Lauri Miettinen

**KLASSISTEN AUTOJEN ELINKAAREN SEURANTA ÄLYKKÄÄLLÄ SOPIMUKSELLA**

**KLASSISTEN AUTOJEN ELINKAAREN SEURANTA ÄLYKKÄÄLLÄ SOPIMUKSELLA**

Lauri Miettinen

Opinnäytetyö

Syksy 2016

Kone- ja tuotantotekniikan koulutusohjelma

Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu

Koulutusohjelma, suuntautumisvaihtoehto

Tekijä(t):

Opinnäytetyön nimi:

Työn ohjaaja(t):

Työn valmistumislukukausi ja -vuosi: Sivumäärä:60 + 5 liitettä

Tiivistelmä on suppea, itsenäinen esitys, joka antaa lukijalle kuvan tehdystä työstä. Se kirjoitetaan sitten, kun työ on valmis. Tiivistelmä kirjoitetaan täydellisin virkkein ja passiivimuodossa. Siinä selostetaan lyhyesti työn aihe, tavoitteet, työn keskeisin toteutustapa ja olennaiset tulokset.

Sisältö jakaantuu yleensä kolmeen kappaleeseen:

1. työn aihe ja tavoitteet

2. menetelmät, toteutustapa ja eteneminen

3. tulokset ja esitetyt johtopäätökset.

Tiivistelmän tulee mahtua yhdelle sivulle rivivälillä 1.

Korvaa nämä tekstit omillasi.

Asiasanat: x, x, x

(Kirjoita 3–7 kpl työtäsi kuvaavaa asiasanaa*.* Käytä hyväksesi esimerkiksi yleistä suomalaista asiasanastoa YSA, <http://onki.fi/fi/browser/overview/ysa>.)

ABSTRACT

Oulu University of Applied Sciences

Degree programme, option

Author(s):

Title of thesis:

Supervisor(s):

Term and year when the thesis was submitted: Pages: 60 + 5 appendices

Vieraskielinen tiivistelmä (Abstract) tehdään työn tilaajan niin vaatiessa tai osaston käytänteiden mukaisesti. Sen ei tarvitse olla suora käännös suomenkielisestä tiivistelmästä, vaan sen voi kirjoittaa kansainväliselle lukijalle.

Remember to include the following in the abstract:

Why – background for the thesis, assigner

What – objective of the thesis

How – background knowledge applied, research methods, material used

Achievements – main results

Exploitability – conclusions, development proposals, further measures

The length of the abstract is 1 page.

Keywords: x, x, x

(List 3–7 keywords that describe your thesis. Use for example keywords from the following controlled vocabulary thesauri:

MeSH <http://www.yso.fi/onto/mesh/conceptscheme>

Agriforest <http://www-db.helsinki.fi/agri/agrisanasto/Welcome_eng.html>

Helecon <http://helecon3.hkkk.fi/helevoc/?lang=eng&dbname=MIX>)

ALKULAUSE

Raportissa voi olla alkusanat, joissa kiitetään toimeksiantajaa, ohjaajia ja mahdollisesti muita tukijoita tai avustajia. Alkusanat voi sisältää lyhyesti myös muuta työhön liittyvää, lukijaa kiinnostavaa tietoa, joka ei muuten tule raportissa esiin. Alkusanat päivätään ja allekirjoitetaan.

Korvaa nämä tekstit omillasi.

SISÄLLYS

Contents

[1 JOHDANTO 8](#_Toc480956785)

[2 SUUNNITTELU JA KÄYTTÖTAPAUKSET 11](#_Toc480956786)

[2.1 Lohkoketjun hyöty sovelluksessa 11](#_Toc480956787)

[3 OHJELMISTOON TUTUSTUMINEN 13](#_Toc480956788)

[3.1 GoEthereum (geth) 14](#_Toc480956789)

[3.2 Ethereum Wallet 14](#_Toc480956790)

[3.3 Browser Solidity 13](#_Toc480956791)

[3.4 TestRPC 13](#_Toc480956792)

[3.5 Truffle-sovelluskehys 13](#_Toc480956793)

[3.6 Meteor 13](#_Toc480956794)

[4 TYÖN KULKU 15](#_Toc480956795)

[5 LOPPUTULOKSET 16](#_Toc480956796)

[5.1 Toteutuneet käyttötapaukset 16](#_Toc480956797)

[5.2 Toteutumattomia käyttötapauksia 16](#_Toc480956798)

[6 YHTEENVETO 18](#_Toc480956799)

SANASTO

Jos työssä toistuvat samat lyhenteet tai suureiden, yksiköiden tai piirrosten merkinnät ja jos niitä on paljon, niistä kootaan luettelo selityksineen. Otsikkona voi olla sisällön mukaan esimerkiksi LYHENTEET, MERKKIEN SELITYKSET tai SANASTO. Luettelo laaditaan aakkosjärjestykseen. Aakkostuksessa noudatetaan SFS-standardia 4600 ja SI-järjestelmää.

