LAPIN AMK
Lapland University of Applied Sciences

TEKOÄLY ASIAKKAANA

Terveyskeskuksen puhelinpäivystys -aiheinen selainpeli

Mustonen Sami

Opinnäytetyö

Tieto- ja viestintätekniikka
Insinööri (AMK)

2021

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tieto- ja viestintätekniikka
Insinööri (AMK)

Tekijä	Sami Mustonen	Vuosi	2021
Ohjaaja	Petri Hannula		
Toimeksiantaja	OmaDigi-hanke		
Työn nimi	Tekoäly asiakkaana Terveyskeskuksen puhelinpäivystys -aiheinen selainpeli		
Sivumäärä	46		

Tämän työn tarkoituksena oli luoda prototyyppi terveyskeskuksen puhelinpäivystykseen liittyvästä selainpohjaisesta pelistä osana hoitoalan digitalisaatioon liittyvää OmaDigi-hanketta. Työn tavoitteena oli rakentaa modulaarinen pelin arkkitehtuuri ja laajennettavissa oleva tekoäly, joka toimii pelissä asiakkaan roolissa ja kykenee keskustelemaan pelaajan kanssa omista oireistaan.

Työ toteutettiin käyttäen ohjelmointikielenä JavaScriptiä ja Reactia, jotka soveltuivat hyvin selainpohjaisen pelin kehittämiseen. Tekoäly rakennettiin siten, että se kykenee yhteydenottosyyn ja pelaajan syöttämästä lauseesta irrotettujen avainsanojen perusteella tunnistamaan keskustelun aiheen ja vastaamaan siihen asianmukaisesti. Tutkimusmateriaali koostuu lähinnä tiedeartikkeleista, kirjoista ja tutkimuksista, jotka liittyvät tekoälyyn, digitalisaatioon ja projektissa käytettyihin teknologioihin.

Lopputulokseksi saatiin käyttöliittymältään yksinkertainen pelin prototyyppi, jossa pelaaja toimii terveyskeskuksen puhelinpäivystäjänä ja ottaa vastaan puheluita tekoälyasiakkailta. Pelin arkkitehtuuri on modulaarinen, ja se koostuu pienemmistä komponenteista. Tekoäly kykenee esittelemään itsensä ja vastaamaan vointiinsa liittyviin kysymyksiin. Se on tarvittaessa laajennettavissa kattamaan myös muita teemoja kuin terveydenhuolto.

Avainsanat: avainsanat, digitalisaatio, JavaScript, lausegenerointi, pelillistäminen, tekoäly

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

Degree Programme in Information and Communication Technology
Bachelor of Engineering

Author Sami Mustonen Year 2021
Supervisor Petri Hannula
Commissioned by OmaDigi-project
Subject of thesis Artificial Intelligence as a Customer
A Browser Game on the Health Center's Customer Service
Number of pages 46

The main purpose of this thesis was to create a prototype of a browser-based game which is related to health center's customer service. The thesis was part of the OmaDigi-project related to the digitalization of healthcare. The aim of the thesis project was to build modular game architecture and expandable artificial intelligence that acts in the role of a customer and is capable of discussing its own symptoms with the player.

The thesis study was carried out using JavaScript and React as programming languages, which were well suited for the development of a browser-based game. Artificial intelligence was built to be able to identify the topic of conversation and respond appropriately to it, based on the keywords detached from the sentences. The research material consists mainly of scientific articles, books and studies related to artificial intelligence, digitization and the technologies used in the project.

The end result was a prototype of the game with simple interface where the player acts as a nurse in healthcare customer service and receives calls from artificial intelligence customer. The architecture of the game is modular and consists of smaller components. Artificial intelligence is able to present itself and answer questions related to its wellbeing. If necessary, it can be extended to cover themes other than healthcare.

Key words artificial intelligence, digitalization, gamification, JavaScript, keywords, sentence generation

SISÄLLYS

KÄYTETYT LYHENTEET JA TERMIT

algoritmi	toimintaohje jonkin ongelman ratkaisemiseen tai palvelun tuottamiseen
asynkroninen	valmiin sovelluksen osia suoritetaan eri järjestyksessä, kuin mitä koodissa lukee
back-end	koodi, joka ajetaan sivuston palvelimella
CSS	Cascading Style Sheets, WWW-dokumentin tyyliohjekieli
data	tieto, aineisto
funktio	aliohjelma, joka suorittaa osan koodista
front-end	kaikki se koodi, joka ajetaan verkkoselaimessa
generaattori	aliohjelma, joka suorittaa tietyn osan ohjelman koodista ja palauttaa arvoja tai objektin
HTML	HyperText Markup Language, hypertekstin merkintäkieli
indeksi	numero, jolla on suora yhteys taulukon arvoon
JavaScript	ohjelmointikieli, jota yleensä käytetään selainscriptien tekemiseen
JSON	JavaScript Object Notation, yksinkertainen avoimen standardin tiedostomuoto tiedonvälitykseen
loop	ohjelmakoodin toistorakenne
luokka	elementti, joka sulkee sisälleen olion tarvitseman tiedon ja tarjoaa metodeja, joiden avulla luokan sisäistä tietoa päästään muuttamaan
muuttuja	paikka muistissa, johon säilötään tieto ja sille annetaan nimi
objekti	ajonaikainen elementti, joka voi sisältää erilaista tietoa
React	JavaScript -kirjasto
renderöinti	tiettyjen kappaleiden tai kuvien esittämistä näytöllä
repository	säilytyspaikka, mihin Git -versionhallinnan tieto tallennetaan
taulukko	eng. array. monta peräkkäistä muuttujaa, jotka on indeksoitu yhden taulukon alle
tekoäly	ihmisen käyttäytymistä ja älykkyyttä jäljittelevä toiminto

1 JOHDANTO

Digitalisaatio muuttaa yhteiskuntaa monella eri tasolla. Sillä on vaikutusta ihmisten toimintaan yksilötasolla, työelämässä ja koulutuksessakin. Digitalisaatiossa on kyse oivalluksesta, miten tietotekniikan avulla voidaan omaa toimintaa muuttaa voimakkaastikin toisenlaiseksi. Suomi on kansainvälisesti terveyden- ja hyvinvoinnin sähköisen tiedonhallinnan kärkimaita. Suuria haasteita on edessä, sillä väestön ikääntyessä tuen tarve lisääntyy ja julkishallinnon kasvava alijäämä pakottaa vastaamaan tähän yhä pienemmillä taloudellisilla resursseilla. Tähän digitalisaation hyödyntäminen toimii yhtenä ratkaisukeinona, kun haetaan parempaa tuottavuutta, vaikuttavuutta ja kustannustehokkuutta. (Sosiaali- ja terveysministeriö 2016, 4 – 5.)

Tämä opinnäytetyö on tehty toimeksiantona Osaamisen mahdollistaminen digitaalisissa hyvinvointipalveluissa (OmaDigi)-hankkeelle. Tavoitteena on luoda toimiva prototyyppi terveyskeskuksen puhelinpäivystykseen liittyvästä selainpohjaisesta pelistä, jota voisi tulevaisuudessa käyttää esimerkiksi yhtenä työkaluna hoitoalan koulutuksessa. Tekoälyn avulla pelaajalle voitaisiin luoda aidontuntuinen kokemus asiakaskohtaamisesta, jossa hoidon tarpeen arviointia pystyy turvallisesti harjoittelemaan.

Pelin arkkitehtuurin tulee olla modulaarinen, jossa front-end ja back-end pidetään toisistaan erillään. Pelissä tulee olla jatkokehitettävissä oleva tekoäly, joka toimii asiakkaan roolissa ja kykenee keskustelemaan hoitajan roolissa toimivan pelaajan kanssa omasta tilanteestaan. Työn aikana on tarkoitus perehtyä syvemmin asiakastekoälyn lausegenerointiin, jotta se kykenisi toimimaan asianmukaisesti, ja vastaamaan terveydenhuollon puhelinpäivystykseen liittyvän opetuspelin tarpeisiin.

Tämän opinnäytetyön teoriaosuus keskittyy digitalisaation, pelillistämisen ja käytettyjen työkalujen lisäksi erityisesti chatbottien ja tekoälyn lauseiden käsittelyyn. Sen jälkeen opinnäytetyössä esitellään pelin toteutus, ja kerrotaan testaamisesta ja niiden palautteista.

2 DIGITALISAATIO JA PELILLISTÄMINEN

Sosiaali- ja terveysministeriön yhteisessä digitalisaation visiossa kerrotaan näin: *”Asiakas on tärkein – Parempaa tietoa –parempia valintoja, toimintaa ja palveluita”*. Linjauksien mukaan väestön ikääntyminen ja tuen tarpeen lisääntyminen aiheuttavat muutosta myös perinteisemmille aloille, kuten terveydenhoitoalalle. Linjauksissa todetaan hallintoalan olevan suurien muutoksien kynnyksellä ja uusia digitalisoituneita palveluita voitaisiin tarjota esimerkiksi tekoälyn ja robotiikan avulla. (Sosiaali- ja terveysministeriö 2016, 8.)

Osaamisen mahdollistaminen digitaalisissa hyvinvointipalveluissa eli OmaDigi-hanke on Pohjois-Pohjanmaan ELY-keskuksen ja Lapin ammattikorkeakoulun ESR-rahoitteinen hanke. Sen tavoitteena on ollut vastata Sairaanhoitajaliiton laatimiin sähköisten terveyspalveluiden strategiaan vuosille 2015 – 2020, sekä Terveyden ja hyvinvoinnin laitoksen (THL) digitalisaation hallintaan liittyviin suosituksiin. Tarkoituksena on kehittää terveydenhuollon ammattilaisten digiosaamista ja digitalisaation käyttöönottoon liittyvää koulutusta siten, että se liitetään tiiviisti aitoihin tai sitä jäljitteleviin asiakastilanteisiin. (Tieranta 2020.) Terveydenhuollon ammattilaisilla on ollut haasteita pysyä ajan tasalla alan viimeisimmistä tiedoista. Alalla käytetyt perinteiset opetusmenetelmät ovat osoittautuneet tehottomiksi, ja kiinnostus pelien käyttämiseen koulutuksessa on ollut kasvussa. (Ribeiro, Antunes, Monteiro & Pereira 2013.)

Digitalisaatio edistää oppimista ja tarjoaa siihen monenlaisia mahdollisuuksia. Pedagogisia mahdollisuuksia ovat esimerkiksi aikojen ja paikkojen suhteen joustavat oppimis-, opetus- ja ohjausmenetelmät, jotka hyödyntävät erilaisia video-, kuva- ja simulaatioita. Oppimateriaalit ovat myös paremmin saavutettavissa, ja osaamisen dokumentoinnissa käytetään yhä enemmän kuvia ja videoita. Digitalisaatio tarjoaa aiempaa henkilökohtaisemmat opintopolut ja mahdollisuuden myös useampien eri koulutuksen järjestäjien tarjontaan. Opettajien näkökulmasta myös ohjausresurssien kohdentaminen tukea tarvitseville oppilaille on digitaalisuuden avulla tehokkaampaa. (Koramo, Brauer & Jauhola 2018, 70.)

Pelit ovat tehokas keino oppia uusia asioita, ja niiden käyttöä oppimisen työkaluna, voidaan kutsua oppimisen pelillistämisenä. Sillä tavoitellaan oppilaan itseohjautuvuuden tehostamista oppimisprosessissa. Pelit rohkaisevat oppimaan, ja kun hauskuus tulee mukaan osaksi oppimista, niin myös motivaatio oppimiseen kasvaa ja stressitasot laskevat. Pelien avulla voidaan myös tarkkailla opiskelijan suoriutumista, ja pelin sisältöä voida muokata vastaamaan paremmin oppijan henkilökohtaisia erityistarpeita ja oppimisnopeutta. (Caballé 2016, 248 – 251.)

Tekoälyyn, pelien simulointeihin, robottiopettajiin ja verkko-opetuksen oppimismenetelmiin kehitetään teknologiaa. Tavoitteena on resurssien säästäminen, kuten opetuksen kustannuksien laskeminen, ja uuden teknologian hyödyntäminen tukee hyvin myös opiskelijoiden itseoppimista. Opiskelijat saattavat myös oppia sellaisiakin asioita, joita opettajalla ei ole välttämättä ollut valmiuksia opettaa. Uhkakuvana on, että nykyisen mallin jatkaminen johtaa oppisisältöjen vähämerkityksisyyteen työelämässä. Mahdollisesti myös opitun soveltamisen kyky jää heikommalle tasolle niillä oppilailla, jotka eivät ole hankkineet tai saaneet oppilaitoksen ulkopuolelta lisäosaamista. Tämä johtaisi tutkintojen arvostuksen laskemiseen. (Valtiovarainministeriö 2017, 39 – 40.)

Työpaikka on houkutteleva kohde pelillistämiselle. Se tekee työstä hauskempaa, motivoivampaa, ja se parantaa myös yrityksen ja organisaation tuloksia. Pelin kehittäjien tulee olla hyvin perillä pelillistämisen kohteena olevasta työtehtävästä, ympäristöstä ja työkaluista, jotta pelistä voidaan rakentaa mahdollisimman motivoiva käyttäjilleen. (Pedreira ym. 2020.) Jotta peli kykenisi integroitumaan paremmin oikean maailman ominaisuuksiin, aiheen pelillistämisessä tulee huomioida erilaiset pelaajien persoonallisuudet ja konteksti. (Siriaraya, Visch, van Dooren & Spijkerman 2018, 3.) Pelillistämisen hyödyt saattavat laskea uutuudenviehätyksen hiipuessa ja siinä vaiheessa jatkokehittämisestä koituu kustannuksia. Muutoin käyttäjät voivat taitojen kehittyessä kokea, että pelin tehtävät ovat liian helppoja. (Basten 2017, 81.)

3 CHATBOTIT, TEKOÄLY JA NLP

3.1 Chatbotit

Chatbotit kehittyvät ja yleistyvät nopeasti. Niillä on kyky ymmärtää mitä käyttäjä sanoo, ja valitsemaan tai luomaan kontekstissa olevan vastauksen syötteeseen. Chatbot joutuu käsittelemään monenlaisia tietohakuja ja sen pitää kyetä ymmärtämään sanojen synonyymit ja asiayhteydet. Kehittäjien tulee ottaa huomioon chatbotin vakaus, laajennettavuus ja mukautuvuushaasteet. (Rahman, Mamun & Islam 2017, 78.) Chatbotit käyttävät luonnollista kielen käsittelyä (NLP, Natural Language Processing) ymmärtämään, mitä käyttäjä kirjoittaa niille ja muodostavat vastauksen keskusteluun. Kattavuus ja monimutkaisuus vaihtelevat paljon käyttötarkoituksen mukaan. (Jylkäs 2020, 30.)

Chatbotit jäljittelevät ihmisten käyttämää kieltä ja kykenevät ymmärtämään ja vastaamaan takaisin. Se tapahtuu siten, että chatbot pyrkii ensin tutkimaan käyttäjän antaman syötteen, ja valikoimaan siitä oleelliset asiat eli syötteen tarkoituksen. Sitten chatbot palauttaa mahdollisimman tarkan vastauksen käyttäjän pyyntöön. Tarkoitus merkitsee käyttäjän lähettämän viestin takana olevaa tarkoitusta eli mitä käyttäjä olettaa saavansa vastaukseksi keskustelusta chatbotin kanssa. (Deepika, Tilekya, Mamatha & Subetha 2020, 1196 – 1197.) Käyttäjän antamaa syötettä verrataan avainsanoihin, ja niistä etsitään vastaavuuksia siten, että niitä löytyy ainakin yhdestä tai useammasta avainsanojen listasta. Jos annettu syöte ei vastaa tavanomaisia tuloksia tai mitään avainsanoja, chatbot palauttaa geneerisen vastauksen. (Shaw 2014, 641.)

3.2 Tekoäly

Tekoäly koostuu erilaisista teknologioista ja sovelluksista, joiden avulla koneet, laitteet, ohjelmat, järjestelmät ja palvelut kykenevät toimimaan asianmukaisella tavalla. Tämä vaatii tekoälyltä erilaisia ominaisuuksia, kuten kykyä tunnistaa eri tilanteita, autonomisuutta, oppivuutta ja suorituskykyä. Tekoälyä on ryhdytty hyödyntämään terveydenhuoltoalalla. Suomessa erilaisilla hoitoalan digitalisoinnin

ohjelmilla pyritään kehittämään laadukkaampia palveluita, tehostamaan toimintoja ja parantamaan asiakaskokemuksia. Tekoäly myös laskee kustannuksia ja parantaa palveluiden saatavuutta. (Rousku ym. 2017, 47; 2019, 27.)

Peleissä tekoälyn tarkoitus on luoda parempi ja realistisempi pelikokemus pelaajille. Se ei välttämättä tarkoita monimutkaisen tekoälyn rakentamista, vaan usein tekoälyt on rakennettu skenaariokohtaisesti siihen rajattuun ympäristöön, jossa ne luovat enemmänkin illuusion älykkäästä käyttäytymisestä. Korkean tason autonomia ja arvaamattomuus, mihin kehittyneemmät tekoälyt kykenevät, koetaan usein tarpeettomiksi peleissä ja se voi jopa olla haitaksi pelattavuudelle. (Petrović 2018, 39979.)

