samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

EETU SALLI

Virtuaalitodellisuutta hyödyntävän kuntoutujan liikedatan tallennus ja hyötykäyttö

TIETOJENKÄSITTELYN TUTKINTO-OHJELMA
2022

Tekijä(t)
Salli, Eetu

Julkaisun laji
Opinnäytetyö, AMK

Päivämäärä
Toukokuu 2022

Sivumäärä
38

Julkaisun kieli
Suomi

Julkaisun nimi
Virtuaalitodellisuutta hyödyntävän kuntoutujan liikedatan tallennus ja hyötykäyttö

Tutkinto-ohjelma
Tietojenkäsittelyn tutkinto-ohjelma

Tiivistelmä

Tämä opinnäytetyö tehtiin Satakunnan ammattikorkeakoululle hankkeeseen ”Verkostoyhteistyöllä vauhtia tekoälypohjaisten virtuaaliteknologioiden saavutettavuuteen”. Annettuna tehtävänä oli kehittää ratkaisu virtuaalitodellisuutta käyttävän kuntoutujan liikedatan keräämiseen ja hyödyntämiseen kuntoutusprosessissa.

Tavoitteena oli suunnitella ja toteuttaa kokonaisuus, jonka kerää kuntoutujan liikedataa, siirtää sitä tarvittavasti seuraavalle laitteistolle, tallentaa sitä, sekä luoda työkalu, jolla liikedataa voi analysoida data-analyysin keinoin.

Työssä ensin valittiin ratkaisuun arkkitehtuurinen kokonaisuus, joka pohjautui toteutuksen yksinkertaisuuteen ja käytön helppouteen, sekä valittiin teknologiat tämän kokonaisuuden rakentamiseen. Tämän jälkeen kokonaisuus toteutettiin.

Tuloksena syntyi prototyyppi, joka demonstroi ratkaisun potentiaalia liikedatan keräämiseen VR-ohjaimista sekä sen analysoinnista data-analyysiä varten rakennetulla työpöytäsovelluksella. Työssä ei oteta kantaa siihen, minkälaista data-analyysiä liikedatalle tulisi tehdä, jotta se tukisi kuntoutusprosessia. Sen sijasta työssä keskitytään enemmän systeemin tekniseen toteutukseen.

Avainsanat
virtuaalitodellisuus, kuntoutus, liikedata, ohjelmistosuunnittelu, C#, JavaScript, Python, data-analyysi

Author(s)
Salli, Eetu

Type of Publication
Bachelor's thesis

Date
May 2022

Number of pages
38

Language of publication:
Finnish

Title of publication
Rehabilitation using Virtual Reality: collecting and utilizing patient's movement data

Degree programme
Business Information Systems

Abstract

This thesis was created for Satakunta University of Applied Sciences for a project called "Verkostoyhteistyöllä vauhtia tekoälypohjaisten virtuaaliteknologioiden saavutettavuuteen". The given task was to develop a solution to collect movement data from a rehab patient using a virtual reality device and to utilize the data in a rehabilitation process.

The goal was to design and implement a system that collects the VR user's movement data, transfers the data between required devices, stores it and to create a tool that you can analyse data using data analytics.

An architecture for the system was selected that was based on simplicity of development and ease of use. Afterwards the technologies were selected that were used to create the system. Finally the system was realized.

The end result was a proof of concept that demonstrates the potential of the solution in collecting movement data from VR controllers and analyzing the data in desktop application that was built for data analytics. This thesis doesn't take a stance on how data analysis should be used to support the rehabilitation process and instead focuses on the technical implementation of the system.

Keywords
virtual reality, rehabilitation, movement data, software design, C#, JavaScript, Python, data analytics

SISÄLLYS

SYMBOLI- JA LYHENNELUETTELO (EI PAKOLLINEN)

VR	Virtual Reality (virtuaalitodellisuus)
VR-lasit	Virtuaalitodellisuuslasit: virtuaalitodellisuuskäyttäjän käyttämä laitteisto
VR-laitteisto	Virtuaalitodellisuuslasit sekä niiden mukana tuleva ohjaimet, sekä muu mahdollinen laitteisto

1 JOHDANTO

Kuntoutuspalveluja käyttävien ihmisten määrä kasvaa Suomessa vuosittain. Vuonna 2020 n. 2,6 % Suomen väestöstä käytti jotakin Kelan kuntoutuspalvelua, kun vastaavasti sama luku vuonna 2010 oli n. 1,5 % (Kelan www-sivut 2022; The World Bankin www-sivut 2022). Työkyvyn ylläpitäminen tai parantaminen on tärkeä yhteiskunnallinen tehtävä. Sen lisäksi elämänlaadun ylläpitäminen tai parantaminen on moraalisesti tärkeä tehtävä. Teknologian kehittyessä ja kuntoutettavien määrän noustessa on teknologiankin rooli noussut yhä tärkeämmäksi osaksi kuntoutusprosessia. Erityisen potentiaalisia teknologioita ovat virtuaalitodellisuus sekä tekoäly. Teknologian vaikutus kuntoutusprosessiin voi olla moninainen. Se voi parantaa diagnoosia tai sen saatavuutta, se voi antaa työkaluja prosessin seurantaan, arviointiin ja kehittämiseen ja se voi parantaa potilaan kokemusta kuntoutusprosessista tai lisätä potilaan sitoutumista.

Tässä opinnäytetyössä käydään läpi prosessia järjestelmän toteuttamisesta, jossa päämääränä on hyödyntää virtuaalitodellisuutta käyttävän kuntoutujan liikedataa kuntoutusprosessissa. Käytännössä toteutettiin prototyyppi järjestelmästä, joka kerää liikedataa, tallentaa sitä ja jota voidaan analysoida käyttöä varten luodulla data-analyysin työkalulla.

2 JÄRJESTELMÄN TAUSTA

2.1 Tilaajayritys ja hanke

Työ on toteutettu Satakunnan ammattikorkeakoululle (SAMK) hankkeeseen *Verkostoyhteistyöllä vauhtia tekoälypohjaisten virtuaaliteknologioiden saavutettavuuteen* kuntoutuksessa. Hanke toteutettiin verkostoyhteistyönä, jossa korkeakouluina yhdessä toimivat Tampereen korkeakouluyhteisö, Metropolia ammattikorkeakoulu sekä Satakunnan ammattikorkeakoulu. Hankkeen yhtenä päämääränä oli testata ja kerätä kokemuksia tekoälyn sekä virtuaaliteknologioiden hyödyntämisessä opiskelu-, työ- ja toimintakyvyn edistämisessä. Sen lisäksi tarkoituksena on varmistaa näiden ratkaisujen saatavuutta sekä tuottaa uusia kuntoutuksen palvelukonsepteja. Tarkoituksena on myös kasvattaa ymmärrystä ja osaamista uusista teknologioista ja lisätä innovaatiota näiden kentällä. SAMK:n rooli lähtökohtaisesti oli tuoda mukanaan edeltävää osaamista ja kokemusta hyötypeliteknologioista sekä virtuaalitodellisuusteknologioista kuntoutuksen kentällä. (Tampereen korkeakouluyhteisön www-sivut 2022.)

2.2 Tehtävän tausta

Sain Satakunnan korkeakoululta tehtävän kehittää ratkaisua virtuaalitodellisuutta käyttävän kuntoutujan liikedatan hyödyntämiseen. Hankkeen silmissä tehtävän voi luokitella sopivan virtuaaliteknologioiden hyödyntämiseen kuntoutuksessa sekä uusien kuntoutuksen palvelukonseptien prototypointiin. Tehtävänannossa ei määritelty käytettäviä laitteistoja tai teknologioita, tai yleistä ohjelmistoarkkitehtuuria, vaan tehtävänä oli itse muotoilla ja toteuttaa ratkaisu.

2.3 Virtuaalitodellisuus sekä virtuaalitodellisuuslaitteisto

Virtuaalitodellisuus (*Virtual reality, VR*) voidaan kattavimmin määritellä todellisuudeksi, joka vaikuttaa olevan olemassa, mutta ei oikeasti ole. Yleisesti ottaen sillä tarkoitetaan kuitenkin tietokoneteknologialla aikaan saatua vaikutelmaa kolmiulotteisesta interaktiivisesta maailmasta, jonka sisällä olevilla objekteilla on avaruudellisen läsnäolon tuntu. Ihminen havaitsee todellisuutta useilla eri aisteilla, mutta pääasiassa

yleisesti virtuaalitodellisuudesta puhuttaessa puhutaan virtuaalimaailman havaitsemisesta visuaalisesti ja auditiivisesti. (Nasa Advanced Supercomputing Divisionin www-sivut 2022.)

Virtuaalitodellisuuslaitteistolla tarkoitetaan tässä työssä laitteita, jotka on luotu erityisesti virtuaalitodellisuuden välittämiseksi käyttäjälle, ja jotka toimivat keskenään yhteistyössä sen aikaansaamiseksi. Esimerkki VR-laitteistosta on *Oculus Quest 2*-virtuaalitodellisuuslaitteisto, joka vuonna 2020 oli myydyin virtuaalitodellisuuslaitteisto (Alsop 2022). Kyseiseen laitteistoon kuuluvat VR-lasit (engl. *VR glasses* tai *VR headset*) sekä käsissä pidettävät ohjaimet (engl. *controllers*). VR-laseihin kuuluvat pienet kaiuttimet korvien lähellä sekä kaksi näyttöä, jotka peittävät suuren osan käyttäjän näkökentästä. Ohjainten avulla käyttäjä voi navigoida ja toimia interaktiivisesti näyttöjen esittämässä virtuaalitodellisuudessa.

2.4 Liikedatan hyödyntäminen data-analyysin keinoin

Datalla tarkoitetaan arvoja tai mittaustuloksia, jotka itsessään eivät tarkoita vielä mitään. Kun dataa käsitellään ja jalostetaan eri tavoin, siitä muodostuu informaatiota. Tällä informaatiolla on olemassa jokin asiayhteys ja selkeä sisältö. Kun informaatiota tulkitaan tietyssä kontekstissa ja sille asetetaan merkitys, voidaan puhua tiedosta (Jalonen 2018). Tästä prosessista voidaan puhua data-analyysinä: datasta tuotetaan merkityksellistä informaatiota eri työkalujen ja keinojen avulla. Tästä data-analyysillä tuotetusta informaatiosta kuntoutuksen ammattilainen voisi mahdollisesti tulkita merkityksellistä tietoa kuntoutuksen tueksi. Data-analyysiin sisältyy esimerkiksi myös kuvaajien tuottaminen datasta. (Master’s in Data Science www-sivut 2022.)

Liikedatalla tarkoitetaan tässä työssä mittaustuloksia, joita voidaan käyttää hyödyksi liikkeen tutkimisessa ja seuraamisessa. Liikesensori voidaan ajatella pisteeksi kolmiulotteisessa avaruudessa, ja liikedataa voidaan tässä ajatella datana, joka jalostamisen jälkeen kertoo jotakin tämän pisteen liikkumisesta ajanjakson aikana. Esimerkiksi nopeus yhdessä kuluneen ajan kanssa voi kuvailla liikettä. Sen lisäksi kiinnostavia arvoja voivat olla sijainti tai kiihtyvyys. Mitatut asiat ovat riippuvaisia käytetystä liikesensorista, mutta usein edellä mainittujen lisäksi saadaan tietoa sensorin suunnasta tai

rotaatiosta. Kuntoutuskäytössä liikedataa voidaan mitata kuntoutuskohtaisesti relevantista paikasta. Esimerkiksi olkapään liikkuvuutta mitattaessa voitaisiin mitata liikettä kyynärtaipeesta.

Tässä työssä ei oteta kantaan millä tapaa liikedataa tulisi analysoida, jotta siitä olisi mahdollista saada hyödyllistä tietoa kuntouksen tueksi. Prototyypissä valittiin esitettäväksi hyvin yksinkertaista datan tulkintaa, kuten esimerkiksi kuvaajia datasta sekä erilaisia keskilukuja, kuten keskiarvo tai mediaani. Tarkoituksena on esittää konsepti, jonka päälle voi rakentaa merkityksellisemmän data-analyysin työkaluja.

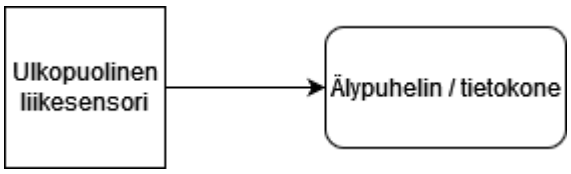
3 JÄRJESTELMÄN SUUNNITTELU

Annetussa tehtävässä ei ollut ennalta määriteltyjä vaatimuksia, vaan VR:ää hyödyntävän kuntoutujan liikedatan hyödyntäminen tehtävänä oli hyvin avoin. Koska laitteistoja, järjestelmäarkkitehtuuria tai käytettyjä teknologioita ei määritelty tai rajattu, piti suunnittelussa ensin kartoittaa mahdollisia eri ratkaisumalleja. Lähtiessäni suunnittelemaan työtä jaoin ongelman karkeasti datan keräämiseen, datan siirtämiseen, datan tallentamiseen sekä datan hyödyntämiseen.

