



Vili Häkkinen

Tekoäly pilvipohjaisessa liiketoimintatadatan analysoinnissa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintäteknikka

Insinöörityö

24.5.2024

Tiivistelmä

Tekijä: Vili Häkkinen
Otsikko: Tekoäly pilvipohjaisessa liiketoimintadatan analysoinnissa
Sivumäärä: 37 sivua
Aika: 27.5.2024

Tutkinto: Insinööri (AMK)
Tutkinto-ohjelma: Tieto- ja viestintätekniikka
Ammatillinen pääaine: IoT ja Cloud computing
Ohjaajat: Osaamisaluepäällikkö Janne Salonen
Toimitusjohtaja Pekka Häkkinen

Tämän insinööriyön tavoitteena oli kehittää asiakasyritykselle tekoälyä hyödyntävä keskustelubotti, joka kykenee vastaamaan käyttäjän luonnollisella kielellä esittämiin kysymyksiin. Ratkaisun liiketoiminnallisena käyttökohteena oli asiakasyrityksen materiaalivaraston riittävyden tunnistaminen, joka lasketaan toimittajien historiallisen toimitusvarmuuden ja materiaalien päiväkohtaisen kulutuksen pohjalta. Työn testauksessa käytettävä lähdedata on asiakasyrityksen toimittamaa ostojärjestelmästä tuotua raakaa dataa, joka puhdistettiin ja formatoitiin SQL-relaatiotietokantamallille ja applikaatiossa käytettävälle tekoälyagentille sopivaksi. Insinööriyö pohjautuu lukuihin yksittäisiin verkosta löytyviin kehittäjäkokemuksiin, teknisiin dokumentaatioihin ja muihin epäsäännöllisiin verkkoaineistoihin.

Toisena työn tavoitteena oli tutkia tekoälyä käyttävien teknologioiden integroitavuutta yrityskentässä laajalti käytössä oleviin pilvialustoihin. Insinööriyössä kuvattu ratkaisu rakennettiin Microsoft Azure -ympäristöön, joka oli sekä työn toimeksiantajayrityksen sekä asiakasyritykseksi valikoituneen yhtiön käytössä oleva pilvialusta. Kirjoitelmassa kuvataan Azureen pystytettyä kokoonpanoa, sen sisältämiä palveluita ja niiden konfiguraatiota. Myös ohjelmistokehitys oli insinööriyössä merkittävä osa, sillä luotu ohjelmistokokonaisuus oli saatava toimimaan interaktiivisen käyttöliittymän kautta niin, että asiakasyrityksen työntekijät kykenevät käyttämään sitä itsenäisesti.

Työn lopuksi arvioidaan rakennetulla ratkaisulla saatujen tulosten laatua, käyttökel-
poisuutta yrityksen data-analytiikka-työkaluna ja potentiaalisia jatkokehitysmahdolli-
suuksia. Ratkaisussa käytettävät teknologiat kehittyvät jatkuvasti, ja insinööriyöhön
implementoidut ratkaisut vanhenevat täten nopeasti.

Avainsanat: tekoäly, BI, Microsoft Azure, suuret kielimallit, SQL, data-analytiikka

Abstract

Author: Vili Häkkinen
Title: Artificial Intelligence as a tool in cloud-based business analytics
Number of Pages: 37 pages
Date: 27 May 2024

Degree: Bachelor of Engineering
Degree Programme: Information and Communication Technology
Professional Major: IoT and Cloud computing
Supervisors: Janne Salonen, Head of School (ICT)
Pekka Häkkinen, CEO

The objective of this thesis was to develop a chatbot that utilizes artificial intelligence and has a capability to process user's questions represented in natural language. The business-related use case for the implemented solution was to identify the sufficiency of the client company's material inventory, which is calculated based on the historical delivery reliability of suppliers and the average daily consumption of materials. The source data used in the testing of the solution was raw data pulled from the client's purchasing system, which was then cleaned and formatted to match the requirements of SQL database model and to enhance AI agent's ability to interpret it. The thesis is based on numerous individual developer experiences, technical documentations, and other irregular online sources.

The secondary objective was to investigate the integrability of AI technologies with widely used cloud platforms in the business environment. The solution described in the thesis was built in Microsoft Azure environment, which was already in use both the company commissioning the thesis work and the company that provided the use case. The document describes the setup established in Azure, the services it includes, and overview of their respective configurations. Software development was also a significant part of the thesis, as the created software solution had to be accessible through an interface in a way that the client company's employees could use it independently.

The conclusion section evaluates the quality of the obtained results with the built solution, its usability as a data analytics tool for business purposes, and potential further development opportunities. The technologies used in the solution are evolving rapidly, and the technical solutions implemented in the thesis will thus become outdated quickly.

Keywords: Artificial Intelligence, Business Intelligence, Microsoft Azure, Large Language Models, SQL, Data-analytics

Sisällys

1	Johdanto	1
2	Pilviympäristön rakenne	3
2.1	Azure-komponentit	3
2.1.1	Azure OpenAI service	3
2.1.2	SQL Server	5
2.1.3	SQL Database	5
2.1.4	Container registry	6
2.1.5	Azure Web App	7
2.1.6	Azure Web App Plan	7
3	Ohjelmistot ja ohjelmistokoodi	9
3.1	Visual Studio Code	9
3.1.1	Langchain	10
3.1.2	Anaconda	10
3.2	Ohjelmakoodi	11
3.3	Docker	17
3.4	Azure Data Studio	18
3.5	Git	19
3.6	KNIME	19
4	Käyttötapauksen kuvaus	20
4.1	Käyttötapauksen data ja sen valmistelu	21
4.1.1	Datan organisointi	21
4.1.2	Datan puhdistus ja formatointi	22
4.1.3	Datan sisällön rikastaminen	22
4.2	Datan visualisointi	23
4.3	Kehotteiden kehittäminen ja tulosten tarkastelu	25
4.3.1	Kehote – Vastaus, esimerkki 1	25
4.3.2	Kehote – Vastaus, esimerkki 2	29
5	Yhteenveto	31
5.1	Insinööriyössä kehitetyn ratkaisun ongelmat	31
5.1.1	Tiedon hankinta	31
5.1.2	Lähdedatan laatuvaatimukset	31

5.1.3	Tekoälyn vastausten validointi	32
5.1.4	Token-raja	32
5.2	Jatkokehitys	33
5.2.1	Tekoälyn käyttötapauksen laajentaminen	33
5.2.2	Integrointi osaksi toiminnanohjausjärjestelmää	33
5.2.3	Kielimallin hienosäätäminen	33
5.2.4	Uusien kielimallien käyttöönotto	34
5.3	Tilaajan lausunto	35
	Lähteet	36

Lyhenteet

- AI: *Artificial Intelligence*. Tekoäly. Keinotekoista älykkyyttä, joka antaa tietokoneohjelmille kyvyn ratkaista esimerkiksi laskentaan ja tietojen yhdistelyyn liittyviä ongelmia sen toimintaa ohjaavien algoritmien avulla.
- LLM: *Large language model*. Suuret kielimallit ovat syviin neuroverkkoihin perustuvia suurella datamäärällä koulutettuja ja koneopetettuja kieli-malleja, jotka kykenevät esimerkiksi tulkitsemaan ja generoimaan luonnollista kieltä.
- SDK: *Software development kit*. Ohjelmistokehityspaketti. Se sisältää tarvittavan joukon työkaluja ja kirjastoja ohjelmistokehittäjien avuksi.
- SQL: *Structured query language*. Strukturoidulla kyselykielellä voidaan tehdä yhteensopivaan relaatiotietokantaan hakuja ja muutoksia.
- API: *Application Programming Interface*. Ohjelmointirajapinta mahdollistaa eri ohjelmien välisen keskinäisen kommunikoinnin. Se voi tarkoittaa joko sovelluksen sisäistä rajapintaa tai muille sovelluksille tarkoitettua ulkoista rajapintaa.
- GPT: *Generative pre-trained transformer*. Viittaa OpenAI:n kehittämään esikoulutettuun kielimallien perheeseen, jonka mallit on koulutettu suurilla tekstikorpuksilla. Mallit opetetaan ennustamaan tekstikatkelman seuraavan sanan, tuoden niille kyvyn generoida luonnollista kieltä.
- CSV: *Comma Separated Values*. Pilkuilla erotellut arvot. Tiedostomuoto, jossa taulukkomuotoinen data on tallennettu tekstitiedostoksi käyttäen pilkkua ja rivinvaihtoa kenttien erottimina.

- BI: *Business Intelligence*. Työkaluja ja ominaisuuksia sisältävä ohjelmisto, jota käytetään liiketoimintadatan analysointiin.
- PaaS: *Platform as a service*. Alusta palveluna. Palvelun käyttäjä on vastuussa esimerkiksi tietokantaan syötetystä datasta, mutta palveluntarjoaja on vastuussa tietokannan infrastruktuurin ylläpidosta.
- IaaS: *Infrastructure as a service*. Infrastruktuuri palveluna. Palvelun käyttäjä on vastuussa verkkojen, virtuaalikoneiden ja sovellusten konfiguroinnista ja hallinnasta, mutta laitteisto on palvelutoimittajan vastuulla.
- SLA: *Service Level Agreement*. Palvelutasosopimus. SLA:ssa palveluntarjoaja määrittää tarjoamiensa palvelujen sisällön, kuten laatuluokituksen palvelun saatavuudessa ja ylläpidossa.

1 Johdanto

Tekoälyn suosio on noussut jatkuvasti 2020-luvun alusta saakka. Erityisesti vuonna 2022 OpenAI:n lanseeraama ChatGPT nosti tekoälyyn pohjautuvat sovellukset suuren yleisön tietoisuuteen, minkä seurauksena markkinoille on ilmestynyt lukuisia sekä kuluttajille että yrityksille suunnattuja suuriin kielimalleihin pohjautuvia ratkaisuja. Data-analytiikka on liiketoiminnan prosessien jatkuvan tehostamisen osalta merkittävässä asemassa, ja tekoäly tarjoaa mielenkiintoisia mahdollisuuksia dataan perustuvan päätöksenteon avuksi.

Insinööriyön tavoitteena oli luoda Microsoft Azure OpenAI:n kielimalleja hyödyntävä yrityskäyttöön soveltuva Microsoft Azure -ympäristössä toimiva BI-työkalu, joka toimii keskustelubotin tapaan vastaten luonnollisella kielellä esitettyihin kysymyksiin ja tuottaen dataa SQL-tietokannassa sijaitsevan liiketoimintadatan pohjalta. Työn kahdessa ensimmäisessä luvussa käsitellään järjestelmäarkkitehtuurin kokoonpanoa ja kuvataan mitä palveluita, ohjelmistoja ja teknologioita ratkaisussa käytetään, sekä luodun sovelluksen koodia ja rakennetta. Sen jälkeen määritetään käyttötapaustutkimuksen liiketoiminnallinen aihe, ja käsitellään asiakasyrityksen luovuttamaa dataa relaatiotietokantaan sopivaksi ja varsinaiseen käyttötarkoitukseen soveltuvaksi. Viimeisenä tutkitaan sovelluksen kehityksessä sekä testauksessa ilmenneitä ongelmia, sekä mahdollisia kehittämismahdollisuuksia järjestelmän toiminnan parantamiseksi.

