet Services: A Case Study of Bitplexus

Abstract

The purpose of this thesis is to develop a secure web-based wallet application allowing cryptocurrency users to manage their funds without any additional obligations (*e.g.* downloading the entire block chain) and responsibilities (*e.g.* encrypting & backing up their private keys). Considerable effort is put into designing the application's multicurrency support to allow for the seamless integration of new block chain-based cryptocurrencies.

To achieve these goals, a simplified system analysis is conducted and a preliminary system architecture established for the aforementioned application (Bitplexus). Next, the system's data layer is implemented and several design decisions explained to identify the most common mistakes in modern cryptocurrency-related services.

The main outcome of this thesis is a universal cryptocurrency wallet service that can be used to manage users' Bitcoin and Litecoin funds. Additionally, a generic domain model for interfacing with other block chain-based cryptocurrencies is proposed. In order to communicate with the Bitcoin and Litecoin networks, two Java wrapper libraries named *btcd-cli4j* and *ltcd-cli4j* are also developed.

The thesis is written in Estonian and contains 80 pages of text, 3 chapters, 16 figures and 3 tables.

Jooniste loetelu

Sisukord

Sissejuhatus

Elektroonilise raha idee sai alguse 1960. aastatel, mil ilmusid tänapäevase interneti esimesed eelkäijad (ARPANET, NSFNET jt). Selle aja jooksul on elektronraha kontseptsioon palju muutunud, jõudes paarikümne aastaga primitiivsetest sularahaautomaatidest kuni täisdigitaalsete e-raha süsteemideni (nt DigiCash, PayPal). Kui sajandivahetuse paiku domineerisid e-raha valdkonda veel finantsettevõtete poolt pakutavad „eravaluutad“ (nt PayPal Cash), siis juba mõni aasta hiljem tegeldi aktiivselt ka erinevate detsentraliseeritud e-raha skeemide uurimisega. Läbimurded partnervõrgu (ingl k *peer-to-peer*) tehnoloogiates (Napster, BitTorrent, Skype jt) ning 2008. aastal lahvatanud majanduskriis süvendasid seda trendi veelgi, pannes lõppkokkuvõttes aluse nn krüptovaluutade (ingl k *cryptocurrencies*) liikumisele.

Krüptovaluutade peamisteks tugevusteks võrreldes teiste analoogsete süsteemidega on nende läbipaistvus, anonüümsus, madalad teenustasud ning sõltumatus kolmandatest osapooltest. Ühtlasi tähendab see aga seda, et iga võrgustikus osaleja peab ise hoolt kandma oma varade turvalisuse eest (sellest ka levinud tunnuslause: *„Be your own bank.“*). Lisaks sellele tuleb igal rahakotitarkvara käitaval isikul alla laadida *kõik* antud valuutas sooritatud tehinguid, mille kogumaht võib populaarsete krüptovaluutade puhul (nt Bitcoin) ulatuda kümnetesse gigabaitidesse. Kõik see teeb krüptovaluutadest äärmiselt kasutajavaenuliku tehnoloogia, mistõttu on viimastel aastatel aina rohkem tähelepanu pöörama hakatud just mugavate rahakotilahenduste väljatöötamisele. Kaheks suurimaks innovatsiooniks selles valdkonnas on nn kergvalideeruvad rahakotitarkvarad ning tsentraliseeritud rahakotiteenused (vt ptk 1.2). Mõlemad neist suurendavad oluliselt krüptovaluutade kasutusmugavust, kuid toovad endaga kaasa ka mitmeid lihtsustusi rakenduse turvalisuses, paindlikkuses ning läbipaistvuses.

Kuna valdav osa tsentraliseeritud rahakotiteenustest luuakse ärilistel eesmärkidel, on nende kohta kättesaadav informatsioon (s.o dokumentatsioon, lähtekood) reeglina üsna piiratud. Seega on käesoleva magistritöö eesmärgiks luua avatud lähtekoodiga (FOSS) rahakotiteenuse prototüüp, mida oleks võimalik kasutada ka teiste sarnaste süsteemide arendamiseks. Lisaks sellele pakutakse välja üldistatud domeenimudel plokiahelapõhiste krüptovaluutadega

liidestumiseks ning analüüsitakse mõningaid krüptovaluuta teenuste tüüpilisi puudusi ja probleeme. Viimaks arendatakse valmis Java konnektorteegid Bitcoini ja Litecoini võrgustikega suhtlemiseks, mida on võimalik hõlpsasti mugandada (ingl k *fork*) ka teistele sarnastele valuutadele.

Töö jaguneb kolmeks suuremaks osaks: esmalt analüüsitakse krüptovaluutade peamisi tööpõhimõtteid ning antakse ülevaade nende krüptograafilistest alustest. Seejärel vaadeldakse lähemalt mõningaid rahakotitarkvara liike ning tuuakse välja nende peamised eelised/puudused. Töö teises osas viiakse läbi loodava infosüsteemi (Bitplexus) strateegiline ja detailanalüüs, sh pannakse paika selle eesmärgid, põhiobjektid, ärireeglid ning funktsionaalsed ja mittefunktsionaalsed nõuded. Kolmandas osas keskendutakse eeskätt süsteemi tehnilise teostuse kirjeldamisele. Esmalt esitatakse vaadeldava rakenduse süsteemiarhitektuur ning kirjeldatakse selle tarkvaralisi ja tehnoloogilisi valikuid. Seejärel realiseeritakse teises osas analüüsitud andmebaas ning antakse ülevaade mõningatest tüüpilistest puudustest ja probleemidest, mida võib kohata krüptovaluuta rahakotitarkvara arendamisel. Kuna tegu on krüptoraha valdkonda laiemalt puudutavate probleemidega, on neile toodud lahendusi võimalik kohandada ka mitmetele teistele sarnastele süsteemidele.

1. Krüptovaluutad

Krüptovaluuta on digitaalne käibevahend ja e-raha alamliik, mis kasutab rahasüsteemi reeglite jõustamiseks erinevaid krüptograafilisi skeeme ning algoritme. Krüptovaluutade ühiseks tunnuseks on nende detsentraliseeritud iseloom, mis tähendab, et rahasüsteemi ei hoia üleval mitte üks keskne juhtorgan (ingl k *central authority*), vaid krüptovaluuta kasutajad ise (Antonopoulos 2014, 1). Kuna kõigil võrgustikus osalejatel on süsteemi suhtes võrdsed õigused, on see tervikuna hästi kaitstud tsensuuri ja muude negatiivsete välismõjude eest. Uute valuutaühikute väljastamine toimub seejuures range ajaplaani alusel, kusjuures paika on pandud ka nende lõplik arv süsteemis. Selliselt on tagatud rahasüsteemi maksimaalne stabiilsus ning hoitud ära mitmeid traditsioonilistest valuutadest tuntud negatiivseid ilminguid nagu hüperinflatsioon ja väliskapitali järsk väljavool.

Krüptovaluutade liikumine sai alguse 2009. aastal, mil avalikustati Bitcoini-nimelist e-raha skeemi tutvustav teadusartikkel ning selle võrgusõlme tarkvara (ingl k *node*) esimene prototüüp. Tänaseks on Bitcoinist välja kasvanud sadu erinevate omaduste ja eesmärkidega krüptovaluutasid, mis kõik jagavad aga paljuski samu tööpõhimõtteid. Märkimisväärseteks näideteks on tehingute kiiremat kinnitamist (Bitcoini 10 minuti asemel 2.5 minutit) võimaldav Litecoin, uusi konsensusmehhanisme (s.o Bitcoini *proof-of-work* asemel nt *proof-of-stake*) katsetavad Peercoin ja Nxt ning kasutajate anonüümsusele keskenduvad Zerocoin ja Darkcoin (e Dash). Huvitavaid võimalusi pakuvad ka Bitcoini võrgustikule ehitatud metaprotokollid nagu Colored Coins ning Counterparty, mis täiendavad baasvaluutat keerukama funktsionaalsusega nagu kasutaja-defineeritud valuutad ning targad lepingud (ingl k *smart contracts*).

Tänu krüptovaluutade laialdasele kasutusele süvaveebi kaubamajades nagu Silk Road ning Atlantis seostatakse neid sageli ka ühiskonna pahupoolega ning organiseeritud kuritegevusega (Sauga 2015, 54). Tegelikkuses ei ole aga detsentraliseeritud e-raha idees mitte midagi kriminaalset. Krüptovaluuta tehingute pseudo-anonüümne iseloom teeb neist lihtsalt äärmiselt mugava vahendi mitmete ebaseaduslike tegevuste rahastamiseks ning läbiviimiseks (sh narkokaubandus, rahapesu, erinevad kelmused jmt) (Perez 2015).

1.1 Ülevaade peamistest tööpõhimõtetest

Käesolevas alajaotises antakse lühikene ülevaade klassikaliste krüptovaluutade peamistest tööpõhimõtetest ning nende krüptograafilistest alustest. Esmalt vaadeldakse lähemalt mõningaid krüptoraha süsteemide keskseid artefakte (rahakotid, võtmed ja aadressid), mis on vajalikud vahendite kuuluvuse tuvastamiseks ning võrgustikus osalejate eristamiseks. Seejärel tutvustatakse krüptovaluuta tehingute elutsüklit ning antakse ülevaade nende haldamiseks kasutatavast *plokiahela* tehnoloogiast. Viimaks seletatakse lahti krüptovaluutade *kaevandamise* fenomen ehk kuidas saavutada laiapõhjaline konsensus tuhandetest võrgusõlmedest koosnevas hajussüsteemis.

1.1.1 Rahakotid, võtmed ja aadressid

Krüptovaluuta rahakott (ingl k *wallet*) on fail, mis koosneb hulgast krüptograafilistest võtmepaaridest, millest igaüks esindab mingit kindlat krüptovaluuta aadressi (Antonopoulos 2014, 84). Võtmepaari salajast poolt kasutatakse seejuures uute tehingute allkirjastamiseks ning vahendite kuuluvuse tõendamiseks. Võtmepaari avalikku poolt kasutatakse aga krüptovaluuta tehingu võimalike otspunktide tähistamiseks. Seetõttu võib võtmepaari avalikku poolt võrrelda ka traditsioonilise pangakonto numbriga (e IBAN-koodiga) ning võtmepaari salajast poolt sellele vastava parooliga.

Kuna kõige levinum viis krüptovaluuta avalike võtmete esitamiseks on 130-kohaliste kuueteistkümnendkoodidena (520 bitti informatsiooni), ei sobi nad eriti hästi inimeste kontode igapäevaseks identifitseerimiseks. Seetõttu on pea kõigis krüptovaluutades kasutusele võetud nn *aadressi* abstraktsioon. Krüptovaluuta aadressi näol on tegu 25–34-kohalise tähtnumbrilise koodiga (Bitcoini puhul), mis identifitseerib üheselt mingi kindla avaliku võtme. Aadresside peamiseks eeliseks on, et nad on avalikest võtmetest tunduvalt lühemad. See võimaldab neid kergemini meelde jätta, üles kirjutada ning erinevatele kujudele teisendada (nt makse-URLiks, QR-koodiks jne).

Tehnilisest küljest põhinevad krüptovaluutades kasutatavad võtmepaarid reeglina elliptkrüptograafilisel *ECDSA* (ingl k *Elliptic Curve Digital Signature Algorithm*) algoritmil ning selle standardsel secp256k1 kõveral (Gorale 2014). Selleks, et tuletada ECDSA avalikust võtmest krüptovaluuta aadressi, rakendatakse sellele järjestikuliselt *SHA-256* ning *RIPEMD-*

160 räsifunktsioone. Seejärel lisatakse räsimise tulemusena saadud sõnele juurde täiendav kontrollsumma (ingl k *checksum*) ning viiakse see üle Base58 kodeeringusse. Base58 on tähtnumbriline kodeering, millest on eemaldatud mõned kergesti segiaetavad tähed ja numbrid nagu „0“ (null), „O“ (suur o), „I“ (suur i) ning „l“ (väike L). Eelkirjeldatud lähenemise eesmärgiks on suurendada krüptovaluuta aadresside veakindlust ning vähendada nende pikkust. Protsessi lõpptulemuseks ongi krüptovaluuta aadress ehk sõne kujul `1FHrg3X79X35bWLzxZdHhjS7E7rTJ1qrN2`.

1.1.2 Tehingud

Krüptovaluuta tehing (ingl k *transaction*) on võti-väärtus paaridest koosnev andmeobjekt, mis tähistab väärtuse liikumist krüptovaluuta võrgustikus (Khaosan 2014). Ülekande sooritamiseks tuleb makse saatjal esmalt valmis genereerida nn tehingu põhi. Viimases pannakse paika kõik tehingu täpsed tingimused, sh selle lähte- ja sihtaadressid, saadetav summa ning vajadusel ka vahendite vabastamise aeg. Seejärel allkirjastatakse tehing ühe või enama salajase võtmega, et tõendada lähteaadresside omaniku huvi osaleda antud tehingus kohustatud poolena. Tehingud, mis ei ole varustatud vajalike signatuuridega, loetakse võrgusõlme tarkvara poolt kehtetuks ning eemaldatakse selle tehingupuhvrist (ingl k *mempool*). Selliselt on tagatud, et aadressil paiknevad vahendid oleksid kättesaadavad vaid isikule, kes valdab korrektset salajast võtit.

Pärast tehingu allkirjastamist ning verifitseerimist saadetakse see laiali kõigile võrgusõlme naabritele, kes kontrollivad samuti eelkirjeldatud signatuuride kehtivust. Sama protsess jätkub rekursiivselt kuni tehing on jõudnud pea kõigi kättesaadavate võrgusõlmedeni (Antonopoulos 2014, 109). Teatud aja tagant (Bitcoini puhul ~ 10 minutit) valib üks võrgusõlmedest (s.o kaevur, vt ptk 1.1.4) oma tehingupuhvrist mõned kinnitamata tehingud ning lisab need vastava valuuta plokiahelasse. Erinevalt kinnitamata tehingutest (ingl k *unconfirmed transactions*) on plokiahelas fikseeritud tehinguid praktiliselt võimatu muuta/tagasi võtta. Seetõttu tasub krüptovaluutadega arveldamisel alati oodata, kuni tehing on saanud võrgustikult vähemalt ühe (soovitatavalt kuni kuus) kinnitust (ingl k *confirmation*).

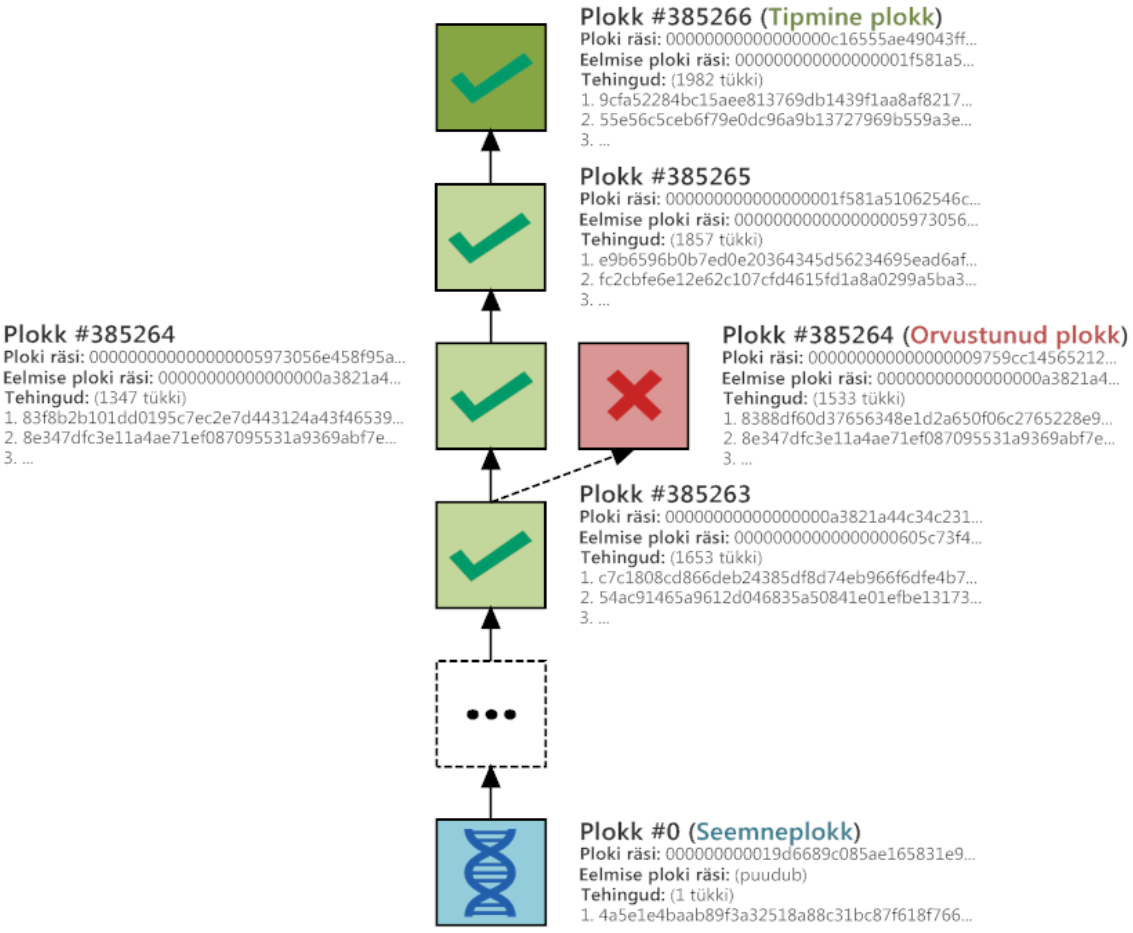
Oluline on silmas pidada, et krüptovaluutade matemaatiline mudel ning partnervõrgu kontseptsioon ei ole omavahel rangelt seotud. Seetõttu on krüptovaluuta tehinguid võimalik genereerida ka seadmetel, millel puudub igasugune ühendus välismaailmaga. Viimase

tüüpilisteks kasutuskohtadeks on nt füüsilised rahakotiseadmed (Trezor, Ledger Nano, KeepKey jt) ning välisvõrgust isoleeritud (ingl k *air-gapped*) rahakotiserverid, mille ainsaks eesmärgiks ongi tehingute allkirjastamine. Taolistes keskkondades koostatud tehingud viiakse tüüpiliselt üle mõnda välisvõrguga ühendatud arvutisse (nt mälupulga abil), kust nad saadetakse laiali vastava valuuta võrgustikule (Siri 2014). Eelkirjeldatud lähenemine võimaldab makseid sooritada viisil, kus kasutaja salajased võtmed ei puutu ealeski kokku ühegi arvutivõrku ühendatud seadmega.

1.1.3 Plokid, plokiahelad ja võrgustikud

Krüptovaluuta plokk (ingl k *block*) on andmeobjekt, mida kasutatakse võrgustikule edastatud tehingute agregeerimiseks ning nende hajusasse tehinguandmete žurnaali talletamiseks (Antonopoulos 2014, 160). Iga plokk koosneb hulgast kinnitamata tehingutest ning selle sisu kirjeldavatest metaandmetest (plokiprotokolli versioon, eelmise ploki SHA-256 räsi, ploki loomise kuupäev ja kellaaeg jmt). Oma olemuselt meenutavad krüptovaluuta plokid mõneti pankadevahelisi kliiringuid, kus teatud ajaperioodi jooksul aset leidnud tehingud klaaritakse (ingl k *settle*) ühe pakktöötlusliku kandena. Kuna iga plokk sisaldab viidet eelmise ploki unikaalsele identifikaatorile, nimetatakse kõigi plokkide järjestamisel moodustuvat andmestruktuuri ka krüptovaluuta plokiahelaks (ingl k *block chain*) (Piasecki 2012, 16).

Krüptovaluuta plokiahel on seega järk-järgult kasvav hajus andmebaas, mis sisaldab kõigi antud valuutas sooritatud tehingute andmeid ning mida tiražeeritakse (ingl k *replication*) iga võrgusõlme tarkvara käitava serverarvuti kõvakettale. Plokiahelat kujutatakse sageli vertikaalse pinuna, mille iga järgnev kiht (plokk) toetub selle eelmisele kihile (plokile) kuni võrgusõlme tarkvarasse sissekodeeritud seemneplokini (ingl k *genesis block*) välja (Antonopoulos 2014, 159). Iga pinusse lisatav plokk sisaldab seejuures kümnete, sadade või isegi tuhandete uute krüptovaluuta tehingute andmeid (vt joonis 1).



Joonis 1. Lihtsustatud krüptovaluuta plokiahel

Kuna iga plokk kannab endas viidet eelmise ploki unikaalsele identifikaatorile (e selle SHA-256 räsile), ei ole võimalik muuta ühe ploki andmeid ilma selle järglasi kehtetuks muutmata. Põhjuseks on, et ploki andmete muutmisel (nt stsenaarium „suunan kõik laekumised enda aadressile“) muutub ka selle unikaalne identifikaator ehk SHA-256 räsi. Järelikult tuleb vastavat identifikaatorit uuendada ka kõnealuse ploki järglasplokis, et vältida viimase kehtetuks tunnistamist võrgusõlme tarkvara poolt. Eelnev toob aga kaasa järglasploki enda identifikaatori muutumise, mistõttu tuleb vastavat viidet uuendada ka järglasploki järglasplokis jne. Teisisõnu tuleb ühe plokiahelas talletatud tehingu võltsimiseks ümber kirjutada sisuliselt kogu sellele järgnev tehingute ajalugu. Kuna uute plokkide genereerimine (e kaevandamine, vt ptk 1.1.4) on aga äärmiselt töömahukas protsess, ei ole taoliste rünnete läbiviimine tänapäeval enam praktiliselt võimalik. Potentsiaalne ründaja peaks selleks kontrollima vähemalt ~ 30–40% kogu võrgustiku arvutusvõimsusest, mis on suuremate krüptovaluutade turukapitalisatsioone arvestades (nt Bitcoin – 6.6 mld USA dollarit) jõukohane vaid üksikutele riikidele ning suurkorporatsioonidele.

1.1.4 Kaevandamine

Kaevandamine (ingl k *mining*) on protsess, mille abil krüptovaluuta partnervõrk juhib võrgustikule edastatud tehingute talletamist ning kontrollib selles käibiva rahavaru (ingl k *money supply*) suurust (Eyal jt 2014, 439–440). Kaevandamise põhiproduktiks on plokid (vt ptk 1.1.3), mis kujutavad endast võrgustiku konsensuslikku otsust lugeda teatud tehingud „lõpetatuks“ ning kanda nende andmed vastava valuuta püsivasse tehinguandmete žurnaali. Lisaks tehingute kinnitamisele täidavad kaevurid ka raha emiteerija rolli, saades iga kaevandatud ploki ees preemiaks teatud summa värskelt „vermitud“ raha (ingl k *block reward*). Taoline levitusskeem aitab raha ringlusesse lasta võimalikult ausal ning detsentraliseeritud viisil, kuna iga kaevuri tegevust kompenseeritakse vastavalt tema panusele (aeg, elekter, riistvara) süsteemi töös. Lisaks eelnevale loob see majandusliku stiimuli (ingl k *incentive*) krüptovaluuta kaevandamisprotsessis osalemiseks, mis toob omakorda kaasa võrgustiku arvutusvõimsuse (ingl k *hash rate*) eksponentsiaalse kasvu. Viimane suurendab aga oluliselt kogu valuuta turvalisust, kuna plokiahela ümberkirjutamiseks peab potentsiaalne ründaja kontrollima vähemalt poolt (> 50%) võrgustiku summaarsest arvutusvõimsusest (Cawrey 2014).

Tehnilisest küljest seisneb krüptovaluutade kaevandamine mingi kindla räsifunktsiooni (Bitcoini puhul nt SHA-256, Litecoini puhul aga *scrypt*) rakendamises võrgustiku poolt hetkel otsitava ploki metaandmetele. Kaevandamisvooru võitjaks kuulutatakse seejuures kaevur, kes suudab esimesena valmis genereerida räsi, mis on väiksem antud vooru sihträsist (ingl k *target*) (Shirriff 2014). Teisisõnu otsitavad kaevurid üksteise võidu räsi, mis algaks võimalikult suure arvu nullidega. Näiteks kui vooru sihträsiks on `0x00000fffffffffff`, siis kaevuri poolt genereeritud räsi `0x0005a95933d115b6` ei vasta antud vooru võidutingimustele. Küll aga rahuldab eeltoodud tingimusi räsi `0x000000b99b0edf75`, mille leidja kroonitakse automaatselt vastava ploki loojaks ning selle plokipreemia laureaadiks (Bitcoini puhul hetkel 25 BTC/plokk, Litecoini puhul aga 50 LTC/plokk).

Oluline on silmas pidada, et krüptovaluuta võrgustiku arvutusvõimsus on ajas äärmiselt kiiresti muutuv suurus. 2012. aasta maikuu seisuga oli Bitcoini võrgustiku summaarseks arvutusvõimsuseks ligikaudu 12 TH/s (ingl k *terahash per second*) (Pallas 2012, 12). Käesoleva töö kirjutamise hetkeks (2015. aasta november) on see arv tõusnud aga 550 000 TH/s piirimaile, mis tähendab, et võiduka lahenduse leidmiseks proovitakse ühes sekundis

läbi u 550 000 000 000 000 000 erinevat räsikombinatsiooni. Seepärast kasutatakse eelmainitud sihträside arvutamiseks reeglina mõnda isereguleeruvat algoritmi (s.o algoritmi, mis reageerib muuhulgas ka muutustele võrgustiku arvutusvõimsuses). Eelnev kaitseb valuutat võimaliku hüperinflatsiooni eest ning aitab tagada selle plokkidevahelise ajaintervalli (ingl k *block time*) püsimise ettenähtud piirides.