3.1 Järjestelmän ratkaisumallien kartoittaminen

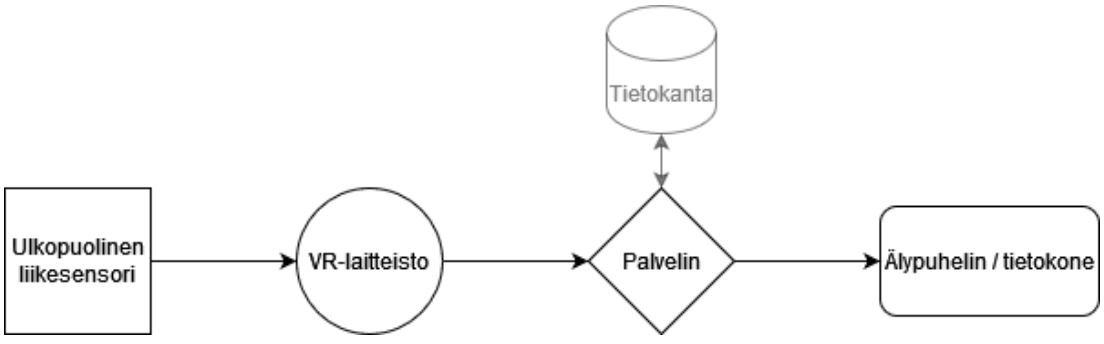
Yleisesti voidaan todeta, että datan kerääminen tapahtuu jollakin liikesensorilla. Datan siirtäminen tapahtuu jotakin tiedonsiirtoprotokollaa tai -protokollia käyttäen. Datan tallentaminen tapahtuu joko jonkin palvelimen tietokantaan tai jonkin laitteen, kuten tietokoneen tai puhelimen, paikalliseen varastoon. Datan hyödyntäminen tapahtuu data-analyysin työkaluja käyttäen jollakin ohjelmistolla, kuten tietokoneeseen tai älypuhelimeen toteutetulla ohjelmistolla. Näille olemassa olevista teknologia- tai ratkaisuvaihtoehdoista voi kuitenkin muodostaa useita erilaisia kokonaisuuksia, ja projektin alussa tarkastelin erilaisia vaihtoehtoja. Työssä ei esitetä kaikkien vaihtoehtojen hyötyjä tai heikkouksia, mutta esitetään eri harkitut vaihtoehdot ja todetaan mikä vaihtoehto niistä päätettiin toteuttaa.

Kaaviossa 1 esitetään yksinkertainen malli: tieto kerätään jollakin ulkopuolisella, kuntoutujaan kiinnitettävällä liikesensorilla ja siirretään suoraan laitteistolle, jossa data tallennetaan ja jossa data-analyysi suoritetaan. Kaavio 2 laajennetaan ensimmäisen ehdotelman ratkaisua lisäämällä väliin uuden komponentin: VR-laitteiston, joka kuntoutujalla on käytössään. Edelleen edellistä laajentaen, kaaviossa 3 esitetään ratkaisumalli, jossa lisätään taas uusi komponentti: VR-laitteistosta data siirtyy palvelimelle ja palvelimelta data-analyysissä käytettävälle laitteistolle. Tässä ratkaisumallissa voidaan myös tallentaa data palvelimen yhteydessä olevaan mahdolliseen tietokantaan. Kaaviossa 4 ja 5 mallit mukailevat edellisiä, mutta ulkopuolisen sensorin sijasta datan keräämiseen hyödynnetään VR-lasien mukana tulevia ohjaimia, jotka jo valmiiksi mittaavat omaa liikettään. Kaaviossa ohjaimet sisältyvät VR-laitteistoon.

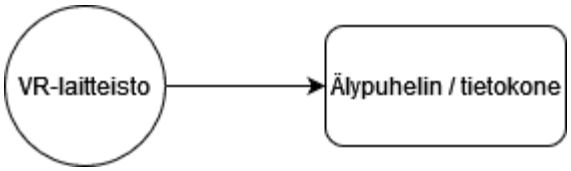


Kaavio 1. Kerätty data siirtyy suoraan ulkopuolisesta liikesensorista data-analyysissä käytettävään laitteistoon. Nuoli kuvaa liikedatan siirtoa.

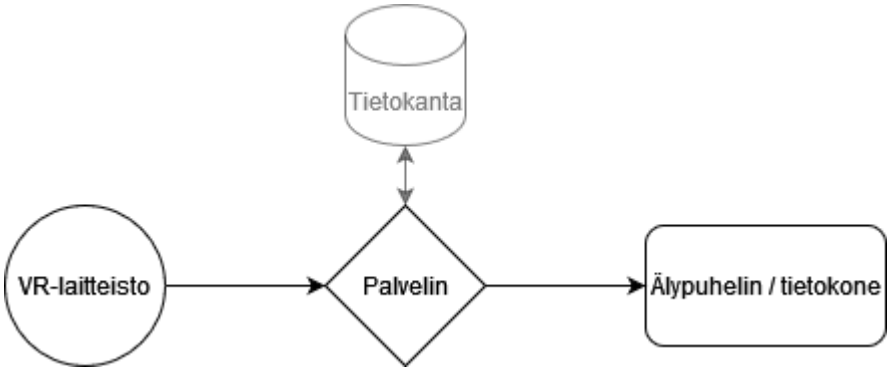
Kaavio 2. Kerätty data siirtyy ensin VR-laitteistoon, joka siirtää sen data-analyysissä käytettävään laitteistoon.



Kaavio 3. Kerätty liikedata siirtyy VR-laitteiston kautta palvelimelle, jossa se mahdollisesti tallentuu tietokantaan. Palvelimelta data siirtyy data-analyysissä käytettävälle laitteistolle.



Kaavio 4. Liikedata kerätään VR-laitteiston ohjaimilla, ja data siirtyy VR-laitteistosta suoraan data-analyysissä käytettävään laitteistoon.



Kaavio 5. Liikedata kerätään VR-laitteiston ohjaimilla, ja data siirtyy palvelimelle, jossa data tallennetaan mahdollisesti tietokantaan. Palvelimelta data siirtyy data-analyysissä käytettävään laitteistoon.

3.2 Ratkaisumallin päättäminen

Edellisessä luvussa esitetyissä malleissa on erilaisia mahdollisia hyötyjä tai heikkouksia. Päätöksenteon tukena käytettiin ajatusta siitä, että kokonaisuus olisi mahdollisimman yksinkertainen ja helppo toteuttaa. Datan keräämisestä voidaan todeta, että ulkopuolinen sensori lisää kokonaisuuteen kompleksisuutta ylimääräisellä laitteistolla, sillä kuntoutujalla on lähtökohtaisesti VR-laitteisto käytössään ja VR-laitteiston avulla on mahdollista kerätä ohjainten liikedataa. Sen lisäksi yksinkertaisuus mielessä pyrittiin välttämään Bluetooth-tiedonsiirtoprotokollan käyttöä, jos vaihtoehtona oli käyttää jotakin internetteknologioissa käytettyjä tiedonsiirtoprotokollia, kuten esimerkiksi HTTP:ta, ja ulkopuoliset liikesensorit pääasiassa välittävät dataa Bluetooth-protokollalla. Bluetoothia päätettiin välttää, koska koin sen olevan työläämpi ratkaisun kannalta. Bluetooth vaatii esimerkiksi myös aina fyysisen läheisyyden toimiakseen, joka myös saattaisi rajoittaa järjestelmän käyttöä esimerkiksi etäkuntoutuksessa.

Yksinkertaisen toteutuksen ja käytön lisäksi nähtiin etuna tiedon tallentaminen tietokantaan, sillä se antaa edellytykset turvalliseen ja luotettavaan tiedonkäsittelyyn ja säilytykseen ja parantaa datan saatavuutta. Lisäksi data-analyysissä käytettäväksi laitteistoksi todettiin parempana vaihtoehtona tietokone kuin älypuhelin. Ratkaisu tehtiin osittain työkalun käyttömukavuuden kannalta. Datan tulkinta ja analysointi saattaa vaatia ruutupinta-alaa, sekä hiiri ja näppäimistö saattavat tehostaa tai helpottaa ohjelmiston käyttöä. Lisäksi ratkaisu perustui sovelluskehitysprosessin helpottamiseen. Koin tietokoneen helpommaksi kohdealustaksi kehittää käytettävä sovellus.

Edellisten pohjalta valituksi ratkaisumalliksi päätyi siis kaavio 5, jossa liikedata kerätään VR-laitteiston ohjainten avulla. Liikedata siirretään palvelimelle, jossa se myös tallennetaan tietokantaan. Palvelimelta liikedata siirretään tietokoneelle data-analyysiohjelmistolle. Tämän toteuttamiseen valittuihin teknologioihin ja datan siirtoon tutustutaan seuraavaksi.

4 KÄYTETYT TEKNOLOGIAT

Valitussa ratkaisumallissa käytetään useita eri teknologioita, jotka esitellään tässä kappaleessa. Joissakin tapauksissa myös saatetaan lyhyesti mainita vaihtoehtoisia teknologioita, jotka olisivat saattaneet olla myös hyviä vaihtoehtoja. Käytössä on VR-laitteistona Oculus Quest, mutta laitteistoksi kävisi potentiaalisesti moni muukin VR-laitteisto. Liikedatan saamiseksi VR-laitteistosta ohjelmallisesti ulos käytetään Unityä ja C#:a. Palvelin on toteutettu Node.js:llä ja JavaScriptilla, sekä tietokanta MongoDB:lla. Data-analyysin työkalu on toteutettu PyQt-ohjelmistokehyksen ympärille Pythonilla. Datan siirrossa on käytetty TCP/IP-protokollien päälle rakennettuja HTTP- sekä WebSocket-protokollia.

4.1 Virtuaalitodellisuuslaitteisto

Esitetyssä ratkaisussa liikedata mitattaisiin VR-laitteiston ohjaimista. VR-ohjaimet käyttävät sijaintinsa ja liikkeensä seuraamiseen eri tekniikoita. Ohjaimet käyttävät

kiihtyvyysantureita, gyroskooppeja tai magnetometrejä liikkeen ja orientaation arviointiin. Näiden lisäksi ohjaimet käyttävät sijainnin ja liikkeen arviointiin myös joko ns. inside-out-seurantaa tai outside-in-seurantaa. Inside-out-seurannassa VR-laseissa on sisäänrakennettuja kameroita, jotka seuraavat ohjainten sijaintia. Käyttäen ohjainten antureiden dataa yhdessä kameroiden kanssa koneoppimisalgoritmi tunnistaa ja ennustaa ohjaimen liikettä ja sijaintia. Outside-in-seurannassa periaate on sama, mutta kamerat ovat lasien ulkopuolella erillisinä laitteina. Inside-out- sekä outside-in-seurannat molemmat mahdollistavat tarkan liikkeen ja sijainnin määrittämisen. Oculus Quest-laitteistoissa käytetään inside-out-seurantaa. (Weis 2018.)

VR-ohjainten on todettu olevan lupaavia vaihtoehtoja kliiniseen käyttöön tarkkuutensa puolesta. Liikkeen arviointi kuitenkin perustuu koneoppimisalgoritmiin sekä eri ohjainten välillä on teknisiä eroja, joten minkä tahansa VR-ohjaimen ei voi olettaa olevan pätevä valinta, vaan se pitää varmentaa tapauskohtaisesti (Shum, Valdés & Van der Loos 2019; Passos & Jung 2020, 19-26; Jost, Nelson & Rylander 2021, 632-636). Sen lisäksi käyttötarkoitukset saattavat vaikuttaa käytettävyyteen. Esimerkiksi suurissa liikkeissä pieni epätarkkuus ei välttämättä haittaa, mutta tarkkuutta vaativissa liikkeissä sentinkin heitto voi olla merkittävä. Sen lisäksi inside-out-seurannassa ohjaimet ovat epätarkkoja silloin, kuin VR-lasien kameroilla ei ole näköyhteyttä ohjaimiin. Tämä epätarkkuus potentiaalisesti rajoittaa ohjainten käyttöä liikkeen mittaamisessa paikoissa, joissa ei ole näköyhteyttä VR-lasien kameraan, kuten esimerkiksi pohkeessa. Yleisesti voidaan kuitenkin todeta VR-ohjainten olevan potentiaalinen mittalaite kliiniseen käyttöön, kunhan käyttöyhteys ja laitteisto harkitaan tapauskohtaisesti.

Rajoitteena datan keräämisessä ohjainten avulla, verrattuna perinteisempään, erilliseen liikesensoriin, on ohjainten rajoittuneempi kiinnitettävyys. VR-ohjaimet ovat käsin pideltäviä, joten niitä ei ole suunniteltu kiinnitettäväksi käsistä poikkeaviin paikkoihin. Ohjain on huomattavasti työläämpi kiinnittää esimerkiksi polveen kuin ulkopuolinen liikesensori. Ratkaisuja tähän rajoitteiseen tuo mahdollisesti kekseliäät kiinnitysmetodit, kuten esimerkiksi kustomoidut 3D-tulostetut pidikkeet ja/tai kustomoidut tekstiiliratkaisut. Näissäkin on kuitenkin huomioitava VR-laitteiston kameroiden näköyhteyden säilyttäminen ohjaimiin.

4.2 Unity ja C#

Datan kerääminen ohjainten avulla tapahtuu virtuaalitodellisuutta varten luodun ohjelmiston kautta. Edellytys datan keräämiseen on sellainen rajapinta ohjelmiston ja VR-lasien välille, joka mahdollistaa pääsyn ohjainten sijainti- ja liikedataan. Datan kerääminen tarkoittaa ohjelmallisesti halutun datan (esimerkiksi sijainnin ja kiihtyvyyden arvojen) tallentamista VR-lasien välimuistiin, ja suorituksen loputtua datan lähettämistä eteenpäin haluttua teknologiaa käyttäen.