Posta tämä sivu, jos et tarvitse.

# JOHDANTO

Omat kommenttini vihreällä

Työn taustassa on tärkeää selittää seuraavat asiat:

1. Mikä on lohkoketju? Mistä se sai alkunsa? Tämä ei ole kovin selvä asia edes monille teknisille ihmisille, joten paras selittää hyvin.
2. Lohkoketjut on helpompi ymmärtää jos selittää Bitcoinin taustan ja historian. Selitä ne lyhyesti.
3. Selitä Ethereum. Mikä se on, mikä sen idea on ja mistä se sai alkunsa?
4. Selitä itse työn tausta-ajatus. Mikä idea on tehdä lohkoketjuihin liittyvä sovellus vintage-autoista?
5. Kuvaile Klassisten autojen kulttuuria. Mitä piirteitä sillä on, ja mitä nämä ihmiset arvostavat?
6. Pohdiskele klassisten autojen kulttuurin arvoja, ja vakuuta lukija siitä, että lohkoketju on hyvä lisä osaksi kulttuuria.
7. Miksi juuri lohkoketjut? Miksei tätä järjestelmää voi tehdä perinteisellä tietokannalla? Pohdi, miten klassisten autojen kulttuuri hyötyy lohkoketjuista.

Eli tekstiä tulee paljon. Pitänee jakaa johdantokappale useampaan osaan. Ehkä voisi kirjoittaa johdantoon lyhyesti tuo neloskohta, ja sitten myöhemmissä kappaleissa lohkoketjujen historia?

Lohkoketjutekniikka sai alkunsa vuonna 2009, kun Satoshi Nakamoto kirjoitti paperin, jossa hän ehdotti tietomallin puolueettoman valuutan luomisesta. Nykyään valuutta liikkuu pankin tai muun laitoksen kautta. Bitcoinin idea oli luoda valuutta josta ei ole vastuussa kukaan keskitetty taho, jossa maksajan ei tarvitse luottaa keneenkään. Nakamoto suunnitteli hajautetun järjestelmän rahan lähettämiselle ja maksutapahtumien todentamiselle. (1)

Bitcoin- maksut tallennetaan hajautettuun tietokantaan, jota kutsutaan lohkoketjuksi. Kukaan yksi yritys tai yksi taho ei ole vastuussa Bitcoin-maksuista. Kukin maksutapahtuma lähetetään vertaisverkkoon, jossa se todennetaan. Kuka tahansa voi liittää tietokoneensa Bitcoin-verkkoon todentamaan maksutapahtumia. Maksutapahtumat todennetaan menetelmällä, joka vaatii paljon laskentatehoa. Järjestelmän huijaaminen vaatisi hyökkääjältä enemmän laskentatehoa kuin mitä on kaikkilla verkossa olevilla rehellisillä osallisilla. Bitcoin-verkkoon hyökkääjän on ainakin teoriassa mahdollista luoda itselleen rahaa, jota hänellä ei oikeasti ole. Käytännössä hyökkääjällä pitäisi silloin olla käytössään enemmän laskentatehoa kuin koko muulla maailmalla. (1.)

Lohkoketju on arvokkaan tiedon tallentamiseen perustuva järjestelmä. Bitcoinin tapauksessa arvokas tieto on varallisuus ja valuutta.

3)

Bitcoinin tekniikka herätti maailmalla mielenkiintoa. Nakamoto pohti hajautettua valuuttaa suunnitellessaan, että olisi suunnitellut laajan hajautetun ohjelmointikielen lohkoketjuille, mutta päättikin tehdä yksinkertaisemman järjestelmän, sillä ymmärsi tekniikan olevan kokeellinen ja haastava luonteeltaan. (2.)

Siitä Vitalik Buterin keksi tehdä Ethereumin, ohjelmointikielen ja sovellusalustan, jonka sovellukset pyörivät lohkoketjussa (3.) Kenties joku parempi lähde tähän. Ehkä se alkuperäinen Buterinin paperi?

Klassiset autot ovat arvokkaita. Niistä voidaan käydä satojen tuhansien (miljoonien?) eurojen kauppoja. Autoharrastajat kokevat, että autot ovat maansa kulttuurille merkityksellisiä, sillä kussakin klassisessa autossa on viitteitä aikansa kulttuurista.

Klassiset autot liittyvät myös liikennöinnin alaan. Liikennöinnin alallakin on paljon tulevaisuuden sovelluksia, joissa voitaisiin hyödyntää lohkoketjuja. Koska työn aihe liittyy oleellisesti myös liikennöintiin, tässä työssä saadut tulokset voivat päteä myös liikennöinnin alalla.