1.2 Rahakotitarkvara liigid

Täisvalideeruv rahakotitarkvara (ingl k *full node wallet*) on rahakotitarkvara, mis peab lisaks kliendi enda aadressidele ning tehingutele arvet ka kõigi teiste antud valuutas sooritatud tehingute üle. Põhimõtteliselt on tegu hariliku võrgusõlme tarkvaraga, millele on lisatud mugav kasutajaliides ning hulk muud abifunktsionaalsust, mis lihtsustab kasutaja varade haldamist. Eelkirjeldatud rahakotiliigi suurimaks eeliseks on selle turvalisus ning sõltumatus teistest osapooltest, kuna iga võrgustikule edastatud tehing on verifitseeritav ilma ühegi teise võrgusõlme poole pöördumata. Selle suurimaks puuduseks on aga äärmiselt kõrge ressursikasutus (nt Bitcoini puhul: vähemalt ~ 50 GB kõvaketta ruumi, ~ 1 GB muutmälu jne), mis on tingitud süsteemi suuremahulise manusandmebaasi haldamisest. Antud liiki rahakotitarkvara on tüüpiliselt realiseeritud töölauarakendusena, kusjuures siia alla käivad ka Bitcoini ja Litecoini rahakotitarkvarade näidisimplementatsioonid (s.o Bitcoin Core ja Litecoin Core).

Kergvalideeruv rahakotitarkvara (ingl k *lightweight wallet* ehk *SPV*) erineb täisvalideeruvast rahakotitarkvarast selle poolest, et plokkide asemel peetakse rakenduses arvet hoopis nende metaandmete üle (Nakamoto 2008, 5). Teisisõnu ei verifitseeri süsteem kõiki võrgustikule edastatud tehinguid, vaid veendub üksnes neid sisaldavate plokkide korrektsuses ning autentsuses. Seejuures on võrgusõlmel võimalik kontrollida ka ploki metaandmetes hoitavat räsipuud (ingl k *merkle tree*), et veenduda üksikute tehingute olemasolus/puudumises. Sellist tüüpi rahakotitarkvara sõltub suuresti teistest võrgusõlmedest ning on seetõttu avatud ka mitmetele täiendavatele ründevektoritele (nn vähksõlmed ja Sybil-rünnakud). Samas iseloomustab seda tunduvalt madalam ressursikasutus (nt Bitcoini puhul: ~ 50 MB kõvaketta ruumi, ~ 5 MB muutmälu), mis teeb temast ideaalse lahenduse piiratud arvutusvõimsusega keskkondades nagu nutitelefonid, tahvelarvutid jne.

Kolmandaks levinumaks rahakotitarkvara liigiks on nn tsentraliseeritud rahakotiteenused (ingl k *web wallet* või *wallet service*), mis kujutavad endast sisuliselt krüptovaluutade haldamiseks mõeldud internetipanku. Taoliste teenuste suurimaks eeliseks on see, et nad võtavad kasutajalt üle pea kõik rahakotitarkvara käitamisega kaasnevad kohustused (sh plokiahelate allalaadimine) ning vastutused (sh salajaste võtmete krüpteerimine ja varundamine). Eelnev parandab oluliselt krüptovaluutade kasutusmugavust ning muudab nad ligipääsetavaks ka mitte-tehnilistele inimestele. Samas toob see endaga kaasa mitmeid täiendavaid riske, kuna kasutajal puudub vahetu kontroll oma salajaste võtmete üle. Teisisõnu tuleb kasutajal arvestada võimalusega, et rakenduse haldaja võib tema rahadega põgeneda, pankrotistuda või hoopistükkis surra. Kuna taolised lahendused pakuvad aga väga kõrget mobiilsust ning ühilduvust, on nad siiski heaks abivahendiks väikeste igapäevatehingute teostamisel.

Käesoleva magistritöö raames loodav infosüsteem (*Bitplexus*) kuulub eeltoodud liigitust silmas pidades samuti nn *tsentraliseeritud rahakotiteenuste* ehk kolmanda kategooria rahakotitarkvarade alla.

2. Loodava infosüsteemi analüüs

Käesolevas peatükis viiakse läbi loodava infosüsteemi (Bitplexus) strateegiline ja detailanalüüs. Kuna tegu on äärmiselt kliendikeskse süsteemiga, on analüüsi esimeses lähenduses keskendutud eelkõige selle kliendi pädevusala vajaduste analüüsimisele. Süsteemi põhiobjektide ülesleidmiseks ning eesmärkide formuleerimiseks on läbi töötatud suur hulk kirjandusallikaid ning sarnaste infosüsteemide kasutajaliideseid. Viimaste põhjal on kokku pandud süsteemi funktsionaalsed nõuded, mis on omakorda aluseks selle kontseptuaalse andmemudeli koostamisele. Lisaks sellele on ära määratud ka infosüsteemi peamised mittefunktsionaalsed nõuded, mis on vastavuses finantsandmete haldamise parimate praktikatega. Tulemuseks on üldistatud domeenimudel plokiahelapõhiste krüptovaluutadega liidestumiseks, mida on võimalik kasutada ka mitmete teiste analoogiliste süsteemide arendamisel.

2.1 Infosüsteemi eesmärgid

Järgnevalt on toodud ülevaade loodava infosüsteemi peamistest eesmärkidest:

- Pakkuda klientidele mugavat viisi oma krüptoraha haldamiseks ilma sellega kaasnevate kohustuste (sh plokiahelate allalaadimine) ning vastutusteta (sh salajaste võtmete krüpteerimine ja varundamine).
- Võimaldada kliendil hallata kõiki oma krüptorahasid läbi ühe universaalse teenuse.
- Võimaldada kliendil genereerida uusi maksenõudeid (koos selle juurde käivate rekvisiitide, s.o standardse makse-URLi ning QR-koodiga).
- Pakkuda klientidele nn aadressiraamatu funktsionaalsust, mille abil on võimalik talletada kolmandate isikute aadresse.
- Võimaldada veebihalduril registreerida uusi plokiahelaid ning muuta olemasolevate valuutade üldandmeid.
- Anda klienditoe spetsialistile õigus vaadata teiste kasutajate andmeid ning vabastada nende kontosid blokeeringutest.
- Omada süstemaatilist teavet ettevõtte töötajate ning neile omistatud rollide kohta.

- Pakkuda kasutajatele turvalist ning usaldusväärset finantsteenust, rakendades selleks andmeturbe parimaid praktikaid (kaheastmeline autentimine, otspunktide vahelise suhtluse krüpteerimine (SSL/TLS) jne).
- Koguda andmeid kasutajate käitumise kohta (keskmise rahakoti väärtus, populaarseimad valuutad jmt), et parandada ettevõtte taktilise ja strateegilise taseme otsuste vastuvõtmist.

2.2 Pädevusalad

Nagu juba eespool mainitud, hõlmab vaadeldav terviksüsteem järgmist kolme pädevusala:

- kliendi pädevusala;
- veebihalduri pädevusala;
- klienditoe spetsialisti pädevusala.

Käesolevas peatükis keskendutakse eelkõige kliendi pädevusala analüüsimisele. Põhjuseks on, et valdav osa süsteemi tuumikfunktsionaalsusest on antud just kliendi pädevusalas. Ülejäänud pädevusalad sisaldavad eelkõige süsteemi põhiprotsesse toetavat abifunktsionaalsust, mida on rakendusele võimalik juurde lisada ka pärast selle esmaväljalaske evitamist.

2.3 Põhiobjektid

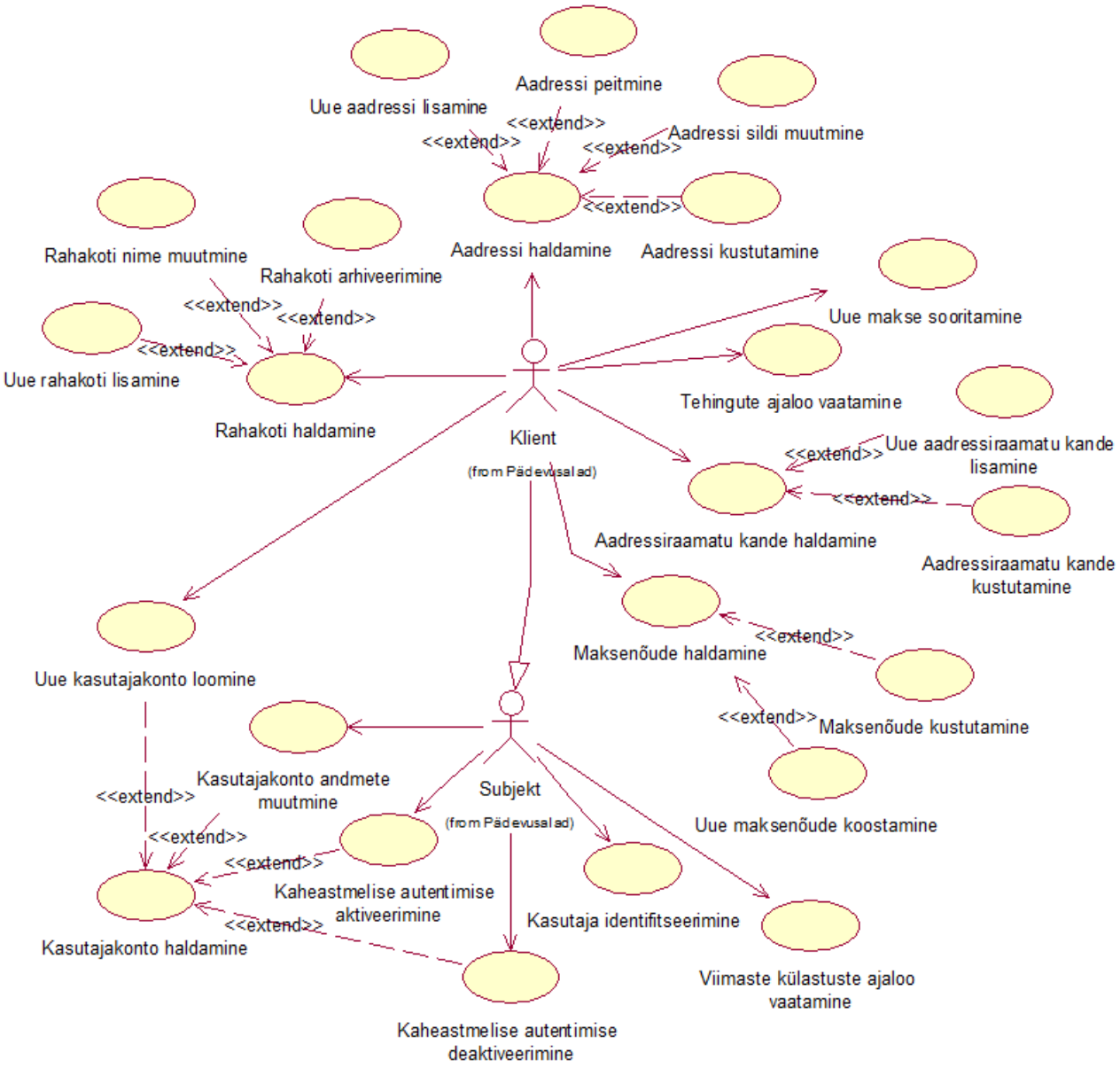
Kuna töö hilisemates etappides (sh kontseptuaalses andmemudelis ning mitmetel arhitektuuri skeemidel) kasutatakse põhiobjektidele viitamiseks eelkõige nende ingliskeelseid nimetusi, on iga põhiobjekti juures ära toodud ka selle ingliskeelne vaste:

- Klient (ingl k C*ustomer*);
- Töötaja (ingl k *Employee*);
- Töötaja roll (ingl k *Employee role*);
- Valuuta (ingl k *Currency*);
- Ahel (ehk Plokiahel) (ingl k *Chain*);
- Aadressi liik (ingl k *Address type*);
- Rahakott (ingl k *Wallet*);
- Aadress (ingl k *Address*);
- Tehing (ehk Makse) (ingl k *Transaction*);

- Tehingu otspunkt (ingl k *Transaction endpoint*);
- Aadressiraamatu kanne (ingl k *Address book entry*);
- Maksenõue (ingl k *Payment request*);
- Külastus (ingl k *Visit*).

2.4 Funktsionaalsed nõuded

Funktsionaalsete nõuete kaardistamiseks luuakse lihtsakoeline kasutusjuhtude mudel. See koosneb UML notatsioonile tuginevast kasutusjuhtude diagrammist (vt joonis 2), millele järgnevad kõigi analüüsitud kasutusjuhtude tekstikirjeldused kõrgtaseme formaadis.



Joonis 2. Kliendi pädevusala kasutusjuhtude diagramm

Kasutusjuht: Uue kasutajakonto loomine (FR-001)
Tegutsejad: Klient
Lühikirjeldus: Klient soovib registreeruda süsteemi kasutajaks, kuna on huvitatud selle poolt pakutavast funktsionaalsusest. Süsteem palub kliendil sisestada oma tulevased volitustõendid (kasutajanimi, parool) ning mõned isiku- ja kontaktandmed (ees- ja perekonnanimi, e-maili aadress ning mobiiltelefoni number). Seejärel esitatakse kliendile väljavõte rakenduse kasutustingimustest, millega nõustumisel viiakse kontoloomise protsess edukalt lõpule.

Kasutusjuht: Kasutaja identifitseerimine (FR-002)
Tegutsejad: Klient, veebihaldur, klienditoe spetsialist (e üldistatult Subjekt)
Lühikirjeldus: Subjekt soovib saada süsteemi poolt autenditud. Selleks edastab ta süsteemile oma kasutajanime, parooli ning ühekordse turvakoodi. Süsteem kontrollib kasutaja sisestusi, sh võrdleb neid andmebaasis talletatud väärtustega ning kasutusel olevate turvaalgoritmide väljunditega. Kui kõigi eeltoodud võrdluste tulemused vastavad oodatule, lubatakse subjektil süsteemi siseneda.

Kasutusjuht: Kasutajakonto andmete muutmine (FR-003)
Tegutsejad: Klient, veebihaldur, klienditoe spetsialist (e üldistatult Subjekt)
Lühikirjeldus: Subjekt uuendab oma isiku- ja/või kontaktandmeid, kuna need on aja jooksul muutunud (nt e-posti aadressi vahetus, neiupõlvenimest loobumine jmt).

Kasutusjuht: Kaheastmelise autentimise aktiveerimine (FR-004)
Tegutsejad: Klient, veebihaldur, klienditoe spetsialist (e üldistatult Subjekt)
Lühikirjeldus: Subjekt avaldab soovi kaheastmelise autentimise aktiveerimiseks. Süsteem kuvab subjektile TOTPi (ingl k *Time-based One-time Password*) algoritmil põhineva salajase võtme (nt QR-koodi kujul). Subjekt kasutab salajase võtme talletamiseks (ning edaspidi ka ühekordsete turvakoodide genereerimiseks) mõnda selleks ettenähtud mobiilirakendust (nt Authy, Google Authenticator vmt).

Kasutusjuht: Kaheastmelise autentimise deaktiveerimine (FR-005)
Tegutsejad: Klient, veebihaldur, klienditoe spetsialist (e üldistatult Subjekt)
Lühikirjeldus: Subjekt avaldab soovi kaheastmelise autentimise deaktiveerimiseks. Süsteem palub toimingu kinnitamiseks subjektilt tema kasutajakonto parooli ning antud ajahetkele vastavat ühekordset turvakoodi.

Kasutusjuht: Uue rahakoti lisamine (FR-006)

Tegutsejad: Klient

Lühikirjeldus: Klient soovib luua uut rahakotti, et eraldada teatud tüüpi tehingud oma konto ülejäänud finantstegevusest (nt eraldi rahakott meelelahutuskulude jaoks, laenude ja liisingute tagasimaksete jaoks jne). Igas rahakotis peaks olema võimalik kasutada kõiki süsteemi poolt toetatavaid krüptovaluutasid.

Kasutusjuht: Rahakoti nime muutmine (FR-007)

Tegutsejad: Klient

Lühikirjeldus: Klient uuendab rahakoti nime, kuna selle funktsioon või kasutusotstarve on aja jooksul muutunud (nt *Juhani igapäevatehingud* → *Juhani pikaajalised hoiused*).

Kasutusjuht: Rahakoti arhiveerimine (FR-008)

Tegutsejad: Klient

Lühikirjeldus: Klient valib rahakottide põhivaatest rahakoti, mis on oma kasutusotstarbe täitnud, ning avaldab soovi selle arhiveerimiseks. Arhiveeritud rahakotte ei saa kasutada uute maksete saatmiseks ega vastuvõtmiseks, kuid nende üldandmed, sh tehingute ajalugu, jäävad endiselt kättesaadavaks (seega vajadusel on tagatud nn *raamatupidamise algdokumentide* säilimine kliendile).

Kasutusjuht: Uue aadressi lisamine (FR-009)

Tegutsejad: Klient

Lühikirjeldus: Klient soovib luua uut krüptovaluuta aadressi, et kasutada seda nt täiendavate maksete vastuvõtmiseks või oma varade ümberpaigutamiseks. Selleks valib ta välja valuuta ning ahela, mille tarvis uut aadressi luua ning lisab sellele ülevaatliku sisuga sildi. Süsteem genereerib valitud valuuta/ahela süntaksireeglitele vastava aadressi valmis ning uuendab aadresside põhivaadet.

Kasutusjuht: Aadressi peitmine (FR-010)

Tegutsejad: Klient

Lühikirjeldus: Klient valib aadresside põhivaatest aadressi, mida peab seal üleliigseks, ning avaldab soovi selle peitmiseks. Aadresside peitmise põhieesmärgiks on eemaldada kasutajaliidesest sellised aadressid, mis ei oma kliendi jaoks sisulist tähendust, kuid mis hõlmavad endas siiski reaalset rahalist väärtust (nt eelmistest tehingutest üle jäänud

peenraha). Peidetud aadressidel paiknev raha arvatakse kontojäägi hulka ning seda kasutatakse ka uute maksete finantseerimiseks.

Kasutusjuht: Aadressi sildi muutmine (FR-011)
Tegutsejad: Klient
Lühikirjeldus: Klient uuendab aadressi silti, kuna selle funktsioon või kasutusotstarve on aja jooksul muutunud (nt *Tagastusraha aadress #1 → Jalgratta „Trek 4500D" müügitulu*).

Kasutusjuht: Aadressi kustutamine (FR-012)
Tegutsejad: Klient
Lühikirjeldus: Kliendil on võimalik aadress kasutajaliidesest lõplikult eemaldada, kui ta on kindel, et sellele ei saadeta tulevikus enam ühtegi uut makset. Aadressi kustutamine on pöördumatu tegevus.

Kasutusjuht: Uue makse sooritamine (FR-013)
Tegutsejad: Klient
Lühikirjeldus: Klient soovib saata teatud rahasummat mingile rahakotivälisele krüptovaluuta aadressile. Selleks valib ta välja tehingu finantseerimiseks kasutatava rahakoti ning täpsustab valuuta, milles makse sooritatakse. Valitud valuuta peab vastama adressaadi oodatavale valuutale – vastasel juhul saaja aadressi kodeeritud kuju valideerimine ebaõnnestub ning makset ei toimu. Vajadusel saab tehingule lisada ka lühikese kommentaari, mida ei kanta aga üle krüptovaluuta võrgustiku ning mis on seetõttu nähtavad vaid kliendile endale.

Kasutusjuht: Tehingute ajaloo vaatamine (FR-014)
Tegutsejad: Klient
Lühikirjeldus: Klient soovib saada ülevaadet mõne oma rahakoti kõigist viimastest tehingutest. Süsteem kuvab tehingute loetelu kronoloogiliselt kahanevas järjekorras, kusjuures iga tehingu kohta on ära toodud selle toimumise kuupäev ja kellaaeg, lähteaadress(id), sihtaadress(id), võrgustikult saadud kinnituste arv, tehingu seisund ning makse kogusumma. Vajadusel on kliendil võimalik vaadata ka iga tehingu juurde käivaid detailandmeid.

Kasutusjuht: Uue aadressiraamatu kande lisamine (FR-015)
Tegutsejad: Klient

Lühikirjeldus: Klient soovib talletada kolmanda isiku makseandmed, et kasutada neid hiljem uute maksete sooritamiseks. Selleks edastab ta süsteemile vastava aadressi kodeeritud kuju koos kirjeldava sildiga, mille abil see hiljem aadressiraamatust uuesti üles leida.

Kasutusjuht: Aadressiraamatu kande kustutamine (FR-016)
Tegutsejad: Klient
Lühikirjeldus: Klient otsustab kolmanda isiku makseandmed kustutada, kuna need on oma eesmärgi täitnud (nt mitmeosaline laen saab täielikult tasutud) või muutunud kasutuskõlbmatuks (nt adressaat teatab, et kaotas kõvaketta vea tõttu ligipääsu oma rahakotile).

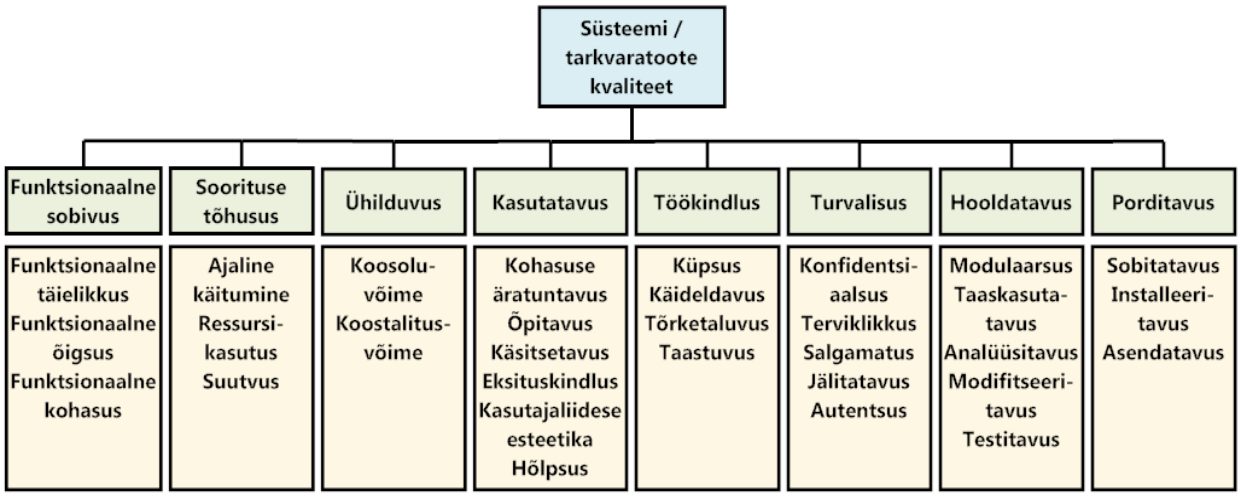
Kasutusjuht: Uue maksenõude koostamine (FR-017)
Tegutsejad: Klient
Lühikirjeldus: Klient soovib, et teine isik saadaks talle teatud rahasumma (nt müüdud kauba, pakutud teenuse või mõne muu hüve eest). Selleks valib ta oma rahakotist välja sobiva sihtaadressi, täpsustab nõutud summa ning lisab vajadusel ka lühikese makseselgituse. Süsteem genereerib sisestatud andmete põhjal standardse makse-URLi ning QR-koodi, mida on võimalik kasutada erinevates rahakotitarkvarades makse kiireks sooritamiseks (Corallo jt 2012).

Kasutusjuht: Maksenõude kustutamine (FR-018)
Tegutsejad: Klient
Lühikirjeldus: Klient avaldab soovi maksenõude kustutamiseks, kuna see on juba saaja poolt täidetud (ühekordse maksenõude puhul) või kaotanud oma aktuaalsuse (nt maksenõue, millele vastavat QR-koodi kasutati e-poe sortimendist välja praagitud toote juures).

Kasutusjuht: Viimaste külastuste ajaloo vaatamine (FR-019)
Tegutsejad: Klient, veebihaldur, klienditoe spetsialist (e üldistatult Subjekt)
Lühikirjeldus: Subjekt soovib näha loetelu oma viimastest külastustest antud portaali. Süsteem kuvab ülevaate kõigist konto turvalisust puudutavatest sündmustest, sh iga sündmuse kuupäeva ja kellaaja, sündmuse liigi, tegutseja IP-aadressi ning IP-aadressi päritoluriigi.

2.5 Mittefunktsionaalsed nõuded

Süsteemi mittefunktsionaalsete nõuete määratlemisel lähtutakse standardis EVS-ISO/IEC 25010:2011 kirjeldatud tootekvaliteedi mudelist. Tegemist on ühe levinuima tarkvara kvaliteedinäitajate süsteemiga, mis on edasiarenduseks standardis ISO/IEC 9126 defineeritud tootekvaliteedi mudelile (Süsteemide ja tarkvara ... 2012, V). Eelnimetatud mudel kujutab endast mitmetasemelist kvaliteediatribuutide hierarhiat, mille iga kriteerium (nt turvalisus) ning sellele vastavad alamkriteeriumid (nt terviklikkus, salgamatus) kirjeldavad tüüpilise tarkvaratoote teatud liiki kvaliteediomadusi (vt joonis 3). Toodud mudeli kasutamise eeliseks on, et see võimaldab paremini organiseerida süsteemile esitatavaid mittefunktsionaalseid nõudeid ning aitab tagada, et süsteem saaks nõuetega ühtlaselt kaetud.



Joonis 3. Tarkvarasüsteemi tootekvaliteedi mudel (EVS-ISO/IEC 25010:2011 järgi)

Nõude ID: NFR-001
Faktor / kriteerium: Sooritуse tõhusus / ajaline käitumine
Lühikirjeldus: Kõik veebiliidese kaudu tehtavad toimingud tuleb lõpule viia mõistliku aja (< 5 s) jooksul. Eelduseks on süsteemi madal (0 – 30 samaaegset kasutajat) kuni keskmine (31 – 70 samaaegset kasutajat) hetkekoormus.

Nõude ID: NFR-002
Faktor / kriteerium: Sooritуse tõhusus / ressursikasutus
Lühikirjeldus: Rakendusserverile langeva koormuse vähendamiseks ning mälukasutuse optimeerimiseks tuleks võimalikult suur osa pakktöötlusliku (ingl k *batch processing*)

iseloomuga toimingutest (nt tehingute kinnitamine) realiseerida andmebaasiserveris talletatud rutiinidena.

Nõude ID: NFR-003
Faktor / kriteerium: Ühilduvus / koostalitusvõime
Lühikirjeldus: Süsteemi poolt väljastatav HTML ja CSS kood peab vastama W3C (World Wide Web Consortium) spetsifikatsioonides kirjeldatud soovitustele ning nõudmistele (standardite versioonid: HTML5 ja CSS3).

