

ASIAKASNEUVOTTELUIDEN AUDION JALOSTAMINEN INFORMAATIOKSI

Tiitto Marko

Opinnäytetyö

Tieto- ja viestintäteknikka
Insinööri (AMK)

2024

Tieto- ja viestintäteknikka
Insinööri (AMK)

Tekijä	Marko Tiitto	Vuosi	2024
Ohjaaja	Aku Kesti		
Toimeksiantaja	OP Osuuskunta		
Työn nimi	Asiakasneuvotteluiden audion jalostaminen informaatioksi		
Sivumäärä	29		

OP Osuuskunnalla kertyy paljon asiakasneuvotteludataa verkkoneuvotteluista. Verkkoneuvotteluista tallentunutta dataa on vaikea hyödyntää ohjelmallisesti tai tilastollisesti, koska data on audiona ja videona. Audiona tallentuvassa viestinnässä on asiakkaista paljon hyödyntämätöntä tietoa, jota voitaisiin käyttää parantamaan yrityksen asiakaslähtöisyyttä.

Opinnäytetyön tavoitteena oli toteuttaa prototyyppi, jolla jalostetaan audiona olevaa asiakasneuvotteludataa hyödynnettävämpään muotoon eli tekstiksi. Lisäksi tekstidataa pyrittiin vielä jatkojalostamaan generatiivisella tekoälyllä ihmiselle säislettävämpään muotoon.

Lopputuloksena syntynyt kokonaisuus muodostui datantuottajasta ja datankuluttajasta. Datantuottaja kykeni litteroimaan useita audiotiedostoja kerralla. Datankuluttajan kautta litteraatteja voitiin tarkastella sekä niistä voitiin muodostaa palautetta ja tiivistelmiä generatiivisen tekoälyn avulla.

Audiotiedostojen litterointi toteutettiin Azure-pilvialustan palveluita hyödyntäen. Azureen rakennettiin työnkulku, joka pystyi litteroimaan monta tiedostoa kerralla ja lähettämään nämä ulkoiseen NoSQL-tietokantaan. Litteraattien tarkastelulle kehitetty web-sovellus toteutettiin käyttäen Spring Bootia ja Reactia. Sovelluksessa käyttäjä pystyi hakemaan litteraatit tietokannasta ja pyytämään litteraattista tiivistelmän ja palautteen.

Avainsanat

Azure, generatiivinen tekoäly, litterointi, tiedon jalostaminen

Study Programme in Information
and Communication Technology
Bachelor of Engineering

Author	Marko Tiitto	Year	2024
Supervisor	Aku Kesti		
Commissioned by	OP Osuuskunta		
Title	Refining Customer Negotiations Audio into Information		
Number of pages	29		

OP Cooperative accumulates a lot of customer negotiation data from online negotiations. It is difficult to use the recorded data from online negotiations programmatically or statistically because the data is in audio and video formats. There is a lot of unused information about customers in the recordings that could be used to improve the company's customer orientation.

The aim of this thesis study was to develop a prototype that refines audio from the online negotiations into a more usable format, i.e. into text. In addition, efforts were made to further refine the text data with generative artificial intelligence into a form that could be internalized easier by humans.

The audio files were transcribed using the services in Azure cloud platform. A workflow was built on Azure which could transcribe multiple audio files asynchronously and send them to an external NoSQL-database. The web-application for transcription viewing was developed using Spring Boot and React. Users could retrieve the transcriptions from the database and request feedback and summaries of them on the web-application.

The end result consists of a data producer and a data consumer. The data producer is capable of transcribing multiple audio files at the same time. Through the data consumer users can examine the transcriptions and generate summaries and feedback from the transcriptions with generative artificial intelligence.

Keywords Azure, data refinement, generative artificial intelligence, transcribing

SISÄLLYS

1	JOHDANTO	5
2	ASIAKASNEUVOTTELUDATASTA INFORMAATIOKSI	6
2.1	Asiakaslähtöisyys	6
2.2	Tiedon jalostaminen.....	6
2.3	Puheesta tekstiksi.....	7
2.3.1	Ohjelmallinen litterointi	7
2.3.2	Äänen litterointi Azuressa	8
2.4	Ketterät menetelmät	8
3	ASIAKASNEUVOTTELUDATAN JALOSTUSPALVELUN SUUNNITELMA .	10
3.1	Hyödynnettävät palvelut	10
3.2	Jalostuspalvelun arkkitehtuuri.....	10
4	ASIAKASNEUVOTTELUIDEN LITTEROINTI AZURESSA.....	13
4.1	Azuren palvelut.....	13
4.2	Logic App -työnkulku	14
5	NEUVOTTELUIDEN TARKASTELU -SOVELLUS.....	17
5.1	Sovelluksen arkkitehtuuri.....	17
5.2	Sovelluksen toteutus.....	18
5.2.1	Kehitystyökalut	18
5.2.2	Spring Boot -palvelu	18
5.2.3	React-asiakasohjelma	19
5.2.4	Käyttöliittymä	20
5.2.5	Sovelluksen isännöinti	21
5.3	Tapahtumien kulku	22
6	INFORMAATION JALOSTAMINEN GENERATIIVISELLA TEKOÄLYLLÄ ...	23
6.1	Promptaus	23
6.2	Generoidut vastaukset.....	24
6.3	Generoidun vastauksen tulkitseminen	25
7	POHDINTA.....	26
	LÄHTEET	28

1 JOHDANTO

Viimeisten vuosien aikana asiakaslähtöisyyttä konkretisoivat käsitteet ovat nousseet tutkimuksen sekä käytännön liiketoiminnan näkökulmasta sovelletuiksi ja suosituiksi tavoiksi johtaa, tutkia ja viestiä liiketoimintaa. Lisäksi asiakaslähtöisyyden keskeiset mittarit auttavat sitomaan asiakastarpeet yrityksen menestymisen mittareihin. (Saarijärvi 2017, 104.)

Asiakaslähtöisyys on keskeinen periaate menestyksessä palvelukehityksessä. OP Osuuskunnalla (myöhemmin OP) asiakaslähtöisyyden kehittäminen on otettu erityisesti huomioon vuodesta 2019 asti. Tavoitteena OP:lla on ollut olla markkinoiden asiakaslähtöisin ja edistyksellisin. (Kangas 2023.)

Työskennellessäni OP:n asiakasviestintävälineiden parissa huomioin kuinka paljon OP:lla tehdään etänä asiakaspalvelua. Tällöin asiakasneuvotteludataa kertyy monista eri lähteistä useissa eri muodoissa, kuten puheluina ja verkkoneuvotteluina. Huomasin tässä mahdollisuuden parantaa asiakaslähtöisyyden mittaamista ja kehittämistä. Jotta asiakaslähtöisyyttä pystytään mittaamaan mahdollisimman kattavasti, jokaisesta asiointikanavasta tallentuvan datan tulisi olla helposti käytettävissä.

Opinnäytetyön tavoitteena on kehittää prototyyppi, jolla mahdollistetaan äänenä tallennetun asiakasneuvotteludatan hyödyntäminen monipuolisesti. Neuvotteluiden sisältö muunnetaan kirjalliseen muotoon, jonka jälkeen kirjallisena olevista neuvotteluista voidaan muodostaa tiivistelmiä ja palautetta generatiivisen tekoälyn avulla.

Työssä on hyödynnetty generatiivisia tekoälysovelluksia auttamaan kehitystyössä ja ideoinnissa. Generoituja vastauksia ei olla käytetty suoraan kirjoituksessa eikä kehitystyössä. Ainoastaan testidatana käytettiin lähes muokkaamattomia vastauksia.

Työssä ei arvioida käytettävien palveluiden tarkkuutta tai laatua, vaan oletetaan että palvelut tuottavat tarpeeksi luotettavaa tietoa. Lisäksi työtä ei rajoiteta asiakastietojen käsittelyyn liittyvillä säännöksillä.