Projektin toimeksiantaja on liikkeenjohdon konsultointiin erikoistunut yritys Cuando Consulting Oy. Yritys on kiinnostunut tekoälyä sisältävien ratkaisujen käyttöönotosta toimitusketjujen prosesseissa syntyvän datan tulkitsemiseksi. Projektin yhteydessä tarkastellaan, tarjoaako kehitetty työkalu lisäarvoa jo olemassa olevien BI-työkalujen rinnalla, ja onko pienillä henkilöstöresursseilla mahdollista kehittää riittävän hyödyllisiä tuloksia antavaa järjestelmää. Projektin aloitushetkellä oleellista oli myös se, ettei markkinoilla ollut varteenotettavia julkisesti saatavilla olevia ja yrityskäyttöön sopivia ratkaisuja, joilla tekoäly saataisiin valjastettua tietokantamuotoisen strukturoidun datan analysointiin.

Merkittävä asia insinööriyön tekemisessä oli aihepiirin tuoreudesta johtuva perinteisten lähteiden olemattomuus. Muun muassa tutkimusyhtiö Gartner 29.2.2024 järjestämässään webinaarissa aiheesta ” What Supply Chain Leaders Are Planning for GenAI Solutions in 2024” raportoi, että tähänastiset käytötapaukset ovat liittyneet hallinnollisiin prosesseihin, käyttäjätukeen sekä ohjelmistokehitykseen, ja että toimitusketjun hallinnan ison datan (big data) hyödyntäminen on vasta alkuvaiheessaan. Projektin kehitysvaiheessa tietoa etsittiin ja poimittiin muun muassa lukuisista muiden kehittäjien blogikirjoituksista, yksittäisistä keskustelufoorumien kommenteista, sekä teknologiayhtiöiden ja ohjelmistokehittäjien dokumentaatioista. Ratkaisun varsinainen käyttötarkoitus ja sen järjestelmärakenne oli myös erittäin rajattu ja spesifi, minkä takia edellä mainituista lähteistä poimittuja tietoja ei usein kyetty suoraan implementoimaan omaan ratkaisuun, vaan niitä piti soveltaa haluttuun lopputulokseen pääsemiseksi.

2 Pilviympäristön rakenne

Tämä kappale kuvaa pystytetyn verkkoympäristön konfiguraatiota ja palvelusäilyä. Projektin suunnitteluvaiheen merkittävin osa oli sopivan pilvipalvelualan valinta. Yrityskäytössä merkittävimmät pilvipalveluita tarjoavat yritykset ovat Microsoft, Amazon ja Google. Sekä työn toimeksiantajayrityksen että tämän asiakasyritysten keskuudessa Microsoft Azure osoittautui kuitenkin selvästi vahvimmin käytössä olevaksi alustaksi, jonka seurauksena Azure valikoitui projektin pilviympäristöksi jo varhaisessa vaiheessa. Microsoftin yhteistyö OpenAI:n kanssa tarjoaa myös merkittäviä synergiaetuja työkalun taustalla käytettävien palveluiden integroinnin sekä ohjelmistokehityksen näkökulmasta.

2.1 Azure-komponentit

Pilviympäristön toiminallisuuden muodostavat Azureen pystytetyt palvelu- ja resurssi-instanssit. Instanssien hallinnoinnin ja monitoroinnin helpottamiseksi kaikki vaadittavat palvelut ovat käytännöllisintä perustaa saman resurssiryhmän (resource group) alle.

Pilvikonfiguraatioon kuuluu seuraavat resurssit:

- Azure OpenAI Service
- SQL Server
- SQL Database
- Container registry
- Web App
- Web App Plan

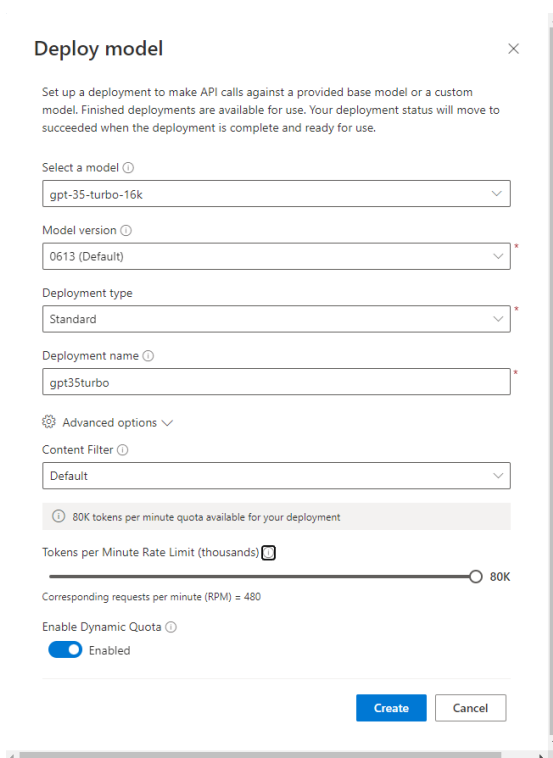
2.1.1 Azure OpenAI service

Azure OpenAI service on projektin kannalta merkittävin palvelu. Sen kautta on mahdollista ottaa käyttöön OpenAI:n yleisesti saatavilla olevat suuret kielimallit. Palvelun kautta saa pääsyn REST-sovellusliittymään ja se tarjoaa Python SDK:n sovelluskehityksen tueksi, joka oli työn ohjelmistokehityksen osalta

tärkeässä roolissa. Itse palveluinstanssin luominen on yksinkertaista. Käyttäjää vaaditaan valitsemaan resurssiryhmän lisäksi instanssin nimi, sen hintataso ja maantieteellinen sijainti, jonka vaihtoehdot määräytyvät Azuren datakeskusten mukaan. Sijaintia valitessa on oleellista tarkistaa, että kaikki tarvittavat LLM:t ja niiden versiot ovat saatavilla kyseisessä sijainnissa.

Käyttöoikeus Azure OpenAI service -palveluun on rajattu, ja vain hyväksytyt yrittäjäasiakkaat ja yhteistyökumppanit voivat saada oikeuden palveluun verkossa täytetyn hakemuksen kautta. Hakijalta vaaditaan yrityksen yhteystietojen ja Azure Subscription ID:n lisäksi täsmennystä palvelun keskinäisistä käyttökohteista, ja hyväksyntää Microsoftin vastuullisen tekoälyn ehtoihin.

Kielimallit otetaan käyttöön Azure OpenAI Studio:ssa (kuva 1), joka tarjoaa verkkopohjaisen käyttöliittymän mallien käyttöönottoon, konfigurointiin, versiohallintaan ja kielimallien sisällön moderointiin (content filter). Projektissa käytettävät mallit ovat OpenAI:n perusmalleja "GPT-3.5 Turbo" ja "GPT-4 Turbo". (1.)



The screenshot shows the 'Deploy model' dialog box in Azure OpenAI Studio. It contains the following fields and options:

- Select a model:** A dropdown menu with 'gpt-35-turbo-16k' selected.
- Model version:** A dropdown menu with '0613 (Default)' selected.
- Deployment type:** A dropdown menu with 'Standard' selected.
- Deployment name:** A text input field containing 'gpt35turbo'.
- Advanced options:** A section with a gear icon and a dropdown arrow, containing:
 - Content Filter:** A dropdown menu with 'Default' selected.
 - 80K tokens per minute quota available for your deployment:** A greyed-out field.
 - Tokens per Minute Rate Limit (thousands):** A slider control set to 80K.
 - Corresponding requests per minute (RPM) = 480:** A text label below the slider.
 - Enable Dynamic Quota:** A toggle switch that is currently turned on (labeled 'Enabled').
- Buttons:** 'Create' and 'Cancel' buttons at the bottom right.

Kuva 1. Kielimallin käyttöönotto Azure OpenAI Studion verkkokäyttöliittymässä.

2.1.2 SQL Server

Azure SQL Server on tuoteperhe, joka tarjoaa SQL-palvelimen tietokantamootorin Azure-pilvipalvelussa. Käyttäjän on mahdollista valita sopiva palvelumalli PaaS ja IaaS -vaihtoehtojen välillä. Projektissa käytetään PaaS -mallia sen helppokäyttöisyyden vuoksi, jolloin SQL-palvelimen ylläpidosta vastaa Azure. Käyttäjä on vastuussa vain palvelimella pyörivän SQL-tietokannan käyttäjähallinnasta ja verkkomäärittelyistä, sekä sinne syötettävästä datasta. IaaS -palvelumallissa on mahdollista ajaa SQL-palvelinta virtuaalikoneilla, jolloin palvelin voidaan konfiguroida tarkasti vastaamaan yrityksen suorituskyky- ja saatavuusvaatimuksiin.

Palvelun pystytys on suoraviivaista, eikä vaadi palvelimen nimen ja sijainnin lisäksi kuin käytettävän autentikointitavan valinnan Entra ID:n ja SQL-tunnistuksen välillä. Asiakasyrityksen tietoturvamääritysten vuoksi SQL-tunnistus osoitautui toimivammaksi metodiksi, sillä se on helppo integroida osaksi projektissa kehitettyyn ohjelmaan (2.).

2.1.3 SQL Database

Azure SQL Database on täysin Azuren ylläpitämä PaaS-mallin mukainen tietokantainstanssi, jolle käyttäjä voi luoda SQL-tietokantoja pystyttämälleen SQL-palvelimelle. Projektissa käytetty tietoaineisto on kokoluokaltaan rajattu varsin suppeaksi, joten Azuren tarjoamista hinnoittelutasoista edullisin viimeisimmän arkkitehtuurisukupolven vaihtoehto (kuva 2) oli sekä muistin että suorituskyvyn osalta riittävä, joten "Standard-series (Gen5)" päätyi täten projektin tietokannan laitteistovalinnaksi, laskentatason ollessa "Serverless". "Palvelimeton" tietokanta on kustannustehokas valinta, sillä se skaalautuu automaattisesti laskenta- ja muistitarpeen mukaan. Laskentatehon ja muistimäärän kohdalla tietokanta rajoitettiin käyttämään enimmillään vain yhtä gigatavua muistia ja yhtä virtuaalista ydintä (vCore).

Tietokannan konfiguroinnissa priorisoitiin kustannusten minimointia. Palvelutasoksi määrittyi ”General Purpose”, jolloin se noudattaa Microsoftin saman nimistä palvelutasosopimusta. Siinä määritellään palvelun saatavuus vuositasona, joka on viimeisimmän Microsoftin SLA:n mukaan SQL-palvelimille suurempi kuin 99.9 %. (3, s. 81) Verkkomäärittelyjen osalta tietokannan palomuriin lisättiin sääntö, joka salli kirjautumisen tietokantaan vain määritellyistä IP-osoitteista tietoturvasyistä. Konfiguraatiossa data on paikallisesti redundanttia, eli Microsoftin käytäntöjen mukaan käytössä oleva muistitila kopioidaan varoimenpiteenä saman fyysisen datakeskuksen sisällä kolmelle erilliselle muistiasemalle. (4.)

SQL hardware configuration
SQL database

Available hardware configurations

Based on your workload requirements, select from available hardware configurations listed below.

Configuration	Description	Max vCores
Gen4	Balanced memory and compute	--
Standard-series (Gen5)	Balanced memory and compute	80
Premium-series	Balanced memory and compute	--
Premium-series - memory optimi...	Memory optimized	--
M-Series	Memory optimized	--
FSv2-series	Compute optimized	--
DC-series	Enables confidential computing	--

Kuva 2. SQL-tietokannan laitteistokonfiguraation tyypivaihtoehdot.