3.3 NLP

NLP (Natural Language Processing) on lyhenne luonnollisen kielen käsittelyn virallisesta määritelmästä. Se on tutkimusala, joka käyttää tietojenkäsittelytieteen tekoälyä ja virallisia kielitieteellisiä käsitteitä luonnollisen kielen analysoimiseksi. Sillä voidaan viitata myös joukkoon työkaluja, joiden avulla saadaan mielekästä ja hyödyllistä tietoa luonnollisista kielilähteistä, kuten verkkosivuilta ja tekstidokumenteista. NLP:tä käytetään monilla eri aloilla erilaisten ongelmien ratkaisemiseksi. Tekstianalyysi suoritetaan tekstille, joka vaihtelee käyttäjän kirjoittamasta muutaman sanan kyselystä kokonaisiin tiivistettäviin asiakirjoihin. Sitä on käytetty esimerkiksi hakukoneiden tulossivujen luomisessa. NLP ei ole helppoa, sillä on olemassa useita satoja luonnollisia kieliä ja jokaisessa on oma kielioppinsa. Joskus sanojen merkitys vaihtelee myös kontekstin mukaan. (Reese & Bhatia 2018, 8 – 9.)

Terveydenhuoltoalalle kehitetyn tekoälyn kielen käsittelyssä täytyy huomioida hoitajien käyttämän ammattikielen eroavaisuudet ja sen standardisoinnissa voi olla haasteta. Pelkästään haavaan liittyvä keskustelu voi poiketa merkittävästi arkielämässä käytetystä kielestä. (Barnard 2007, 497 – 501.)

4 TYÖKALUT JA TEKNIIKAT

4.1 Front-end

Vuonna 1995 Netscapen ohjelmoija Brandan Eich kehitti **JavaScriptin**. Alun perin sen nimi oli Mocha, joka pian vaihtui LiveScriptiksi ja myöhemmin lopulta JavaScriptiksi. Sitä ei kuitenkaan tule sekoittaa Java-ohjelmointikieleen, sillä niillä ei ole juurikaan mitään yhteistä. (Wirfs-Brock & Eich 2020, 77.) JavaScript pitää sisällään myös monenlaisia kirjastoja, esimerkiksi Ember, Angular, React ja Vue (DeGroat 2019).

JavaScript antaa web-kehittäjille sivustolla käytettävän ohjelmointikielen, jonka avulla he voivat suorittaa monenlaisia tehtäviä. Näitä ovat esimerkiksi elementtien ja dokumenttien lukeminen ja uusien elementtien ja tekstin lisääminen dokumentteihin, tekstin muokkaaminen, matemaattisten laskelmien tekeminen, eri tapahtumien mahdollistaminen painikkein, ikkunan koon määrittely tai käyttäjälle virheilmoituksien lähettäminen. (Larsen 2013, 341.)

JavaScript on täysin integroitu toimimaan HTML ja CSS teknologioiden kanssa (Javascript.info 2021). JavaScript mielletään helpoksi ohjelmointikieleksi oppia, ja siihen on saatavilla paljon tukimateriaalia, kuten kokonaisia JavaScript aiheisia nettisivuja, artikkeleita, ohjeita ja YouTube-videoita. (Dhaniwala & Jaimini 2016, 2374.)

HTML (Hyper Text Markup Language) on merkintäkieli, joka määrittelee nettisivun dokumentin rakennetta ja tyyliä. Sen avulla voidaan esimerkiksi erotella tekstiosat otsikoista tai lisätä dokumenttiin taulukoita tiedon säilyttämistä varten. Tarkoituksena sillä on tarjota rakenne, joka tekee dokumentin ymmärtämisestä helpompaa. Sen elementit pitävät sisällään tageja, jotka kertovat kunkin elementin tarkoituksen. (Larsen 2013, 3 – 8.)

CSS (Cascading Style Sheets) määrittelee säännöt elementeille, jotka näkyvät nettisivulla. HTML-dokumentin sisällön ulkoasua voidaan muuttaa, sillä nämä säännöt määrittelevät, miten elementtien sisältö renderöidään ruudulle. CSS-

säännöt ovat kaksiosaisia. Ne pitävät sisällään valinnan, joka kertoo mitä elementtejä sääntö koskee ja määrittelyn eli millainen tyyli valituilla elementeillä täytyy olla. CSS voidaan lisätä osaksi HTML-dokumenttia tai tehdä sitä varten erillinen tiedosto, joka linkitetään HTML-dokumenttiin. (Larsen 2013, 192 – 193.)

ReactJS on Facebookin, sekä yksittäisten kehittäjien ja yritysten muodostaman yhteisön kehittämä avoimen lähdekoodin JavaScript-kirjasto. Sen avulla voidaan rakentaa tehokkaasti ja joustavasti käyttöliittymiä pienemmistä toisista erillään olevista koodikokonaisuuksista, joita kutsutaan komponenteiksi. (React 2021.) React on kehitetty lähinnä ratkaisuksi nettisovelluksien suorituskykyyn liittyviin haasteisiin. Se käyttää virtuaalista DOM:ia eli dokumenttiobjektimallia, joka tekee päätöksen siitä, täytyykö komponentin latautua uudelleen vai ei. Tämä vähentää sovelluksen tarpeettomia uudelleenlatauksia. (Javeed 2019, 1.)

React Router on Reactin päälle rakennettu kirjasto, jota käytetään sovelluksen sisällä rakentamaan reittejä eri näkymiin. Sen avulla voidaan näyttää useampaa näkymää yhtä aikaa yhdellä sivulla. (Javatpoint 2021.)

4.2 Back-end

Node.js on alusta, jolle voidaan rakentaa nettisovelluksia ja servereitä. Sen avulla voidaan kehittää JavaScript-sovelluksia, jotka toimivat nettiselaimen ulkopuolella. Se on suunniteltu äärimmäiseen skaalautuvuuteen verkkosovelluksissa ja asynkronoituun ohjelmointiin. Sen suosio on kasvanut viime vuosien aikana paljon ja esimerkiksi PayPal-yritys on siirtänyt palvelunsa Java-pohjalta Node.js:lle. (Herron 2018, 7 – 9.)

MongoDB on suosittu NoSQL-tietokanta, jonka rakenne muistuttaa hyvin paljon JSON-tiedostomuodon rakennetta. Sen avulla voidaan tallentaa serverille tietoa ja jakaa sitä netin välityksellä eri käyttäjille. MongoDB:ssä määritellään kokoelmat, jotka ovat hyvin samankaltaisia kuin taulukot relaatiotietokannoissa. Erona on, että kokoelmat voivat pitää sisällään erilaisilla rakenteilla olevia objekteja, jotka pitävät sisällään objekti- ja taulukko-ominaisuuksia. (Weil 2017, 45 – 46.)

4.3 Projektinhallinta

Git on avoimen lähdekoodin versionhallintatyökalu, joka mahdollistaa useamman kehittäjän työskentelyn yhtä aikaa saman projektin äärellä. Työskentely tapahtuu paikallisesti, ja muutokset ladataan yhteiseen hakemistoon, jota kutsutaan repositoryksi. Git voi pitää sisällään useita eri haaroja, joita kutsutaan brancheiksi, ja jokaisella niistä on oma historiansa. Projektiin osallistujat tekevät committeja, jotka ovat lisäyksiä master branchissa olevaan varsinaiseen projektiin. Git mahdollistaa palaamisen projektin aiempiin versioihin. (Afzali, Torres-Arias, Curtmola & Cappos 2018, 469 – 470.)

GitLab on netissä oleva Git-versionhallintapalvelu. Se tarjoaa mahdollisuuden avoimien ja suljettujen repositoryjen ylläpitämiseen. Sen avulla ammattilaiset voivat hallinnoida kaikkia projektin tapahtumia suunnittelusta koodin hallintaan, seurantaan ja turvallisuuteen. GitLab auttaa tiimin jäseniä tekemään yhteistyötä ja parantaa tuottavuutta. (Gaba 2021.) GitLab on viime vuosina kasvanut merkittävästi ja varsinkin vuonna 2019 se sai paljon lisää käyttäjiä, kun Microsoft hankki vastaavanlaisen palvelun, GitHubin vuonna 2019. (Safari, Sabri, Shashavan & Bahrak 2020, 194.)

Visual Studio Code tunnetaan yleisesti nimellä VS Code. Se on Microsoftin valmistama avoimen lähdekoodin tekstieditori. Se tukee monia ohjelmointikieliä, kuten Java, C++, Python, CSS, Go, Dockerfile ja JavaScript. Se tarjoaa myös useita valinnaisia lisätyökaluja ohjelmointia varten. (Mustafeez 2021.)

Discord on videopeliyhteisöille suunniteltu, vuonna 2015 julkaistu ilmainen ääni-, video- ja chat-applikaatio, jonka kehittäjä on ollut Discord Inc. Sen avulla käyttäjät voivat keskustella keskenään eri ryhmissä viesteitse ja myös puheluiden avulla. Lisäksi sen avulla käyttäjät voivat jakaa toisilleen materiaalia, kuten tiedostoja tai suorana toistettua videokuvaa. (Discord Inc 2021.)

5 PELIN TOTEUTUS

5.1 Suunnittelu ja kehitysympäristö

Heti alkuvaiheessa oli selvää, että pelin prototyypin tulisi olla nettiselaimessa toimiva sovellus. Tavoitteena oli helppo käytettävyys ja jaettavuus testaamisenkin kannalta. Ensin asiakastekoälylle suunniteltiin monenlaisia ominaisuuksia ja asioita, jotka voisivat vaikuttaa sen käytökseen ja generoitujen lauseiden muotoon. Asiakastekoälylle ideoitiin esimerkiksi erilaisia persoonallisuuksia, ja niiden vaikutusta lausegeneroinnin sisältöön ja puhenopeuteen.

Hyvin nopeasti persoonallisuus kuitenkin jouduttiin rajaamaan ensimmäisen prototyypin ulkopuolelle, sillä se olisi monimutkaistanut lauserakenteiden suunnittelua ja lisännyt lauseiden esikirjoittamiseen liittyvää työmäärää merkittävästi. Neutraali lauserakenne päätettiin pitää ja jättää esimerkiksi vihaisen tai hiljaisen asiakastekoälyn lauseet tässä vaiheessa pois, mutta kuitenkin siten, että ne ovat mahdollisia lisätä kohtuullisella jatkokehityksellä myöhemmin. Neutraalia lauserakennetta tuki myös ajatus sovelluksen eettisyydestä, sillä todentuntuista illuusiota hakiessa päätettiin generoida asiakastekoälylle oikeita nimiä ja henkilötunnuksia.

Alkuvaiheessa suunniteltiin, mikä ratkaisu toimisi parhaiten pelaajan lauseen rakentamiseen. Vaihtoehtoina olivat pelaajalle esikirjoitettujen lauseiden monivalinta, sana kerrallaan rakentuva lausevalinta tai vapaa tekstin syöttäminen. Päädyttiin vapaaseen tekstin syöttämiseen, sillä se antaisi enemmän testimateriaalia myös tekoälyn lausegeneroinnin kehittämiseen ja peliä olisi tarkoitus testata myös hoitoalalla olevilla testaajilla.

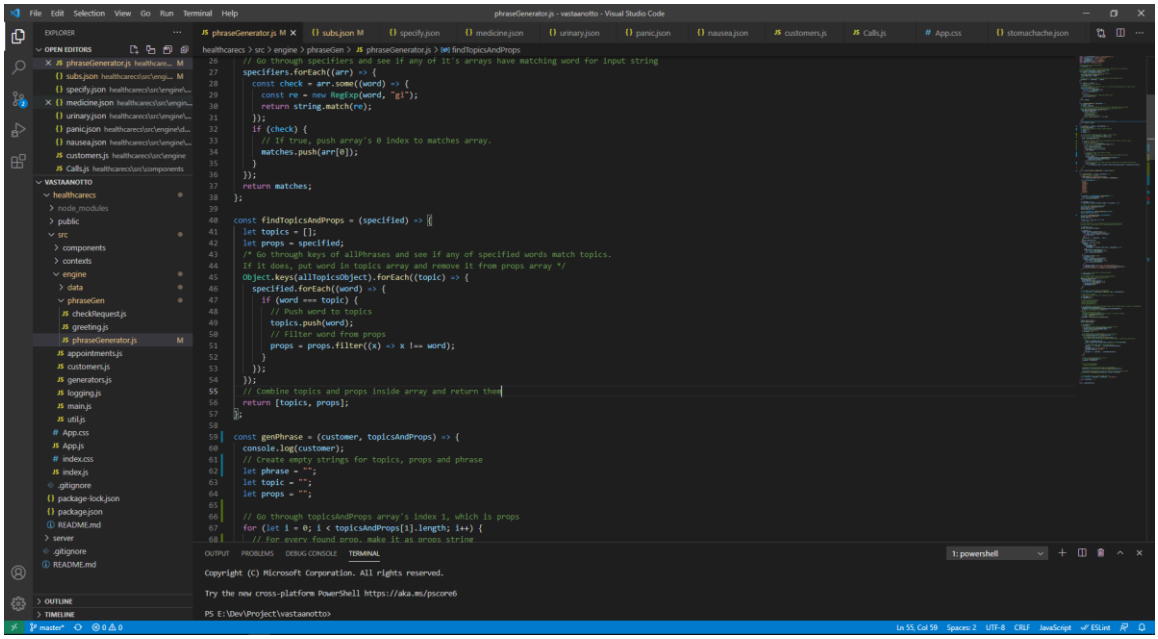
Avainsanojen perusteella toimivan lausegeneroinnin tulisi kuitenkin olla laajennettavissa käyttämään muitakin syöttötapoja, jotta peliin voitaisiin tulevaisuudessa rakentaa esimerkiksi erilaisia vaikeustasoja. Lauseiden avainsanojen suunnittelussa otettiin huomioon hoitoalan erilaisen kielen, johon liittyvistä haasteita Barnard (2007, 498 – 500.) kertoo hoitoalan kielen käsittelyyn liittyvässä artikkelissaan. Menetelmäksi valikoitui lääketieteellisten ja latinankielisten nimien

käyttäminen oireiden ja sairauksien nimissä, sillä ne ovat standardeja eri kielten välillä. Näin rakennettuna peli olisi myös jatkossa helpompi kääntää eri kielille pohjarakenteen pysyessä samana.

Sovelluksen kehittämiseen päädyttiin käyttämään JavaScriptiä ohjelmointikielenä, sillä se oli kehittämiseen osallistuville pääosin tuttu kieli ja soveltuisi selainpelin kehittämiseen hyvin. JavaScript-kirjastosta käytettiin lisäksi Reactia, sillä se soveltuisi hyvin eri näkymien renderöintiin ja antaisi hyvän pohjan front-endille. Lisäksi Reactin avulla saataisiin myös käyttöliittymän puoli paloiteltua hyvin useampaan pienempään kokonaisuuteen, sillä tavoitteena oli pelin modulaarisuus.

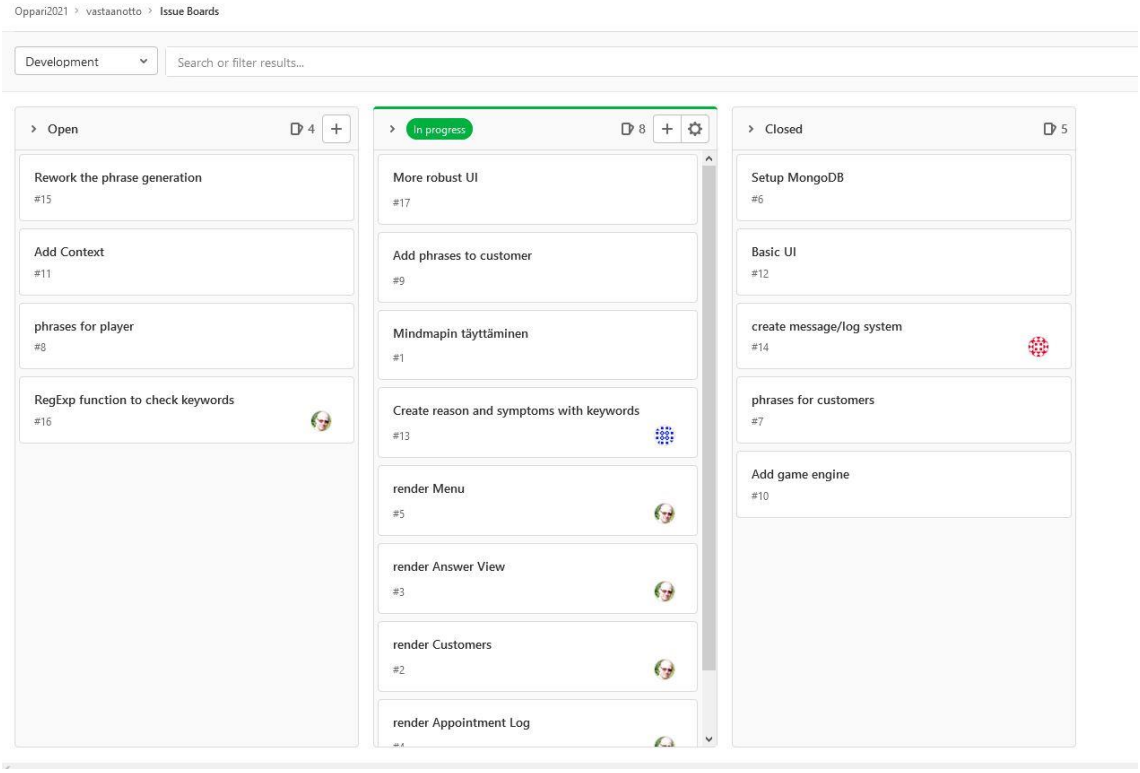
Tavanomaisia haasteita Reactin kanssa tuli komponenttien tahattomien renderöitymisen suhteen, mutta ne ratkesivat nopeasti. Tietokannaksi valikoitui MongoDB, sillä sen ominaisuudet olivat kehittäjille ennestään tuttuja ja se soveltuisi myös hyvin asiakastekoälyn monitasoisen objektipohjaisen tiedon tallentamiseen.