2 ASIAKASNEUVOTTELUDATASTA INFORMAATIOKSI

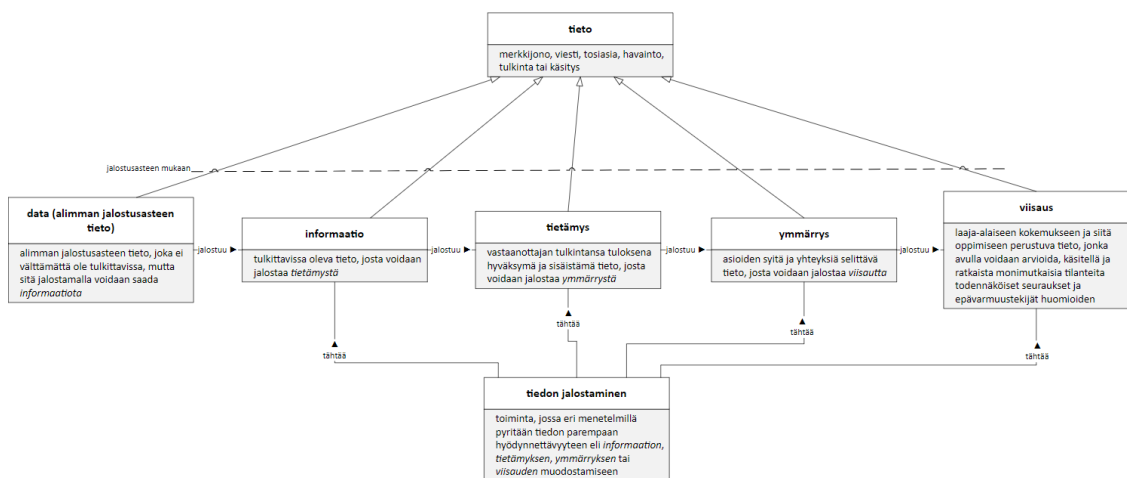
2.1 Asiakslähtöisyys

Asiakslähtöisyyden avulla pyritään ymmärtämään, mitkä tuotteet ja palvelut ovat asiakkaalle lopulta merkityksellisiä. Mitä paremmin asiakkaita pystytään ymmärtämään, sitä merkityksellisiä palveluita pystytään kehittämään ja tarjoamaan. Asiakkaat, jotka kokevat palvelut ja tuotteet heille merkityksellisiksi, haluavat todennäköisemmin jatkossakin asioida yrityksen kanssa. (Saarijärvi 2017, 101.)

Asiakstarpeiden ymmärtäminen vaatii jatkuvaa yrityksen ja asiakkaan välisen toiminnan seuraamista. Eri asiointikanavat helpottavat OP:illa asioimista, mutta esimerkiksi puhelusta tallentuva asiakasneuvotteludata on audiota, joka on vaikeasti tulkittavaa analytiikan kannalta. Jotta puhelusta ja verkkoneuvotteluista kertynyttä asiakasdataa voidaan hyödyntää helposti, audio tulisi jalostaa tulkittavampaan muotoon.

2.2 Tiedon jalostaminen

Ohjelmistoprojektissa jalostetaan alinta tiedon muotoa eli dataa seuraavaan hyödynnettävämpään muotoon eli informaatioon. Informaatiota pyritään jalostamaan tietämykseksi (kuvio 1) generatiivisen tekoälyn avulla. (Finto 2018). Data on tässä projektissa asiakasneuvotteluista nauhoitettua audiota.



Kuvio 1. Tiedon jalostaminen (Finto 2018)

Audiona oleva data halutaan muuttaa mahdollisimman tarkasti tekstiksi. Teksti on informaatiota, jota on helppo tulkita ohjelmallisesti. Teksti on siis sopiva tietomuoto esimerkiksi analytiikkaa varten, mutta suuret tekstimassat eivät välttämättä ole ihmisille helposti tulkittavia.

Saadun informaation jalostamista tietämykseksi ja ymmärrykseksi yritetään parantaa generatiivisen tekoälyn ChatGPT:n avulla. Generatiivisella tekoälyllä tekstiä voidaan muokata helpommin tulkittavaan muotoon esimerkiksi tiivistämällä tekstiä.

2.3 Puheesta tekstiksi

Puheen ja toiminnan purkaminen kirjoitettuun muotoon on litterointia. Litteroinnista puhutaan usein laadullisen tutkimuksen tekemisessä, sillä laadulliset aineistot ovat usein puhetta ja vuorovaikutusta ihmisten välillä. Litteraatti on tilanteen tapahtuman kuvaus. Litteraatin tarkkuus valitaan haluttujen vastausten ja analyysimenetelmien mukaisesti. Tarkkuudella tarkoitetaan, kuinka paljon litteraattiin kuvataan puheen sisältämiä taukoja, äänen painoja tai keskustelijoiden eleitä. Karkean litteraatin tulisi kuitenkin pystyä vähintään kuvaamaan mitä sanotaan. Aineistojen litterointi on työläs prosessi, mutta sitä voidaan automatisoida nykyään hyödyntämällä eri ohjelmistoja ja palveluita. (Kallio 2023.)

2.3.1 Ohjelmallinen litterointi

Puheen muuntaminen tekstiksi eli Speech to Text on monivaiheinen prosessi. Ohjelmisto, joka muuntaa puhetta tekstiksi, toimii kuuntelemalla ääntä ja toimittamalla sanatarkan litteraatin.

Puhe on sarja erilaisia värähtelyitä, jotka pystytään tulkitsemaan. Puhe tekstiksi-ohjelmisto toimii poimimalla nämä värähtelyt ja kääntämällä ne digitaaliselle kielelle analogi-digitaalimuuntimen kautta. Analogi-digitaalimuunnin poimii ääniä äänitiedostosta, mittaa aallot erittäin yksityiskohtaisesti ja suodattaa ne erottaakseen asiaankuuluvat äänet. Esimerkiksi keskusteluissa olevat tauot suodattuvat

tällöin pois. Sen jälkeen äänet segmentoidaan sekuntien sadasosiksi tai tuhannesosiksi ja sovitetaan foneemiin. Foneemi on ääniyksikkö, joka erottaa sanan toisesta missä tahansa kielessä. Esimerkiksi englannin kielessä on noin 40 foneemia. Foneemit ajetaan sitten verkon läpi matemaattisen mallin avulla, joka vertaa niitä tunnettuihin lauseisiin, sanoihin ja lauseisiin. Lopuksi teksti esitetään tekstinä tai tietokonepohjaisena vaatimuksena äänen todennäköisimmän version perusteella. (Amazon Web Services 2023.)

2.3.2 Äänen litterointi Azuressa

Ohjelmistoprojektissa asiakasneuvotteluiden litteroimiseen hyödynnetään Microsoftin pilvialusta Azuren tekoälypalveluiden puhepalvelua. Palvelussa on puheesta tekstiksi litteroinnille kaksi vaihtoehtoa: Real-time speech to text ja Batch transcription.

Real-time speech to text -palvelu litteroi puhetta reaaliaikaisesti. Tämä palvelu on tarkoitettu tilanteisiin, joissa litteraattia tarvitaan heti käytettäväksi. Esimerkiksi tekstityksen tuottaminen suoriin lähetyksiin on yksi käyttötapaus tälle palvelulle. Vuorostaan Batch transcription -palvelu pystyy litteroimaan suuria määriä tallennettua audiota, kuten päivän aikana tallennettuja puheluita, ja palauttamaan litteraatit asynkronisesti. (Microsoft Learn 2023d.) Käyttämällä Batch transcriptionia pystytään tuottamaan ratkaisu, jolla saadaan litteroitua suuret määrät verkkoneuvotteluita.

2.4 Ketterät menetelmät

Ennen ohjelmistoprojektin aloitusta tulisi valita kehitysmenetelmä, joka sopii toteutukseen parhaiten. Kehitysmenetelmän valitseminen ja siinä pidättäytyminen on tärkeä osa kehitysprojektia, sillä tietyt menetelmät sopivat paremmin tiettyihin projekteihin kuin toiset.

Scrum on yksi monista tuotekehityksen johtamistekniikoista, joka kuuluu ketterän ohjelmistokehityksen kattavaan kategoriaan. Ketterät menetelmät on suunniteltu

tukemaan joustavia, kestäviä ja iteratiivisia menetelmiä. Scrum sopii erityisen hyvin web- ja mobiiliohjelmistojen kehitykseen, jossa keskitytään kehittämään modulaarisia ja mitattavia ominaisuuksia, joita voidaan joutua muuttamaan nopeasti. (Green 2016, 1.)

Ohjelmistoprojekti toteutetaan mukaillen Scrum-kehitysmetodia, koska projektissa halutaan edetä nopealla tahdilla ja käytettävien palveluiden yhteensopivuudessa on epävarmuutta. Projektissa pidetään viikon mittaiset sprintit, jotta voidaan varmistaa kehityksen edistyminen nopeasti. Sprintin lopussa pidetään retrospektiivi ja suunnittelupalaveri. Retrospektiivissä arvioidaan edellisen sprintin tulokset ja suunnittelupalaverissa suunnitellaan seuraavan sprintin tavoitteet.