Nõude ID: NFR-004
Faktor / kriteerium: Kasutatavus / käsitsetavus
Lühikirjeldus: Süsteemi kasutajaliides peab olema lihtne ja intuitiivne – ühegi alamvormini navigeerimiseks ei tohiks kuluda rohkem kui neli hiirevajutust.

Nõude ID: NFR-005
Faktor / kriteerium: Kasutatavus / eksitusekindlus
Lühikirjeldus: Ebakorrektsete väärtuste sisestamisel näidatakse kasutajale selgitavat veateadet, mitte programmi täitmisel heidetud erindit. Kasutajaliideses ei tohiks sisalduda ühtegi viidet vea pinurajale (ingl k *stack trace*) ega muudele tehnilistele detailidele.

Nõude ID: NFR-006
Faktor / kriteerium: Kasutatavus / hõlpsus
Lühikirjeldus: Süsteemi poolt väljastatav HTML kood peab sisaldama WAI-ARIA 1.0 (Web Accessibility Initiative – Accessible Rich Internet Applications) spetsifikatsioonis kirjeldatud spetsiaalseid elemente ja atribuute, mis aitavad parandada rakenduse ligipääsetavust erivajadustega inimestele.

Nõude ID: NFR-007
Faktor / kriteerium: Töökindlus / tõrketaluvus
Lühikirjeldus: Talitushäired krüptovaluuta taustaprotsessides või muudes välistes programmiliidestes ei tohiks häirida terviksüsteemi tööd. Sellistele häiretele tuleb reageerida vastava funktsionaalsuse ajutise blokeerimisega, mitte teenuse üleüldise maasolekuga.

Nõude ID: NFR-008

Faktor / kriteerium: Töökindlus / taastuvus

Lühikirjeldus: Süsteemi poolt hallatavaid andmeid tuleb jooksvalt varundada (andmebaasi tiražeerimine, logifailide igaminutiline kopeerimine), et minimeerida tõrgetest põhjustatud majanduslikku kahju ning rakenduse taastamiseks kuluda võivat aega.

Nõude ID: NFR-009

Faktor / kriteerium: Turvalisus / konfidentsiaalsus

Lühikirjeldus: Registreeritud kasutajate paroole tohib andmebaasis hoida vaid räsitud kujul. Selleks oleks mõistlik kasutada spetsiaalset räsifunktsiooni nimega *bcrypt* (iteratsioonide arv: $\geq$ 12). Vältida tuleks aegunud ja/või osaliselt murtud räsifunktsioonide nagu MD5 ning SHA-1 kasutamist.

Nõude ID: NFR-010

Faktor / kriteerium: Turvalisus / konfidentsiaalsus

Lühikirjeldus: Süsteemi erinevaid osi ühendavad suhtluskanalid peavad olema kaitstud transpordikihi turbeprotokolliga (SSL/TLS). Lisaks süsteemisisestele muudatustele eeldab see ka usaldusväärse SSL-sertifikaadi paigaldamist rakenduse majutamiseks kasutatavale serverile.

Nõude ID: NFR-011

Faktor / kriteerium: Turvalisus / salgamatus

Lühikirjeldus: Kõik süsteemi kasutajate poolt algatatud andmemuudatused ning rakenduse töö käigus esinenud vead tuleb jäädvustada andmebaasist eraldiseisvas logifailis. Mitteaktuaalseid logifaile on nõutav säilitada vähemalt seitse aastat pärast nende viimase sissekande tegemise hetke.

Nõude ID: NFR-012

Faktor / kriteerium: Hooldatavus / modifitseeritavus

Lühikirjeldus: Süsteem peab võimaldama uute lokaatide (ingl k *locale*) lisamist ilma, et selleks oleks vaja muuta rakenduse varasemat koodi. Taolise võimekuse eelduseks on korraliku mitmekeelsuse toe väljaehitamine kõigis rakenduse kihtides.

Nõude ID: NFR-013
Faktor / kriteerium: Porditavus / sobitatavus
Lühikirjeldus: Süsteem ei tohi olulisel määral sõltuda seda ümbritseva käituskeskkonna eripäradest (st operatsioonisüsteemist, absoluutsetest kataloogiteedest, eksootilistest Java virtuaalmasina parameetritest jmt).

Nõude ID: NFR-014
Faktor / kriteerium: Porditavus / sobitatavus
Lühikirjeldus: Süsteemi tööks vajaliku rakendusserveri tarkvara väljavahetamine ei tohiks olla väga ajamahukas protsess. Süsteem peab võimaldama üleminekut Apache Tomcatilt mistahes teisele Java rakendusserveri tarkvarale (nt WildFly).

2.6 Ärireeglid

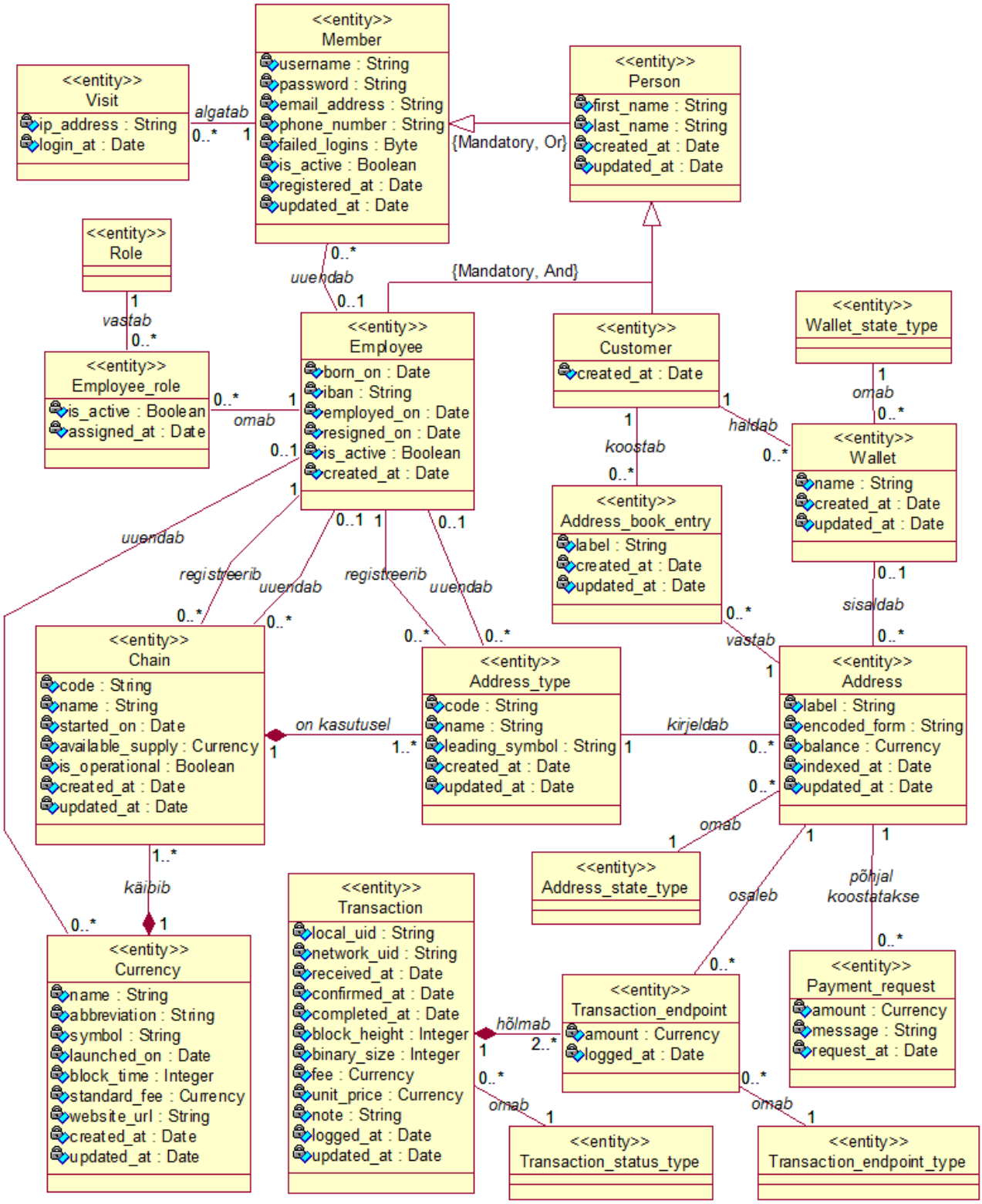
Järgnevalt esitatakse infosüsteemi ärireeglid, mis annavad täpsema ülevaate süsteemi põhiobjektide vahel valitsevatest seostest ning on aluseks kontseptuaalse andmemudeli koostamisele.

- Kliendil on õigus luua üks või enam rahakotti.
- Iga rahakott peab kuuluma ühele ja ainult ühe kliendile.
- Rahakott võib sisaldada ühte või enamat aadressi.
- Aadress võib olla aluseks ühele või enamale tehingu otspunktile.
- Iga tehingu otspunkt peab põhinema ühel ja ainult ühel aadressil.
- Tehing peab hõlmama kahte või enamat tehingu otspuntki (vähemalt ühte sisendit ja ühte väljundit).
- Iga tehingu otspunkt peab kuuluma ühe ja ainult ühe tehingu koosseisu.
- Klient võib luua üks või enam aadressiraamatu kannet.
- Klient võib esitada üks või enam maksenõuet.
- Iga aadress peab vastama ühele ja ainult ühele aadressi liigile.
- Iga aadressi liik kirjeldab ühte ja ainult ühte ahelasse kuuluvaid aadresse.
- Ahelas peab korraga kasutusel olema üks või enam aadressi liiki.
- Valuuta peab koosnema ühest või enamast paralleelselt toimivast plokiahelast.
- Töötaja võib (vastavalt protokolli muudatustele) defineerida ühe või enam uut aadressi liiki.

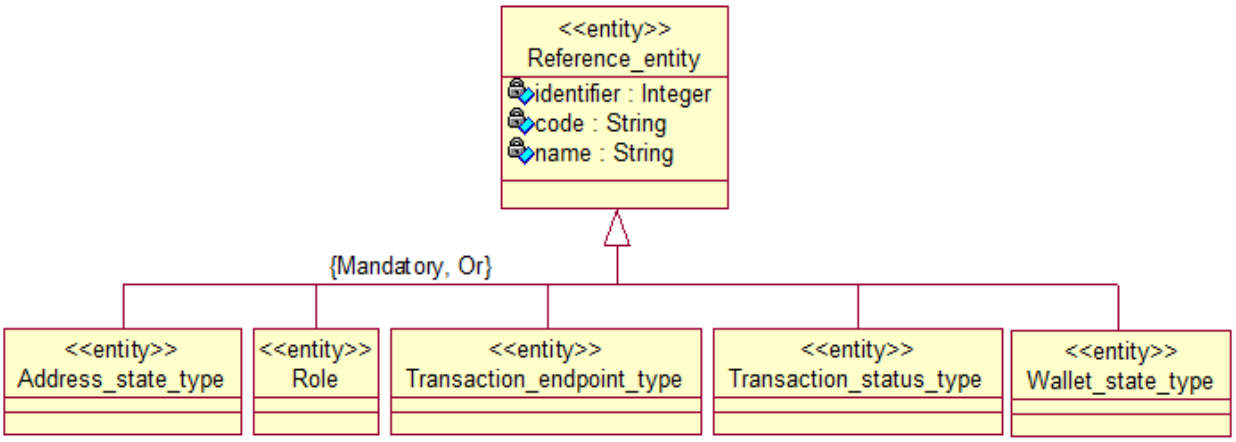
- Töötaja võib (vastavalt protokolli muudatustele) luua ühe või enam uut ahelat.
- Kontojäägi arvutamisel võetakse arvesse vaid nendel aadressidel paiknevaid rahalisi vahendeid, mille seisundiks on *Aktiivne*, *Kasutatud* või *Peidetud*.
- Kontojäägi arvutamisel võetakse arvesse vaid nendest tehingutest laekunud rahalisi vahendeid, mille seisundiks on *Lõpetatud*.
- Kliendil on võimalik enda loodud rahakotti hallata siis ja ainult siis, kui selle seisundiks on *Loodud* või *Aktiivne*.
- Aadressi kustutamisel (seisund *Kustutatud*) kaotab klient kõik õigused selle edasiseks haldamiseks; aadressile tulevikus laekuda võivad summad ei kuulu tagastamisele.
- Maksenõudeid saab vormistada vaid sellistele aadressidele, mis kuuluvad mõnda kliendi isiklikest rahakottidest.
- Tehingu teenustasu peab võrduma selle sisendite väärtuste summa ning väljundite väärtuste summa vahega, st kehtib tingimus: $/\,\text{Tehing.teenustasu} = \sum_{i=0}^{n}(\text{Tehingu\_otspunkt}_{SISEND\ i}.\text{väärtus}) - \sum_{i=0}^{n}(\text{Tehingu\_otspunkt}_{VÄLJUND\ i}.\text{väärtus})$.
- Aadressi kodeeritud kuju versiooniprefiks (esimene tähemärk) peab langema kokku sellele vastava aadressi liigi versiooniprefiksiga.
- Süsteemi kasutaja võib samaaegselt olla nii klient kui ka ettevõtte töötaja.

2.7 Kontseptuaalne andmemudel

Lähtudes süsteemianalüüsi käigus kaardistatud põhiobjektidest ning rakendusele esitatud funktsionaalsetest ja mittefunktsionaalsetest nõuetest, on koostatud vaadeldava terviksüsteemi esialgne andmemudel ning seda kirjeldav olemi-suhte diagramm (vt joonised 4 ja 5).



Joonis 4. Terviksüsteemi esialgne olemi-suhte diagramm (1/2)



Joonis 5. Terviksüsteemi esialgne olemi-suhte diagramm (2/2)

Kõigi joonistel 4/5 kujutatud olemitüüpide ning olemitüüpide atribuutide definitsioonid on täpsemalt lahti kirjutatud tabelites 1 ja 2.

Tabel 1. Olemitüüpide definitsioonid

Olemitüübi nimi	Definitsioon
Member	Süsteemis kasutajakontot omav tegutseja, kellel on vaadeldava ettevõtte suhtes mingid kindlad ootused ja/või kohustused.
Person	Kasutajakontole ligipääsu omav füüsiline isik või juriidilise isiku seadusjärgne esindaja.
Customer	Isik, kes on ettevõttega seotud läbi soovi tarbida selle ettevõtte poolt pakutavaid teenuseid.
Employee	Ettevõttega töösuhet omav isik, kelle eesmärgiks on täita talle töölepingu alusel määratud ametikohustusi.
Employee_role	Töötaja kuulumine teatud rolli esindajate hulka.
Role	Privileegide ja juurdepääsuõiguste kogum, mis on vajalik teatud tüüpi tööülesannete täitmiseks. Näiteks: veebihaldur, klienditoe spetsialist, andmebaasi administraator.
Currency	Rahasüsteem, milles käibivad rahaühikud täidavad kõiki raha funktsioone, sh on kasutatavad üldotstarbelise maksevahendina. Näiteks: Bitcoin, Litecoin, Dash, Namecoin.
Chain	Järk-järgult kasvav hajus andmebaas, mis sisaldab kõigi antud valuutas sooritatud tehingute andmeid ning mida tiražeeritakse iga krüptovaluuta taustaprotsessi käitava serverarvuti kõvakettale. Näiteks:

Olemitüübi nimi	Definitsioon
	Bitcoini *mainnet*-ahel, Bitcoini *testnet3*-ahel, Litecoini *mainnet*-ahel.
Address_type	Sarnaste tehniliste omadustega krüptovaluuta aadresse koondav kategoriseering. Näiteks: harilikud (P2PKH) Bitcoini aadressid, skriptipõhised (P2SH) Bitcoini aadressid.
Wallet	Kliendi poolt hallatav krüptovaluuta aadresside kogum, mis võimaldab eraldada teatud tüüpi tehingud kasutajakonto ülejäänud finants-tegevusest (nt eraldi rahakott olmekulude jaoks, lühi- ja pikaajaliste säästude jaoks jne).
Wallet_state_type	Rahakoti võimalikud seisundis süsteemis. Näiteks: loodud, aktiivne, arhiveeritud, kustutatud.
Address	Lähte- ja sihtpunkt uute krüptovaluuta tehingute saatmiseks ning vastuvõtmiseks. Vähim informatsioon, mida teades on võimalik teisele osapoolele krüptovaluuta võrgustiku kaudu raha saata. Võrreldav traditsioonilise pangakonto numbriga.
Address_state_type	Aadressi võimalikud seisundid süsteemis. Näiteks: reserveeritud, aktiivne, kasutatud, peidetud.
Transaction	Toiming, mille käigus konto laekumiste tulemusena kogunenud lunastamata krüptovaluuta väljundid (e vaba kontojääk) kantakse ühelt või enamalt krüptovaluuta aadressilt (e lähteaadressi(de)lt) edasi ühele või enamale järgmisele krüptovaluuta aadressile (e sihtaadressi(de)le).
Transaction_status_ type	Tehingu võimalikud seisundid süsteemis. Näiteks: kinnitamata, kinnitatud, lõpetatud, nurjunud.
Transaction_ endpoint	Aadressi kuulumine teatud tehingu sisendite või väljundite hulka. Tehingu otspunkt on tehingu komposiitosa, mis ei saa kuuluda korraga enam kui ühe tehingu koosseisu ega eksisteerida tehingust sõltumatult.
Transaction_ endpoint_type	Tehingu otspunkti roll vaadeldavas tehingus. Näiteks: sisend, põhiväljund, tagastusraha väljund.
Address_book_entry	Kontaktkirje, mis koosneb rahakotivälisest krüptovaluuta aadressist ning viimast kirjeldavast sildist. Võimaldab kliendil talletada kolmandate isikute makseandmeid, et kasutada neid hiljem uute maksete sooritamiseks.
Payment_request	Kliendi poolt kolmandale isikule esitatav dokument, milles nõutakse

Olemitüübi nimi	Definitsioon
	teatud rahasumma tasumist mingile etteantud krüptovaluuta aadressile. Maksenõue esitatakse kohustatud osapoolele reeglina kas standardse makse-URLi või sellele vastava QR-koodi näol.
Visit	Kasutaja sisselogimiskatset kajastav sündmus, milles on toodud selle toimumise kuupäeva ja kellaaeg, tegutseja IP-aadress ning IP-aadressi päritoluriik.
Reference_entity	„Mistahes andmed, mida kasutatakse andmebaasis teiste andmete liigitamiseks või andmebaasis olevate andmete seostamiseks väljaspool organisatsiooni vastutusala oleva informatsiooniga.“ (Chisholm 2000)

Tabel 2. Olemitüüpide atribuutide definitsioonid

Olemitüübi nimi	Atribuudi nimi	Definitsioon	Näiteväärtus
Member	username	Liikmele omistatud unikaalne (avalik) nimetus, mida kasutatakse tema süsteemi-poolseks autentimiseks.	adam.wright
Member	password	Salajane fraas või sõne, mida kasutatakse liikme süsteemipoolseks autentimiseks.	Ru$tyf0rk38
Member	email_address	Aadress, mis võimaldab liikmega ühendust võtta arvutivõrgu vahendusel.	adam.wright @gmx.com
Member	phone_number	Numbrikombinatsioon, mis võimaldab liikmega ühendust võtta telefonivõrgu vahendusel.	+44 141 519 3781
Member	failed_logins	Liikme ebaõnnestunud sisselogimiste arv (al. viimasest õnnestunud sisselogimisest).	0
Member	is_active	Tõeväärtus, mis näitab, kas liikmel on õigus süsteemi siseneda (TRUE) või mitte (FALSE).	TRUE
Member	registered_at	Kuupäev ja kellaaeg, mil liige avaldas soovi käesoleva kasutajakonto loomiseks.	2015-03-19 23:19:57
Member	updated_at	Kuupäev ja kellaaeg, mil kasutajakonto andmeid viimati muudeti (kas liikme enda	2015-03-19 23:41:36

Olemitüübi nimi	Atribuudi nimi	Definitsioon	Näiteväärtus
		või mõne ettevõtte töötaja poolt).	
Person	first_name	„Lapsele tema sünni registreerimisel antav nimi. Eesnimi on osa isikunimest.“ (EKSS 2009 *sub* eesnimi)	Adam
Person	last_name	„Vanemailt lapsele kanduv nimi. Perekonnanimi on osa isikunimest.“ (EKSS 2009 *sub* perekonnanimi)	Wright
Person	created_at	Kuupäev ja kellaaeg, mil isiku andmed esmakordselt süsteemi sisestati.	2015-03-19 23:19:57
Person	updated_at	Kuupäev ja kellaaeg, mil isiku andmeid viimati muudeti.	2015-03-19 23:41:36
Customer	created_at	Kuupäev ja kellaaeg, mil kliendi andmed esmakordselt süsteemi sisestati.	2015-03-26 18:42:39
Employee	born_on	Kuupäev, mil töötaja on sündinud.	1978-09-18
Employee	iban	Töötaja rahvusvaheline kontonumber, mis identifitseerib üheselt tema mistahes riigi mistahes pangas asuva pangakonto.	GB81NRNB40 642573908441
Employee	employed_on	Kuupäev, mil töötaja alustas vaadeldavas ettevõttes töötamist.	2015-03-02
Employee	resigned_on	Kuupäev, mil töötaja lõpetas vaadeldavas ettevõttes töötamise.	2015-10-28
Employee	is_active	Tõeväärtus, mis näitab, kas süsteemi liikmele omistatud töötaja roll on aktuaalne (TRUE) või mitte (FALSE).	TRUE
Employee	created_at	Kuupäev ja kellaaeg, mil töötaja andmed esmakordselt süsteemi sisestati.	2015-03-19 23:19:57
Employee_ role	is_active	Tõeväärtus, mis näitab, kas ettevõtte töötajale omistatud ametiroll on aktuaalne (TRUE) või mitte (FALSE).	FALSE
Employee_ role	assigned_at	Kuupäev ja kellaaeg, mil töötaja lisati vaadeldava ametirolli esindajate hulka.	2015-03-19 23:22:58

Olemitüübi nimi	Atribuudi nimi	Definitsioon	Näiteväärtus
Currency	name	Valuuta ametlik nimetus.	Bitcoin
Currency	abbreviation	Valuuta ametlik lühend, mis vastab ISO 4217 standardis kirjeldatud valuutakoodi formaadile.	XBT
Currency	symbol	Valuuta ametlik tähis.	฿
Currency	launched_on	Kuupäev, mil avalikustati käesoleva krüptovaluuta võrgusõlme tarkvara (sh rahakoti- ja kaevandamistarkvara) esialgne näidisimplementatsioon (ingl k *reference implementation*).	2009-01-03
Currency	block_time	Eeldatav ajahulk, mis kulub vastava krüptovaluuta kaevurite võrgustikul keskmiselt iga järgneva ploki kaevandamiseks. Mõõtühikuks on sekund (s).	600
Currency	standard_fee	Tehingu soovituslik teenustasu iga 1 kB andmete kohta, mida tahetakse vastava valuuta plokiahelasse kirjutada. Kõik krüptovaluutades väljendatud rahasummad tuleb esitada täpsusega vähemalt kaheksa kohta peale koma.	0.00010000
Currency	website_url	Valuuta ametlikule koduleheküljele viitav internetiaadress.	https://bitcoin.org/
Currency	created_at	Kuupäev ja kellaaeg, mil valuuta andmed esmakordselt süsteemi sisestati.	2015-03-19 23:48:41
Currency	updated_at	Kuupäev ja kellaaeg, mil valuuta andmeid viimati muudeti.	2015-03-19 23:55:07
Chain	code	Ahelat üheselt identifitseeriv tähtnumbriline kood. Mõeldud vaid süsteemisiseseks kasutamiseks.	BITCOIN_MAIN
Chain	name	Ahela ametlik nimetus.	Mainnet chain

Olemitüübi nimi	Atribuudi nimi	Definitsioon	Näiteväärtus
Chain	started_on	Kuupäev, mil kaevandati käesoleva plokiahela esimene plokk (ingl k *genesis block*), e pandi alus vastava valuuta tehinguandmete hajusale andmebaasile.	2009-01-03
Chain	available_ supply	Vaadeldavaks ajahetkeks antud plokiahelas ringlusesse lastud raha kogumaht.	14765825.000 00000
Chain	is_operational	Tõeväärtus, mis näitab, kas vastav ahel on täielikult töökorras ning klientidele kättesaadav (TRUE) või mitte (FALSE).	TRUE
Chain	created_at	Kuupäev ja kellaaeg, mil ahela andmed esmakordselt süsteemi sisestati.	2015-03-19 23:52:13
Chain	updated_at	Kuupäev ja kellaaeg, mil ahela andmeid viimati muudeti.	2015-03-19 23:57:04
Address_type	code	Aadressi liiki üheselt identifitseeriv tähtnumbriline kood. Mõeldud vaid süsteemisiseseks kasutamiseks.	BITCOIN_ P2PKH_MAIN
Address_type	name	Aadressi liigi ametlik nimetus.	P2PKH address
Address_type	leading_ symbol	Aadressi liigi versiooniprefiks. Kõigi antud tüüpi aadresside kodeeritud kujud (atribuut *encoded_form*) peavad algama eeltoodud tähemärgiga.	m
Address_type	created_at	Kuupäev ja kellaaeg, mil aadressi liigi andmed esmakordselt süsteemi sisestati.	2015-03-20 00:06:21
Address_type	updated_at	Kuupäev ja kellaaeg, mil aadressi liigi andmeid viimati muudeti.	2015-03-20 00:13:49
Wallet	name	Rahakotti kuuluvaid aadresse ning nendega tehtavaid tehinguid kirjeldav lühike nimetus.	Juhani pikaaja-lised hoiused
Wallet	created_at	Kuupäev ja kellaaeg, mil rahakoti andmed esmakordselt süsteemi sisestati.	2015-03-26 20:26:08
Wallet	updated_at	Kuupäev ja kellaaeg, mil rahakoti andmeid	2015-03-26