Tekstieditoriksi valittiin VS Code (Kuvio 1), joka oli ennalta tuttu ja toimisi tarkoitukseen hyvin. Muokattujen rivien värikoodaus ja useamman rivin yhtäaikainen muokkaaminen olivat erityisen hyödyllisiä ominaisuuksia projektin aikana.



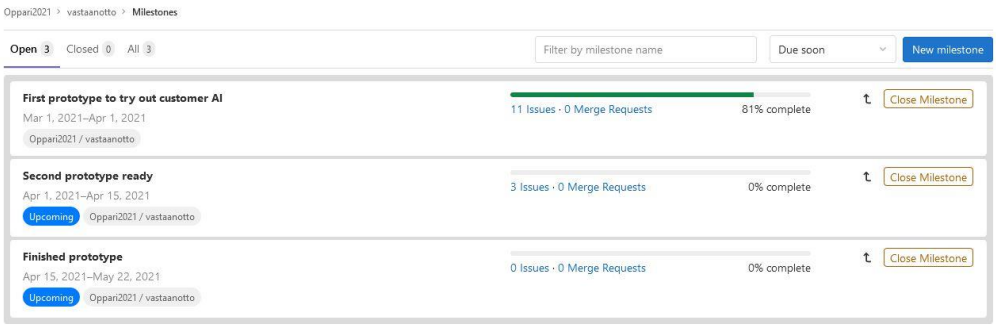
Kuvio 1. Visual Studio Coden perusnäkymä projektin ohjelmoinnin aikana

Projektissa hyödynnettiin GitLabin perinteisen versionhallinnan lisäksi sen projektinhallintatyökalujen boardia ja milestoneja eli tehtävänäkymää (Kuvio 2) ja välietappeja. Suurin osa tehtäviin liittyvistä keskusteluista käytiin Discordin puolella, ja GitLabin projektinhallintatyökaluja hyödynnettiin lähinnä kokonaistilanteen ja aikatauluihin liittyvien tavoitteiden seurantaan, mihin se soveltuikin hyvin. Gitistä version päivittäminen GitLabiin oli ajoittain todella hidasta ja turhauttavaa. Tehtävät kokonaisuudet jaettiin pienempiin osatehtäviin, ja niitä tehtiin yksi kerrallaan valmiiksi. Välillä tehtäviä jouduttiin pilkkomaan vieläkin pienempiin osiin yksittäisten ominaisuuksien perusteella.



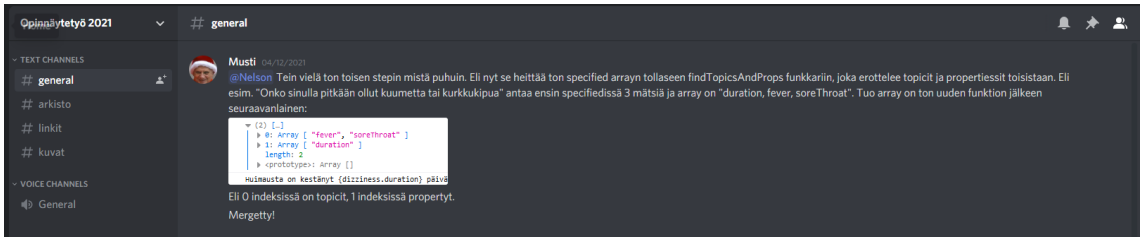
Kuvio 2. Tehtävänäkymä GitLabissa 30.3.2021

Välietapit (Kuvio 3) kuvaavat laajempien kokonaisuuksien valmistumista, ja niitä hyödynnettiin hahmottamaan ensimmäisen testausvaiheen ajankohtaisuutta. Kuviossa 3 näkyy 30.3.2021 päiväyksen tilanne, kun ensimmäinen vaihe alkoi olemaan valmis. Toiseen ja kolmanteen vaiheeseen ei vielä tuolloin laitettu useita tavoitteita, sillä ensimmäinen vaihe oli melko tarkoin rajattu ja tarkoituksena saada ensin palautetta sen hetkisestä tilanteesta, ennen seuraaviin vaiheisiin siirtymistä.



Kuvio 3. Välietapit GitLabissa 30.3.2021

Yleiseen yhteydenpitoon ja suunnitteluun käytettiin Discordia (Kuvio 4). Se toimi pienenä ryhmänä toteutettuun projektiin oikein hyvin. Kanavan rakenne oli jaettu useampaan osioon, joissa projektin kannalta tärkeille kuville ja linkeille oli omat osionsa. Discordin avulla pystyttiin pitämään viikottain pieniä palavereita, jakamaan suorana ruutua tekijöiden välillä ja keskustelemaan viestein. Sen avulla tilanteisiin oli helppo tarttua nopeasti ja yhteistyö oli sujuvaa.



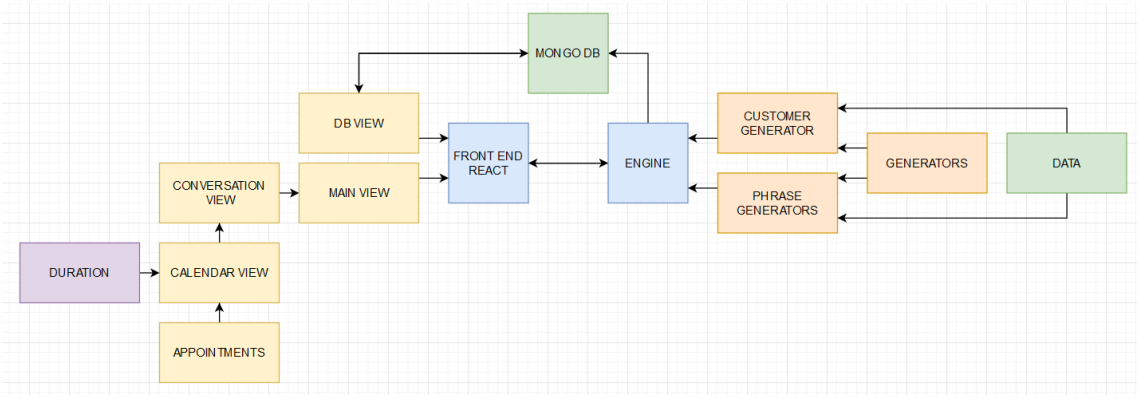
Kuvio 4. Osittainen näkymä 12.4.2021 käydystä Discord-keskustelusta

5.2 Pelin arkkitehtuuri

Arkkitehtuuri koostuu kahdesta kokonaisuudesta, jotka keskustelevat keskenään. Rakenne pyrittiin pitämään mahdollisimman modulaarisena (Kuvio 5). Ensimmäinen niistä on Reactilla rakennettu käyttöliittymä, joka on suoraan yhteydessä pelaajaan. Se tarjoaa pelaajalle visuaalisen yhteyden peliin ja antaa tarvittavat hallintatyökalut pelin pelaamiseen ja asiakkaiden kanssa keskustelemiseen. Front-end on erillään pelimoottorista, ja se käytännössä vain lähettää tiedon käsiteltäväksi pelimoottorille, joka onkin toinen osa kokonaisuutta. Ja-

vaScriptillä rakennettu pelimoottori pitää huolen asiakkaiden ja lauseiden muodostamisesta. Se käsittelee käyttöliittymästä saadun tiedon lausegeneraattorissa, ja palauttaa sen takaisin pelaajan nähtäväksi.

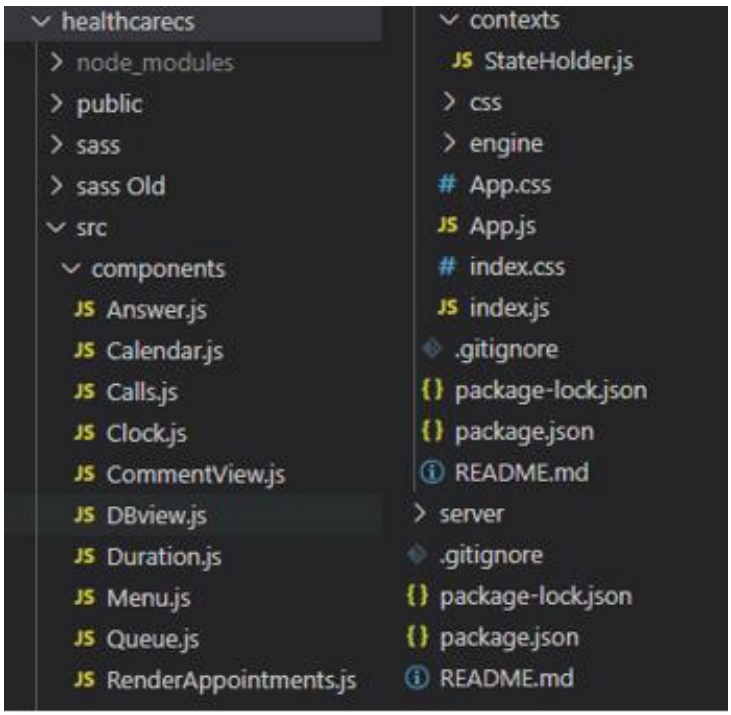
Taustalla on MongoDB:llä rakennettu tietokanta, johon opinnäytetyön aikana kerättiin testidata. Data piti sisällään asiakkaan tiedot asiakkaan ja pelaajan välillä tapahtuneen keskusteluketjun. Kuviossa 5 näkyy pelin eri komponenttien modulaarinen rakenne. Nuolet osoittavat pääasiallista tiedon kulkusuuntaa eri komponenttien välillä.



Kuvio 5. Pelin modulaarinen rakenne

Käyttöliittymän eli Reactin kansiorakenne ja tiedostot on jaoteltu pienempiin modulaarisiin kokonaisuuksiin (Kuvio 6). Kaikki eri näkymät ja ominaisuudet on jaettu eri komponentteihin, jotta ne toimivat itsenäisesti ja ovat koodipuolella selkeämmin ymmärrettävissä. Esimerkiksi Duration.js pitää sisällään vain aikalaskurin puhelun kestoa varten, ja se tuodaan alikomponenttina Answer.js-näkymän sisään.

Eri komponenttien välillä tietoa kulkee pääosin hierarkisesti pääkomponenteista alikomponentteihin, mutta rakensin komponenttien välisen datan liikutteluun myös kontekstin (Context), jonka avulla erilaisten, kuten yleisien pelitilanteeseen liittyen arvojen säilömiseen oli oma paikkansa. Kontekstitiedoston koodi kykenee keskustelemaan minkä tahansa komponentin kanssa, joten sen sisältämiä arvoja pystyttiin hyödyntämään tarvittaessa missä komponentissa tahansa sen sijaan, että niitä olisi juoksutettu vain pääkomponenteista alikomponentteihin.



Kuvio 6. Kansiorakenne ja tiedostojen jaottelu eri komponentteihin

Pelimoottorin tiedostot on jaettu eri hakemistoihin ja kokonaisuuksiin (Kuvio 7). Sen lähtöpiste on main.js, jonka initGame-funktiota kutsutaan käyttöliittymässä pelin alkaessa. Tuolloin initGame kutsuu customers.js-tiedoston sisällä olevaa generaattoria luomaan halutun määrän asiakkaita. Lisäksi main.js vastaanottaa pelaajan syöttämän tekstin, ja välittää sen lausegeneraattoriin.

Asiakkaiden ja lauseiden muodostamiselle sekä lauseiden tarkastamiselle on erilliset tiedostonsa. Lisäksi on yleisiä generaattoreita ja satunnaislukujen tai satunnaisten alkioiden valintaan kykeneviä funktioita omissa tiedostoissaan. Data-kansio pitää sisällään tiedostot mitä generaattorit hyödyntävät rakentaessaan asiakkaita ja lauseita.

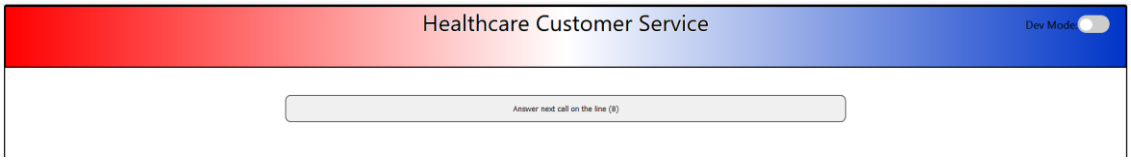
Kuviossa 7 näkyy, että topics-kansiossa on pitkä lista eri keskusteluaiheiden tiedostoja ja ne pitävät sisällään lauseobjekteja. Eri aiheet kootaan yhteen allPhrases.js-tiedostossa, joka kokoaa aiheet yhteen taulukkoon ja sen generaattoreille. Aiheita ja lauseobjekteja lisäämällä asiakas kykenee laajentumaan käsittelemään eri teemoja ja tarpeita.

Kuvio 7. Pelimoottorin kansio- ja tiedostorakenne

5.3 Käyttöliittymä

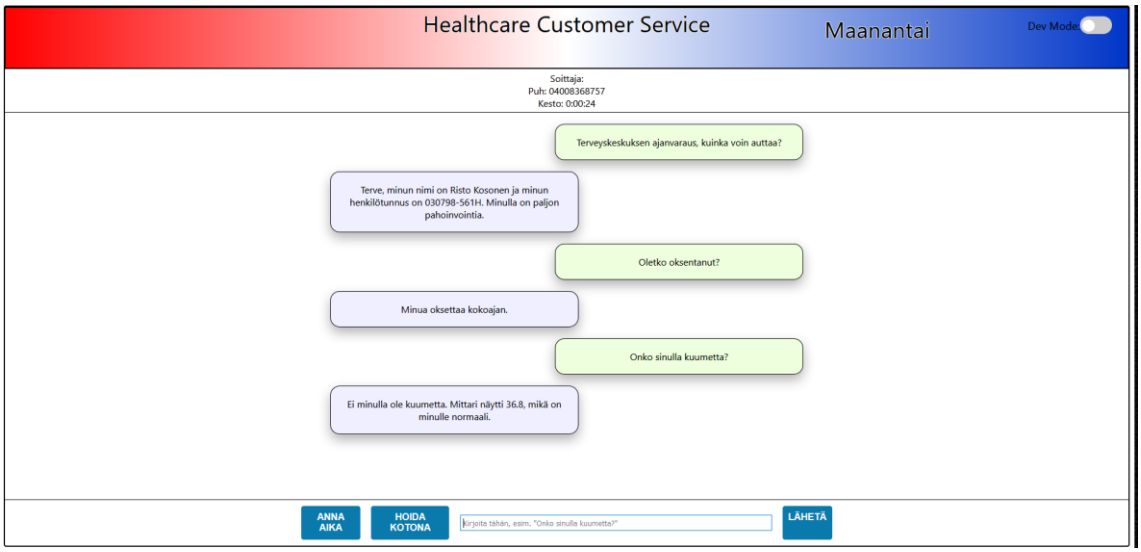
Käyttöliittymä on jaoteltu eri näkymiin, joiden välillä siirtymiseen käytetään React Router-kirjastoa. Yläpalkin muodostava komponentti on mukana jokaisessa näkymässä. Ajanvarauskalenterin tuodaan vastausnäkymän päälle erillisenä elementtinä. Alkunäkymä on pelkistetty näkymä, jossa pelaaja voi ottaa vastaan seuraavan puhelun. Vastausnäppäimessä oleva numero näyttää puhelujonossa olevien asiakkaiden määrän (Kuvio 8). Sitä painamalla valitaan asiakaslistalta seuraava henkilö, ja näkymä vaihtuu keskustelunäkymään, jonne asiakkaan tiedot viedään mukana.

Testausvaiheessa alkunäkymään oli liitetty myös tiedote ja palautelaatikko. Lisäksi käyttöliittymän tyyliä muutettiin ennen viimeisintä testausvaihetta responsiivisemmaksi, jotta se toimisi paremmin myös puhelimilla, ja helpottaisi testaamista.