Autoja huoltaessa huoltotietojen tallennus olisi haviteltava ominaisuus. Sekä tavallisille autoille, että klassisille autoille. Tavallisilla autoilla huoltotietoja voivat hyödyntää vakuutusyhtiöt, sekä auton omistaja, jotta pysyisi ajan tasalla auton kunnosta ja muistaisi huoltotoimenpiteet.

Klassisia autoja on huollettava, sillä ne voivat olla kymmeniä vuosia vanhoja, ja osat kuluvat väistämättä ajaessa. Lohkoketjuun tallennettua tietoa on vaikea muokata jälkeenpäin. Jos klassisen auton huoltotiedot, käyttötilastot (kilometrimittari) tallennettaisiin lohkoketjuun, auton omistaja voisi vakuuttaa huutokaupoissa ostajan autonsa arvosta.

Nykyään älykkäät sopimukset eivät vielä ole lyöneet tulille (paranna). Lohkoketjutekniikkaa tutkitaan vielä yliopistoissa (viite). Älykkäitä sopimuksia ei juurikaan käytetä kaupallisesti.

Tässä työssä oli tarkoitus tehdä käytännön sovellus. Työn perimmäinen tarkoitus oli luoda konkreettinen kuva siitä, mihin älykkäitä sopimuksia voitaisiin käyttää tulevaisuudessa.

Työssä pohditaan liiketoimintamalli ja käyttötapaukset lohkoketjutekniikkaa hyödyntävälle sovellukselle. Tavoite oli luoda Ethereum -älykäs sopimus, jota voitaisiin hyödyntää klassisen auton elinkaaren seurantaan.

Toinen osa työtä oli luoda web-käyttöliittymä, jolla lohkoketjuun tehtäviä merkintöjä voi tehdä ja tarkastella. Tässä työssä kuvaillaan työn suunnittelusta, totetuksesta ja lopputuloksesta.

# SUUNNITTELU JA KÄYTTÖTAPAUKSET

Auton omistajalla on käytössään auton elinkaarta seuraava älykäs sopimus.

Yhdelle autolle tehdään yksi älykäs sopimus.

Ajatuksena verkkosovelluksessa on, että kuka tahansa ihminen maailmassa voisi mennä auton älykkään sopimuksen kotisivuille tarkastelemaan auton nykytilaa, siinä olevia kohokohtamerkintöjä ja historiaa.

Ajatuksena oli tehdä yleisiä merkintöjä. Merkinnöistä käytetään nimeä kohokohta, eli highlight.

Ulkopuoliset käyttäjät voivat tehdä verkkosovelluksella kohokohtapyyntöjä (highlight request) älykkääseen sopimukseen. Kohokohtapyyntönä voi olla esimerkiksi auton huoltaminen korjaamolla tai asiantuntija-arvio auton autenttisuudesta alkuperälleen. Kohokohtapyynnön tekijä voi pyytää pienen rahallisen korvauksen vaivannäöstä.

Merkintöjen tavoite on vakuuttaa tuleva ostaja auton arvokkuudesta. Kukin kohokohtapyyntö maksaisi auton omistajalle useita euroja tai kymmeniäkin euroja per merkintä, ne maksavat itsensä takaisin moninkertaisesti, mikäli auton ostaja päättää tarjota huutokaupassa suuren summan autosta, johtuen arvosta jonka merkinnät tuovat autolle.

Lisäksi kuka tahansa ihminen maailmassa voisi tehdä tarjouksen autosta. Kaikki tarjoukset ovat julkisia, ja kaikki pystyvät tarkastelemaan niitä. Auton omistaja voi hyväksyä tarjouksen, jolloin hän voi toimittaa auton ostajalle. Sopimuksen omistajuusoikeus siirtyy autokaupan yhteydessä.

## Lohkoketjun hyöty sovelluksessa

Mitä hyötyä on juuri lohkoketjusovelluksella? Miksei tätä voisi tehdä ihan perinteisenä sovelluksena, jossa merkinnät tallennetaan tietokantaan?

Yksi etu lohkoketjussa on tietysti se, että lohkoketjusovelluksessa ei ole lainkaan palvelinkustannuksia. Tavallisessa palvelussa palvelinkustannukset täytyy tienata takaisin liiketoiminnalle, kuten mainoksilla. Tämä palvelu on luonteeltaan sellainen, että sitä ei välttämättä käytetä kovin usein. Liiketoiminnan tekeminen voi olla haastavaa

Toinen etu on lohkoketjumerkintöjen pitkäikäisyys. Klassisen auton elinkaari voi olla vuosikymmeniä, eivätkä kovin monet yritykset ole olemassa niin kauaa. Jos tavallinen palvelu menee nurin, tietokannassa olevat merkinnät katoaisivat sen mukana. Lohkoketju on paljon pitkäikäisempi, ja koska kopiot tiedoista ovat läsnä jokaisella ketjua käyttävällä tietokoneella, tiedot eivät edes voi tuhoutua lopullisesti.

