Lohkoketjuteknologia aiheinen portfolio

Joonas Saarela

Haaga-Helia ammattikorkeakoulu

Tradenomi tutkinto

Opinnäytetyö

2023

Tiivistelmä

Tekijä(t)
Joonas Saarela

Tutkinto
Tradenomi, Tietojenkäsittely

Raportin/Opinnäytetyön nimi
Lohkoketjuteknologia aiheinen portfolio

Sivu- ja liitesivumäärä
35 + 1

Tämän portfolio tyyppisen opinnäytetyön tarkoituksena on esitellä tekijän osaamista Ethereumin lohkoketjuteknologiasta ja sen soveltamisesta erilaisiin käyttötapauksiin. Projektien avulla tekijä osoittaa kykenevänsä hyödyntämään Ethereumin lohkoketjuteknologiaa digitaalisen taiteen kokoelmien luomisessa ja liiketoiminnassa.

Tämä opinnäytetyö käsittelee kahta projektia, joissa molemmissa hyödynnetään Ethereumin lohkoketjuteknologiaa. Ensimmäinen projekti on digitaalinen taidekokoelma, joka toteutetaan käyttäen NFT-standardeja. NFT:t (Non-Fungible Token) mahdollistavat digitaalisten taideteosten tokenisoinnin, mikä antaa omistajalle oikeuden hallita kyseistä taideteosta ja myydä sitä.

Toinen projekti käsittelee kuukausitilausten tokenisointia Ethereumin lohkoketjussa. Tämä tarkoittaa kuukasitilausten muuttamista lohkoketjussa oleviksi "tokeneiksi", joita voidaan käyttää esimerkiksi tilauksen seurannassa. Tämä mahdollistaa esimerkiksi tilauksen omistajanvaihdoksen tilauksen ollessa vielä voimassa.

Asiasanat
Lohkoketju, lohkoketjuteknologia, tokenisointi, Ethereum, Solidity, Smart contract, älysopimus

Sisällys

1 Johdanto

Lohkoketjuteknologia on vielä suhteellisen uutta, eivätkä sen tuomat mahdollisuudet ole vielä monien ihmisten tiedossa. Monet ihmiset, jotka ovat kuulleet lohkoketjuteknologiasta, ovat todennäköisesti kuulleet niistä kryptovaluuttojen- tai viime vuosien aikana kovassa suosiossa olleiden NFT:eiden (Non-Fungible Token) yhteydessä. Näistä molemmista on saattanut jäädä huono mielikuva, sillä lohkoketjuteknologian ollessa niin uutta, ei sitä käsitteleviä lakeja ole ollut ennen kuin vasta parin edellisen vuoden aikana, mikä on mahdollistanut kaikenlaisien huijareiden ja rikollisten hyödyntää tätä teknologiaa muiden kustannuksella.

Lohkoketjuteknologiaa voidaan kuitenkin hyödyntää monessa asiassa jo nykypäivänä, kuten esimerkiksi rahoitusalalla, toimitusketjujen hallinnassa ja äänestyksissä. Sen avulla voidaan luoda hajautettuja ja läpinäkyviä järjestelmiä, jotka tarjoavat turvallisuutta, luotettavuutta ja tehokkuutta eri toimialoilla. Lohkoketjuteknologia on kehittyvä ja lupaava alue, jolla on valtavasti potentiaalia muuttaa eri toimialojen toimintatapoja.

Tämä portfolio tyyppinen opinnäytetyö toimii esityksenä tekijän osaamisesta Ethereumin lohkoketjuteknologiasta ja sen soveltamisesta erilaisiin käyttötapauksiin. Opinnäytetyössä esitellään kaksi projektia, joissa molemmissa hyödynnetään Ethereumin lohkoketjuteknologiaa. Opinnäytetyön tavoitteena on esitellä tekijän kyky hyödyntää Ethereumin lohkoketjuteknologiaa erilaisten projektien avulla. Opinnäytetyön avulla lukija saa kattavan kuvan tekijän osaamisesta Ethereumin lohkoketjuteknologiasta, sekä sen soveltamisesta digitaalisien taidekokoelmien luomisessa ja liiketoiminnallisissa prosesseissa.

Ensimmäinen projekti keskittyy digitaalisen taidekokoelman toteuttamiseen käyttäen NFT-standardeja (Non-Fungible Token). NFT-tokenit mahdollistavat digitaalisten taideteosten tokenisoinnin, mikä puolestaan antaa taideteoksen omistajalle oikeuden hallita kyseistä taideteosta ja mahdollisuuden myydä sitä. Tämän projektin kautta tekijä osoittaa kykenevänsä hyödyntämään Ethereumin lohkoketjuteknologiaa digitaalisen taiteen kokoelmien luomisessa. Opinnäytetyössä käydään läpi tämän projektin toteutusprosessia, tarvittavia teknisiä ratkaisuja sekä saavutettuja tuloksia.

Toinen projekti keskittyy kuukausitilausten tokenisointiin Ethereumin lohkoketjussa. Tämä tarkoittaa kuukausitilausten muuttamista lohkoketjussa oleviksi "tokeneiksi", joita voidaan käyttää esimerkiksi tilauksen seurannassa. Tämä mahdollistaa tilauksen omistajanvaihdoksen, kun tilaus on vielä voimassa. Tämän projektin avulla tekijä osoittaa kykenevänsä soveltamaan Ethereumin lohkoketjuteknologiaa liiketoiminnallisten prosessien kehittämisessä. Opinnäytetyössä käsitellään tämän projektin suunnittelua, toteutusta ja saavutettuja tuloksia.

Opinnäytetyön rakenne etenee seuraavasti: johdannon jälkeen esitellään Ethereumin lohkoketjuteknologiaa yleisellä tasolla, minkä jälkeen keskitytään syventymään NFT-standardeihin ja tokenisointiin. Sen jälkeen esitellään ensimmäinen projekti, digitaalisen taidekokoelman toteutus, ja sen jälkeen toinen projekti, kuukausitilausten tokenisointi. Lopuksi opinnäytetyö päättyy pohdintaan, jossa arvioidaan ja pohditaan opinnäytetyöprosessia eri näkökulmista.

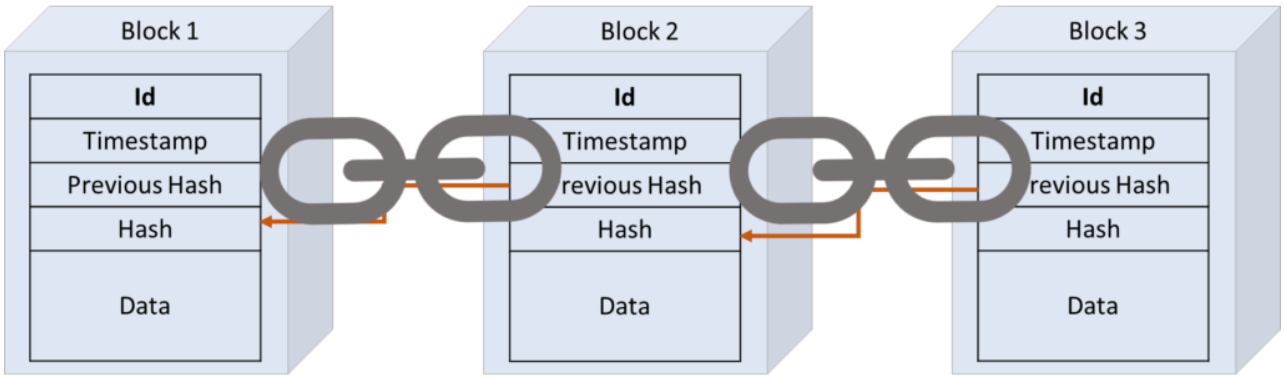
2 Lohkoketjuteknologia

Tässä luvussa käsitellään laajasti lohkoketjuteknologiaa, sen erilaisia sovelluksia ja erityisesti Ethereumin lohkoketjua. Lisäksi tarkastellaan Ethereumin tarjoamia tokeneita, älysopimuksia, Solidity-ohjelmointikieltä ja Non-Fungible Tokeneita (NFT). Luvussa käsitellään myös erilaisia NFT-standardeja, sekä Merkle-puuta, jotka ovat olennaisia käsitteitä Ethereumin lohkoketjuteknologiassa.

Tämä luku tarjoaa perustiedot lohkoketjuteknologiasta yleisesti, esittelee Ethereumin ja sen keskeiset ominaisuudet, sekä tutustuttaa lukijan Ethereumin tarjoamiin tokeneihin, älysopimuksiin, NFT-varoihin ja niihin liittyviin standardeihin. Lisäksi luvussa käsitellään Solidity-ohjelmointikieltä ja Merkle-puun käyttöä Ethereumin älysopimuksissa.

2.1 Lohkoketju

Tietojärjestelmät ovat kehittyneet valtavasti viime vuosikymmeninä ja yksi niistä teknologioista, joka on noussut esiin, on lohkoketju. Lohkoketju on digitaalisen tiedonjakamisen ja hallinnan teknologia, joka perustuu nimensä mukaan ketjumaisesti toisiinsa linkitettyihin tietoa sisältäviin lohkoihin. Se on luotettava ja turvallinen tapa hallita digitaalisia tietoja ja varoja ilman keskitettyä hallintoa tai luottamushenkilöitä (Kuva 1). (Hayes 2023.)



Kuva 1. Lohkoketju käytännössä (Paiementor 2020.)

Lohkoketjut koostuvat lohkoista, jotka sisältävät tietoja ja tiedot ovat sidottuja toisiinsa matemaattisten algoritmien avulla. Jokainen lohko sisältää myös uniikin tunnisteen, joka mahdollistaa tietojen

jäljittämisen ja varmennuksen. Uusia tietoja ei voi muokata tai poistaa aiemmin luoduista lohkoista, mikä takaa tiedon eheyden ja luotettavuuden. (IBM s.a.)

Lohkoketjulla on useita sovelluskohteita, kuten digitaaliset valuutat, tiedon jakaminen, äänestysjärjestelmät ja luottamusvälineiden hallinta. Esimerkiksi digitaalisilla valuutoilla, kuten Bitcoinilla, on oma lohkoketju-pohjainen järjestelmänsä, joka mahdollistaa nopeat ja turvalliset rahansiirrot ilman välikäsiä. (Rodeck & Curry 2022.)

Lohkoketjun avulla voidaan myös parantaa tietoturvaa ja läpinäkyvyyttä erilaisissa liiketoiminnoissa. Esimerkiksi äänestysjärjestelmissä lohkoketjulla voidaan varmistaa, että äänestysprosessi on luotettava ja että äänten laskenta tapahtuu oikeudenmukaisesti. (Forbes 2022.)