2.1.4 Container registry

Azure Container Registry on Azuren tarjoama hallinnoitu Docker -rekisteripalvelu. Se perustuu Docker Registry 2.0 -versioon ja mahdollistaa sen käyttäjille konttikuvien (Docker image) rakentamisen, tallentamisen ja hallinnan. Azuren konttirekisteriin on suoraviivaista työntää luodut konttikuvat suoraan terminaaliosovelluksesta, ja säilössä olevat konttikuvat voidaan ottaa suoraan käyttöön

Azure Web App- resurssissa. Palvelua ei tarvitse hinnoittelutason sekä nimeämisen jälkeen toiminnallisuuden osalta konfiguroida, kunhan "Admin user" -asetus on laitettu päälle. Tällöin terminaalisovelluksesta voidaan kirjautua suoraan konttirekisteriin, jolloin uusien konttikuvien puskeminen rekisteriin on mahdollista. (5.)

2.1.5 Azure Web App

Azure Web App on palvelu, jota käytetään projektissa konttikuvien käyttöönotossa. Se mahdollistaa verkkosovellusten tehokkaan rakentamisen, käyttöönoton ja skaalauksen, sekä on helposti integroitavissa muiden Azure -palveluiden kanssa. Web App -palvelua käytetään projektissa Docker-kontin julkaisemiseen, jolloin sovelluksen verkkokäyttöliittymä saadaan asiakasyrityksen Azure-tenantin alle, mahdollistaen tietoturvastandardien mukaisen sisäänpääsyn määritellylle organisaation Entra ID:llä tunnistautuvalle joukolle. Web App -palvelu toimii projektissa myös osana versionhallintaa, "Deployment Center" -alavalikon alta on nopea vaihtaa aktiivista konttia toiseen konttirekisteriin puskettuun konttiversioon. Palvelun konfiguroinnissa oleellista oli oikean käyttöjärjestelmän valitseminen, joka projektin tapauksessa oli Linux. Kontin julkaisemisen jälkeen palvelu tarjoaa käyttäjälle automaattisesti verkkotunnuksen, jonka kautta kontitetun sovelluksen käyttöliittymän saa verkkoselaimessa auki. Hinnoittelumalli Web App -palvelulle asetetaan Web App Plan -palvelussa. (6.)

2.1.6 Azure Web App Plan

Azure Web App Plan on palvelu, joka antaa mahdollisuuden määritellä ja hallita Web App -palvelun taustalla pyöriviä resursseja ja infrastruktuuria. Palvelun kautta valitaan hinnoittelutaso (Kuva 3), ja skaalataan laitteistoresursseja vastaamaan kysyntää. Projektissa hinnoittelutasoksi määräytyi "Basic B1", sillä se kuuluu Microsoftin SLA:n piiriin, sekä tarjoaa riittävän määrän suorituskykyä pienelle joukolle yhtäaikaisia käyttäjiä. (7.)

Select App Service Pricing Plan

Hardware view Feature view Showing 14 App Service pricing plans

Name	ACU/vCPU T	vCPU	Memory (GB)	Remote Storage (GB)	Scale (instance)	SLA	Cost per hour (instance)	Cost per month (instance)
Dev/Test (For less demanding workloads)								
Free F1	60 minutes/day...	N/A	1	1	N/A	N/A	Free	Free
<input checked="" type="checkbox"/> Basic B1	100	1	1.75	10	3	99.95%	0.018 USD	13.14 USD
Basic B2	100	2	3.5	10	3	99.95%	0.036 USD	26.28 USD
Basic B3	100	4	7	10	3	99.95%	0.071 USD	51.83 USD
Production (For most production workloads)								
Premium v3 P0V3	195*	1	4	250	30	99.95%	0.116 USD	84.68 USD
Premium v3 P1V3	195	2	8	250	30	99.95%	0.178 USD	129.94 USD
Premium v3 P2V3	195	4	16	250	30	99.95%	0.356 USD	259.88 USD
Premium v3 P3V3	195	8	32	250	30	99.95%	0.712 USD	519.76 USD
Legacy								
Standard S1	100	1	1.75	50	10	99.95%	0.095 USD	69.35 USD
Standard S2	100	2	3.5	50	10	99.95%	0.19 USD	138.70 USD
Standard S3	100	4	7	50	10	99.95%	0.38 USD	277.40 USD
Premium v2 P1V2	210	1	3.5	250	30	99.95%	0.115 USD	83.95 USD
Premium v2 P2V2	210	2	7	250	30	99.95%	0.231 USD	168.63 USD
Premium v2 P3V2	210	4	14	250	30	99.95%	0.462 USD	337.26 USD

Kuva 3. Azuren tarjoamat Web App Plan -hinnoittelutasot vaihtelevat laitteisto-
kokoonpanon mukaan.

3 Ohjelmistot ja ohjelmistokoodi

Tässä kappaleessa kuvataan insinööriyönä luodussa ratkaisussa käytettyjä ohjelmistoja, varsinaisen sovelluksen kehityksessä käytettyjä työkaluja sekä kehitetyn sovelluksen koodia, joka perustuu kirjoitushetkellä käytössä olevaan sovellusversioon. Käytetyt ohjelmistot valikoituivat projektiin niiden helppokäyttöisyyden ja tunnettavuuden perusteella, ja niiden yhteensopivuus keskenään sekä Azure -palveluiden kanssa olivat työn kannalta tärkeässä asemassa. Ohjelmointikieleksi valikoitui Python, sillä Azure OpenAI -palvelu tarjoaa Pythonia käyttävän SDK:n, sekä ratkaisussa käytetty Langchain -tekoälyagentti on Python -tuettu.

3.1 Visual Studio Code

Visual Studio Code on Microsoftin kehittämä ohjelmistokehittäjille suunnattu avoimen lähdekoodin tekstieditori. Insinööriyön ohjelmistokehitys tehtiin täysin Visual Studio Code:ssa, sillä se tarjosi tavanomaisen toiminnallisuuden lisäksi kattavan valikoiman projektin kannalta hyödyllisiä laajennuksia, kuten tärkeimpänä:

- Docker
- Python
- Jupyter

Ohjelman alkuvaiheen kehitys ja rakentaminen tapahtui Jupyter Notebook:n avulla. Se mahdollisti koodin osien suorittamisen solujen muodossa. Näin koko ohjelmaa ei tarvinnut suorittaa kokonaisuudessaan kehitysvaiheessa, kun eri ohjelmistokomponenttien yhteensopivuutta kokeiltiin. Esimerkiksi Esimerkkikoodi 7 pystyttiin suorittamaan omana solunaan kehoitteita testatessa ennen kuin Streamlit -verkkokäyttöliittymä oli pystytetty.

3.1.1 Langchain

Projektin ohjelmistoratkaisussa Langchain -kehys oli merkittävässä osassa, sillä se sisälsi käyttötarkoituksen vaatimat työkalut ja oli suoraan integroitavissa Azure OpenAI -kielimallien kanssa. Opinnäytetyössä käytettävä SQL-agentti yhdistää luonnollisen kielen prosessoinnin (Natural language processing) SQL-tietokantoihin, antamalla käyttäjille mahdollisuuden tehdä tietokantakyselyitä ilman SQL-osaamista (8.). Alla on Langchain -agentin kolme tärkeintä komponenttia:

- Mallikäärijä (LLM wrapper): sen avulla agentti integroituu kielimalliin
- Kehotehallinta (prompt template): kehotemallit varmistavat, että käyttäjän muodostamat syötteet muunnetaan kielimallille tehokkaasti käsiteltävään muotoon
- Työkalut (toolkit): ohjelmassa käytetty SQL-agentti hyödyntää SQL-tietokantatyökaluja, jotka auttavat kyselyiden muodostuksessa ja suorittamisessa

3.1.2 Anaconda

Insinööriyössä Anaconda toimi ohjelmistossa käytettävänä virtuaaliympäristönä, jolla saavutettiin staattiset kehitysolosuhteet ohjelmistokehityksen tueksi. Sen ympäristönhallintamahdollisuuksien avulla voitiin projektille luoda itsenäinen kehitysympäristö kaikilla tarvittavilla kirjastoilla ja riippuvuuksilla, mikä minimoi mahdolliset yhteensopivuusongelmat helpottaen samalla kehitystyötä, ennen kuin sovelluksen toiminnallisuus oli riittävän pitkällä Docker -ympäristössä ajettavaksi. "Conda" -pakettihallinnan avulla oli suoraviivaista asentaa ja päivittää ohjelmassa käytettäviä kirjastoja ja paketteja. (9.)

3.2 Ohjelmakoodi

Python -ohjelman alussa (Esimerkkikoodi 1) olevilla lausekkeilla tuodaan (import) sovelluksessa vaaditut kirjastot ja moduulit, joiden avulla se integroituu SQL-tietokannan ja Azure OpenAI -palvelun kanssa. Koodikatkelma alustaa ympäristön lataamalla ympäristömuuttujat ".env" -tiedostosta ja tuo työkalut tietokantayhteyteen, Langchain -agentille, kielimallin kehoitteelle ja web-käyttöliittymää varten.

```
import os
from sqlalchemy import create_engine
from langchain_core.prompts import ChatPromptTemplate
from langchain_openai import AzureChatOpenAI
from langchain.agents import create_sql_agent
from langchain.sql_database import SQLDatabase
from langchain.agents.agent_toolkits.sql.toolkit import SQLDatabaseToolKit
from langchain_community.callbacks import StreamlitCallbackHandler
import streamlit as st
from dotenv import load_dotenv
```

Esimerkkikoodi 1. Ohjelmassa käytettävät moduulit tuodaan erilaisilla import -lausekkeilla

Seuraavassa osassa ohjelmistokoodia määritetään kehotemalli (prompt template), joka ohjaa tekoälyä käyttävää agenttia vastaamaan käyttäjän kysymyksiin kuvaamalla käyttökohteen pohjalta rakennettua tietokantaa (Esimerkkikoodi 2). Se toimii kontekstina SQL-kyselyiden luomiseen, ja asettaa käyttäjän kysymyksen muotoilupohjan sekä ohjeita tekoälyn toimintalogiikan ohjaamiseksi. Projektin aikana huomattiin, että kontekstia tarkentamalla kielimallilta saatiin tarkempia vastauksia. Tästä syystä myös SQL-tietokannan taulujen rakenne ja sisältö oli järkevää kirjoittaa kehotemallin kontekstiin.

```

prompt = ChatPromptTemplate.from_messages(
    [
        ("system",
         """
         You are a very capable AI assistant and SQL expert that
         answer to the best of your abilities about user's ques-
         tions about Materials, Stock, Suppliers and Purchase Or-
         ders.

         Use following context to create the SQL query:
         "Material table contains information about Materials in-
         cluding Location where material is used, Material Id,
         LifeCycle and respective Manager managing purchase or-
         ders for his/her materials.
         Material Stock table contains the Stock quantity as of
         StockDate and Stock Coverage in days for each Material
         at a Location.
         Stock Coverage in Days is instrumental data in determin-
         ing if current stock and incoming Deliveries are suffi-
         cient to satisfy demand and avoid stockouts.
         PurchaseOrders table contains details of incoming deliv-
         eries of Materials from respective Suppliers to stock.
         Supplier table contains both the name of each Supplier
         and its historical performance.
         AverageDelayInDays is a critical column in determination
         if there is a risk for stockout due to delay in delivery
         date of a purchase order from the Supplier.

         Under no circumstances should you attempt to call func-
         tions / tools that are not available to you.

         """,
        ),
        ("user", "{question}\n ai: "),
    ]
)

```

Esimerkkikoodi 2. Kielimallille kehote -mallissa annetaan konteksti ohjaamaan sen toimintaa.

