Beskrivelse av prosjektet | IS-314 | Gruppe 13

Gruppemedlemmene

Navn	Rolle	LinkedIn
Amund Mikalsen	Gruppeleder	LinkedIn
Simon Portillo	Hovedutvikler	LinkedIn
Martin Goberg	Utvikler	LinkedIn
Anders Fløysvik	Test-utvikler	LinkedIn
Jone M. Skribeland	Data-agent utvikler	LinkedIn
Petter Nærum	Frontend-utvikler	LinkedIn

Oppdragsgiver – Egde og Miljøfyrtårn

Egde er et konsulentselskap som leverer teknologiske løsninger og rådgivning innen digitalisering, programvareutvikling og kunstig intelligens, og er vår oppdragsgiver i prosjektet. Vi inngår i faggruppen Smia, med veiledere innen data og analyse.

Stiftelsen Miljøfyrtårn er en sertifiseringsordning som hjelper virksomheter med å dokumentere og forbedre sitt miljøarbeid, og er kunde i prosjektet. De håndterer store mengder sertifiseringsdata, noe som gjør dem til en god kandidat for moderne data-grensesnitt.

Prosjektbestilling

Den endelige prosjektbestillingen var å utvikle en database- og klientagnostisk MCP-server med en tilhørende MCP-klient. Løsningen lar brukere stille spørsmål i naturlig språk gjennom et intuitivt chat-grensesnitt og motta svar generert av en agent basert på relevante data. Et særlig fokus ble lagt på en interoperabel arkitektur som kan integreres med ulike datakilder og klienter uten å være låst til spesifikke teknologivalg.

Som en deloppgave ble Microsoft Fabric Data Agent introdusert etter at vår veileder hos Egde deltok på en Microsoft-konferanse. Fabric Data Agents lar brukere snakke med strukturerte data i naturlig språk, hvor en språkmodell oversetter spørsmål til relevante dataforespørsler. I motsetning til hovedoppgaven er dette en ferdig «hyllevare» løsning fra

Microsoft. Målet med deloppgaven var å sammenligne de to tilnærmingene, slik at Egde og Miljøfyrtårn kan vurdere når den ene framgangsmåten passer bedre enn den andre.

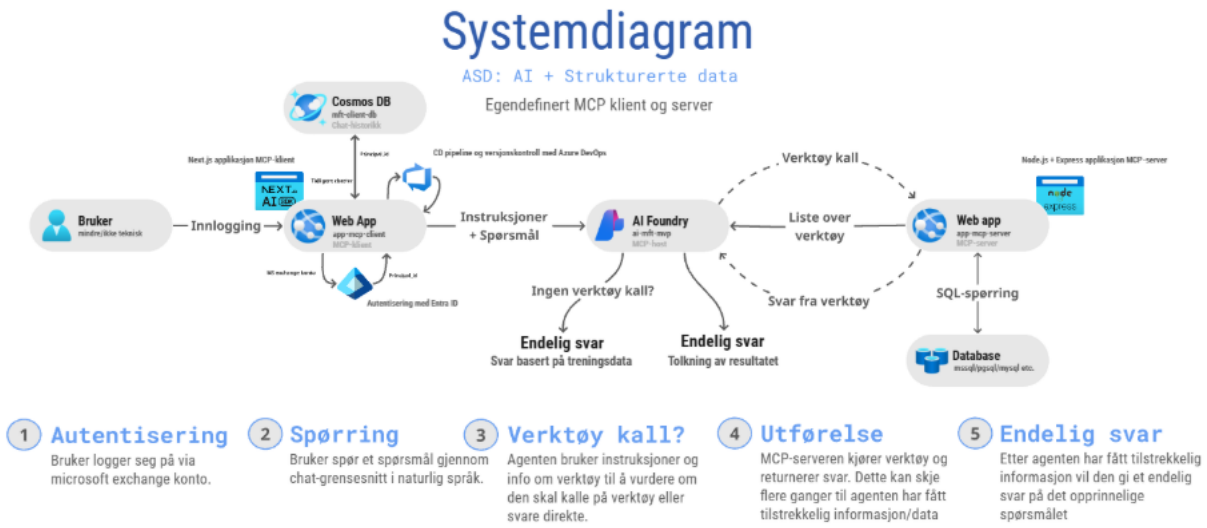
Utfordringer

Et sentralt smertepunkt var kombinasjonen av Next.js og Azure App Service. Next.js forventer å kjøre som en vanlig Node-server, men Azure App Service sin Node-hosting samsvarer dårlig med disse forutsetningene. Dette tvang oss til å tilpasse bygg- og deploy-prosessen for å etterligne miljøet Next.js forventer, noe som resulterte i tre lange dager og over femti forsøk før CI/CD-pipelinen fungerte. I ettertid er det tydelig at enten Vercel-hosting eller et container-miljø som Azure Container Apps ville gitt en langt enklere vei, da begge samsvarer direkte med hvordan Next.js forventer å kjøre.