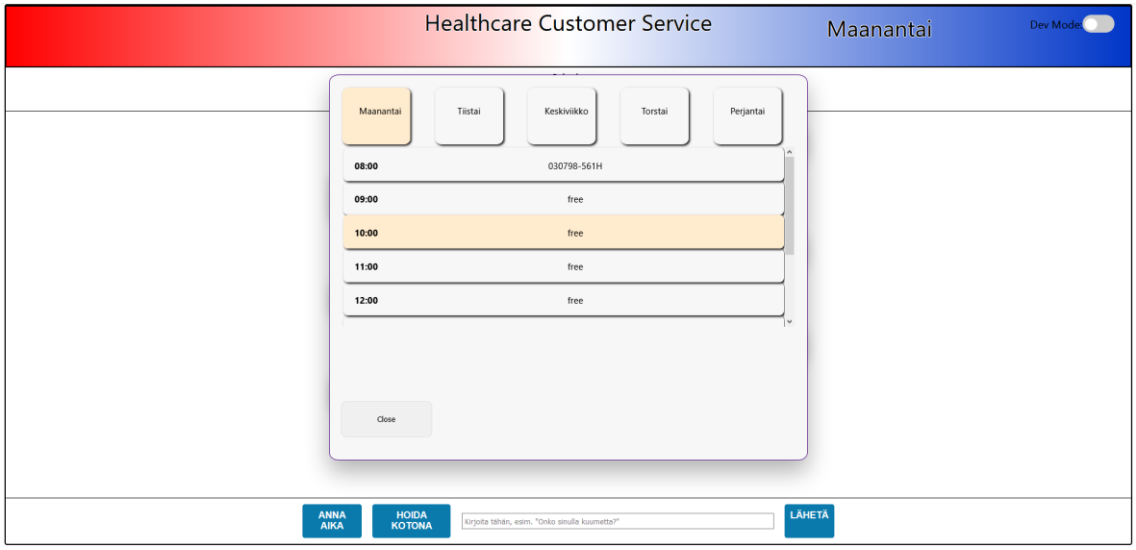
Kuvio 8. Alkunäkymä ja vastausnäppäin

Keskustelunäkymässä (Kuvio 9) pelaaja pystyy vapaasti kirjoittamalla käymään keskustelua asiakkaan kanssa. Keskustelun avaus tapahtuu automaattisesti, minkä jälkeen asiakas aloittaa esittelemällä itsensä, ja kertoo omien henkilötietojensa lisäksi jotain omista vaivoistaan. Keskustelun aikana pelaaja pystyy kysymään asiakkaalta lisäkysymyksiä, joiden avulla hoidon tarve ja kiireellisyys selviävät.



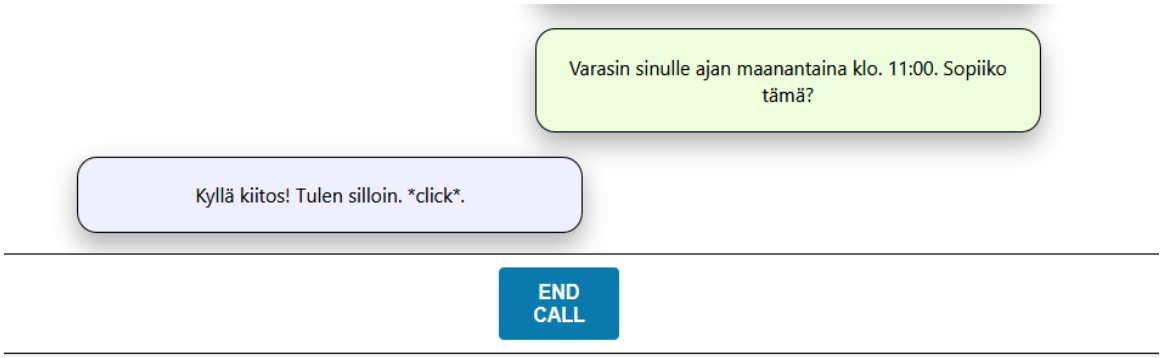
Kuvio 9. Keskustelunäkymä puhelun aikana

Kalenterinäkymä (Kuvio 10) on rakenteeltaan yksinkertainen, yhden viikon kerrallaan näyttävä näkymä. Sen avulla pelaaja pystyy varaamaan haluamansa viikonpäivän ja kelloajan asiakkaalle. Aikoja voi antaa yhden kerrallaan, ja toisen ajan päälle ei voi varata uutta aikaa. Kuviossa 10 näkyy, miten maanantaipäivä on valittuna ja pelaajan hiiri on kello 10 aikavarauksen päällä. Kello 8 aika on jo varattu toiselle asiakkaalle.



Kuvio 10. Kalenterinäkymä ajan varaamista varten

Vapaata aikaa klikatessa kalenteri sulkeutuu automaattisesti, ja ohjelma muodostaa annetusta ajasta lauseen, jonka se toimittaa keskustelunäkymään asiakkaalle. Asiakas hyödyntää vastauksessaan ajanvaraamiseen liittyvää aihetta, jonka avainsanojen perusteella se kykenee vastaamaan tarjottuun aikaan (Kuvio 11). Asiakkaan lauseen lopussa on teksti **click**, joka kuvastaa asiakkaan puhelimen sulkemista. Lauseiden sisältöä tarkkaillaan jatkuvasti taustalla toimivan lauseentarkistuksen avulla. Asiakkaan sulkiessa puhelimen hallintapaneeli muuttuu, ja pelaaja ei voi enää tehdä muuta kuin sulkea puhelun.



Kuvio 11. Asiakkaan päättämä keskustelu

5.3 Asiakastekoäly

Asiakkaana pelillistetyssä terveyskeskuksen puhelinpäivystyksessä on tekoälyn hallinnoimia asiakkaita. Pelimoottori luo yksilöllisiä asiakkaita (Kuvio 12), joilla jokaisella on omat syynsä ja oireensa olla yhteydessä puhelinpäivystykseen. Syyt ja oireet muodostetaan pelimoottorissa erilaisten rajoitettujen satunnaisgenerointien kautta. Tekoäly osaa hyödyntää itselle arvottuja ominaisuuksia keskustellessaan pelaajan kanssa ja muodostaessaan lauseita. Asiakas muodostetaan siten, että luokan avulla sille määritellään ominaisuudet eli muuttujat, ja sen toiminnallisuudet eli metodit. Asiakkaita luodaan pelin aikana useita kymmeniä, ja muuttujien arvot tekevät lopulta jokaisesta asiakkaasta yksilöllisen omine tietoineen ja yhteydenottosyineen. Pelaaja ei tiedä tekoälyasiakkaan yhteydenottosyytä etukäteen, vaan sen voi päätellä vain asiakastekoälyn kanssa keskustelemalla.

Kuviossa 12 näkyy luokan muodostaminen, ja sen sisältämät ominaisuudet, kuten nimet, puhelinnumero ja yhteydenottosyy. Niiden alapuolella näkyy myös getPhrase-metodi, jonka avulla asiakas pystyy yhdistelemään pelaajan antaman syötteen pelimoottorissa oleviin avainsanoihin ja lausegeneraattoreihin.

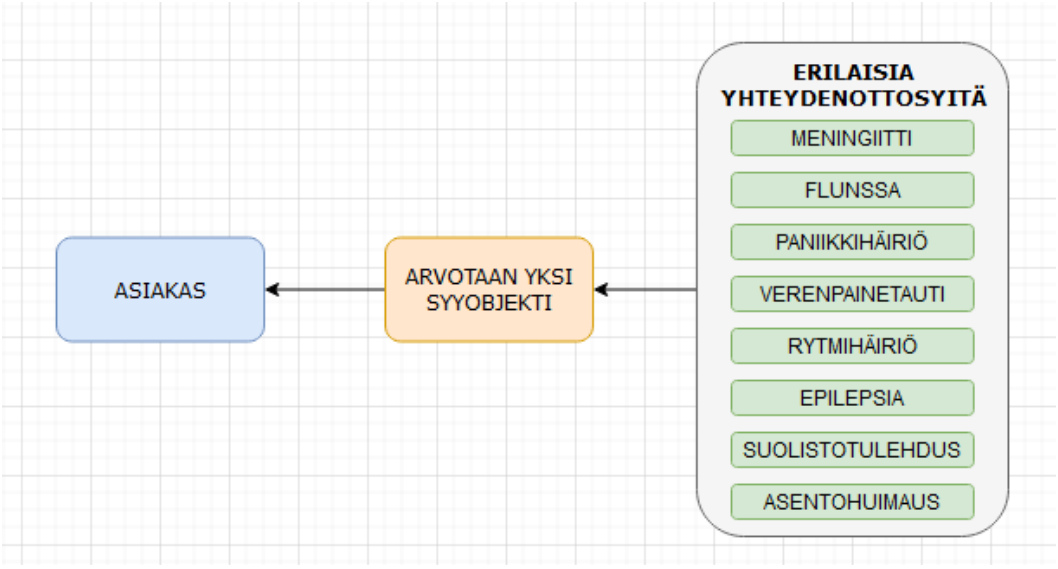
```
// Base class for the customers.
class Customer {
  constructor() {
    this.socialSecurityNumber = genSocialSecurityNumber();
    this.gender = genGender(this.socialSecurityNumber);
    this.firstName = generateName("firstName", this.gender);
    this.lastName = generateName("lastName");
    this.phoneNumber = generatePhoneNumber();
    this.reason = generateReason();
    this.othersArray = null;
  }

  getPhrase = (userInput) => {
    // Call generatePhrase function to return one phrase in renderer. At first round it will return a Greeting
    const phrase = generatePhrase(this, userInput);
    return phrase;
  };
}
```

Kuvio 12. Yksilöllisen asiakkaan muodostaminen

5.3.1 Syiden ja aiheiden muodostaminen

Asiakas saa muodostusvaiheessaan itselleen listalta satunnaisen syyn olla yhteydessä pelaajaan. Syitä voi olla käytännössä millaisia tahansa, mutta pelin teeman vuoksi tämän opinnäytetyön kirjoitushetkellä niitä ovat esimerkiksi flunssa,

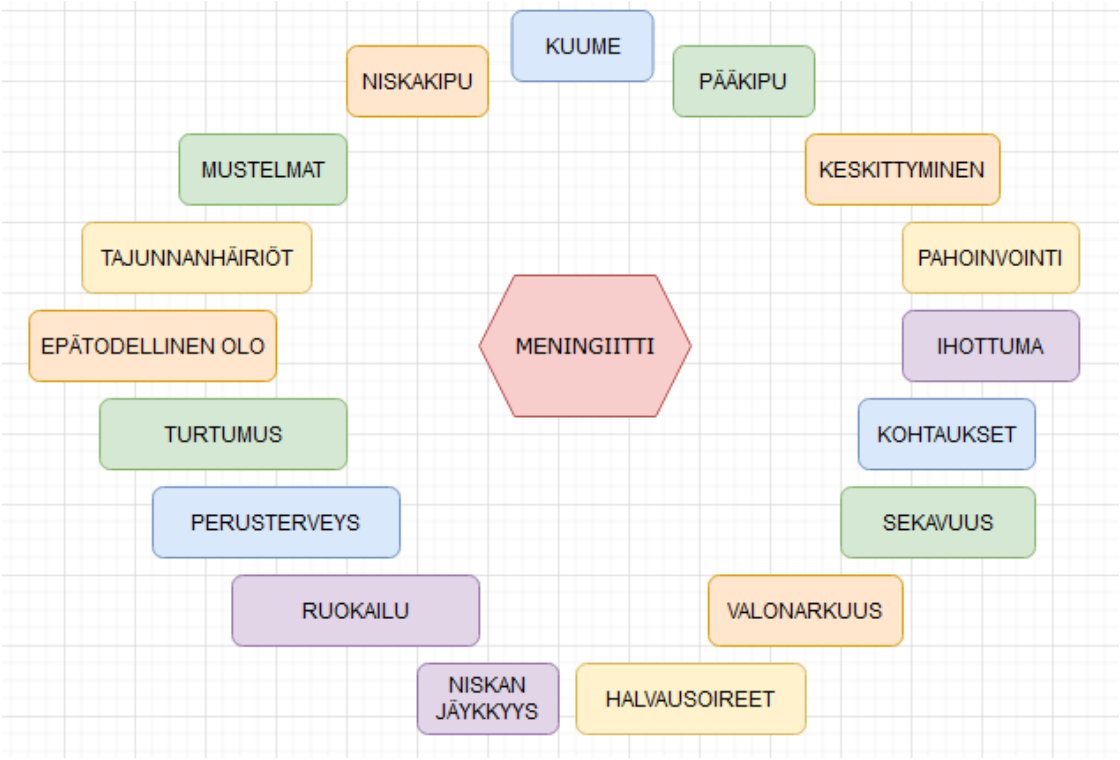
meningiitti, gastroenteriitti ja paniikkihäiriö. Erillinen funktio (Kuvio 13) hakee satunnaisen syyobjektin allReasons-taulukosta.



Kuvio 13. Syyobjektin valitseminen satunnaisesti allReasons-taulukosta

Jokainen syyobjekti (Kuvio 14) pitää sisällään erilaisia aihealueita, jotka määrittelevät asiakkaalle monenlaisia oireita ja taustatietoja. Aiheiden avulla asiakas kykenee myös keskustelemaan yksityiskohtaisemmin yhteydenottosyystään eli tässä tapauksessa sairauden oireista. Asiakas kykenee keskustelemaan myös muistakin aiheista, mutta tällöin vastaukset ovat geneerisempiä *”Ei minulla ole ongelmaa tämän asian kanssa”* -tyylisiä lauseita.

Kuviossa 14 näkyy esimerkiksi yhteydenottosyy meningitti. Se pitää sisällään meningiitin oireille tavanomaisia oireaiheita ja aiheita, mistä pelaaja todennäköisesti kysyy selvitellessään asiakkaan hoidon tarvetta. Koska eri aihealueet ovat liitetty osaksi meningiittiä, algoritmi käy läpi jokaisen aiheen yksitellen luodessaan asiakasta ja antaa niille yksilölliset arvot ja ominaisuudet. Tämä mahdollistaa sen, että jokainen asiakas on erilainen arvoiltaan ja ominaisuuksiltaan, vaikka yhteydenottosyy olisikin yhtenevä.



Kuvio 14. Meningiitti-objekti ja sen sisältämät pääasialliset keskusteluaiheet

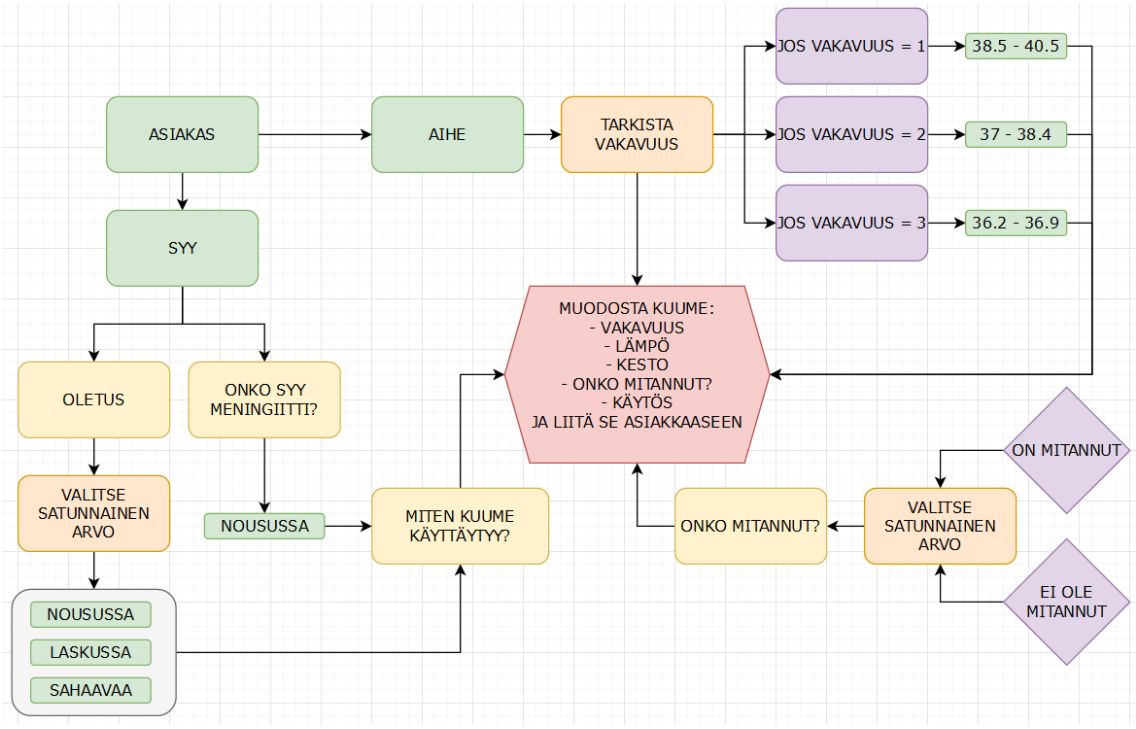
Jokainen aihe pitää sisällään taulukon (Kuvio 15), jonka sisällä on kolme alimuuttujaa, joille on annettu numeroarvo. Ne kuvaavat prosentuaalista mahdollisuutta sille, kuinka vakavasta oireesta on kysymys. Ensimmäinen alimuuttujan arvo kuvastaa mahdollisuutta saada ensimmäisen vakavuusasteen aihe, joka on voimakkuudeltaan vakavin. Toinen alimuuttujan arvo kertoo mahdollisuuden saada toisen vakavuusasteen aihe, joka tarkoittaa, että kyseisen aiheen kanssa on jonkin verran ongelmia. Kolmas alimuuttujan arvo tarkoittaa sitä, että aiheen kanssa ei ole ongelmia. Täten esimerkiksi kuviossa 15 näkyvällä meningiittipotilaalla [80,20,0] tarkoittaa sitä, että asiakkaalla on 80 prosentin todennäköisyydellä korkea kuume ja 20 prosentin todennäköisyydellä ainakin jonkin verran kuumetta.

```
16    "name": "meningitis",
17    "topics": [
18      ["fever", [80,20,0]],
19      ["neckPain", [75,25,0]],
20      ["neckStiffness", [75,25,0]],
21      ["headache", [75,25,0]],
22      ["lightSensitivity", [25,50,25]],
```

Kuvio 15. Yksi yhteydenottosyy ja osa sen painotuksista

Kuumeeton meningiitti on tarkoituksella rajattu pois, sillä kuten infektiosairauksien erikoislääkäri Lumio Jukka (2019) toteaa Duodecimin lääkärikirjassa, kuume on meningiitin klassinen oire. Painotuksien avulla voidaan siis jäljitellä todenmukaisia sairauksia, ja saada pelaajallekin todentuntuinen kokemus.

