Høgskulen på Vestlandet

BACHELOROPPGAVE

Applikasjon for brukerstatistikk fra porteføljesystem

Application for user statistics from a portfolio system

Gruppe D37

Adrian Birkedal

Anders Lerang

Nicolai Holmefjord

DAT191

Fakultet for ingeniør- og naturvitenskap

Institutt for Data- og realfag

Veileder: Sven-Olai Høyland

Oppdragsgiver: Stacc Escali

Innleveringsdato: 22.05.2023

Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

TITTELSIDE FOR HOVEDPROSJEKT

Rapportens tittel: Applikasjon for brukerstatistikk fra porteføljesystem:	Dato: 22.05.2023
Forfatter(e): Adrian Birkedal, Anders Lerang og Nicolai Holmefjord	Antall sider u/vedlegg: 41
	Antall sider vedlegg: 44
Studieretning: Dataingeniør og informasjonsteknologi	Antall disketter/CD-er: 0
Kontaktperson ved studieretning: Sven-Olai Høyland	*Gradering:* Ingen
Merknader:	

Oppdragsgiver: Stacc Escali	Oppdragsgivers referanse:
Oppdragsgivers kontaktperson: Gerdt Sverre Vedeler	Telefon: 926 22 063

Sammendrag:

Stacc Escalis hovedløsning er et porteføljesystem som leverer økonomiske tjenester til kunder i privat og offentlig sektor. Innsamling av brukerstatistikk fra hovedløsningen er en manuell og tidskrevende prosess. Bachelorgruppen har fått i oppgave å utvikle en del av et system for innsamling av brukerstatistikk fra porteføljesystemet. Gruppen har utviklet et API som henter og lagrer brukerstatistikk i en tildelt database, eid av Stacc Escali. Deretter skal brukerstatistikken presenteres på et dashbord.

Stikkord:

API	Brukerstatistikk	Dashbord

Høgskulen på Vestlandet, Fakultet for ingeniør- og naturvitenskap
Postadresse: Postboks 7030, 5020 BERGEN Besøksadresse: Inndalsveien 28, Bergen
Tlf: 55 58 75 00 Fax: 55 58 77 90 E-post: post@hvl.no Hjemmeside: http://www.hvl.no

Forord

Denne rapporten presenterer arbeidet som er gjort i prosjektet "Applikasjon for brukerstatistikk fra porteføljesystem". Prosjektet er en del av bacheloroppgaven til Adrian Birkedal, Anders Lerang og Nicolai Holmefjord, som alle studerer informasjonsteknologi og dataingeniørstudier ved Høgskulen på Vestlandet.

Vi vil gjerne takke Stacc Escali for oppgaven, forretningsutvikler i Stacc Escali – Gerdt Vedeler og alle som har vært involvert i prosjektet. Disse har vært en betydelig ressurs for oss, og har delt sin erfaring og hjulpet oss med å løse utfordringer underveis i prosjektet.

Vi ønsker å takke Sven-Olai Høyland for gode råd og tilbakemeldinger under prosjektet.

I tillegg ønsker vi også å takke Sondre Knutsen, utvikler i Stacc Escali, for teknisk hjelp og innspill som har hjulpet å forme systemet.

Med vennlig hilsen,

Adrian, Anders og Nicolai

Innholdsfortegnelse

FORORD ..... **ii**

INNHOLDSFORTEGNELSE ..... **iii**

FIGURLISTE ..... **vi**

TABELLISTE ..... **vi**

ORDLISTE ..... **vii**

1 INNLEDNING **1**

1.1 KONTEKST 1

1.2 MOTIVASJON 1

1.3 PROSJEKTEIER 2

1.4 PROBLEMBESKRIVELSE OG MÅL 2

1.4.1 Problembeskrivelse *2*

1.4.2 Mål *2*

1.5 OPPBYGGING AV RAPPORTEN 3

2 PROSJEKTBESKRIVELSE **4**

2.1 PRAKTISK BAKGRUNN 4

2.1.1 Tidligere arbeid *4*

2.1.2 Initielle krav *4*

2.1.3 Initiell løsnings-idé *4*

2.2 AVGRENSNINGER 5

2.3 RESSURSER 6

3 DESIGN AV PROSJEKTET **7**

3.1 FORSLAG TIL LØSNING 7

3.1.1 Alternative løsninger for testverktøy *7*

3.1.2 Alternative løsninger for visualisering av data *8*

3.2 VALGT LØSNING 9

3.3 VALG AV ANDRE VERKTØY 10

3.4 PROSJEKTMETODIKK 10
3.4.1 Utviklingsmetodikk 10
3.4.2 Prosjektplan 11
3.4.3 Risikoanalyse 12
3.5 EVALUERINGSPLAN 12
4 DETALJERT LØSNING 13
4.1 SYSTEMARKITEKTUR 13
4.2 API FOR BRUKERSTATISTIKK 14
4.2.1 Model 15
4.2.2 Handlers 15
4.2.3 Controllers 16
4.2.4 Data transfer object (DTO) 16
4.2.5 Swagger 17
4.3 DATABASELØSNING 17
4.3.1 Omstrukturering av database 17
4.3.2 Visning 18
4.4 VISUALISERING AV DATA 20
4.4.1 Innhenting av data 20
4.4.2 Behandling av data 20
4.4.3 Presentasjon av data 21
4.4.4 Bruk av dashbordet 22
5 RESULTATER 23
5.1 EVALUERINGSMETODE 23
5.2 EVALUERINGSRESULTAT 24
5.2.1 Enhetstesting 24
5.2.2 Integrasjonstesting 24
5.2.3 Systemtest 25
5.2.4 Brukertest 25