Olemitüübi nimi	Atribuudi nimi	Definitsioon	Näiteväärtus
		viimati muudeti.	20:37:44
Address	label	Aadressi kasutusotstarvet kirjeldav silt.	Skulptuuri „The Trickster“ müügitulu
Address	encoded_form	Krüptovaluuta taustaprotsessi poolt genereeritav 26- kuni 35-kohaline tähtnumbriline kood, mis tähistab üht võimalikku sihtpunkti antud krüptovaluuta maksete jaoks (Lodi 2014). Aadresside tuletamise protsess võib valuutade vahel veidi erineda, kuid üldjuhul kujutab see endast tehingute allkirjastamiseks kasutatava krüptograafilise võtmepaari avaliku poole mingisugust teisendust (nt kärpimine, räsimine, kodeerimine jne). Tuntud ka kui *aadressi kodeeritud kuju*.	mqQSkX987x665dPfr3inveo6pQstB3j9XU
Address	balance	Eelmiste tehingute tulemusena aadressile kogunenud lunastamata väljundite summa (e aadressi saldo).	3.82240000
Address	indexed_at	Kuupäev ja kellaaeg, mil aadressi andmed esmakordselt süsteemi sisestati.	2015-05-31 07:58:13
Address	updated_at	Kuupäev ja kellaaeg, mil aadressi andmeid viimati muudeti.	2015-06-02 13:20:00
Transaction	local_uid	Süsteemi poolt tehingule omistatav 16-baidine unikaalne identifikaator, mis dubleerib krüptovaluuta taustaprotsessi poolt genereeritavat identifikaatorit (atribuut *network_uid*). Mõeldud vaid süsteemisiseseks kasutamiseks.	0190695a-1c35-402a-9122-ba9e32d2d0cc
Transaction	network_uid	Krüptovaluuta taustaprotsessi poolt	4b88d8e0900ff

Olemitüübi nimi	Atribuudi nimi	Definitsioon	Näiteväärtus
		tehingule omistatav 32-baidine unikaalne identifikaator (ehk nn *TxID*). Esitatakse harilikult 64-kohalise kuueteistkümnend-koodina.	8ae1445603a9e48ebe96f303b5d2f81796fd9c7ed1e3975da89
Transaction	received_at	Kuupäev ja kellaaeg, mil kohalik krüpto-valuuta võrgusõlm sai teada antud tehingu olemasolust.	2015-06-02 13:28:53
Transaction	confirmed_at	Kuupäev ja kellaaeg, mil tehing sai võrgustikult esimese kinnituse (e lisati jäädavalt plokiahelasse).	2015-06-02 15:30:05
Transaction	completed_at	Kuupäev ja kellaaeg, mil tehing sai võrgustikult *n*-inda kinnituse (kus *n* = valuutaspetsiifiline kinnituste arv, mille järel võib tehingu tagasipööratavust küberrünnakute tulemusena pidada praktiliselt võimatuks).	2015-06-02 15:36:30
Transaction	block_height	Järjekorranumber, mis viitab plokiahela plokile, millesse on talletatud käesoleva tehingu andmed.	446465
Transaction	binary_size	Binaarkujule teisendatud tehinguandmete maht. Mõõtühikuks on bait (B).	192
Transaction	fee	Tehingule lisatav teenustasu. Optimaalse väärtuse arvutamisel lähtutakse valuuta teenustasu standardmäärast (atribuut *standard_fee*) ning tehinguandmete mahust (atribuut *binary_size*).	0.00020000
Transaction	unit_price	Tehingu aluseks oleva valuuta kurss USA dollari suhtes tehingu sooritamise hetkel. Kõik klassikalistes (ingl k *fiat*) valuutades väljendatud rahasummad tuleb esitada	222.89

Olemitüübi nimi	Atribuudi nimi	Definitsioon	Näiteväärtus
		täpsusega vähemalt kaks kohta pärast koma.	
Transaction	note	Vabas vormis kommentaar, mida ei kanta üle krüptovaluuta võrgustiku ning mis on seetõttu nähtav vaid kliendile endale.	Autoritasu pildi „The Woodsie Lord“ kasutamise eest.
Transaction	logged_at	Kuupäev ja kellaaeg, mil tehingu andmed esmakordselt süsteemi sisestati.	2015-06-02 13:28:53
Transaction	updated_at	Kuupäev ja kellaaeg, mil tehingu andmeid viimati muudeti.	2015-06-02 15:36:30
Transaction_ endpoint	amount	Tehingu otspunkti rahaline väärtus (e millise rahasumma võrra suurenes või vähenes vastava aadressi lõppsaldo).	0.10020000
Transaction_ endpoint	logged_at	Kuupäev ja kellaaeg, mil tehingu otspunkti andmed esmakordselt süsteemi sisestati.	2015-06-02 13:28:53
Address_ book_entry	label	Rahakotivälist (st kolmandale isikule kuuluvat) krüptovaluuta aadressi kirjeldav lühikene silt.	Bitstampi sissemaksete aadress #1
Address_ book_entry	created_at	Kuupäev ja kellaaeg, mil aadressiraamatu kande andmed esmakordselt süsteemi sisestati.	2015-05-31 08:41:26
Address_ book_entry	updated_at	Kuupäev ja kellaaeg, mil aadressiraamatu kande andmeid viimati muudeti.	2015-05-31 08:47:51
Payment_ request	amount	Maksenõude saaja poolt tasumisele kuuluv summa.	0.75000000
Payment_ request	message	Vabatekstiline sõnum, mis võimaldab maksenõudele lisada täiendavaid andmeid. Nähtav ka maksenõude saajale (kodeeri-takse nii standardse makse-URLi kui ka sellele vastava QR-koodi hulka).	Palun tasuda auto igakuine liisingumakse (2015. juuni) – 0.75 BTC.
Payment_	requested_at	Kuupäev ja kellaaeg, mil klient avaldas	2015-05-31

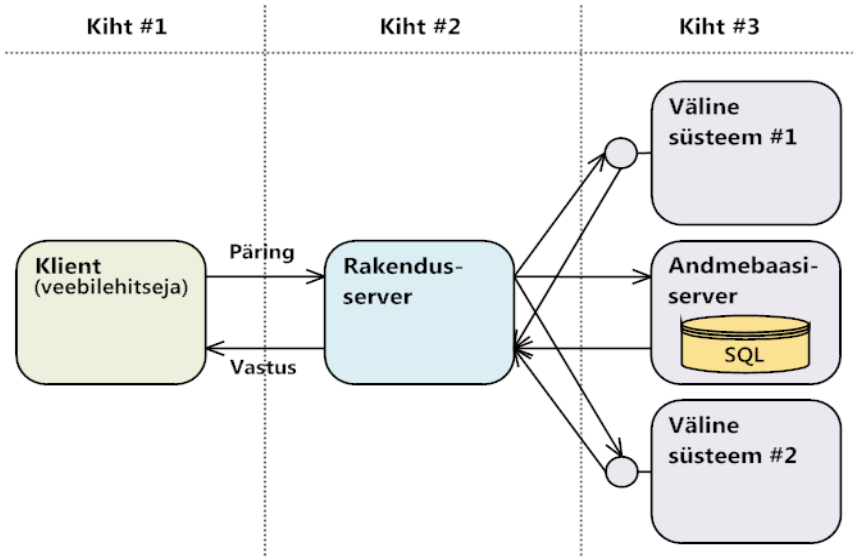
Olemitüübi nimi	Atribuudi nimi	Definitsioon	Näiteväärtus
request		soovi käesoleva maksenõude koostamiseks.	09:32:15
Visit	ip_address	Tähtnumbriline kood, mis määrab üheselt ära süsteemi sisenemiseks kasutatud lõppseadme (nt konkreetne lauaarvuti, mobiiltelefon) või seda hõlmava kohtvõrgu.	82.45.227.253
Visit	login_at	Kuupäev ja kellaaeg, mil tegutseja algatas süsteemis uue sisselogimiskatse.	2015-06-02 16:05:12
Reference_ entity	identifier	Klassifikaatori väärtusele viitav numbriline kood. Mõeldud vaid süsteemisiseseks kasutamiseks.	4
Reference_ entity	code	Klassifikaatori väärustele viitav tekstiline kood. Mõeldud vaid süsteemisiseseks kasutamiseks.	COMPLETED
Reference_ entity	name	„Klassifikaatori väärtuse ametlik nimetus.“ (Eessaar 2012, 51)	Completed

3. Loodava infosüsteemi realisatsioon

Käesolevas peatükis antakse lühikene ülevaade loodava infosüsteemi tehnilisest teostusest. Kõigepealt esitatakse vaadeldava rakenduse süsteemiarhitektuur ning põhjendatakse selle tarkvaralisi ja tehnoloogilisi valikuid. Seejärel disainitakse eespool analüüsitud andmebaas ning kirjeldatakse lähenemisi, mida rakendus kasutab andmete kvaliteedi parandamiseks ning pakktöötluslike toimingute optimeerimiseks. Viimaks antakse ülevaade mõningatest tüüpilistest puudustest ja probleemidest, mida võib kohata krüptovaluuta rahakotitarkvara arendamisel. Kuna tegu on krüptoraha valdkonda laiemalt puudutavate probleemidega, on alltoodud lahendusi võimalik kohandada ka mitmetele teistele sarnastele süsteemidele.

3.1 Süsteemiarhitektuur

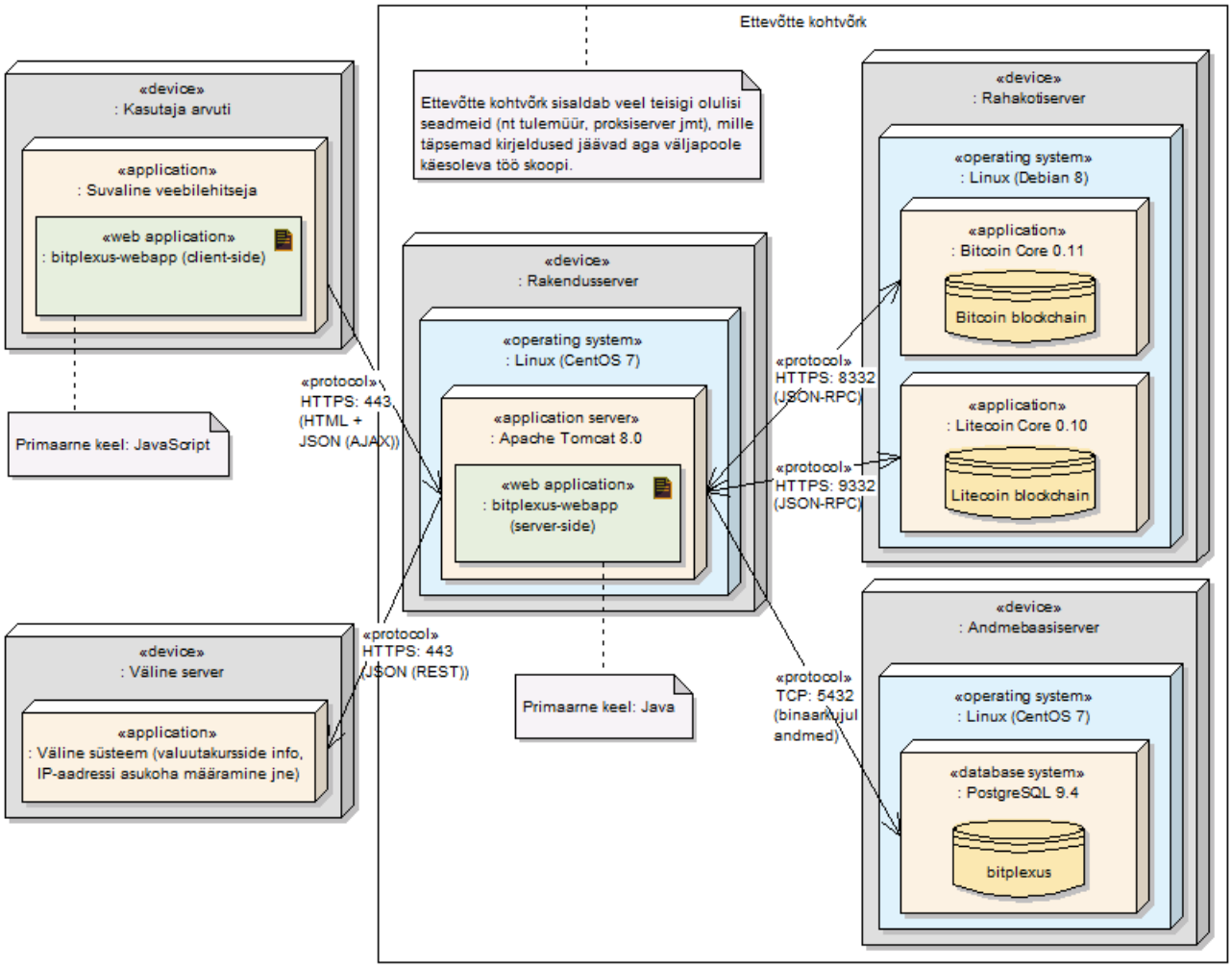
Käesoleva süsteemi arhitektuuri kavandamisel on lähtud nn n-kihilise arhitektuuri (ingl k *n-tier architecture*) mudelist, mille eesmärgiks on tagada loodava infosüsteemi maksimaalne modulaarsus ning paindlikkus. Mudeli kasutamine seisneb selles, et süsteemi funktsionaalsed komponendid (kasutajaliides, äriloogika, andmepöördus jmt) jagatakse mitmeks eraldiseisvaks protsessiks, mida on seejärel võimalik paigutada ühele või enamale füüsilisele serverarvutile (Tarhini 2011). Antud töös kasutatakse n-kihilise arhitektuuri klassikalist kolmekihilist varianti, kusjuures andmekihis on lisaks andmebaasisüsteemile kasutusel veel ka väliste süsteemide poolt pakutavad veebiteenused (vt joonis 6).



Joonis 6. Täiendatud kolmekihiline süsteemiarhitektuur

3.1.1 Tehnilise lahenduse ülevaade

Lähtudes eelmises jaotises kirjeldatud n-kihilise arhitektuuri mudelist on koostatud loodava infosüsteemi tehnilist arhitektuuri visualiseeriv UML-i evitusdiagramm (vt joonis 7). Kuna kõigi süsteemi komponentide sisu ei pruugi olla iseenesestmõistetav, on ära toodud ka iga evitusdiagrammil kujutatud elemendi eesmärki ning dünaamikat iseloomustav lühikirjeldus.



Joonis 7. Loodava terviklahenduse süsteemiarhitektuur

Nagu jooniselt 7 näha, on infosüsteemi esimese toodanguversiooni evitamiseks vaja vähemalt kolme eraldiseisvat töökeskkonda. Esialgu võib selleks kasutada virtualiseerimist ja/või erinevaid PaaS (ingl k *Platform as a Service*) lahendusi (nt Heroku, Jelastic, OpenShift jne). Kasutajate arvu märkimisväärse kasvu korral tuleks aga kaaluda privaatse serveripargi rajamist, et vähendada sõltuvust kolmandatest osapooltest (teenusepakkujad) ning maandada rahakottide *off-site* majutamisega kaasnevat riski.

Tuleks veelkord rõhutada, et joonisel 7 kujutatud süsteemiarhitektuuri näol on tegu loodava terviklahenduse loogilise vaatega, mis tähendab, et ükski selle komponentidest ei ole piiratud vaid ühele füüsilisele serverarvutile. Järgnevalt kirjeldataksegi iga evitusdiagrammil kujutatud komponendi täpsemaid tööpõhimõtteid ning selle peamisi ülesandeid vaadeldavas infosüsteemis.

- Rakendusserver – infosüsteemi keskne komponent, mille ülesandeks on vastata kasutaja poolt saadetavatele HTTP päringutele ning uuendada andmebaasiserveris hoitavaid andmeid. Suhtleb rahakotiserveri vahendusel krüptovaluuta võrgustikega ning kontrollib kasutajate sisestusi. Vastutab selle eest, et andmebaasiserveris oleks igal ajahetkel kõige uuem versioon plokiahela andmetest (s.o tehinguid ja aadresse puudutav info).
- Andmebaasiserver – infosüsteemi komponent, mille ülesandeks on teenindada rakendusserverit ning pidada arvet rakenduse tööks vajalike andmete üle. Lisaks harilikule teabele (kasutajakontod, rollid jmt) hoitakse siin ka informatsiooni krüptovaluuta aadresside kuuluvuse kohta. Loobutud on ülitundlike andmete (nt aadresside salajased võtmed) salvestamisest, et tagada klientide varade puutumatus ka võimalike andmelekete korral.
- Rahakotiserver – infosüsteemi komponent, mis majutab erinevate krüptovaluutade täielikke võrgusõlmi (ingl k *full node*) ning klientide varasid sisaldavaid rahakotifaile (*wallet.dat* failid). Antud komponendi turvalisuse tagamine on ettevõtte jaoks missioonikriitilise tähtsusega, mistõttu tuleks veenduda, et seda käitavad seadmed on välisvõrgust täielikult isoleeritud (v.a kontrollitud ühendused teiste krüptovaluuta võrgusõlmedega). Rakendusserveri poolt saadetavad JSON-RPC päringud tuleks samuti kaitsta transpordikihi turbeprotokolliga (SSL/TLS), et ennetada võimalikke vahendusründeid (NFR-010, vt ptk 2.5).
- Välised serverid – vaadeldav infosüsteem kasutab oma tööks ka mitmeid väliste süsteemide poolt pakutavaid teenuseid, kusjuures eelistatud suhtlusprotokolliks on HTTPS ning andmevahetusvorminguks JSON (ingl k *JavaScript Object Notation*). Hetkel on kasutusel kaks peamist rakendusliidest – *FreeGeoIP API* IP-aadresside asukoha määramiseks ning *Cryptonator API* krüptovaluuta kursside jälgimiseks.
- Kasutajate arvutid – kõigi süsteemi ärikasutajate (klient, veebihaldur, klienditoe spetsialist) töökohad on kättesaadavad läbi ühe universaalse veebiliidese. See tähendab, et infosüsteemi sisenemiseks ei ole vaja muud, kui suvalist nutiseadet

(arvuti, mobiiltelefon) koos mõne levinuma veebibrauseriga (nt Mozilla Firefox, Google Chrome). Lõppseadme enda valik ei ole oluline, kuna rakendus on ehitatud adaptiivse (ingl k *responsive*) veebidisaini põhimõtteid silmas pidades. Optimaalse kasutuskogemuse huvides tuleks aga veenduda, et kasutatavas brauseris on lubatud JavaScripti skriptide käivitamine.

3.1.2 Tarkvaraplatvormi valikute põhjendamine

Nagu juba eespool mainitud, peaks vaadeldava infosüsteemi serverikomponent olema üles ehitatud Java programmeerimiskeelt kasutades. Sellest eeltingimusest lähtuvalt valitaksegi järgnevas välja konkreetsed tehnoloogiad ja vahendid (sh rakendusserverid, andmebaasisüsteemid, tarkvarateegid jmt), mis on vajalikud käesoleva rakenduse esialgse toodanguversiooni väljatöötamiseks ning evitamiseks.

Operatsioonisüsteemid

Süsteemi füüsiline infrastruktuur võiks tugineda mõnele levinumale Linuxi distributsioonile. Selleks otstarbeks sobib ideaalselt tasulisel RHEL-il (Red Hat Enterprise Linux) põhinev CentOS 7, mille tugevateks külgedeks on tema töökindlus ning aktiivne kasutajaskond. Lisaks CentOSile kasutatakse veel ka laialt levinud Debian 8 distributsiooni, mis on installeeritud kõigile süsteemi rahakotiserverit käitavatele serverarvutitele. Eelneva abil on tagatud tarkvarakeskkonna mõningane heterogeensus, mis aitab vähendada distributsioonispetsiifiliste turvaaukude mõju süsteemile ning pärsib õnnestunud murrete süvenemist.

Rakendusserverid ja andmebaasisüsteemid

Vaadeldava rakenduse serverikomponent põhineb Java EE platvormil, kuid ei kasuta oma tööks ühtegi selle eksootilisematest võimalustest (EJB, JMS jmt). Seetõttu on rakenduseserveri tarkvarana kasutusele võetud Apache Tomcat 8, mis täidab kõik süsteemi funktsionaalsed eeldused (Servlet API, JSP, EL tugi), kuid on tunduvalt lihtsakoelisem oma alternatiividest (WildFly, WebSphere jt) (NFR-014, vt ptk 2.5).

Rakenduse primaarse andmebaasisüsteemina plaanitakse kasutada PostgreSQLi, mis on kõigile vabalt kättesaadav ning pakub äärmiselt mitmekülgset funktsionaalsust (välisvõtme

kitsendused, hetktõmmised, trigerid, talletatud rutiinid jmt). Viimase kasuks räägib ka selle põhjalik dokumentatsioon ning asjaolu, et kasutajate poolt leitud vead parandatakse väga lühikese aja jooksul. Lisaks eelnevale on PostgreSQLis võimalik kasutada erinevaid tiražeerimise lahendusi (pgpool, slony jt), mis on vajalikud käesoleva süsteemi jooksva varundamise nõude (NFR-008, vt ptk 2.5) rahuldamiseks.

Tarkvarateegid

Käesoleva süsteemi serverikomponent on üles ehitatud Springi rakendusraamistikule, mis lihtsustab oluliselt Java veebirakenduste arendamist ning omab äärmiselt põhjalikku dokumentatsiooni. Eelnimetatud rakendusraamistik koosneb suurest hulgast erinevatest moodulitest ja alamprojektidest, millest on käesolevas töös kasutusel näiteks Spring Beans/Context/Core (raamistiku tuumikkomponendid), Spring Data JPA (andmepöörduse lihtsustamine), Spring Transactions (äriteenuste atomaarsuse tagamine), Spring Web (väliste süsteemidega suhtlemine ning saabuvate HTTP päringute suunamine) jne.

Süsteemi andmepöörduseks kasutatava JPA (Java Persistence API) spetsifikatsiooni alusraamistikuna on rakendatud ORMi teeki nimega Hibernate, mis on tuntud oma töökindluse ning paindlikkuse poolest. Lisaks sellele on süsteemi andmekihis kasutusel veel ka PostgreSQLi ametlik JDBC draiver ning andmebaasiühenduste korduvkasutuse (ingl k *connection pooling*) teek nimega BoneCP. Viimase tugevateks külgedeks on tema suur töökiirus (oluliselt kiirem kui klassikalised lahendused nagu DBCP ning c3p0) ning suhteline lihtsus.

Selleks, et lihtsustada äriloogika kihis läbiviidavat domeeni- ja andmevahetusobjektide (ingl k *data transfer object*) vahelist translatsiooni, otsustati kasutada abiteeki nimega GeDA. GeDA eeliseks on, et see on pea täielikult annotatsioonipõhine (st puudub vajadus täiendavate konfiguratsioonifailide järele) ning teistest alternatiividest (Dozer, Orika jt) oluliselt kiirem. Bitcoini ja Litecoini võrgustikega suhtlemiseks kasutatakse aga vaheteeke nimega btcd-cli4j ning ltcd-cli4j, mis on käesoleva töö autori enda kirjutatud. Viimased on rakendust leidnud ka teiste autorite krüptoraha teemalistes projektides (How to communicate ... 08.12.2015) ning nende peamisteks tugevusteks on töökindlus, turvalisus ja ajakohasus.

Süsteemi serverikomponendi eesosa (ingl k *front end*) põhineb Java EE servlet-tehnoloogial (koos Springi Web ja Web MVC abimoodulitega), kusjuures vaadete loomiseks on kasutatud Java EE JSP (JavaServer Pages) ning JSTL (JSP Standard Tag Library) teeke. Ehkki tänaseks on tegu juba veidi aegunud tehnoloogiatega, pakuvad need siiski terviklikku platvormi Java veebirakenduste arendamiseks. Vaadete disainimisel on aga kasutatud Bootstrapi veebiraamistiku, mis lihtsustab oluliselt adaptiivsete veebilahenduste arendamist ning annab kogu süsteemile ühtlase välimuse (NFR-006, vt ptk 2.5). Vaadete juurde käiv abiloogika (AJAXi päringud, vormide valideerimine jmt) on realiseeritud JavaScripti baasil, kusjuures mitmete tegevuste optimeerimiseks on kasutatud ka jQuery teeki.

Teistest teekidest on kasutusel veel näiteks SLF4J ja Log4j, mille abil jäädvustatakse kõik süsteemi töö käigus aset leidnud sündmused. SLF4J eeliseks on, et see seob rakenduse lahti konkreetsest logimisraamistikust (antud juhul Log4j), mistõttu on arendajal võimalik logimise realisatsioone hiljem muuta (nt võtta kasutusele Apache Commons Logging). Üleliigsest koodist vabanemiseks on aga kasutatud abiteeke nimega lombok ning Google Guava. Esimese puhul on tegu sisuliselt korduvkasutatavate meetodite (getterid, setterid jmt) koodigeneraatoriga, teise puhul aga standardseid Java abifunktsioone koondava tugipaketiga.

Võrgusõlme tarkvara

Võrgusõlme tarkvara valikul on lähtutud põhimõttest, mille kohaselt tuleks võimalusel eelistada iga krüptovaluuta nn „ametlikku“ tarkvaraplatvormi. Selliselt on tagatud protokolli viimaste uuendustega kaasas käimine ning tulevaste veaparanduste võimalikult kiire kättesaadavus. Lisaks eelnevale maandab see võrgusõlme plokiahelast lahknemise riski, st olukorda, kus võrgusõlm satub alternatiivses implementatsioonis tehtud tarkvaravea tõttu nn kõrvalisele plokiahelale (e tehingute andmebaasis tehakse muudatus, mida teised implementatsioonid ei pea korrektseks). Töö kirjutamise hetkel olid Bitcoini ja Litecoini *de facto* näidisimplementatsioonideks vastavalt Bitcoin Core 0.11 (~ 87% kõigist võrgusõlmedest) ning Litecoin Core 0.10 (~ 98% kõigist võrgusõlmedest). Valida oli aga ka mitmete konkureerivate võrgusõlme realisatsioonide vahel, nt Bitcoini puhul: Bitcoin XT, btcd, bitcoin-ruby jpt.

3.2 Andmebaasi loogiline ja füüsiline disain

Käesolevas alajaotises vaadeldakse lähemalt, kuidas kohandada lõputöö analüüsietapis väljatöötatud kontseptuaalset andmemudelit (vt ptk 2.7) konkreetsele andmebaasi tüübile (relatsiooniline andmemudel) ning eelmises peatükis valituks osutunud andmebaasisüsteemile (PostgreSQL 9.4).