Monet VR-pelit ja -sovellukset toteutetaan Unity-alustalla, jossa olemassa oleva rajapinta mahdollistaa tämän. Unity on Unity Technologiesin kehittämä pelimoottori, jolla saa luotua niin 2D- kuin 3D-pelejä monelle eri alustalle, kuten älypuhelimille, konsoleille ja tietokoneille. Unityssä liikedata saadaan VR-ohjaimista käyttämällä Unityn XR-rajapintaa (Unityn dokumentaatio 2022). Rajapinta, tai ohjelmointirajapinta, tarkoittaa ohjelmallista toteutusta, joka mahdollistaa kahden eri ohjelman kommunikaation keskenään. Tässä tapauksessa rajapinta mahdollistaa Unityllä tehdyn sovelluksen ja VR-lasien välisen kommunikaation. VR-lasit puolestaan kommunikoivat laitteiston ohjainten kanssa.

Unitylle sovellusta kehittäessä käytetään pääasiassa ohjelmointikielenä C#:a. C# on Microsoftin kehittämä avoimen lähdekoodin oliopohjainen ohjelmointikieli. Sitä käytetään monipuolisesti erilaisissa ohjelmistoprojekteissa, kuten esimerkiksi peleissä, työpöytäsovelluksissa ja web-applikaatioissa. (Chand 2020.)

4.3 Node.js -pohjainen web-palvelin

Web-palvelin tarkoittaa palvelinta, joka keskustelee toisten osapuolten kanssa HTTP-protokollaa noudattaen. Tässä tapauksessa määrittelyyn lisätään myös WebSocket-protokollaa hyödyntävä kommunikaatio. Koska palvelin käsittelee tässä tapauksessa mahdollisesti arkaluontoista dataa, on palvelimen tietoturvallisuus äärimmäisen tärkeä. Tietoturvallisuuden näkökulmaa ei kuitenkaan tässä opinnäytetyössä tarkastella. Web-palvelimille ovat ominaisia *endpointit*, jotka ovat eri päätteitä, joihin voidaan lähettää erilaisia viestejä. Endpointit palvelevat eri tarkoituksia tai tarpeita: esimerkiksi yksi endpoint voi olla pääte, joka lähettää dataa vastauksena tiedusteluihin ja toinen

endpoint voi olla pääte datan tallentamiseen palvelimelle, kun se vastaanottaa dataa. (Oracle WSIT-dokumentaatio 2010)

Prototyypin web-palvelin toteutetaan Node.js:ää hyödyntäen. Node.js on alustariippumaton ajoympäristö JavaScript-koodin suorittamiseen palvelimella, joka pohjautuu Googlen Chrome V8 JavaScript-moottoriin. Node.js:llä on paljon etuja, joista yksi on äärimmäinen suosio (Stack Overflow 2018), joka tarkoittaa laajaa dokumentaatiota sekä runsaasti löytyvää tukimateriaalia internetistä. Sen lisäksi siihen löytyy lähes lukematon määrä eri kirjastoja, jotka mahdollistavat ja nopeuttavat kehitystyötä. Node.js -ympäristössä ohjelmointikielenä on JavaScript, joka on hyvin suosittu kieli ja jota on helppo ja nopea kirjoittaa. Nämä yhdessä tarkoittavat, että Node.js -ympäristössä kehitystyö on suhteellisen nopeaa ja helppoa. Node.js myös käsittelee suuren käyttäjämäärän tehokkaasti ja on yleisesti hyvin responsiivinen. (Kaneriya 2020.)

Node.js ei ole parhaimmillaan käytössä, jossa tehdään raskaista laskentoja prosessorilla. Mikäli palvelin tekisi runsaasti laskentoja isoilla datamäärillä, Node.js ei olisi välttämättä tehokkain teknologiaratkaisu palvelimelle. Tätä ongelmaa voisi kuitenkin kiertää kutsumalla jotakin toista prosessia palvelimen koodista. Tämä toinen prosessi käyttäisi jotakin laskennallisesti tehokkaampaa kieltä tai alustaa kuin Node.js, ja suorittaisi raskaat laskennat. (Kaneriya 2020.)

4.4 MongoDB:llä toteutettu tietokanta

Tietokannalla tarkoitetaan organisoitua kokoelmaa rakenteellisesta informaatiosta, joka on tallennettu yleensä sähköisessä muodossa tietokonejärjestelmään. Tietokantaan liittyy myös sen hallintasysteemi (Oraclen www-sivut 2022). Eräs tietokantamalli on NoSQL-malli, jotka ovat tulleet suosioon viimeisen vuosikymmenen aikana (Schaefer 2022). Hyvin pelkistetysti esitettynä, NoSQL-tietokannat tarjoavat ratkaisun suurelle määrälle dataa, joka ei ole välttämättä hyvin strukturoitua. NoSQL-tietokannassa data tallennetaan sillä tapaa, että tietokantaan ei tarvitse luoda nk. skeemoja tietokantaan ennen sen käyttöönottoa. Skeema on erilainen joukko sääntöjä ja relaatioita, joka määrittelee syötettävän datan yhteyksiä toisiinsa etukäteen. Skeeman puuttumisen johdosta NoSQL on hyvä vaihtoehto myös silloin, kuin tallennettavan tiedon

rakennetta ei täysin tiedetä, tai se saattaa muuttua. (Conceptatech 2017). Eräs NoSQL-tietokanta on MongoDB-tietokanta, joka valittiin yksinkertaisesti sen kohtalaisen suosion johdosta.

Toinen vaihtoehto tietokantaratkaisuksi olisi ollut jokin relaatiotietokanta (SQL-pohjainen tietokanta), joiden etuna verrattuna NoSQL-tietokantoihin on korostuneempi painotus datan säilyvyyttä ja eheyttä kohtaan (Conceptatech 2017). Potilasdatan kanssa relaatiotietokanta saattaisi siis olla perustellumpi. Tässä työssä NoSQL-tietokanta valittiin sen takia, että työ oli luonteeltaan kehittävää ja kokeilevaa, ja tallennettavan tiedon rakenne muuttui kehitystyön aikana. Prototyypistä kehitettävästä lopullisemmassa tuotteessa relaatiotietokanta olisi mahdollisesti parempi vaihtoehto.

4.5 Data-analyysiin käytettävä työpöytäsovellus: Python ja PyQt

Data-analyysiin käytettävä työpöytäsovellus päätettiin tehtäväksi Pythonilla. Valinta perustuu työn tekijälle tuttuuteen, helppouteen sekä sen olevan yleisesti perusteltu valinta data-analytiikkakäyttöön. Python on suosittu, helppo ja nopea kieli tuottaa ohjelmistoja. Sen lisäksi siihen on olemassa suuri määrä kirjastoja ja ohjelmistokehyksiä sekä tukimateriaalia internetissä. Python on myös yksi suosituimmista kielistä data-analytiikan tekemiseen. Muita vaihtoehtoja olisi kuitenkin teoriassa ollut luoda ohjelmisto käyttäen esimerkiksi Javaa, C/C++:aa tai R:ää.

Ohjelmisto vaatii käyttöliittymän, johon tässä työssä valitaan tässä graafinen käyttöliittymä, eli GUI (*graphical user interface*). Pythonille on olemassa paljon ohjelmistokehyksiä tai rajapintoja GUI:n rakentamiseen. Yksi suosituimmista vaihtoehdoista GUI:n toteuttamiseen Pythonilla ovat rajapinnat Qt-ohjelmistokehykseen (DEV 2019). PyQt on Pythonille tehty rajapinta Qt-kehitysympäristölle. Qt tarjoaa työkaluja graafisen käyttöliittymän sekä muiden toiminnallisuuksien rakentamiseen. Toinen rajapinta Qt-kehitysympäristölle on PySide, joka on hyvin samankaltainen PyQt:n kanssa. Tarkasteluun valittiin PyQt, sillä se on tällä hetkellä PySideä suositumpi, joten sillä kehittämisen tueksi löytyy enemmän tukiaineistoa (Fitzpatrick 2019). PyQt:n mukana tulee GUI:n suunnitteluun tarkoitettu ohjelmisto, Qt Designer, jolla suunniteltuja käyttöliittymiä on helppo sisällyttää PyQt-ohjelmistoihin. Tämän lisäksi PyQt:lla

tehdyn ohjelman käyttöliittymään on helppo liittää data-analytiikkaan liittyviä toiminnallisuuksia, kuten esimerkiksi erilaisia kuvaajia. Suosittu valinta kuvaajien piirtämiseen Pythonissa on matplotlib-kirjasto, jolla onnistuu laajasti monenlaisten kuvaajien piirto. PyQt-kehyksen kanssa 2D-kuvaajien ja 3D-kuvaajien piirtämiseen voidaan käyttää myös PyQtGraph-kirjastoa. Näiden lisäksi PyQt tarjoaa Qt:n toiminnallisuuksista myös mahdollisuudet HTTP- sekä WebSocket -kommunikaatioon.

4.6 Datan siirtäminen HTTP- sekä WebSocket-protokollilla.

VR-laitteistolla kerätty liikedata tulee siirtää eteenpäin palvelimelle, sekä myös työpöytäsovellukselle. Tämä siirtäminen tapahtuu internetin välityksellä käyttäen HTTP (*Hypertext Transfer Protocol*)- sekä WebSocket-protokollilla.

HTTP ja WebSocket ovat molemmat tiedonsiirtoprotokollia, jotka ovat rakennettu TCP-protokollan (*Transmission Control Protocol*) päälle. HTTP on yleinen tiedonsiirtoprotokolla, jota käytetään yleensä kommunikaatioon web-palvelimien kanssa. Periaatteellisena erona HTTP:n ja WebSocketin välillä on se, että HTTP:lla kommunikaatio perustuu ”request & response” -tyyppiseen yhteydenpitoon. Asiakasohjelma lähettää pyynnön palvelimelle, johon palvelin vastaa. Yhteydenpito on siis yksisuuntaista siltä osin, että sen aloittaa aina asiakasohjelma. WebSocket-protokollassa yhteydenpito on kaksisuuntaista: palvelin voi oma-aloitteisesti kommunikoida asiakasohjelmalle (Choudhary 2022). Tästä syystä WebSocketia pidetään hyvänä ratkaisuna tapauksissa, joissa priorisoidaan nopeita, reaaliaikaisia päivityksiä myös palvelimen suunnalta. Esimerkkinä voidaan pitää esimerkiksi sellaista ohjelmistoa, joka käyttää tietonaan pörssikursseja, ja tarvitsee mahdollisimman reaaliaikaisia tietoja muutoksista. WebSocketin heikkoutena on lisätty tietty monimutkaisuus HTTP:hen verrattuna, sekä rajatummat kommunikaatiomahdollisuudet tiettyjen systeemien kanssa (Exosite 2016). HTTP-protokollan ratkaisuissa asiakasohjelma ei tiedä, onko palvelimella asiakasohjelmalle relevanttia tietoa, vaan se joutuu tietyin väliajoin tiedustelemaan palvelimelta, onko sille tullut uutta, relevanttia tietoa (nk. *HTTP poll*). HTTP-protokollan ympärille on kehitetty kuitenkin ratkaisuja, jotka mahdollistavat pseudokaksisuuntaisen kommunikaation (esim. *HTTP long polling*). HTTP long pollingissa asiakasohjelma lähettää pyynnön palvelimelle. Asiakasohjelma jää odottamaan

palvelimen vastausta, ja palvelin jättää vastaamatta siihen asti, kunnes sillä on uutta tietoa lähetettävänä. Tällöin palvelin voi lähettää asiakasohjelmalle tietoja heti, kun sillä on uutta tietoa lähetettävänä.

HTTP-protokollaan kuuluu eri pyyntömetodit (*request methods*), joista yleisimpiä ovat GET ja POST. Käytännössä pyyntömetodi tarkoittaa tietyn tiedon lisäämistä pyynnön yhteyteen, jonka avulla pyynnön vastaanottava taho tunnistaa minkä tyyppinen pyyntö on kyseessä. Karkea yksinkertaistus on, että GET-metodia käytetään, kun tiedustellaan tietoa ja POST-metodia käytetään, kun lähetetään tietoa (MDN Web Docs 2022).

Systeemissä ei käytännössä tarvittaisi WebSocket-protokollan reaaliaikaista tiedonpäivittämistä. Toteutuksessa kuitenkin käytettiin WebSocketia palvelimen ja data-analyysisovelluksen välisen kommunikaatioon. Syyksi tähän nähtiin mahdollinen tuleva tarve reaaliaikaiselle kommunikaatiolle, mikäli systeemin toiminnallisuudet laajenisivat reaaliaikaisempaan tiedon päivittämiseen. Esimerkkinä tästä olisi esimerkiksi reaaliaikainen liikkeen kuvaaminen työpöytäsovelluksessa. Tällaista ei prototyypissä kuitenkaan toteutettu. Tuossa tapauksessa VR-laitteisto ja palvelin eivät siltikään tarvitsisi välillensä WebSocket-yhteyttä, sillä niiden välillä vain yhdensuuntainen reaaliaikainen datan lähettämiseen olisi edelleen tarpeellista.