5.3 PROSJEKTRESULTAT 26
6 DISKUSJON 26
6.1 FRA INITIELL IDÉ TIL ENDELIG PRODUKT 26
6.1.1 Programvaregrensesnitt 27
6.1.2 Databaseløsning 27
6.1.3 Visning 28
6.2 DRØFTING 29
6.3 OPPSUMMERING AV PROSJEKTET 30
7 KONKLUSJON OG VIDERE ARBEID 31
7.1 KONKLUSJON 31
7.2 VIDERE ARBEID 31
8 REFERANSER 32
9 VEDLEGG 34
9.1 VEDLEGG A – GANTT-DIAGRAM 34
9.2 VEDLEGG B - RISIKOANALYSE 35
9.3 VEDLEGG C – DASHBORD-ELEMENTER 36

Figurliste

Figur 1: Sammenhengen mellom de ulike instansene 13
Figur 2: Design av API 15
Figur 3: Swaggers interaktive grensesnitt 17
Figur 4: ER-Modell av databaseløsning 18
Figur 5: Første del av spørringen henter ut relevante kolonner for kundetilbakemeldinger . 18
Figur 6: Andre del av spørringen bruker union for mer omfattende datavisning 19
Figur 7: Stegene som blir gjort i Power Query 20
Figur 8: Samspill mellom kundeliste og diagrammer 21
Figur 9: Oppdaterte diagrammer ved bruk av søkefunksjon og valg av én kunde 21
Figur 10: Det endelige dashbordet 22
Figur 11: Tidlig design av API 26
Figur 12: Skisse av databaseløsning 27
Figur A-1: GANTT-Diagram 34
Figur C-1: Liggende søylediagram for antall tilbakemeldinger per modul 36
Figur C-2: Kort-element som viser antall tilbakemeldinger etter filtrering 36
Figur C-3: Måler-element som viser gjennomsnittlig vurdering 36

Tabelliste

Tabell 1: Kort-element som viser antall tilbakemeldinger etter filtrering 24
Tabell 2: Måler-element som viser gjennomsnittlig vurdering 25
Tabell B-1: Risikoanalyse 35

Ordliste

Porteføljesystem	Programvare for administrasjon og overvåking av investeringer.
HVL	Høgskulen på Vestlandet
Dashbord	Også kalt instrumentbord, er en visuell presentasjon av data og informasjon i form av grafer, tabeller og diagrammer.
Modul	Standardisert grunnelement, byggekloss som vil inngå i en helhet ved oppbygging av forskjellige produkter
FinTech	Forkortelse for Finansiell Teknologi. Brukes om teknologiske løsninger og innovasjoner til å forbedre og automatisere finansielle tjenester.
API / Programvaregrensesnitt	Et sett med protokoller, metoder og verktøy for å bygge og integrere applikasjoner.
Relasjonsdatabase	En database som organiserer data i tabeller som er koblet med relasjoner.
IDE	En IDE (Integrated Development Environment) er en programvareplattform som gir utviklere verktøy for å skrive og teste kode.
Debugge	Prosessen med å finne og fikse feil i programvarekode.
HTTP-Forespørsel	Melding fra en klient til en webserver.
Serialiserer	Omforme data i en datastruktur til enkel tekst eller binært format
Forretningslogikk	Manipuleringer i et dataprogram
Mellomvare	Programvare som ligger mellom systemer
Overhead	Løpende utgift for å drive en virksomhet

1 INNLEDNING

Dette kapittelet gir en innledende introduksjon til prosjektet. Det første delkapittelet tar for seg den overordnede rammen for prosjektet. Videre blir motivasjonen og prosjektets eier presentert. Deretter blir problembeskrivelsen og målsetningen grundig gjennomgått.

1.1 Kontekst

"Statistikk er vitenskapen om innsamling, oppsummering og analysering av data. Data kan være informasjon fra personer, bedrifter eller andre enheter." (Frøslie og Bjørnstad, 2021). Statistikk er vesentlig for å ta innsiktsfulle beslutninger. Bedrifter og organisasjoner samler statistikk for å kunne måle progresjon og ytelse. Deretter blir det identifisert problemer for analyse og prioritering av tiltak.

Stacc Escalis hovedløsning er porteføljesystemet Escali Financials. Dette er en komplett løsning for blant annet administrasjon, forvaltning og regnskapsføring av finansielle instrumenter. Systemet består av ulike integrerte moduler. Modulene dekker områder innenfor gjeld, leasing, obligasjoner, risikomåling og derivater. Kundene av porteføljesystemet har mulighet til å ta i bruk en eller flere moduler. For å få bedre oversikt over bruken av systemet, ønsker Stacc Escali å implementere en programvare som automatiserer innsamling av brukerstatistikk. Som bacheloroppgave skal gruppen utvikle en del av dette systemet.

I oppgavebeskrivelsen står det at gruppen skal lage en applikasjon for brukerstatistikk fra porteføljesystemet. Programvaren skal samle inn statistikk over bruken av ulike moduler fra hovedsystemet. Videre skal det lages en løsning for å presentere brukerstatistikken. Stacc Escali har ikke en slik programvare fra før. Dermed har gruppen ansvar for å utføre undersøkelser, eksperimentere og utvikle en god løsning.

1.2 Motivasjon

Stacc Escali benytter seg i dag av årlige spørreundersøkelser og møter med kunder for å hente inn tilbakemeldinger. Dette begrenser tilgangen til brukerstatistikk og hindrer bedriften i å gjøre kontinuerlige forbedringer av sine systemer.