Lohkoketjussa kaikki maksutapahtumat ovat julkisia ja läpinäkyviä. Läpinäkyvyys sopii tämän sovelluksen käyttötapauksiin hyvin. Jos kaikki auton elinkaaressa tulleet tapahtumat ovat julkisia, on huijausten ja väärinkäytösten tekeminen kannattamatonta auton omistajalle. Sovelluksella tehtyjä merkintöjä tarkkailevat voivat tehdä tutkimusta, ja havaita jos auton elinkaaren merkinnöissä esiintyy väärää tietoa tai ristiriitoja. Sosiaalisen median aikakautena tieto petollisesta auton omistajasta leviää nopeasti ihmiseltä ihmiselle. Maineen menetyksen pelko voi olla suuri. Mikäli tällainen sovellus on oikeasti käytössä tulevaisuudessa, autonomistajat varmasti ymmärtävät, että on kannattavaa pysyä rehellisenä.

# OHJELMISTO

Laita kaikkiin jokin viite mistä sen ohjelman saa ladattua

## Työn kannalta oleellinen ohjelmisto

### Ethereum-alusta

Ethereumiin kuuluu lohkoketju, joka toimii ikään kuin tietokantana. Sekä virtuaalikone, joka on älykkäiden sopimusten suoritusympäristö.

### Ethereum-virtuaalikone

Virtuaalikone on tietokoneella pyörivä. Se sisältää Ethereum-koodiajoympäristön. Se tulkitsee Ethereum-koodin (Onko se ”ethereum-koodia” silloin kun se on käännetty ja sitä tulkitaan? Tulkitaanko sitä edes, vai miksi sitä kutsutaan, kun se kone tekee juttuja?) ja tekee loogisia operaatioita. Joku äijä on tehnyt teknisen spesifikaation siitä, mitä käskyjä Ethereum-virtuaalikoneen kuuluu toteuttaa (Linkkaa spesifikaatioon. Löytyyköhän sitä mistään? Jos ei löydy, niin sano sori.). Täten Ethereum-virtuaalikoneen voi kuka tahansa toteuttaa millä tahansa kielellä. Virtuaalikone on tehty esimerkiksi Go-kielellä (ks. GoEthereum), sekä Pythonilla, myös testrpc on javascriptillä tehty (Onko tämä totta? Ymmärsinkö?)

### Solidity-kääntäjä

Solidity-kääntäjä kääntää Solidty-kielen Ethereum tavukoodiksi. Ethereum-virtuaalikone osaa tulkita tavukoodia. Solidity-kieli on hyvin yksinkertainen ja helposti opittava. Se muistuttaa Javascriptiä.

### Web3-kirjasto

Web3 on javascript-kirjasto, joka osaa puhua paikalliselle ethereum-solmulle

Web3 osaa lähettää tietoa. Se osaa myös näyttää vastauksia. Mitäs muuta kaikkea se osaa?

## Ohjelmointityössä käytetty ohjelmisto

Tässä kuvaillaan ohjelmistoa, jota päädyttiin käyttämään lopullisessa työssä

### Browser Solidity

Browser solidity on selaimessa käytettävä helppokäyttöinen kääntäjä. Javascriptin suorituskyvystä johtuen browser solidity voi olla hidas. Sen etuna on, että se toimii kaikilla tietokoneilla ilman ylimääräistä ohjelmien asennusta. Käyttöliittymä on hyvä ja selkeä.

### TestRPC

TestRPC on node-palvelinsovellus, joka on tehty mukailemaan hyvin tarkasti Ethereum-lohkoketjun toimintaa.

Maksutapahtumien todentaminen tapahtuu testrpc:ssä hyvin nopeasti.

TestRPC:n asentaminen on helppoa linuxilla. Windowsilla asentaminen on valitettavasti tuskallista.

### Truffle-sovelluskehys

Truffle on Ethereum -älykkäiden sopimusten kehittämistä varten luotu sovelluskehys, jonka avulla

### Meteor

Meteor on sovelluskehys, joka yhdistää node-palvelimen mongodb -tietokannan hyvin helppokäyttöisesti.

## Muu työhön liittyvä ohjelmisto

### GoEthereum (geth)

Go-kielellä ohjelmoitu Ethereum-virtuaalikone.

Paikallinen geth-solmu oli painajaismaisen hidas. Kuulin testrpc-ohjelmasta (onko parasta kutsua sitä ”ohjelmaksi”? Palvelin? Sovellus?).