5 VALMIS PROTOTYYPPI: VR-SOVELLUS, PALVELIN JA TIETOKANTA

5.1 Unityllä toteutettu VR-sovellus

Prototyypin VR-sovellus on demo datan keräämiseen ja lähettämiseen, eikä se toteuta mitään muuta tarkoitusta, kuten esimerkiksi hyötysovelluksen tai pelin tarkoitusta. Sovellus koostuukin käytännössä kahdessa osasta: datan keräämisestä sekä datan lähettämisestä palvelimelle.

5.1.1 Liikedatan kerääminen ja tallennus VR-sovelluksessa

Liikedata saadaan VR-ohjaimista Unityllä tehdyssä sovelluksessa käyttämällä UnityEngine.XR-rajapintaa. Kuvassa 1 nähdään esimerkki siitä, miten rajapinnasta saadaan vasemman käden ohjaimesta sen hetkisen kiihtyvyyden arvo. Rajapinnan avulla ensin haetaan viite vasemman käden ohjaimeen, ja sen jälkeen viitteestä voidaan tiedustella ohjaimen sen hetkistä kiihtyvyyttä. Sovelluksessa voidaan kutsua kiihtyvyyden arvon tiedustelua tietyin väliajoin, esimerkiksi viisi kertaa sekunnissa, ja tallentaa tietoja välimuistiin. Samalla tapaa toimii muiden arvojen kerääminen, joita ovat sijainti ja nopeus. Kiihtyvyys, sijainti ja nopeus ovat sovelluksessa kolmiulotteisia vektoriarvoja. Näiden lisäksi tallennetaan aika-arvoja. Prototyypissä datan tallentaminen laukaistiin painamalla vasemman käden ohjaimen *Trigger*-nappulaa.

```
// Get left-hand controller(s)
List<UnityEngine.XR.InputDevice> leftHandedControllers = new List<UnityEngine.XR.InputDevice>();
var desiredCharacteristics = UnityEngine.XR.InputDeviceCharacteristics.HeldInHand |
                             UnityEngine.XR.InputDeviceCharacteristics.Left |
                             UnityEngine.XR.InputDeviceCharacteristics.Controller;

UnityEngine.XR.InputDevices.GetDevicesWithCharacteristics(desiredCharacteristics, leftHandedControllers);

// Get left-hand controller acceleration
leftHandedControllers[0].TryGetFeatureValue(UnityEngine.XR.CommonUsages.deviceAcceleration, out acceleration);
```

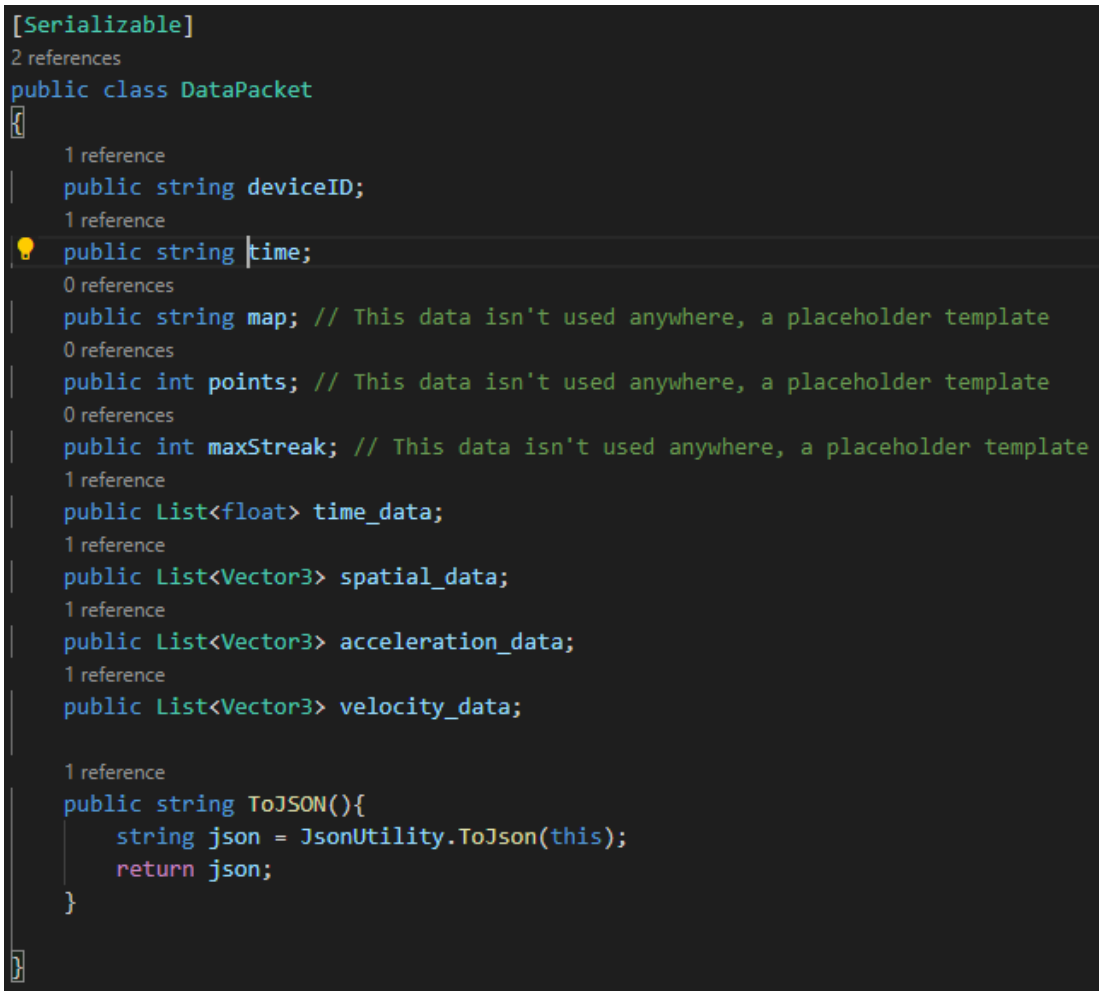
Kuva 1. UnityEngine.XR-rajapinnan avulla saadaan viite vasemman käden ohjaimeen, tai ohjaimiin, sekä saadaan ohjaimesta kyseisen ajanhetken kiihtyvyysarvo.

5.1.2 Datapakettien muodostaminen ja lähettäminen VR-sovelluksesta palvelimelle

Sovellus käyttää yhteydenottoon palvelimelle UnityEngine.Networking -rajapintaa (Unityn dokumentaatio 2020). Prototyypissä data pyrittiin lähettämään automaattisesti sen jälkeen, kun datan tallentaminen pysäytettiin. Kun datan tallennus pysäytetään, ohjelma muodostaa tallennetusta datasta tarkoitusta varten luodun DataPacket-luokan olion. DataPacket-luokka sisältää kentät kerätyn datan tallentamista varten (kuva 2).

DataPacket-luokan määrittelyyn on lisätty *Serializable*-ominaisuus (Unityn dokumentaatio 2022), joka mahdollistaa sen luokan olioiden muuntamisen sellaiseen muotoon, että sen voi tallentaa. Tässä tapauksessa hyödynnetään *Serializable*-ominaisuuden mahdollistavaa datan JSON-muotoon määrittämistä. Kuvassa 2 nähdään DataPacket-luokan määrittely, jossa on määritelty metodi ToJSON(), joka palauttaa kyseisen luokan olion JSON-muotoisena tekstinä. JSON (JavaScript Object Notation) on kevyt dataformaatti, jota käytetään yleisesti tietoliikenteen viestinnässä (json.org www-sivusto 2022). Kuvasta 2 nähdään myös, että luokassa on määritelty myös tietoja nimeltä *map*, *points* ja *maxStreak*, jotka luotiin valmiiksi mahdollisia tulevia toimintoja varten. Kyseisessä työssä niitä ei kuitenkaan hyödynnetty. DataPacket-olioon tallennetaan mitattujen arvojen lisäksi myös VR-laseille ominainen uniikki ID (*deviceID*) sekä olion muodostamisen ajankohta (*time*).

Kun tallennettu liikedata saadaan JSON-muotoisena, voidaan se lähettää käyttämällä UnityEngine.Networking -rajapinnan avulla palvelimelle. Palvelimella JSON-muotoinen data on helppo purkaa ja käsitellä halutulla tavalla.



```
[Serializable]
2 references
public class DataPacket
{
    1 reference
    public string deviceID;
    1 reference
    public string time;
    0 references
    public string map; // This data isn't used anywhere, a placeholder template
    0 references
    public int points; // This data isn't used anywhere, a placeholder template
    0 references
    public int maxStreak; // This data isn't used anywhere, a placeholder template
    1 reference
    public List<float> time_data;
    1 reference
    public List<Vector3> spatial_data;
    1 reference
    public List<Vector3> acceleration_data;
    1 reference
    public List<Vector3> velocity_data;

    1 reference
    public string ToJSON(){
        string json = JsonUtility.ToJson(this);
        return json;
    }

}
```

Kuva 2. DataPacket-luokan määrittely.

5.2 Web-palvelin ja tietokanta

Prototyypin Node.js:llä toteutettu web-palvelin toteuttaa kolmea tehtävää. Se vastaanottaa ja lähettää dataa, se tallentaa ja pyytää dataa tietokannasta ja se laskee joitakin tilastollisia arvoja valmiiksi työpöytäsovellukselle. MongoDB-tietokanta yksinkertaisesti ottaa vastaan dataa ja tallentaa ne dokumentteina, sekä lisäksi tarjoaa dokumentteja, kun sille tehdään pyyntöjä. Web-palvelin ja tietokanta molemmat ajettiin lokaalisti sovelluksen kehittäjän tietokoneella, koska se oli riittävä ratkaisu prototyypin luomiseen. Todellisessa tuotteessa web-palvelin ja tietokanta pitäisi laittaa erilliselle palvelinlaitteistolle tai pilvipalveluun ajettavaksi.

5.2.1 Web-palvelimen toteutus

Node.js-ympäristö toimii hyödyntäen eri moduuleita, jotka tarjoavat eri toiminnallisuuksia. Näiden moduuleiden hallintaan käytettiin projekteissa npm-hallintatyökalua. HTTP-yhteyksien mahdollistavaan palvelintoiminnallisuuteen käytettiin moduulia *http*, WebSocketin vastaavaan *websocket* sekä MongoDB:n käyttöön *mongodb*-moduulia.

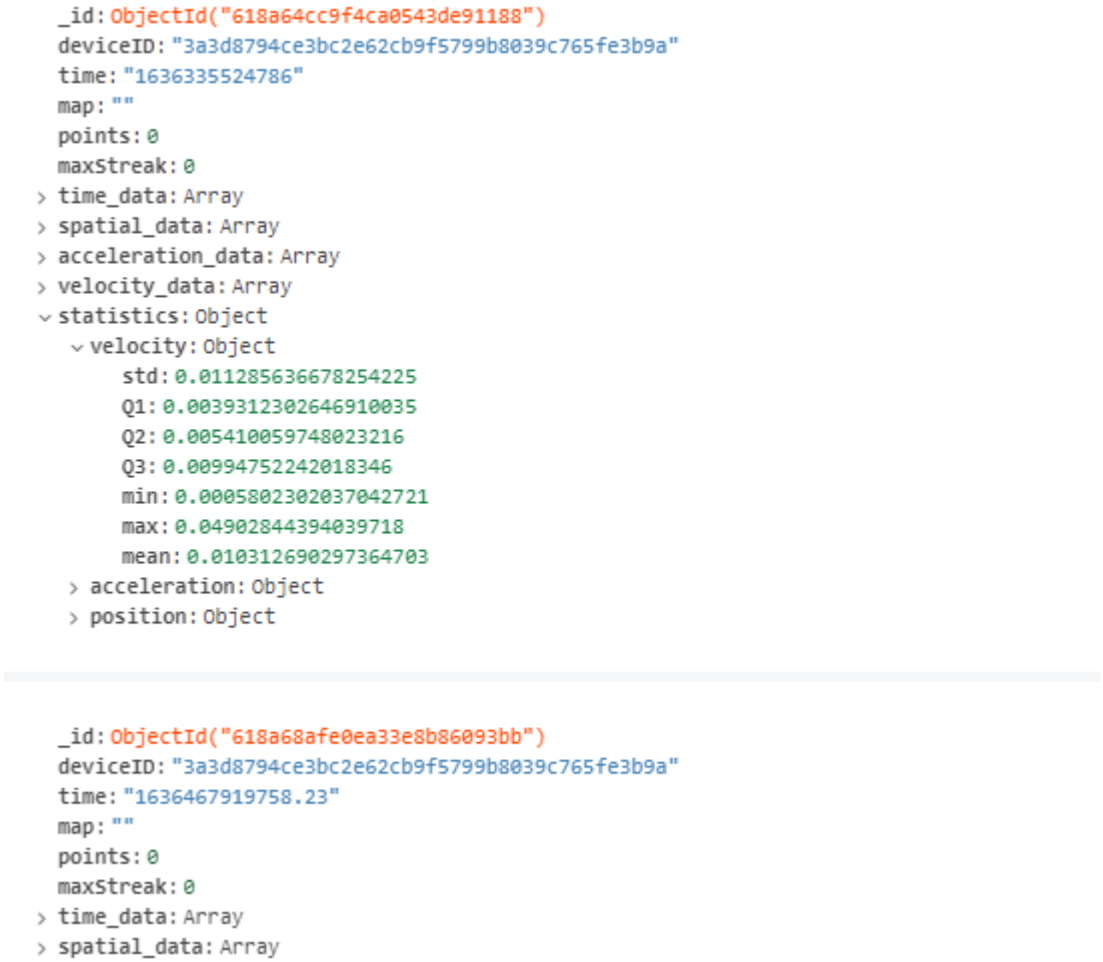
Palvelimelle luotiin erillisiä endpointteja vastaamaan eri pyyntöihin. Kun palvelin saa HTTP-protokollan mukaisen POST-pyynnön, se olettaa yhteydenottajan olevan VR-laitteisto, joka lähettää dataa. Toisia saapuvia pyyntöjä ovat WebSocket-protokollan pyynnöt. WebSocket-protokollan pyynnöistä parsittiin eri avainsanoja pyynnön alusta, joiden mukaan pyynnön tyyppi tunnistetaan. Sanojen parsiminen pyyntöjen alusta ei ole välttämättä parhain tapa eritellä pyyntöjä, vaan pyynnöt olisivat voineet olla esimerkiksi JSON-formaatissa ja omata oman parametrin pyyntötyyppiä kohden. Parempia tapoja saattaisi olemassa, mutta toteutuksessa tyydyttiin tähän, sillä tarkoituksena oli luoda toimiva prototyyppi, josta tarvittaessa voidaan jalostaa finalisoidumpi kokonaisuus myöhemmin. Tekijän osaaminen web-rajapintojen luomiseen projektia toteuttaessa ei myöskään ollut kovin kypsynyt.