3 ASIAKASNEUVOTTELUDATAN JALOSTUSPALVELUN SUUNNITELMA

Palvelu muodostuu datantuottajasta ja datankuluttajasta. Datantuottaja muodostaa asiakasneuvotteluista litteraatteja ja datankuluttajan kautta käyttäjä voi hyödyntää litteraatteja eri tavoin. Käyttäjä voi hakea generatiivisen tekoälyn generoimia tiivistelmiä ja palautteita litteraateista.

3.1 Hyödynnettävät palvelut

Ohjelmistoprojekti aloitetaan kartoittamalla toteutukseen vaadittavat palvelut. Aloitusvaiheessa kannattaa pysyä avarakatseisena ja vertailla saatavilla olevien palveluita ja kehitysvälineitä. Useimmiten käytettävät palvelut rajoittuvat vahvasti yrityksen tai yhteistyökumppanin mukaisesti, sillä yrityksillä voi olla yhteistyösopimuksia esimerkiksi tietyille pilvipalveluntarjoajalle tai käytössä olevat palvelut ovat rakennettu tietylle alustalle. Jos valintaan voi vaikuttaa, kannattaa huomioida muun muassa palveluiden kustannus, muokattavuus ja jatkokehityksen mahdollisuudet.

Ohjelmistoprojektissa käytetään Microsoftin Azure pilvipalveluita. Azuren käyttöön päädyttiin, sillä Microsoft on OP:n teknologiakumppani ja Azuressa on saatavilla työssä tarvittavat ominaisuudet, kuten puheen litterointi. Azuren palveluiden lisäksi käytetään MongoDB:n NoSQL-tietokantajärjestelmää ja OpenAI:n ChatGPT-tekoälysovellusta. MongoDB:n järjestelmän käyttöön päädyttiin, koska se on helppokäyttöinen ja se tarjoaa valmiit rajapinnat tietokannoille. Vuorostaan ChatGPT-sovellukseen päädyttiin, sillä se tarjoaa hyvin dokumentoidut rajapinnat sovelluksen käytölle sekä OpenAI:n palvelut tulevat saataville Azureen.

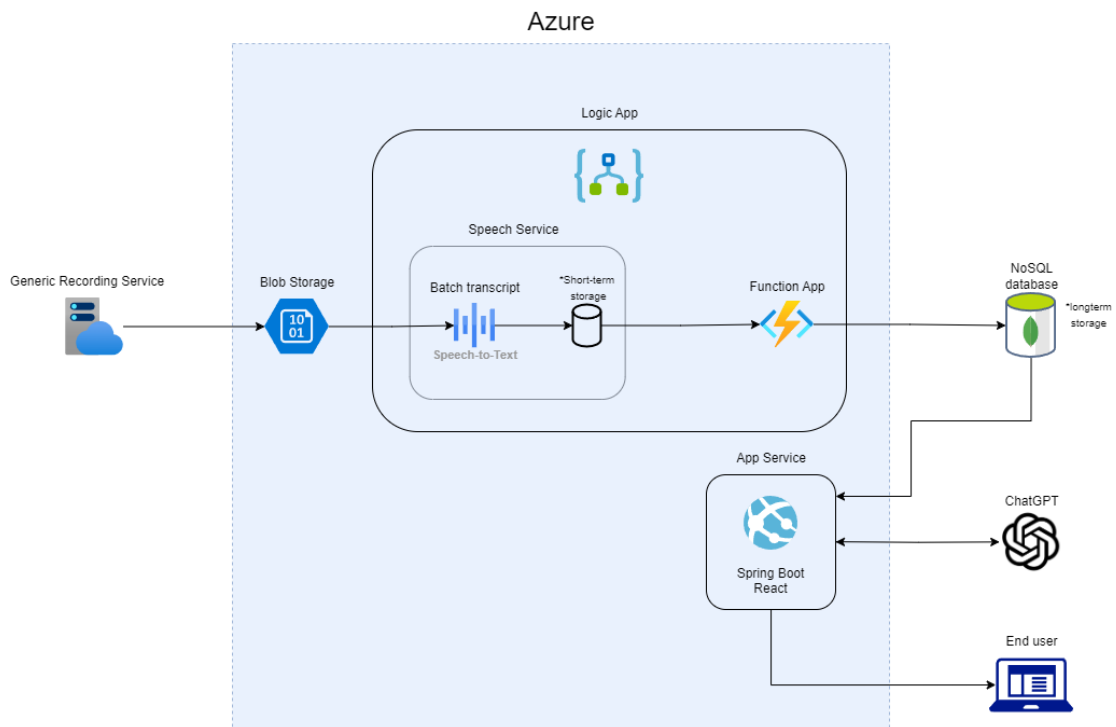
3.2 Jalostuspalvelun arkkitehtuuri

Kun käytettävissä olevat palvelut on selvitetty, voidaan aloittaa palvelun arkkitehtuurin suunnitteleminen. Onnistunut arkkitehtuuri kuvaa järjestelmän kokonaisrakenteeseen ja käyttäytymiseen liittyviä suunnittelupäätöksiä. Arkkitehtuuri auttaa myös sidosryhmiä ymmärtämään ja analysoimaan, kuinka järjestelmä saavuttaa olennaiset ominaisuudet. (Software Engineering Institute 2023.)

Arkkitehtuurikuvauksessa palvelun eri osat ja niiden mahdolliset liitännäisyydet toisiin palveluihin ilmaistaan selkeästi. Näin nähdään mitä kukin eri osa tarvitsee toimiakseen. Lisäksi voidaan hahmottaa, missä paikoissa ja missä muodossa palvelun data liikkuu.

Eryteisesti finanssialaan kohdistuvien tarkkojen säännösten ja tietoturva-vaatimusten takia datan säilyttäminen ja liikkuminen tulee huomioida heti suunnitteluvaiheessa. Finanssialalla toimiminen on luvanvaraista ja luvan saaneet toimijat kuuluvat Finanssivalvonnan jatkuvan valvonnan piiriin (Finanssivalvonta 2020). Tämän projektin toteutusta ei kuitenkaan rajata säännöksillä, koska tarkoituksena on toteuttaa prototyyppi ja työssä käytettävät datat eivät ole oikeita asiakasneuvotteluita.

Kuviossa 2 kuvataan kaikki ohjelmistoprojektissa käytettävät palvelut. Dataliikennettä kuvataan palveluiden välisten nuolien avulla. Lisäksi datan väliaikainen ja pysyvä säilytys on merkattu.



Kuvio 2. Palvelun arkkitehtuurin kuvaus

Kun arkkitehtuuri on kuvattu, voidaan helpommin tunnistaa projektin kannalta kriittisimmät osa-alueet, joihin kannattaa ensisijaisesti panostaa. Projektin tapauksessa kuviosta 2 huomataan Logic App -kokonaisuuden olevan toteutuksen ensimmäinen suurempi vaihe, joka on myös vastuussa tuotteistettavan datan tuottamisesta. Täten Logic Appin kehitykseen keskitytään ensimmäisenä, sillä muuten myöhemmille vaiheille ei ole käytettävää dataa.

4 ASIAKASNEUVOTTELUIDEN LITTEROINTI AZURESSA

4.1 Azuren palvelut

Pilvialusta Azuressa hyödynnetään neljää eri alustan tarjoamaa palvelua toteuttamaan ohjelmistoprojektin datantuottaja. Hyödyntämällä pilvipalveluita prototyyppi pystytään toteuttamaan ilman suurta ajallista ja rahallista sijoitusta. Azuressa käytettävät palvelut ovat

- Azure Logic Apps
- Azure Functions
- Azure Blob Storage ja
- Speech Service.

Azure Logic Apps on low-code-no-code -pilvialusta, joka on tarkoitettu automatisoimaan järjestyksessä tapahtuvia monivaiheisia tehtäviä eli työkulkuja ilman ohjelmiston kirjoittamista. Alusta sisältää monia valmiita operaatioita, joilla pystytään rakentamaan nopeasti erilaisia työkulkuja. (Microsoft Learn 2023c.)