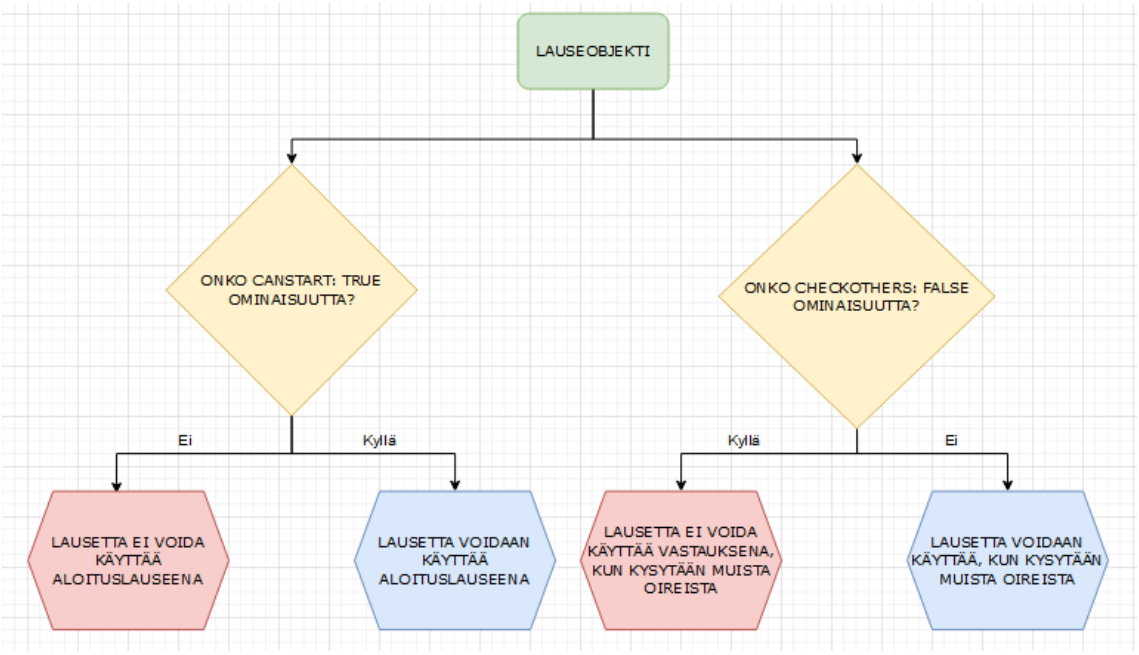
Osaan aiheista on lisätty lisäominaisuuksia ja arvoja, jotka luodaan asiakkaalle aiheen vakavuuden perusteella ja sitä varten rakennetulla erillisellä generaattorilla (Kuvio 16). Ne on lisätty osaksi asiakasta, jotta sen vastaukset ovat järkeviä ja yhdenmukaisia useastikin samasta asiasta kysyttäessä. Tällaisia aiheita ovat esimerkiksi kuume, jonka mittausarvo halutaan tietää tarkasti tai altistuminen, jotta tiedetään tarkasti asiakkaan ulkomaanmatkan sijainti. Kuviossa 16 näkyy esimerkkinä kuumeen muodostaminen, jossa kuume saa vakavuusarvon, lämpötilan, keston, tiedon onko asiakas mitannut kuumettaan, ja sen onko kuume noussa vai laskussa.



Kuvio 16. Kuumeen muodostaminen vakavuuden ja syyn perusteella

5.3.2 Lauseiden muodostaminen

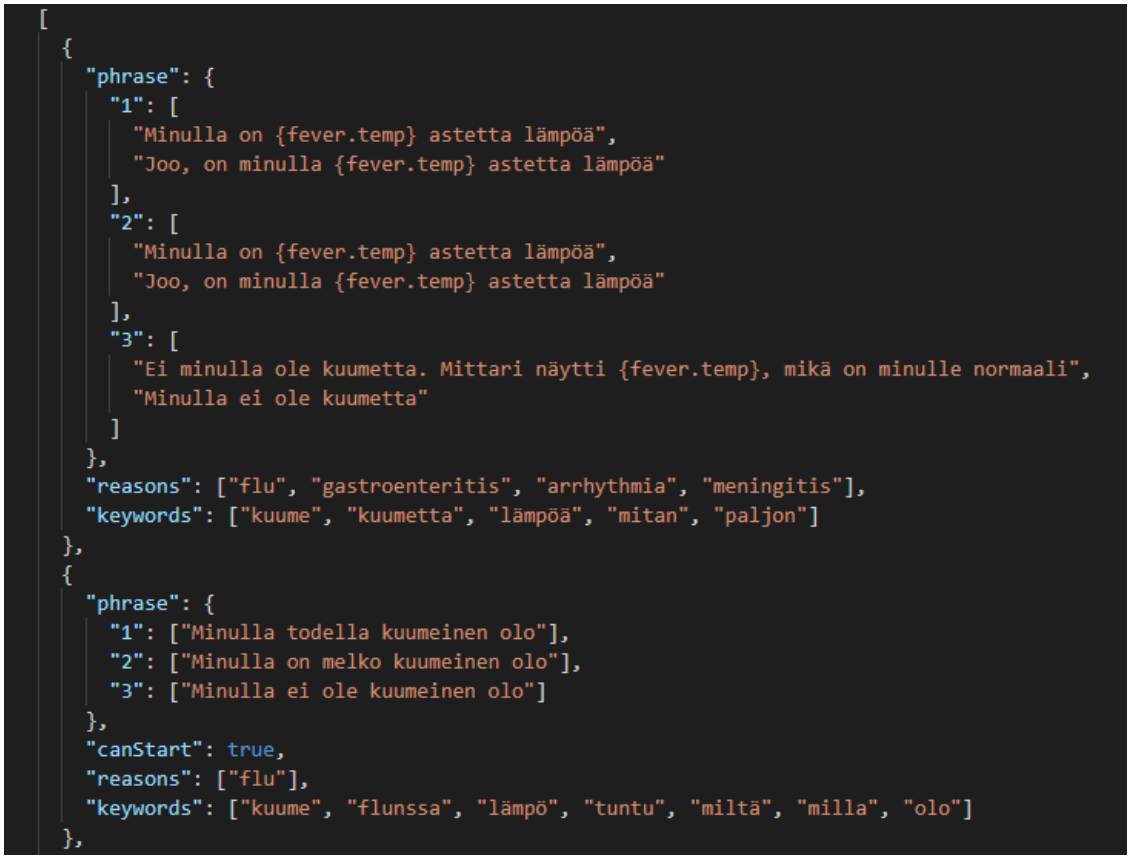
Jokainen lauseobjekti pitää sisällään erilaisia ominaisuuksia, kuten kolmen eri vakavuusasteen lausetaulukot, mahdolliset syyt minkä yhteydessä lauseita voidaan käyttää, avainsanat, joiden perusteella lauseobjekti voi tulla valituksi. Lauseobjektit saattavat pitää myös sisällään erilaisia ominaisuuksia (Kuvio 17), kuten canStart: true-ominaisuuden, jonka perusteella asiakas voi käyttää näin merkittyjä lauseita keskustelun avaamisessa. Lisäksi osa lauseista on merkitty checkOthers: false-ominaisuudella, jos ne ovat epäsopivia käytettäväksi hoitajan kysyessä muista oireista. Algoritmin eri tarkistusvaiheissa näitä arvoja käytetään määrittelemään tai rajaamaan lauseobjektin käyttötarkoitusta.



Kuvio 17. Lauseobjektien käyttötarkoitusta määrittelevät ominaisuudet

Otetaan esimerkiksi kuume-aiheen sisällä olevat ensimmäiset kaksi lauseobjektia (Kuvio 18). Ne pitävät sisällään kolme eri vakavuusasteista lausetaulukkoa. Niiden sisällä saattaa olla useampia vaihtoehtoja samasta lauseesta, jotta tekoäly tuntuisi luonnollisemmalta, ja vastaus ei olisi jokaisella kerralla samanlainen. Lisäksi lauseobjektit pitävät sisällään yhteydenottosyyt, joiden yhteydessä kyseistä lauseobjektia yleensä käytetään. Kuviossa 18 näkyy, miten ensimmäisessä kuumeen määrään liittyvässä lauseobjektissa on useampia syitä minkä yhteydessä asiakas kykenee lausetta käyttämään.

Toisessa lauseobjektissa käyttötarkoitusta on haluttu rajoittaa flunssa-aiheeseen. Näin saadaan tarvittaessa rakennettua oirekohtaisia yksilöityjä vastauksia tiettyihin tilanteisiin. Toinen lauseobjekti pitää sisällään myös edellä mainitun käyttötarkoitusta määrittelevän canStart: true-ominaisuuden, eli sen lauseita voidaan käyttää keskustelun avaamisessa. Kumpikaan ei pidä sisällään checkOthers: false-ominaisuutta, joten ne voivat tulla valituiksi muista oireista kysyttäessä. Lisäksi molemmat lauseobjektit pitävät sisällään myös avainsanat, joihin lauseobjektit todennäköisimmin reagoivat.



```
[
  {
    "phrase": {
      "1": [
        "Minulla on {fever.temp} astetta lämpöä",
        "Joo, on minulla {fever.temp} astetta lämpöä"
      ],
      "2": [
        "Minulla on {fever.temp} astetta lämpöä",
        "Joo, on minulla {fever.temp} astetta lämpöä"
      ],
      "3": [
        "Ei minulla ole kuumetta. Mittari näytti {fever.temp}, mikä on minulle normaali",
        "Minulla ei ole kuumetta"
      ]
    },
    "reasons": ["flu", "gastroenteritis", "arrhythmia", "meningitis"],
    "keywords": ["kuume", "kuumetta", "lämpöä", "mitan", "paljon"]
  },
  {
    "phrase": {
      "1": ["Minulla todella kuumeinen olo"],
      "2": ["Minulla on melko kuumeinen olo"],
      "3": ["Minulla ei ole kuumeinen olo"]
    },
    "canStart": true,
    "reasons": ["flu"],
    "keywords": ["kuume", "flunssa", "lämpö", "tuntu", "miltä", "milla", "olo"]
  },
```

Kuvio 18. Kuume-aiheen kaksi lauseobjektia

Asiakkaan lauseiden muodostaminen on monivaiheinen prosessi. Sen ensimmäinen vaihe on ottaa pelaajan vapaasti kirjoittama teksti käsittelyyn, jossa hyödynnetään lauseobjektien tietoja.

5.3.3 Aiheen selvittäminen ja avainsanojen laskeminen

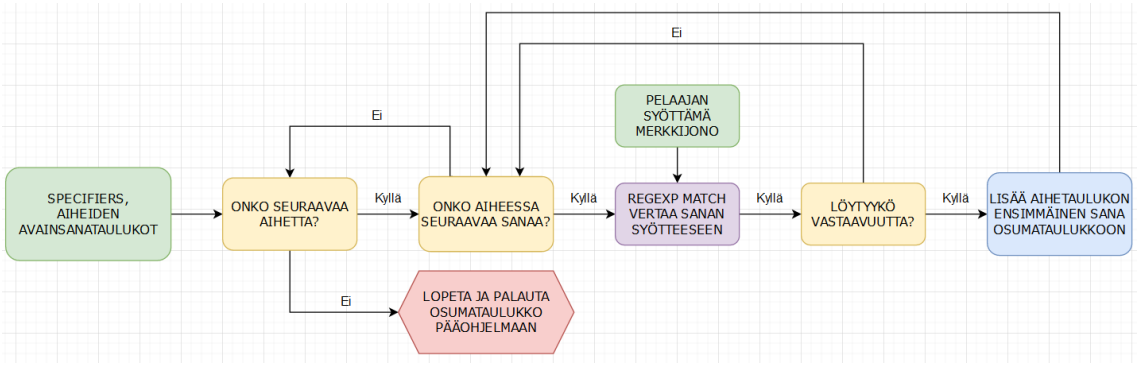
Lauseenvalintamenetelmiä on tämän opinnäytetyön kirjoitushetkellä kaksi erilaista, joista ensimmäisen rakensin täydentämään vanhempaa perinteisempää menetelmää. Uusi menetelmä on helpommin skaalautuva, ja se muodostaa pienemmällä koodimäärällä enemmän monipuolisia lauseita. Uuden menetelmän vahvuus on erityisesti useamman aiheen yhtäaikaisessa käsittelyssä, ja siinä, ettei valmiita lausepohjia tarvitse kirjoittaa valmiiksi kovin monta, vaan se hyödyntää yhtä pohjaa kattamaan jokaisen aiheen samaan ominaisuuteen liittyviä asioita. Esimerkiksi kestolauseessa ei ole väliä, onko aiheena kuume, pääkipu tai jonkin muun aiheen kesto. Toistaiseksi se käsitteleekin lähinnä aiheen kestoon liittyviä lauseita, ja se ajetaan koodissa ennen muita tarkistuskohtia.

Jos uusi menetelmä ei kykene löytämään lausevastinetta, niin algoritmi käyttää perinteisempää menetelmää. Uudempi menetelmä toimii siten, että ensin muodostetaan taustalle erillinen specify-taulukko (Kuvio 19), joka pitää sisällään eri aiheiden ja ominaisuuksien alitaulukoita, ja niiden avainsanoja. Ensimmäisessä 0-indeksissä on aiheen nimi, ja loput ovat siihen liittyviä avainsanoja.

Kuvio 19. Specify-taulukot eri aiheille. 0-indeksissä varsinainen aiheen nimi

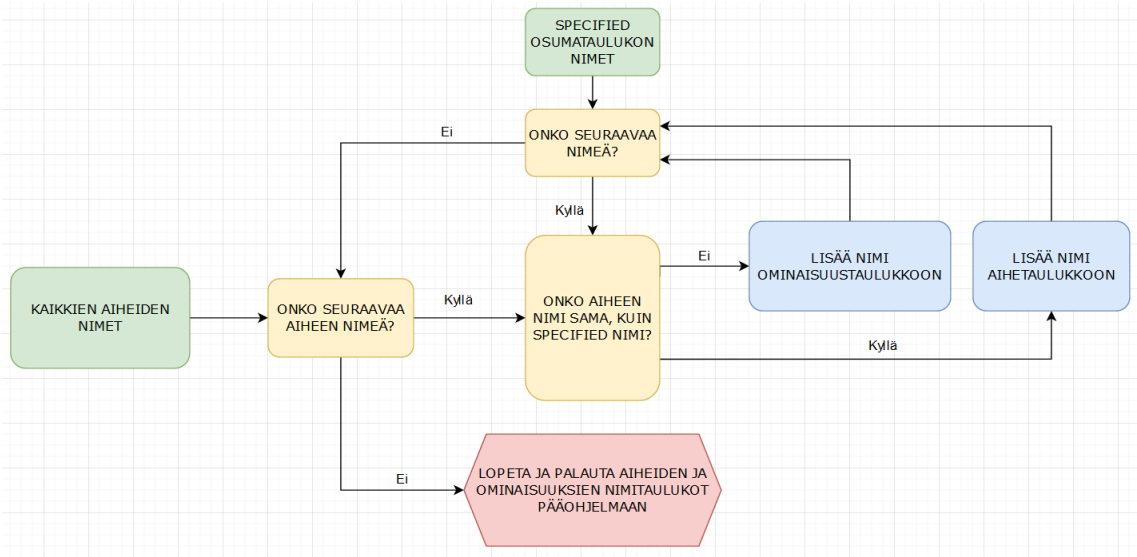
Tämän jälkeen otetaan käyttöön ensimmäinen tarkistusvaihe, joka on sitä varten rakennettu erillinen specify-funktio (Kuvio 20). Sen tarkoituksena on verrata specify-taulukoiden avainsanoja pelaajan antamaan syötteeseen, ja yrittää selvittää, mistä asiasta pelaajan kertomassa lauseessa on kyse. Kaikki löytyneet vastaavuudet aiheuttavat sen, että specify-taulukon ensimmäinen 0-indeksi otetaan talteen, ja esimerkiksi *”Onko sinulla ollut pitkään kuumetta?”* -kysymys löytäisi vas-

tineita kestosta ja kuumeesta, jotka asetetaan eri aiheita sisältävään osumataulukkoon. Vastaavasti taas *”Onko sinulla ollut pitkään kuumetta tai pääkipua?”* löytäisi vastineet keston ja kuumeen lisäksi myös pääkivusta. Näin saadaan siis luotua taulukko, jossa on listattu kaikki tunnistetut aiheet ja ominaisuudet.