Kun palvelin saa HTTP POST-pyynnön, se parsii saadun JSON-datan, muuntaen sen JavaScript-objektiksi. Luodusta objektista rakennetaan DataStatistics-olio, jonka luokka luotiin projektia varten palvelimelle suorittamaan joidenkin tilastollisten arvojen laskemista datasta työpöytäsovellukselle valmiiksi. Tässä luokassa muunnetaan ensin VR-laseilta saadut kolmiulotteiset nopeus- ja kiihtyvyysvektoriarvot skalaarimuotoisiksi. Se tehdään tilastollisten arvojen laskemisen mahdollistamiseksi. Nopeudelle, kiihtyvyydelle ja sijainnille lasketaan keskihajonta, minimi, maksimi, keskiarvo sekä alakvartiili, keskikvartiili ja yläkvartiili. Keskikvartiili on myös mediaani. Tämän jälkeen vastaanotetusta JSON-datasta muodostettuun JavaScript-objektiin lisätään ”statistics” nimisen avaimen alle lasketut tilastoarvot. Tämän muotoisena data lähetetään tietokantaan. POST-pyynnön yhteydessä myös tarkastetaan, onko palvelimella avoinna olevia WebSocket-yhteyksiä työpöytäsovelluksien kanssa, ja jos on, niin päivitetään vastaanotettu data myös niille automaattisesti.

WebSocket-pyynnöt palvelimelle on jaoteltu kahdenlaisiksi. Pyynnössä on joko ’RequestDataList’ avainsana tai ’RequestDataMs’ avainsana. RequestDataList on työpöytäsovellukselta saapuva pyyntö, jossa pyydetään listausta kaikista tietokantaan tallennetuista dataseteistä. Yksi datasetti tarkoittaa VR-laitteistolta tullutta DataPacket-pakettia, johon on lisätty palvelimella laskettuja tilastollisia arvoja. Jokaiseen datasettiin on tallennettu arvo niiden luomisen ajankohdasta millisekuntien tarkkuudella, ja tätä käytetään datasettejä erottelevana tekijänä. Erottelevana tekijänä voisi käyttää myös parempia arvoja, mutta prototyypin käytössä se toimi riittävän hyvin. Palvelin vastaa pyyntöön siis listauksella datasettien nauhoittamisen ajankohdista. RequestDataMs pyyntö sisältää mukanaan ajankohdan millisekuntien tarkkuudella. Palvelin tarkastaa löytääkö se tietokannasta datasetin, jossa olisi pyyntöä vastaava aika. Jos vastaava datasetti löytyy, palvelin pyytää dokumentin tietokannasta, muuntaa sen taas JSON-muotoiseksi ja lähettää sen työpöytäsovellukselle vastauksena.

5.2.2 Tietokannan toteutus

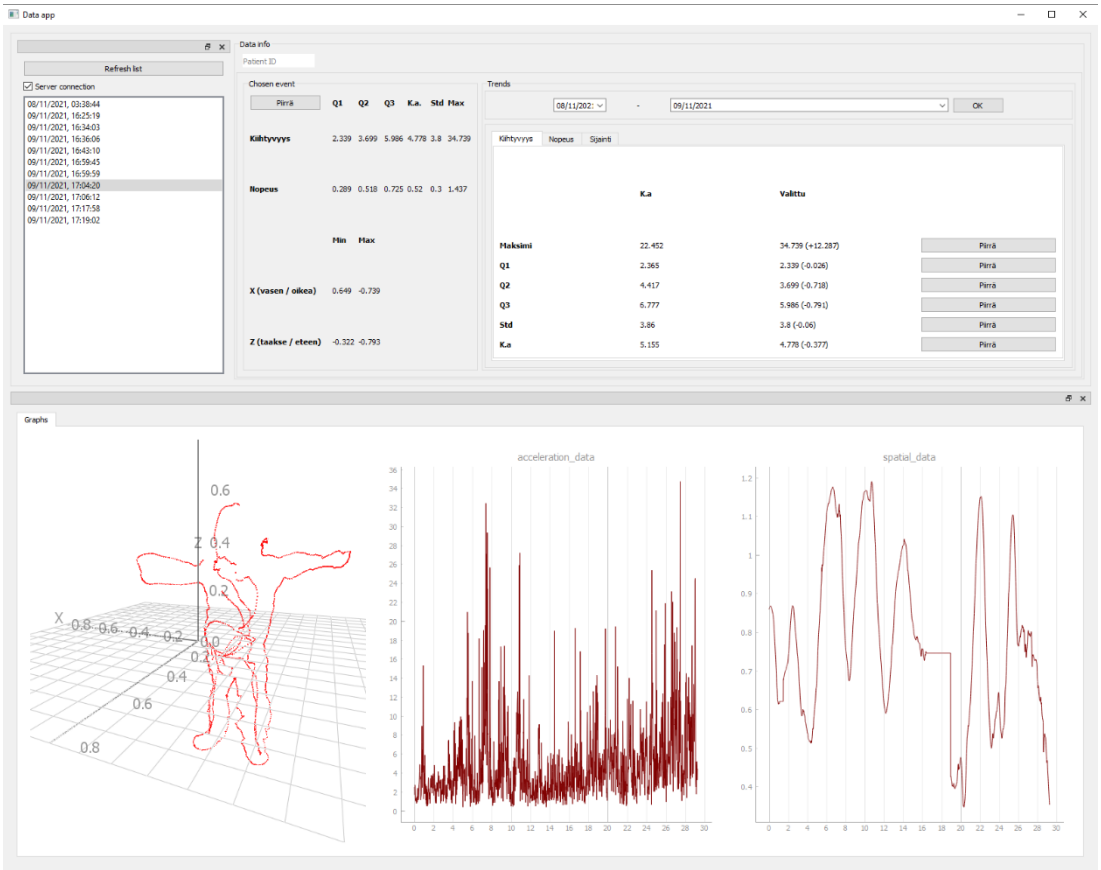
MongoDB-tietokantaa ajettiin paikallisesti kehittäjän tietokoneelta. Koska NoSQL-tietokannat eivät vaadi skeemaa toimiakseen, eli tietokantaan voi tallentaa minkä muotoista tietoa tahansa, oli tietokannan toteutus lähes triviaali. Tietokanta asennettiin tietokoneelle (MongoDB-dokumentaatio 2022) ja asetettiin käyntiin. Tämän jälkeen tietokanta oli palvelimen käytettävissä. Kuvassa 3 on osanäkymä MongoDB:n mukana tulevaan työkaluun MongoDBCompass, jonka avulla voi selata ja hallita tietokantoja graafisen käyttöliittymän avulla. Kuvassa näkyy prototyypin tietokantaan tallennettua esimerkkidataa. Huomataan *deviceID*, joka on VR-laitteiston uniikki id, *time*, joka on datan muodostamisen ajankohta VR-laitteistossa (millisekuntimuodossa) sekä taulukot datapisteen nauhoituksen aika-, sijainti- ja kiihtyvyysdatasta sekä statistics-objekti, jonka alta löytyy palvelimen tallentamat tilastolliset arvot.



Kuva 3. Otanta MongoDBCompass-ohjelman näkymästä tietokannasta.

6 VALMIS PROTOTYYPPI: DATA-ANALYYSIN TYÖPÖYTÄSOVELLUS

Työpöytäsovellus data-analyysin työkaluksi toteutettiin Pythonilla sekä käyttämällä PyQt:n toiminnallisuuksia GUI:n sekä tietoverkkoyhteyksien toteuttamiseen. Koska työkalu on tekniseltä toteutukseltaan pitkällinen ja monimutkainen, ei sen teknistä toteutusta kuvata tässä työssä, vaan rajoitetaan tarkastelu sovelluksen toiminnallisuuksiin ja ulkoasuun. Kuvassa 4 on näkymä sovelluksesta. Sovelluksella ei ole muita näkymiä, vaan koko sovellus on käytön kannalta kuvassa esillä. Näkymä voidaan jakaa eri osiin, joilla on erilaiset, mutta toisiinsa kytkeytyvät toiminnallisuudet, ja joihin työssä tutustutaan seuraavaksi. Näkymiksi voidaan ajatella datasettien listauksen näkymän, datasettien tilastollista tietoa esittävän näkymän sekä datan kuvaajia esittävän näkymän.

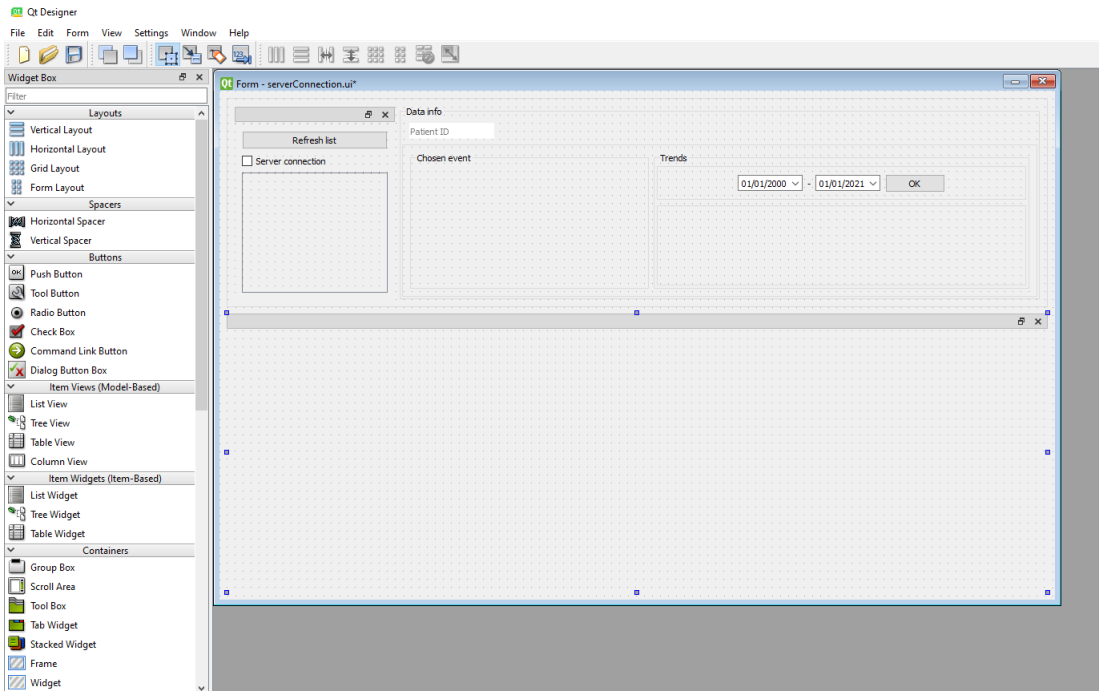


Kuva 4. Näkymä työpöytäsovelluksesta.

6.1 Graafisen käyttöliittymän toteutus Qt Designer -työkalulla

Qt Designer on Qt-ohjelmistokehystä tukeva työkalu, jolla on helppo suunnitella käyttöliittymiä Qt-ohjelmistokehystä hyödyntäviin ohjelmiin. Tässä tapauksessa PyQt-ohjelmistokehystä käyttävään ohjelmaan. Qt Designer -työkalussa suunniteltavaan käyttöliittymään lisätään nk. *widgettejä*, jotka ovat erilaisia näkymiä tai funktionaalisia komponentteja käyttöliittymässä. Tällaisia ovat esimerkiksi nappulat (Qt-kehys: *QPushButton*) tai näkymä, jota voi vierittää hiirellä (Qt-kehys: *QScrollArea*) riippuen siitä, onko siellä niin paljon sisältöä, ettei kaikki mahdu näkymään.