Kontekstin kuvaamisen ja kehotteen mallin määrittämisen jälkeen määritellään ODBC-ajuri (Open Database Connectivity), jonka avulla ohjelma muodostaa yhteyden Azure -ympäristöön pystytettyyn SQL-tietokantaan (Esimerkkikoodi 3). Siinä asetetaan ympäristöön ODBC-ajurin asennettua versiota vastaava nimi ja muodostetaan yhteyslauseke (connection string) parametreineen tietokantayhteyden muodostamiseksi käyttäen ".env" -tiedostossa sijaitsevia ympäristömuuttujia, jotka tuodaan ohjelmaan aiemmin mainitulla "load_dotenv()" -funktiolla.

```

driver = '{ODBC Driver 18 for SQL Server}'
connection_str = (
    f"mssql+pyodbc:///?odbc_connect=Driver={driver};"
    f"Server=tcp:{os.getenv('SQL_SERVER')}.database.win-
dows.net;PORT=1433;"
    f"DATABASE={os.getenv('SQL_DB')});"
    f"Uid={os.getenv('SQL_USERNAME')});"
    f"Pwd={os.getenv('SQL_PWD')});"
    f"Encrypt=yes;TrustServerCertificate=no;Connection Timeout=30;"
)

```

Esimerkkikoodi 3. ODBC-ajurin yhteyslausekkeen parametrit.

Seuraavaksi koodissa luodaan "AzureChatOpenAI" -objekti tallentamalla se "llm"-muuttuja (Esimerkkikoodi 4). Objekti sisältää parametrejä, joilla saadaan yhteys haluttuun Azure OpenAI -palvelussa käyttöön otettuun kielimalliin ja määrittellään sen asetukset:

- "deployment name" -arvo luetaan ympäristömuuttujista, ja se vastaa Azure OpenAI -palvelussa käyttöön otetulle kielimallille annettua nimeä.
- "temperature" -arvolla säädetään kielimallin luovuutta, joka on koodissa säädetty mahdollisimman matalaksi vastausten satunnaisuuden minimoimiseksi ja vastausten ennustettavuuden parantamiseksi.
- "openai_api_key" on ympäristömuuttujiin asetettu arvo, joka on käyttäjän Azure OpenAI -palvelun henkilökohtainen API-avain.
- "verbose" -asetuksen avulla ohjelma tulostaa kielimallin toimintavaiheet, joka helpottaa sen antamien vastausten ymmärtämistä ja virheenkorjausta.
- "streaming" -parametri mahdollistaa kielimallin vastauksen tulostamisen merkki kerrallaan, joka parantaa verkkokäyttöliittymän käyttäjäkokemusta.

```

llm = AzureChatOpenAI (
    deployment_name=os.getenv("OPENAI_CHAT_MODEL"),
    temperature=0,
    openai_api_key=os.getenv("AZURE_OPENAI_API_KEY"),
    verbose=True,
    streaming=True
)

```

Esimerkkikoodi 4. AzureChatOpenAI -objektin luonti ja parametrien määrittely

Ohjelmassa luodaan seuraavaksi tietokantamoottori ja määritellään käytössä olevat SQL-työkalut (Esimerkkikoodi 5):

- "db_engine" -muuttujaan tallennetaan tietokantamoottori "create_engine" -komennolla käyttäen Esimerkkikoodi 3:ssa määriteltyä yhteyslauseketta.
- "db" -muuttujaan tallennetaan "SQLDatabase" -objekti, joka luodaan käyttämällä edellä mainittua tietokantamoottoria.
- SQL-työkalut määritellään luomalla "sql_toolkit" -muuttuja, johon tallennetaan "SQLDatabaseToolkit" -objekti. SQL-työkalut ovat ohjelman kannalta kriittisessä roolissa tietokantakyselyiden muodostuksessa.

```
db_engine = create_engine(connection_str)
db = SQLDatabase(db_engine)
sql_toolkit = SQLDatabaseToolkit(db=db, llm=llm)
```

Esimerkkikoodi 5. Tietokantamoottorin luonti ja työkalujen määrittely.

Seuraava koodin osa luo ja määrittää Langchain -kirjastosta tuodun SQL-agentin (Esimerkkikoodi 6), joka käyttää suurta kielimallia ja SQL-työkaluja tietokantakyselyjen suorittamiseen ja käsittelyyn. Agentin tehtävä on toimia kielimallin kanssa rinnakkain, minkä takia ohjelma kykenee muodostamaan luonnollisella kielellä annetusta kehoitteesta (prompt) SQL-kyselyitä käyttäjän kuvaaman kysymyksen ratkaisemiseksi.

- "agent_executor" -muuttujaan tallennetaan uusi agentti "create_sql_agent" -funktiolla
- "llm=llm" -lauseke asettaa agentille käytettävän kielimallin
- "toolkit=sql_toolkit" -lauseke asettaa agentin käyttämään SQL-työkaluja
- "agent_type" -määrittää käytettävän agentin tyyppin.
- "agent_executor_kwargs={'handle_parsing_errors': True}" -parametri ohjaa agenttia eteenpäin käsittelemällä kehoitteen suorittamisen vaiheissa ilmenneitä syntaksi- tai jäsentelyvirheitä, pyrkimällä korjaamaan ne automaattisesti.
- "max_iterations" asettaa varoimenpiteenä maksimimäärän agentin ottamille vaiheille, jolla estetään sitä jäämästä luuppiin.
- "top_k" määrittää sen määrän vastauksia, jotka agentti ottaa huomioon. Projektin tapauksessa se on säädetty vastaamaan tietokantataulujen maksimirivimääriä.

```

agent_executor = create_sql_agent(
    llm=llm,
    toolkit=sql_toolkit,
    agent_type=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
    agent_executor_kwargs={'handle_parsing_errors': True},
    max_iterations=20,
    top_k=3000,
)

```

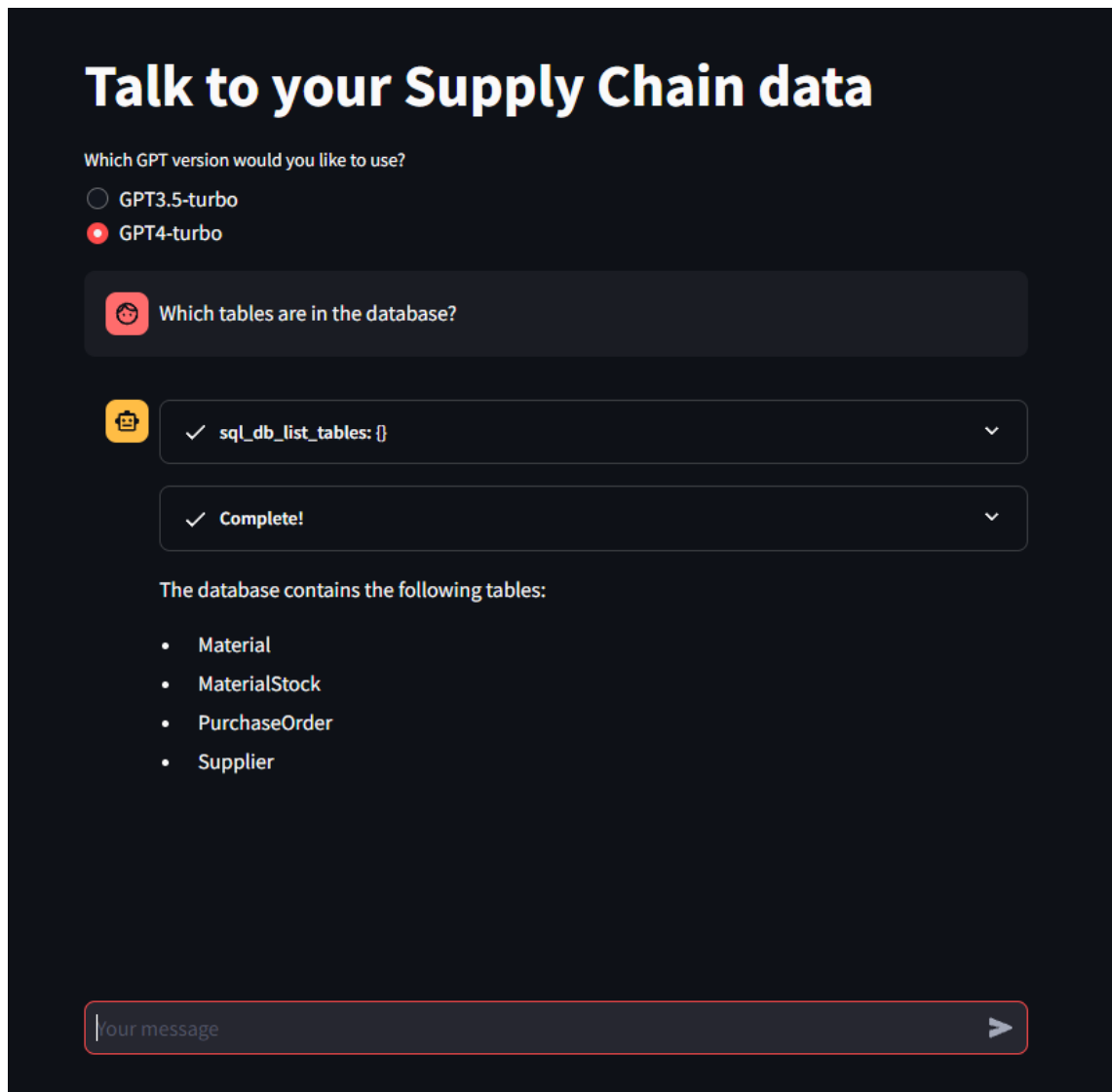
Esimerkkikoodi 6. SQL-agentin luonti ja määrittely parametrien avulla.

Ohjelman viimeisessä osassa luodaan ohjelmaan tuodun Streamlit -kirjaston avulla interaktiivinen chat-käyttöliittymä (Esimerkkikoodi 7), jossa käyttäjä voi syöttää kehoitteita ja saada tekoälyä käyttävältä agentilta vastauksia (Kuva 4). Siihen implementoitiin myös yksinkertainen Streamlit -valitsinelementti, jonka avulla käyttäjä kykenee vaihtamaan ohjelman käyttämää GPT-versiota. Koodi toimii seuraavalla tavalla:

- "if prompt := st.chat_input():" odottaa käyttäjän syöttämää kehoitetta käyttöliittymän kirjoituskenttään.
- "st.chat_message("user").write(prompt)" näyttää käyttäjän syöttämän viestin käyttöliittymässä.
- "with st.spinner('Please wait...'):" näyttää latauspyörän, joka osoittaa järjestelmän käsittelevän pyyntöä.
- "st_callback = StreamlitCallbackHandler(st.container())" mahdollistaa Streamlit -komponenttien reaaliaikaisen päivityksen kehoitteen suorituksen aikana.
- "response = agent_executor.invoke({"input": prompt}, {"callbacks": [st_callback]})" kutsuu aiemmin määritettyä agenttia, antaen sille käyttäjän syöttämän kehoitteen.
- "st.write(response["output"])" tulostaa agentin generoiman vastauksen verkkokäyttöliittymään.

```
if prompt := st.chat_input():
    st.chat_message("user").write(prompt)
    with st.spinner('Please wait...'):
        with st.chat_message("assistant"):
            st_callback = StreamlitCallbackHandler(st.container())
            response = agent_executor.invoke({"input": prompt},
                                             {"callbacks": [st_callback]})
            st.write(response["output"])
```

Esimerkkikoodi 7.



Kuva 4. Streamlit-verkkokäyttöliittymä.

3.3 Docker

Docker on avoimen lähdekoodin alusta, joka mahdollistaa sovellusten ja niiden sisältämien riippuvuuksien pakkaamisen ja suorittamisen konttimuotoisesti.