Kuvio 20. Specify-funktio ja aihealueiden etsiminen pelaajan syötteestä

Seuraavaksi specify-funktion palauttama osumataulukko viedään erilliseen findTopicsAndProps-funktioon (Kuvio 21), joka erottelee osumataulukon aiheet ja ominaisuudet erillisiin taulukoihin. Se vertaa taulukon arvoja kaikkien aihealueiden nimiin, ja osuman löytäessään se siirtää arvon aihetaulukkoon. Loput osumataulukkoon jääneet arvot ovat ominaisuuksia. Aihe- ja ominaisuustaulukot palautetaan pääohjelmaan yhteisen päätaulukon sisällä.



Kuvio 21. Aiheiden ja ominaisuuksien erottelu eri taulukoihin

Seuraava vaihe hyödyntää subs.json-tiedoston aihetaulukkoa, joka pitää sisällään aiheobjekteja (Kuvio 22). Objektin pääarvona on aiheen nimi ja ominaisuuksina lauseessa käytettävä lokalisoitu vastine sekä kestossa käytettävä oikea sanamuoto. Esimerkiksi kuumeen kestossa puhutaan päivistä, mutta esimerkiksi paniikkioireiden kuvastamisessa voidaan haluta käyttää päivien sijaan kuukausia. Pienellä jatkokehityksellä kaikki ajat laskettaisiin aina vuorokausina ja lauseen muoto vaihdettaisiin vuorokausimäärän mukaan, mutta tässä versiossa tämän uudenkin menetelmän täytyy olla yhteensopiva vanhemman järjestelmän kanssa, joka arpoo kestoja pienemmillä arvoilla.

```
{
  "fever": {
    "name": "kuume",
    "duration": "päivää"
  },
  "headache": {
    "name": "pääkipu",
    "duration": "päivää"
  },
  "neckPain": {
    "name": "niskakipu",
    "duration": "päivää"
  },
  "cough": {
    "name": "yskä",
    "duration": "päivää"
  },
  "urinary": {
    "name": "virtsavaivat",
    "duration": "päivää"
  },
  "vomiting": {
    "name": "oksentelu",
    "duration": "päivää"
  },
```

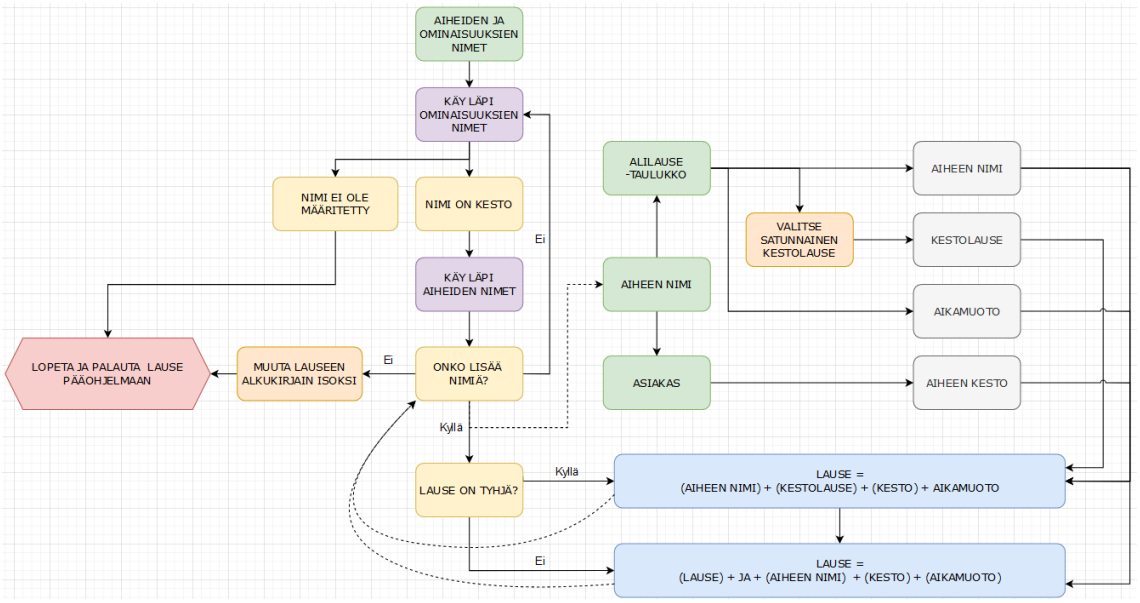
Kuvio 22. Subs tiedoston sisällä olevia aiheobjekteja, joilla lauseita rakennetaan

Tämän jälkeen toisistaan erotellut aiheet ja ominaisuudet viedään erilliseen lauseen rakennusfunktioon (Kuvio 23), joka muodostaa niiden perusteella lauseen. Perinteisestä menetelmästä eroten tämä funktio kykenee rakentamaan moniosaisen lauseen annettujen tietojen perusteella.

Ensin muodostetaan lauseelle, aiheelle ja ominaisuudelle omat tyhjät merkkijonot. Sen jälkeen käydään läpi ominaisuustaulukko, ja aina ominaisuuden löytyessä, lisätään ominaisuuden nimi merkkijonoon. Tämä merkkijono tarkistetaan,

ja jos kyse on kestosta, tarkistetaan vielä aiheen nimi, ja näiden tietojen perusteella haetaan erillisestä subs-taulukkolistasta aiheen alta oikea lokalisoitu vastine aiheen nimelle, ominaisuudelle ja muille halutuille asioille kuten vaikkapa erilaiselle minälauseelle. Algoritmi asettaa tiedot peräkkäin, jolloin saadaan muodostettua valmis lause.

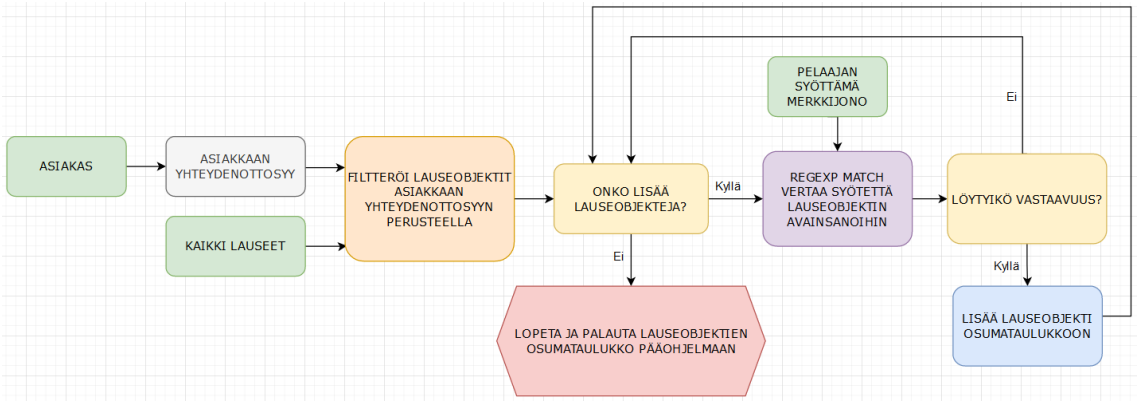
Kuviossa 23 näkyy esimerkiksi oikeassa alakulmassa kohta, jossa lause rakentuu subs-taulukoista löytyvistä aiheen nimestä, satunnaisesta kestosanasta, asiakkaan tiedoista haetusta ominaisuuden arvosta ja subs-taulukon aiheen alla olevasta kestosanasta. Tämä tarkoittaa sitä, että jos algoritmi asettelee sanat *”kuume”, ”on kestänyt”, ”5”* ja *”päivää”* peräkkäin, niistä saadaan muodostettua kokonainen järkevältä kuulostava lause. Tämän jälkeen algoritmi jatkaa aiheiden läpikäymistä, ja tarvittaessa jatkaa lausetta toisenkin aiheen kestolauseella, lisäten väliin *”ja”* sanan. Tällöin lause olisi kokonaisuudessaan muotoa: *”kuume on kestänyt 5 päivää ja pääkipu 4 päivää”.* Ennen pääohjelmaan palautusta lauseeseen varmistetaan vielä iso alkukirjain.



Kuvio 23. Funktio, joka yhdistelee subs-taulukon ja asiakkaan tietoja

Jos specify-menetelmä ei löytänyt lauseelle vastinetta, niin algoritmi ottaa käyttöön vanhemman menetelmän, joka hyödyntää erilaisia taulukkometodeja etsimään lauseet, jotka täsmäävät asiakkaan yhteydenottosyyhyn.

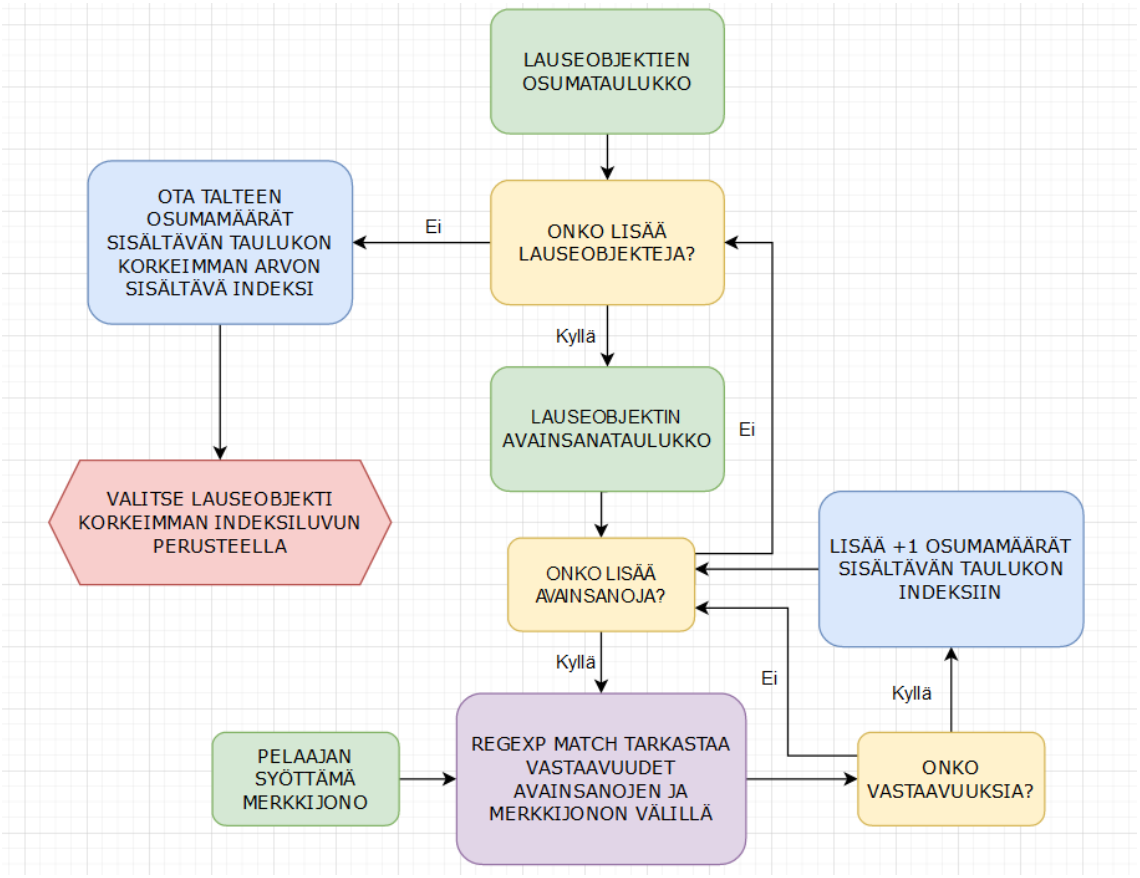
Lauseen tarkistamisessa (Kuvio 24) käytetään myös RegExpin (Regular Expression) match-menetelmää, joka vertaa lausetta kaikkien eri aiheiden avainsanoihin. RegExpin sääntöinä on ”gi”, joista ”g” tarkoittaa globaalia hakua, joka käy koko syötetyn merkkijonon läpi, eikä pysähdy ensimmäiseen osumaan. Jälkimmäinen ”i” taas tarkoittaa sitä, että haku on case-insensitive eli RegExp ei kiinnitä huomiota kirjainkokoon. Avainsanojen perusteella saadaan luotua taulukko osumia varten, jotka pitävät sisällään vain oikean aihealueen lauseobjekteja.



Kuvio 24. Oikea-aiheisten lauseobjektien etsiminen

Osumataulukon lauseobjektien avainsanat verrataan vielä uudelleen annettuun syötteeseen ja osumien määrät lasketaan (Kuvio 25). Niistä tehdään numerotaulukko, jonka jokainen indeksi vastaa osumataulukkoa ja pitää sisällään numeroarvon, joka kertoo avainsanojen osumamäärän. Näin saadaan tietää, miten hyvin eri lauseobjektien avainsanat vastasivat pelaajan antamaan lauseeseen. Numerotaulukon arvoista poimitaan korkein arvo, jonka perusteella saadaan tietoon korkeimman tarkkuuden vastaavan lauseobjektin indeksiluku.

Tasatilanteessa lauseobjektista valitaan yksi lause arpomalla. Arpomiseen käytetään erillistä funktiota, joka poimii satunnaisen arvon taulukon sisältä. Kuviossa 25 näkyy vaihe, jossa avainsanoja verrataan RegExpin avulla pelaajan antamaan syötteeseen ja jokaista osumaa kohden indeksin osumamäärä kuvastavaa lukua kasvatetaan yhdellä. Sen jälkeen otetaan talteen korkein indeksiarvo.

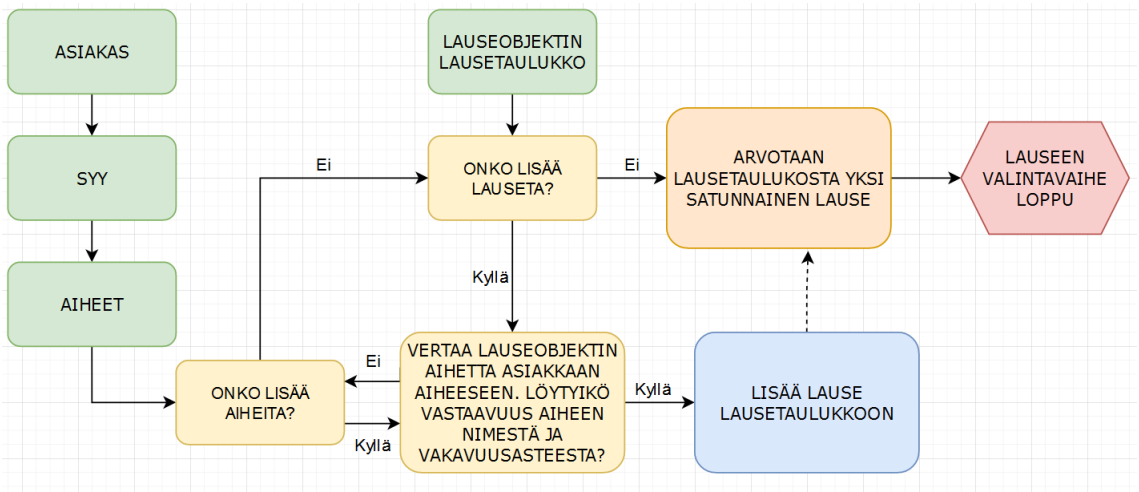


Kuvio 25. Avainsanojen osumamäärien laskeminen

5.3.4 Lauseen valinta lauseobjektin sisältä.

Perinteisessä menetelmässä lauseobjektin sisällä olevien lausetaulukoiden vakavuusasteita verrataan asiakkaan aiheiden vakavuusarvoihin (Kuvio 26), ja saadaan tietoon oikea-asteiset lausetaulukot. Lisäksi varmistetaan syyn ja aiheen sopivuus, sillä jos pelaaja on esimerkiksi kysynyt ”*Onko muita oireita?*” algoritmi tunnistaa avainsanoista, että kyse on tarkista muut-aiheesta, ja tällöin oikeanlainen tarkistuslause voidaan suoraan asettaa valituksi.

Sopivista lauseista muodostetaan lausetaulukko, jonka sisältä arvotaan yksi lause. Kuviossa 26 näkyvät objektin arvojen läpikäyminen ja edellä mainitut eri tarkistusvaiheet.