Lohkoketjun kehitys ja käyttöönotot ovat edelleen alkuvaiheessa, mutta se on lupaava teknologia tulevaisuuden digitaalisten liiketoimintojen ja hallinnan tarpeisiin. Se tarjoaa mahdollisuuden luoda luotettavia ja turvallisia järjestelmiä ilman keskitettyä hallintoa ja lisää läpinäkyvyyttä erilaisissa toiminnoissa. (Forbes 2022.)

2.2 Erilaisia lohkoketjuteknologioita

Lohkoketjuteknologia on kehittynyt nopeasti viime vuosina ja nykyään on olemassa useita erilaisia lohkoketjuteknologioita, jotka ovat erikoistuneet erilaisiin tarpeisiin ja sovelluskohteisiin. Tärkeimmät ja tunnetuimmat lohkoketjuteknologiat ovat Bitcoin, Ethereum, ja Solana.

Bitcoin on ensimmäinen ja tunnetuin digitaalinen valuutta, joka perustuu lohkoketjuteknologiaan. Se tarjoaa turvalliset ja nopeat rahansiirrot ilman välikäsiä ja mahdollistaa pääsyn kansainväliselle rahamarkkinalle. Bitcoinilla on myös suuri yhteisö ja sitä käytetään laajalti maailmanlaajuisesti. (Bitcoin 2023.)

Ethereum on hajautettu, avoimen lähdekoodin lohkoketjualusta, joka mahdollistaa kehittäjille hajautettujen sovellusten (Decentralized Applications) rakentamisen ja käyttöönoton älysopimusten avulla. Ethereumin älysopimus toiminnallisuus mahdollistaa kehittäjien luoda itseään suorittavia sopimuksia, jotka suoritetaan automaattisesti, kun tiettyjä ehtoja täyttyy. Ethereum mahdollistaa myös hajautettujen autonomisten organisaatioiden (Decentralized Autonomous Organization) kehittämisen, joita ohjaa koodi keskitetyn auktoriteetin sijaan. (Ethereum 2023.)

Ethereumin oma kryptovaluutta on Ether (ETH), jota käytetään siirtomaksujen ja laskennallisien palveluiden maksamiseen verkossa. Ethereumin hajautetun luonteen ansiosta se on myös vastustuskykyinen sensuurille ja käyttökatkoille, koska verkkoa ylläpidetään suurella määrällä itsenäisiä solmuja (nodes) keskitetyn yksikön sijaan, mikä estää yhden solmun tai yksittäisen toimijan

sensuroimasta tiettyjä tapahtumia tai lohkoja. Lisäksi Ethereumin verkon lähdekoodi on avoin ja hajautetusti kehitetty, mikä tekee sen vaikeaksi muuttaa tai manipuloida yksipuolisesti. (Ethereum 2023.)

Solana on lohkoketjualusta, joka erikoistuu nopeaan ja tehokkaaseen tiedonkäsittelyyn. Se tarjoaa mahdollisuuden luoda hajautettuja sovelluksia, kuten pörssit ja digitaaliset valuutat, jotka vaativat nopeita ja tehokkaita tietojenkäsittelyjä. Solana on myös erittäin skaalautuva, mikä mahdollistaa suurten käyttäjämäärien käsittelyn ilman suorituskykyongelmia. (Solana 2023.)

2.3 Ethereumin tokenit

Ethereumin tokenit ovat digitaalisia omaisuuseriä, jotka on rakennettu Ethereumin lohkoketjun päälle. Nämä tokenit voivat edustaa erilaisia asioita, kuten rahaa, omaisuutta tai muita arvoja. Tässä projektissa tokenit edustavat oikeutta maksumuurin takana olevaan materiaaliin, kuten esimerkiksi suoratoistopalvelun kuukausitilaus.

Fungible tokenit ovat tokeneita, joilla on samat ominaisuudet kuin toisilla samanlaisilla tokeneilla. Yksinkertaisesti sanottuna, ne ovat vaihdettavissa keskenään, eikä niillä ole yksilöllistä arvoa. Esimerkkinä fungible-tokenista on Ethereumin Ether (ETH), joka on digitaalinen valuutta, jolla on tietty arvo, mutta yksittäinen Ether-token ei ole erilainen kuin toinen Ether-token. Ether on Ethereumin lohkoketjun tapahtumissa käytettävä digitaalinen valuutta.

Non-fungible tokenit (NFT) ovat sen sijaan ainutlaatuisia tokeneita, joilla on yksilöllinen arvo. Nämä tokenit edustavat usein ainutlaatuisia esineitä tai teoksia, kuten digitaalisia taide- tai peliesineitä. Yksittäinen NFT-token on ainutlaatuinen eikä vaihdettavissa suoraan muiden NFT-tokenien kanssa, koska jokaisella NFT:llä on oma ainutlaatuinen tunniste ja ominaisuudet. (Ethereum 2023.)

2.4 Ethereumin älysopimukset

Ethereumin älysopimukset ovat koodinpätkiä, jotka on tallennettu Ethereumin lohkoketjuun ja jotka mahdollistavat automaattisten sopimusten toteuttamisen ilman välikäsiä. Älysopimukset ovat itsenäisiä ja toimivat ennalta määritettyjen ehtojen mukaisesti. Ne voivat hallita varoja, tietoja tai muita resursseja, ja niitä voidaan käyttää monenlaisiin sovelluksiin.

Älysopimukset toimivat lohkoketjuteknologian periaatteella, mikä tarkoittaa, että ne ovat hajautettuja ja läpinäkyviä. Niitä ei voi muuttaa tai manipuloida yksipuolisesti, mikä tekee niistä luotettavia ja turvallisia.

Älysopimusten avulla voidaan toteuttaa esimerkiksi digitaalisia sopimuksia, äänestyksiä, rahansiirtoja, peliohjelmistoja ja paljon muuta. Ethereumin älysopimukset ovat avoimia kaikille ja kehittäjät voivat käyttää niitä rakentaakseen uusia sovelluksia. (Zapotochnyi 2022.)

2.5 Solidity

Solidity on oliopohjainen korkean tason ohjelmointikieli, jota käytetään älysopimusten toteuttamiseen. Älysopimukset ovat ohjelmia, jotka hallitsevat Ethereum-tilan sisällä olevien tilitapahtumien käyttäytymistä. Solidity on kaarisulkuohjelmointikieli, joka on suunniteltu Ethereum Virtual Machine (EVM) -alustaa varten. Solidity on saanut vaikutteita C++, Python ja JavaScript -ohjelmointikielistä. (Solidity 2023.)

Solidity on staattisesti tyypitetty ja tukee muun muassa periytymistä, kirjastoja ja monimutkaisia käyttäjän määrittelemiä tyyppejä. Solidityn avulla voit luoda älysopimuksia esimerkiksi äänestämiseen, joukkorahoitukseen, sokkohuutokauppoihin ja usean allekirjoituksen lompakoihin (multi-signature wallet). (Solidity 2023.)

2.6 Non-Fungible Token, eli NFT

NFT (non-fungible token) tarkoittaa digitaalista hyödykettä, joka on ainutlaatuinen ja käyttää lohkoketjuteknologiaa varmistaakseen sen omistajuuden ja autenttisuuden. NFT:t voivat olla esimerkiksi digitaalisia taideteoksia, GIF-animaatioita tai jopa tweettejä, jotka on muutettu digitaaliseen muotoon ja tallennettu lohkoketjuun. Jokaisella NFT:llä on oma koodinsa, jonka avulla sen omistajuus voidaan todistaa ja sen ainutlaatuisuus varmistaa. NFT:t ovat tulleet tunnetuksi taide- ja viihdealalla, sillä ne tarjoavat uuden tavan myydä ja omistaa digitaalisia teoksia. (Ethereum 2023b.)

NFT-markkina on kasvanut räjähdysmäisesti viime vuosien aikana, ja se on herättänyt paljon keskustelua ja kiinnostusta eri aloilla. Taiteilijat, muusikot, urheilijat ja jopa brändit ovat hyödyntäneet NFT-teknologiaa luodakseen ainutlaatuisia digitaalisia teoksia ja tuotteita, joita voi ostaa ja myydä. NFT:t tarjoavat taiteilijoille ja luojille uuden tulonlähteen, sillä he voivat saada suoran korvauksen jokaisesta myydystä teoksesta ilman perinteisiä välikäsiä. (Leech 2022.)

2.7 ERC721-standardi

ERC721 on Ethereumin lohkoketjussa käytetty standardi, joka määrittelee tavan luoda ainutlaatuisia, ei-vaihdettavia digitaalisia hyödykkeitä (NFT). Tämä tarkoittaa sitä, että jokainen ERC721-standardin mukainen NFT on ainutlaatuinen ja sitä ei voi vaihtaa toisen NFT:n kanssa. (Ethereum 2023b.)