Applikasjonen for brukerstatistikk skal hente inn og lagre data. Når data er lagret, skal informasjonen presenteres på et dashbord. Resultatet av dette er enklere tilgang på statistikk. Dette kan ledelsen i Stacc Escali bruke som grunnlag for beslutninger og identifisere områder hvor det kan gjøres forbedringer.

1.3 Prosjekteier

Stacc Escali er et programvareselskap som utvikler digitale løsninger innenfor finans og porteføljeforvaltning. Stacc Escali er en av ni bedrifter som er eid av FinTech-konsernet Stacc. Hovedkontoret ligger ved høyteknologisenteret i Bergen. De viktigste markedssegmentene er store kapitalkrevende virksomheter som banker, forsikringsselskaper, pensjonskasser og lignende.

1.4 Problembeskrivelse og mål

1.4.1 Problembeskrivelse

Porteføljesystemet har over hundre forskjellige kunder, der gjeldende praksis for innsamling av brukerstatistikk er en manuell løsning. Som følge av dette tar det lang tid å lage en oversikt som viser bruken av porteføljesystemet. I tillegg foregår innsamlingen av brukerstatistikk sjeldent. Dette fører til følgende problemstilling: Hvordan utvikle et dynamisk system som henter tilbakemeldinger, brukerstatistikk og feilmeldinger, og deretter presentere dette.

Løsningen vil samle informasjon om hvordan ulike moduler brukes. I tillegg skal feilmeldinger og advarsler rapporteres. Ved å samle og visualisere denne informasjonen, vil ledelsen i Stacc Escali få bedre innsikt i bruken av deres systemer. Oppdragsgiver har gitt uttrykk for at behovet for statistikk kan endre seg over tid, så dette må løsningen ta hensyn til.

1.4.2 Mål

Målet for oppgaven er å ha et fungerende sluttprodukt som møter kravene til oppdragsgiver. Dette innebærer et produkt i form av en applikasjon. De viktigste funksjonene vil være å motta og lagre data i en eksisterende database, eid av Stacc Escali. For å få dette til må databasen modifiseres, slik at den kan motta data fra applikasjonen. Oppdragsgiver ønsket at data skulle struktureres i form av en visning (view) og deretter visualiseres på et dashbord.

1.5 Oppbygging av rapporten

Kapittel 1 – Innledning: Innledende presentasjon av prosjektet, beskriver kontekst og motivasjon, og tar for seg problemstilling og mål.

Kapittel 2 – Prosjektbeskrivelse: Beskriver bakgrunnen for prosjektet. Inneholder de initielle kravene, første løsnings-idé til oppdragsgiver, avgrensninger og ressurser gruppen har brukt.

Kapittel 3 – Design av prosjektet: Beskriver teknologier og metoder gruppen har tatt i bruk for å gjennomføre bachelorprosjektet.

Kapittel 4 – Detaljert løsning: Forklarer i detalj hvordan løsningen er satt sammen.

Kapittel 5 – Resultater: Beskriver hvilke evalueringsmetoder som er brukt, samt hvordan resultatene bidrar til å oppnå det overordnede målet.

Kapittel 6 – Diskusjon: Drøfting av tilnærmingene for å oppnå resultater og utførelsen av prosjektet.

Kapittel 7 – Konklusjon og videre arbeid: Oppsummerer de viktigste funnene i prosjektet og diskuterer mulighetene for videre utvikling.

Kapittel 8 – Referanser: Inneholder alle kilder som er brukt i rapporten.

Kapittel 9 – Vedlegg: Inkluderer tilleggsinformasjon som kan være nyttig for leseren.

2 PROSJEKTBESKRIVELSE

Dette kapittelet beskriver bakgrunnen for prosjektet, kravene som ble stilt og de første idéene til gruppen. Kapittelet vil også ta for seg avgrensinger og beskrive ressursene som er tatt i bruk.

2.1 Praktisk bakgrunn

2.1.1 Tidligere arbeid

Tidligere har det ikke blitt utført noe arbeid knyttet til utviklingen av et system som henter, lagrer og visualiserer brukerstatistikk fra hovedsystemet til Stacc Escali. Gruppen fikk tilgang til en relasjonsdatabase av oppdragsgiver for lagring av data.

2.1.2 Initielle krav

Oppgaven hadde et klart mål, men var åpen for forskjellige løsninger. Oppdragsgiver ønsket en applikasjon for å hente brukerstatistikk fra et porteføljesystem og lagre dem i en database. Deretter skulle data bli visualisert. I startfasen ble det stilt følgende funksjonelle krav:

- En måte å hente brukerstatistikk.
- En måte å lagre brukerstatistikken.
- Databasen skal være i stand til å motta brukerstatistikk.
- Lage en visning som strukturerer data.
- Visualisere informasjon på et dashbord.
- En måte å teste løsningen.

2.1.3 Initiell løsnings-idé

Da de initielle kravene ble fastsatt, var det opp til gruppen å bestemme hvilke hjelpemidler som skulle brukes og hvordan systemet skulle utvikles. Etter en planleggingsfase ble det besluttet at gruppen skulle omstrukturere den tildelte databasen og utvikle et programvaregrensesnitt (API). Programvaregrensesnittet skulle være en generell mottaker av data. Deretter skulle data bli strukturert som en visning og visualisert på et dashbord ved hjelp av en programvare.

Idéen er å legge til rette for at kunder i hovedsystemet kan gi tilbakemelding og vurdere sin tilfredshet på en enkel måte.