En åpenbaring senere i prosjektet var at MCP-serveren kunne utvides til å returnere interaktive grafiske grensesnitt med MCP-Apps og ikke bare tekst. Dette ble oppdaget underveis i prosjektet etter vi allerede hadde utviklet et diagramverktøy i klienten. I ettertid burde vi tatt i bruk utvidelsen fra start, slik at alle klienter som koblet seg på serveren kunne dra nytte av vektøyet. Videre førte dette til apps implementering på både serveren, og klienten.

Systemdiagram

Dette systemdiagrammet viser den endelige løsningen som er en skreddersydd MCP-Server og klient med egenutviklede verktøy for databaseinteragering.



Teknologi

Både MCP-Serveren og klientapplikasjonen er utviklet i TypeScript, med Next.js som fullstack-rammeverk for klienten. Valget av et felles programmereingsspråk på tvers av

applikasjonen gjorde det mulig å dele typedefinisjoner mellom server og klient, noe som reduserte risikoen for skjemadrift og gir en mer sømløs utviklingsprosess. For å bygge den agentiske oppførselen i klienten ble Vercel AI SDK brukt som et abstraksjonslag over Azure-APIet, som gjør koden modell-agnostisk og ikke låst til en enkelt AI-tilbyder.

For å bygge brukergrensesnittet brukte vi Shadcn som komponenbibliotek. Ved å bruke Shadcn så ga det oss friheten til å tilpasse komponentene etter behov, uten å være låst til et stivt designsystem.

All infrastruktur er satt opp i Azure, i tråd med oppdragsivers eksisterende miljø. Applikasjonene hostes i Azure App Service, mens autentisering håndteres av Microsoft Entra ID, som sikrer at kun autoriserte brukere fra Miljøfyrtårn får tilgang til løsningen. CI/CD-pipelines ble satt opp i Azure DevOps for automatisert bygging og utrulling.

For strukturert data benyttes Microsoft SQL Server og MSSQL-database, som MCP-serveren kommuniserer med ved hjelp av SQL-spørringer generert av agenten. I tillegg er CosmosDB tatt i bruk som NOSQL-Database for lagring av samtalehistorikk i klientapplikasjonen.

Ting vi er fornøyde med

Prosjektet har gitt oss verdifull erfaring med arbeidsstrukturering og rollefordeling i praksis. Gjennom hele semesteret har vi samarbeidet tett og fordelt oppgaver på en strukturert måte ved hjelp av verktøy som Jira og Azure DevOps. Disse verktøyene er vi fornøyde med fordi det har gitt oss en god oversikt og en effektiv fremdrift i prosjektet.

Mye av tiden vår har gått til å utforske og eksperimentere med hvordan AI-agenter fungerer i praksis, spesielt hvordan kontekst, instruksjoner og begrensninger påvirker oppførsel og kvalitet i svarene. Dette har gitt oss innsikt i hvor sensitivt slike systemer er for små endringer, og hvor viktig der er å designe dem på en måte som gir både forutsigbarhet og fleksibilitet. Gjennom iterativ testing og justering har vi gradvis oppnådd en løsning som vi er fornøyde med. Den oppleves stabil, konsistent og godt tilpasset sitt bruksområde.

Gruppen har opparbeidet seg verdifull erfaring innen systemutvikling, MCP, kunstig intelligens generelt igjennom eksperimentering og forskning, og ikke minst svært god erfaring i å jobbe sammen på et stort prosjekt av dette omfanget. Vi er veldig fornøyde med hvor engasjerte veilederne har vært i de ulike teknologiene og hvordan det kan påvirke og hjelpe deres egen utvikling av bedriften.

Resultat

Resultatet av dette prosjektet er en velfungerende MCP-Server med en tilknyttet MCP-Klient. Vi er svært fornøyde med løsningen vi har utviklet, både når det gjelder funksjonalitet, stabilitet og brukervennlighet. Prosjektet oppfyller de kravene som ble satt fra starten av prosjektperioden. Løsningen vi har laget har potensial for videreutvikling og utvidelse med mer funksjonalitet i fremtiden, for eksempel flere MCP-Apps og verktøy. Vi mener dette prosjektet har vært veldig lærerikt og vellykket, og at sluttresultatet viser kompetansen og innsatsen som er lagt ned i arbeidet.

Slik ble det endelige produktet seende ut. Det har sterke trekk fra det klassiske chat-brukergrensesnittet.

Dette bilde viser muligheten til å fremstille grafer i brukergrensesnittet.

Slik kan tabeller fremstilles i brukergrensesnittet fra serveren via MCP-Apps. Bilde viser vår klient-agnostiske MCP-Server med Claude som klient.