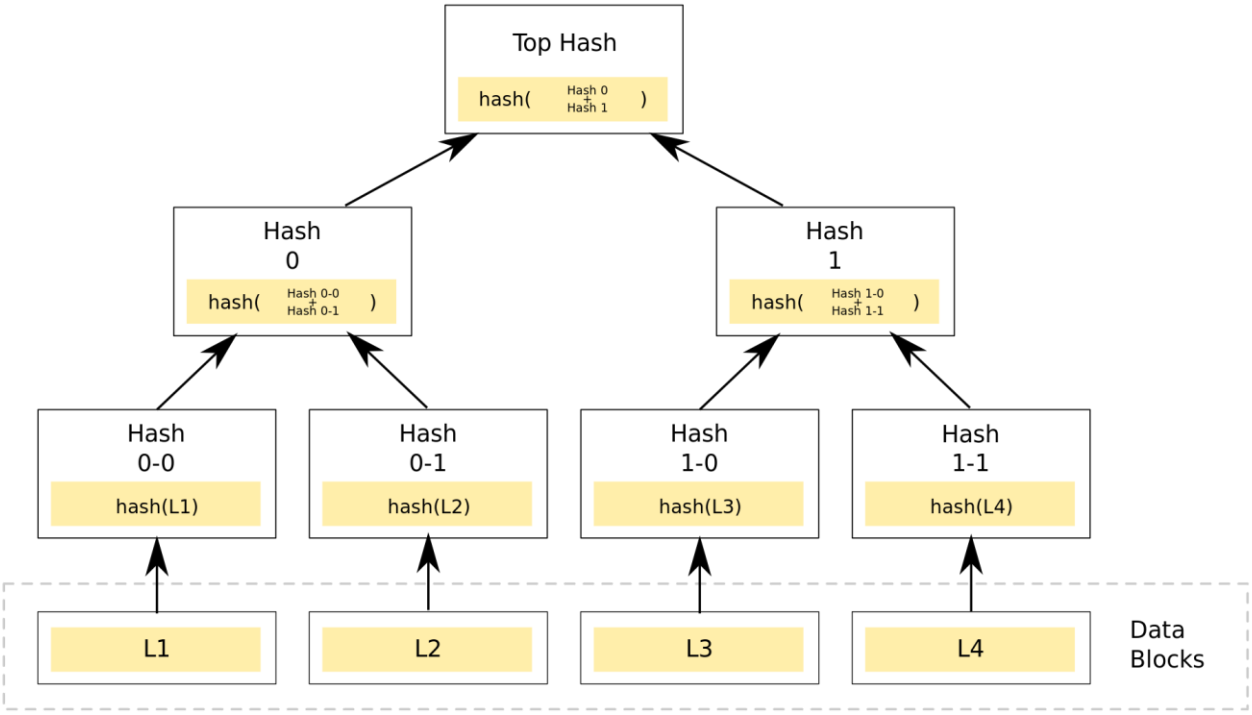
ERC721-standardi määrittelee, miten NFT-luokkaa hallitaan ja miten yksittäisiä NFT:itä luodaan, siirretään ja poistetaan. Jokaisella NFT:llä on oma tunnisteensa, jota käytetään sen tunnistamiseen ja seuraamiseen lohkoketjussa. Tämä mahdollistaa NFT:iden ainutlaatuisuuden ja omistajuuden varmistamisen. (Ethereum 2023b.) ERC721-standardia käytetään laajalti eri aloilla, kuten taide- ja peliteollisuudessa, joissa se mahdollistaa digitaalisten hyödykkeiden myynnin ja omistamisen. (Ethereum 2023b.)

2.8 ERC721A- ja ERC721ACustom standardit

ERC721A-standardi on paranneltu toteutus IERC721-standardista, joka mahdollistaa useiden tokenien luomisen lähes samaan hintaan kuin yhden. (Azuki 2023.) **ERC721ACustom**-standardi on ERC721A-standardista mukauttamani versio, jossa tokenId muuttuja alkaa arvosta 1 arvon 0 sijaan.

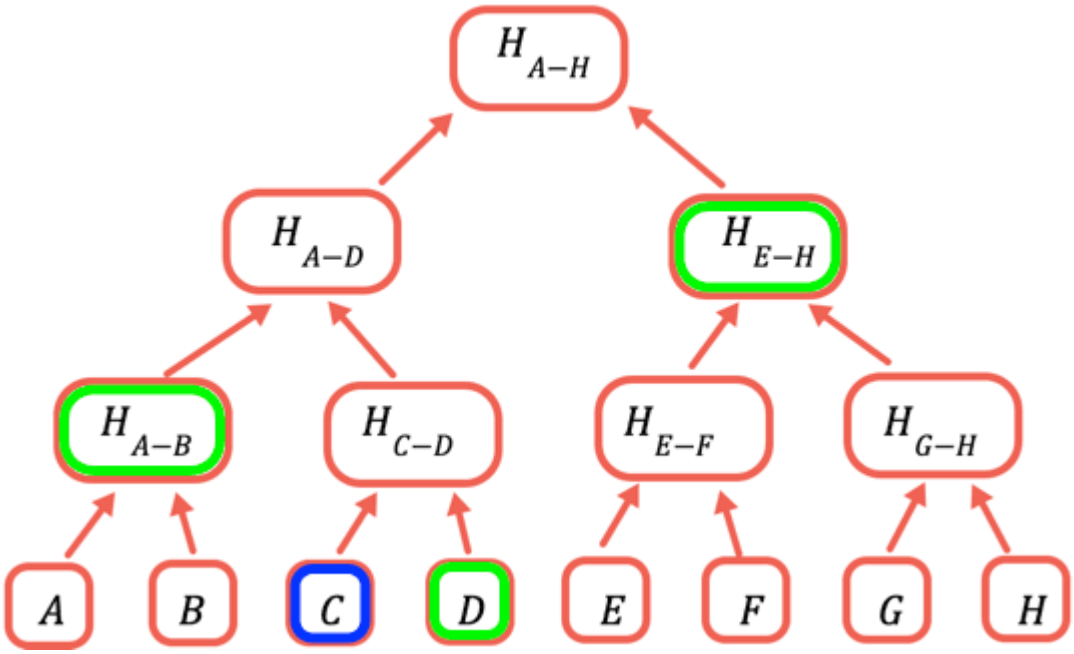
2.9 Merkle-puu

Merkle-puu on tietorakenne, joka käyttää hajautusfunktiota (tunnetaan myös nimellä "hash-funktio") rakentaakseen puurakenteen, joka auttaa tarkistamaan, onko tieto muuttunut tai onko se aito. Merkle-puu koostuu monista hash-arvoista, jotka edustavat alkuperäistä tietoa, ja jokainen hash-arvo on laskettu kahden lähtötiedon hash-arvosta. Puun ylin hash-arvo tunnetaan yleensä nimellä "juuri hash-arvo" ja se edustaa kaikkia alkuperäisiä hash-arvoja (Kuva 2).



Kuva 2. Merkle-puu (Wikipedia 2023)

Merkle-puun arvojen todistamiseen käytetään Merkle-todistetta. Todistaaksemme jonkin Merkle-puun arvon, annamme kaikki Merkle-puun hash-arvot, jotka on yhdistettävä todistettavan arvon kanssa juuren saamiseksi (Kuva 3). (Ethereum 2023c.)



Kuva 3. arvon C todistamiseen tarvittaisiin D, H(A-B) ja H(E-H) hash-arvot. (Ethereum 2023c.)

3 Älysopimus taiteilijan NFT projektiin

Kesällä 2022 sain tilaisuuden osallistua mielenkiintoiseen projektiin älysopimuskehittäjän roolissa. Projektin tavoitteena oli luoda kaksi erilaista NFT-taidekokoelmaa ja niille omat älysopimukset. Lähdin projektiin mukaan ilman aiempaa kokemusta älysopimusten toteuttamisesta. Projektissa keskityttiin erityisesti digitaalisten, uniikkien taideteosten myyntiin ja haluttiin mahdollistaa tietyille henkilöille etuoikeus ostaa teoksia ennen muita hieman edullisempaan hintaan. Työskentelin yhdessä web-kehittäjän kanssa, joka toteutti verkkosivut taideteosten myyntiä varten. Projektiryhmämme oli monikansallinen ja kommunikaatio tapahtui englanniksi. Tässä luvussa esittelen yhden kahdesta toteutetusta älysopimuksesta. Olen muuttanut koodissa näkyviä nimiä, jotka voitaisiin yhdistää valmiiseen tuotteeseen.

3.1 Taustatietoa projektista

Olin 2022 kesällä mukana yhdessä projektissa älysopimuskehittäjänä toteuttamassa älysopimuksia NFT-taide projektiin. Projektin tavoitteena oli toteuttaa kaksi erilaista NFT-taidekokoelmaa ja niihin molempiin omat älysopimukset. Esittelen tässä vain toista näistä kahdesta älsyopimuksesta, niiden ollessa suurimmaksi osaksi rakenteeltaan samanlaisia. Olen muuttanut tähän demonstraatioon älysopimuksessa näkyviä nimiä, jotka voitaisiin yhdistää valmiiseen tuotteeseen.

Projekti alkoi minun kohdallani Solidity-ohjelmointikieleen- ja älysopimuksiin tutustumalla. Parin viikon opiskelujakson jälkeen määriteltiin, mitä kaikkia ominaisuuksia älysopimuksella tulee olla. Tässä projektissa tavoitteena oli luoda älysopimus, joka mahdollistaa digitaalisien, uniikkien taideteoksien myynnin. Lisäksi haluttiin, että tietyt valitut henkilöt pääsevät ostamaan taideteoksia ennen muita, ja hieman halvempaan hintaan.

Työskentelin suurimmaksi osaksi yhdessä yhden web kehittäjän kanssa, joka toteutti verkkosivut taideteoksien myyntiin. Muun projektiryhmän kanssa tapasimme vähintään joka toinen päivä, ja kävimme yhdessä läpi, missä vaiheessa projektin eri osat ovat. Projektiryhmämme koostui eri puolilta maapalloa olevista ihmisistä ja kaikki kommunikointi tapahtui englanniksi.

3.2 Toteutus

Valitsin tähän projektiin pohjaksi mukautetun ERC721ACustom-standardin, jotta taideteoksia voisi ostaa useita kerralla pienemmillä kuluilla, ja ensimmäinen taideteos saa id:n ”1”. Aluksi loin uuden, tyhjän Solidity-tiedoston, ja määritin SPDX-Lisenssin, sekä käytettävän Solidity-version. Seuraavaksi lisäsin kaikki tarvittavat kirjastot (Kuva 4).

- **Ownable.sol** on sopimusmoduuli, joka tarjoaa perustason käyttöoikeuksien valvontaa, jossa on tili (omistaja), jolle voidaan myöntää yksinoikeus tiettyihin toimintoihin. (OpenZeppelin 2023a.)
- **ReentrancyGuard.sol** on sopimusmoduuli, joka auttaa estämään toistuvia kutsuja funktiolle. (OpenZeppelin 2023b.)
- **Pausable.sol** on sopimusmoduuli, jonka avulla voidaan toteuttaa hätäpysäytysmekanismi, jonka valtuutettu tili voi käynnistää. (OpenZeppelin 2023b.)
- **Strings.sol** mahdollistaa uint-arvojen muuttamisen ASCII-merkkijonoarvoiksi. (OpenZeppelin 2023c.)
- **Merkleproof.sol** mahdollistaa Merkle-puiden (hash-puiden) verifioinnin älysopimuksessa (OpenZeppelin 2023d.)

```
1   // SPDX-License-Identifier: MIT
2
3   pragma solidity ^0.8.16;
4
5   import "./ERC721ACustom.sol";
6   import "@openzeppelin/contracts/access/Ownable.sol";
7   import "@openzeppelin/contracts/security/ReentrancyGuard.sol";
8   import "@openzeppelin/contracts/security/Pausable.sol";
9   import "@openzeppelin/contracts/utils/Strings.sol";
10  import "@openzeppelin/contracts/utils/cryptography/MerkleProof.sol";
11
```

Kuva 4. SPDX-Lisenssi, Solidity-versio ja Kirjastot

Kirjastojen lisäyksen jälkeen, loin itse älysopimuksen ja lisäsin vaadittavia muuttujia (Kuva 5).