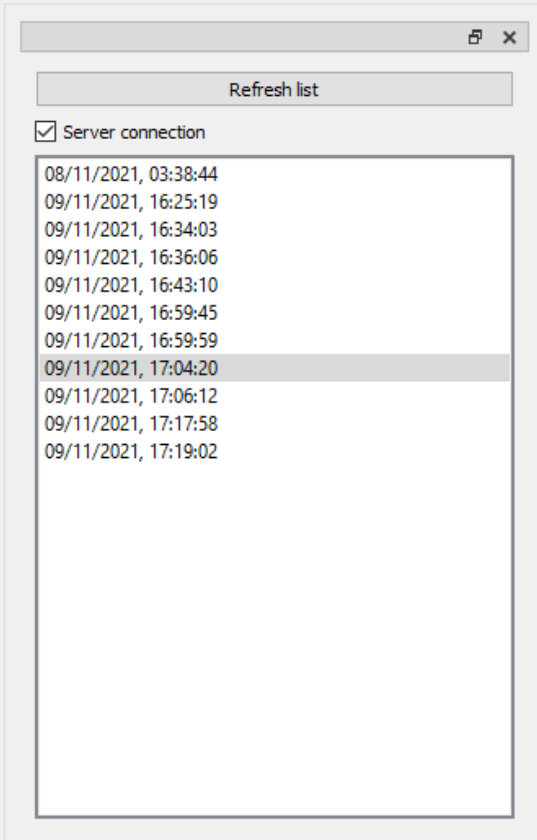
Kuvasta 5 nähdään näkymä suunnitellusta käyttöliittymästä Qt Designer -ohjelmassa. Kuvasta voi nähdä, että näkymä muistuttaa hyvin vahvasti kuvan 4 näkymää, mutta näkymä on populoitu datalla ja teksteillä. Käyttöliittymä suunniteltiin prototyypille juuri riittäväksi ja pragmaattiseksi, eikä ulkoasuun juuri panostettu.



Kuva 5. Sovelluksen UI Qt Designer -työkalussa.

6.2 Listatut datasetit

Kuvassa 4 vasemmassa yläkulmassa näkyy listaus palvelimelta palautetuista datasetistä, joista jokainen kuvastaa yhtä VR-laitteiston nauhoittamaa datan nauhoituskertaa. Käytännössä voitaisiin jokainen listauksen rivi ajatella esimerkiksi yhtenä kuntoutusharjoituskertana. Kuvassa 6 on listan näkymä lähemmin esitettynä. Listauksen kokonaisuuteen kuuluu 1) Refresh list -painike, 2) Server connection -indikaattori sekä 3) listatut datasetit.



Kuva 6. Palvelimelta saadut datasetit listattuna.

Refresh list -painike lähettää palvelimelle pyynnön WebSocket-protokollaa käyttäen, sisällyttäen pyyntöön avainsanan ”RequestDataList”. Pyynnöstä palautuu palvelimelta listaus kaikkien datasettien nauhoitusajoista millisekunteina, jonka jälkeen ohjelma päivittää datasettien listauksen vastauksen mukaan. Tämän yhteydessä myös pyydetään sellaisten datasettien dataa, jota ohjelmalla ei vielä mahdollisesti ole.

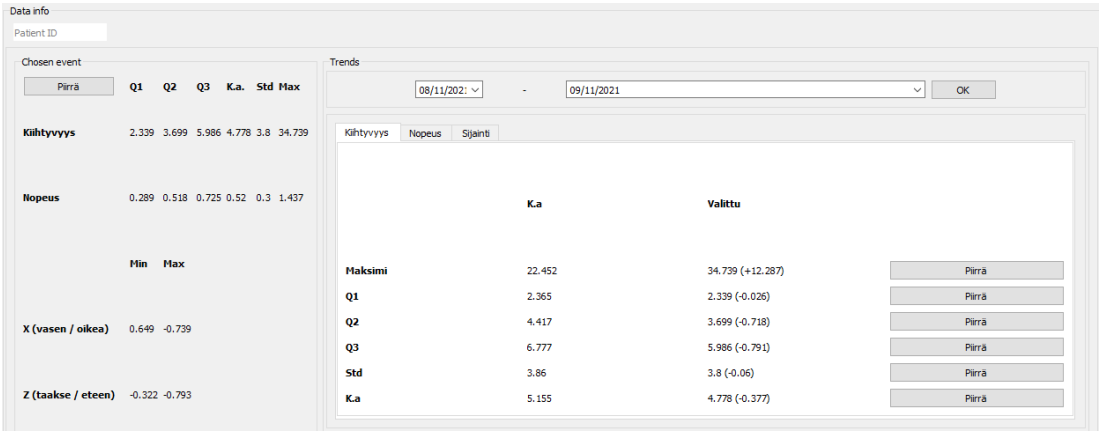
Server connection -indikaattori, eli kuvassa 6 valintaruutu (engl. *checkbox*), kertoo käyttäjälle, onko WebSocket-yhteys palvelimeen olemassa. Valintaruudun klikkaamisella ei ole toiminnallisuutta, vaan ohjelma ’ruksittaa’ tai poistaa ’ruksin’ valintaruudusta palvelinyhteyden mukaan. Indikaattorin voisi toteuttaa paremmin, sillä valintaruutu implikoi käyttäjälle kontrollia yhteyden olemassaoloon, mutta näin ei ole. Tässäkin asiassa oli kyse asiaa lähestyminen toimivan prototyypin kannalta.

Listatut datasetit ovat nousevassa aikajärjestyksessä, esitettynä aikaformaatissa *pv/kk/v, t:min:s*. Esitetty ajankohta listauksessa tarkoittaa milloin datasetti on nauhoitettu VR-laitteistoissa. Listauksessa voi kursorilla painamalla valita eri listan yksilöitä, joiden valitsemisesta aiheutuu ohjelmassa muissa näkymissä muutoksia.

Sisäisesti ohjelma pyytää datasettien listausta palvelimelta saadessaan ensimmäisen kerran yhteyden palvelimeen, eli yleensä ohjelman käynnistyessä. Listauksen saatuaan ohjelma pyytää listauksen mukaan jokaisen datasetin datan, ja ne vastaanottaessaan tallentaa ne sisäisesti välimuistiin.

6.3 Yksittäisen datasetin informaatio sekä trendit

Kuvan 4 näkymän yläkeski-yläoikea-alueella on näkymä, jonka otsakkeena on ”Data info”. Tämän näkymän alta löytyy vasemmasta osiosta näkymä, jossa esitetään yksittäisen datasetin dataan liittyviä tilastollisia arvoja, sekä oikean puolimmaisesta osiosta toinen näkymä, jossa esitetään tietyn aikajakson sisällä olevien datasettien tilastollisten arvojen trendejä. Kuvassa 7 on lähempi otos tästä näkymästä.

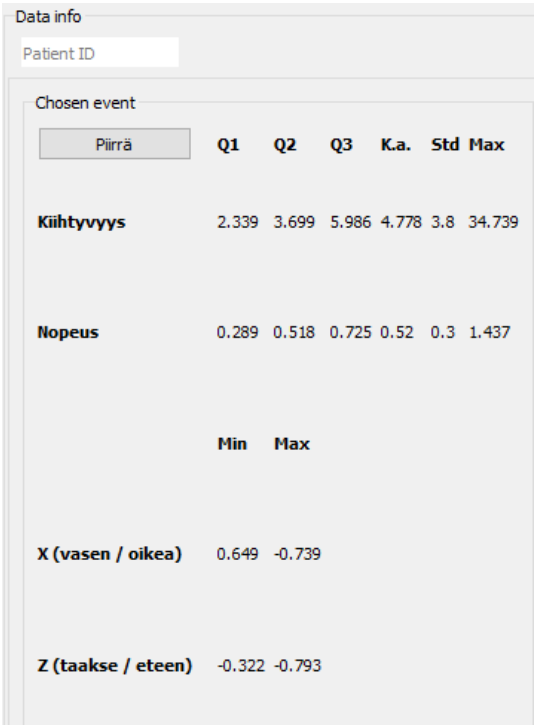


Kuva 7. Data info -näkymä, jossa näkyy yksittäisen datasetin tietoja sekä trenditietoja dataseteistä jonkin ajanjakson ajalta.

6.3.1 Yksittäisen datasetin tilastollisten tietojen näkymä

Kuvan 7 vasemman puolimmaisessa osiossa on yksittäisen datasetin näkymä, alaotsakkeella ”Chosen event”. Yksittäisen datasetin näkymästä voidaan lukea erilaisia tilastollisia arvoja datasettiin liittyen (kuva 8). Näitä arvoja ovat kiihtyvyyden ja nopeuden kvartiileja (*Q1, Q2, Q3*), keskiarvoja (*K.a.*), keskihajontaa (*Std*) sekä maksimiarvoja (*Max*). Sen lisäksi siitä voidaan tulkita datan sijaintien ääriarvoja x (vasen / oikea)- sekä z (eteen / taakse)-akseleilta. Yksiköt arvoille ovat kullekin arvolle yleisesti käytettyjä, esimerkiksi nopeudelle m/s (metriä per sekunti). Yksiköiden puuttuminen on kuitenkin selkä puute. Datasetti voidaan valinta datasettien listauksesta kursorilla valitsemalla. Sen lisäksi näkymässä on Piirrä-painike, josta painettaessa sovellus piirtää valitun datasetin kuvaajien näkymään (kuva 11).

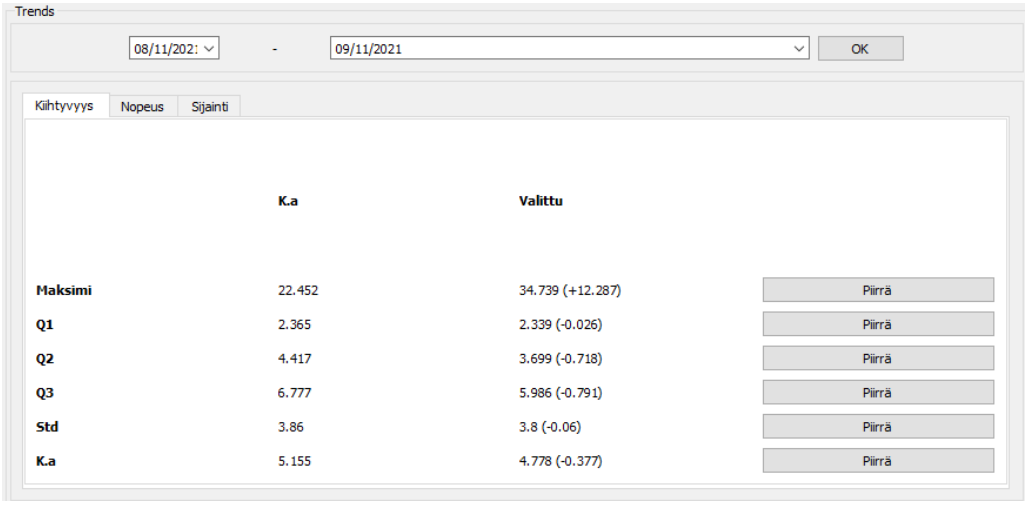
Näkymässä on myös tekstikenttä ”Patient ID”, joka on tarkoitettu sovelluksen käyttäjälle mahdollisuudeksi eritellä eri kuntoutujien datasetit keskenään. Kentän olisi tarkoitus karsia listanäkymästä sellainen data, joka ei kuulu kuin halutulle kuntoutujalle. Tätä toiminnallisuutta ei kuitenkaan prototyypissä implementoitu.



Kuva 8. Chosen event -näkymä.

6.3.2 Tilastollisten trendien näkymä

Kuvan 7 oikean puolimmaisessa osuudessa on näkymä alaotsakkeella ”Trends” (kuva 9). Näkymässä voi tarkastella erilaisia trendejä valitun ajanjakson ajalta usean datasetin kesken. Näkymän tarkoituksena on tarkastella eri arvojen, esimerkiksi kiihtyvyyden alakvartiilien kehittymistä tietyn ajanjakson aikana, eli käytännössä kuntoutusprosessin aikana.



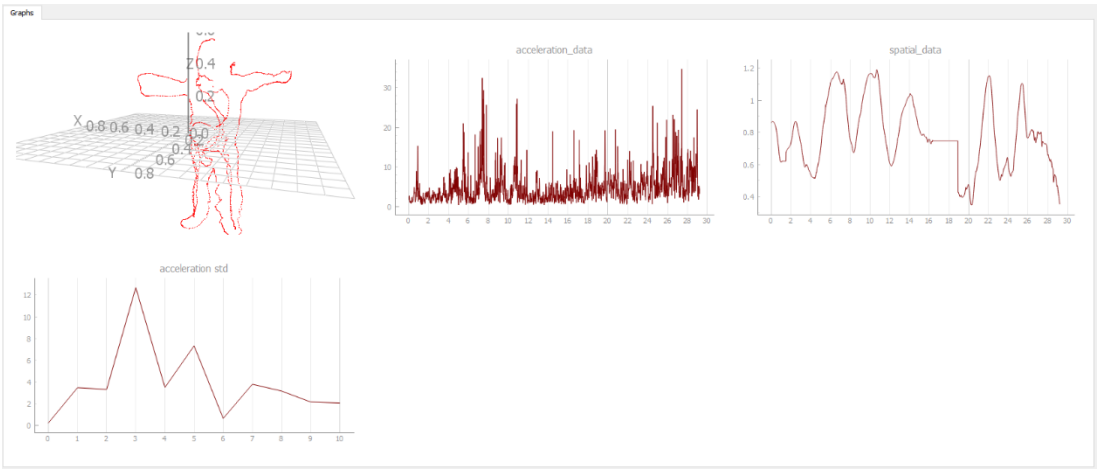
Kuva 9. Trends-näkymä.