Hovedsystemet sender data til programvaregrensesnittet som videresender til databasen. Funksjonaliteten i hovedsystemet, som programvaregrensesnittet er avhengig av, er ikke ferdig utviklet. Dermed må gruppen ta i bruk et testverktøy som simulerer en klient.

For lagring, strukturering og visualisering var det behov for en mer dynamisk database. Oppdragsgiver ønsket at den tildelte databasen skulle modifiseres, slik at den kunne ta imot korrekte data fra programvaregrensesnittet. All brukerstatistikk som hentes blir lagret her.

2.2 Avgrensninger

I oppgavebeskrivelsen sto det oppført hva gruppen skulle gjøre og målsetningen for prosjektet. Etter et møte med oppdragsgiver var det naturlig å lage en prioritering over de forskjellige kravene løsningen skulle inneholde. Dermed ble omstrukturering av databasen og utvikling av API-et prioritert før visualisering av data.

Det var noen tekniske avgrensninger gruppen måtte forholde seg til. Databasen som ble tildelt var en relasjonsdatabase og det var bestemt at dette skulle være lagringsplassen for brukerstatistikk. Modifikasjonene som skulle gjøres i databasen måtte også bli godkjent av oppdragsgiver. For kodeutveksling var det ønskelig at gruppen brukte GitHub som oppbevaringssted. I tillegg var IDE, programmeringsspråk og rammeverket for API-et forhåndsbestemt.

Gruppen hadde noe frihet til å utforme systemet selv. Brukertilfredshetsfunksjonen som løsningen skulle benytte seg av var ikke ferdigutviklet i hovedsystemet. Dermed var det opp til gruppen å velge et verktøy som kunne bruke ressursene og teste funksjonaliteten til programvaregrensesnittet. I tillegg måtte gruppen velge en arkitektur og et designmønster for å gjøre API-et dynamisk. Det var også nødvendig å velge en programvare som kunne importere og presentere data.

2.3 Ressurser

For utvikling av applikasjon for brukerstatistikk fra porteføljesystemet, er gruppen utstyrt med egne datamaskiner. Applikasjonen skal utvikles i et Microsoft-utviklingsmiljø. Dermed var det ønskelig fra oppdragsgiver at gruppen hadde Windows som operativsystem.

Gruppen har hatt tilgang på interne og eksterne ressurspersoner under prosjektet. Intern veileder var Sven-Olai Høyland som er førsteamanuensis ved HVL. Høyland bisto i veiledning av rapporten og de tilhørende vedleggene. Ekstern veileder var Gerdt Sverre Vedeler, forretningsutvikler og grunnlegger av Escali. Vedeler utviklet den første versjonen av Escali Financials og innehar mye kunnskap innenfor FinTech-bransjen. I tillegg har Sondre Knutsen, utvikler i Stacc Escali, bistått gruppen med verdifull kompetanse og teknisk støtte. Gruppen hadde lite kjennskap til de ulike teknologiene som skulle brukes. Dermed var det en stor fordel å ha en erfaren ressursperson tilgjengelig.

Under prosjektet har gruppen arbeidet på kontorene til Stacc Escali. Hovedgrunnen var at den tildelte databasen kun var tilgjengelig på det interne nettverket. I tillegg var det enklere tilgang på ressurspersoner som hadde faglig innsikt i gruppens prosjekt. Dermed var det lettere og mer gunstig å arbeide i lokalene til oppdragsgiver, enn å arbeide på høyskolen. I tillegg fikk gruppen tildelt kontorplass, tilgang på pc-skjermer og gratis lunsj i kantinen sammen med resten av Escali-teamet. Møterommene i lokalet har blitt brukt til statusmøter med både intern- og ekstern veileder.

Kommunikasjon innad gruppen har foregått stort sett på Facebook Messenger og Discord. Mellom gruppen og ressurspersoner har fysiske møter og epost blitt benyttet. Canvas ble hovedsakelig brukt for kommunikasjon med intern veileder. Canvas er en digital læringsplattform og informasjonskanal som ble brukt til å sende meldinger til forelesere og intern veileder. GitHub ble brukt for kodeutveksling og OneDrive ble tatt i bruk for oppbevaring og deling av filer. OneDrive er en skytjeneste som kobler gruppen til alle felles filer (Microsoft, u. å, a). Skytjenesten beskytter filene og gjør det enkelt å dele filer med andre innad i gruppen.

3 DESIGN AV PROSJEKTET

3.1 Forslag til løsning

Oppgaven stiller krav til en applikasjon for henting av brukerstatistikk. Det trengs dermed en generell mottaker av data i form av et API. Videre skulle brukerstatistikken lagres i den tildelte relasjonsdatabasen. Til slutt skulle data bli visualisert i en programvare. For å opprette og lagre ressurser er det nødvendig å ta i bruk et testverktøy som simulerer en klient. Det er flere slike verktøy som kan brukes. Videre i kapittelet diskuteres verktøy for å teste API-et for brukerstatistikk. Deretter diskuteres valg av programvare for datavisualisering.

3.1.1 Alternative løsninger for testverktøy

Postman

Postman er en API-plattform for å teste og debugge programvaregrensesnitt. En API-plattform er en programvare som hjelper utviklere i å ta korrekte beslutninger under utviklingsfasene og forenkler utviklingssyklusen (Postman, 2021). Verktøyet passer utmerket for å lage og teste API-forespørsler. Det tilbyr en klient med et brukervennlig grensesnitt, i tillegg er det gratis og åpen kildekode.