Dockerin avulla voidaan luoda yhtenäinen ja luotettavasti toimiva tuotantoympäristö, mikä eliminoi mahdollisia muusta järjestelmäympäristöstä johtuvia ongelmia. Projektissa Docker:n avulla ohjelmistokoodi pakattiin konttikuvaksi, joka puskettiin Azure:ssa sijaitsevaan konttirekisteriin.

Docker käyttää kontteja eristämään sovellukset ja niiden riippuvuudet muista järjestelmäympäristöistä luomalla kontin sisään sovellukselle oman käyttöjärjestelmälustan. Kontit on mahdollista konfiguroida kevyiksi ja nopeasti suoritettaviksi, mikä oli projektin ohjelman käyttötarkoitus huomioon ottaen käytännöllistä ja täten sopivampi valinta perinteisen virtuaalikoneen sijaan. (10.) Dockerin keskeisiä komponentteja ovat:

- Dockerfile: Tekstitiedosto, joka sisältää konttikuvien rakentamiseen käytettävät määrittäykset
- Docker Image: Konttikuva, joka sisältää tarvittavat tiedot ja määrittäykset sovelluksen ajamista varten.
- Docker Container: Kuvan suorituksen aikainen kontti-instanssi, jossa sovellus ajetaan.

Esimerkkikoodi 8 on projektin Dockerfile -tiedosto, joka määrittää miten sovelluksen Docker Image rakennetaan projektin sovellukselle. Se käyttää peruskuvana (base image) kevyttä Python 3.11.8-kuvaa, joka on optimoitu Debian 12 -jakelulle (distribution).

Dockerfilen sisältö on seuraava:

- Määritellään kontissa käytettävä peruskuva.
- Päivitetään pakettiluettelot, ja ladataan ja lisätään Microsoftin julkinen avain luotettuihin avaimiin, jotta ODBC-ajurit voidaan asentaa Debian 12 -järjestelmään.
- Hyväksytään ODBC -ajurin käyttöoikeussopimus ja asennetaan sen toiminallisuuden edellyttämät paketit.

- Määritetään kontti kuuntelemaan porttia, jota käytetään sovelluksen suorittamisessa.
- Määritetään työskentelyhakemisto (work directory), mihin Python -riippuvuudet, sovelluksen lähdekoodi sekä ympäristömuuttujat kopioidaan.
- Määritetään sovelluksen käynnistyskomento ja konfiguroidaan verkkoasetukset.

```
FROM python:3.11.8-slim-bookworm

RUN apt-get update && apt-get install -y wget gnupg \
    && wget -qO microsoft.asc https://packages.microsoft.com/keys/microsoft.asc \
    && apt-key add microsoft.asc \
    && rm microsoft.asc

RUN curl https://packages.microsoft.com/keys/microsoft.asc | tee
/etc/apt/trusted.gpg.d/microsoft.asc
RUN echo "deb [arch=amd64,arm64,armhf] https://packages.microsoft.com/debian/12/prod bookworm main" >
/etc/apt/sources.list.d/mssql-release.list

RUN apt-get update

RUN ACCEPT_EULA=Y apt-get install -y msodbcsql18
RUN apt-get install -y unixodbc-dev unixodbc

EXPOSE 8501

WORKDIR /usr/src/orasdemoapp
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
COPY app app
COPY .env .
CMD ["streamlit", "run", "app/main.py", "--server.port=8501", "--server.address=0.0.0.0", "--server.enableWebsocketCompression=false"]
```

Esimerkkikoodi 8. Dockerfile-tiedoston sisältö.

3.4 Azure Data Studio

Azure Data Studio on Microsoftin kehittämä avoimen lähdekoodin ohjelmisto, joka on suunniteltu tietokantojen käyttämiseen ja hallintaan. Projektissa sitä käytettiin SQL-tietokannan taulujen tuomiseen (import) CSV-tiedostoina, sekä tietokannan rakenteen muodostamiseen ja relaatioiden luontiin. Azure Data Studio sisältää myös kyselyeditorin, minkä avulla pystyttiin varmistamaan, että tekoälyagentin muodostamat tietokantakyselyt olivat toimivia ja tutkimaan käsitteellekö agentti kyselyn tuloksia oikein. (11.)

3.5 Git

Git on laajalti käytössä oleva hajautettu versionhallintajärjestelmä, jota käytettiin projektissa ohjelmistokehityksen apuvälineenä. Vaikka ohjelmiston kehitystyöhön ei osallistunut muita henkilöitä, Git mahdollisti joustavan työskentelyn useilla laitteilla sekä riippumattomuuden yksittäisestä fyysisestä työpisteestä. Sen käytön suurin hyöty liittyi versionhallintaan ja ohjelmiston muutoksien toimivuuden kokeiluun, ja poisti tarpeen paikallisten tiedostojen siirtämisestä eri laitteiden välillä. Työssä ohjelmistokoodia varten luotiin yksityinen säilö (private repository), jotta myös henkilökohtaiset tunnistimet, kuten ympäristömuuttujissa listattavat API-avaimet, sisältävät tiedostot pystyttiin synkronoimaan Git:iin.

Jatkokehityksen osalta Git tarjoaa myös työkalun CI/CD-työkalun (Continuous Integration and Continuous Deployment), jolla on mahdollista rakentaa automatisoitu työkulku (workflow) uuden sovellusversion julkaisun nopeuttamiseksi. GitHub Actions:n avulla on mahdollista konfiguroida Docker -konttikuvan automaattinen rakentuminen Git -säilöstä, ja siirtää se projektissa käytössä olleeseen Azure Web App -palveluun. Kyseisen toiminnallisuuden onnistunut rakentaminen olisi ollut hyödyksi, sillä se olisi vähentänyt ohjelmistokoodin päivityksestä seurannutta manuaalisen työn tarvetta. (12.)

3.6 KNIME

KNIME on avoimen lähdekoodin ohjelmisto, jota käytetään insinööriyössä raakadatan aggregointiin ja manipulointiin käyttötapaukseen ja ohjelmistokokoonpanoon sopivaksi. KNIME:llä voidaan rakentaa noodien (node) avulla modulaarisia työkulkuja, joiden avulla päivitetty data-aineisto voidaan aina käsitellä samalla tavalla valmiiksi konfiguroitujen työkulkupohjien mukaan. KNIME on myös integroitavissa osaksi pilvialustaan rakennettuja ETL-prosesseja, joka mahdollistaa datan käsittelyn automatisoinnin. (13.)

4 Käyttötapauksen kuvaus

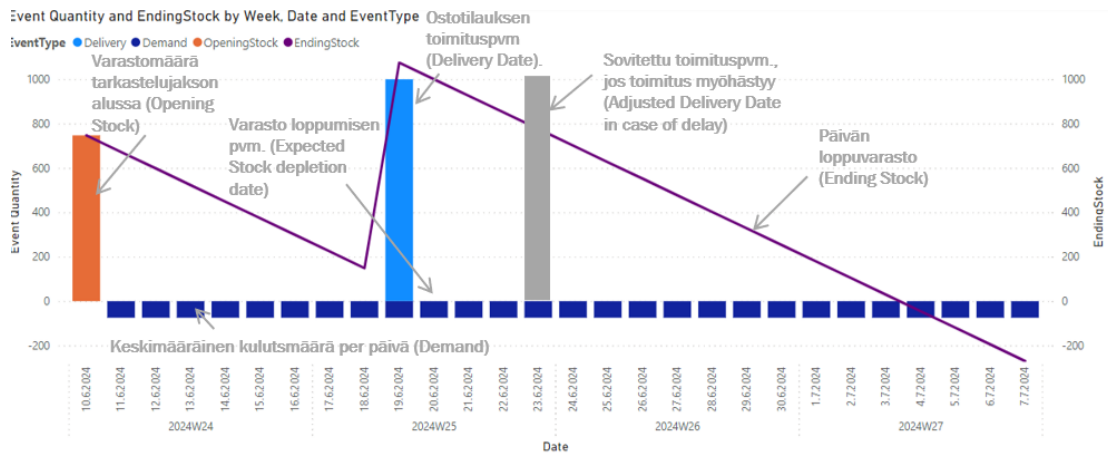
Käyttötapaus liittyy yrityksen X ostotoimintaan ja ostotilausten myöhästymisriskin tunnistamiseen. Yritys halusi selvittää, voiko kehitetyn AI-ratkaisun avulla tunnistaa ostotilaukset, joilla on riski myöhästyä sovitusta toimitusajasta luoden uhkan koko materiaalin loppumiseen varastosta. Riskin päättelyssä käytettävät arvot perustuvat materiaalien toimittajien historialliseen toimitusvarmuuteen, eli kykyyn toimittaa ostotilaus vahvistettuna toimituspäivänä.

Yrityksen muotoilema tavoite käyttötapaukselle oli seuraava:

Tunnista ostotilaukset, joiden toimitusaika tulee erikseen varmistaa toimittajalta perustuen toimittajan historialliseen toimitusvarmuuteen ja riskiin toimituksen myöhästymisestä.

Ostotilausten määrät ja saapumisten ajoitus perustuvat ostettujen materiaalien keskimääräiseen päiväkohtaiseen kulutukseen. Yritys tilastoi jatkuvasti tavarantoimittajien toimituksissa toteutuneita poikkeamia sovitusta toimituspäivistä. Kunkin toimittajan osalta lasketaan myöhästymisten keskiarvo "MYÖH-KA". Se kertoo, kuinka monta päivää toimittajan toimitus on keskimäärin myöhästynyt. Samalta toimittajalta voidaan ostaa yhtä tai useampaa materiaalia, jolloin toimittajan MYÖH-KA -lukua sovelletaan kaikkiin kyseiseltä toimittajalta tehtyihin ostotilauksiin.

Kuva 5 havainnollistaa tilanteet, joiden tunnistaminen oli yrityksen tavoitteena ja käyttötapauksen perustana. Jos päivän loppuvarasto "Ending Stock" laskee nol- laan tai negatiiviseksi siksi, että ostotilauksen toimituspäivämäärä "Delivery Date" myöhästyy toimittajan MYÖH-KA:n verran, on varaston loppumisen päivä- määrä "Expected Stock Depletion Date" ennen sovitettua toimituspäivämäärää "Adjusted Delivery Date".



Kuva 5. Esimerkki ratkaisun käyttötapauksesta

4.1 Käyttötapauksen data ja sen valmistelu

Yritys toimitti seuraavan käyttötapaukseen liittyvän CSV-muotoisen datan, joka tuotettiin yrityksen ostojärjestelmästä:

- Materiaalit: Aineisto koski kolmen tehtaan (Location) materiaaleja ja sisälsi 1717 erilaista toimipisteen ja materiaalin yhdistelmää.
- Ostotilaukset: yhteensä 2717 ostotilausriviä (yksi tai useampia rivejä yhdellä ostotilauksella).
- Toimittajat: 122 toimittajaa.

Data valmisteltiin ja muokattiin tekoälyagentille helpommin tulkittavaan muotoon saadusta aineisto seuraavien alaotsikoiden kuvaamien vaiheiden mukaisesti:

4.1.1 Datan organisointi

Data järjesteltiin relaatiotietokantarakenne noudattaen seuraaviin tietokantaan tuotuihin tauluihin:

- Material: Tehdaskohtaiset materiaalien perustiedot
- MaterialStock: Tehdaskohtaiset materiaalien varasto- ja keskimääräiset kulutustiedot
- PurchaseOrders: Tehdaskohtaiset materiaalien ostotilaustiedot
- Supplier: Toimittajat ja niiden luotettavuustiedot (MYÖH-KA).