Trends-näkymä voidaan jakaa kahteen osaan, joista ylemmässä osassa voidaan valita ajanjakso, jota halutaan tarkastella, ja joista alemmassa voidaan tarkastella ajanjakson arvoja. Alemmassa osiossa voidaan kolmesta välilehdestä erikseen valita, tarkastellaanko kiihtyvyyden, nopeuden vai sijainnin trendejä ajanjakson ajalta. Sijainti-välilehden trendit ovat prototyypissä toteuttamatta. Valitusta välilehdestä, kuten esimerkiksi kiihtyvyyden välilehdestä, löytyy kiihtyvyyden tilastollisten arvojen, kuten maksimin tai keskihajonnan keskiarvoja, sekä valitun datasetin vastaava arvo. Valitun datasetin vastaavan arvon jälkeen löytyy myös suluissa datasetin vastaavan arvon erotus koko ajanjakson datasettijoukon keskiarvoon, siten, että plusmerkki ennen lukua tarkoittaa valitun datasetin vastaavan arvon olevan keskiarvoa luvun verran isompi. Miinusmerkki ennen lukua tarkoittaa valitun datasetin vastaavan arvon olevan datasettijoukon keskiarvoa luvun verran pienempi. Esimerkiksi kuvassa 9 rivillä ”Maksimi” kolumnissa ”K.a” arvo on 22.452, ja kolumnissa ”Valittu” on arvo 34.739 (+12.287). Tämä tarkoittaa, että ajanvälin datasettijoukon kiihtyvyyden maksimin keskiarvo on 22.452 (m/s^2) ja datasetti listasta valitun datasetin arvo on 34,739 (m/s^2), joka on 12.287 (m/s^2) joukon keskiarvoa suurempi.

Tämän lisäksi jokaisella rivillä kiihtyvyys- ja nopeusvälilehdillä on Piirrä-painike, joka piirtää valitun ajanjakson välille sijoittuvien datasettien sen mukaiset arvot, miltä riviltä ja välilehdeltä painiketta on painettu (kuva 14). Esimerkiksi Kiihtyvyys-välilehdeltä ja Std-riviltä painettu Piirrä-painike aiheuttaa valitun ajanjakson datasettien kiihtyvyyksien keskihajonta-arvojen piirtymisen sovelluksen näkymään, johon kuvaajat piirtyvät.

6.4 Kuvaajien näkymät

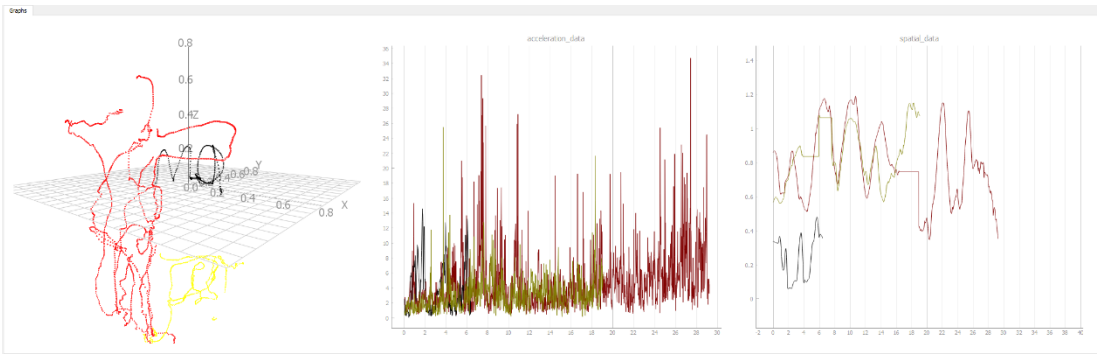
Kuvassa 4 sovelluksen alaosassa on alue sovelluksen piirtämille kuvaajille, alaotsakkeella ”Graphs”. Kuvaajat piirretään käyttäen Pythonin kirjastoa pyqtgraph. Kuvassa 10 näemme lähemmän kuvan Graphs-näkymästä. Kuvaajanäkymiä piirtyy lisääntyvästi eri määrä käyttäjän piirtokäskyjen mukaisesti. Esimerkiksi kuvassa 10 on neljä kuvaajanäkymää ja kuvassa 11 on kolme kuvaajanäkymää. Ohjelma sijoittaa kuvaajanäkymiä käytettävissä olevalle ruutupinta-alalle, muuttaen näkymien kokoja siten, että kaikki mahtuvat käytettävissä olevaan tilaan.



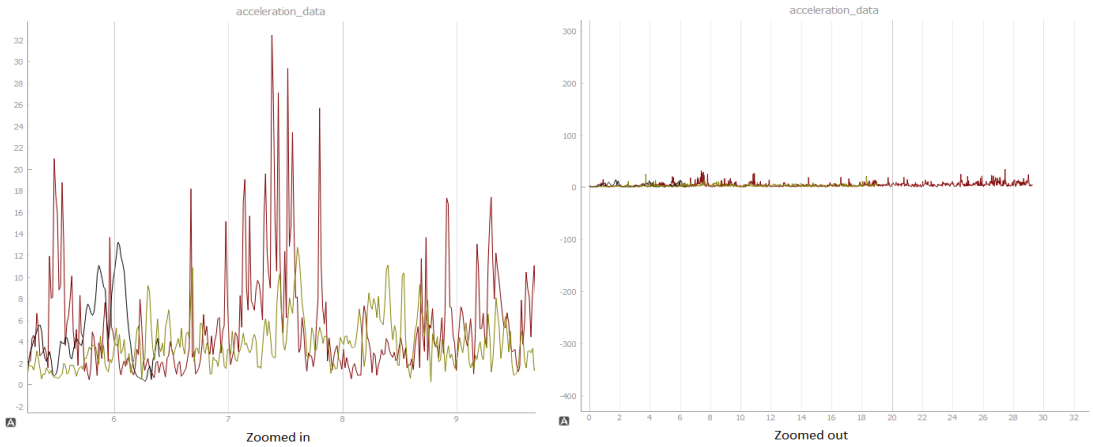
Kuva 10. Graphs -näkymä.

Kuvassa 10 nähdään kahdenlaisia kuvaajia: ylävasemmalla, järjestyksessä ensimmäinen, on 3D-kuvaaja ja sen jälkeen kolme 2D-kuvaajaa. Voidaan myös sanoa, että siinä on neljä kuvaajanäkymää, ja kuvaajalla tarkoitetaan tietyn datan kuvaajaa kuvaajanäkymässä. Kolme ylimmäistä ovat kuvaajanäkymiä, jotka syntyvät, kun yksittäisen datasetin Piirrä -painiketta painetaan Chosen event -näkymässä. Näitä ovat 3D-kuvaaja sijaintidatasta, 2D-kuvaaja kiihtyvyyden arvoja ajan suhteen ja 2D-kuvaaja skalaarietäisyydestä mittaustilanteen lähtösijainnista ajan suhteen. 3D-kuvaaja esittää siis missä ohjain on sijainnut 3D-avaruudessa. 2D-kuvaajilla on sisältöään kuvaavia otsikoita, kuten esimerkiksi ”acceleration_data”.

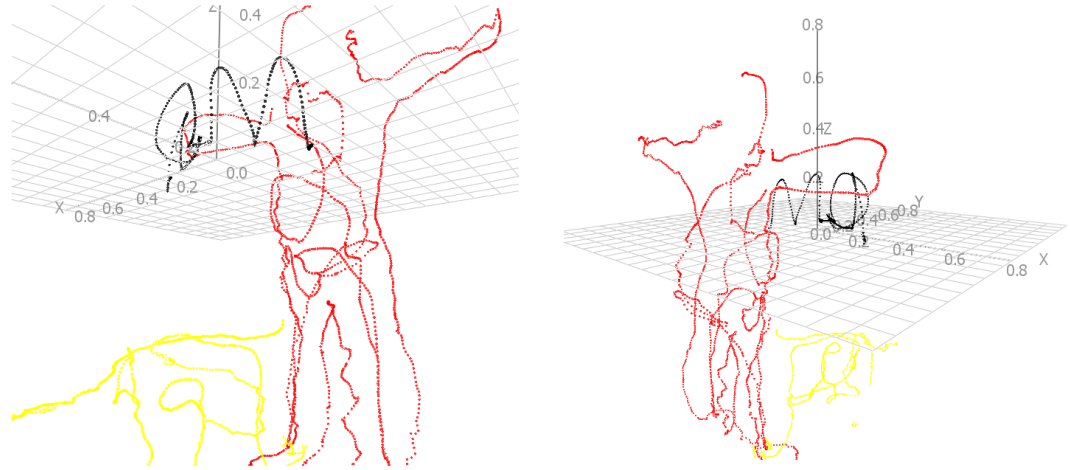
Yksittäisten datasettien kuvaajat piirtyvät myös päällekkäin. Kun eri dataseteistä painetaan Piirrä -painiketta, piirtyvät näiden datasettien datat jaetuille kuvaajanäkymille (kuva 11). Kuvaajien värit vaihtuvat datasettien välillä, mutta pysyvät kuvaajanäkymien väleillä konsistentteinä: esimerkiksi punainen väri kuvastaa jokaisessa kolmessa kuvaajanäkymässä samaa datasettiä. 2D-kuvaajia pystyy zoomaamaan sisään ja ulos, sekä levittämään akseleiden skaaloja halutusti (kuva 12). 3D-kuvaajissa ”kameraa” pystyy siirtämään, kiertämään ja zoomaamaan (kuva 13). 2D- ja 3D-kuvaajien zoomauksen ja muun muutoksen hallinta tapahtuu hiiren ja näppäimistön yhdistelmällä, käyttäen hiiren vasenta ja oikeaa nappia sekä rullaa, sekä näppäimistöstä control-näppäintä.



Kuva 11. Kolmen eri datasetin kuvaajat piirrettynä samoille kuvaajanäkymille.



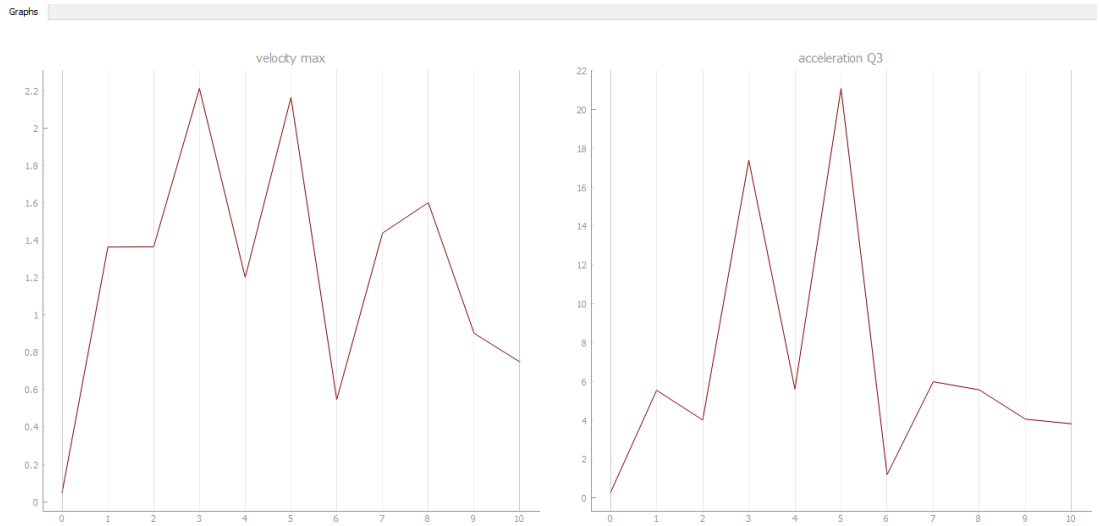
Kuva 12. Samat kuvaajat eri zoomauksen tasoilla. ”Zoomed in” ja ”Zoomed out” tekstit lisätty jälkikäteen.



Kuva 13. Sama 3D-kuvaajanäkymä kameran eri sijainnilla ja rotaatiolla.

Trendien kuvaajat piirtyvät omille kuvaajanäkymilleen, ja nekin otsikoituvat piirrettävän datan mukaisesti. Kuvassa 14 nähdään tietyn ajanjakson trendien kuvaajat nopeuden maksimiarvoille (*velocity max*) sekä kiihtyvyyden yläkvartiileille (*acceleration*

Q3). Pystyakseli kuvastaa haluttuja tilastollisia arvoja ja vaaka-akseli kuvastaa mittaustuloksia ajan mukaan järjestyksessä, eli datapisteitä vastaavia datasettejä aikajärjestyksessä. Vaaka-akselilla on siis nollasta lähtien järjestyslukuja dataseteille mittausjärjestyksen mukaan. Varsinaiset mittausajankohdat olisi siis tulkittava ajanjakson aikaväliä ja aikavälille sijoittuvia datasettejä tulkiten.



Kuva 14. Nopeuden maksimiarvot sekä kiihtyvyyden yläkvartiilien arvot ajanjakson aikana.