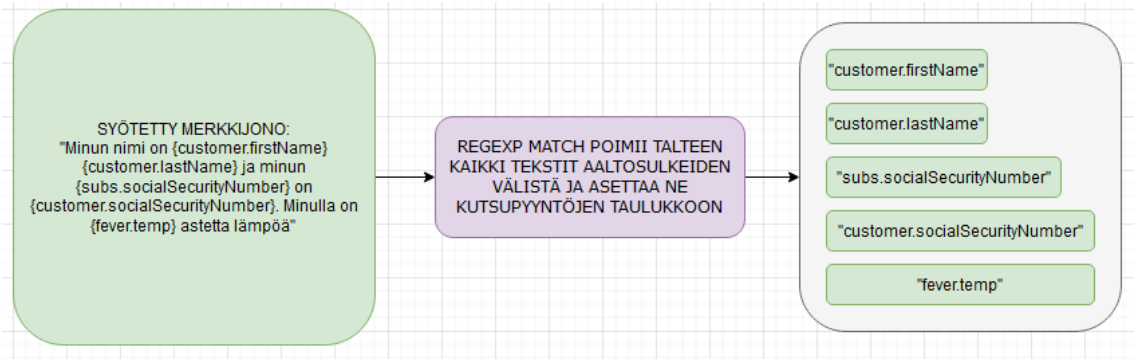


Kuvio 26. Lausetaulukon valinta lauseobjektin sisältä ja lauseen arvonta

5.3.5 Valitun lauseen tarkistaminen ja täydentäminen

Valittu lause tarkistetaan vielä ennen pelaajalle lähetystä erillisessä checkRequests-funktiossa (Kuvio 27), sillä mahdolliset asiakkaan eri arvojen tai täytelauseiden kutsupyynnöt on asetettu aaltosulkeiden sisään ja checkRequests-funktio kykenee erottelemaan ja käsittelemään jokaisen pyynnön erikseen. Erilaisten asiakkaiden arvojen tai täytesanojen kutsupyyntöjä voi yhdessä lauseessa olla useampia, ja ne kaikki asetetaan pyyntötaulukon sisään.

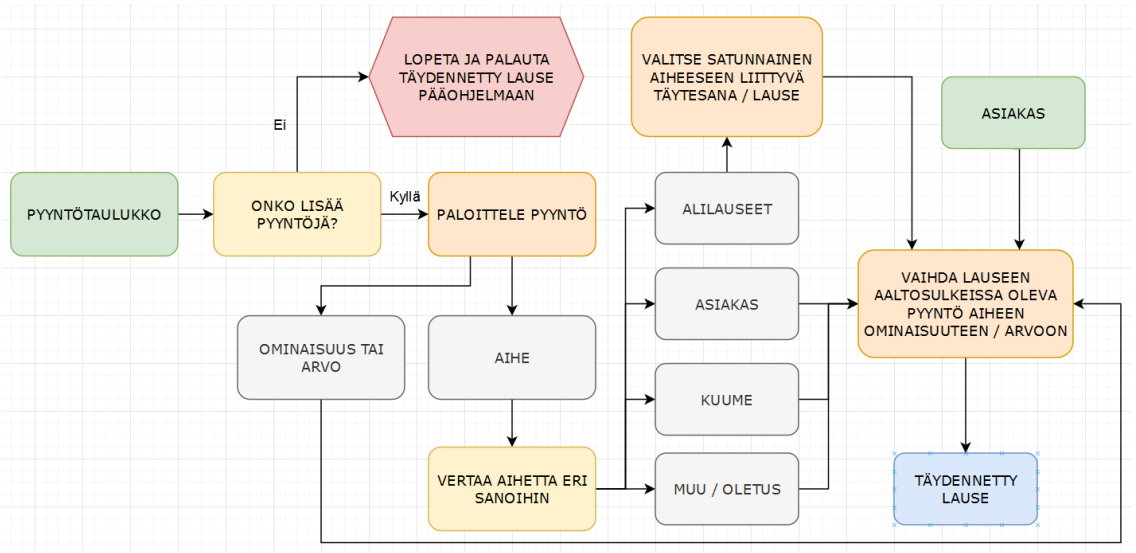
Pyynnöt ovat lauseessa esimerkiksi kuumeen arvoa pyytäessä muodossa {fever.temp} ja pyyntötaulukkoon siirrettäessä aaltosulkeet poistetaan käyttäen RegExpin match-menetelmää, jättäen jäljelle vain fever.temp-merkkijonon. Kuviossa 27 näkyy funktio, joka muodostaa parametrinä sisään tulleen lauseen aaltosulkeiden välistä löytyneiden sanojen perusteella kutsupyyntötaulukon.



Kuvio 27. Kutsupyyntöjen etsintä lauseesta

Sen jälkeen kutsupyynnöt paloitellaan pisteen perusteella kahteen osaan. Ensimmäinen osa kertoo, minkä aiheen ominaisuudesta on kyse. Toinen osa kertoo sen, mitä arvoa tarkalleen haetaan. Esimerkiksi edellä mainittu fever.temp kertoo, että asiakkaan yhteydenottosyyn alla olevan fever-objektin sisältä haetaan lämpötila-arvoa. Ensimmäinen aihesana tarkistetaan ja riippuen aiheesta se voidaan käsitellä eri tavoin. Oletuksena algoritmi pääsee tietojen perusteella suoraan käsiksi pyydettyyn arvoon, mutta joissain tapauksissa halutaan tehdä poikkeuksia.

Algoritmiin on rakennettu erillinen koodiosuus, jossa loopin avulla (Kuvio 28) saadaan vain asiakkaan perustietoja esiin, kuten customer.socialSecurityNumber tai alilausetaulukosta poimittua erilaisia täytesanoja ja synonyymejä, kuten erilaisia tervehdyksiä subs.greetings-taulukosta. Haetut tiedot tai sanat liitetään alkuperäisen kutsupyynnön paikalle, ja tällä tavalla saadaan muodostettua kokonainen vaihteleva lause asiakkaan omien yksilöllisten tietojen perusteella. Tämä lause palautetaan asiakkaan sanomana pelaajan nähtäväksi.

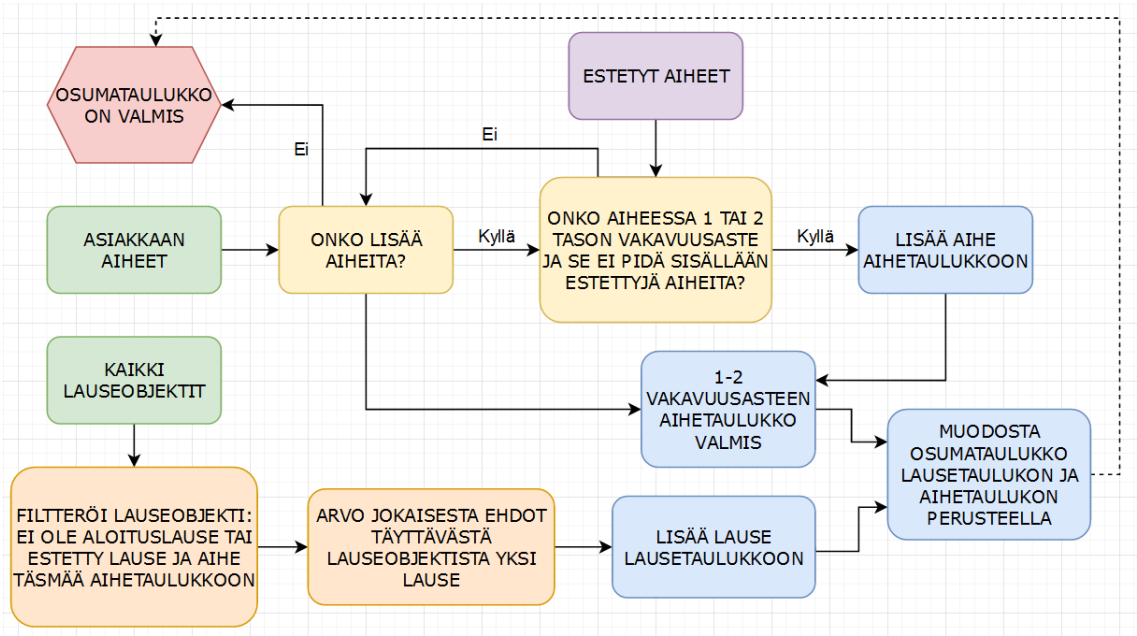


Kuvio 28. Kutsupyyntöjen paloittelu yksitellen ja lauseen täydentäminen

Hoitoalan teeman huomioiden pelaaja todennäköisesti kysyy jossain vaiheessa keskustelua asiakkaan muista oireista. Tällöin avainsanojen perusteella valitaan {checkOthers}-pyyntö, jonka käsittelyssä checkRequests-funktio (Kuvio 29) kutsuu checkOthers-funktiota tarkistamaan asiakkaan muut aiheet niiden vakavuuden perustella. Tässä vaiheessa suoritetaan samankaltainen prosessi, kuin nor-

maalissakin lausevalinnassa, ja saadaan osumataulukko, joka pitää sisällään asiakkaalle merkitseviä aiheita. Sellaisia ovat esimerkiksi aiheet, joissa asiakkaalla on ensimmäisen tai toisen tason vakavuusaste. Yleisesti ottaen tämä tarkoittaa sitä, että kyse on asiakkaan sairauden erilaisista liitännäisoireista, jotka asiakas kokee vakavina.

Tarkistusvaiheessa estetään tiettyjen epäsopivien aiheiden valituksi tuleminen erillisellä estolistalla. Tällaisia aiheita ovat esimerkiksi päivittäinen liikunta, suolan käyttö ruoassa tai altistuminen.

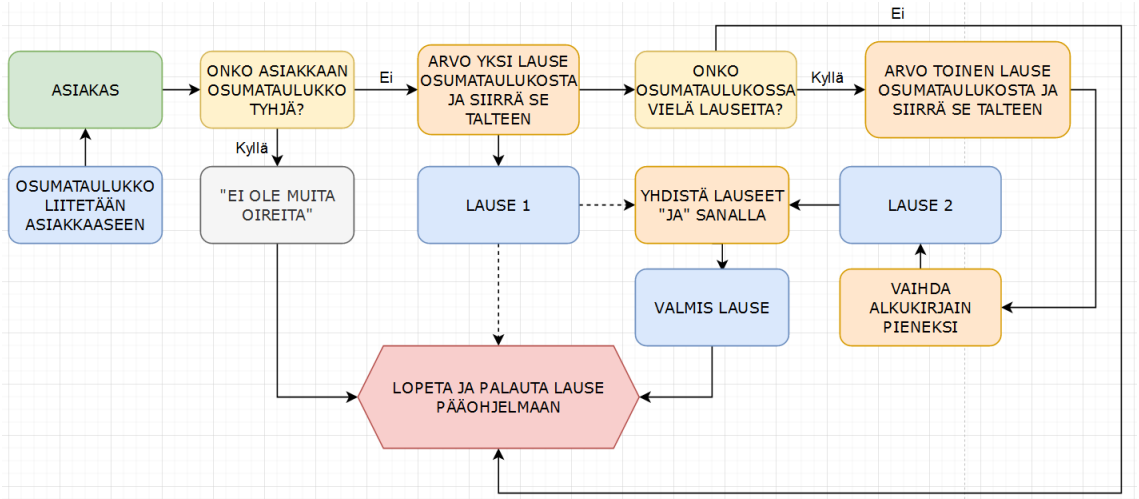


Kuvio 29. Asiakkaalle merkittävien aiheiden tarkistaminen

Osumataulukko liitetään osaksi asiakasta (Kuvio 30), koska käsitellyt aiheet poistetaan taulukosta niistä keskustelemisen jälkeen. Tällöin asiakas ei vahingossakaan sano esimerkiksi lausetta *”Minulla on kurkku kipeä”* kahta kertaa peräkkäin, vaan kertoo jostain muusta oireesta, josta ei vielä ole ollut puhetta.

Löydetyistä lauseista rakennetaan kaksiosainen lause, jos asiakkaalla on jäljellä useampi kuin yksi oire. Lauseiden väliin asetetaan ja-sana, jonka jälkeen jälkimmäisen lauseen ensimmäinen kirjain muutetaan pienellä kirjaimella alkavaksi. Tällöin asiakas kykenee sanomaan esimerkiksi lauseen *”On kovia paniikkituntemuksia, enkä osaa rauhoittua ja on ollut jonkin verran vaikeuksia*

hengittää”. Jos taas jäljellä olevia oireita ei ole kuin yksi, niin asiakas palauttaa yksittäisen lauseen pelaajan nähtäväksi ilman ja-sanaa.



Kuvio 30. Yhden tai kahden lauseen liittäminen asiakkaaseen

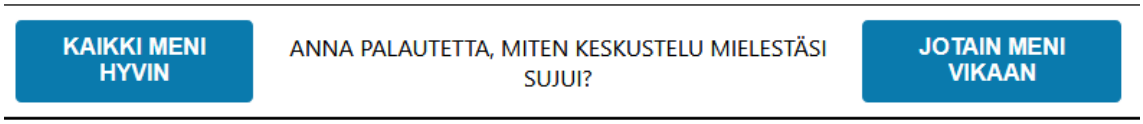
6 PELIN TESTAAMINEN JA PALAUTE

6.1 Pelin testaus

Pelin testaaminen jaettiin eri vaiheisiin. Ensimmäisessä vaiheessa pelin tekoälyn toimivuutta testattiin tekijöiden lisäksi noin viiden eri testikäyttäjän avulla, jotka olivat pelin kehittäjien lähipiireistä. Mukana ensimmäisessä vaiheessa oli myös yksi hoitoalalla työskentelevä testaaja. Ensimmäisen vaiheen loppupuolella tekoälyn kanssa käytyjä keskusteluja oli noin 300. Keskustelujen läpikäynnin ja korjauksien jälkeen pelistä tehtiin myös responsiivisempi, jotta testaaminen olisi helpompaa puhelimella ja tableteilla.

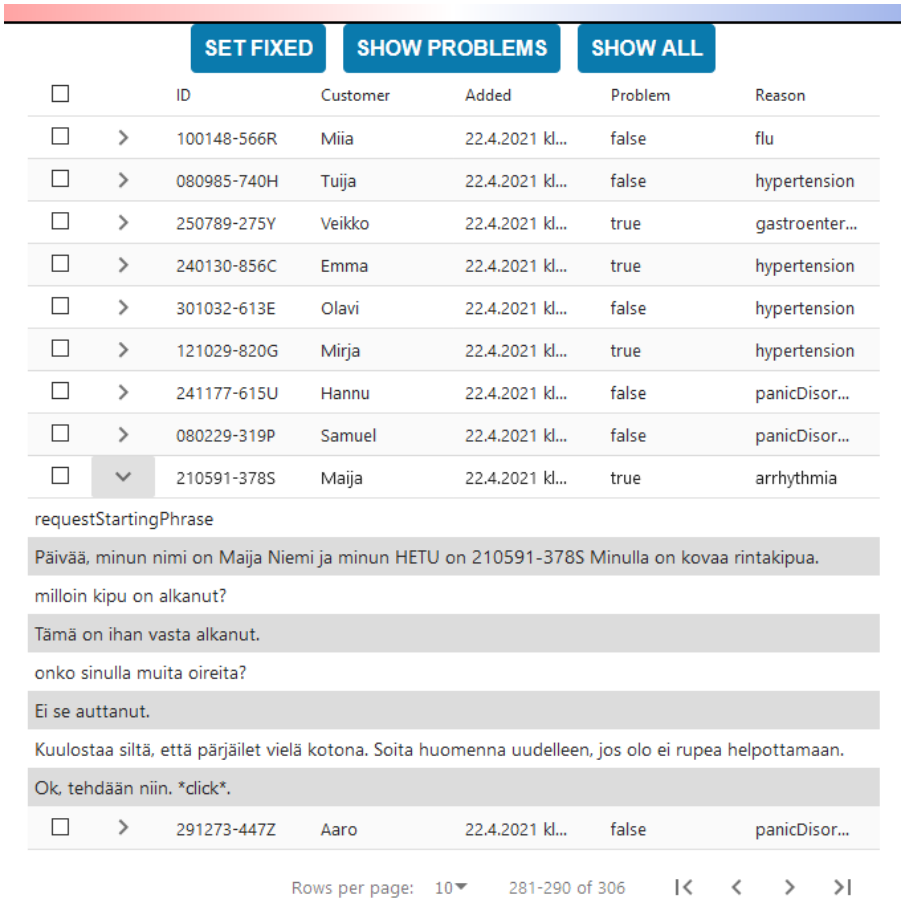
Sen jälkeen testiversion linkki jaettiin toimeksiantajallemme Outi Tierannalle, joka välitti peliä eteenpäin hoitoalan opiskelijoille, opettajille ja omalle verkostolleen. Toisen testikierroksen oltua käynnissä noin viikon verran, tekoälyn kanssa käytyjä keskusteluja oli kertynyt yhteensä 505. Tämä oli odotettua vähemmän, mutta testimateriaalia kertyi silti riittävä määrä epäkohtien löytämiseen, ja tekoälyn kehitystarpeiden tunnistamiseen.

Pelin testaamiseen osallistuminen oli vapaaehtoista, ja tietoa kerättiin käyttäjiltä anonyymisti. Heti sovelluksen avautuessa tuli ilmoitusteksti missä kerrottiin testaajille testikierroksen odotuksista ja tavoitteista. Ilmoitustekstissä kerrottiin myös, mitä tietoja pelin kokeilemisesta tallennetaan, ja että kerätty tieto hävitetään viimeistään projektin päättyessä. Testaajille annettiin myös mahdollisuus jättää vapaasti palautetta etusivulla olleeseen palautelootaan. Pelin testidatan kerääminen mahdollistettiin siten, että jokaisen puhelun päätyttyä pelaajalta pyydettiin palautetta tekoälyn kanssa käydystä keskustelusta (Kuvio 31). Käyttäjän painettua palautenäppäintä, keskustelun ja asiakastekoälyn tiedot siirtyivät MongoDB-tietokantaan myöhempää tarkastelua varten.