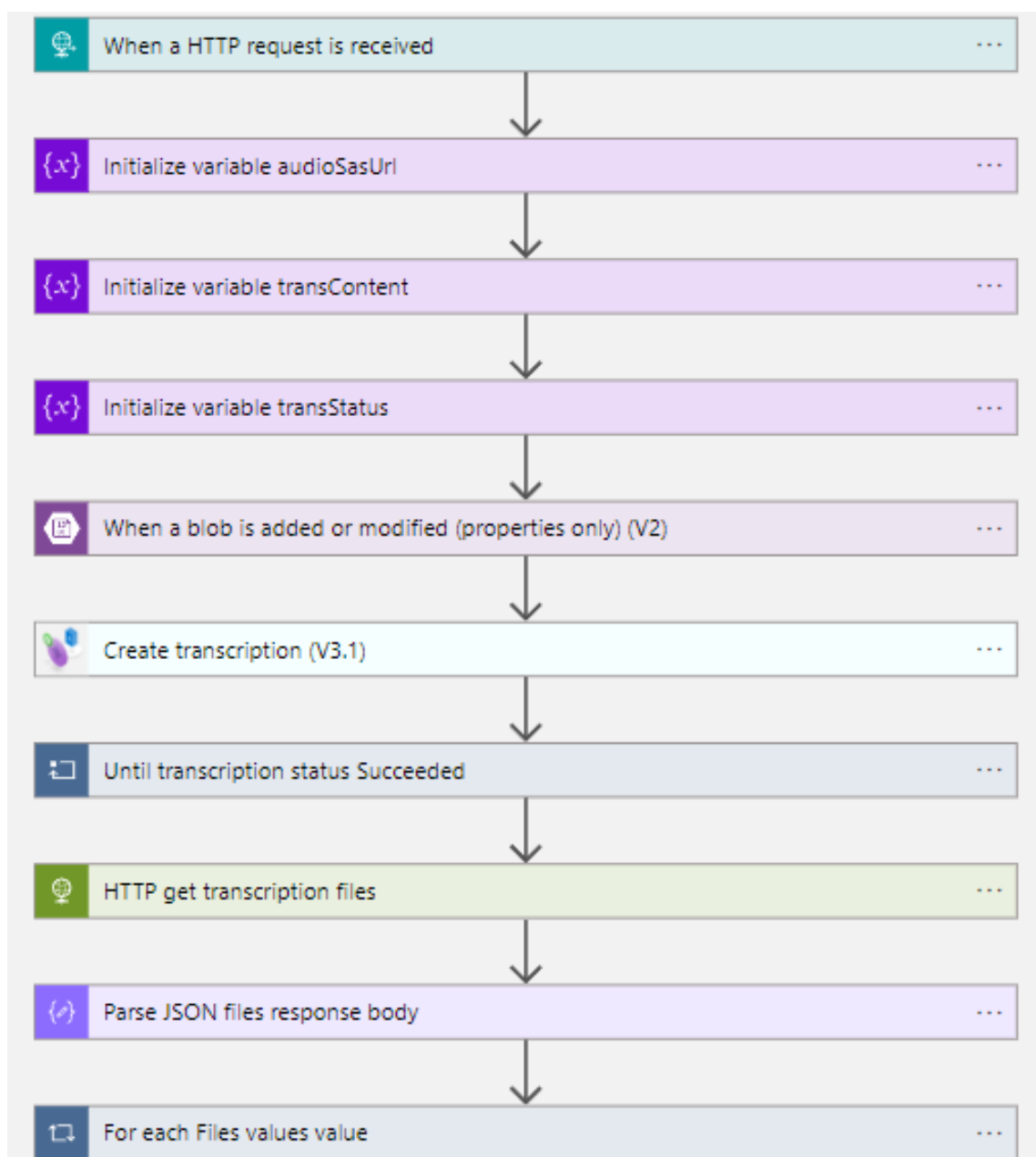
Azure Functions on palvelimeton tietojenkäsittely ratkaisu. Palvelimeton tietojenkäsittely on pilvipalvelumuoto, jossa ohjelmiston ajoon tarvittavat resurssit vuokrataan palveluntarjoajalta. Tällöin kehittäjän ei tarvitse huolehtia muusta kuin tarvittavan ohjelmiston kehittämisestä. (Microsoft Learn 2023a.) Functionsiin voidaan tehdä monia eri Function Apejä, joita voidaan käyttää kutsumalla niitä esimerkiksi HTTP-pyyntöillä.

Blob storage on Azuren pilvitallennusratkaisu. Se on tarkoitettu suurien strukturoimattomien datamäärien, kuten videoiden tai kuvien, säilyttämiseen. (Microsoft Learn 2023b.)

Speech Service sisältää Azuren tekoälypalveluiden puhetta käsittelevät palvelut, kuten puheen kääntämisen ja puheesta tekstiksi muuntamisen. Palvelua voidaan käyttää omilla puhemalleilla tai suoraan valmiina palveluna. (Microsoft Learn 2023e.)

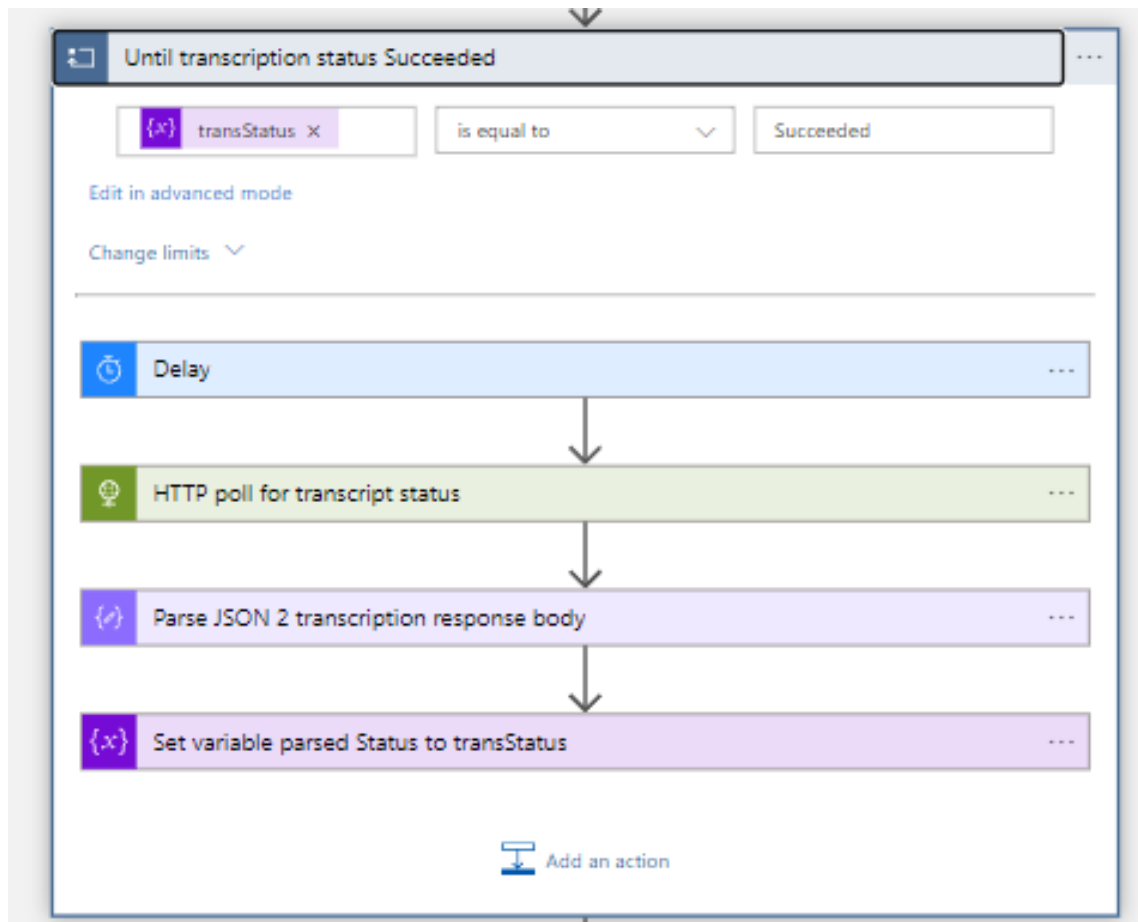
4.2 Logic App -työnkulku

Logic Appia hyödynnetään tuottamaan nopeasti ratkaisu litteroimaan useita audiotiedostoja kerralla. Kuvioista 3 nähdään kehitetty työnkulku. Aluksi initialisoidaan myöhemmin tarvittavat muuttujat. Seuraavaksi odotetaan, että Blob Storageen tulee muutoksia. Kun muutos tapahtuu, luodaan litteraatti kaikista Blob Storageessa olevista tiedostoista. Tämän jälkeen odotetaan Speech Servicen prosessoivan tiedostot Batch Transcriptionilla. Sitten Speech Servicen luomat litteraatit lähetetään tietokantaan.