7 POHDINTA

Työssä tähdättiin minimitasolla toimivaan prototyyppiin tai demoon, ja mielestäni siinä onnistuttiin. Dataa kerätään, tallennetaan ja saadaan esitettyä tilastollisina arvoina ja kuvaajina työpöytäsovelluksessa. Varsinaisen järjestelmän toteuttaminen sisälsi kuitenkin todella paljon uuden opettelua, ja prototyypissä on paljon asioita, jotka teknisesti toteuttaisin eri tavalla. Se kuitenkin on prototypoinnin hyvä puoli; asiat voi toteuttaa eri tavalla seuraavassa iteraatiossa. Tulokseksi saatiin kuitenkin aikaiseksi toimiva prototyyppi ja laaja kokonaisuus, joka demonstroi liikedatan tallentamisen VR-laitteistosta sekä samaisen datan hyödyntämisen mahdollisuutta.

7.1 Työpöytäsovelluksen puutteet tai kehittämiskohteet

Prototyypissä on selkeät puutteensa. Kokemattomuuteni vuoksi en osannut suunnitella tai ennakoida kaikkia asioita riittävän hyvin, minkä johdosta jotkin tekniset toteutukset saattoivat arveluttavia tai niin kutsuttuja ”purkkavirityksiä”. Sen lisäksi ajankäyttöä oli optimoitava ja raja vedettävä johonkin, jonka seurauksena tiettyjä asioita ei ehditty hiomaan tai välttämättä edes toteuttamaan.

7.1.1 Yleiset puutteet

Yksi isoimmista puutteista työpöytäsovelluksessa on yksiköiden puuttuminen sovelluksesta kokonaan. Vaikka monissa tapauksissa yksikön voi päätellä kontekstista, kuten esimerkiksi nopeuden yksikön voi arvata olevan m/s, ei tämä menettely ole kuitenkaan sopiva työkalun käyttämiseksi ammatillisessa ympäristössä.

Selkeä puute prototyypissä on myös kyvyttömyys aidosti erottaa eri potilaat toisistaan, tai eri VR-laitteistot toisistaan. Tietokantaan tallennetaan datasettien yhteyteen tieto VR-laitteistojen uniikeista id:stä, jota voitaisiin käyttää apuna datan erottelussa eri sovellusten käyttäjien välillä. Työpöytäsovelluksen käyttäjä ei kuitenkaan pysty tietämään lasiensa uniikkia id:tä helposti, eikä myöskään pystyisi halutessaan sitä edes sovellukseen syöttämään. Jos demoa haluisi käyttää useampi kuntoutuksen ammattilainen tai edes yksi ammattilainen usean eri kuntoutujan kanssa, systeemi ei toimisi. Tätä varten pitäisi systeemin luoda jokin malli käyttäjistä, ja tapa erotella kuntoutujat toisistaan. Konseptidemona tämä toimii juuri ja juuri näin ”kovakoodattunakin”, mutta mikäli prototyyppiä haluttaisiin lähteä testaamaan kentälle, olisi tuon muuttaminen yksi ensimmäisistä muutettavista asioista.

Tämän lisäksi on iso kysymysmerkki, että ovatko työpöytäsovelluksessa olevat tilastolliset arvot ja kuvaajat millään tapaa merkityksellisiä indikaattoreita kuntoutusprosessin kulusta. Näiden lisäksi ohjelma vaatisi paljon hiomista, esimerkiksi kielenkäytön yhtenäistämistä, valiten kieleksi joko englannin tai suomen, eikä molempia sekaisin. Myös ulkoasua, luettavuutta ja käytettävyyttä tulisi hioa yleisesti.

7.1.2 Kuvaajien puutteet

Kuvaajien toteutuksessa on puutteensa tai kehittämiskohteensa. Kuvaajanäkymiä piirtyy dynaamisesti käytettävissä olevalle tilalle siten, että kaikki kuvaajanäkymät mahtuvat tilaan. Kuvaajanäkymiä ei kuitenkaan saa prototyypissä poistettua yksittäisesti, vaan toteutuksessa kuvaajanäkymiä piirtyy maksimissaan kuusi, jonka jälkeen yrittäessä piirtää seitsemättä kuvaajanäkymää kaikki poistuvat. Tämän jälkeen kuvaajia voi piirtää taas normaalisti. Toiminnallisuus yksittäisten kuvaajanäkymien poistamiseen, tai erillisestä painikkeesta kaikkien kuvaajanäkymien poistamiseen, lisäisi käyttäjän kontrollia sovelluksen toiminnasta.

Toinen merkittävä puute kuvaajissa on akselien yksiköiden puuttuminen. Kuvaajista olisi oleellista pystyä lukemaan aika-akselin yksiköt, esimerkiksi sekunteja, minuutteja tai trendikuvaajissa päivämääriä. Tämän lisäksi pystyakselista olisi oleellista pystyä lukemaan myös käytetyt yksiköt, kuten m/s^2 kiihtyvyydelle tai m/s nopeudelle. 3D-kuvaajassa akseleiden yksiköt ovat metreissä, mutta sitäkään ei kuvaajassa ole ilmaistu.

Trendien kuvaajissa heikkoutena on myös se, että datapisteet esitetään tasaisin välein vaaka-akselilla datasettien aikajärjestyksen mukaisesti. Heikkous tulee siitä, että tasavälit implikoivat sovelluksen käyttäjälle sitä, että mittaukset olisi tehty tasaisin väliajoin. Todellisuudessa tietyn kahden datasetin tallentamisen aika ero voi olla esimerkiksi yksi päivä, ja toisen kahden tallentamisen aikaero esimerkiksi kuukausia. Tasaiset välit helposti siis välittävät väärää informaatiota työkalun käyttäjälle. Vaaka-akselille sijoittaminen tulisi siis tehdä tallennusajan mukaan, eikä järjestyksen mukaan tasavälein. Tämän jälkeen kuvaajat kuvaisivat kuntoutusprosessia trendien avulla todenmukaisemmin.

Hyvänä lisätoiminnallisuutena yksittäisten datasettien kuvaajissa olisi käyttäjän voida valita hiirellä eri kuvaajia silloin, kun yhdessä kuvaajanäkymässä on useiden eri datasettien kuvaajia. Kuvaajan valitessa nähtäisiin kuvaajaa vastaavan datasetin tietoja, kuten esimerkiksi sen datan keräämisen ajankohta. Tämä helpottaisi kuvaajien vertailua, joka lähtökohtaisesti on samalle kuvaajalle piirtämisen tarkoitus. Samoin monen kuvaajan näkymissä olisi myös huomioitava kuvaajien piirtovärien riittävä luettavuus.

Esimerkiksi kuvassa 13 keltainen (kuvassa alin pistejoukko) kuvaaja eri erotu valkoista taustaa vasten riittävän hyvin. Olisi myös hyvä tarkastella sitä, että onko samalle kuvaajanäkymälle kuvaajien piirtäminen työkalussa edes lähtökohtaisesti vertailua helpottavaa, mutta toisaalta tässä työssä ei ollut tarkoitus tarkastella kyseisen sovelluksen data-analyysin hyödyllisyyttä kuntoutusprosessissa ylipäätänsäkään.

7.2 Lopuksi

Opin työtä tehdessä paljon. Node.js tai ylipäätänsä web-palvelimen teko, PyQt, Unityllä VR-sovellusten tekeminen, MongoDB tai ylipäätänsä NoSQL, olivat kaikki itselleni enemmän tai vähemmän uusia tuttavuuksia. Sen lisäksi ylipäätänsä näin laajan kokonaisuuden suunnittelu yksin oli varsin opettavainen kokemus. Suurimman opin koen olleen suunnitelmallisuuden ja esimerkiksi rajapintojen määrittelyn tärkeydestä. Vajavainen määrittely tai suunnittelu saattavat aiheuttaa lisää työtunteja, sekä tuottavat helposti epäsuoraan huonompaa koodia. Toisaalta, koska sain aikaan toimivan demon, ja opin paljon uutta, koen projektin olleen, puutteistaan huolimatta, menestys.

LÄHTEET

Alsop, T. 2022. Virtual reality (VR) headset unit sales worldwide in 4th quarter 2019 and 4th quarter 2020, by device. Viitattu 20.05.2022. https://www.statista.com/statistics/987701/vr-unit-sales-brand/

Chand, M. What is C#. Päivitetty 07.03.2020. Viitattu 20.05.2022. https://www.c-sharpcorner.com/article/what-is-c-sharp/

Choudhary, V. Difference between HTTP and WebSocket (HTTP 2.0). Viitattu 10.01.2022. https://developerinsider.co/difference-between-http-and-http-2-0-websocket/

Concepta Technologies LLC:n www-sivut 2017. NoSQL vs SQL: Which is Right For Your Project? [INFOGRAPHIC]. Viitattu 10.01.2022. https://www.conceptatech.com/blog/nosql-vs-sql-which-is-right-for-your-project

Exosite 2016. Embedded IoT Protocols: WebSocket's Benefits and Drawbacks. 10.5.2016. Viitattu 10.01.2022. https://blog.exosite.com/embedded-iot-protocols-websocket

Fitzpatrick, M. 2019. PyQt5 vs PySide2. Päivitetty 18.09.2021. Viitattu 10.01.2022. https://www.pythonguis.com/faq/pyqt5-vs-pyside2/

Jalonen, H. 2018. Data puhuu, mutta kuulemmeko? Viitattu 20.05.2022. https://www.harrijalonen.fi/fi/niita_naita/data_puhuu_mutta_kuulemmeko

Jost, T.A., Nelson, B. & Rylander, J. 2021. Quantitative analysis of the Oculus Rift S in controlled movement. Disability and Rehabilitation: Assistive Technology 16:6, 632-636. https://doi.org/10.1080/17483107.2019.1688398

json.orgin www-sivusto. Introducing JSON. Viitattu 20.05.2022. https://www.json.org/json-en.html

Kaneriya, T. 2020. Advantages & Disadvantages of Node.js : Why to Use Node.js? Viitattu 10.01.2022. https://www.simform.com/blog/nodejs-advantages-disadvantages/

Kelan www-sivusto 2021. Kelan kuntoutusetuudet. Kuntoutuspalveluja ja kuntoutusrahaa saaneet 1968-2020. Viitattu 10.01.2022. https://www.kela.fi/kelan-tilastot-kuvina#Kuviot-sarja

Master's in Data Sciencen www-sivut 2022. What is Data Analytics? Päivitetty 4.2022. Viitattu 20.05.2022. https://www.mastersindatascience.org/learning/what-is-data-analytics/

MDN Web Docs-dokumentaatio 2022. HTTP. Päivitetty 8.5.2022. Viitattu 20.05.2022. https://developer.mozilla.org/en-US/docs/Web/HTTP

MongoDB-dokumentaatio. Install MongoDB. Dokumentaation versio 5.0. Viitattu 20.05.2022. https://www.mongodb.com/docs/manual/installation/

Nasa Advanced Supercomputing (NAS) Divisionin www-sivut. Virtual Reality: Definition and Requirements. Viitattu 20.05.2022. https://www.nas.nasa.gov/Software/VWT/vr.html

Oracle WSIT-dokumentaatio 2010. What is a Server-Side Endpoint? Viitattu 20.05.2022. https://docs.oracle.com/cd/E17802_01/webservices/webservices/reference/tutorials/wsit/doc/Initialization2.html

Oraclen www-sivut. What Is a Database? Viitattu 20.05.2022. https://www.oracle.com/database/what-is-database/

Passos, D.E. & Jung, B. 2020. Measuring the Accuracy of Inside-Out Tracking in XR Devices Using a High-Precision Robotic Arm. Communications in Computer and Information Science 1224, 19-26. https://doi.org/10.1007/978-3-030-50726-8_3

Schaefer, L. 2022. NoSQL vs SQL Databases. Viitattu 10.01.2022. https://www.mongodb.com/nosql-explained/nosql-vs-sql

Shum, L.C., Valdés, B.A. & Van der Loos, H.M. 2019. Determining the Accuracy of Oculus Touch Controllers for Motor Rehabilitation Applications Using Quantifiable Upper Limb Kinematics: Validation Study. JMIR Biomedical Engineering 2019, 4. https://doi.org/10.2196/12291

Sosiaali- ja Terveysministeriö 2022. Asiakas- ja potilastiedot. Viitattu 10.01.2022. https://stm.fi/asiakastietojen-potilastietojen-salassapito

Stack Overflow Developer Survey Results 2018. Viitattu 10.01.202. https://insights.stackoverflow.com/survey/2018/

Tampereen korkeakouluyhteisön www-sivusto. Virtuaalikuntoutusverkosto. Viitattu 20.05.2022. https://projects.tuni.fi/virtuaalikuntoutusverkosto/

The World Bankin www-sivusto. DataBank, Suomen populaatio. Viitattu 10.01.2022. https://data.worldbank.org/country/finland?view=chart

Unity-dokumentaatio 2020. Namespace UnityEngine.Networking. Dokumentaation versio Multiplayer HLAPI 1.0.8. Viitattu 20.05.2022. https://docs.unity3d.com/Packages/com.unity.multiplayer-hlapi@1.0/api/UnityEngine.Networking.html

Unity-dokumentaatio 2022. Script serialization. Dokumentaation versio 2021.3 (LTS). Viitattu 20.05.2022. https://docs.unity3d.com/Manual/script-Serialization.html

Unity-dokumentaatio 2022. XR API reference. Dokumentaation versio 2021.3 (LTS). Päivitetty 15.05.2022. Viitattu 20.05.2022. https://docs.unity3d.com/Manual/VRReference.html

Weis, S. 2018. An overview of roomscale tracking technologies. Viitattu 10.01.2022. https://packet39.com/blog/2018/10/07/roomscale-tracking-technologies/