### Ethereum Wallet

Valitettavasti Ethereum Wallet ei toimi testrpc:n kanssa, koska testrpc hyödyntää vain osan kaikista rajapinnoista, joita geth-solmussa on. EthereumWallet käyttää rajapintaa signAndSendTransaction, mitä ei ole implementoitu testrpc:ssä. Ethereum Walletin käyttö, ja sen avulla sopimuksen kehittäminen ja testaus olisi muuten helppoa ja nopeaa, mutta tällä hetkellä käyttö ei onnistu.

Ethereum Walletissa on Solidity-kääntäjä, jonka suorituskyky on parempi kuin browser solidityssä. Tätä varten Ethereum Wallet voi olla hyödyllinen.

# TYÖN KULKU

Tässä vaiheessa voisin ehkä kertoa, mitä ohjelmistoja päädyin käyttämään.

1. tutustuttiin ethereumiin käyttämällä ethereum-wallettia. Pystytettiin testiverkko
2. Todettiin että testiverkko oli liian hidas. Tuskailtiin testrpc:n kanssa. Onnistunkohan nielemään ylpeyteni ja sanomaan, etten vieläkään tiedä, mitä kaikkea ohjelmistoa testrpc:n asentaminen vaatii?
3. Ryhdyin käyttämään trufflea. Automaattisia testejä tuli tehtyä. Lopulta totesin, että automaattiset testit olivat liian työläitä. Trufflea käytettiin depottamiseen.
4. Tuliko nyt sitten Meteor? Tein Meteorilla harjoitustyön ennen kuin aloin tekemään ethereum-sovellusta
5. Tästä alkoi pitkä taival
6. Selitä miten päädyit ratkaisemaan sen iteroinnin ja koko tietorakenteen sisällön haun. Kritisoi ratkaisutapaa järkevissä määrin. Älä ole mulkvisti.
7. Yhdessä vaiheessa pyrin tekemään kohokohdista alisopimuksia. Ei kelvannut, koska tekniset rajoitteet. Ei saanut string-funktiota.

Tehtiin alkuvaiheessa automaattisia testejä. Automaattisten testien koodaus oli kumminkin hyvin työlästä, joten niiden kehittämisestä luovuttiin myöhemmin.

1. Lopuksi tein käyttöliittymän kuntoon. Tulin työn aikana kokeilleeksi dappstyles.styl-tyyliä. Todettiin, että jonkun toisen tekemän tyylin käyttö on haastavaa, jos ei ole pohjalla kokemusta css- tai styl-tyylien tekemisestä. Päädyttiin tekemään oma yksinkertainen css-tyyli.

# LOPPUTULOKSET

Kuvaile tänne mitä tuli tehtyä, miten toimii ja miten onnistui.

Laita kuvakaappaus valmiista työstä.

## Toteutuneet käyttötapaukset

Ehkä tähän voisi lyhyesti listata käyttötapaukset? Esitellään ne käyttötapaukset jossain muualla.

* Auton omistaja näkee sovelluksessa eri toimintoja kuin muut käyttäjät.
* Kun käyttäjä käynnistää sovelluksen, hän pystyy vaihtamaan käyttäjätiliään. Kun käyttäjätiliä vaihtaa, käyttöliittymä päivittyy välittömästi
* Autentikaatio tapahtuu lohkoketjussa. Kuka tahansa voisi esimerkiksi kutsua verkkosivuilla funktiota, jolla saisi auton omistajuuden (gainOwnership). Tällöin lohkoketjussa havaitaan, ettei kutsun lähettänyt tili olekaan auton omistaja, jolloin mitään ei tapahdu.
* Omistaja pystyy lisäämään kohokohtia omaan autoonsa. Kohokohdassa on tietueina muun muassa merkinnän lisäämisaika, viesti, merkinnän tekijä.
* Omistaja voi poistaa kohokotamerkintöjä.
* Muut tilit pystyvät lisäämään kohokohtapyyntöjä. He voivat pyytää pienen summan rahaa palkkioksi auton omistajalta siitä vaivannäöstä, että he tekivät kohokohtapyynnön.
* Omistaja pystyy hyväksymään kohokohtapyynnön, jolloin se lisätään auton kohokohtien joukkoon. Samalla omistajalta siirtyy pyydetty summa kohokohtapyynnön tekijälle.
* Omistaja voi kieltäytyä kohokohtapyynnöstä, jolloin se poistetaan.
* Muu kuin auton omistaja pystyy tekemään tarjouksen autosta.
* Tarjouksen tekijä voi myös poistaa tarjouksensa jos muuttaa mieltään.
* Auton omistaja voi kieltäytyä tarjouksesta, jolloin se poistuu tarjousten listalta.
* Omistaja pystyy hyväksymään tarjouksen, jolloin sopimus siirtyy ”omistajuus on vaihtumassa” -tilaan.
* Kun auton omistajuus on vaihtumassa, auton ostaja näkee painikkeen, jolla hän pystyy saamaan sopimuksen omistusoikeudet. Ajatuksena on, että ostaja painaa tästä painikkeesta, kun auto on toimitettu hänelle.
* Painikkeen painamisen jälkeen uusi omistaja saa kaikki käyttöoikeudet mitä edellisellä omistajalla oli. Edellinen omistaja näkee samat ominaisuudet kuin muutkin tavalliset käyttäjät.
* Jokainen yllä mainittu toiminto tallentaa sopimukseen historiatietomerkinnän (Solidity-kielessä ”event”). Merkintöjä tallentuu myös silloin kun tietueita poistetaan. Historiatietomerkintöjä voi tarkastella kuka tahansa, eikä niitä voi poistaa kukaan.