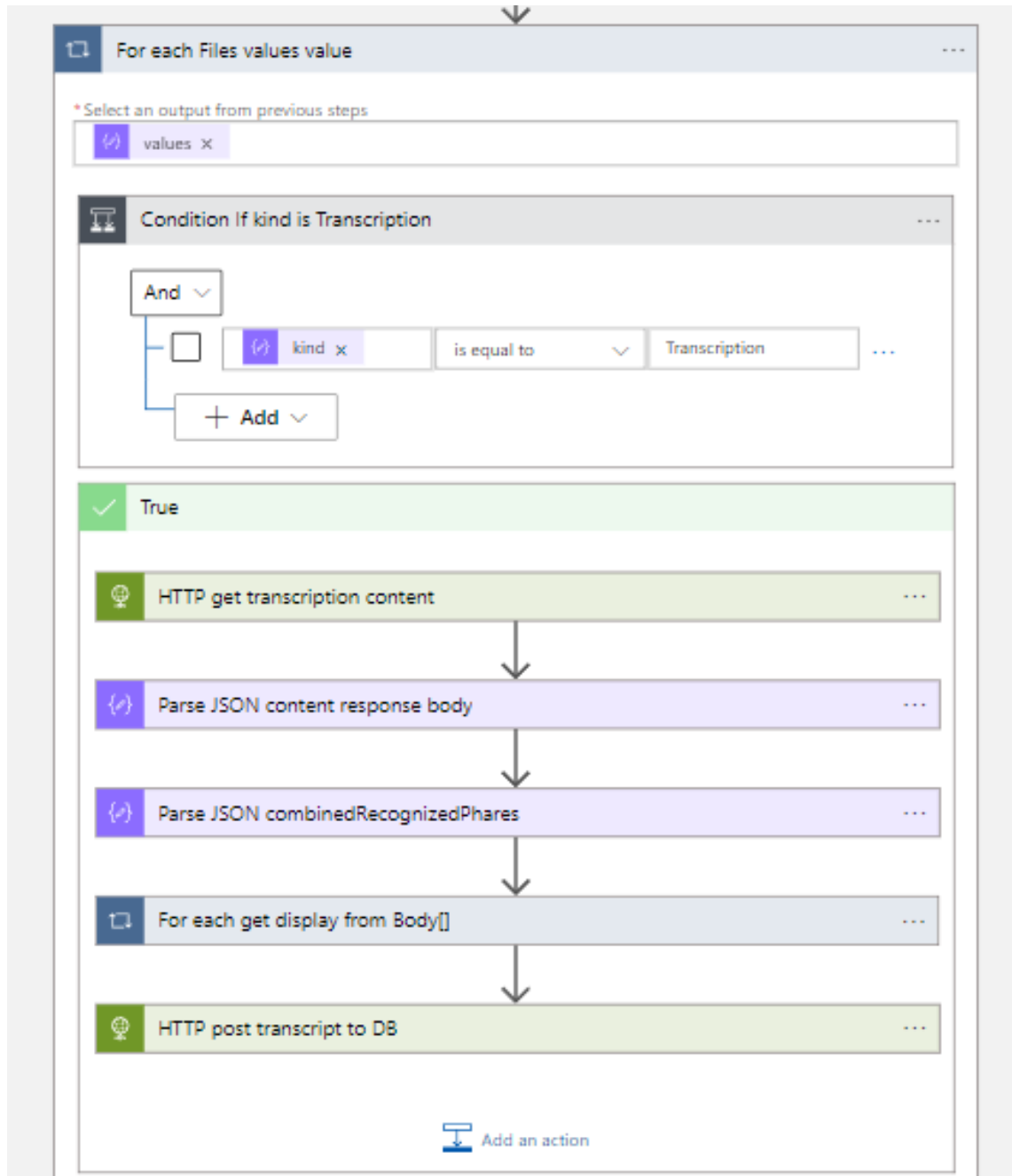
Kuvio 3. Logic App -työnkulku

Vaiheen 7 tarkempi kuvaus nähdään kuvioista 4. Tässä vaiheessa pyydetään Speech Servicen litteroinnin statusta, kunnes statuksen arvoksi saadaan "Succeeded".



Kuvio 4. Litterointi statuksen pyyntö

Viimeisessä vaiheessa (kuvio 5) Speech Servicen luomista tiedostoista haetaan litteraattiosuudet ja ne lähetetään Function Appin avulla tietokantaan. Tämä Function App sisältää Python-ohjelman, joka lähettää muodostetun objektin NoSQL-tietokantaan. Vaihe on asetettu peräkkäin suoritettavaksi, jotta käytetyt muuttujat kuvastavat varmasti oikeaa tietoa.



Kuvio 5. Litteraatin sisällön lähetyt tietokantaan

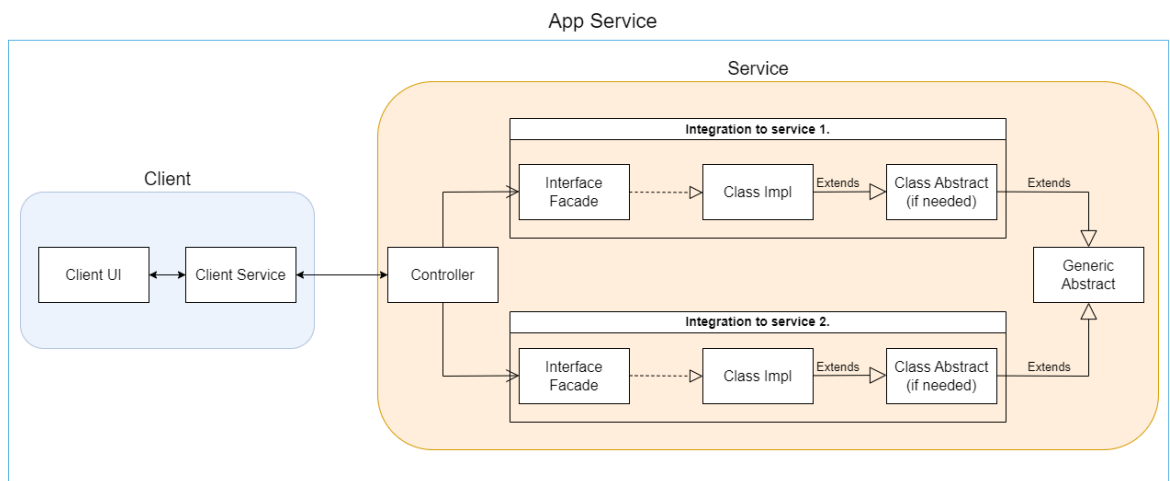
Tietokantaan lähetettävät objektit sisältävät litteraatin, aikaleiman ja lähdetiedoston nimen. Aikaleima ja lähdetiedoston nimi lisättiin, jotta litteraatit voidaan yhdistää alkuperäiseen tallenteeseen helpommin.

5 NEUVOTTELUIDEN TARKASTELU -SOVELLUS

5.1 Sovelluksen arkkitehtuuri

Tavoitteena on pitää sovelluksen toteuttaminen ja mahdollinen jatkokehitys suoraviivaisena. Tähän pyritään pitämällä asiakas- ja palvelinohjelman välinen kommunikaatio yksinkertaisena ja ohjelmat omina kokonaisuuksinaan. Lisäksi lähes kaikki datan käsittely toteutetaan palvelimen puolella, jotta käyttöliittymän puolella ei tarvitse huolehtia muusta kuin ulkoasun luomisesta ja datan visualisoinnista.

Usein monimutkaiset järjestelmät voidaan pelkistää muutamiin käyttötapauksiin, jotka alijärjestelmä tuo näkyville. Tällöin asiakasohjelman ei tarvitse tietää järjestelmästä muuta kuin näkyville tuotu osa. Tätä rakennetta kutsutaan Facadeksi. (Singh, Ianculescu & Torje 2018, 97.) Tämä rakennemalli sopii ohjelmistoprojektin aiemmin mainittuihin tavoitteisiin. Kuvioista 6 nähdään Facade-rakennemallin mukainen arkkitehtuuri sovellukselle.



Kuvio 6. Sovelluksen arkkitehtuurin kuvaus

Sovelluksen palvelinohjelmiston puolelle voidaan tarvittaessa lisätä uusia integraatioita, eikä asiakasohjelman tarvitse silti olla yhteydessä kuin yhteen rajapintaan. Uusien integraatioiden tekeminen ei ole myöskään muista palvelinohjelmiston implementaatioista riippuvainen, mikä helpottaa ohjelmiston laajentamista.

Lisäksi hyödynnetään perinnöllisyyttä sekä abstraktiota vähentämään ohjelmistokoodin turhaa toistuvuutta ja pitämään tietyt osat, kuten integraatiot, erillisinä toisistaan.