Swagger

Swagger er et rammeverk som tilbyr ulike API-utviklerverktøy og er utviklet av SmartBear Software (SmartBear, u. å.). Rammeverket gjør det enkelt å legge til API-dokumentasjon som en veiledning i hvordan programvaregrensesnittet skal bli brukt. Videre tilbyr verktøyet et interaktivt grensesnitt som kategoriserer og organiserer metoder. Verktøyet er brukervennlig og tillater utviklere og sluttbrukere å teste og eksperimentere med ressursene til API-et. Det er lett å ta i bruk og krever lite kode og vedlikehold. Kildekoden er åpen, og det er gratis å ta i bruk.

Diskusjon av testverktøy for API

Både Swagger og Postman er gode verktøy som er gratis å bruke. Begge kan teste programvaregrensesnittet på en god måte og har sine sterke og svake sider. Ingen av deltakere på gruppen hadde benyttet seg av slike API-testverktøy tidligere. De viktigste egenskapene til testverktøyet er å sende test-forespørsler og utnytte ressursene til programvaregrensesnittet. Postman er enkelt å sette opp og egner seg for å sende og motta enkle HTTP-forespørsler. Begge verktøyene har et brukervennlig grensesnitt for sending av forespørsler. En ulempe med Postman er at man ikke kan legge til API-dokumentasjon. Dermed blir testing mindre brukervennlig for testpersoner som ikke har en teknisk bakgrunn. Postman er en desktop-applikasjon og må lastes ned på egen maskin for å teste programvaregrensesnittet.

Swagger er enkelt å sette opp og verktøyet blir en del av API-et. Dermed slipper testpersoner å laste ned eksterne verktøy for å teste systemet. Brukergrensesnittet organiserer og kategoriserer metoder og egner seg godt for å utnytte ressursene til programvaregrensesnittet. I tillegg er det også mulig å legge til dokumentasjon for programvaregrensesnittet. Dermed blir testingen brukervennlig for testpersoner uten teknisk bakgrunn.

3.1.2 Alternative løsninger for visualisering av data

Power BI

Power BI er en skybasert forretningsanalysetjeneste levert av Microsoft. Tjenesten gir Windows brukere mulighet til å enkelt koble til og analysere data fra en rekke kilder (Microsoft, u. å, b). Eksempler på dette er Excel-regneark, SQL Server-databaser og skytjenester som Azure. Brukere kan opprette interaktive dashbord og rapporter ved å dra og slippe ulike visuelle elementer på en arbeidsflate. Power BI har en gratis- og en betalt versjon. Premium versjonen tilbyr flere avanserte funksjoner, inkludert maskinlæring og prediktiv analyse. Rapportene kan deretter deles og publiseres på nettet eller i et internt nettverk.

Looker Studio

Looker Studio er en skybasert rapporterings- og visualiseringsplattform som tilbys av Google. Looker Studio gjør det mulig for brukere å koble til og analysere data fra ulike kilder, inkludert SQL-databaser (Google, u. å.). Fordeler med Looker Studio er at det er gratis å bruke, og det finnes ingen begrensing på antall rapporter som kan opprettes. Rapportene kan på lik linje med Power BI deles på nett, eller i et privat nettverk.

Diskusjon av programvare for visualisering

Både Power BI og Looker Studio er gode løsninger som tilbyr funksjonene som trengs. Det er nødvendig at verktøyet støtter Microsoft SQL Server som datakilde, noe begge gjør. Power BI er en del av Microsofts økosystem og kan integreres med andre Microsoft-verktøy. Eksempler på dette er Teams og Microsoft Azure. Dette er tjenester som er benyttet av Stacc Escali. Integrasjonen med Google-økosystemet Looker Studio tilbyr, er ikke særlig relevant da Stacc Escali ikke bruker verktøyene i Google-økosystemet.

Noen av ulempene med Power BI er at det har en brattere læringskurve, og de mest avanserte funksjonene er bak en betalingsmur. I tillegg er brukerne nødt til å ha en Windows maskin for å redigere dashbordet. Maskiner med andre operativsystemer, kan kun se rapporten på Power BI sin web-applikasjon.

3.2 Valgt løsning

Testverktøy

Valg av testverktøy endte på Swagger. Dette skyldes i stor grad at gruppen utvikler systemet i et utviklingsmiljø som forenkler bruken av verktøyet. En klar fordel med Swagger er at verktøyet har god API-dokumentasjon og tillater brukere å interagere med operasjonen til programvaren. Verktøyet vil gjøre det enklere å teste ressursene til systemet. En annen fordel er at verktøyet blir kjørt sammen med programvaregrensesnittet. Dermed slipper brukerne å laste ned eksterne programmer for testing.

Visualiseringsverktøy

Når det gjaldt visualisering av data, valgte gruppen Power BI. Dette skyldtes hovedsakelig anbefaling fra oppdragsgiver. Det var ønskelig at dashbordet kunne bli integrert med andre Microsoft-verktøy. Det var også flere ressurspersoner som hadde erfaring med Power BI.

3.3 Valg av andre verktøy

Det er ingen tidligere applikasjon for henting og lagring av brukerstatistikk i systemene til Stacc Escali. Det som finnes fra før, er den tildelte relasjonsdatabasen. Det er vesentlig at verktøyene som tas i bruk er kompatible med det nåværende systemet. Her er de viktigste verktøyene som bli benyttet under prosjektet:

- Visual Studio som IDE for programmering av API
- C# og .Net som programmeringsspråk og rammeverk
- Server for å kjøre programvaregrensesnittet
- Microsoft SQL relasjonsdatabase for lagring av brukerstatistikk
- Power BI – Programvare for å visualisere brukerstatistikk
- Swagger for testing av systemet

Programvaren ble utviklet i C# og .Net der Visual Studio ble brukt som utviklingsmiljø. Rammeverket .Net tilbyr utviklerverktøy og en plattform for å bygge og kjøre applikasjoner (Microsoft, u. å, c). Visual studio tilbyr også en integrert web-server for å kjøre selve programvaregrensesnittet. De resterende verktøyene blir grundigere forklart i Kapittel 4.