3.2.1 Andmebaasi disaini põhimõtted

Käesoleva andmebaasi disain põhineb suuresti nn „paksu andmebaasi" (ingl k *thick database*) teoorial, mis tähendab, et rakenduse arendamisel on püütud maksimaalselt ära kasutada kõiki andmebaasisüsteemi poolt pakutavaid võimalusi (Dorsey 2013). Kuna antud käsitluse puhul realiseeritakse suur osa funktsionaalsusest otse andmebaasiserveris, väheneb oluliselt rakenduse erinevate komponentide vahel liikuva informatsiooni hulk. Selle tulemusena paraneb ka süsteemi kui terviku töökiirus, kuna varasemast suurem osa tööd on võimalik ära teha ilma täiendavaid vahepöördumisi tegemata.

Andmebaasi turvapoliitika kujundamisel on aga lähtutud minimaalse pääsu poliitikast (Hanson jt 1998, 37). See tähendab, et andmebaasisüsteemiga suhtlemiseks kasutatakse paari eeldefineeritud kasutajakontot, millest igaühele on omistatud kindla kasutajagrupi vajadusi rahuldavad privileegid. Kõik, mis ei ole antud kasutajatele lubatud, on vaikimisi keelatud (*Ibid.*, 37). Sama reegel laieneb ka teistele andmebaasisüsteemi kasutajatele, kellel puudub seega igasugune õigus vaadeldava infosüsteemi andmetele juurdepääsuks.

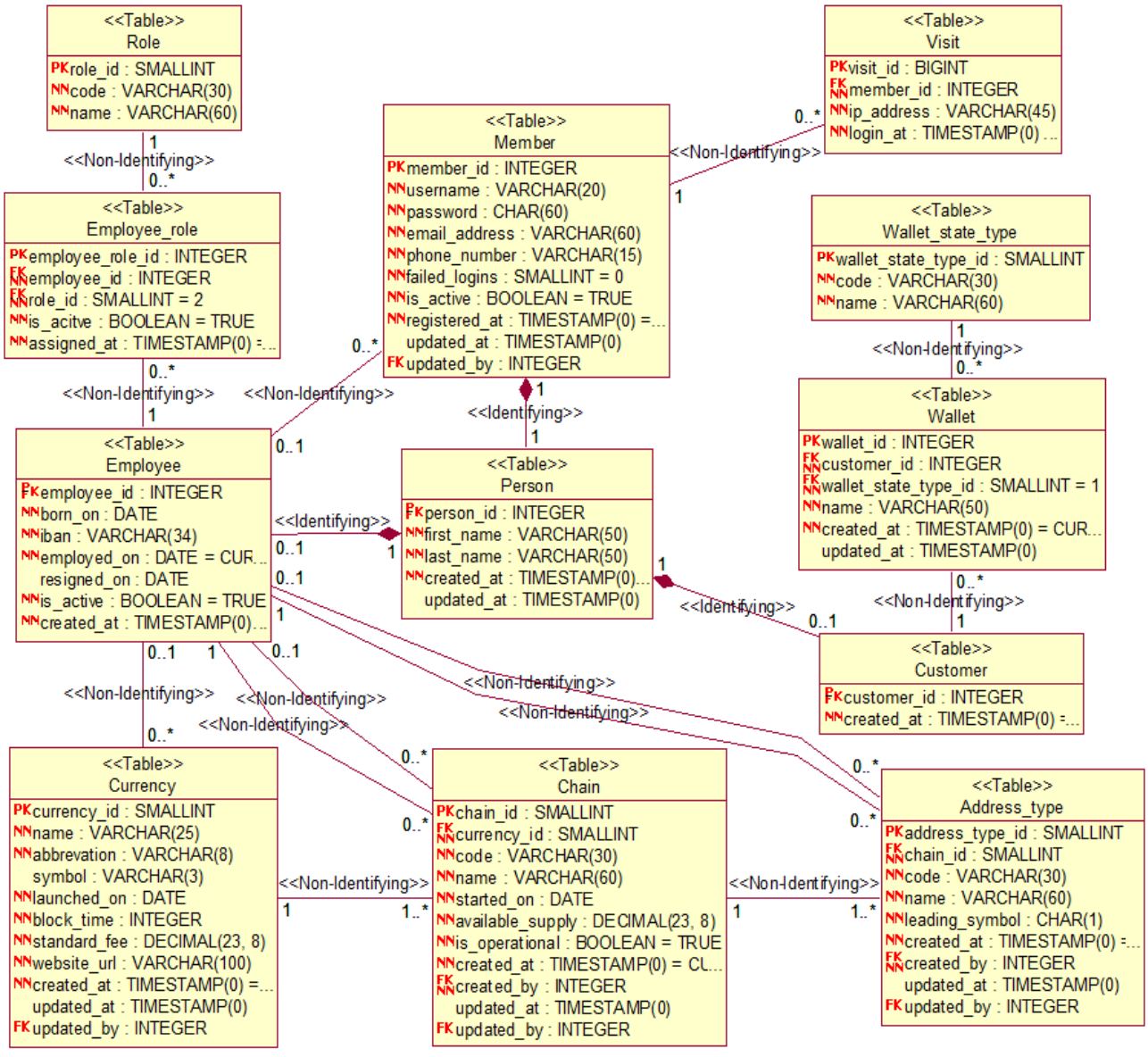
Järgnevalt on loetletud mõningaid teisi loogilise ja füüsilise disaini põhimõtteid, mida on käesoleva andmebaasi ehitamisel silmas peetud.

- Kõik andmetabelid on viidud vähemalt kolmandale normaalkujule (3NF, ingl k *third normal form*), et minimeerida andmete liiasust.
- Tabeliveergude andmetüübid on valitud selliselt, et nad oleksid võimalikult täpselt kooskõlas seal hoitavate väärtustega. Näiteks: veeru *failed_logins* esitamiseks sobib kõige paremini PostgreSQLi andmetüüp nimega SMALLINT, kuna ebaõnnestunud sisselogimiste arv ei ole iialgi suurem kui 3.
- Olemite terviklikkuse tagamiseks on loodud primaarvõtme kitsendused.
- Viidete terviklikkuse tagamiseks on loodud välisvõtme kitsendused, kusjuures:

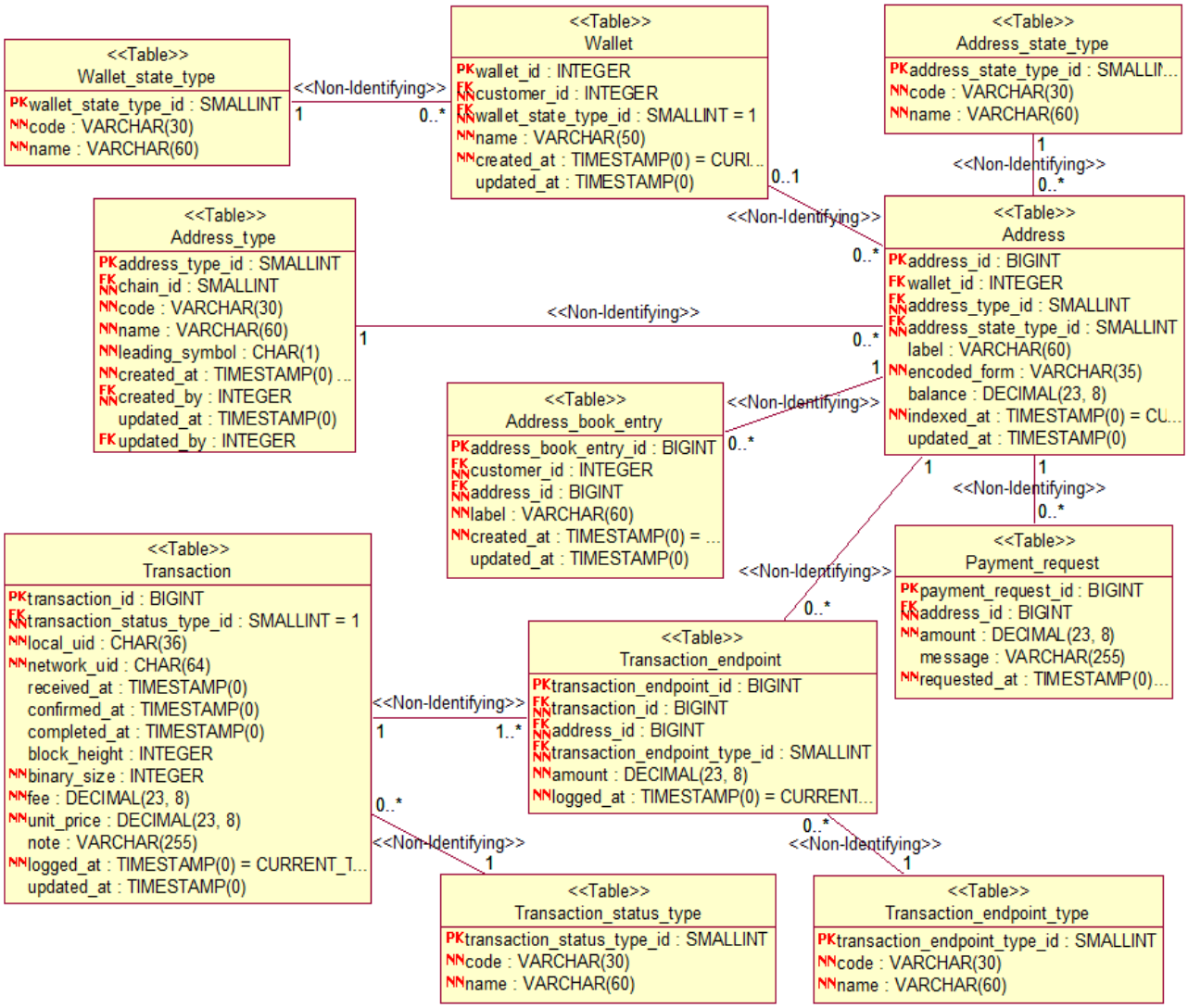
    - Kui tegu on sisulist tähendust omava võtmega, mille väärtusi hallatakse käsitsi, on kasutatud kompenseerivat tegevust ON UPDATE CASCADE.
    - Kui tegu on sõltuva olemiga, mis ei oma iseseisvat tähendust (nt kompositsiooni puhul), on kasutatud kompenseerivat tegevust ON DELETE CASCADE.
- Võimalikult suur osa veergudest on kaetud NOT NULL kitsendustega, et lihtsustada puuduvate andmete haldamist ning töötlemist süsteemis.
- Mitte-primaarvõtme veergudele, milles hoitakse unikaalseid väärtusi, on defineeritud unikaalsuse kitsendused.
- Ärireeglite täidetuse kontrollimiseks on koostatud CHECK kitsendused ning trigerid.
- Töömahukate andmemuudatuste tegemiseks on loodud spetsiaalsed talletatud rutiinid.
- Välisvõtme veergudele on lisatud B-puu indeksid, et kiirendada andmete otsimist sõltuvatest tabelitest.

3.2.2 Andmebaasi struktuur

Andmebaasi struktuuri kavandamisel on lähtutud jaotises 2.7 koostatud kontseptuaalsest andmemudelist ning jaotises 3.2.1 kirjeldatud andmebaasi disaini põhimõtetest. Alljärgnev diagramm (vt joonised 8 ja 9) on esitatud loogilise disaini täpsusega, et seda oleks võimalik kohandada ka teistele andmebaasisüsteemidele. Diagramm on jagatud kaheks osaks, et tagada sellel kujutatud elementide parem loetavus.



Joonis 8. Terviksüsteemi loogilise disaini andmebaasi diagramm (1/2)



Joonis 9. Terviksüsteemi loogilise disaini andmebaasi diagramm (2/2)

3.2.3 Andmete korrektsuse tagamine

Krüptovaluutade üheks peamiseks nõrkuseks on nende madal valulävi lõppkasutaja eksimuste suhtes. Eelnev väljendub näiteks asjaolus, et iga võrgustikule edastatud tehing on lõplik – st puudub mehhanism probleemsete maksete tühistamiseks (ingl k *chargeback*). Lisaks sellele võib aadressile vastava võtmepaari kaotsiminekut võrrelda sellel paiknevate vahendite hävimisega. Seetõttu on äärmiselt oluline, et andmebaasis hoitavad andmed oleksid kaitstud juhuslike vigade ning muude ärireegleid eiravate andmemuudatuste eest. Näiteks: tabelis *Address* paiknevad read peaksid lisaks andmebaasi diagrammil toodud kitsendustele (s.o andmetüübid, veergudele seatud mahupiirangud jmt) rahuldama veel ka järgmiseid tingimusi.

- Aadressi kodeeritud kuju (veerg *encoded_form*) peab olema vähemalt 26 tähemärki pikk.

- Aadressi kodeeritud kuju (veerg *encoded_form*) peab vastama Base58Check kodeeringu reeglitele, st see ei tohi sisaldada numbrit „0“ ega tähti „l“, „I“ ning „O“.
- Aadressi saldo (veerg *balance*) peab olema positiivne ratsionaalarv.
- Aadressi registreerimise kuupäev (veerg *indexed_at*) ning selle viimase muutmise kuupäev (veerg *updated_at*) peavad jääma ajavahemikku 01.01.1900 kuni 01.01.2100.
- Aadressi registreerimise kuupäev (veerg *indexed_at*) ei tohi olla hilisem selle viimase muutmise kuupäevast (veerg *updated_at*).

Taoliste kontrollide jõustamiseks andmebaasi tasemel on kasutatud erinevat tüüpi deklaratiivseid kitsendusi (s.o harilikke CHECK kitsendusi ning *CREATE RULE* lause abil loodavaid reegleid).

```
ALTER TABLE Address
ADD CONSTRAINT ck_address_encoded_form_length CHECK (length(encoded_form) > 25),
ADD CONSTRAINT ck_address_encoded_form_in_base58
   CHECK (encoded_form ~ '^[1-9a-km-zA-HJ-NP-Z]*$'),
ADD CONSTRAINT ck_address_balance_in_range CHECK (balance >= 0),
ADD CONSTRAINT ck_address_indexed_at_in_range
   CHECK (indexed_at BETWEEN '1900-01-01' AND '2100-01-01'),
ADD CONSTRAINT ck_address_updated_at_in_range
   CHECK (updated_at BETWEEN '1900-01-01' AND '2100-01-01'),
ADD CONSTRAINT ck_address_updated_at_chrono_order CHECK (indexed_at <= updated_at);
```

Kõigi ärireeglite jõustamiseks aga ainuüksi deklaratiivsetest kitsendustest ei piisa. Sellistel puhkudel on appi võetud andmebaasi trigereid ning trigeri protseduurid. Näiteks tuleb maksenõuete koostamisel kontrollida, kas kliendi poolt sisestatud sihtaadress kuulub ikka vaadeldava infosüsteemi käsutusse. See tähendab, et rahakotiserveris on olemas salajane võti antud aadressile saadetud vahendite lunastamiseks (teisisõnu, tabeli *Address* veerg *wallet_id* ei sisalda väärtust *NULL*). Vastasel juhul võib tekkida olukord, kus maksenõude saanud isik tasub nõutud summa nn rahakotivälisele aadressile, ehk aadressile, mille üle rakendusel tegelikult kontroll puudub. Taoliste probleemide ennetamiseks ongi koostatud järgnev andmebaasi triger ning trigeri protseduur.

```
CREATE OR REPLACE FUNCTION f_check_payment_request_address_id() RETURNS TRIGGER AS $$
DECLARE
   wallet_id INTEGER;
BEGIN
   SELECT a.wallet_id INTO STRICT wallet_id FROM address AS a
   WHERE a.address_id = NEW.address_id;
   IF (wallet_id IS NULL) THEN
      RAISE EXCEPTION '% failed (table ''%'') - the recipient (address_id = %) must
         be an ''internal'' (wallet) address, not an ''external'' (non-wallet)
         one.', TG_OP, TG_TABLE_NAME, NEW.address_id USING ERRCODE = '30210';
   END IF;
   RETURN NULL;
END
$$ LANGUAGE plpgsql;
```

```
CREATE CONSTRAINT TRIGGER tr_payment_request_address_id_check
AFTER INSERT OR UPDATE OF address_id ON payment_request
INITIALLY DEFERRED
FOR EACH ROW
EXECUTE PROCEDURE f_check_payment_request_address_id();
```

Kuna andmete kvaliteedi tagamine on käesoleva infosüsteemi jaoks missioonikriitilise tähtsusega, on sarnast lähenemist rakendatud ka kõigi teiste andmetabelite disainimisel. Kokku on vaadeldava infosüsteemi esialgses toodanguversioonis defineeritud 84 CHECK kitsendust ning 29 trigerit.

3.2.4 Andmekesksete operatsioonide realiseerimine

Käesolev infosüsteem kasutab oma tööks mitmeid väliseid andmeallikaid, millest pärinev informatsioon koondatakse rakenduse tsentraalsesse andmebaasi. Selliselt kogutud teabe ajakohasena hoidmiseks tuleb andmebaasis regulaarselt läbi viia mitmeid töömahukaid andmemuudatusi (nt aadresside saldode uuendamine, tehingute seisundite muutmine jne). On selge, et taolisi toiminguid ei ole mõtet teostada läbi rakenduse püsivuskihi (ingl k *persistence layer*), kuna see suurendaks oluliselt süsteemi mälukasutust ning komponentidevahelist liiklust. Seetõttu on valdav osa süsteemi töömahukatest operatsioonidest realiseeritud andmebaasiserveris talletatud rutiinidena (NFR-002, vt ptk 2.5).

Vaatleme näitena krüptovaluuta tehingute lõpetatuks lugemise operatsiooni, mida käivitatakse vahetult pärast iga uue ploki kaevandamist. Antud protseduuri eesmärgiks on kontrollida, millised viimastest tehingutest on saanud võrgustikult piisava arvu kinnitusi, et lubada neist laekunud vahendite kulutamine. Ehkki teoreetiliselt peaks tehing olema lõplikult fikseeritud juba pärast esimest kinnitust, on maksimaalse ohutuse huvides (ründed, protokollivead)

soovitatav oodata, kuni sellele lisandub veel paar kinnitust (Bitcoini võrgustiku puhul piisab üldjuhul kolmest kinnitusest, Litecoini võrgustiku puhul aga kuni 12-st kinnitusest). Eelkirjeldatud tingimuste kontrollimiseks ongi koostatud alljärgnev talletatud rutiin, mis vaatab läbi kogu tehingute tabeli (tabel *Transaction*) ning märgib lõpetatuks kõik need tehingud, mis on saanud võrgustikult nõutud arvu kinnitusi (parameeter *in_confirmation_count*).

```
CREATE OR REPLACE FUNCTION f_complete_transactions(in_confirmation_count SMALLINT,
in_block_height INTEGER, in_block_time TIMESTAMP(0), in_chain_code VARCHAR(30))
RETURNS CHAR(64)[] AS $$
WITH completed_transactions AS (
   UPDATE transaction AS t SET transaction_status_type_id = 4,
      completed_at = in_block_time
   FROM transaction_endpoint AS te, address AS a, address_type AS at, chain AS ch
   WHERE t.block_height <= (in_block_height - (in_confirmation_count - 1))
      AND t.transaction_status_type_id = 3 AND t.transaction_id = te.transaction_id
      AND te.address_id = a.address_id AND a.address_type_id = at.address_type_id
      AND at.chain_id = ch.chain_id AND ch.code = in_chain_code
   RETURNING t.network_uid
)
SELECT array_agg(network_uid) FROM completed_transactions;
$$ LANGUAGE sql STRICT;
```

Analoogilisel viisil on lahendatud ka kõigi teiste töömahukate operatsioonide täitmine, st vajadusel on mindud mööda rakenduse ORMi-põhisest domeenimudelist ning pöördutud otse andmebaasiserveris talletatud rutiinide poole. Kokku on vaadeldava infosüsteemi esialgses toodanguversioonis defineeritud 26 talletatud rutiini.

3.2.5 Kasutajagrupid ja nende õigused

Selleks, et viia ellu jaotises 3.2.1 kirjeldatud minimaalse pääsu poliitika, on andmebaasis loodud viis erineva sisuga kasutajagruppi (vt Tabel 3). Rakenduse serverikomponent kasutab andmebaasiga suhtlemiseks eelkõige kliendi rolli (*bitplexus_customer*) ning töötaja rolli (*bitplexus_employee*), kusjuures süsteemisiseselt on kasutusel veel ka autonoomseid taustaprotsesse teenindav abiroll (*bitplexus_drone*). Andmebaasi administreerimiseks mõeldud tugirolle (*bitplexus_dbm* ja *bitplexus_dba*) on aga võimalik kasutada vaid läbi selleks ettenähtud administreerimise tarkvara. Viimane aitab leevendada potentsiaalsete murrete mõju andmebaasile ning garanteerib, et eduka ründaja privileegid piirduvad vaid paari andmetabeli lugemis- ja kirjutamisõigusega.

Tabel 3. Andmebaasis defineeritud kasutajagrupid

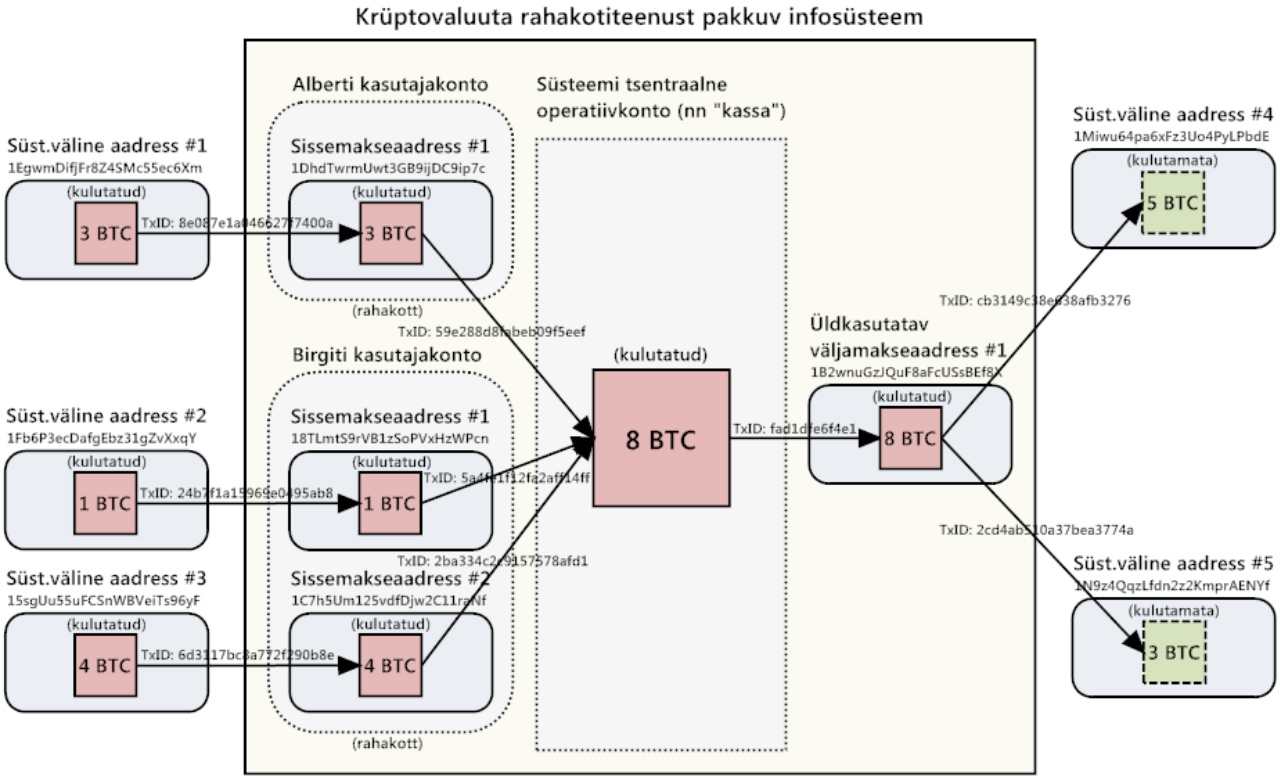
Kasutajagrupp	Õiguste kirjeldus
Klient (*bitplexus_customer*)	Võimaldab tegutsejal vaadata ning muuta enda kasutajakontoga seonduvaid andmeid. Annab õiguse luua uusi rahakotte, aadresse, aadressiraamatu kandeid ja maksenõudeid ning hallata nende elutsükleid. Lubab registreerida uusi krüptovaluuta tehinguid ning näha varasemalt teostatud tehingute andmeid.
Töötaja (*bitplexus_employee*)	Võimaldab tegutsejal vaadata ning muuta probleemsete tehingute andmeid (kordussaatmine, arhiveerimine jne). Annab õiguse registreerida uusi plokiahelaid ja aadressi liike ning värskendada olemasolevate krüptovaluutade andmeid. Lubab vaadata kõigi süsteemi tegutsejate isikuandmeid ning vabastada nende kontosid blokeeringutest.
Andmebaasihaldur (*bitplexus_dbm*)	Võimaldab vaadata, lisada, muuta ning kustutada kõiki süsteemi poolt hallatavaid andmeid. Ei oma superkasutaja õigusi ega omanikuõigusi ühegi andmebaasiobjekti suhtes. Mõeldud andmebaasis hoitavate andmete kvaliteedi parandamiseks ning muude kasutajaliideseväliste operatsioonide teostamiseks (nt klassifikaatori väärtuste haldamine).
Andmebaasi administraator (*bitplexus_dba*)	Annab tegutsejale õiguse lisada, muuta ning kustutada kõiki andmebaasiserveris hoitavaid andmebaasiobjekte. Mõeldud vaid superkasutaja privileege nõudvate struktuurimuudatuste teostamiseks (nt indeksite lisamine, vaadete loomine, skeemi evolutsioneerimine jne).
Infosüsteemi taustaprotsess (*bitplexus_drone*)	Võimaldab vaadata ning muuta viimati esitatud tehingute ja nendes viidatud aadresside andmeid. Koosneb üksikutest spetsiifilistest privileegidest, mis on vajalikud välistes süsteemides aset leidnud sündmuste ülekandmiseks käesolevasse süsteemi (nt uue ploki kaevandamine).