- **whitelistMerkleroot** on Merkle-puun juuri, jonka arvo asetettiin älysopimuksen julkaisuvaiheessa. Tämän muuttujan avulla voidaan tarkistaa, onko ostajan Ethereum-osoite niin sanotulla ”whitelistillä”, eli ennakkomyyntiin oikeuttavalla listalla.
- **baseURI** osoittaa osoitteeseen, mistä taideteoksien varsinaiset kuvat sekä ominaisuudet säilytetään. Tämänkin muuttujan arvo asetettiin älysopimuksen julkaisuvaiheessa.
- **treasury** on Ethereum-osoite, johon myytyjen taideteoksien tuotot menevät. Tämänkin muuttujan arvo asetettiin älysopimuksen julkaisuvaiheessa.
- **adminAddresses** on lista Ethereum-osoitteita, joilla on oikeus käyttää rajoitettuja funktioita.
- **whitelistClaimed** on avain-arvo tietorakenne, jossa Ethereum-osoitteet toimivat avaimina, ja ostettujen tokenien määrä arvona. Tällä tarkkaillaan ennakkomyynnistä ostettujen taideteostokenien määrää jokaista Ethereum-osoitetta kohden.

- **publicClaimed** on avain-arvo tietorakenne, jossa Ethereum-osoitteet toimivat avaimina, ja ostettujen tokenien määrä arvona. Tällä tarkkaillaan julkisesta myynnistä ostettujen taideteos-tokenien määrää jokaista Ethereum-osoitetta kohden.

```
12  contract NFTContract is ERC721A, Pausable, ReentrancyGuard, Ownable {
13      using Strings for uint256;
14      bytes32 public whitelistMerkleRoot;
15
16      string public baseURI;
17      address private treasury;
18
19      mapping(address => bool) public adminAddresses;
20      mapping(address => uint256) public whitelistClaimed;
21      mapping(address => uint256) public publicClaimed;
```

Kuva 5. Älysopimus ja muuttujat

Loin älysopimukseen käyttöoikeuksia rajoittavan määritteen, jota voidaan käyttää funktioiden käytön rajoittamiseen. Määrite tarkistaa, onko funktiota kutsuva Ethereum-osoite älysopimuksen omistaja tai omistaako tämä admin-oikeudet. Kuvassa näkyvä **msg.sender** on älysopimusta/funktiota kutsuvan käyttäjän Ethereum-osoite (Kuva 6).

```
23      modifier onlyAdminOrOwner() {
24          bool isAdmin = false;
25          if (adminAddresses[msg.sender] == true) {
26              isAdmin = true;
27          }
28          if (msg.sender == owner()) {
29              isAdmin = true;
30          }
31          require(isAdmin == true, "Not an admin");
32          _;
33      }
```

Kuva 6. onlyAdminOrOwner-määrite

Seuraavaksi loin taideteoksien myyntiin liittyvistä tiedoista ”olion”. Solidityn ”struct” rakenne mahdollistaa useiden data tyyppien yhdistämisen yhdeksi kokonaisuudeksi (Kuva 7). (Alchemy 2023.)

- **presaleActive** määrittää, onko ennakkomyynti käynnissä.
- **publicSaleActive** määrittää, onko julkinen myynti käynnissä.
- **whitelistPrice** määrittää ennakkomyynnin taideteoksien hinnan.
- **price** määrittää julkisen myynnin hinnan.
- **maxSupply** määrittää taideteostokenien maksimimäärän, eli montako niitä tulee myyntiin.
- **maxPublicMint** määrittää, montako taideteosta julkisesta myynnistä saa ostaa maksimissaan yhtä Ethereum-osoitetta kohden.
- **maxWhitelistMint** määrittää, montako taideteosta ennakkomyynnistä saa ostaa maksimissaan yhtä Ethereum-osoitetta kohden.

```
35      struct SaleConfig {
36          bool presaleActive;
37          bool publicSaleActive;
38          uint256 whitelistPrice;
39          uint256 price;
40          uint256 maxSupply;
41          uint256 maxPublicMint;
42          uint256 maxWhitelistMint;
43      }
44      SaleConfig public saleConfig;
```

Kuva 7. SaleConfig-rakenne

Seuraavaksi loin älysopimukselle ”constructorin”. Älysopimuksen ”constructorissa”, eli alustusfunktiossa määritetään vaadittavien muuttujien arvot ja tokenin tuleva nimi, sekä lyhenne. Alustusfunktion parametrina olevien muuttujien arvot syötetään älysopimuksen julkaisuvaiheessa, tai testauksen aikana testiverkkoon julkaistaessa (Kuva 8).

```
46      constructor(    infinite gas 4170000 gas
47          uint256 _maxSupply,
48          uint256 _maxPublicMint,
49          uint256 _maxWhitelistMint,
50          bool _presaleActive,
51          bool _publicSaleActive,
52          bytes32 _wlMerkleRoot,
53          address _treasury
54      ) payable ERC721A("NFT1", "NFT1") {
55          saleConfig.maxSupply = _maxSupply;
56          saleConfig.price = 0.05 ether;
57          saleConfig.whitelistPrice = 0.03 ether;
58          saleConfig.presaleActive = _presaleActive;
59          saleConfig.publicSaleActive = _publicSaleActive;
60          saleConfig.maxPublicMint = _maxPublicMint;
61          saleConfig.maxWhitelistMint = _maxWhitelistMint;
62          whitelistMerkleRoot = _wlMerkleRoot;
63          treasury = _treasury;
64      }
```

Kuva 8. Älysopimuksen "constructor", eli alustusfunktiossa

Kun vaadittavat muuttujat ja alustusoperaatiot olivat valmiit, ryhdyin luomaan julkisen myynnin mahdollistavaa funktiota. Julkisella myynnillä tarkoitetaan, että kuka tahansa Ethereum-osoitteen omaava voi ostaa taideteoksen, jos vain omistaa riittävästi Etheriä (Kuva 9).

- **require**-lausunnot määrittävät tietyt ehdot funktion toteutumiselle. Jos kaikki require-lausuntojen ehdoista eivät täyty, funktion suoritus keskeytetään.
- **transfer**-funktiolla siirretään ostajan maksama summa Etheriä aiemmin määritettyyn Ethereum-osoitteeseen.
- **publicClaimed**[msg.sender] -mappingiin päivitetään käyttäjän ostama tilaustokenien määrä.
- **_safeMint**-funktio luo x määrän uusia taideteos-tokeneita haluttuun Ethereum-osoitteeseen. Parametrina syötetään osoite, johon tokenit luodaan sekä luotavien tokenien määrä.

```
101     function publicMint(uint256 _amount) external payable nonReentrant whenNotPaused {    infinite gas
102         require(saleConfig.publicSaleActive == true, "Public sale inactive");
103         require(
104             totalSupply() + _amount <= saleConfig.maxSupply,
105             "Cannot mint more than max supply"
106         );
107         require(
108             msg.value >= (saleConfig.price * _amount),
109             "Insufficient funds"
110         );
111         require(
112             publicClaimed[msg.sender] + _amount <= saleConfig.maxPublicMint,
113             "Mint exceeds max mint per address"
114         );
115 
116         payable(treasury).transfer(msg.value);
117         unchecked {
118             publicClaimed[msg.sender] += _amount;
119         }
120         _safeMint(msg.sender, _amount);
121     }
```

Kuva 9. publicMint, eli julkinen myynti

Seuraavaksi lähdin toteuttamaan ”whitelist” -myynnin, eli ennakkomyynnin mahdollistavaa funktiota. Whitelist-myynnin ollessa käynnissä, vain tietyt ennalta määritellyt Ethereum-osoitteet pääsevät ostamaan taideteos-tokeneita muita aikaisemmin ja hieman edullisempaan hintaan. Tässä projektissa ennakkomyyntiin oikeutettujen Ethereum-osoitteiden määrä oli niin suuri, ettei olisi ollut kustannustehokasta tallentaa osoitteita suoraan älysopimukseen. Mikäli olisin syöttänyt kaikki Ethereum-osoitteet suoraan älysopimukseen, olisi Ethereumin lohkoketjun palvelumaksu ollut noin 2 ETH, mikä oli sen aikaisella Etherin hinnalla noin 2500 euroa.

Kuluja vähentääksemme päädyimme Merkle-puu ratkaisuun. Tämä mahdollisti sen, että kaikkien ennakkomyynnin Ethereum-osoitteiden suoraan älysopimukseen tallentamisen sijaan tallensimme vain kaikista ennakkomyynnin Ethereum-osoitteista luodun Merkle-puun juuri hash-arvon älysopimukseen. Projektin web-kehittäjä toteutti Merkle-puun, sekä rajapinnan, josta löytyy jokainen Ennakkomyyntiin oikeutettu Ethereum-osoite, sekä osoitetta vastaava Merkle-todiste. Merkle-todisteen avulla voidaan todistaa, että syötetty Ethereum-osoite on osa aiemmin luotua Merkle-puuta.

Älysopimuksessa ennakkomyyntifunktiossa ”whitelistMint” syötetään parametreina ostettavien taideteos-tokenien määrä, sekä Merkle-todiste. Käyttäjän ei itse tarvitse syöttää Merkle-todistetta, vaan sen hoitaa käyttöliittymä. Käyttöliittymä hakee rajapinnasta käyttäjän Ethereum-osoitetta vastaavan Merkle-todisteen, mikäli sellainen löytyy. Jos käyttäjän Ethereum-osoitetta vastaavaa validia Merkle-todistetta ei löydy, ei käyttäjä pysty ostamaan taideteos-tokeneita ennakkomyynnistä.