4.1.2 Datan puhdistus ja formatointi

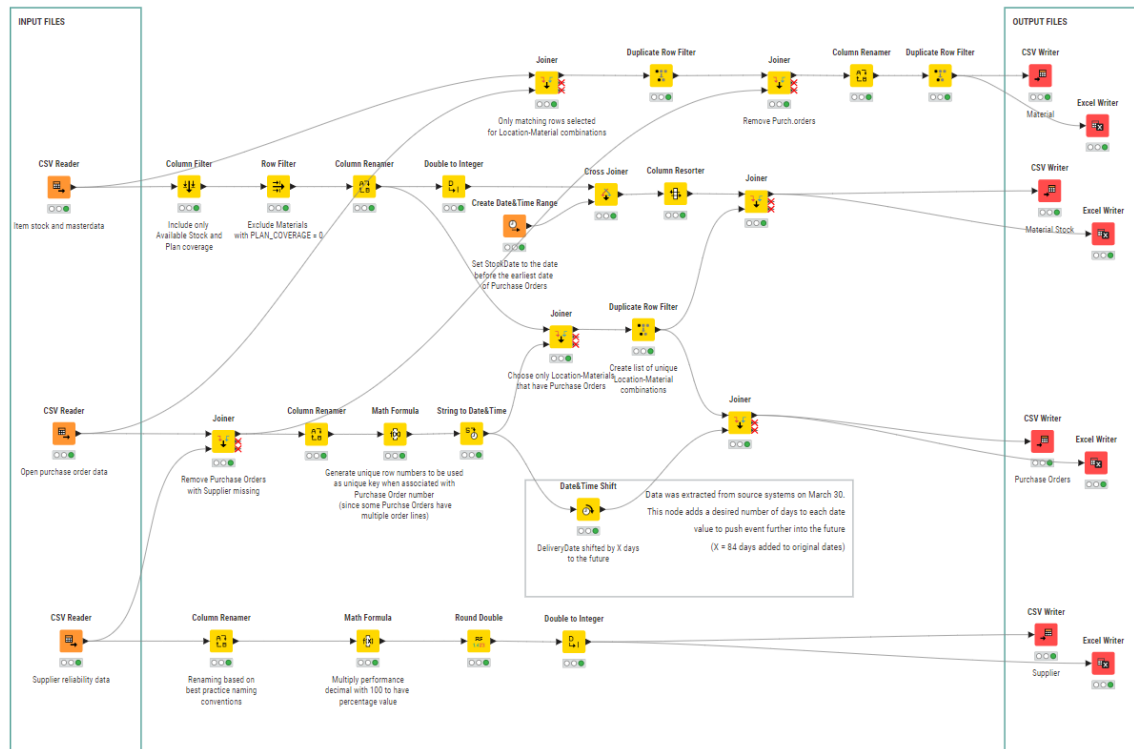
Data puhdistettiin tekoälyn tuottamien vastausten tarkkuuden ja oikeellisuuden parantamiseksi seuraavilla tavoilla:

- Useampaan kertaan esiintyvien sarakkeiden (duplikaatit) poistaminen.
- Ostotilauksille rivinumeroiden generointi yksikäsitteisen tauluavaimen luomiseksi.
- Sellaisten materiaalien poisto aineistosta, joilla ei ollut ostotilauksia.
- Data formatoitiin sellaiseen muotoon, että sarakkeiden tietotyypit saatiin asetettua niiden sisältämiä arvoja vastaaviksi.

4.1.3 Datan sisällön rikastaminen

- Sarakkeiden uudelleennimeäminen sen sisältämiä arvoja selkeästi kuvaavaan muotoon, jotta tekoälyagentin on helpompi yhdistää kehotteen luonnollisen kielen termit niitä vastaaviin taulukon sarakkeisiin.
- Keskimääräisen kulutusmäärän laskenta varaston riittävyyteen (päivien lukumäärä) perustuen.
- Aineistossa esiintyvien päivämäärien muokkaus niin, että työn aikana päivämäärät esiintyivät tulevaisuudessa (ei siis niin, että esim. saapuva ostotilaus näkyi menneellä päivämäärällä).

Edellä mainitut valmistelut toteutettiin käyttäen kappaleessa 3.6 kuvattua KNIME-ohjelmistoa, jolla automatisoitiin datan valmistelu. Näin muokausprosessia oli käytännöllistä konfiguroida muuttuvien tarpeiden mukaan ja ajaa uudestaan ilman manuaalista datan muokkausta. Kuva 6 esittää KNIME:llä luotua datan muokausprosessia, ja sen pohjalta luotua työnkulkua (workflow).



Kuva 6. Datan muokkaukseen käytetty KNIME-työnkulku

4.2 Datan visualisointi

Keskustelubotin antamien vastausten oikeellisuuden ja logiikan tarkastelun tuoksi data visualisoitiin Microsoft Power BI -sovelluksella. Tehdyt raportit helpottivat ja nopeuttivat vastausten tarkastusta sekä kehotteiden parantamista. Visualisointi oli myös tarpeen, kun tuloksia esiteltiin ja havainnollistettiin asiakasyrityksen henkilöille. Syöttötiedot Power BI:lle valmisteltiin jälleen KNIME-ohjelmistolla. Kuva 3 esittää tulosten tarkasteluun tehdyn Power BI -raportin. Raportti koostuu seuraavista osista:

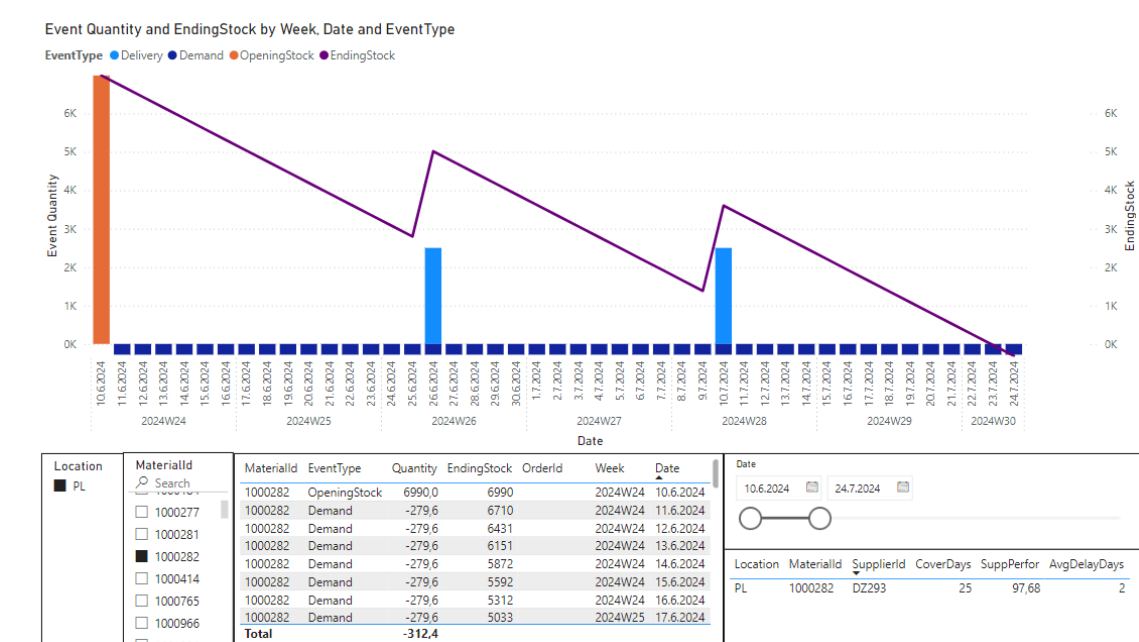
Yhdistetty pylväs ja viivadiagrammi:

- **Delivery:** Ostotilauksen vahvistettu saapumispäivämäärä (vaaleansininen pylväs)
- **Demand:** Materiaalin keskimääräinen päiväkohtainen kulutus (tummansininen pylväs)
- **OpeningStock:** Materiaalin varasto valitulla tehtaalla tarkastelujakson alussa (oranssi pylväs)

- EndingStock: Edellä mainittujen lukujen perusteella laskettu päivämääräkohtainen loppuvarasto (violetti viiva)

Tarkasteltavan tehtaan ja materiaalin valintalistat:

- Location: Tehdas.
- MaterialId: Materiaali.
- Luvut taulukkomuodossa.
- Aikajakson valitsin: Mahdollisuus tarkentaa haluttuun kohtaan tarkasteluajavälillä.
- Materiaalin tehdaskohtaiset perustiedot.
- CoverDays: Varaston kesto päivien lukumääränä.
- AvgDelayDays: MYÖH-KA, toimittajan keskimääräinen myöhästymisen vahvistetusta toimituspäivämäärästä kalenteripäivinä.



Kuva 7. Vastausten validoinnissa käytettävä Power BI -raportti

4.3 Kehotteiden kehittäminen ja tulosten tarkastelu

Keskustelubotin testaus tapahtui kehotteiden sisältöä muuttamalla, tarkastamalla vastauksia ja muutosten vaikutuksia sekä tutkimalla keskustelubotin ratkaisulogiikkaa sen antaman sanallisen ratkaisukuvauksen ja muodostaman SQL-kyselyn perusteella. Useiden kokeilujen kautta saadut lupaavat vastaukset koottiin ja dokumentoitiin seuraavien alalukujen mukaisesti. Dokumentointia käytettiin mallipohjana uusien kehotteiden luomiselle.

Esimerkit sisältävät syötetyn kehoteen (Kehote), keskustelubotin antaman vastauksen (Vastaus), osia kielimallin tulostamasta päättelyketjusta (Logiikka) sekä vastauksen validointikeinoja.

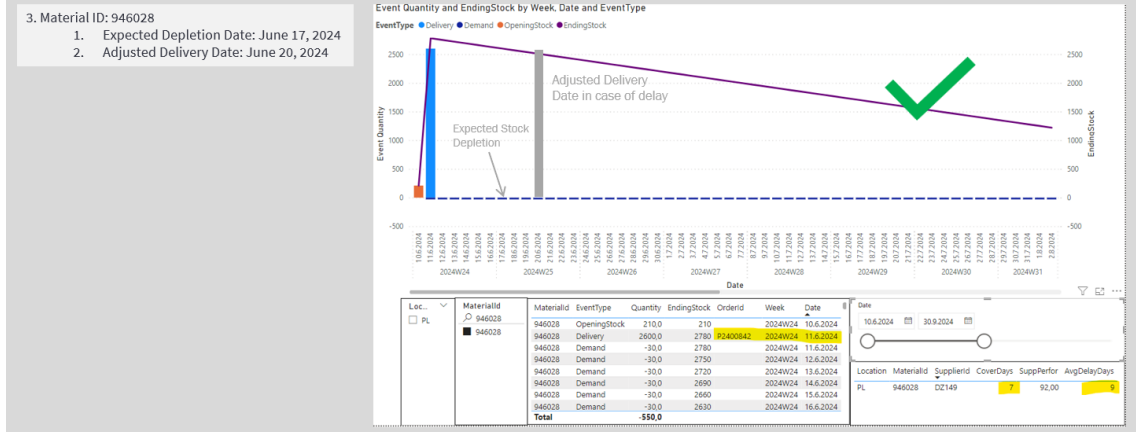
4.3.1 Kehote – Vastaus, esimerkki 1

Ensimmäisessä esimerkissä kehote pyytää tekoälyagenttia listaamaan kaikki ne tietokannassa olevat materiaalit PL-tehtaalla, jotka loppuvat varastosta materiaalien toimittajien MYÖH-KA -arvon mukaan. Tavoitteena on saada vastaus, jossa tekoälyagentti laskee kunkin toimittajan keskimääräisen toimitusviiveen ostotilauksen vahvistetun toimituspäivän päälle muodostaen uuden sovitetun toimituspäivän, ja listaa kaikki ne materiaalit, joiden varastosaldo ajautuu nolnaan mahdollisen toimitusviiveen seurauksena. Kehote myös pyytää sivuuttamaan tuloksissa materiaalit, joiden varastosaldot ajautuvat nolnaan toimittajista riippumattomista syistä. Taulukko 1 kuvaa tekoälyn generoimaa vastausta, ja vastauksen validointi tehdään Kuva 8:ssä käyttäen aiemmin kuvattua Power BI -raporttia.