3.3 Tüüpilised probleemid ja nendega toimetulemine

Järgnevalt on kirjeldatud mõningaid tüüpilisi puuduseid ja probleeme, mida võib kohata krüptovaluuta rahakotitarkvara arendamisel. Lisaks probleemide detailsetele kirjeldustele on välja toodud ka nende võimalikud lahendused, mis on vastavuses krüptoraha valdkonna parimate praktikatega. Probleemide analüüsimisel on järgitud nn ülalt-alla lähenemist. See tähendab, et kõigepealt on vaadeldud rahakotihalduse pilve viimisega kaasnevaid üldiseid ohtusid ning alles seejärel on pöördutud detailsemate tehniliste probleemide kirjeldamisele.

3.3.1 Teenuse läbipaistvus

Üheks peamiseks ajendiks, mis viis krüptovaluutade loomiseni, oli inimeste massiline pettumine traditsioonilises pangandussüsteemis ning selle kesksetes tööpõhimõtetes (ingl k *fractional-reserve banking*). Seetõttu on krüptovaluutade arendamisel eriti suurt tähelepanu pööratud just arvepidamise läbipaistvusele ning finantsandmete auditeeritavusele. Paraku leidub ka krüptoraha maailmas selliseid ettevõtteid (nt Coinbase, Circle), kes ei ole huvitatud avaliku tehinguandmete žurnaali (e plokiahela) kasutamisest. Viimaste poolt arendatavad teenused meenutavad sageli klassikalisi internetipanku, kus reaalne väärtusvahetus toimub tegelikult ettevõtte enda andmebaasis (ingl k *off-chain transactions*). Krüptovaluuta võrgustikega ühendutakse seejuures vaid juhul, kui raha on vaja saata süsteemi piiridest väljapoole (vt joonis 10). Selleks kasutatakse sageli nn kliiringulaadset lähenemist, kus väljamaksed sooritatakse üksikute suuremahuliste (nt 95 kB) kannetena. Iga taoline kanne võib korraga debiteerida sadu või isegi tuhandeid väliseid krüptovaluuta aadresse.

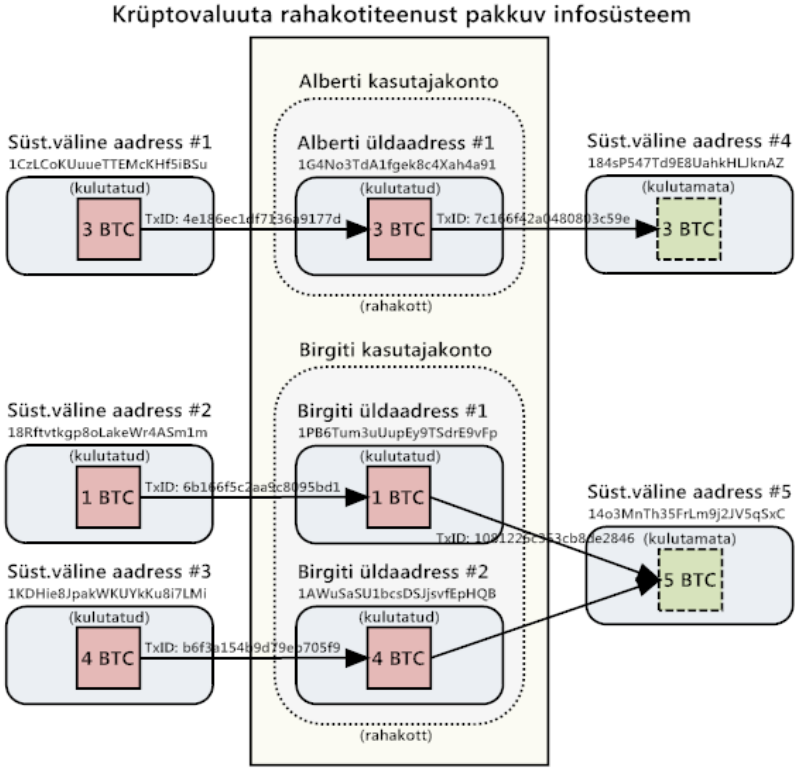


Joonis 10. Plokiahelavälist arvepidamist kasutav süsteem (halb lahendus)

Lisaks halvale läbipaistvusele on plokiahelaväline arvepidamine (ingl k *internal ledger*) problemaatiline ka süsteemi turvalisuse seisukohalt. Nagu jooniselt 10 näha, kantakse kõik süsteemi sissemakseaadressidele jõudnud rahalised vahendid sealt edasi rakenduse ühisele

operatiivkontole (ingl k *hot wallet*). Eelnimetatud sammu käigus kaotab süsteem ülevaate sellest, milline konkreetne väljund kuulus millisele kliendile. Ühtlasi tähendab see ka seda, et väljamaksete teostamisel lähtutakse edaspidi rakenduse kohalikus andmebaasis hoitavatest kontojääkidest. Seega piisab potentsiaalsel ründajal vaid paari andmebaasikirje muutmisest, et varastada ära kogu rakenduse operatiivkontol paiknev raha.

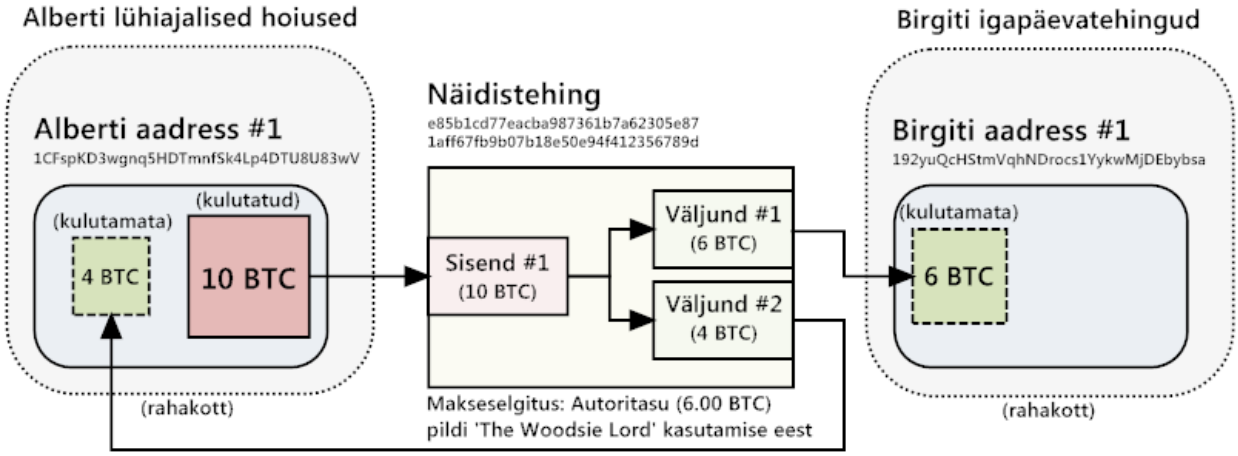
Selleks, et vältida eelkirjeldatud probleemide tekkimist, on käesolevas töös rakendatud nn plokiahelapõhist arvepidamist (vt joonis 11). See tähendab, et süsteem ei dubleeri sisemiselt krüptovaluuta avaliku žurnaali funktsionaalsust, vaid pigem täiendab seda kontode loomise ning aadresside haldamise loogikaga. Sellest tulenevalt puudub ka igasugune vahendite puulimise süsteem (nn ühine operatiivkonto), mis tegeleks erinevatelt sissemakseaadressidelt pärinevate krüptovaluuta väljundite agregeerimisega. Iga rakenduse poolt teostatav tehing eeldab kliendi eelnevat nõusolekut, kusjuures missioonikriitiliste tingimuste (nt kontojääk) kontrollimiseks pöördutakse otse vastava valuuta plokiahela poole. Makse edukal sooritamisel väljastab süsteem kliendile tehingu unikaalse identifikaatori (nn TxID, vt ptk 2.7, tabel 2), mida on võimalik kasutada vastava ülekande verifitseerimiseks. Selline lähenemine tagab süsteemi maksimaalse läbipaistvuse ning annab kõigile osapooltele võimaluse veenduda ettevõtte maksejõulisuses ning enda isiklike varade puutumatuses.



Joonis 11. Plokiahelapõhist arvepidamist kasutav süsteem (parem lahendus)

3.3.2 Aadresside korduvkasutus

Nagu juba eespool mainitud, salvestatakse kõik krüptovaluuta võrgustikule edastatud tehingud selle valuuta plokiahelasse (e avalikku tehingute andmebaasi). Ühelt poolt pakub selline lähenemine traditsioonilise pangandussüsteemiga võrreldes mitmeid olulisi eeliseid nagu kannete läbipaistvus, tsenseerimatus, jälitatavus jne. Teiselt poolt tähendab see aga seda, et avalikku žurnaali jõuavad ka kasutajate kõige delikaatsemad ning valgustkartvamad tehingud, mis on sealt vabalt kättesaadavad ka 100 aasta pärast. Seetõttu tuleks krüptovaluuta rahakotitarkvara arendamisel väga suurt tähelepanu pöörata just kasutajate privaatsuse säilitamisele. Kõige olulisemat rolli mängib seejuures krüptovaluuta aadresside haldamine, kuna viimased on ainsaks lüliks kasutaja, talle kuuluvate varade ning nendega teostatud tehingute vahel (Avoiding Key Reuse ... 21.11.2015). Seega kui inimene kasutab maksete sooritamiseks pidevalt ühte ja sama krüptovaluuta aadressi, on tema tehingute ajaloo põhjal võimalik teha üsna mitmesuguseid järeldusi (sh tema majandusliku seisundi, tarbimisharjumuste ning peamiste tehingupartnertite kohta) (vt joonis 12).

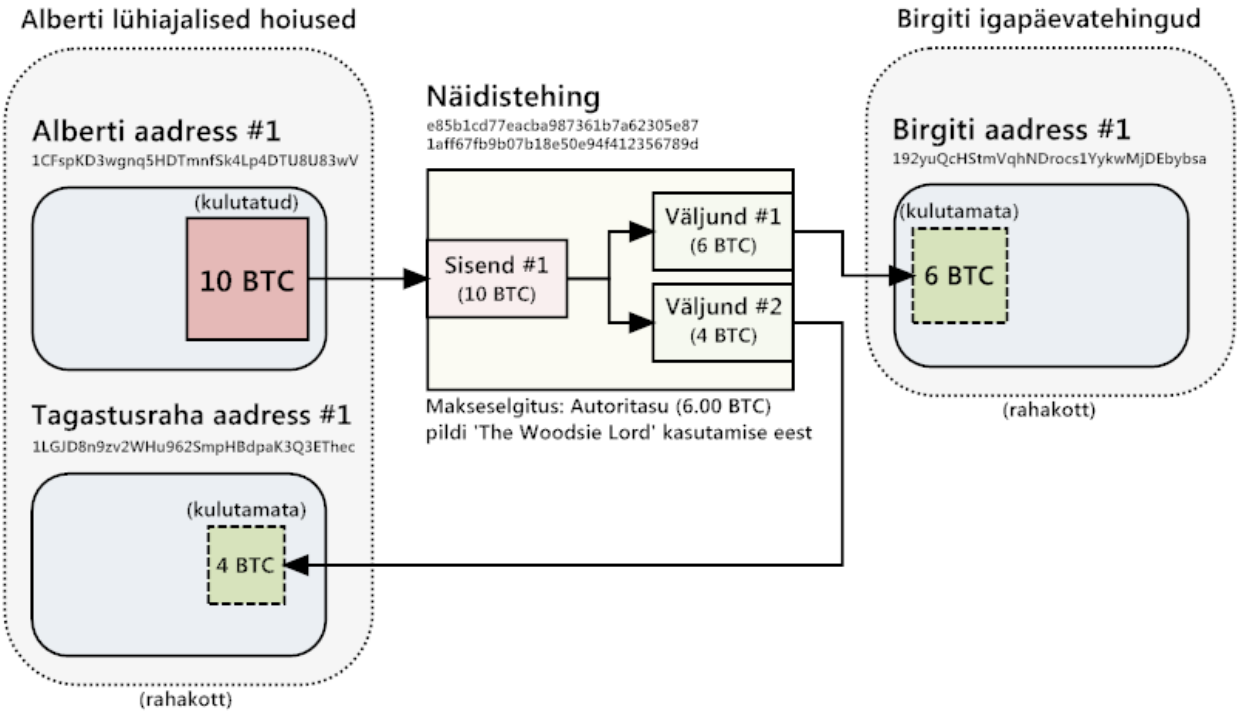


Joonis 12. Näidistehing aadresside korduvkasutuse puhul (halb lahendus)

Ehkki visuaalselt tundub joonisel 12 kujutatud lähenemine igati mõistlik, kahjustab see oluliselt aadressi omaniku (Alberti) privaatsust. Tuleb silmas pidada, et kõigil kasutaja aadressi teadvatel isikutel (sh lähedased, tuttavad, töökaaslased jne) on vaba ligipääs nii aadressi bilansile kui ka sellelt varem sooritatud tehingute andmetele. Taolise informatsiooni avalikuks tulemine võib aga kasutajale kaasa tuua mitmeid ebameeldivusi (nt häbi, vargused, suhete purunemine jne). Seetõttu on äärmiselt oluline, et kasutajad väldiksid igal võimalikul juhul staatiliste krüptovaluuta aadresside kasutamist. Ideaalis peaks iga aadress plokiahelas

kajastuma vaid kahel korral: esimest korda makse vastuvõtmisel ning teist korda sellest tulenevate vahendite kulutamisel (*Ibid.*). Niiviisi tagatakse finantstehingu steriilsus üleliigsest taustinformatsioonist, sh viidetest konkreetsetele isikutele, kasutusotstarvetele ning muudele süsteemivälistele olemitele.

Tõhusaim viis aadresside korduvkasutuse vältimiseks on ehitada rahakotitarkvara üles selliselt, et see soosiks kasutajate ohutut käitumist. Näiteks suunab käesolev rakendus kõik edukatest tehingutest ülejäänud rahasummad (ehk nn peenraha) selleks spetsiaalselt loodud tagastusraha aadressidele (vt joonis 13). Ka kasutajaliidese kujundamisel on erilist tähelepanu pööratud just uute aadresside genereerimise lihtsustamisele. Näiteks on maksete vastuvõtmisel oluliselt mugavam luua uusi aadresse, kui pöörduda selleks vanade/juba olemasolevate aadresside poole. Lisaks sellele kuvatakse aadresside põhivaates vaid selliseid aadresse, millel paiknevaid rahalisi vahendeid ei ole veel ära kulutatud. Ajaloolisi aadresse on aga võimalik näha vaid läbi selleks ettenähtud vaadete, kusjuures viimaste kasutamine on rangelt mittesoovitatav. Teatud juhtudel võib see aga osutuda möödapääsmatuks: näiteks kui kaupmees teeb kliendile praaktoote eest tagasimakse (ingl k *refund*), kantakse vaidlusalune summa reeglina tagasi selle lähteaadressile.



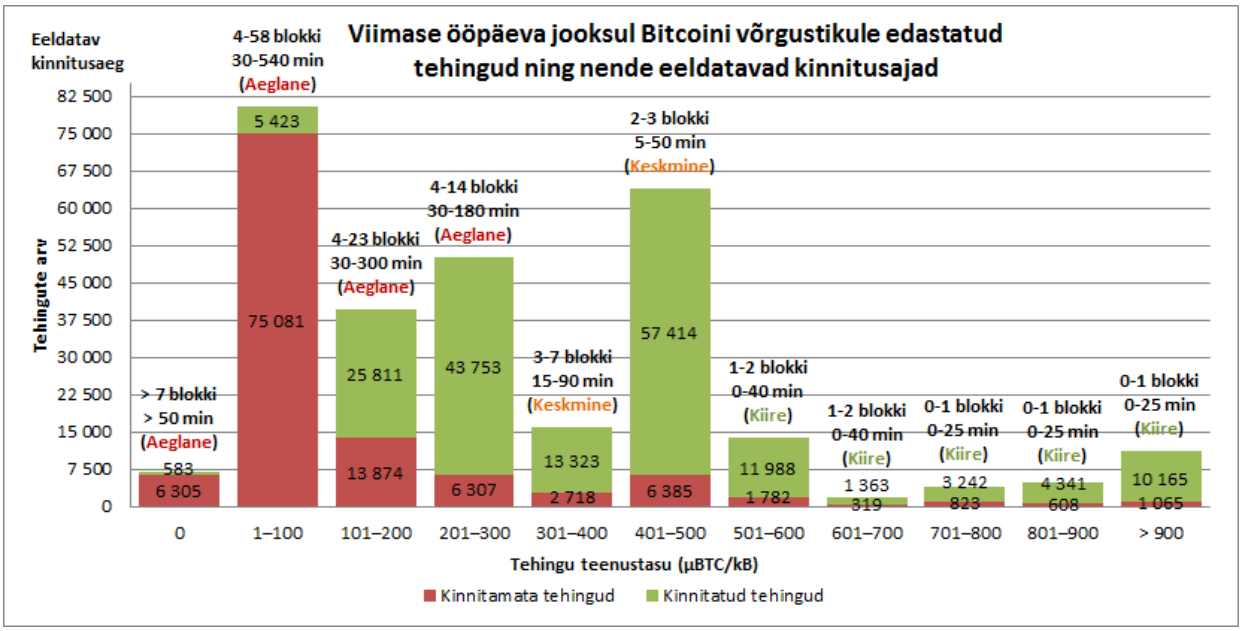
Joonis 13. Näidistehing aadresside korduvkasutuse vältimisel (parem lahendus)

Järgnevalt on toodud lühikene koodinäide rakenduse serverikomponendi tehingumoodulist. Koodinäide illustreerib, kuidas aadressi jääksaldo (muutuja *excessAmount*) kantakse tehingu sooritamisel automaatselt üle uuele krüptovaluuta aadressile (muutuja *changeAddress*).

```
private Map<String, BigDecimal> buildRecipientList(List<Output> selectedOutputs,
      PaymentDetailsDto paymentDetails, BigDecimal fee, Integer walletId,
      String chainCode) {
   Map<String, BigDecimal> recipients = new HashMap<String, BigDecimal>();
   ...
   BigDecimal excessAmount = getOutputSum(selectedOutputs).subtract(paymentDetails
      .getAmount().add(fee));
   if (excessAmount.compareTo(BigDecimal.ZERO) > 0) {
      AddressDto changeAddress = addressService.createNewWalletAddress(walletId,
         chainCode);
      recipients.put(changeAddress.getEncodedForm(), excessAmount);
   }
   ...
   return recipients;
}
```

3.3.3 Tehingu teenustasu optimeerimine

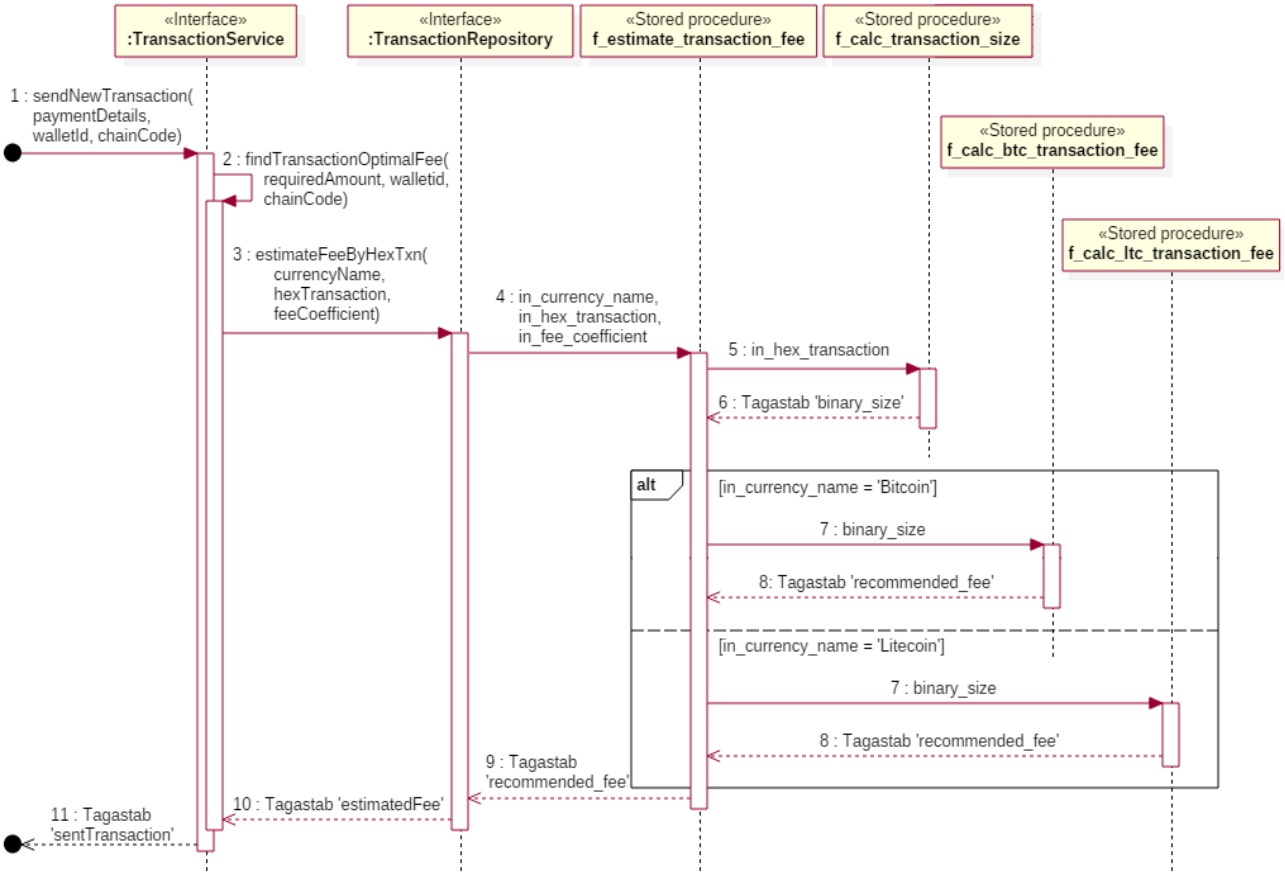
Üheks levinumaks väärarusaamaks, mida krüptovaluutade puhul sageli kohata võib, on arvamus, et raha saatmine punktist A punkti B on kasutaja jaoks täiesti tasuta. Ehkki teatud juhtudel võidakse võrgustikule edastatud tehinguid tõepoolest töödelda teenustasust sõltumata (ingl k *high-priority transactions*), eeldab valdab osa neist siiski teatava lõivu tasumist. Konkreetse tehingu teenustasu suurus sõltub mitmest erinevast tegurist nagu selle rahaline väärtus, tehinguandmete maht (kilobaitides) jne. Teenustasu peamiseks eesmärgiks on motiveerida kaevureid, ehk spetsiaalseid võrgusõlmi, kellel on vaadeldava plokiahela suhtes kirjutamisõigus. Kuna kaevurid püüavad plokke täita enda jaoks võimalikult kasumlikul viisil, tähendab suurem teenustasu reeglina ka seda, et tehing lisatakse plokiahelasse eelisjärjekorras. Joonisel 14 ongi toodud näide kõigist Bitcoini võrgustikule edastatud tehingutest ning nende teenustasudest ühe juhuslikult valitud ööpäeva lõikes (2015. aasta sügis) (CoinTape ... 30.11.2015).



Joonis 14. Bitcoini tehingu eeldatav kinnitusaeg (sõltuvalt teenustasu suurusest)

Nagu jooniselt 14 näha, jäävad paljud madalama teenustasuga tehinguid väga pikaks ajaks nn ooteseisundisse. See tähendab, et makse saajal puudub ajutiselt õigus laekunud rahasumma kasutamiseks, kuna vastavat kannet ei ole veel plokiahelas fikseeritud. Samas ei ole aga tehingut võimalik ka taasedastada (nt suurema teenustasuga), kuna võrgustikule teadaolevalt on antud väljundid juba eelmise tehingu raames kulutatud. Selline tupikseis võib halvemal juhul kesta nädalaid ning nõuab peaaegu alati rakenduse haldaja manuaalset sekkumist. Teiseks levinud probleemiks on aga tehingutasude ülemaksmine ehk olukord, kus teenustasudena makstakse välja hoopis liiga suur summa raha (nt algoritmi vea tõttu). Ehkki mõningad kaevandusfondid (ingl k *mining pool*) tegelevad ka taoliste eksimuste kasutajale korvamisega (0.93 BTC Lost ... 17.11.2015), on selline käitumine pigem harv erand kui üldlevinud reegel.

Eelkirjeldatud probleemide lahendamiseks on loodud hulk andmebaasiserveris talletatud rutiine, mis võimaldavad mugavalt välja arvutada mistahes tehingu soovitusliku teenustasu (vt joonis 15 ja lisa 1). Teenustasude arvutamisel lähtutakse Bitcoini ja Litecoini dokumentatsioonis toodud soovitustest, mille kohaselt tuleks iga plokiahelasse lisatava 1 kB tehinguandmete eest kaevurile välja käia vähemalt 0.0001 BTC (~ 0.03 EUR) (Bitcoin Core ... 01.12.2015) või 0.001 LTC (~ 0.003 EUR) (Litecoin Core ... 01.12.2015) lõivu.



Joonis 15. Tehingu optimaalse teenustasu arvutamise protsess

Nagu jooniselt 15 näha, pöördub rakenduse serverikomponent enne tehingu teele saatmist talletatud rutiini *f_estimate_transaction_fee(..)* poole, mis võtab parameetrina sisse vastava valuuta nime (*in_currency_name*) ning tehingu binaarkujule teisendatud algandmed (*in_hex_transaction*). Kuna tehingu teenustasu suurus sõltub otseselt selle andmemahust, kutsutakse esmalt välja talletatud rutiini nimega *f_calc_transaction_size(..)*. Viimasest saadav tehinguandmete maht (nt 0.412 kB) on eelkõige ennustuslikku laadi ning ei pruugi seetõttu olla kõige täpsem. Seejärel pöördutakse rutiinide *f_calc_btc_transaction_fee(..)* ning *f_calc_ltc_transaction_fee(..)* poole, et selgitada välja vastava valuuta teenustasu standardmäär (Bitcoini puhul 0.0001 BTC). Viimase kahe korrutamisel võrgustiku hetkeolukorda peegeldava teenustasu koefitsiendiga (parameeter *in_fee_coefficient*) (nt 3.0x) saadaksegi kätte tehingu lõplik teenustasu summa (antud juhul u 0.0003 BTC).