Ensin funktiossa tarkistetaan, onko ennakkomyynti alkanut. Jos ennakkomyynti on alkanut, generoidaan käyttäjän Ethereum-osoitteesta hash-arvo ja syötetään se Merkle-todisteen ja Merkle-puun

juuri hash-arvon kanssa Merkle-todisteen tarkistamisfunktioon. Mikäli käyttäjä on oikeutettu ennakkomyyntiin, tarkistetaan seuraavaksi, onko tämä jo ostanut maksimi sallitun määrän taideteos-tokeneita ennakkomyynnistä. Lopuksi vielä tarkistetaan, että käyttäjä on maksamassa taideteos-tokeneista tarpeeksi Etheriä, ja ettei ostettavien tokenien määrä ylitä maksimimäärää, mitä tokeneita on kokonaisuudessaan myynnissä. Näiden tarkastuksien jälkeen, siirretään ensin käyttäjän maksama Ether ennalta määritettyyn Ethereum osoitteeseen. Sitten päivitetään käyttäjän ostamien taideteos-tokenien määrä aiemmin luotuun **whitelistClaimed** -avain-arvo tietorakenteeseen. Lopuksi luodaan ostettu määrä uusia taideteos-tokeneita käyttäjän Ethereum-osoitteeseen **_safeMint**-funktiolla (Kuva 10).

```
66      function whitelistMint(uint256 _amount, bytes32[] calldata _merkleProof) external payable nonReentrant whenNotPaused
67      {
68          require(saleConfig.presaleActive == true, "Presale inactive");
69
70          bytes32 leaf = keccak256(abi.encodePacked(msg.sender));
71          require(
72              MerkleProof.verify(_merkleProof, whitelistMerkleRoot, leaf),
73              "Not whitelisted"
74          );
75          require(
76              whitelistClaimed[msg.sender] + _amount <=
77                  saleConfig.maxWhitelistMint,
78              "Whitelist mint exceeded"
79          );
80
81          require(
82              msg.value >= _amount * saleConfig.whitelistPrice,
83              "Insufficient funds"
84          );
85          require(
86              totalSupply() + _amount <= saleConfig.maxSupply,
87              "Cannot mint more than max supply"
88          );
89
90          payable(treasury).transfer(msg.value);
91          unchecked {
92              whitelistClaimed[msg.sender] += _amount;
93          }
94          _safeMint(msg.sender, _amount);
95      }
96
```

Kuva 10. whitelistMint-, eli ennakkomyynti-funktio

Saatuani taideteos-tokenien ostamisen/luomisen mahdollistavat funktiot valmiiksi, lähdin toteuttamaan älysopimuksen muuta toiminnallisuutta. Käyttöliittymän toteutuksen helpottamiseksi loin **isWhitelisted**-funktion, joka tarkistaa, onko syötetty Ethereum-osoite ennakkomyyntilistalla, ja palauttaa vastauksen boolean muodossa. Tämän funktion avulla voidaan tarkistaa, onko käyttäjä ennakkomyyntilistalla jo käyttäjän yhdistäessä Ethereum-lompakkonsa myyntiverkkosivulle ja muuttaa käyttöliittymän näkymää sen perusteella (Kuva 11).

```
119     function isWhitelisted(bytes32[] calldata _merkleProof, address _address)
120         external
121         view
122         returns (bool)
123     {
124         bytes32 leaf = keccak256(abi.encodePacked(_address));
125         return MerkleProof.verify(_merkleProof, whitelistMerkleRoot, leaf);
126     }
127
```

Kuva 11. isWhitelisted-funktio

Seuraavaksi loin **_baseURI-** ja **tokenURI**-funktiot. **_baseURI**-funktio palauttaa merkkijonona taideteos-tokenien metadatan säilytykseen käytettävän kansion sijainin verkossa. Tätä funktiota käytetään vain älysopimuksen sisäisesti. **tokenURI**-funktio palauttaa merkkijonona jonkin tietyn, parametrina syötetyn taideteos-tokenin metadatan sijainnin verkossa hyödyntäen **_baseURI**-funktiota ja jokaisen tokenin uniikkia **tokenId**:ta (Kuva 12)

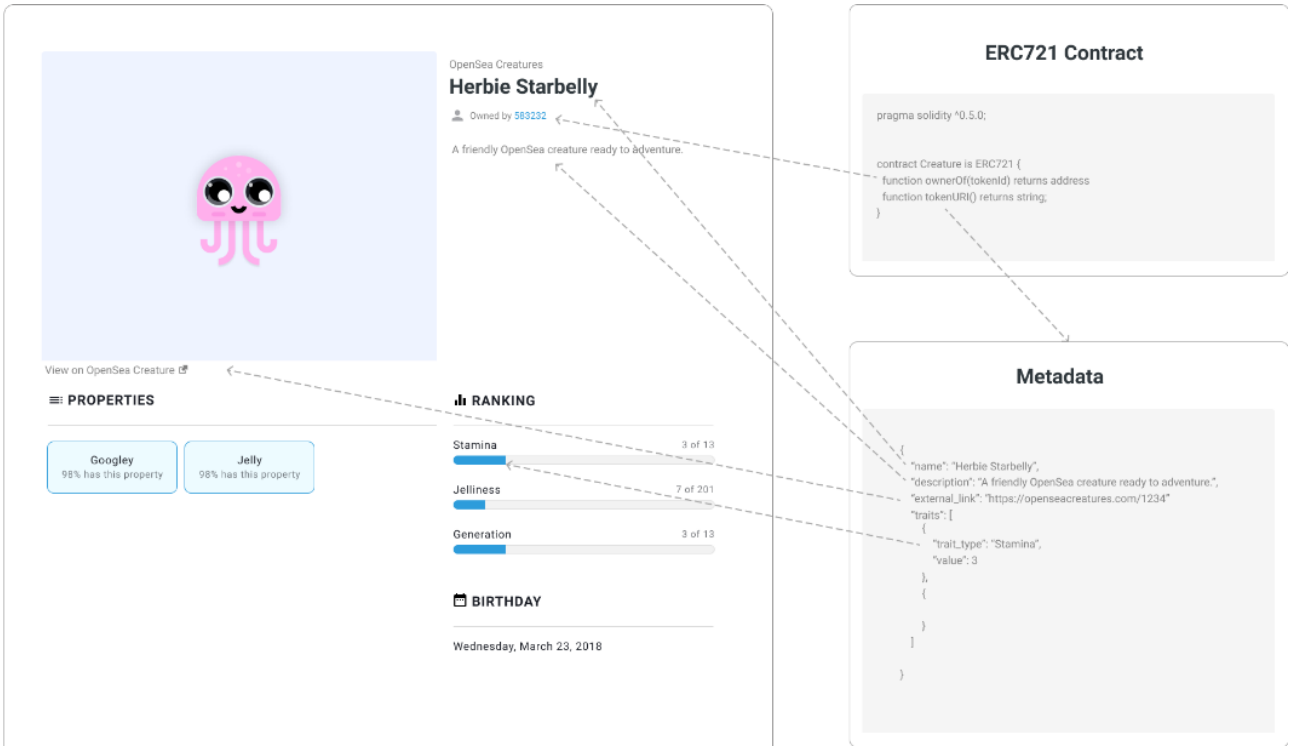
```
128     function _baseURI() internal view virtual override returns (string memory) {
129         return baseURI;
130     }
131
132     function tokenURI(uint256 tokenId)    infinite gas
133         public
134         view
135         virtual
136         override
137         returns (string memory)
138     {
139         require(
140             _exists(tokenId),
141             "ERC721Metadata: URI query for nonexistant token"
142         );
143         string memory currentBaseURI = _baseURI();
144         return
145             bytes(currentBaseURI).length > 0
146                 ? string(abi.encodePacked(currentBaseURI, tokenId.toString()))
147                 : "";
148     }
```

Kuva 12. _baseURI- ja tokenURI-funktiot

NFT-markkinapaikat käyttävät tätä **tokenURI**-funktiota jokaisen NFT:n metadatan noutamiseen. Metadatan ansiosta digitaalisille omaisuuksille, kuten NFT:t, voidaan antaa lisäominaisuuksia, kuten esimerkiksi nimi, kuvaus ja kuva. Ilman metadatan tuomia lisäominaisuuksia, on jokainen NFT yleensä **tokenId**:ta lukuunottamatta identtinen (Kuvat 13 ja 14). (OpenSea 2023.)

```
{
  "description": "Friendly OpenSea Creature that enjoys long swims in the ocean.",
  "external_url": "https://openseacreatures.io/3",
  "image": "https://storage.googleapis.com/opensea-prod.appspot.com/puffs/3.png",
  "name": "Dave Starbelly",
  "attributes": [ ... ]
}
```

Kuva 13. Esimerkki metadatan rakenteesta (OpenSea 2023.)



Kuva 14. Metadatan hyödyntäminen käytännössä (OpenSea 2023.)

Lopuksi lähdin toteuttamaan järjestelmänvalojille rajoitettuja funktioita. Näitä funktioita voivat käyttää vain järjestelmänvalvojan oikeudet omaavat käyttäjät, sillä funktioilla voidaan esimerkiksi muuttaa taideteos-tokenien hintaa. Nämä funktiot mahdollistavat myös ongelmatilanteen tullessa älysopimuksen kriittisimpien funktioiden keskeyttämisen, suurempien vahinkojen välttämiseksi.

- **setAdminAddresses**-funktiolla voidaan antaa parametrina syötetylle listalle Ethereum-osoitteita järjestelmänvalvojan oikeudet (Kuva 15).

```
152     function setAdminAddresses(address[] calldata _wallets)
153         external
154         onlyAdminOrOwner
155     {
156         for (uint256 i = 0; i < _wallets.length; i++) {
157             adminAddresses[_wallets[i]] = true;
158         }
159     }
```

Kuva 15. setAdminAddresses-funktio

- **removeAdminAddresses**-funktiolla voidaan poistaa parametrina syötetyn Ethereum-osoitteiden listan admin oikeudet (Kuva 16).

```
161     function removeAdminAddresses(address[] calldata _wallets)
162         external
163         onlyAdminOrOwner
164     {
165         for (uint256 i = 0; i < _wallets.length; i++) {
166             adminAddresses[_wallets[i]] = false;
167         }
168     }
```

Kuva 16. removeAdminAddresses-funktio

- **setBaseURI**-funktiolla voidaan määrittää taideteos-tokenien metadatakansion sijainti verkossa (Kuva 17).

```
170     function setBaseURI(string memory _newBaseURI) public onlyAdminOrOwner {
171         baseURI = _newBaseURI;
172     }
```

Kuva 17. setBaseURI-funktio

- **setPrice**-funktiolla voidaan muuttaa julkisen myynnin hintaa (Kuva 18).

```
175     function setPrice(uint256 _newPrice) external onlyAdminOrOwner {
176         saleConfig.price = _newPrice;
177     }
```

Kuva 18. setPrice-funktio

- **setPreSalePrice**-funktiolla voidaan myyntää ennakkomyynnin hintaa (Kuva 19).

```
178     function setPreSalePrice(uint256 _newPrice) external onlyAdminOrOwner {
179         saleConfig.whitelistPrice = _newPrice;
180     }
```

Kuva 19. setPreSalePrice

- **burnToken**-funktiolla voidaan poistaa olemassa olevia tokeneita pysyvästi. Vain käyttäjä, joka omistaa tokenin, voi poistaa sen (Kuva 20).

```
182     function burnToken(uint256 _tokenId) external onlyAdminOrOwner {
183         address owner = ERC721A.ownerOf(_tokenId);
184         require(owner == msg.sender, "Not the owner of this token");
185         super._burn(_tokenId);
186     }
```