3.4 Prosjektmetodikk

3.4.1 Utviklingsmetodikk

Scrum

Gruppen valgte å benytte seg av Scrum som en smidig utviklingsmetode. Scrum er et enkelt rammeverk for å optimalisere produktutvikling – i utgangspunktet programvarebaserte produkter (Glasspaper, u. å.). Rammeverket bygger ikke på konkrete metodikker eller teknikker, men legger til rette for en fleksibel utviklingsprosess. Scrum fokuserer på utvikling og levering av produkter i korte iterasjoner, også kjent som sprinter. Hver sprint inneholder en rekke oppgaver som skal fullføres innen sprintens avslutning. Dette gjør det mulig for gruppen å fokusere trinnvis på funksjonene til produktet og foreta nødvendige justeringer underveis.

Scrum er en tilpasningsdyktig metode som legger til rette for regelmessig kommunikasjon og samarbeid mellom gruppen og oppdragsgiver. Ukentlige fysiske statusmøter med oppdragsgiver tilrettelegger evaluering av fremdriften og tilpassing av planene etter behov. Gruppen har også korte interne møter i forkant av de ukentlige møtene for å sikre identifisering av eventuelle problemer som må tas opp. Dette gjør Scrum til et effektivt rammeverk for å opprettholde et organisert og effektivt teamarbeid.

Kanban

Kanban er en prosjektstyringsmetode der hovedformålet er å optimalisere flyten av arbeidet gjennom prosjektet. Kanban bruker et visuelt styresystem for å gi gruppen en oversikt over arbeidet som må gjøres, hva som foregår underveis og hva som er ferdigstilt (Kanbanize, u. å.).

Gruppen har valgt å benytte seg av Kanban i form av en tjeneste som leveres av Trello. Trello tilbyr enkel og fleksibel tilgang til Kanban-styring, og gjør det mulig for gruppen å legge til og endre oppgaver.

Gruppen har tre forskjellige Kanban-tavler: Rapport, utvikling og EXPO. Tavlene blir delt opp i tre kategorier: "Gjøremål", "Pågår" og "Ferdig". Hver oppgave som må gjøres blir plassert i kategorien "Gjøremål", og deretter flyttet til "Pågår" når arbeidet påbegynnes. Når oppgaven er fullført, flyttes den til kategorien "Ferdig". Ved å bruke Kanban på denne måten kan gruppen håndtere eventuelle forsinkelser eller problemer som kan oppstå. Dermed opprettholdes en jevn og effektiv flyt i prosjektet.

3.4.2 Prosjektplan

For å planlegge prosjektet har gruppen valgt å benytte seg av et GANTT-diagram. Et GANTT-diagram er en visuell representasjon av prosjektplanen, og gir en oversikt over tidslinjen til de ulike oppgavene som må fullføres (Gantt.com, u. å.). Hver oppgave blir representert som en horisontal linje, hvor start- og sluttdato for oppgaven er markert. Det er også et felt for hvor mange timer som er påtenkt til oppgaven. Gruppen har valgt å strukturere GANTT-diagrammet etter ulike faser i prosjektet. I tillegg er det lagt inn milepæler for å markere viktige tidspunkter og innleveringer. Ved å bruke GANTT-diagrammet på denne måten kan gruppen enkelt identifisere flaskehalser. Dermed er det mulig å tilpasse planene etter behov for å sikre at prosjektet blir levert innenfor den avtalte tidsrammen. GANTT-diagrammet er vedlagt som Vedlegg A i rapporten.

3.4.3 Risikoanalyse

En risikoanalyse blir definert som en studie av risiko for å få innsikt i hva slags hendelser som kan skje, hvorfor de kan skje og hva konsekvensene kan være (Aven, 2017).
I prosjekthåndboken finnes det en risikoanalyse som tar for seg hendelser som kan påvirke prosjektet negativt. Den er også vedlagt som Vedlegg B i rapporten.

I risikoanalysen får hver risiko en verdi for sannsynlighet og konsekvens fra en til fem. Produktet av sannsynlighet og konsekvens utgjør risikovurderingen for en enkelt risiko. I analysen er det vurdert overordnede risikoer som kan oppstå i forbindelse med et gruppeprosjekt med varighet på et halvt år. I tillegg har hver risiko et felt som er ment å foreslå tiltak for å forhindre at risikoene oppstår. Den risikoen med høyest risikoprodukt er vurdert til å være begrenset arbeidskapasitet.

Prosjekthåndboken inneholder også ukentlige statusrapporter som gir en oppdatert oversikt over prosjektets fremgang. Disse rapportene gir en detaljert beskrivelse av spesifikke risikoer og problemer som har oppstått i løpet av prosjektet.

3.5 Evalueringsplan

For å evaluere et produkt er det nødvendig med verifisering og validering. Verifisering handler om å sikre at produktet fungerer som det skal i henhold til de funksjonelle kravene. Validering går ut på om gruppen har utviklet et produkt som møter de definerte kravene. Det blir brukt forskjellige metoder for å evaluere prosjektet.