Eelnevast oluliselt mugavamaks lahenduseks oleks olnud kasutada Bitcoini alustarkvara poolt pakutavat *estimatefee* JSON-RPC käsku, mis annab samuti esialgse lähenduse tehingu soovituslikule teenustasule. Paraku on aga tegu Bitcoini-spetsiifilise laiendusega, mida ei ole veel integreeritud peaaegu ühegi teise krüptovaluuta alustarkvarasse (sh Litecoin Core'i,

2015. aasta augusti seisuga). Võimaliku alternatiivina kaaluti ka kolmandate osapoolte poolt pakutavate teenuste ning APIde kasutamist (nt CoinTape). Viimaste puhul kerkis aga esile terve rida teisi puuduseid ja probleeme (madal töökindlus, muutlik API, HTTPS-i toe puudumine jne), mis muutsid nende kasutamise samuti ebaotstarbekaks.

3.3.4 Tehinguandmete deformeeritavus

Tehinguandmete deformeeritavus (ingl k *transaction malleability*) on üks krüptovaluutade seas kõige laiemalt levinud probleeme, mis seisneb kinnitamata tehingute unikaalse identifikaatori ootamatus muudetavuses (Wirdum 2015). Nagu juba varem öeldud, on tehingu unikaalne identifikaator (ehk nn TxID) võrgusõlme tarkvara poolt genereeritav 64-kohaline kuueteistkümnendkood, mis määrab üheselt ära iga võrgustikule edastatud tehingu. Identifikaatori väärtus sõltub seejuures otseselt tehingu sisust, kuna selle arvutamiseks kasutatakse tehingu enda binaarkujule teisendatud andmeid.

Tehing ise koosneb hulgast võti-väärtus paaridest, millest tähtsamad (nt sihtaadress(id), lunastatavad väljundid jmt) on kaetud veel omakorda krüptograafilise pitseriga. Viimase eesmärgiks on kaitsta tehingu elutähtsaid osi pahatahtlike muudatuste eest ning tagada, et see jõuaks kaevuriteni moonutamata kujul. Iga kord, kui võrgusõlme tarkvara saab võrgustikult uue tehingu, kontrollitakse eelnimetatud signatuuri, et veenduda tehinguandmete autentsuses. Juhul kui signatuuri arvutamisel selgub, et tehinguandmeid on vahendusründe (ingl k *man-in-the-middle attack*) korras muudetud, loeb võrgusõlme tarkvara tehingu kehtetuks ning keeldub seda teistele sõlmedele edastamast. Selliselt hoitakse ära ebakorrektsete (nt võltsitud, andmeedastuse käigus korrumpeerunud vmt) tehingute sattumine vastava valuuta plokiahelasse.

Probleem seisneb aga selles, et tehingu autentsust tõendav signatuur katab vaid osa kõigist veergudest, mida kasutatakse selle unikaalse identifikaatori (ehk nn TxID) arvutamiseks (Felten 2014). Seetõttu on pahatahtlikel võrgusõlmedel võimalik muuta tehingu metaandmeid ilma, et see mõjutaks eelnimetatud signatuuri kehtivust. Tulemuseks on olukord, kus võrgustikus ringleb korraga sadu identseid (kuid erinevat identifikaatorit omavaid) tehinguid, mis kõik viitavad samadele väljunditele. Kuna samade väljundite mitmekordne kulutamine (ingl k *double-spending*) on aga protokolli reeglitega rangelt keelatud, jõuab eelnimetatud tehingutest lõpuks plokiahelasse vaid üks eksemplar. Kõik ülejäänud eksemplarid (sh nende

põhjal koostatud uued tehingud) tunnistatakse automaatselt kehtetuks ning eemaldatakse võrgusõlme tehingupuhvrist.

Tehinguandmete deformeeritavus raskendab seega oluliselt krüptovaluutadega seonduva tarkvara arendamist, kuna see vähendab tehingu universaalse identifikaatori (ehk nn TxID) kasulikkust ning sunnib otsima alternatiivseid meetodeid võrgustikule edastatud tehingute identifitseerimiseks. Lisaks sellele seab see kahtluse alla ka kinnitama tehingute usaldusväärsuse, kuna rakendusel puudub võime teha vahet nn tavalistel ja kloonitud tehingutel.

Vaatleme näiteks olukorda, kus ettevõte müüb internetis käsitööna valmistatud kaelakeesid ja sõrmuseid, mille eest on võimalik tasuda nii pangalingi kaudu kui ka bitcoinides. Kelm valib firma kodulehelt välja neli ühesuguse hinnaga (nt 0.5 BTC) kaelakeed ning avaldab soovi nende tellimiseks. Süsteem genereerib kliendile uue arve ning lisab sellele personaalse Bitcoini aadressi. Kelm sisestab eelmainitud aadressi oma rahakotitarkvarasse ning kannab sellele 0.5 BTC, tasudes ettevõttele seega vaid ühe kaelakee eest. Seejärel kasutab ta spetsiaalselt modifitseeritud võrgusõlme tarkvara, et muuta äsjasaadetud tehingu metaandmeid ning koostada selle põhjal kolm peaaegu identset makset. Olles märganud kliendi poolt saadetud nelja tehingut koguväärtuses ~2 BTC, pakib ettevõtte töötaja tellitud kauba kenasti ära ning annab selle kohaletoimetamiseks üle kullerile. Paar tundi hiljem lisatakse mõni eelkirjeldatud tehingutest lõpuks plokiahelasse. See toob aga kaasa kõigi ülejäänud tehingute kehtetuks muutumise. Lõpptulemuseks on olukord, kus klient on saanud firmalt vähemalt 2 BTC (ehk u 628 EUR) väärtuses kaupu, mille eest on reaalselt makstud vaid ~ 0.5 BTC (ehk u 157 EUR).

Kuna taoliste petuskeemide võimaldamine toob tõsiseid probleeme ka rahakotiteenust pakkuvale ettevõttele endale, on käesolevas töös loobutud nn kinnitamata tehingute usaldamisest. See tähendab, et kliendil on võimalik oma laekumisi kulutada alles siis, kui need on saanud krüptovaluuta võrgustikult piisava arvu kinnitusi (nt Bitcoini puhul 3, Litecoini puhul aga 12 kinnitust). Lisaks kliendi huvide kaitsmisele aitab taoline lähenemine ära hoida olukordi, kus vaadeldavat süsteemi kasutatakse petturlike tehinguahelate (ingl k *chain of unconfirmed transactions*) koostamiseks. Viimaste puhul on üsna levinud praktikaks, et kaupmehele tekitatud kahju nõutakse välja hoopis kelmi tegevust vahendanud rahakotiteenuselt, mitte kelmilt endalt. Veel tuleks silmas pidada, et kuna tehingu unikaalne

identifikaator (ehk nn TxID) ei ole muutumatu suurus, tuleks kannete identifitseerimiseks kasutada mõnda teist unikaalset väärtust. Käesolevas töös on selleks loodud spetsiaalne tabeliveerg nimega *local_uid*, mis sisaldab andmebaasisüsteemi poolt tehingule omistatud 36-kohalist tähtnumbrilist koodi. Taolise alternatiivvõtme kasutamine võimaldab süsteemi tehingumootori üles ehitada selliselt, et see on pea täielikult immuunne tehinguandmete deformeeritavusega kaasnevate komplikatsioonide vastu.

Tehinguandmete deformeeritavust kasutatakse sageli ka krüptovaluuta võrgustike vastu suunatud ummistusrünnete (ingl k *denial-of-service attack*) korraldamiseks. Seetõttu on äärmiselt oluline, et käesolev süsteem tuleks toime tuhandete või isegi miljonite kloonitud tehingute töötlemisega. Kuna suurem osa taolistest tehingutest kaob võrgustikust mõne tunni jooksul, peab süsteem hoolt kandma selle eest, et kasutajatele kuvataks vaid aktuaalseid tehinguid. Samas ei tohi kloonitud tehinguid ka andmebaasist kustutada, kuna neid võib vaja minna hilisemate konfliktide (sh pettused, vargused jmt) lahendamisel (n-ö tõendusmaterjalina). Eelkirjeldatud nõuete rahuldamiseks on käesolevas töös loodud andmebaasiserveris talletatud rutiin nimega *f_drop_transactions(..)* (vt alljärgnev koodinäide). Tegu on rakenduse serverikomponendi poolt perioodiliselt väljakutsutava pakktöötlusliku rutiiniga, mille eesmärgiks on arhiveerida teatud ajaperioodi jooksul (parameeter *in_txn_timeout*) kinnitamata jäänud krüptovaluuta tehinguid. Kõik arhiveerituks (DROPPED) loetud tehingud jäävad muutmata kujul andmebaasi alles, kuid on rakenduse serverikomponendi jaoks sisuliselt nähtamatud.

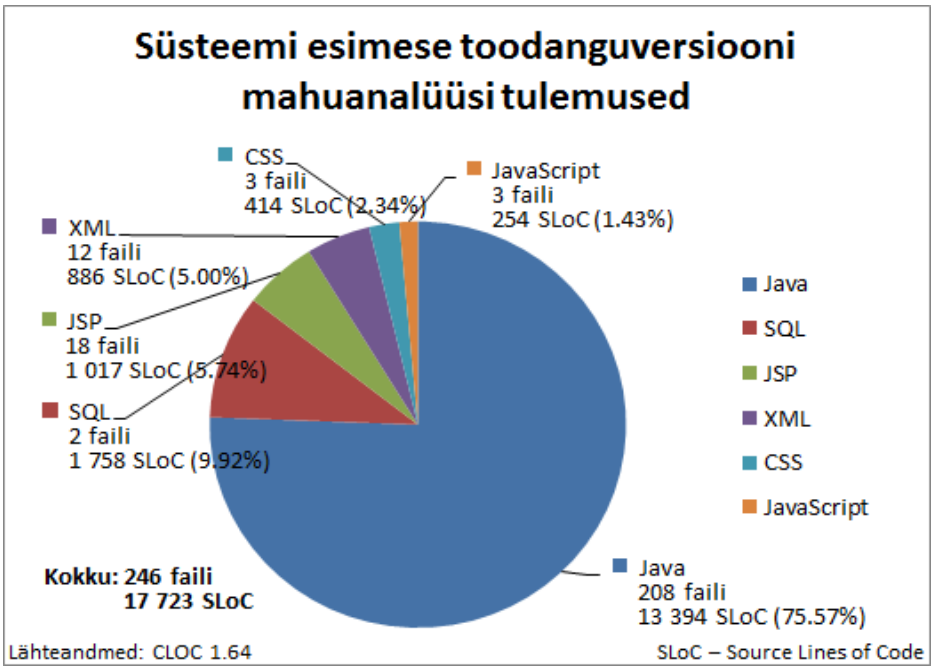
```
CREATE OR REPLACE FUNCTION f_drop_transactions(in_txn_timeout INTEGER,
in_chain_code VARCHAR(30)) RETURNS CHAR(64)[] AS $$
WITH dropped_transactions AS (
   UPDATE transactions AS t SET transaction_status_type_id = 6
   FROM transaction_endpoint AS te, address AS a, address_type AS at, chain AS ch
   WHERE extract(epoch FROM (CURRENT_TIMESTAMP(0) - t.received_at)) >=
         in_txn_timeout AND t.transaction_status_type_id = 2
      AND t.transaction_id = te.transaction_id AND te.address_id = a.address_id
      AND a.address_type_id = at.address_type_id AND at.chain_id = ch.chain_id AND
         ch.code = in_chain_code
   RETURNING t.network_uid
)
SELECT array_agg(network_uid) FROM dropped_transactions;
$$ LANGUAGE sql STRICT;
```

3.4 Järeldused ja ülevaade loodud süsteemist

Nagu eelnevast näha, on tsentraliseeritud krüptovaluuta rahakotiteenuste arendamine ning ülalpidamine väga töömahukas protsess. Iga rakenduse toodanguversiooni minev koodi- ja struktuurimuudatus nõuab pikka testimisperioodi ning äärmiselt põhjaliku eelretsenseerimist. Seetõttu sobivad taolised süsteemid hästi kokku nn kogukondliku arendusmudeliga. Paraku on aga krüptoraha näol tegu niivõrd noore valdkonnaga, et vähesed tsentraliseeritud rahakotiteenuseid pakkuvatest ettevõtetest on nõus oma toodete lähtekoodi avalikustama. Taoline turvapoliitika (nn *security through obscurity*) ei ole aga jätkusuutlik ning julgustab madalakvaliteedilise programmikoodi kirjutamist. Lisaks sellele õõnestab see kasutajate usaldust süsteemi loojate suhtes, kuna puudub selge ülevaade rakenduse arendusprotsessist ning selle peamistest tööpõhimõtetest.

Käesoleva magistritöö raames valminud rahakotiteenust (Bitplexus) on algusest peale arendatud n-ö avalikus vormis (Bitplexus ... 07.12.2015). Süsteemi lähtekood on litsentseeritud GNU GPLv3 alusel, mis tähendab, et kõigil huvilistel on võimalik seda vabalt lugeda, muuta ning kasutada. Ainsaks tingimuseks on, et kõik tuletatud programmid (st vaadeldava rakenduse lähtekoodi kasutavad süsteemid) peavad samuti oma lähtekoodi avalikustama. Selliselt on tagatud kogukonna jätkuv panus esialgse koodibaasi arendamisse ning elementaarne juriidiline kaitse ebaeetiliste ettevõtete eest.

Saamaks paremat ülevaadet loodud süsteemi keerukusest, on selle koodibaasi põhjal läbi viidud ka esialgne mahuanalüüs (vt joonis 16). Nagu jooniselt 16 näha, on süsteemi peamine keerukus koondunud eelkõige selle serverikomponenti (Java – 75.57%) ning andmebaasi (SQL – 9.92%), kusjuures kasutajaliides moodustab vaid väikese osa süsteemi kogumahust (JSP – 5.74%, CSS – 2.34%, JavaScript – 1.43%). Selline jagunemine on ka igati ootuspärane, kuna valdav osa rakenduse arendamisele pühendatud ajast (kokku ~ 5 kuud aktiivset tööd) kuluski just selle multivaluuta toe ehitamisele ning tehingumootori väljatöötamisele. Tulemuseks on universaalne rahakotiteenus, mida on võimalik operatiivselt (~ 1 päev tööd) täiendada teiste sarnaste krüptovaluutadega (vt lisa 2).



Joonis 16. Süsteemi esimese toodanguversiooni mahuanalüüsi tulemused

Autorile teadaolevalt on tegu ainsa omalaadse projektiga (s.o tsentraliseeritud, multivaluuta tuge omav, Java EE platvormil põhinev rahakotiteenus), mille lähtekood on internetis vabalt kättesaadav. Seetõttu on ka kogukonna huvi kõnealuse süsteemi vastu üsna suur – eelmainitud koodihoidlat külastab päevas keskmiselt 10–15 inimest. Rahakotiteenusest veelgi enam tähelepanu on aga võitnud Bitcoini ja Litecoini võrgustikega suhtlemiseks loodud Java konnektorteegid *btcd-cli4j* (btcd-cli4j ... 07.12.2015) ja *ltcd-cli4j*. Viimased refaktoreeriti projekti varajases arengustaadiumis ümber iseseisvateks alamprojektideks, et suurendada lõpptoote modulaarsust. Sealt on nad aga edasi jõudnud mitmetesse teistesse krüptoraha teemalistesse süsteemidesse ning foorumiaruteludesse (How to communicate ... 08.12.2015). Autorile teadaolevalt on käesoleva töö raames valminud konnektorteeki *btcd-cli4j* kasutatud ka nt Saksa finantstehnoloogia idufirma (ingl k *fintech*) *NUMBER26* toodetes.

Ehkki praktikas on tsentraliseeritud rahakotiteenused äärmiselt levinud, kaasneb nendega ka mitmeid peidetud puuduseid ja probleeme. Kliendi vaatevinklist on suurimaks ohuks kindlasti kontrolli loovutamine oma salajaste võtmete üle. Eelnev on sisuliselt võrreldav olukorraga, kus klient vahetab mingi füüsilise käibevahendi (nt kuldmünt) välja teenusepakkuja poolt antava lubaduse vastu (jääk pangakontol). Kuna krüptoraha valdkonna teenusepakkujad on aga niivõrd vähetuntud, ei maksa nende lubadused praktiliselt midagi. Seetõttu tuleks käesoleva töö raames valminud süsteemi (ja selle derivaate) kasutada vaid väikeste

igapäevatehingute sooritamiseks. Suuremate summade haldamiseks on soovitatav kasutada nn täisvalideeruvat (ingl k *full validation*) rahakotitarkvara (nt Bitcoin Core, Armory, mSIGNA) või spetsiaalseid riistvaralisi rahakotiseadmeid (Trezor, Ledger Nano, KeepKey jt).

3.5 Edasised arenguvõimalused

Krüptoraha tehingud on anonüümsed, momentaansed ning tagasivõtmatud (Foxman 2013). See muudab nad ideaalseks sihtmärgiks varastele ja petturitele, kuna ohvril puudub praktiliselt igasugune võimalus kurjategija tabamiseks ning oma kontojäägi ennistamiseks. Seetõttu hinnatakse krüptovaluuta rahakotiteenuseid sageli just selle järgi, milliseid vahendeid nad kasutavad oma klientide varade kaitseks. Üheks kasulikuks skeemiks, mida plaanitakse rakendada ka käesoleva süsteemi tulevastes versioonides, on nn multiallkirja aadressid (ingl k *multisignature addresses*). Sisuliselt on tegu krüptovaluuta aadressidega, millel on rohkem kui üks salajane võti. Selleks, et lunastada eelmainitud aadressile kogunenud rahalisi vahendeid, tuleb tehing allkirjastada vähemalt *m*-i salajase võtmega kõigist *n*-ist olemasolevast.

Käesoleva süsteemi puhul oleks mõistlik rakendada nn kaks-kolmest multiallkirja skeemi (ingl k *2-of-3 signature scheme*). See tähendab, et esimese võtmega allkirjastamise õigus antakse vaid rahakotiteenust pakkuvale ettevõttele (antud juhul Bitplexus LLC). Teine ja kolmas võti saadetakse automaatselt välja kliendi mobiiltelefonile (SMSi vahendusel), kusjuures teine võti võetakse arvele ka erapooletu audiitorfirma (e oraakli) juures (nt CryptoCorp). Kolmas võti jääb aga kliendile tagavaravõtmeks ning seda tuleks kasutada vaid juhtudel, kui teenusepakkuja ja/või oraakel ei ole mingil põhjusel kättesaadavad. Selliselt on tagatud kliendi ainukontroll talle kuuluvate varade üle (läbi kliendi/oraakli ühise võtme + tagavaravõtme). Igapäevatehingute sooritamiseks kasutab süsteem aga kombinatsiooni enda ja oraakli salajastest võtmetest. See aitab parandada multiallkirja aadresside kasutusmugavust, võttes kliendilt ära salajaste võtmete manuaalse sisestamise kohustuse. Multiallkirja aadresside peamiseks eeliseks on, et nad tagavad kliendi varade kättesaadavuse ka pärast kriitilisi sündmusi nagu ettevõtte pankrotistumine, serveriruumi hävimine jne.

Teiseks oluliseks arengusuunaks oleks täiendada käesoleva töö raames valminud süsteemi nn deterministliku aadressihalduse (ingl k *hierarchical deterministic wallet*) võimalustega. Hetkel kasutab süsteem uute aadresside genereerimiseks nn mittedeterministliku (ingl k

nondeterministic wallet) lähenemist, mis tähendab, et kõik klientidele eraldatud aadressid on täiesti juhuslikud (st nende loomisel ei ole järgitud mingit läbivat loogikat). Taoline lähenemine muudab aga väga keeruliseks klientide salajaste võtmete varundamise, kuna aadresse (ja neile vastavaid salajasi võtmeid) tekib ajas pidevalt juurde. Lisaks eelnevale raskendab see mitme rahakotiseadme paralleelset kasutamist (nt arvuti + nutitelefon), kuna puudub tõhus meetod salajaste võtmete jagamiseks/grupiviisiliseks migreerimiseks.

Deterministliku aadressihalduse puhul tuletatakse aga kõik kliendi krüptovaluuta aadressid talle omistatud personaalsest seemnefraasist (ingl k *seed phrase*). Kuna aadresside loomine toimub kindla algoritmi alusel (Bitcoini standardid BIP0032 ning BIP0044), on kogu protsess täielikult deterministlik (e korratav). Seetõttu piisab süsteemile vaid kliendi personaalse seemnefraasi teadmisest, et taastada kogu tema rahakoti eelnev seisund. Tulemuseks on olukord, kus nii rahakottide varundamise, eksportimise kui ka importimise ülesanded taanduvad sisuliselt eelmainitud seemnefraaside haldamisele. Loomulikult kaasneb taolise lähenemisega ka mitmeid tõsiseid probleeme – näiteks dilemma, kuidas kaitsta seemnefraase võõraste silmade eest (ilma neid ühesuunaliselt krüpteerimata). Lisaks sellele tuleks uurida, millised olemasolevatest teekidest/võrgusõlme realisatsioonidest üldse toetavad taolist funktsionaalsust ning mis on nende peamised eelised ja puudused.

Kokkuvõte

Käesoleva töö eesmärgiks oli luua tsentraliseeritud krüptovaluuta rahakotiteenus, mida oleks võimalikult lihtne täiendada uute plokiahelapõhiste krüptovaluutadega. Selleks tutvuti erinevate krüptovaluutade tööpõhimõtetega ning viidi läbi vastava infosüsteemi (Bitplexus) lihtsustatud süsteemianalüüs. Analüüsi tulemuste põhjal pandi paika rakenduse esialgse süsteemiarhitektuur ning arendati välja selle aluseks olev andmebaas. Seejärel valiti välja kaks kõige levinumat krüptovaluutat (Bitcoin ja Litecoin) ning liidestati nad süsteemi esimese toodanguversiooniga. Krüptovaluutade valiku kriteeriumiks oli, et nad oleksid võimalikult populaarsed (kasutajate arv) ning perspektiivikad (turukapitalisatsiooni suurus). Mitme valuuta kaasamisega veenduti loodava süsteemi valuuta-agnostilisuses ning uute valuutade lisamise lihtsuses.

Töö olulisimaks tulemuseks on selle raames valminud rahakotiteenuse prototüüp, mida on tänu tema paindlikule litsentsile (GNU GPLv3) võimalik kasutada ka mitmete teiste analoogiliste süsteemide arendamisel. Töö autorile teadaolevalt on tegu ainsa omalaadse projektiga (s.o tsentraliseeritud, multivaluuta tuge omav, Java EE platvormil põhinev rahakotiteenus), mille lähtekood on internetis vabalt kättesaadav. Lisaks sellele loodi käesoleva magistritöö raames ka Bitcoini ja Litecoini võrgustikega suhtlemiseks mõeldud Java konnektorteegid nimega *btcd-cli4j* ning *ltcd-cli4j*. Viimaste eeliseks võrreldes teiste analoogiliste teekidega on nende töökindlus (konfigureeritav logimine), turvalisus (HTTPS tugi) ning ajakohasus (ühilduvus erinevate võrgusõlme tarkvaradega). Lisaks konkreetsetele artefaktidele pakuti käesolevas töös välja ka üldistatud domeenimudel plokiahelapõhiste krüptovaluutadega liidestumiseks. Ühtlasi vaadeldi mõningaid krüptoraha teenuste tüüpilisi puuduseid ja probleeme ning esitati üldkasutatavad lahendused nende leevendamiseks.

Ehkki süsteemi lõpliku toodanguversioonini jõudmiseks tuleb selles teha veel mitmeid olulisi täiendusi, said kõik antud töös püstitatud funktsionaalsed ja mittefunktsionaalsed nõuded täielikult rahuldatud. Saadud tulemustest võib järeldada, et eelkirjeldatud metoodika (sh üldistatud domeenimudel ning lahendused tüüpprobleemidele) sobib igati universaalsete krüptoraha teenuste väljatöötamiseks. Praeguse lahenduse suurimaks puuduseks paistab olevat

selle kehv turvalisus, mistõttu võiks edaspidises uurida nt erinevate multiallkirja skeemide poolt pakutavaid võimalusi. Lisaks sellele võiks kaaluda ka deterministliku aadressihalduse integreerimist käesolevas töös vaadeldud süsteemi.

Summary

The purpose of this thesis was to develop a centralized cryptocurrency wallet service with built-in support for integrating new block chain-based cryptocurrencies. To achieve this goal, several cryptocurrencies were examined and a simplified systems analysis conducted for the aforementioned system (Bitplexus). The resulting specification was used to establish a preliminary system architecture for the application and to fully implement its data layer. Next, two widely accepted cryptocurrencies (Bitcoin and Litecoin) were chosen to be integrated into the first production release of the system. The initial currencies were selected based on their prominence (*i.e.* market capitalization) and popularity in the community (*i.e.* number of users). By integrating multiple cryptocurrencies, the author was able to verify the viability of the application's multicurrency engine and to polish the process of adding new currencies.

The main outcome of this thesis is an open-source (GNU GPLv3) multicurrency web wallet that serves as a basis for constructing other similar systems. To the author's knowledge, it is the only centralized cryptocurrency wallet service (based on the Java EE platform) whose source code is publicly available on the internet. In order to communicate with the Bitcoin and Litecoin networks, two Java wrapper libraries named *btcd-cli4j* and *ltcd-cli4j* were also developed. The main advantage of these libraries compared to other wrappers available on the market is their reliability (configurable logging), security (full HTTPS support) and compatibility. In addition to the concrete results described above, several abstract conclusions were also drawn. Firstly, a generic domain model for interfacing with block chain-based cryptocurrencies was proposed. Secondly, a number of common problems affecting cryptocurrency-related services were described and universal solutions presented to mitigate them.

Although parts of the system still need significant improving before full production readiness can be achieved, all the functional and nonfunctional requirements outlined in this thesis were fully met. Thus, it can be concluded that the methodology described above (incl. the generic domain model and solutions to well-known problems) is completely fit for developing cryptocurrency-related services. As the greatest weakness of the current system is its poor

security, a potential direction for future work would be to examine the pros and cons of various multisignature schemes. Additionally, better tools for implementing hierarchical deterministic wallet capabilities could be developed.