Kuva 20. burnToken-funktio

- **pauseContract**-funktiolla voidaan keskeyttää älysopimuksen whenNotPaused-määritteen omaavien funktioiden käyttö (Kuva 21)

```
188     function pauseContract() public onlyAdminOrOwner {
189         _pause();
190     }
```

Kuva 21. pauseContract-funktio

- **unpauseContract**-funktiolla voidaan jatkaa keskeytettyjen funktioiden käyttö (Kuva 22).

```
192     function unpauseContract() public onlyAdminOrOwner {
193         _unpause();
194     }
```

Kuva 22. unpauseContract-funktio

- **setMaxSupply**-funktiolla voidaan muuttaa taideteos-tokeneiden maksimi myyntimäärää (Kuva 23).

```
196     function setMaxSupply(uint256 _newMaxSupply) external onlyAdminOrOwner {
197         saleConfig.maxSupply = _newMaxSupply;
198     }
```

Kuva 23. setMaxSupply-funktio

- **setPublicSale**-funktiolla voidaan aktivoida julkinen myynti, jolloin publicMint-funktio muuttuu käytettäväksi (Kuva 24).

```
200     function setPublicSale(bool _status) external onlyAdminOrOwner {
201         saleConfig.publicSaleActive = _status;
202     }
```

Kuva 24. setPublicSale-funktio

- **setPreSale**-funktiolla voidaan aktivoida tai deaktivoida ennakkomyynti, jolloin whitelistMint-funktion käytettävyys muuttuu (Kuva 25).

```
208     function setPreSale(bool _status) public onlyAdminOrOwner {
209         saleConfig.presaleActive = _status;
210     }
```

Kuva 25. setPreSale-funktio

- **activatePresale**-funktiolla voidaan aktivoida ennakkomyynti, jolloin whitelistMint-funktio muuttuu käytettäväksi (Kuva 26).

```
204     function activatePreSale() external onlyAdminOrOwner {
205         setPreSale(true);
206     }
```

Kuva 26. activatePreSale-funktio

- **switchToPublicPhase**-funktiolla vaihdetaan ennakkomyynti vaiheesta julkisen myynnin vaiheeseen. Tällöin whitelistMint-funktio poistetaan käytöstä ja publicMint-funktio tulee käyttöön (Kuva 27).

```
212     function switchToPublicPhase() external onlyAdminOrOwner {
213         saleConfig.presaleActive = false;
214         saleConfig.publicSaleActive = true;
215     }
```

Kuva 27. switchToPublicPhase-funktio

- **setWhitelistMerkleRoot**-funktiolla voidaan muuttaa älysopimukseen tallennettua Merkle-puun juuri hash-arvoa (Kuva 28).

```
217     function setWhitelistMerkleRoot(bytes32 _newWhitelistMerkleRoot)
218         external
219         onlyAdminOrOwner
220     {
221         whitelistMerkleRoot = _newWhitelistMerkleRoot;
222     }
```

Kuva 28. setWhitelistMerkleRoot-funktio

- **setTreasuryAddress**-funktiolla voidaan muuttaa älysopimukseen tallennettua Ethereum-osoitetta, johon taideteos-tokenien myynnin tuotot menevät (Kuva 29).

```
224     function setTreasuryAddress(address _treasury) external onlyAdminOrOwner {
225         treasury = _treasury;
226     }
227 }
```

Kuva 29. setTreasuryAddress-funktio

Älysopimusta testattiin koko kehityksen aikana ja valmis älysopimus puskettiin Goerli-verkkoon, mikä on yksi Ethereumin testiverkoista. Projektin web-kehittäjä integroi älysopimuksen taideteosten myyntiin käytettävän verkkosivun kanssa, ja testasimme yhdessä, että kaikki funktiot toimivat. Kun älysopimus oli testattu kokonaan ja todettu toimivaksi, puskimme sen Ethereumin pääverkkoon, joka tunnetaan nimellä Mainnet. Lopuksi syötimme älysopimukseen kaikki alustavat arvot ja integroimme sen verkkosivun kanssa toimivaksi, ja olimme valmiit taideteosten myyntiin.

3.3 Tulos

Projektin lopputuloksena toteutin kaksi toimivaa, lähes samanlaista älysopimusta taiteilijan NFT-taide projektiin. Projektin taideteosten myynti onnistui ongelmitta ja kaikki halukkaat saivat ostettua digitaalisia taideteoksia. Projektin aikana toteuttamani älysopimukset ovat edelleen Ethereumin lohkoketjulla ja toimivat hyvin, eivätkä todennäköisesti poistu sieltä koskaan. Lähdin projektiin mukaan käytännössä ilman mitään tietoa Ethereumin lohkoketjusta, ja projektin päätyttyä osasin toteuttaa älysopimuksia kustannustehokkaasti monenlaisiin tarpeisiin. Kehityin projektin aikana myös englannin kielen puhumisessa todella paljon, kun kaikki kommunikointi ryhmän kesken tapahtui englanniksi. Myös työskenteleminen projektiryhmässä tuli hyvin tutuksi, ja erityisesti kommunikaation tärkeys projektiryhmässä tuli hyvin esille. Alkuun oli vaikeaa selittää joitain teknisiä asioita projektiryhmän muille jäsenille, joille teknologia oli vähemmän tuttu, mutta loppua kohden opin kertomaan asiat selkeämmin niin, että kuka tahansa voisi ymmärtää mistä puhun.

Projektin aikana intohimoni ohjelmointia kohtaan heräsi uudelleen, ja sain lisävarmistusta siitä, että tämä on se juttu mitä haluan tulevaisuudessakin tehdä. Kohtasin projektin aikana ongelmia lähes päivittäin, mikä piti mielenkiintoani yllä. Ihan projektin alussa minusta tuntui, että minun osaamiseni ei riitä tähän projektiin, mutta projektiryhmän kannustaminen sai minut jatkamaan ja jossain kohtaa sain yhtäkkiä niin sanotun ”ahaa elämyksen”, ja teknologia ei vaikuttanutkaan niin vaikealta enää. Projekti oli kokonaisuudessaan melko rankka. Kävin projektin aikana päivisin töissä ja illat menivät kokonaan projektin parissa. Olen kuitenkin iloinen, että lähdin projektiin mukaan. Projektin aikana saamani kokemus ja tietotaito ovat erittäin tärkeitä tulevaisuuteni kannalta. Sain projektiryhmältäni hyvää palautetta koko projektin ajan, ja kaikki vaikuttivat olleen tyytyväisiä työhöni. Olen myös itse tyytyväinen koko projektiryhmän työhön ja hyvään ryhmähenkeen. Kohokohtia projektissa olivat erityisesti ne hetket, kun oma toteuttamani älysopimus puskettiin Ethereumin lohkoketjuun ja kaikki toimi niin kuin pitikin.

Jätän taiteilijan ja taideteoskokoelman nimen kertomatta projektiryhmän pyynnöstä. NFT-maailmassa käydään kauppaa oikealla rahalla, ja siihen sisältyy paljon riskejä, minkä takia on yleistä,

että projektit toteutetaan nimimerkin takana. Tässäkin tapauksessa vain projektiryhmän kesken tiesimme toistemme oikeat nimet.

4 Kuukausitilausten tokenisointi Ethereumin lohkoketjuteknologian avulla

Tämän projektin tavoitteena on tuoda esille lohkoketjuteknologian mahdollisuuksia tilausten tokenisoinnin avulla. Tokenisointi on prosessi, jossa muutetaan jokin oikean maailman omaisuuserä tai omistusoikeus digitaaliseksi tokeniksi lohkoketjuun. Tämän projektin aiheena on tilausten tokenisointi, eli suoratoistopalveluiden ja muiden kuukausimaksu-periaatteella toimivien palveluiden muuttaminen digitaaliseksi tokeniksi, joka oikeuttaa käyttäjän kyseisen palvelun tilaukseen. Projektin tavoitteena on luoda vaihtoehtoinen tapa erilaisille tilauksille Ethereumin lohkoketjuteknologian avulla, joka mahdollistaisi osittain käytettyjen tilausten myynnin eteenpäin. Projektissa keskitytään pelkästään back-end puoleen, eli älysopimukseen.

4.1 Taustatietoa projektista

Tällä projektilla haluan tuoda esille lohkoketjuteknologian mahdollisuuksia esimerkkiprojektin muodossa. Tokenisointi on prosessi, jossa muutetaan jokin tietty oikean maailman omaisuuserä, tai omistusoikeus digitaaliseksi tokeniksi lohkoketjuun (DeJesus 2022.). Projektin aiheena on tilausten tokenisointi, mikä tässä tapauksessa tarkoittaa esimerkiksi suoratoistopalveluiden ja muiden kuukausimaksu periaatteella toimivien palveluiden muuttamista digitaaliseksi tokeniksi, joka oikeuttaa käyttäjän jonkin palvelun tilaukseen.

Tilausten tokenisointi-projektin tavoitteena on luoda vaihtoehtoinen tapa erilaisille tilauksille Ethereumin lohkoketjuteknologian avulla. Tyypillisen kiinteän kuukausihinnan sijaan asiakkaat voisivat ensin ostaa palveluntarjoajala tilaustokenin, joka on voimassa jonkin tietyn ennalta sovitun ajan, esimerkiksi 30 päivää. Tokenin ostettua käyttäjä voi vapaasti hajautetulla markkinapaikalla ostaa ja myydä osittain käytettyjä tokeneita. Tämä mahdollistaisi osittain käytettyjen tilausten myymisen eteenpäin, mikäli tilausta ei enää tarvitse loppu ajaksi. Vaihtoehtoisesti mikäli tarvitset tilauksen vain pariksi päiväksi, voisi sen ostaa toiselta käyttäjältä ilman, että joutuu maksamaan täyttä 30 päivän hintaan. Rajaan tämänkin projektin pelkästään back-end puoleen, eli älysopimukseen.

4.2 Toteutus

Aloitin projektin älysopimuksen kirjoittamisesta. Minulta löytyy jo aiempaa kokemusta Ethereumin älysopimuksien kehittämisestä ollessani kesällä 2022 mukana edellä mainitussa projektissa älysopimuskehittäjänä, joten lähdin aiemman osaamiseni pohjalta kirjoittamaan tilaustokenien ostamisen mahdollistavaa älysopimusta.

Projekti alkoi vaatimusmäärittelystä. Projektin vaatiman älysopimuksen täytyy sisältää vain pari toimintoa: funktio tilaustokeneiden ostamiseen/luomiseen, sekä tilaustokeneihin liittyvän datan (osto pvm, päättymis pvm) tallentamisen mahdollistava toiminto.