For verifisering blir det utført enhetstesting, integrasjonstesting og systemtesting. Enhetstesting vil undersøke individuelle biter av programvaren, mens integrasjonstesting tester hvordan de forskjellige delene i systemet fungerer sammen. Til slutt vil systemtesting teste samspillet mellom alle delene for å sikre at alt fungerer.

For validering har gruppen hatt ukentlige Scrum møter med oppdragsgiver som sikret at produkt-idéen var riktig. I tillegg skulle ledelsen i Stacc Escali utføre en brukertest av det endelige produktet etter verifiseringen.

Som resultat får man en grundig evaluering av sluttproduktet. Dermed blir eventuelle feil og mangler identifisert, slik at man kan gjøre nødvendige forbedringer. Dette sikrer at produktet oppfyller brukerens behov og krav.

4 DETALJERT LØSNING

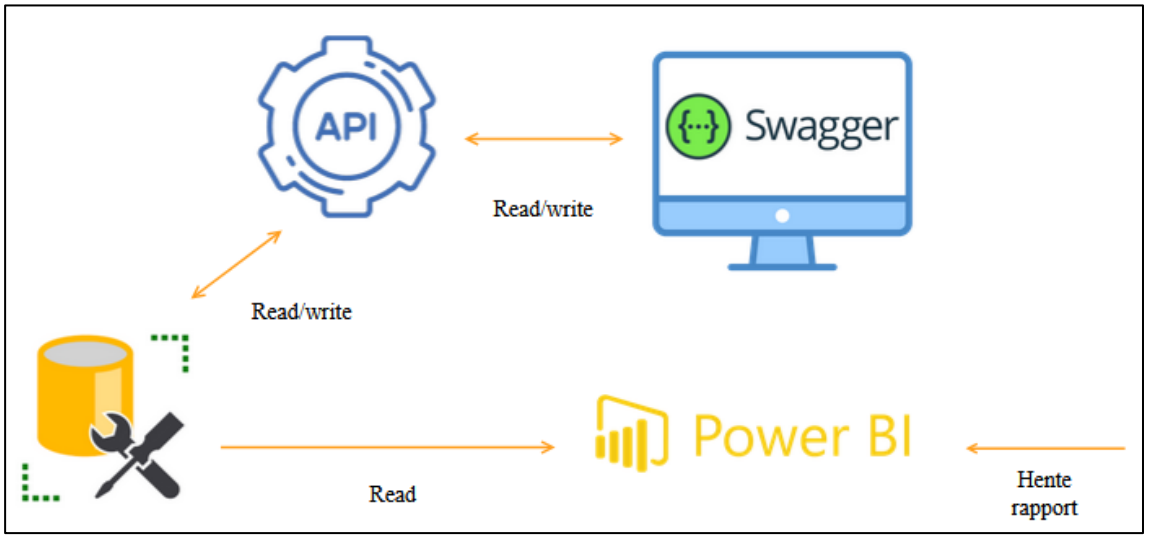
Programvaregrensesnittet er utviklet med tanke på at det skal være dynamisk, enkelt å ta i bruk og kompatibelt med porteføljesystemet til Stacc Escali. Systemet kjører på localhost ved hjelp av en server som er integrert med API-et. Det kan kjøres i terminalen ved en enkel kommando. Programvaren består av en pakke som har ett sett med filer. Ved å laste ned pakken har man alt som trengs for å kjøre programvaregrensesnittet på egen maskin (så lenge .Net er installert). Pakken er tilgjengelig på gruppens GitHub arkiv.

Dashbordet er laget for å gi brukeren muligheter for datavisning. Brukere skal enkelt kunne filtrere data på ulike parametere som for eksempel kunde eller dato. Den endelige rapporten lagres som en Power BI fil. Denne filen kan åpnes i Power BIs web-applikasjon, eller lastes ned i Power BI programvaren.

Dette kapittelet gir en fordypning i de ulike delene som utgjør systemet for innsamling av brukerstatistikk og hvilken rolle de har. Videre vil de ulike delene som utgjør systemet bli referert til som instanser.

4.1 Systemarkitektur

Systemet er satt sammen av tre separate instanser som gjør hver sin oppgave. API-et, databasen og Power BI rapporten gjør hver sin del. Testverktøyet Swagger vil simulere en klient.



Figur 1: Sammenhengen mellom de ulike instansene

Programvaregrensesnittet fungerer som en generell mottaker av data. API-et har to ulike bruksområder: henting og lagring av data. Data som er assosiert med en tilbakemelding kan bli hentet ved hjelp testverktøyet. Et eksempel er aktive kunder. Dette er informasjon som blir brukt som parameter for å sende en tilbakemelding. Sending av en tilbakemelding blir også gjort i testverktøyet. Programvaregrensesnittet tar imot tilbakemeldingen og lagrer det i databasen. Ressursene til programvaregrensesnittet blir utnyttet ved hjelp av testverktøyet.

Visningen inneholder den viktigste informasjon for brukerstatistikken. Dette inkluderer kunde, hovedkategori, underkategori, modul, vurdering og kommentar.

Videre blir visningen importert i Power BI, og et dashbord opprettes. Dette kan lastes ned og redigeres av brukere, eller leses med lesetilgang gjennom Power BIs web-applikasjon. I web-applikasjonen kan brukere med administratorrettigheter oppdatere dashbordets underliggende data.