## Toteutumattomia käyttötapauksia

- Spämmäyksen estojärjestelmä. Highlight request rights. Voi antaa oikeuksia tietyille tilille tehdä highlighteja. Ainoastaan ulkopuoliset käyttäjät, joilla on oikeus, voivat lähettää highlight requesteja. Strict highlight request mode. Voi ottaa pois päältä, jos auto on epäsuosittu. Jos auto saa näkyvyyttä enemmän lehdistössä tai internetissä, strict moden voi laittaa päälle.

-Roskapostin estojärjestelmä. Nykyisessä sovelluksessa yksikin käyttäjä pystyisi lähettämään tuhansia kohokohtapyyntöjä päivässä. Tätä varten kehitettiin työn aikana ominaisuus, jota voisi luonnehtia kohokohtien pyyntöoikeuksiksi (highlight request rights). Tavoitteena oli kehittää järjestelmä, jossa auton omistaja voisi antaa eri käyttäjille oikeudet lisätä kohokohtapyyntöjä. Ominaisuus olisi hyödyllinen esimerkiksi julkisuuden henkilölle, jonka auto on kirjattu sovellukseen. Julkinen auto saa paljon huomiota sosiaalisessa mediassa, jolloin roskaposti voi olla riesa auton omistajalle. Epäsuositun auton omistaja voisi laittaa asetuksen pois päältä, jolloin kuka tahansa käyttäjä voi lähettää kohokohtapyyntöjä. Yhdessä ohjelman versiossa tätä ominaisuutta kehitettiin, mutta kehitys lopetettiin, koska päätettiin, ettei käyttötapauksen demonstroiminen ollut tarpeeksi tärkeää työn tarkoituksen kannalta.

- Historiatietojen suodattaminen ja järjestely. Lopullisessa sovelluksessa kaikki historiatiedot ovat samassa listassa peräjälkeen. Olisi käytettävyyden kannalta parempi, jos listaan voisi lisätä suotimia ja siitä voisi tehdä hakuja.

* Uusien ajoneuvojen lisääminen sovelluksella. Näytä kuva siitä puolivalmiista käyttöliittymästä.
* Huoltotietolomake. Huoltotiedot voisi lisätä lomakkeella. Tämä oli haastava toteuttaa, sillä Solidityssä on keskinkertainen tuki muuttuvan pituisille tietorakenteille, kuten taulukoille. Suunniteltiin yksinkertainen lomake, jossa pystyi asettamaan huoltotietojen tilan renkaille, moottorille, sisätilalle, blaa ja blaalle. Tämä käyttötapaus olisi hyvin vakuuttavalla tavalla voinut vakuuttaa älykkäiden sopimusten hyödystä liikennöinnin alalla. Lopulta kumminkin todettiin, ettei käyttötapaus ollutkaan elintärkeä työn tarkoituksen kannalta, eikä sitä toteutettu.
* Auton myynnissä ja ostossa on joitakin tietoturva-aukkoja, jotka voisi korjata. Nykyinen järjestelmä vaatii ostajaa lähettämään koko tarjoamansa summan sopimukseen. Kymmenien tai satojen tuhansien eurojen ostossa, on epäkäytännöllistä kiinnittää niin isoja rahasummia pitkäksi aikaa. Olisi suositeltavaa, jos sekä ostaja, että myyjä lähettäisivät pienen osuuden kaupasta, esimerkiksi 5% auton hinnasta sopimukseen. Auton ostaja lähettäisi rahat tarjouspyyntöä tehtäessä, ja myyjä lähettäisi tarjouspyynnön hyväksyessä. Kun molemmilla osapuolilla on rahaa kiinnitettynä sopimukseen, molemmilla on vähemmän motiiveja yrittää petosta. Auton myyjä saisi koko kaupan rahat kun ostaja on hyväksynyt kaupan (Painamalla painikkseesta ”Olen saanut ajoneuvon. Ota sopimuksen omistajuus”)
* Auton tarjouksissa myös Solidity-kielen rajoitteet tuottivat ongelmia. Nykyisessä järjestelmässä on mahdollista saman käyttäjän tehdä useampi tarjous. Olisi parempi jos yksi käyttäjä voisi tehdä yhden tarjouksen.