Valitsin ohjelmointiympäristöksi tähän projektiin Remixin, mikä on selaimessa toimiva kehitysympäristö. Remix on hyvä kehitysympäristö pienempiin ja yksinkertaisiin älysopimuksiin sen hyvien ja nopeiden testiominaisuuksien vuoksi. Suuremmissa ja monimutkaisemmissa älysopimuksissa ei Remix kuitenkaan ole paras vaihtoehto sen pyöriessä selaimessa. Ohjelmointikieleksi valitsin Solidityn, joka on yleisin ohjelmointikieli Ethereumin älysopimuksille.

Vaatimusten ja ohjelmointiympäristön ollessa selvillä, lähdin kirjoittamaan älysopimusta. Loin uuden Solidity-tiedoston ja nimesin sen projektin aiheen mukaan: SubscriptionToken.sol. Aluksi määritetään SPDX-lisenssi, sekä älysopimuksessa käytettävä Solidity-versio. 0.8.19 on projektia tehdessäni uusin Solidityn versio (Kuva 30).

```
1  // SPDX-License-Identifier: MIT
2
3  //Solidity version
4  pragma solidity ^0.8.19;
5
```

Kuva 30. Älysopimuksen määritys

Seuraavaksi lisään projektiin tarvittavat kirjastot. Tämän projektin tokenit pohjautuvat ERC721 standardiin, eli ne ovat Non-Fungible-Tokeneita, eli uniikkeja tokeneita. ERC721 kirjastojen lisäksi lisäsin turvallisuutta parantavat Ownable ja Pausable -kirjastot. Ownable -kirjasto mahdollistaa joidenkin älysopimuksen toimintojen rajoittamisen vain älysopimuksen omistajan käytettäväksi. Pausable -kirjasto mahdollistaa älysopimuksen toimintojen keskeyttämisen. Lopuksi lisäsin vielä Strings kirjasto, mikä mahdollistaa uint -muuttujien muuntamisen String muuttujiksi (Kuva 31).

```
1  // SPDX-License-Identifier: MIT
2
3  //Solidity version
4  pragma solidity ^0.8.19;
5
6  import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
7  import "@openzeppelin/contracts/token/ERC721/IERC721.sol";
8  import "@openzeppelin/contracts/access/Ownable.sol";
9  import "@openzeppelin/contracts/utils/Strings.sol";
10 import "@openzeppelin/contracts/security/Pausable.sol";
11
```

Kuva 21. Älysopimuksen kirjastot

Seuraavaksi loin itse älysopimuksen, ja lisäsin alkuun tarvittavia muuttujia. Alussa näkyvä baseURI on vain siltä varalta, jos tulevaisuudessa halutaan lisätä tokeneille lisäominaisuuksia metadatan avulla (Kuva 32).

```
13  contract SubscriptionToken is ERC721, Pausable, Ownable {
14      using Strings for uint256;
15      string baseURI;
16
17      struct TokenConfig {
18          uint256 tokenId;
19      }
20      TokenConfig public tokenConfig;
21
22      struct SubscriptionDetails {
23          uint256 duration;
24          uint256 tier;
25          uint256 subDate;
26          uint256 subEnds;
27      }
28
29      mapping(uint256 => SubscriptionDetails) public subscriptions;
30
31
32      constructor() ERC721("Subscription Token", "SUB") {    infinite gas 2612800 gas
33          tokenConfig.tokenId = 0;
34      }
35
```

Kuva 32. Älysopimus ja muuttujat

- **TokenConfig** -struct sisältää tällä hetkellä vain tokenId muuttujan, joka kasvaa aina, kun uusi tokeni luodaan. Näin jokaisella tokenilla on uniikki id.
- **SubscriptionDetails** -struct sisältää tietoa tilaustokenin tilauksen kestosta, tasosta, sekä tilauksen alkamis- ja päättymispäivämäärät UNIX formaatissa.
- **Subscriptions** -avain-arvo tietorakenne sisältää jokaisen luodun tokenin tiedot, ja siitä on mahdollista hakea tietoa yksittäisistä tokeneista tokenId:n avulla.
- **Constructorissa**, eli alustusfunktiossa määritetään tokenin tuleva nimi ja lyhenne, sekä asetetaan tokenId:n arvoksi alustavasti 0.

Seuraavaksi luon tilaustokeneiden ostamisen/luomisen mahdollistavan funktion. Tässä ersimerkkiprojektissa tokenin ostaminen/luominen on ilmaista testauksen helpottamiseksi (Kuva 33).

```
36      //subscription durations and tiers for this example are 15/30 and 1/2
37      function mint(uint256 _duration, uint256 _tier) external whenNotPaused {    infinite gas
38          require(_duration == 15 || _duration == 30, "incorrect duration input");
39          require(_tier == 1 || _tier == 2, "incorrect duration input");
40
41          uint256 tokenId = tokenConfig.tokenId;
42          uint256 duration = _duration;
43          uint256 tier = _tier;
44
45          unchecked {
46              tokenConfig.tokenId++;
47          }
48          updateSubDetails(tokenId, duration, tier);
49
50          _safeMint(msg.sender, tokenId);
51      }
```

Kuva 33. Mint -funktio

Solidity:ssa näitä ostamis/luomis funktioita kutsutaan yleensä nimellä ”mint”, mutta funktion voi nimetä miten itse haluaa.

- **mint**-funktio ottaa parametreina vastan kaksi uint256 arvoa, jotka määrittävät tilauksen pituuden ja tason.
- **require**-lausunnot määrittävät tietyt ehdot funktion toteutumiselle. Tässä esimerkissä funktio ottaa vastaan vain tiettyjä arvoja, mutta tässä voitaisiin esimerkiksi varmistaa, että ostaja maksaa tokenista tarpeeksi.

Funktioiden sisällä määritetään usein tilapäinen muuttuja, joka perustuu alussa määritettyihin muutujiin. Tämä ja ”unchecked” lausunto ovat vain yksinkertaisia tapoja tehdä funktiosta tehokkaampi.

updateSubDetails -funktio tallentaa tokeniin liittyvät tiedot aiemmin mainittuun **subscriptions**-avain-arvo tietorakenteeseen (Kuva 34).

```
68      function updateSubDetails(uint256 tokenId, uint256 duration, uint256 tier) internal {    infinite gas
69          subscriptions[tokenId].duration = duration;
70          subscriptions[tokenId].tier = tier;
71          subscriptions[tokenId].subDate = block.timestamp;
72          if (duration == 15) {
73              subscriptions[tokenId].subEnds = block.timestamp + 15 days;
74          }
75          if (duration == 30) {
76              subscriptions[tokenId].subEnds = block.timestamp + 30 days;
77          }
78      }
79
```

Kuva 34. updateSubDetails -funktio

_safeMint -funktio luo uuden tokenin, ja lisää sen ostajan Ethereum -lompakkoon. Funktiossa oleva msg.sender arvo hakee automaattisesti funktion käyttäjän Ethereum -osoitteen (Kuva 5).

Loin vielä yhden funktion tokenien voimassaolon tarkistamiseen. Funktio vertaa tokenin tietoihin tallennettua tilauksen päättymispäivämäärää tämänhetkiseen päivämäärään ja antaa vastauksen merkkijonona (Kuva 35).

```
81    function hasValidSubscription(uint256 tokenId) external view returns (string memory){    infinite gas
82        string memory response;
83        if (subscriptions[tokenId].tier == 0) {
84            response = "Token does not exist";
85        }
86        else if (subscriptions[tokenId].subEnds < block.timestamp) {
87            response = "Not valid";
88        }
89        else if (subscriptions[tokenId].subEnds > block.timestamp) {
90            response = "Valid";
91        }
92        return response;
93    }
```

Kuva 35. hasValidSubscription -funktio

Lopuksi vielä testasin kaikkia funktioita ja varmistin, että ne toimivat niin kuin pitääkin (Kuva 36).

Kuva 36. tokenId 0 omaavan tokenin voimassaolon tarkistus

4.3 Tulos

Olen projektin tulokseen tyytyväinen ja projekti osoittautui mielestäni onnistuneeksi. Älysopimus, jonka toteutin, on toimiva ja demonstroi hyvin ideani käytännöllisyyttä. Älysopimuksella pystyy ostamaan ja luomaan tilaustokeneita, ja sen avulla on myös mahdollista tarkistaa, ovatko tilaukset voimassa. Lisäksi, jos päätän puskea älysopimuksen Ethereumin lohkoketjulle, tokeneita voitaisiin myydä hajautetuilla markkinapaikoilla, kuten OpenSea. Voisin vielä tulevaisuudessa laajentaa projektia, ja luoda tokeneille oman markkinapaikan, sekä käyttöliittymän, mikä näyttää sinulle jotain

tiettyä informaatiota riippuen siitä, omistatko voimassa olevaa tilaustokenia. Tämä projekti kuitenkin keskittyi vain back-endiin ja se mielestäni onnistui hyvin.

Tämä voisi olla palveluntarjoajankin kannalta jopa parempi ratkaisu, kuin nykyinen. Tilaus tokeneihin voitaisiin esimerkiksi määrittää jokin tietty prosenttimäärä, joka lähtee jokaisesta tokenin myynnistä palveluntarjoajalle. Tällä tavalla jälkimarkkinat voisivat tuoda takaisin ”menetettyjen” tilausmaksujen tuotot, ja mahdollisesti jopa lisätä tuottoa.

5 Pohdinta

Pohdinta luvussa tarkastelen opinnäytetyöprosessiani monesta eri näkökulmasta, tarjoten kattavan arvioinnin työni etenemisestä ja tuloksista. Ensinnäkin arvioin prosessin toteutusta ja sujuvuutta tarkastelemalla, kuinka hyvin pystyin noudattamaan suunnittelemiani vaiheita ja aikatauluja, sekä mitä muutoksia jouduin tekemään matkan varrella. Kerron myös opinnäytetyöprosessin aikana vastaan tulleista haasteista ja siitä, miten niistä selvittiin.

Lisäksi haluan keskittyä oppimiseen ja henkilökohtaiseen kehitykseen. Reflektoin, mitä uutta opin ja millaisia taitoja kehitin opinnäytetyöni edetessä. Arvioin myös, kuinka hyvin opinnäytetyöprosessi tuki omia oppimistavoitteitani ja miten se vaikutti ammatilliseen kasvuuni. Tämä itsereflektio antaa minulle arvokasta tietoa omasta kehityksestäni ja auttaa minua ymmärtämään, miten kehityin tämän prosessin aikana. Tämä prosessi myös auttaa minua ymmärtämään vahvuuteni ja heikkouteni sekä tunnistamaan ne haasteet, joita kohtasin matkan varrella. Pohdinnassa arvioin myös, kuinka hyvin onnistuin kommunikoimaan ja selittämään monimutkaisia käsitteitä lukijalle, joka ei ehkä ollut ennestään tuttu lohkoketjuteknologian kanssa.