5.2 Sovelluksen toteutus

5.2.1 Kehitystyökalut

Ohjelmistoprojektin kehityksessä käytetään Microsoftin tarjoamaa Visual Studio Code -ohjelmointiympäristöä. Tämä ohjelmointiympäristö sisältää paljon kehitystä nopeuttavia ominaisuuksia kuten integroidun komentorivin ja monipuoliset laajennukset.

Kehityksessä hyödynnetään ohjelmointiympäristön Azure-, Java- sekä Spring Boot -laajennuksia. Laajennukset tuovat erilaisia oikoteitä ja integraatioita ympäristöön, mikä nopeuttaa ja helpottaa kehitystyötä.

5.2.2 Spring Boot -palvelu

Sovelluksen pohjana käytetään Spring Bootia. Spring Boot on Spring Frameworkille rakennettu työkalu, joka tekee web-sovellusten ja mikropalveluiden kehittämisestä helpompaa ja nopeampaa mahdollistamalla muun muassa automaattisen konfiguroinnin ja itsenäisten sovellusten luomisen (IBM 2023). Spring Boot -sovelluksen rungon voi generoida Spring Initializr -laajennusta hyödyntäen tai saman pohjan voi ladata myös Spring Initializr -verkkosivulta (Kuvio 7). (VMWare 2023).



Project

Gradle - Groovy
 Gradle - Kotlin
 Java
 Kotlin
 Groovy

Maven

Spring Boot

3.2.1 (SNAPSHOT)
 3.2.0
 3.1.7 (SNAPSHOT)
 3.1.6

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging Jar War

Java 21 17

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Rest Repositories WEB
Exposing Spring Data repositories over REST via Spring Data REST.

Kuvio 7. Spring initializr -verkkosivu (VMware 2023)

Kuviossa 7 nähdään projektissa käytettävät versiot ja riippuvuudet. Koska sovelusta ei isännöidä omassa ympäristössä, tulee sovelluksessa käytettävät versiot valikoida hyödynnettävän isännöintipalvelun tukemien versioiden mukaan. Muuten ohjelmointikieli ja koontiautomaatiotyökalu voidaan valita oman mieltymyksen mukaan.

5.2.3 React-asiakasohjelma

Käyttöliittymän toteuttamiseen käytetään Reactia. React on JavaScriptille tehty käyttöliittymäkirjasto, joka sujuvoittaa käyttöliittymien ohjelmointia. React-sovelluksen ajamiseen tarvitaan JavaScript-ajoympäristö. Lisäksi tarvitaan paketinhallintatyökalu tarvittavien pakettien lataamiseen. Projektissa käytetään Node.js-ajoympäristöä ja npm-paketinhallintatyökalua.

React-sovelluksen rungon voi generoida samaan tapaan kuin aiemman Spring Boot -sovelluksen. Ajamalla komento `"npx create-react-app my-app"` luodaan kansio "my-app", jonka sisälle luodaan tarvittavat rakenteet ja kirjastot sovelluksen ajamiseen. Komentoon voidaan lisätä malli lisäämällä `"--template [template name]"` aiemman komennon jatkoksi. Projektissa käytetään TypeScript-mallia. Tulee huomioida, että sovellus generoidaan siihen paikkaan, missä komento ajetaan.

5.2.4 Käyttöliittymä

Sovelluksen käyttöliittymä pidetään yksinkertaisena, koska prototyypissä halutaan ensisijaisesti vain esitellä mahdollisia toiminnallisuuksia. Ulkoasusta saadaan kuitenkin interaktiivinen ja modernin näköinen (Kuvio 8) helposti hyödyntämällä Chakra-UI-komponenttikirjaston valmiita React-komponentteja ja TanStackin React Table -kirjaston taulukkorunkoa.

Online meeting Transcripts

[Get Transcripts](#)

_ID	SCRIPT	CREATED_TIME	FILE_NAME
12	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer odio tellus, semper a...	2012-12-12T12:12:12Z	demoName.wav.json
11	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer odio tellus, semper a...	2011-11-11T11:11:11Z	demoName.wav.json

Kuvio 8. Web-sovelluksen käyttöliittymä

Chakra-UI nopeuttaa web-sivun kehittämistä todella paljon, sillä kehittäjän ei välttämättä tarvitse tehdä ollenkaan omia tyylejä komponenteille. Lisäksi Chakra-UI sisältää yleisimmät web-sivuilla käytetyt interaktiiviset osat, kuten kuviossa 9 oleva modaali.

Online meeting Transcripts

[Get Transcripts](#)

_ID	SCRIPT	CREATED TIME	FILE NAME
12	Transcripts Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer odio tellus, semper ac varius pretium, varius eget purus. Phasellus scelerisque nulla ex, eget vestibulum leo egestas sit amet. Maecenas sollicitudin eu eros nec ultrices. Suspendisse ex dolor, vulputate a sem interdum, congue pulvinar urna. Fusce mauris enim, dignissim suscipit magna eu, pharetra convallis enim.		son
11	Shortened Generated script here		son

Feedback
Generated feedback here

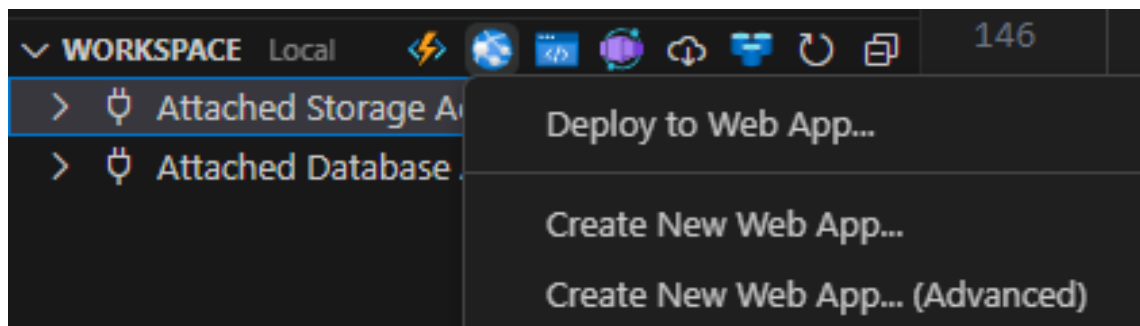
[Shorten](#) [Feedback](#)

Kuvio 9. Käyttöliittymän modaali

5.2.5 Sovelluksen isännöinti

Sovelluksen viemistä isännöitäväksi varten projekti tulee paketoita yhdeksi kokonaisuudeksi, jota voidaan ajaa Azuren HTTP-pohjaisessa isännöintipalvelu - Azure App Servicessä. Tämä toteutetaan lisäämällä projektin juureen generoituun POM-tiedostoon frontend-maven-plugin-liitännäinen. Liitännäisen avulla pystytään helposti tuomaan käyttöliittymä irtonaisena koontiversion mukaan. Liitännäisellä määritetään projektiin paikallisesti asennettavat Node.js- ja npm-versiot, joita käytetään käyttöliittymän koontiin ja ajamiseen. (Sletteberg 2022). Lisäksi käytetään maven-resources-plugin-liitännäistä kopiomaan käyttöliittymän koonti Spring Bootin koontin mukaan.

Sovellus voidaan viedä helposti Azureen isännöitäväksi Visual Studio Coden Azure-laajennuksella (kuvio 10). Sovellus voidaan joko viedä entisen sovelluksen tilalle tai siitä voidaan tehdä uusi sovellus Azure App Serviceen.