# ARVIO TEKNIIKASTA

Truffle on varmasti hyödyllinen projektissa, jossa on tuhansia rivejä Solidity-koodia, sekä suurempi kehitystiimi, jossa ainakin yksi kehittäjä voi kuluttaa aikansa testien koodaamiseen. Tämän laajuisessa työssä Truffle-sovelluskehyksestä saatu hyöty jäi vähäiseksi.

Meteorilla työskentely oli nopeaa ja sujuvaa. Meteorissa hyvä ominaisuus on javascript-koodien yhdistäminen ja universaalisuus koko sovelluksessa. Mallipohjat ja mallipohjien apurifunktiot ovat helppokäyttöisiä kehittäjälle, sekä hyvin monipuolisia. Niiden avulla voi tehdä hyvin monipuolisia sovelluksia.

Tässä sovelluksessa, sekä kaikissa muissakin Ethereum-sovelluksissa Meteorin reaktiivisuus on hyvin hyödyllinen.

Ainoa ongelma Meteor-kehityksessä oli hidas alkuunpääsy Ethereum-integraation kanssa. Internetissä on melko vähän opiskelumateriaalia.

Solidity-kieli jätti toivomisen varaa. Kielessä on paljon teknisiä rajoitteita. Osa rajoitteista johtuu ohjelmointikielen vähäisestä kehityksestä. Solidity on tällä hetkellä versiossa 0.4.7. Osa solidity-kielen rajoitteista johtuu Ethereum-virtuaalikoneen rajoitteista.

Solidity-kielessä on ollut menneissä versioissa turvallisuusongelmia. Turvallisuusongelmat ovat vaarantaneet koko Ethereum-alustan.

Sovelluksen pitäisi kyetä hyppäämään yli ne tietueet jotka on jo poistettu. Tietorakenteen ohjelmoiminen ei ole vaikeaa, mutta sen koodaaminen virheettömästi on silti aikaa vievää. Kehittäjän näkökulmasta voi olla silti turhauttavaa, että tämän kaltaisen perustoiminnallisuuden koodaamiseen on kulutettava aikaa. Vaikka mapping-tietorakenteessa on teknisiä rajoitteita, Solidity-kieleen olisi mahdollista kehittää tietorakenne, joka laajentaa mapping-tietorakenteen toiminnallisuutta, jossa tässä mainitut ongelmat voisivat olla korjattu. Vaikka tietorakenne olisikin suorituskyvyltään hidas, olisi se silti tärkeä ominaisuus, koska se lisäisi innovaatioiden määrää Ethereum-sovelluksien kehittäjien keskuudessa. Innovaatiot ovat hyvin tärkeitä Ethereumille.

Ei voi kutsua vieraassa sopimuksessa olevaa funktiota, joka palauttaa string-tyyppisen muuttujan. Kerro mistä johtuu ja miten tämä rajoitti työskentelyä.

On suunnattoman vaikeaa, että solidityssä ei ole null-käsitettä, mitä useammissa ohjelmointikielissä on. Monissa ohjelmissa haluaisi tehdä funktioita joka joko palauttaa arvon kun funktion kutsu onnistuu, mutta palauttaa null silloin kun se ei onnistu.

Ei voi hakea koko mapping-tietorakenteen kaikkia alustettuja alkioita. Kehittäjän täytyy itse kirjoittaa funktiot koko mappingin alkoiden haulle. Olisi mahdollista, että kuka tahansa ihminen maailmassa voisi tehdä kotisivun/sovelluksen, jokatoimisi ikään kuin käyttöliittymänä Classic Car Chain -sovellukselle. Mutta sovelluksessa on varta vasten tehty koodia koko mapping-tietorakenteen haulle, joka voi olla vaikea kopioida. Tämä tulee hidastamaan huomattavasti hajautettujen sovellusten kehitystä, niin kauan kuin se on olemassa.

Vaikka rajoitteet kuulostavat pieniltä, ne kumminkin rajoittavat suuresti kaikkia mahdollisia käyttötapauksia, mitä Ethereum-alustalla voisi muuten toteuttaa. Rajoitteet tulevat hidastamaan Ethereumin läpimurtoa yleiseen, maailmanlaajuiseen käyttöön. Tästä huolimatta, Soldity-kieltä kehitetään parhaillaan, ja joitakin hyvin oleellisia parannuksia on tehty viime aikoinakin, kuten (etsi se joku foorumi missä ihmiset toivoivat jotain ominaisuutta ja linkkaa dokumentaatiossa johonkin paikkaan, missä todistat, että ominaisuus on jo tehty).