Lopuksi haluan jakaa omia henkilökohtaisia tuntemuksiani, onnistumisen kokemuksiani ja tulevaisuuden näkymiäni. Tämä lisää pohdintaani henkilökohtaista ulottuvuutta ja auttaa lukijaa ymmärtämään, miten opinnäytetyöprosessi vaikutti minuun kokonaisvaltaisesti.

5.1 Opinnäytetyöprojektin pohdintaa

Halusin tuoda esittelemilläni tuotoksilla esille omaa osaamistani Ethereumin älysopimuksista ja niiden soveltamisesta erilaisissa käyttötarkoituksissa. Esittelemäni tuotokset antavat mielestäni hyvin näyttöä osaamisestani ja tietämyksestäni älysopimuskehityksestä, mutta kaikki osaamiseni on vaikea tiivistää kahteen tuotokseen. Jos tekisin tämän saman projektin uudelleen, paremmalla aikataululla, haluaisin lisätä vielä yhden esiteltävän tuotoksen, joka käsittelee hajautettua autonomista organisaatiota (DAO) antaakseni laajempaa näyttöä osaamisestani.

Ethereumin lohkoketjuteknologian ollessa vielä suhteellisen uutta, oli välillä haastavaa kirjoittaa asiat niin, että kuka tahansa voisi ymmärtää ne. Joillekin lohkoketjuteknologiaan liittyville käsitteille ei meinannut löytyä vastaavaa suomenkielistä sanaa, mikä hankaloitti tekstin kirjoitusta. Yritin kuitenkin selittää ja avata näitä käsitteitä mahdollisimman hyvin tietoperustassa.

Esittelemieni tuotosten toiminnan näyttäminen käytännössä oli hieman vaikeaa ilman erillistä käyttöliittymää, mikä vaikeuttaa tuotosten kokonaisuuden hahmottamista, jos tuotoksissa käytetty teknologia ei ole ennestään tuttu. Toivon kuitenkin, että tuotosten toiminnot tulevat tekstin ja kuvien avulla tarpeeksi hyvin selville myös niille, joille tämä teknologia ei ole ennestään tuttu.

5.2 Oman oppimisen pohdintaa

Projektini tärkeimmät tavoitteet olivat tuoda esille osaamiseni Ethereumin älysopimuksista ja niiden soveltamisesta erilaisissa käyttötarkoituksissa, sekä tarjota näyttöä älysopimuskehityksen osaamisestani. Mielestäni onnistuin hyvin näiden tavoitteiden saavuttamisessa, ja esittelemäni tuotokset osoittivat osaamiseni ja tietämykseni aihetta kohtaan.

Opinnäytetyössä esittelemieni tuotosten aihe on mielestäni erittäin ajankohtainen lohkoketjuteknologian kehittyessä kovaa vauhtia. Mistä sain idean tehdä opinnäytetyön tästä aiheesta? Ajatus on ollut mielessäni jo viime kesästä asti, kun pääsin syventymään lohkoketjuteknologiaan tekemäni projektin avulla. Tiesin jo silloin, että tämä on se juttu, mitä haluan tulevaisuudessa tehdä. Tämän seurauksena valitsin tänä keväänä tekemääni tutkimusprosessi opintojakson tutkimukseen aiheeksi tässä esittelemäni tilausten tokenisointi projektin. Projektin alkaessa, minulla oli ollut pitkä tauko älysopimuskehityksestä, mutta päästyäni alkuun innostukseni heräsi taas. Tässä vaiheessa heräsi ajatus, että laajennan tämän tutkimuksen osaksi opinnäytetyötäni.

Minulla oli aluksi mielessä myös toinen aihe opinnäytetyölle, mutta aikataulun ollessa niin tiukka, päädyin tähän aiheeseen ja olen tähän päätökseen erittäin tyytyväinen. Opinnäytetyöprojekti herätti uudelleen innostukseni lohkoketjuteknologiaa kohtaan ja uskon, että tulen työskentelemään tämän teknologian parissa tulevaisuudessakin. Joko omissa henkilökohtaisissa projekteissa, tai töiden parissa.

Projektin aikana opin paljon itsestäni ja omista vahvuuksistani. Sain vahvistusta siitä, että nautin teknisten haasteiden ratkaisemisesta ja pystyn omaksumaan uutta tietoa nopeasti. Kehityin myös projektinhallinnassa, aikatauluttamisessa ja itsenäisessä työskentelyssä. Tämä projekti antoi minulle arvokkaita taitoja, joita voin hyödyntää tulevissa projekteissa ja ammatillisella urallani. Jatkossa haluaisin kehittää edelleen älysopimuskehityksen taitojani ja syventää ymmärrystäni lohkoketjuteknologiasta. Tämä projekti auttoi minua asettamaan tavoitteeni näille alueille ja antoi minulle hyvän lähtökohdan jatkokehitykselle. Projektin aikana sain useita yllättäviä oivalluksia ja oppimiskokemuksia. Esimerkiksi tajusin, miten laajasti lohkoketjuteknologiaa voidaan soveltaa eri aloilla ja miten valtava sen potentiaali on.

Yksi tärkeimmistä opinnäytetyöprojektin aikana oppimistani asioista oli se, miten paljon pystyn saamaan aikaan lyhyessä ajassa, kun panostan asiaan ihan täysillä. Opinnäytetyöprojektini alkoi varsinaisesti alle kuukausi sitten, ja olen nyt viimeisen kahden viikon aikana tehnyt suurimman osan työstä. Tiukan aikataulun tuoma stressi saattoi jopa parantaa minun työskentelytahtiani, kun en pystynyt keskittymään mihinkään muuhun stressin takia. Tästä syystä työskentelin projektin parissa useita tunteja lähes joka päivä. Kokonaisuudessaan opinnäytetyöprojekti oli erittäin antoisa.

Opin paljon omasta työskentelytavastani ja kehityin raporttien kirjoittamisessa huomattavasti. Sanoisin että tämä projekti oli ehdottomasti opiskeluaikani parhaiten opettava tuotos.

Lähteet

Alchemy 2023. Learn Solidity: What is a struct? Luettavissa: https://www.alchemy.com/overviews/solidity-struct. Luettu: 08.05.2023.

Azuki 2023. ERC721A. Luettavissa: https://www.erc721a.org/. Luettu: 27.04.2023.

Bitcoin 2023. Bitcoin is an innovative payment network and a new kind of money. Luettavissa: https://bitcoin.org/. Luettu: 09.02.2023.

Buterin V. 2014. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. Luettavissa: https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_White-paper_-_Buterin_2014.pdf. Luettu: 09.02.2023.

Conti R. 2023. What Is An NFT? Non-Fungible Tokens Explained. Luettavissa: https://www.forbes.com/advisor/investing/cryptocurrency/nft-non-fungible-token/. Luettu: 26.04.2023.

DeJesus T. 2022. What is Tokenization and How Does It Work? Luettavissa: https://www.nasdaq.com/articles/what-is-tokenization-and-how-does-it-work. Luettu: 09.05.2023.

Ethereum 2023a. Welcome to Ethereum. Luettavissa: https://ethereum.org/. Luettu 09.02.2023.

Ethereum 2023b. ERC-721 Non-Fungible Token Standard. Luettavissa: https://ethereum.org/en/developers/docs/standards/tokens/erc-721/. Luettu: 26.04.2023.

Ethereum 2023c. Merkle Proofs For Offline Data Integrity. Luettavissa: https://ethereum.org/en/developers/tutorials/merkle-proofs-for-offline-data-integrity/. Luettu: 09.05.2023.

Hayes A. 2022. Blockchain Facts: What is it, how it works, and how it can be used. Luettavissa: https://www.investopedia.com/terms/b/blockchain.asp. Luettu: 11.03.2023.

IBM s.a. What is blockchain technology? Luettavissa: https://www.ibm.com/topics/blockchain. Luettu 20.02.2023.

Leech O. 2022. What Are NFTs and How Do They Work? Luettavissa: https://www.coindesk.com/learn/what-are-nfts-and-how-do-they-work/. Luettu: 18.05.2023.

Nakamoto S. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. Luettavissa: https://bitcoin.org/bitcoin.pdf. Luettu: 09.02.2023.

Opensea 2023. Metadata Standards. Luettavissa: https://docs.opensea.io/docs/metadata-standards. Luettu: 09.05.2023.

OpenZeppelin 2023a. Ownership. Luettavissa: https://docs.openzeppelin.com/contracts/2.x/api/ownership#Ownable. Luettu: 01.05.2023.

OpenZeppelin 2023b. Security. Luettavissa: https://docs.openzeppelin.com/contracts/4.x/api/security. Luettu: 01.05.2023.

OpenZeppelin 2023c. Utils. Luettavissa: https://docs.openzeppelin.com/contracts/3.x/api/utils#Strings. Luettu: 01.05.2023.

OpenZeppelin 2023d. Cryptography. Luettavissa: https://docs.openzeppelin.com/contracts/3.x/api/cryptography. Luettu: 01.05.2023.

Paiementor 2020. Blockchain explained and its application to payments. Luettavissa: https://www.paiementor.com/blockchain-explained-application-payments/. Luettu: 20.03.2023.

Rodeck D. & Curry B. 2022. What Is Blockchain? Luettavissa: https://www.forbes.com/advisor/investing/what-is-blockchain/. Luettu: 20.02.2023.

Solana 2023. Powerful for developers. Fast for everyone. Luettavissa: https://solana.com/. Luettu: 09.02.2023.

Solidity 2023. Solidity. Luettavissa: https://docs.soliditylang.org/en/latest/index.html#getting-started. Luettu: 11.03.2023.

Wikipedia 2023. Merkle tree. Luettavissa: https://en.wikipedia.org/wiki/Merkle_tree. Luettu: 26.04.2023.

Yakovenko A. 2017. Solana: A new architecture for a high performance blockchain v0.8.13. Luettavissa: https://solana.com/solana-whitepaper.pdf. Luettu: 09.02.2023.

Zapotochnyi A. 2022. What are Smart Contracts? Luettavissa: https://blockgeeks.com/guides/smart-contracts/. Luettu: 20.02.2023.

Liitteet

Liite 1. GitHub

https://github.com/lo6ical/SubscriptionToken

Liite 2. GitHub

https://github.com/lo6ical/NFTcontract