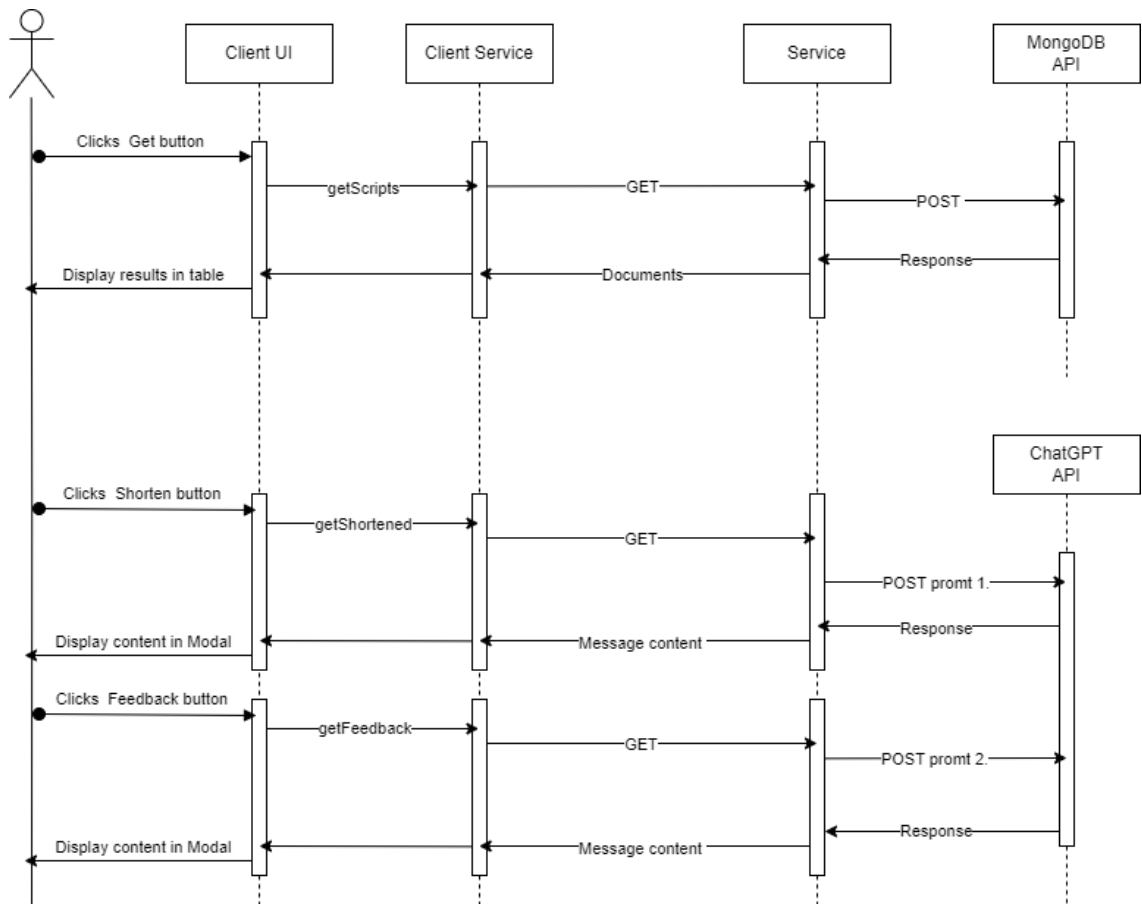


Kuvio 10. Sovelluksen vienti Azure App Serviceen.

Jos Maven koontiautomaatiotyökalu ei ole asennettu kehityksessä käytettävälle kehityskoneelle, tulee projekti paketoita ensin `./mvnw package` komennolla. Tämän jälkeen voidaan käyttää laajennusta ja ohittaa oletuspaketointi, jolloin viedään isännöitäväksi aiemmalla komennolla paketoitu projekti.

5.3 Tapahtumien kulku

Sovelluksen arkkitehtuurin ansiosta tiedon saaminen asiakasohjelmalle ulkoisista ohjelmointirajapinnoista on suoraviivaista. Kuviosta 11. nähdään kuinka käyttäjän tekemät interaktiot etenevät sovelluksen eri osien läpi.



Kuvio 11. Sekvenssikaavio Web-sovelluksen tapahtumista

Interaktio aloittaa tapahtumaketjun vasemmasta reunasta edeten oikealle ulkoiseen rajapintaan. Rajapinta palauttaa sitten pyydetyn datan, josta palvelu palauttaa vain tarvittavan osan asiakasohjelmalle. Esimerkiksi ChatGPT:n ohjelmointirajapinnasta saadusta vastauksesta palautetaan ainoastaan generoitu vastaus.

6 INFORMAATION JALOSTAMINEN GENERATIIVISELLA TEKOÄLYLLÄ

6.1 Promptaus

Generatiiviset tekoälysovellukset, kuten ChatGPT, toimivat pääasiassa tekstikomennoilla. Komennot tulee muotoilla tarpeeksi selkeästi, jotta saadaan parempia tuloksia. (Ojanperä 2023, 48.) Annetuista komennoista käytetään vakituisesti termiä prompt.

Yleisessä käytössä generatiivisten tekoälysovelluksien vastaukset vaikuttavat todella päteviltä, mutta tasalaatuisten vastausten saaminen voi olla haastavaa. Tarpeeksi varmasti toimivan promptin kehittämiseen tarvitaankin kohtuudella iterointia ja testausta. Jos tekoälylle syötettävässä promptissa ohjeistetaan ja rajataan vastausta liian paljon, vastauksista voi tulla yksipuolisia. Vuorostaan liian vähäinen ohjeistus voi antaa vastauksia ohi aiheesta.

ChatGPT:n ohjelmointirajapintaa käytettäessä prompti on hyvä jaotella osiksi samaan tapaan kuin sovellusta käytettäessä. Esimerkiksi promptin jakaminen konteksti-, ohje- ja sisältöosiin helpottaa promptin muodostamista. Kuviossa 12. jaottelua käyttäen on muodostettu prompti antamaan asiakaspalvelijalle palautetta asiakasneuvottelusta.

Konteksti:

Olet pankin asiakaskokemusten asiantuntija. Saat asiakkaan ja asiakaspalvelijan välisen keskustelun. Tarkoituksenasi on antaa palautetta, kuinka asiakaspalvelija suoriutui keskustelusta. Anna rakentavaa palautetta ja kehitysideoita keskustelussa tapahtuneesta kommunikaatiosta.

Ohjeet vastaukselle:

Vastauksen täytyy olla kolme virkettä. Vastauksen tulee olla helposti ymmärrettävä

Keskustelu:

<Asiakasneuvottelun sisältö>

Kuvio 12. Prompti palautteen saannille

6.2 Generoidut vastaukset

Toiminnallisuuden esittelyä varten ChatGPT:llä generoitiin keskustelu asiakkaan ja pankin asiakaspalvelijan välillä lainan hakemisesta. Kuvioista 13. nähdään koko generoidun keskustelun sisältö.

Hei! Mietin tässä lainatarjouksia ja erilaisia vaihtoehtoja, joita olis tarjolla. Tervetuloa! Minkä tyyppistä lainaa olet harkinnut ja millaisilla ehdoilla? Mulla olis hakusessa noin 50 000 euron laina ja mietin, millaisia vaihtoehtoja teillä on korkojen ja takaisinmaksuaikojen suhteen. Ymmärrän. Meillä on tarjolla erilaisia lainavaihtoehtoja, joiden korkotasot ja takaisinmaksuehdot vaihtelevat. Haluaisitko kuulla niistä tarkemmin? Kyllä, kiinnostaisi erityisesti mahdolliset kiinteäkorkoiset lainat tai joustavat vaihtoehdot, jos sellaisia on tarjolla. Meillä on valikoimissa sekä kiinteäkorkoisia lainoja että erilaisia joustavia vaihtoehtoja, kuten joustoluottoja tai vaihtelevakorkoisia lainoja. Voisin kertoa niistä tarkemmin ja miten ne soveltuisivat tarpeisiisi. Kuulostaa hyvältä, kiitos! Miten voitais edetä tästä? Voisimme aluksi keskustella tarkemmin tarpeistasi ja taloudellisesta tilanteestasi. Sen jälkeen voisin esitellä sinulle vaihtoehtoja, jotka voisivat olla sinulle sopivia. Selvä homma! Milloin voitais jatkaa tätä keskustelua? Olen täällä auttamassa sinua aina tarvittaessa. Voit ottaa yhteyttä minuun milloin tahansa, niin jatketaan lainavaihtoehtojen selvittämistä!

Kuvio 13. ChatGPT:n generoima keskustelu

ChatGPT generoi kuvion 13 keskustelusta kuviossa 14 olevat kaksi eri vastausta. Promptina vastauksille käytettiin kuvion 12 promptia. Vastauksia tarkasteltaessa huomataan ensimmäisen vastauksen myötäilevän hyvin annettua promptia, mutta toisen vastauksen virkemäärä ei ole noudattanut annettua ohjetta.

Asiakaspalvelija suoriutui keskustelusta hyvin. Hän oli aktiivinen ja kuunteli asiakasta huolellisesti. Kehitysideoita voisi olla vielä tarkemman tiedon tarjoaminen eri lainavaihtoehtoista ja niiden soveltuvuudesta asiakkaan tarpeisiin.

Asiakaspalvelija suoriutui hyvin keskustelusta, sillä hän oli aktiivinen ja tarjosi asiakkaalle erilaisia vaihtoehtoja. Kehitysideana voisi olla, että asiakaspalvelija olisi voinut kysyä asiakkaalta tarkemmin tämän tarpeista ja taloudellisesta tilanteesta jo keskustelun alussa, jotta olisi voinut tarjota vielä räätälöidympiä vaihtoehtoja.