Voi olla kannattavaa kertoa EtherSimistä, ja kertoa kuinka käsittämättömän hämmentävää on pysyä mukana Ethereum-ohjelmistokehityksen nykytilanteesta. Aina kaikkea uutta tehdään ja jotain heitetään roskakoriin. Vaikea löytää mikä on uusinta tietoa. Voit laittaa esimerkiksi sen uhkapelisovelluksen, jossa kehittäjä turhautui Mist/Ethereum Wallet sekoiluun, ja dumppasi koko sovelluksen.

Silti kukaan ei voi kieltää Solidity-kielen mahdollisuuksia. Koska kieli on niin yksinkertainen ja kuka tahansa jolla on ymmärrystä ohjelmoinnista voi oppia sen. Kielen helppokäyttöisyydellä ja yksinkertaisuudella on suuri merkitys Ethereum-alustan maailmanlaajuisessa läpimurrossa.

# LOPPUSANAT

Lohkoketjutekniikan tutkijat puhuvat, että historia voi jakaantua ennen lohkoketjuja olleeseen, ja lohkoketjujen jälkeiseen maailmaan, että lohkoketjut ovat keksintönä yhtä mullistava kuin internet.

Väite kuulostaa hurjalta, mutta on hyvinkin mahdollisuuksien rajoissa. Lohkoketjulla ja älykkäillä sopimuksilla on samoja ominaisuuksia mitä internetillä on (tällekin lauseelle voisi olla viite). Myös monet lohkoketjutekniikan tutkijat väittävät, että lohkoketju on yhtä merkityksellinen keksintö kuin internet. Internetistä teki mullistavan keksinnön se, että sen tekniikka oli avoin ja vapaasti käytettävissä (liitteenä ted radio hour podcast open source-jakso).

Jää nähtäväksi, tuleeko ennustuksesta totta. Elämmekö tulevaisuudessa maailmassa, jossa raha, tavaroiden ja palvelujen korvaukset liikkuvat ihmiseltä ihmiselle, ilman välikäsiä, kaikki käyttäen älykkäitä sopimuksia lohkoketjualustalla?

LÄHteet

1. <https://bitcoin.org/bitcoin.pdf>
2. <https://www.youtube.com/watch?v=3PdO7zVqOwc>
3. <http://futurethinkers.org/vitalik-buterin-ethereum-decentralized-future/>

LIITTEET

Liite 1 Lähtötietomuistio

**Tekijä** Lauri Miettinen

**Tilaaja** Oulun Ammattikorkeakoulu, tekniikan laitos

**Tilaajan yhteyshenkilö(t) ja yhteystiedot**

Janne Kumpuoja ([Janne.Kumpuoja@oamk.fi](mailto:Janne.Kumpuoja@oamk.fi))**,**

Veijo Väisänen (Veijo.Vaisanen@oamk.fi)

**Projektin nimi**

Vintage-auton elinkaaren seuranta lohkoketjutekniikalla -opinnäytetyö

**Projektin tavoitteet**

Tavoite on luoda Ethereum-lohkoketjukehitysalustan avulla älykäs sopimus, jota voitaisiin hyödyntää Vintage-auton elinkaaren seurantaan. Tavoitteena on tehdä web-käyttöliittymä, jonka avulla käyttäjä voi vuorovaikuttaa lohkoketjun kanssa. Työn tuloksista raportoidaan opinnäytetyödokumentissa.

Työssä suunnitellaan ja esitellään liiketoimintamalli, jota valmis tuote voisi käyttää.

Opinnäytetyödokumentissa kuvaillaan luotua järjestelmää ja esitellään sen taustalla oleva tekniikka. Lukijalle esitellään lyhyesti lohkoketjutekniikan perusteet ja historia.

**Projektissa käytettävä prosessimalli (mahdollinen vaihejakoon perustuva aikataulutus)**

28.9.2016: Harjoitustyön vaatimusmäärittelydokumentti oltava valmiina.

21.10.2016: Sprint 1 release. Projektin pohjimmaiset järjestelmät, sovelluskehykset ja palvelimet oltava käyttö- ja kehityskelpoisia.

16.12.2016: Sprint 2 release. Sovelluksesta oltava valmiina versio, joka on esittelykelpoinen (minimum effort). Kirjallisessa työssä oltava kuvauksia projektin etenemisestä.

10.3.2017: Sprint 3 release. Sovelluksen minimituote (minimum viable product) oltava esittelykepoinen.

28.4.2017: Sprint 4 release. Kirjallinen työ, sovellus ja liiketoimintamalli on oltava valmiina. Pidetään lopetuspalaveri.

**Projektissa käytettävät menetelmät ja teknologiat**

Ethereum; scrum-menetelmä; Github –versionhallinta; Meteor, tai muu Javascript-pohjainen sovelluskehys.

**Projektin alustava aikataulu** Työ alkaa 13.9.2016, ja päättyy 30.4.2017