Taulukko 1. Kehote – Vastaus, esimerkki 1

Kehote	Vastaus	Logiikka
<p>Have a look at the materials at Location PL, their stock quantities, stock coverages in days and delivery dates of purchase orders. However, ignore the Materials that run out of stock before the delivery of the earliest purchase order. For remaining Materials, assume that the delivery dates of the earliest purchases orders get delayed by the number of days based on average delay of respective suppliers. Which materials will run out of stock and when?</p>	<p>Here are some examples of materials at location "PL" that will run out of stock before the adjusted delivery date:</p> <p>Material ID: 219495 Expected Depletion Date: June 17, 2024 Adjusted Delivery Date: June 26, 2024</p> <p>Material ID: 438266 Expected Depletion Date: June 17, 2024 Adjusted Delivery Date: June 17, 2024</p> <p>Material ID: 946028 Expected Depletion Date: June 17, 2024 Adjusted Delivery Date: June 20, 2024</p> <p>Material ID: 418134 Expected Depletion Date: June 20, 2024 Adjusted Delivery Date: June 23, 2024</p> <p>Material ID: 1012090</p> <p>These materials will run out of stock before the adjusted delivery dates based on the earliest purchase orders and the average delay of respective suppliers.</p>	<p>To answer the question, I will perform the following steps:</p> <p>Identify materials at location "PL" from the MaterialStock table.</p> <p>For each material, calculate the expected stock depletion date based on the stock quantity and coverage in days.</p> <p>Retrieve the earliest delivery date for each material from the PurchaseOrder table. Check if the stock depletion date is before the earliest delivery date. If so, exclude these materials.</p> <p>For the remaining materials, adjust the earliest delivery date by the average delay from the Supplier table. Determine which materials will run out of stock before the adjusted delivery date.</p>

2024 Location PL / Material 946028



Kuva 8. Kehote - Vastaus, esimerkki 1. Alkuvaiheen validointi ja tuloksen visualisointi tehtiin materiaaleittain.

Sattumalta kaikki tekoälyagentin sanallisen vastauksen sisältämät materiaalit täyttivät halutun ehdon, eli varaston riittävyyden perusteella laskettu varaston loppumispäivämäärä sijoittui ostotilauksen vahvistetun toimituspäivämäärän ja sovitun toimituspäivän väliin. Seuraava vastauksen validointi tehtiin tekoälyagentin itsensä tuottaman listauksen perusteella, joka kattoi kaikki sen vastaukseen tunnistamat materiaalit.

Kuva 9 sisältää ensimmäiset 15 riviä tekoälyagentin tuottamasta 216 rivin listauksesta, sisältäen seuraavat sarakkeet:

- Row
- MaterialId
- DepletionDate
- DeliveryDate
- AvgDelayDays
- Adjusted DeliveryDate
- Correct - sisältää kaavan vastauksen oikeellisuuden päättelyyn.

Oletettavasti sattumalta vastauksen ensimmäiset ja sanalliseen esimerkkiin valikoituneet rivit olivat oikein. Ehtona on, että "StockDepletionDate" on suurempi

tai yhtä suuri kuin "DeliveryDate" ja "StockDepletionDate" on pienempi tai yhtä suuri kuin AdjustedDeliveryDate.

Validointi kuitenkin paljastaa, että vaikka tekoälyagentti pyrki toteuttamaan lo-
giikkakuvauksessa näkyvän ehdon "Determine which materials will run out of
stock before the adjusted delivery date", tulokseen kuitenkin valikoitui materiaa-
leja, joiden "StockDepletionDate" oli myöhäisempi päivämäärä kuin "Adjusted-
DeliveryDate". 216 materiaalin listalla ehdon täyttäviä (Correct = "YES") oli 50,
lopuilla riveillä ehto ei täyttnyt.

Yhteenvedoksi voidaan siis päätellä, että tekoälyagentin tuottama suppea vas-
taus oli ensimmäisten havaintojen perusteella oikein, mutta kokonaisuudessaan
sen vastaus sisälsi myös materiaaleja, joita kehoitteessa määritellyn logiikan
mukaan sen ei olisi pitänyt sisältää.

Row	MaterialId	DeliveryDate	StockDepletionDate	AdjustedDeliveryDate	Correct
1	219495	2024-06-17	2024-06-17	2024-06-26	YES
2	438266	2024-06-13	2024-06-17	2024-06-17	YES
3	946028	2024-06-11	2024-06-17	2024-06-20	YES
4	418134	2024-06-19	2024-06-20	2024-06-23	YES
5	1012090	2024-06-19	2024-06-20	2024-06-23	YES
6	159353	2024-06-18	2024-06-20	2024-06-22	YES
7	249266	2024-06-19	2024-06-21	2024-06-23	YES
8	219498	2024-06-17	2024-06-21	2024-06-26	YES
9	438255	2024-06-18	2024-06-21	2024-06-23	YES
10	946057	2024-06-18	2024-06-21	2024-06-21	YES
11	945232	2024-06-18	2024-06-21	2024-06-21	YES
12	602610	2024-06-19	2024-06-23	2024-06-21	LATER THAN ADJUSTED DATE
13	602588	2024-06-19	2024-06-24	2024-06-21	LATER THAN ADJUSTED DATE
14	438256	2024-06-20	2024-06-24	2024-06-24	YES

Kuva 9. Kuvakaappaus tekoälyagentin tuottamasta vastauksesta taulukoituna.

4.3.2 Kehote – Vastaus, esimerkki 2

Esimerkissä 1 mainittua kehotetta pyrittiin kehittämään tarkemmaksi tekoälyagentin päättelylogiikan parantamiseksi, ja sitä kautta vastauksen tarkentamiseksi. Kehotteella kysymys on sisällöltään sama, mutta tekoälyn päättelylogiikkaa johdatellaan yksityiskohtaisemmin. Taulukko 2 kuvaa muotoiltua kehotetta, ja Kuva 10 sen avulla saatua vastausta.

Taulukko 2. Esimerkki 1:n kehotteen pohjalta kehitetty kehoite, jonka avulla saatu vastaus tuotti halutun tuloksen.

Kehote	Vastaus	Logiikka
<p>Have a look at the materials at Location PL, their stock quantities, stock coverages in days and delivery dates of purchase orders. Considering suppliers' average delay, for each material please calculate the adjusted delivery date only for the earliest purchase order taking into account the delay of the respective supplier. Then identify the materials with stock coverage ending on date between the delivery date and the adjusted delivery date. Which materials fulfil this criteria and what are the respective dates of delivery, stock depletion and adjusted delivery?</p>	<p>Here are the materials that fulfill the criteria where the stock depletion date falls between the delivery date and the adjusted delivery date, along with the respective dates: (Kuva 10)</p>	<p>To address the query, I will perform the following steps:</p> <p>Identify the earliest purchase order for each material at location "PL".</p> <p>Calculate the adjusted delivery date by adding the average delay from the respective supplier to the delivery date of the earliest purchase order.</p> <p>Determine the stock depletion date by adding the stock coverage in days to the stock date for each material at location "PL".</p> <p>Filter materials where the stock depletion date falls between the original delivery date and the adjusted delivery date.</p>

Tuloksena saatiin lista 50 materiaalista. Sisältö vastasi esimerkin 1 listaa sillä poikkeuksella, että esimerkin 1 rivit "CORRECT = LATER THAN ADJUSTED DATE" eivät esiintyneet tuloksessa. Toisaalta kaikki esimerkin 1 tuloksessa esiintyneet "CORRECT = YES" rivit sisältyivät tekoälyagentin antamaan vastaukseen. Tulokset olivat siis "YES"- rivien osalta identtiset.

Vastaus oli siis kokonaisuudessaan oikein, ja se sisälsi oikein rajatun joukon vaaditun loogisen ehdon täyttäviä rivejä. Kehotteiden muotoilu oli tärkeässä roolissa haluttujen vastauksien saamisessa, sillä yksittäiset sanamuodot ja lauserakenteet vaikuttivat tekoälyagentin kehotekäsittelyyn merkittävästi. Myös agentin päättelylogiikan yksityiskohtaisempi ohjaaminen kehotteessa auttoi sitä käsittelemään kehote oikein.

Row	MaterialId	DeliveryDate	StockDepletionDate	AdjustedDeliveryDate	Correct
1	219495	2024-06-17	2024-06-17	2024-06-26	YES
2	438266	2024-06-13	2024-06-17	2024-06-17	YES
3	946028	2024-06-11	2024-06-17	2024-06-20	YES
4	418134	2024-06-19	2024-06-20	2024-06-23	YES
5	1012090	2024-06-19	2024-06-20	2024-06-23	YES
6	159353	2024-06-18	2024-06-20	2024-06-22	YES
7	249266	2024-06-19	2024-06-21	2024-06-23	YES
8	219498	2024-06-17	2024-06-21	2024-06-26	YES
9	438255	2024-06-18	2024-06-21	2024-06-23	YES
10	946057	2024-06-18	2024-06-21	2024-06-21	YES
11	945232	2024-06-18	2024-06-21	2024-06-21	YES
14	438256	2024-06-20	2024-06-24	2024-06-24	YES
21	1002390	2024-06-27	2024-06-27	2024-07-03	YES
24	158556-77	2024-06-27	2024-06-27	2024-07-01	YES

Kuva 10. Kuvakaappaus agentin tuottamasta taulukoidusta vastauksesta.

5 Yhteenveto

Tässä kappaleessa kuvataan insinööriyön kehityksessä ilmenneitä vaatimuksia ja haasteita, sekä sen käyttöä rajoittavia teknisiä tekijöitä. Lopuksi pohditaan myös työn lopputuloksena syntyneen ratkaisun jatkokehitysmahdollisuuksia.

5.1 Insinööriyössä kehitetyn ratkaisun ongelmat

5.1.1 Tiedon hankinta

Kuten kirjoitelman johdannossa kuvattiin, insinööriyön alussa yrityskäyttöön soveltuvista strukturoitua dataa analysoivista tekoälysovelluksista oli varsin vähän tietoa saatavilla. Työn teknisen osuuden eteneminen oli tästä syystä varsin hidasta. Projektin aikana seuraavat loogiset työvaiheet eivät olleet aina selkeitä, sillä käyttötarkoitukseen soveltuvien ja keskenään yhteensopivien alustojen, teknologioiden ja ohjelmistojen löytäminen oli haastavaa. Tekoälyä hyödyntävät ohjelmistot myös kehittyivät nopeasti, ja muutoksien ja niiden vaikutusten ymmärtäminen ratkaisua kehittäessä vaati jatkuvaa oman tiedon päivittämistä.