Kuvio 14. Generoidut palautteet lainakeskustelusta

6.3 Generoidun vastauksen tulkitseminen

Generatiivisen tekoälyn tuottamat vastaukset saattavat olla melko ympäröireitä tai jopa virheellisiä. Tekoälyä hyödyntäessä tulee muistaa, ettei se tarkista vastausten todellisuutta tai lähteitä, se vain ennustaa sopivimpia sanoja. Vastausten oikeudellisuuden tulkitseminen jääkin aina loppukäyttäjän vastuulle. (Ojanperä 2023, 50.)

Generoitujen vastausten käyttämisessä joutuu olemaan hieman varovainen aiemmin mainittujen puutteiden takia. Turvallisempi vaihtoehto on jättää vastaukset aina ihmisille tulkittavaksi. Tämän takia projektissa generatiivinen tekoäly pidetään niin sanotusti apulaisen roolissa, eikä sitä käytetä suoraan statistiikan tuottamiseen.

7 POHDINTA

Työn tavoitteena oli valmistaa prototyyppi, joka jalostaa tallennettuja asiakasneuvotteluita hyödynnettävämpään muotoon. Tällöin mahdollistetaan lisäarvon tuottaminen tallenteista, joita tulee säilyttää ja pitää käytettävissä. Työn myötä pyrittiin myös kehittämään osaamista Microsoftin pilviympäristö Azuressa ja kokeilemaan generatiivisen tekoälyn mahdollisuuksia.

Työn tuloksena syntynyt prototyyppi onnistui mielestäni hyvin. Tuotettu palvelu on toimiva kokonaisuus alusta loppuun asti. Datantuottajapalvelu pystyy automaattisesti litteroimaan useita äänitiedostoja kerralla ja lähettämään ne erilliseen tietokantaan säilytettäväksi. Datankuluttajapalvelun kautta litteraatteja pystyttiin lukemaan helposti ja litteraattien sisällöstä pystyttiin tuottamaan tiivistelmiä ja palautetta.

Työssä käytetyt kehitystyökalut ja palvelut soveltuivat hyvin työhön ja edistivät projektin tekemistä suuresti. Erityisesti Visual Studio Coden eri laajennukset nopeuttivat web-sovelluksen kehitystä ja käyttöönottoa. MongoDB:n ja OpenAI:n ohjelmointirajapinnat olivat helppokäyttöisiä ja hyvin dokumentoituja, joka teki niiden integroinnista suoraviivaista.

Azuren käyttäminen oli pääasiassa sulavaa, mutta työn myötä ilmeni muutamia kompastuskiviä. Microsoftin dokumentaatio ei ole jokaisen Azuren palvelun osalta selkeää tai riittävää. Logic Appin kehittäminen oli hieman nihkeää, sillä palvelun suunnittelukäyttöliittymä oli mielestäni liian pelkistetty ja turhauttavan hidas. Muuten käyttökokemus oli sulava, eikä aiheuttanut viivästyksiä työssä.

Generatiivisen tekoälyn hyödyntäminen työssä oli erittäin mielenkiintoista. Tekoälyn käyttäminen ideoinnissa, kehitystyössä sekä palvelussa konkretisoi käsitykseni tekoälyn käyttömahdollisuuksista. Myös tekoälyä käytettäessä huomioitavat asiat selkenivät työn myötä.

Projektia jatkettaessa palveluun olisi tärkeä lisätä litteraattien oikeellisuuden ilmoittaminen tai mittaaminen, jotta virheelliset tiedot voidaan huomioida ja mahdollisesti korjata. Lisäksi Logic Appiin olisi hyvä lisätä litteraattien rinnakkainen lähetys tietokantaan. Toiminnallisuutta voitaisiin lisätä uusilla prompteilla. Esimerkiksi puheensävyn tai keskustelun tunnelman arviointi voisi olla hyödyllinen asiakasneuvotteluiden käsittelyyn. Promptien kanssa voisi olla erityisen luova, mutta niiden antamien tulosten hyöty yritykselle ja työntekijöille tulisi arvioida kriittisesti.

Kokonaisuutena työn toteuttaminen oli erittäin opettavainen minulle ja edisti suuresti tulevaisuudessa arvostettavaa osaamista. Vaikka valmiiksi palveluksi pääsemiseen tarvittaisiin vielä paljon testaamista ja kehitystä, eri oppeja ja osia työstä voidaan poimia jo nyt mukaan uusiin projekteihin.

LÄHTEET

Amazon Web Services 2023. What is Speech To Text? Viitattu 27.10.2023 <https://aws.amazon.com/what-is/speech-to-text>.

Finanssivalvonta 2020. Tietoturvallisuus. Viitattu 20.11.2023 <https://www.finanssivalvonta.fi/finanssisektorin-toimijalle/pankki/fintech--finanssialan-innovaatiot/tietoturvallisuus/>.

Finto 2018. Tiedon jalostaminen. Viitattu 17.10.2023 <https://finto.fi/tt/fi/page/t27>.

Green, D. 2016. Scrum: Novice to Ninja. Collingwood: SitePoint Pty, Ltd. Viitattu 31.10.2023 https://luc.finna.fi/lapinamk/Record/luc_electronic_amk.994843926306246?sid=3725493455.

IBM 2023. What is Java Spring Boot? Viitattu 3.11.2023 <https://www.ibm.com/topics/java-spring-boot>.

Kallio, A. 2023. Litterointi. Teoksessa J. Vuori (toim.) Laadullisen tutkimuksen verkkokäsikirja. Tampere: Yhteiskuntatieteellinen tietoarkisto. Viitattu 23.10.2023 <https://www.fsd.tuni.fi/fi/palvelut/menetelmaopetus/kvali/laadullisen-tutkimuksen-prosessi/litterointi/>.

Kangas, E. 2023. Digipalvelumme on markkinoiden paras, mutta se ei ole (vain) omaa ansiotamme. Viitattu 30.11.2023 <https://www.op-media.fi/puheenvuorot/digipalvelumme-on-markkinoiden-paras-mutta-se-ei-ole-vain-omaa-ansiotamme>.

Microsoft Learn 2023a. Azure Functions overview? Viitattu 26.11.2023 <https://learn.microsoft.com/en-us/azure/azure-functions/functions-overview>.

Microsoft Learn 2023b. Introduction to Azure Blob Storage. Viitattu 26.11.2023 <https://learn.microsoft.com/fi-FI/azure/storage/blobs/storage-blobs-introduction>.

Microsoft Learn 2023c. What is Azure Logic Apps? Viitattu 26.11.2023 <https://learn.microsoft.com/fi-FI/azure/logic-apps/logic-apps-overview>.

Microsoft Learn 2023d. What is speech to text? Viitattu 27.10.2023 <https://learn.microsoft.com/fi-fi/azure/ai-services/speech-service/speech-to-text>.

Microsoft Learn 2023e. What is the Speech service? Viitattu 26.11.2023 <https://learn.microsoft.com/en-us/azure/ai-services/speech-service/overview>.

Ojanperä, T. 2023. Tekoälyn vallankumous: käsikirja. Helsinki: Alma Talent Oy. Viitattu 5.12.2023 https://luc.finna.fi/lapinamk/Record/luc_electronic_amk.994856441706246?sid=3474719533.

Saarijärvi, H. 2017. Miten viestiä ja mitata asiakaslähtöisyyttä? Teoksessa E. Juholin & V. Luoma-aho (toim.) Mitattava viestintä. Helsinki: ProCom – Viestinnän ammattilaiset ry, 100–111. Viitattu 30.11.2023 <https://helda.helsinki.fi/server/api/core/bitstreams/8594fe47-4738-4703-840c-e62a8f10587b/content>.

Singh, K., Ianculescu, A. & Torje, L-P. 2018. Design Patterns and Best Practices in Java : A Comprehensive Guide to Building Smart and Reusable Code in Java. Birmingham: Packt Publishing Ltd. Viitattu 12.11.2023 https://luc.finna.fi/lapinamk/Record/luc_electronic_amk.994713638906246?sid=3685312487.

Sletteberg, E. 2022. frontend-maven-plugin. Viitattu 4.11.2023 <https://github.com/eirslett/frontend-maven-plugin>.

Software Engineering Institute 2023. Software Architecture. Viitattu 4.12.2023 <https://www.sei.cmu.edu/our-work/software-architecture>.

VMware 2023. Spring Initializr. Viitattu 3.11.2023. <https://start.spring.io/>.