Kuvio 31. Palautenäppäimet puhelun päättyessä

Palautteiden tarkastelu tapahtui erikseen sitä varten rakennetussa ympäristössä, jonne MongoDB-tietokannasta ladattiin jokainen testaamisen aikana käyty keskustelu (Kuvio 32). Käyttäjän antaman palautteen pohjalta tietoihin oli liitetty ongelman kohdalle true- tai false-arvo, jonka perusteella keskusteluista oli helpompi etsiä korjausta vaativat kohteet. Lisäksi voitiin painaa show problems-näppäintä, joka näytti kaikki true-arvon sisältävät kohteet. Kuviossa 32 näkyy avoinna oleva keskustelu, jonka pelaaja oli merkinnyt ongelmalliseksi.



Kuvio 32. Pelaajan ongelmalliseksi merkitsemä keskustelu testiympäristössä

Korjauksien jälkeen voidaan ruksata käsitellyt aiheet, ja painaa set fixed-näppäintä, joka vaihtaa problem-arvon falseksi tietokantaan. Tällä tavalla jo käsiteltyjä viestiketjuja pystyttiin seuraamaan paremmin.

6.2 Testauksen palaute

Testidatan ja ongelmallisiksi koettujen viestiketjujen perusteella eniten korjauksia piti odotetusti tehdä lauseiden avainsanoihin liittyen ja välillä pelaaja oli saattanut

kysyä asiakkaalta aiheesta, johon ei vielä ollut olemassa aihealuetta. Avainsanojen ja aiheiden lisääminen olemassa olevaan runkoon oli kuitenkin nopeaa, joten tekoälyn sanavarasto laajeni hyvää vauhtia testaamisenkin aikana. Lisäsin peliin esimerkiksi aiheita ja lauseita, jotka osasivat vastata esimerkiksi näkökykyyn, vatsakivun sijaintiin, sekä asiakastekoälyn omaan arvioon hoidon tarpeestaan. Palautteissa tuli esiin myös ongelma, jossa ohjelma saattoi kaatua käyttäjän syöttäessä lauseisiin erikoismerkkejä. Lisäsin syötteen käsittelyn alkuvaiheeseen replace-ominaisuuden, joka poistaa syötetystä merkkijonosta kaikki erikoismerkit.

Alustavasti joissain aihealueissa jouduttiin käyttämään väliaikaisia vastauslauseita, sillä yksityiskohtaisempien generaattorien rakentaminen vie enemmän aikaa, ja vaatii suunnittelua. Tästä johtuen esimerkiksi kaikki potilaat vastaavat migreenistä kysyttäessä toistaiseksi *”Ei minulla ole migreeniä”*. Kun peliin halutaan lisätä migreenin vuoksi yhteyttä ottava asiakas tai tehdä migreenistä satunnaisesti ilmaantuva ongelma potilaille, se voidaan kuitenkin helposti lisätä peliin. Testidatan perusteella päädyin lisäämään ominaisuuden, jossa pelaajan kysyessä muista oireista tekoäly tuottaa yhden oireen sijaan kaksi oiretta sisältävän lauseen. Näin pelaaja saa hieman enemmän vaihtoehtoja ja johdattelua seuraava kysymystä varten.

Kirjallisesti ja suullisesti saatiin melko vähän palautetta, ja sitä olisi voinut olla enemmän. Saatu palaute oli kuitenkin pääosin positiivista. Seuraavassa esimerkki hoitoalalla olevan testaajan antamasta palautteesta, kun tätä ohjattiin lähettämään kaikki tekoälyasiakkaat kotona hoidettaviksi testaamisen nopeuttamiseksi: *”Hmm kävin ny tsekkaamassa tuon ja mun mielestä ne oli ihan järkeviä. Teki vähän pahaa, kun oireet oli monilla sellasia että oikeesti en ois vaan käskeny hoidella kotona xD”.* Toinen testikäyttäjä antoi seuraavanlaista palautetta: *”ainaki oma kokemus on että keskustelut oli loogisempia pääpiirteittäin”.*

7 POHDINTA

Tämän työn tarkoituksena oli luoda toimiva prototyyppi terveyskeskuksen puhelinpäivystykseen liittyvästä selainpohjaisesta pelistä osana hoitoalan digitalisaatioon liittyvää *Osaamisen mahdollistaminen digitaalisissa hyvinvointipalveluissa* (OmaDigi)-hanketta. Työn tavoitteena oli rakentaa jatkokehityksellä laajennettavissa oleva tekoäly, joka toimii pelissä asiakkaan roolissa, ja kykenee keskustelemaan pelaajan kanssa.

Projektin haasteena oli rajallinen aikataulu sekä pyrkimys pitää pelin arkkitehtuuri modulaarisena ja tekoäly mahdollisimman laajennuskelpoisena muitakin käyttökohteita ajatellen. Aikatauluja ja lausegenerointiin perehtymistä ajatellen asiakkaan eri persoonallisuuksien pois rajaaminen oli tähän kehitysvaiheeseen oikea ratkaisu ajan säästämiseksi ja lausegeneroinnin perusrakenteen toimivuuden varmistamiseksi.

Käytetyt työkalut soveltuivat projektin tekemiseen hyvin. Peli ohjelmoitiin JavaScriptiä ja Reactia hyödyntäen, mikä toimi hyvin selainpohjaisen pelin kehittämiseen. Versionhallintatyökaluina käytettiin Gitiä ja GitLabia, joista GitLab osoittautui ajoittain todella hitaaksi ja tietojen siirto repositoryyn ei välillä toiminut lainkaan. Projektinhallintaominaisuuksiltaan GitLab osoittautui kuitenkin hyväksi työkaluksi. Yhteyden pitämiseen käytettiin Discordia, jonka ominaisuudet toimivat pienelle kehittäjämäärälle todella hyvin.

Terveydenhuoltoalaan liittyvän tekoälyn rakentamisessa työvaiheiden ja tuotteen eettisyys on tärkeää. Aidontuntuisen tilanteen luomiseksi tekoälyasiakkaat käyttivät oikeita satunnaisgeneroituja suomalaisia etu- ja sukunimiä sekä henkilötunnuksia, joten oli tärkeää pitää pelin kokonaisuus mahdollisimman asiallisena.

Lopputuloksena saatiin tehtyä toimiva prototyyppi pelistä, jossa pelaaja toimii terveyskeskuksen puhelinpäivystyksen työntekijänä, ja vastaa tekoälyasiakkaiden puheluihin. Tekoäly kykenee melko hyvin tunnistamaan keskustelun aiheen ja vastaamaan siihen asianmukaisesti joko valmiiksi rakennetuin tai satunnaisesti

generoiduin lausein. Arkkitehtuuriltaan peli saatiin rakennettua modulaariseksi ja niin käyttöliittymän, kuin pelimoottorin osalta.

Uuden ja vanhan lauseenvalintajärjestelmän yhdistäminen ja optimaalisen ratkaisun etsiminen voisi olla jatkossa järkevää. Lausegenerointi ja avainsanojen laatiminen nykyisellään vaatii melko paljon aikaa ja tuntemusta käsiteltävästä aiheesta, mutta se on kohtuu helposti hallittavissa ja rajattavissa eri tarpeisiin. Pelin houkuttelevuutta voisi myös lisätä siten, että asiakkaille lisättäisiin erilaiset luonteenpiirteet, jotka vaikuttavat lauseiden rakentumiseen. Tekoälylle arvotun luonteen avulla peliin olisi mahdollista saada mukaan enemmän huumoria ja arvaamattomiakin tilanteita. Pelillisiä elementtejä peliin olisi muutenkin hyvä lisätä, kuten pelaajan tekemän hoidon tarpeen onnistumisen pisteyttäminen ja satunnaiset tapahtumat, jotka vaikuttavat pelin kulkuun.

Jatkokehittämisen kohteena voisi olla myös koneoppimiseen liittyvien menetelmien, kuten TensorFlow.js- tai Brain.js-kirjastojen hyödyntäminen osana tekoälyn algoritmia. Terveydenhuoltoalaa ajatellen peliin olisi myös mahdollista liittää tarkat hoitokriteerit ja eri sairauksiin liittyvät realistiset todennäköisyydet nykyisten suuntaa antavien arvojen tilalle. Tällöin se palvelisi myös terveydenhoitoalan työntekijöiden ja opiskelijoiden tarpeita entistä paremmin. Koen, että asiakastekoäly on kokonaisuudessaan asiallinen ja käyttökelpoinen terveyskeskuksen puhelinpäivystyksessä tapahtuvan asiakaskohtaamisen pienimuotoiseen harjoitteluun.

Koin opinnäytetyön erittäin mielenkiintoisena projektina ja oman hoitoalan työhistorian huomioiden aihe oli myös helposti lähestyttävissä. Koen, että hoitoalan koulutuksessa ja työelämässäkin voitaisiin tulevaisuudessa hyödyntää tekoälyä entistä enemmän. Sen avulla voitaisiin tarjota mahdollisuus harjoitella eri työtehtävissä tarvittavia taitoja turvallisesti ja itselle sopivana ajankohtana. Tekoälyn tehokas hyödyntäminen voisi tarjota myös oppilaitoksille mahdollisuuden järjestää eri aiheisiin liittyvää opetusta entistä enemmän etänä.

LÄHDELUETTELO

Afzali, H., Torres-Arias, S., Curtmola, R. & Cappos, J. 2018. le-git-imate: Towards Verifiable Web-based Git Repositories. Viitattu 17.4.2021 https://dl.acm.org/doi/10.1145/3196494.3196523.

Basten, D 2017. Gamification. IEEE Software, no. 5. Viitattu 24.4.2021 http://dx.doi.org/10.1109/MS.2017.3571581.

Barnard, A. 2007. The Nursing Profession: Implications for AI and Natural Language Processing. International Conference on Natural Language Processing and Knowledge Engineering. Viitattu 19.4.2021 http://dx.doi.org/10.1109/NLPKE.2007.4368077.

Caballé, S. 2016. Formative assessment, Learning Data Analytics and Gamification. Elsevier Science & Technology. Viitattu 26.4.2021 https://luc.finna.fi/lapinamk/, ProQuest Ebook Central.

DeGroat, T. J. 2019. The History of JavaScript: Everything You Need To Know. Viitattu 12.4.2021 https://www.springboard.com/blog/history-of-javascript/.

Deepika, K., Tilekya, V., Mamatha, J & Subetha, T. 2020. Jollity Chatbot- A contextual AI Assistant. Third International Conference on Smart Systems and Inventive Technology (ICSSIT). Viitattu 27.4.2021 http://dx.doi.org/10.1109/ICSSIT48917.2020.9214076.

Dhaniwala, M. & Jaimini, U. 2016. JavaScript Empowered Internet of Things. 3rd International Conference on Computing for Sustainable Global Development (INDIACom). Viitattu 15.4.2021 https://ieeexplore.ieee.org/document/7724589.

Discord Inc 2021. What is Discord? Viitattu 11.4.2021 https://discord.com/safety/360044149331-What-is-Discord.

Gaba, I. 2021. What is GitLab and How To use It. Viitattu 14.4.2021 https://www.simplilearn.com/tutorials/git-tutorial/what-is-gitlab.

Herron, D. 2018. Node.js Web Development: Server-Side Development with Node 10 Made Easy, 4th Edition. Packt Publishing, Limited. E-kirja. Viitattu 26.4.2021 https://luc.finna.fi/lapinamk/, ProQuest Ebook Central.

Javatpoint 2021. React Router. Viitattu 11.4.2021 https://www.javatpoint.com/react-router.

Javascript.info 2021. An Introduction to JavaScript. Viitattu 14.4.2021 https://javascript.info/intro.

Javeed, A. 2019. Performance Optimization Techniques For ReactJS. IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT). Viitattu 20.4.2021 http://dx.doi.org/10.1109/ICECCT.2019.8869134.

Jylkäs, T. 2020. Service Design and Artificial Intelligence in Designing Human-Centered Digital Services. Lapin yliopisto. Taiteiden tiedekunta. Väitöskirja. Viitattu 24.4.2021 www.urn.fi/URN:ISBN:978-952-337-227-6.

Koramo, M., Brauer, S. & Jauhola, L. 2018. Digitalisaatio ammatillisessa koulutuksessa. Viitattu 8.5.2021 https://www.oph.fi/fi/tilastot-ja-julkaisut/julkaisut/digitalisaatio-ammatillisessa-koulutuksessa.

Larsen, R. 2013. Beginning of HTML and CSS. John Wiley & Sons, Incorporated. E-kirja. Viitattu 24.4.2021 https://luc.finna.fi/lapinamk/, ProQuest Ebook Central.

Lumio, J. 2019. Aivokalvotulehdus (meningiitti) aikuisilla. Viitattu 12.4.2021 https://www.terveyskirjasto.fi/dlk00558.

Mustafeez, A. Z. 2021. What is Visual Studio Code? Viitattu 14.4.2021 https://www.educative.io/edpresso/what-is-visual-studio-code.

Pedreira, O., Garcia, F., Piattini, M., Cortiñas, A. & Cerdeira-Pena, A. 2020. An Architecture for Software Engineering Gamification. Tsinghua Science and Technology, no. 6. Viitattu 25.4.2021 http://dx.doi.org/10.26599/TST.2020.9010004.

Petrović, V. M. 2018. Artificial Intelligence and Virtual Worlds – Towards Human-Level AI Agents. IEEE Access. Viitattu 19.4.2021 http://dx.doi.org/10.1109/ACCESS.2018.2855970.

Rahman, A., Mamun, A. & Islam, A. 2017. The Programming challenges of chatbot: Current and future prospective. IEEE Region 10 Humanitarian Technology Conference (R10-HTC). Viitattu 27.4.2021 http://dx.doi.org/10.1109/R10-HTC.2017.8288910.

React 2021. Tutorial: Intro to React. Viitattu 9.4.2021 https://reactjs.org/tutorial/tutorial.html.

Reese, M. R., Bhatia, A. 2018. Natural Language Processing with Java: Techniques for Building Machine Learning and Neural Network Models for NLP, 2nd Edition, Packt Publishing, Limited. E-kirja. Viitattu 4.5.2021 https://luc.finna.fi/lapinamk/, ProQuest Ebook Central.

Ribeiro, C., Antunes, T., Monteiro, M. & Pereira, J. 2013. Serious Games in Formal Medical Education: An Experimental Study. 5th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES). Viitattu 5.5.2021 http://dx.doi.org/10.1109/VS-GAMES.2013.6624240.

Rousku, K., Andersson, C., Stenfors, S., Lähteenmäki, I., Limnêll, J., Mäkinen, K., Kopponen, A., Kuivalainen, M. & Rissanen, O.-P. 2019. Pilkahduksia tulevaisuuteen. Tietopolitiikka, tekoäly ja robotisaatio hyvinvoinnin ja taloudellisen menestyksen mahdollistajana Suomessa. Valtiovarainministeriön julkaisuja 2019:22. Viitattu 23.4.2021 http://urn.fi/URN:ISBN:978-952-367-002-0.

Rousku, K., Linturi, R., Andersson, C., Stenfors, S., Lähteenmäki, I., Kärki, T. & Limnêll, J. 2017. Pilkahduksia tulevaisuuteen – digitalisaation ja robotisaation

mahdollisuudet. Valtiovarainministeriön julkaisuja 10/2017. Viitattu 20.4.2021 http://urn.fi/URN:ISBN:978-952-251-836-1.

Safari, H., Sabri, N., Shashavan, F. & Bahrak, B. 2020. An Analysis of Gitlab's Users and Projects Networks. 10th International Symposium on Telecommunications (IST). Viitattu 19.4.2021 http://dx.doi.org/10.1109/IST50524.2020.9345844.

Shaw, A. 2014. A System of Simple Sentence Parsing Rules to Produce "Answer Matching" Chatbots. 11th International Conference on Information Technology: New Generations. Viitattu 21.4.2021 http://dx.doi.org/10.1109/ITNG.2014.108.

Siriaraya, P., Visch, V., van Dooren, M. & Spijkerman, R. 2018. Learnings and Challenges in Designing Gamifications for Mental Healthcare: The Case Study of the Readysetgoals Application. 10th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games). Viitattu 5.5.2021 http://dx.doi.org/10.1109/VS-Games.2018.8493430.

Sosiaali- ja terveysministeriö 2016. Digitalisaatio terveyden ja hyvinvoinnin tukena. Sosiaali- ja terveysministeriön digitalisaatiolinjaukset 2025. Viitattu 17.4.2021 http://urn.fi/URN:ISBN:978-952-00-3782-6.

Tieranta, O. 2020. OmaDigi - Osaamisen mahdollistaminen digitaalisissa hyvinvointipalveluissa. Viitattu 21.4.2021 http://urn.fi/URN:NBN:fi:amk-202004165119.

Weil, A. 2017. Learn Meteor: Node.js and MongoDB JavaScript platform. J. Ross Publishing. E-kirja. Viitattu 26.4.2021 https://luc.finna.fi/lapinamk/, ProQuest Ebook Central.

Wirfs-Brock, A. & Eich, B. 2020. JavaScript: The First 20 Years. Viitattu 18.4.2021 https://dl.acm.org/doi/10.1145/3386327.