Kasutatud materjalid

1. 0.93 BTC Lost in Transaction Fees, BitFury? – Reddit. [WWW] https://www.reddit.com/r/Bitcoin/comments/3jnadz/093_lost_btc_in_transaction_fee_bitfury/cuqsdi1/ (17.11.2015)
2. Antonopoulos, A. M. (2014). Mastering Bitcoin: Unlocking Digital Cryptocurrencies. 1st ed. Sebastopol : O'Reilly Media.
3. Avoiding Key Reuse – Bitcoin Developer Guide. [WWW] https://bitcoin.org/en/developer-guide#avoiding-key-reuse (21.11.2015)
4. Bitcoin Core 0.9.0 Release Notes. [WWW] https://bitcoin.org/en/release/v0.9.0 (01.12.2015)
5. Bitplexus – A Proof-Of-Concept Universal Cryptocurrency Wallet Service. [WWW] https://github.com/priiduneemre/bitplexus/ (07.12.2015)
6. btcd-cli4j – A Simple Java Wrapper Around Bitcoin Core's JSON-RPC Interface. [WWW] https://github.com/priiduneemre/btcd-cli4j/ (07.12.2015)
7. Cawrey, D. (2014). Are 51% Attacks a Real Threat to Bitcoin? [WWW] http://www.coindesk.com/51-attacks-real-threat-bitcoin/ (22.09.2015)
8. Chisholm, M. (2000). Managing Reference Data in Enterprise Databases: Binding Corporate Data to the Wider World. 1st ed. San Francisco : Morgan Kaufmann.
9. CoinTape – Predicting Bitcoin Transaction Fees. [WWW] http://www.cointape.com/ (30.11.2015)
10. Corallo, M., Schneider, N. (2012). BIP 21: A URI Scheme for Making Bitcoin Payments. [WWW] https://github.com/bitcoin/bips/blob/master/bip-0021.mediawiki/ (09.10.2015)
11. Dorsey, P. (2013). Don’t Be Thick: Use a „Thick Database“ Approach. [WWW] http://dulcian.com/dont-be-thick-use-a-thick-database-approach/ (06.11.2015)
12. Eessaar, E. (2012). Ülikooli infosüsteemi õpingukavade allsüsteem (7.16). TTÜ õppeaine „Andmebaasid II“ (IDU0230) näidisprojekt.
13. Eesti keele seletav sõnaraamat. (2009). 2nd ed. Tallinn : Pakett AS.
14. Eyal, I., Sirer, E. G. (2014). Majority Is Not Enough: Bitcoin Mining Is Vulnerable. – *Lecture Notes in Computer Science*, 8437, 436–454. [Online] SpringerLink LNCS (19.09.2015)

15. Felten, E. (2014). Understanding Bitcoin's Transaction Malleability Problem. [WWW] https://freedom-to-tinker.com/blog/felten/understanding-bitcoins-transaction-malleability-problem/ (28.11.2015)
16. Foxman, S. (2013). Hackers Are Trying to Create an Untraceable and Comprehensive Financial System Using Bitcoin. [WWW] http://qz.com/75457/using-bitcoin-hackers-are-trying-to-create-an-untraceable-financial-system/ (03.12.2015)
17. Gorale, A. (2014). Explaining the Math Behind Bitcoin. [WWW] https://www.cryptocoinsnews.com/explaining-the-math-behind-bitcoin/ (24.09.2015)
18. How to communicate between Java and bitcoind? – Bitcoin Stack Exchange. [WWW] http://bitcoin.stackexchange.com/q/7529/6945/ (08.12.2015)
19. Infosüsteemide turve II: Turbetehnoloogia. (1998). / A. Ansper, A. Buldas, V. Hanson, H. Lipmaa, T. Martens, V. Tulit. 1st ed. Tallinn : Cybernetica.
20. Khaosan, V. (2014). How a Bitcoin Transaction Works. [WWW] https://www.cryptocoinsnews.com/bitcoin-transaction-really-works/ (14.09.2015)
21. Litecoin Core 0.8.6.2 Release Notes. [WWW] http://blog.litecoin.org/2014/01/litecoin-0862-release-notes.html (01.12.2015)
22. Lodi, G. (2014). Regular Expression To "Validate" a Bitcoin Address. [WWW] http://mokagio.github.io/tech-journal/2014/11/21/regex-bitcoin.html (04.10.2015)
23. Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. [WWW] https://bitcoin.org/bitcoin.pdf (12.10.2015)
24. Pallas, R. (2012). Bitcoin Security : magistritöö. Tallinna Tehnikaülikool, Tallinn.
25. Perez, Y. B. (2015). Europol: Cryptocurrency Serves a 'Crime-As-A-Service' Business Model. [WWW] http://www.coindesk.com/europol-cryptocurrency-promotes-crime-as-a-service-business-model/ (07.09.2015)
26. Piasecki, P. (2012). Design and Security Analysis of Bitcoin Infrastructure Using Application Deployed on Google Apps Engine : magistritöö. Lodz University of Technology, Łódź.
27. Sauga, A. (2015). Mis toimub krüptorahandusega? – *Investeeri*, 2, 54–57.
28. Shirriff, K. (2014). Bitcoin Mining the Hard Way: The Algorithms, Protocols, and Bytes. [WWW] http://www.righto.com/2014/02/bitcoin-mining-hard-way-algorithms.html (19.10.2015)
29. Siri, L. (2014). The Closest You Can Get to Perfectly Secure Bitcoin Transactions (Short of Doing Them in Your Head). [WWW] https://www.turnkeylinux.org/blog/secure-bitcoin-transactions/ (14.09.2015)

30. Süsteemide ja tarkvara kvaliteedinõuded ja kvaliteedi hindamine. (2012). Süsteemide ja tarkvara kvaliteedimudelid : Eesti standard EVS-ISO/IEC 25010:2011. Tallinn : Eesti Standardikeskus.
31. Tarhini, A. (2011). Concepts of Three-Tier Architecture. [WWW] https://alitarhini.wordpress.com/2011/01/22/concepts-of-three-tier-architecture/ (28.10.2015)
32. Wirdum, A. (2015). The Who, What, Why and How of the Ongoing Transaction Malleability Attack. [WWW] https://bitcoinmagazine.com/articles/the-who-what-why-and-how-of-the-ongoing-transaction-malleability-attack-1444253640/ (27.11.2015)

Lisa 1

Järgnevalt on esitatud talletatud rutiin, mis võimaldab välja arvutada suvalise krüptovaluuta tehingu optimaalse teenustasu. Rutiini käivitamiseks tuleb sellele ette anda järgmised väärtused:

- *in_currency_name* – lähtevaluuta ametlik nimetus (nt „Bitcoin“);
- *in_hex_transaction* – tehingu binaarkujule teisendatud algandmed;
- *in_fee_coefficient* – teenustasu võimenduskoefitsient, mis peegeldab võrgustiku hetkeseisundit/teenustasu survet (ingl k *fee pressure*) (nt 2.5x).

```
CREATE OR REPLACE FUNCTION f_estimate_transaction_fee(in_currency_name VARCHAR(25),
in_hex_transaction TEXT, in_fee_coefficient NUMERIC(3, 1)) RETURNS NUMERIC(23, 8)
AS $$
DECLARE
   binary_size INTEGER;
BEGIN
   binary_size := f_calc_transaction_size(in_hex_transaction);
   IF (in_currency_name = 'Bitcoin') THEN
      RETURN CAST(f_calc_btc_transaction_fee(binary_size) *
         GREATEST(in_fee_coefficient, 1) AS NUMERIC(23, 8));
   ELSIF (in_currency_name = 'Litecoin') THEN
      RETURN CAST(f_calc_ltc_transaction_fee(binary_size) *
         GREATEST(in_fee_coefficient, 1) AS NUMERIC(23, 8));
   ELSE
      RAISE EXCEPTION 'Unable to estimate txn fee - currency (name = %)
         invalid/unsupported.', in_currency_name USING ERRCODE = '30302';
   END IF;
END
$$ LANGUAGE plpgsql STABLE STRICT;
```

Eelkirjeldatud rutiin sõltub omakorda rutiinist *f_calc_transaction_size(..)*, mille abil on võimalik välja arvutada mistahes krüptovaluuta tehingu (parameeter *in_hex_transaction*) andmemaht (baitides).

```
CREATE OR REPLACE FUNCTION f_calc_transaction_size(in_hex_transaction TEXT)
RETURNS INTEGER AS $$
SELECT octet_length(decode(in_hex_transaction, 'hex')) AS binary_size;
$$ LANGUAGE sql IMMUTABLE LEAKPROOF STRICT;
```

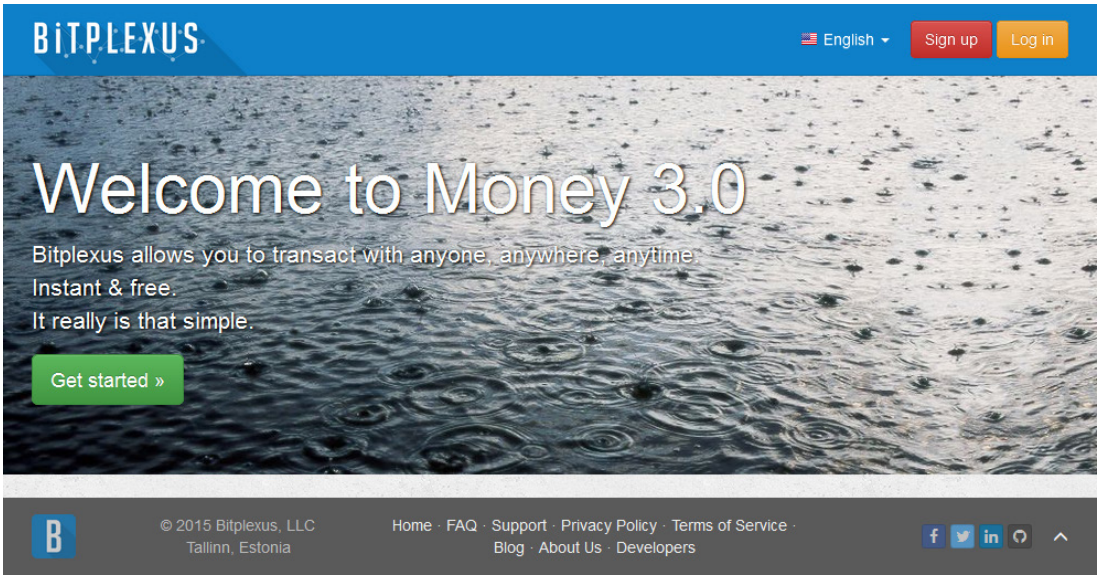
Lisaks sellele kasutab rutiin *f_estimate_transaction_fee(..)* veel ka abirutiine *f_calc_btc_transaction_fee(..)* ning *f_calc_ltc_transaction_fee(..)*, mille eesmärgiks on koondada tasuarvutusprotsessi valuutaspetsiifilisi samme. Näiteks (Bitcoini puhul):

```
CREATE OR REPLACE FUNCTION f_calc_btc_transaction_fee(in_binary_size INTEGER)
RETURNS NUMERIC(23, 8) AS $$
SELECT CAST(GREATEST(standard_fee, ceiling(CAST(in_binary_size AS NUMERIC) /
   CAST(1000 AS NUMERIC)) * standard_fee) AS NUMERIC(23, 8)) AS recommended_fee
FROM currency
WHERE name = 'Bitcoin';
$$ LANGUAGE sql STABLE LEAKPROOF STRICT;
```

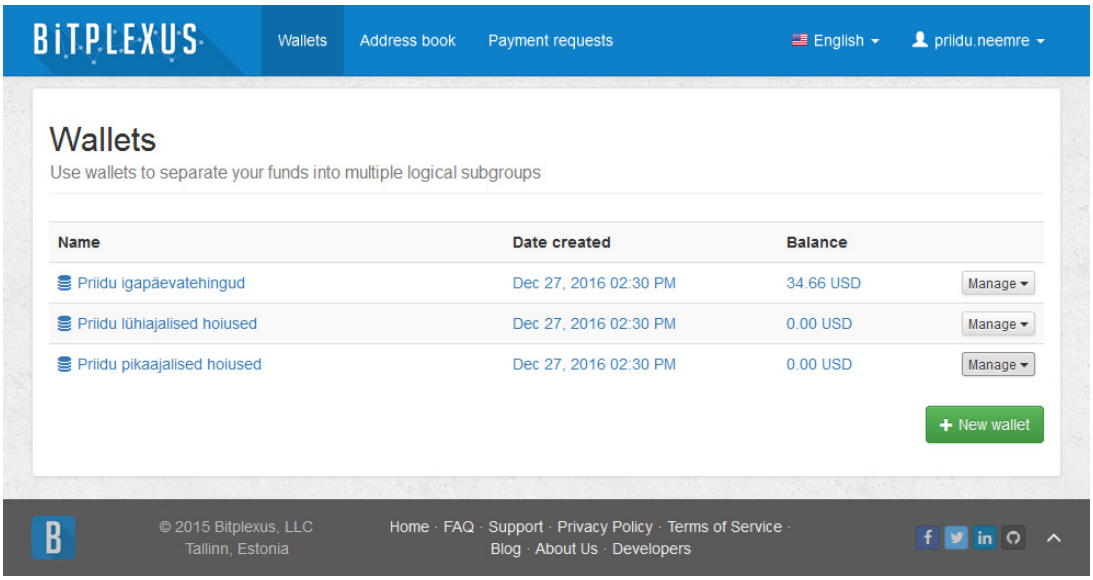
Ehkki Bitcoini ja Litecoini puhul on eelkirjeldatud rutiinid peaaegu identsed, ei pruugi see nii olla mitmete eksootilisemate krüptovaluutade puhul (nt Dash, Ethereum jt). Seetõttu on iga valuuta jaoks loodud eraldi pistikrutiin (ingl k *plugin routine*), et säilitada üldise algoritmi (*f_estimate_transaction_fee(..)*) universaalsus.

Lisa 2

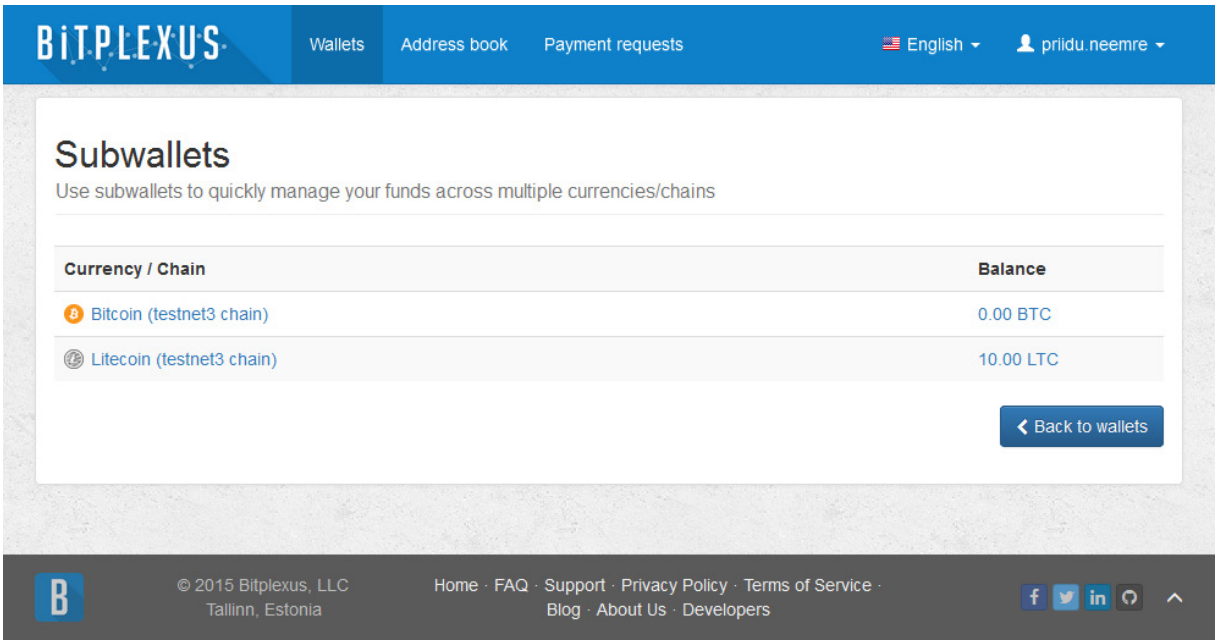
Saamaks paremat ülevaadet käesoleva töö raames loodud süsteemi välimusest, on järgnevas esitatud mõned ekraanipildid selle veebipõhisest kasutajaliidesest (vt joonised L2-1, L2-2, L2-3, L2-4, L2-5 ning L2-6). Iga vaate kohta on ära toodud ka selle kuuluvus pädevusalasse ning funktsionaalsed nõuded, mida antud ekraanivorm rahuldab. Nagu juba eespool mainitud, on süsteemi esimeses toodanguversioonis keskendutud eelkõige selle kliendi pädevusala kasutajaliidese väljatöötamisele.



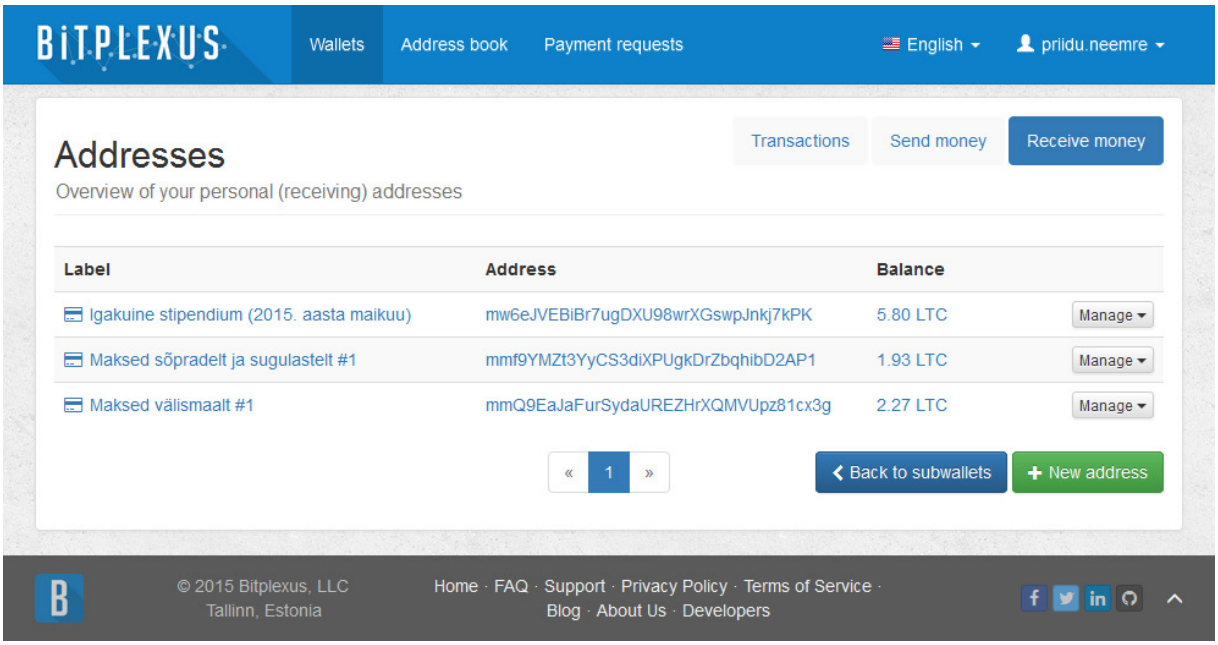
Joonis L2-1. Süsteemi avaleht (enne kasutaja identifitseerimist)



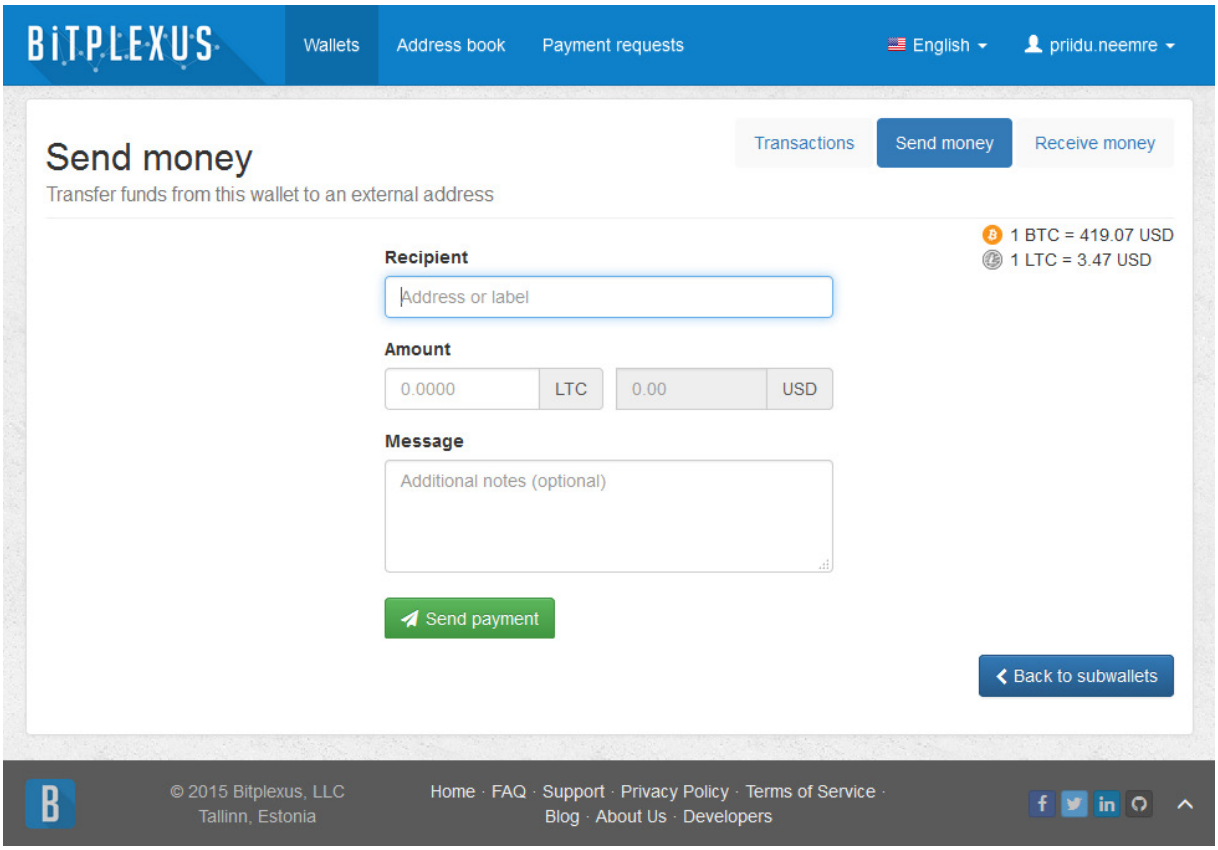
Joonis L2-2. Kliendi pädevusala – rahakottide haldamine (FR-006, FR-007, FR-008)



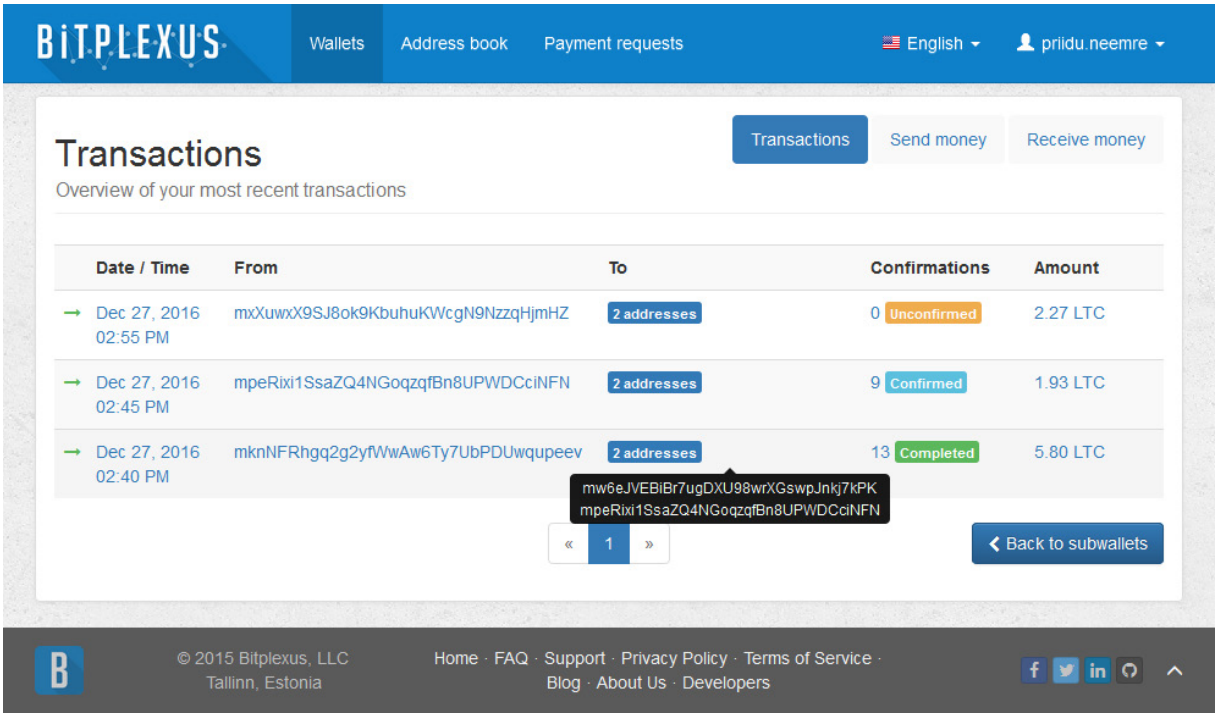
Joonis L2-3. Kliendi pädevusala – alamrahakottide vaatamine



Joonis L2-4. Kliendi pädevusala – aadresside haldamine (FR-009, FR-010, FR-011, FR-012)



Joonis L2-5. Kliendi pädevusala – uue makse sooritamine (FR-013)



Joonis L2-6. Kliendi pädevusala – tehingute ajaloo vaatamine (FR-014)