5.1.2 Lähdedatan laatuvaatimukset

Tekoälyratkaisun pohjana toimivan lähdedatan eheys ja johdonmukaisuus on sen koko toiminnan kannalta kriittisessä roolissa. Datan perusteellinen läpikäynti ja käsittely voi toimitetun datan laadusta ja rakenteesta riippuen olla vaativa prosessi. Esimerkiksi insinööriyön käyttötapauksessa lähdedatasta jouduttiin siivoamaan kaikki ne artikkelit, joilta puuttuivat halutun vastauksen saamisen kannalta vaaditut perustiedot. Epätäydellinen lähdedata saattoi häiritä tekoälyn päättelyä, ja täten vaikuttaa suoraan vastausten laatuun. Datan käsittelyprosessissa on hyödyllistä käyttää ohjelmistoja, jotka tarjoavat monipuoliset työkalut datan muokattavuuteen ja integrointimahdollisuuden suoraan osaksi dataputkia (data pipeline), jotta koko dataprosessi on automatisoitavissa.

5.1.3 Tekoälyn vastausten validointi

Tekoälyn tuottamien vastausten oikeellisuuden validointi on merkittävä osa koko ratkaisun kehittämistä ja käyttöä. Käyttäjän syöttämien kehoitteiden laatu vaikuttaa suoraan ratkaisun tuottamien vastausten oikeellisuuteen, joten kehoitteiden oikeaoppisen muodostamisen opettelu on tärkeää. Tekoälyä hyödyntävien data-analytiikkaratkaisujen käyttöönotto edellyttää siis yritykseltä varsinaisen teknisen ratkaisun rakentamisen lisäksi panostusta myös varsinaisen loppukäyttäjryhmän koulutukseen. Halutulla tavalla toimien kehoitteiden dokumentointi on tärkeää, sillä niiden pohjalta käyttäjien on luonnollisesti helpompi muodostaa uuteen ongelmaan oikean vastauksen tuottama kehote.

5.1.4 Token-raja

Kielimallit käsittelevät tekstin ”tokeneina”, joka on luonnollisen kielen prosessoinnissa käytettävä datayksikkö. Kehotteen ja vastauksen tekstin pituus ja niiden merkkisisällöt muodostavat numeerisen ”token”-arvon. Jokaisella kielimallilla on mallikohtainen ”token”-raja, joka määrittää kuinka paljon tekstiä se voi vastaanottaa syötteenä ja tuottaa vastauksena yhdessä istunnossa. Insinööriön käyttötapauksessa tekoälyn antamat vastaukset sisälsivät usein suuren joukon tekstiä tietokannassa olevien rivien muodossa, jolloin kielimalli saattoi ta-pauskohtaisesti rajoittaa sen tuottaman vastauksen laajuutta ”token”-rajan täytyessä. Edellä mainitusta syystä esimerkiksi keskusteluhistorian käyttöönottoa ei pidetty ratkaisun toimivuuden kannalta järkevänä, sillä kielimallina pääasiallisesti käytetyn ”GPT-4 Turbo”:n tulosteen 4096 ”tokenin”-raja ei aina olisi riittävä jatkotulosteiden generoimiseen tekoälyn aiempien vastausten pohjalta. (14.)

5.2 Jatkokehitys

5.2.1 Tekoälyn käyttötapauksen laajentaminen

Esitetyssä käyttötapauksessa lähtötietoina tarjottiin tilastoitua dataa materiaali-toimittajien suorituskyvystä ja toimitusten myöhästymisistä. Tutkimusyhtiö Gartnerin tutkimuksen ”What Supply Chain Leaders Are Planning for GenAI Solutions in 2024” tuloksessa tutkitut yritykset kertoivat suunnitelmistaan vuodelle 2024, että liiketoimintanäkökulmasta merkittävin käyttökohde tekoälylle liittyy liiketoiminnan suorituskykyindikaattorien (Key performance indicator) tuottamiseen ja diagnostiikkaan. Esitetyssä käyttötapauksessa kehitysaskel voisikin olla juuri toimittajien suorituskyvyn ja toimitusten luotettavuutta arvioivien suorituskykyindikaattorien suora muodostus tekoälyn avulla. Laajennetussa ratkaisussa tekoälyllä voitaisiin analysoida toimittajien suorituskyvyn merkityksellisyyttä muodostettujen tunnuslukujen pohjalta, yhdistämällä materiaalihankintoihin liittyvä data yrityksen myynti- ja toimitusdataan.

5.2.2 Integrointi osaksi toiminnanohjausjärjestelmää

Ratkaisun vaikuttavuutta lisäisi sen integrointi osaksi yrityksen toiminnanohjausjärjestelmää ja sen sisältämiä tietokantoja. Se avaisi tien tekoälyn laajalle käyttöönotolle liiketoiminnan kriittisillä osa-alueisilla, ja mahdollistaisi järjestelmässä olevan reaaliaikaisen datan pohjalta tehtävät kyselyt.

5.2.3 Kielimallin hienosäätäminen

Kielimallin hienosäätäminen (fine-tuning) on prosessi, jossa käytettävää kielimallia pyritään kouluttamaan ennalta määritellyn käyttötarkoituksen mukaisesti sille syötettävän koulutusmateriaalin avulla. Insinööriyön käyttötapauksessa hienosäätö tarkoittaisi laajan aineiston kokoamista, joka sisältäisi satoja pareja luonnollisella kielellä esitettyjä kysymyksiä ja niitä vastaavia SQL-kyselyitä. Sen avulla olisi pystytty vähentämään ratkaisun tuottamia syntaksivirheitä ja epäoptimaalisia kyselyrakenteita, ja täten parantamaan kehotteiden käsittelyä.

Hienosäädetyllä kielimallilla on kyky ymmärtää käyttäjän tarpeita tarkemmin, joka lisää käyttäjäystävällisyyttä toimivien kehoitteiden muodostamisen ollessa helpompaa. Myös kielimallin tarjoamat vastaukset olisivat selkeämpiä ja vastaisivat paremmin käyttötarkoitusta.

Insinööriyössä kielimallia ei päädytty hienosäätämään, sillä se olisi vienyt paljon aikaa ja vaatinut syvää perehtymistä aihepiiriin. Koulutusmateriaalin tulee muodostua laadukkaista ja monipuolisista kehote-vastaus-pareista, ja hienosäädetyllä kielimallilla toiminta on validoitava, jotta sen tuottamat vastaukset eivät painotu liikaa koulutusmateriaalina käytettyyn dataan. Mallin hienosäätö tuo myös lisäkustannuksia koulutustyön (training job) vaatiman laskentatehon vuoksi.

(15.)

5.2.4 Uusien kielimallien käyttöönotto

Tulevaisuudessa julkaistavat uudet kielimalliversiot tulevat tarjoamaan merkittäviä parannuksia tekoälyä käyttävien sovelluksien toimintaan. Luonnollisen kielin prosessointi tehostuu jatkuvasti, mikä lisää tekoälyratkaisulle syötettävän kehoitteen käsittelyn nopeutta sekä niillä saatavien vastausten laatua, samalla kun tekoälyn käytön kustannukset laskevat. Myös aiemmin mainittujen ”token”-rajojen noustessa suuremmiksi uusien kielimalliversioiden mukana, tekoälylle voidaan jatkossa syöttää monimutkaisia ja monimuotoisia ongelmia ratkaistavaksi, sekä saada ulos käyttötapaukseen räätälöityjä kattavia ja havainnollistavia vastauksia.

5.3 Tilaajan lausunto

Opinnäytetyön tilannut Cuando Consulting Oy on toimitusketjun suunnitteluun ja analytiikkaan erikoistunut konsultointiyhtiö. Yhtiö auttaa asiakkaitaan toimitusketjun parantamisessa päästä-päähän kattaen kysynnän, tuotannon, varastojen, logistiikan ja hankintojen suunnittelun ja niihin liittyvän analytiikan.

Cuandon asiakkaat ovat joko aloittaneet tai kiinnostuneita aloittamaan tekoälyn käytön selvityksen toimitusketjuprosessien suunnittelussa. Kolmesta opinnäytetyöstä kiinnostuneesta yrityksestä nopeimmin konkreettisen casen ja tarvittavat resurssinsa tarjonnut valikoitui kohdeyritykseksi.

Opinnäytetyön lopputulos, Azure OpenAI -chatbot, vastasi hyvin kohdeyrityksen tarpeisiin. Tavoitteena oli arvioida ja ymmärtää tekoälyn kyky ja tapa käsitellä luonnollisella kielellä esitettyjä operatiivisia kysymyksiä. Lisäksi yritys halusi ymmärtää, miten tekoäly voidaan valjastaa lukemaan relaatiotietokantaan tallennettuja toiminnanohjausjärjestelmien tietoja. Koska kehoitteiden suunnittelu on vastausten laadun kannalta keskeisessä roolissa, kehitetty keskustelubotti palveli hyvin oppimista. Yrityksen muut kysymykset liittyivät lähinnä dataintegraation toteutukseen toiminnanohjausjärjestelmästä chatbotin lukemaan tietokantaan, mutta nuo kysymykset eivät kuuluneet työn varsinaiseen aihepiiriin.

Cuando Consulting Oy sai opinnäytetyön tuloksena työkalun sekä analytiikkapalvelujensa myyntiin että perustan jatkokehitykselle. Kiitän Vili Häkkistä sinnikkydestä aiheen omaksumisessa ja ratkaisun toteutuksessa sekä laadukkaasta lopputuloksesta.

Pekka Häkkinen
Toimitusjohtaja, omistaja
Cuando Consulting Oy

Lähteet

- 1 What is Azure OpenAI Service. <<https://learn.microsoft.com/en-us/azure/ai-services/openai/overview>>. 15.2.2024.
- 2 What is Azure SQL Database? <<https://learn.microsoft.com/en-us/azure/azure-sql/database/sql-database-paas-overview?view=azuresql>>. 12.2.2024.
- 3 Service Level Agreement for Microsoft Online Services (WW). <<https://www.microsoft.com/licensing/docs/view/Service-Level-Agreements-SLA-for-Online-Services?lang=1>>. 5/2024.
- 4 High availability for Azure SQL Database. <<https://learn.microsoft.com/en-us/azure/azure-sql/database/high-availability-sla>>. 12.3.2024.
- 5 Introduction to Azure Container Registry. <<https://learn.microsoft.com/en-us/azure/container-registry/container-registry-intro>>. 23.5.2024.
- 6 App Service overview. <<https://learn.microsoft.com/en-us/azure/app-service/overview>>. 31.8.2023.
- 7 Azure App Service plan overview. <<https://learn.microsoft.com/en-us/azure/app-service/overview-hosting-plans>>. 26.5.2023.
- 8 Querying structured data with Azure OpenAI, Valentina Alto. <<https://medium.com/microsoftazure/querying-structured-data-with-azure-openai-e59ee43867e5>>. 13.2.2023.
- 9 Anaconda dokumentaatio. <<https://docs.anaconda.com/index.html>>.
- 10 Docker -dokumentaatio. <https://docs.docker.com/>.
- 11 What is Azure Data Studio? <<https://learn.microsoft.com/en-us/azure-data-studio/what-is-azure-data-studio>>. 12.10.2023.
- 12 GitHub -dokumentaatio. <<https://docs.github.com/en>>.
- 13 KNIME -dokumentaatio. <<https://docs.knime.com/>>.
- 14 What are tokens and how to count them? <<https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them>>.

- 15 Fine-tuning. OpenAI. <<https://platform.openai.com/docs/guides/fine-tuning>>.
- 16 Microsoft Learn -portaalin muut alisivut. <https://learn.microsoft.com/>.
- 17 Medium -sivuston muut alisivut. <https://medium.com>.
- 18 Langchain -dokumentaatio. <https://python.langchain.com/v0.2/docs/introduction/>.
- 19 OpenAI -yhteisön foorumi. <https://community.openai.com/>.
- 20 Stack overflow -portaali. <https://stackoverflow.com/>.
- 21 Docker -foorumi. <<https://forums.docker.com/>>.