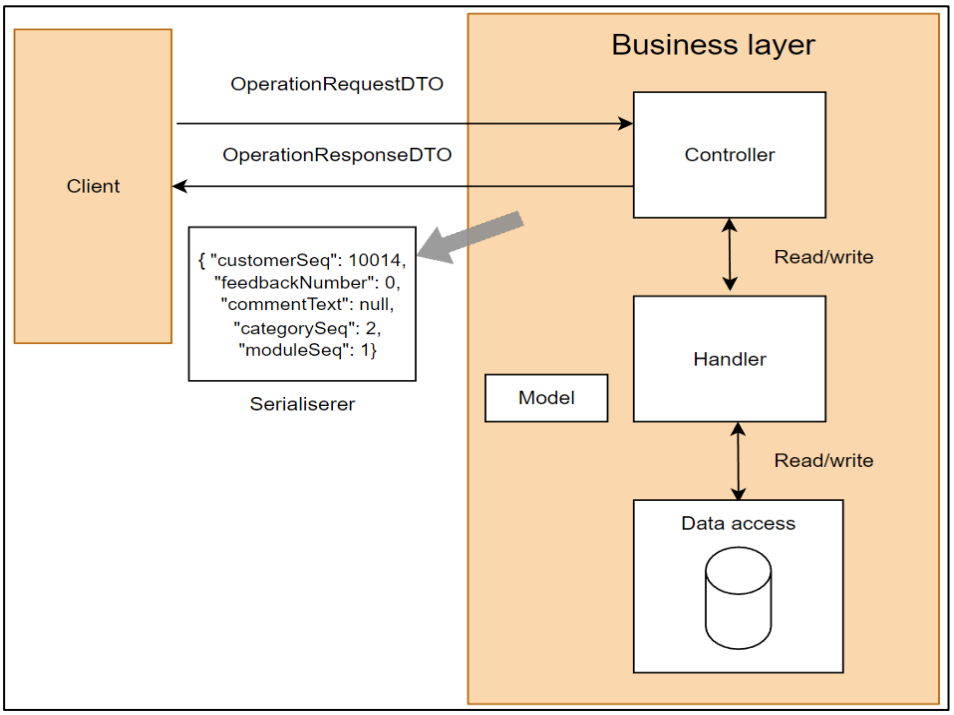
4.2 API for brukerstatistikk

Representational State Transfer (REST) er en arkitektur som sørger for en løs kobling mellom to tjenester i en integrasjon (Microsoft, 2023, a). Kommunikasjonsarkitekturen gjør det mulig for en klient å lese og manipulere ressurser til API-et med HTTP som kommunikasjonsprotokoll. REST følger et sett med prinsipper som begrenser hvordan applikasjonen kan prosessere informasjon fra en klient. De viktigste prinsippene som ble brukt er følgende:

- **Separering av tjener og klient:** Sikkerhet, databasetilgang og forretningslogikk er programvaregrensesnittets ansvarsområde. Klienten er ansvarlig for å opprette brukergrensesnittet.
- **Uniformt grensesnitt:** Ressursene i programvaregrensesnittet kan identifiseres ved hjelp av en unik URI. For å manipulere ressurser blir metodene i kommunikasjonsprotokollen brukt. Oppretting av ressurser bruker POST og henting bruker GET.
- **Kan mellomlagres:** Alle GET-forespørsler fra programvaregrensesnittet blir mellomlagret midlertidig på klientsiden.
- **Tilstandsløs:** Programvaregrensesnittet og klient trenger ikke kjenne hverandres kontekst. All nødvendig informasjon blir sendt eller hentet til og fra klienten.
- **Lagdeling:** Det er mulig å ha flere lag mellom programvaregrensesnittet og klienten.

Ved å følge disse prinsippene blir programvaregrensesnittet mer pålitelig, skalerbart, portabelt og yter bedre (Codecadamy, u. å.).

Model-View-Controller er brukt som designmønster for å organisere kildekoden. Designmønsteret deler problemet inn i komponenter som kan løses uavhengig av hverandre (Smith, 2022). Dermed blir programvaren enklere å videreutvikle og vedlikeholde. Videre blir de forskjellige komponentene som utgjør programvaregrensesnittet beskrevet.



Figur 2: Design av API

4.2.1 Model

En modell er en klasse som representerer data som skal bli behandlet, samt en entitet fra databasen (Anderson og Larkin, 2023). Programvaregrensesnittet har fire ulike modeller som er nødvendige for å hente og lagre tilbakemeldinger. En tilbakemelding består av ulike attributter fra modellene.

4.2.2 Handlers

Denne komponenten er en hjelpeklasse som håndterer lese- og skrive-operasjoner til og fra databasen. For hver modell finnes det en handler. Klassen fungerer som et mellomledd mellom databasen og kontrolleren. Dermed separeres databaseoperasjoner fra resten av programvaren. Strukturen blir ryddigere og mer lesbar. Det blir også enklere å teste logikken i andre komponenter når databaseoperasjon er adskilt fra resten av programmet.

4.2.3 Controllers

I programvaregrensesnittet er dette en komponent som håndterer HTTP-forespørsler og alle brukerinteraksjoner (Smith, 2022). Hver enkel kontroller er også et eget endepunkt. De offentlige metodene i en slik klasse håndterer forretningslogikken til API-et og kalles handlingsmetoder. Når programvaregrensesnittet får en forespørsel fra en klient, blir forespørselen dirigert til en handlingsmetode. Metodene tar i bruk HTTP-verbene POST og GET. Disse bestemmer hva metoden skal gjøre. GET-metodene henter data ut fra databasen og sender til klienten. POST-metoden oppretter og lagrer nye tilbakemeldinger i databasen. Tilbakemeldinger sendes fra testverktøyet som et JSON objekt. Det inneholder informasjon om hvilken kunde som sendte tilbakemeldingen, hvilken modul og kategori det er gitt tilbakemelding på, vurdering av fornøydhet og kommentar.

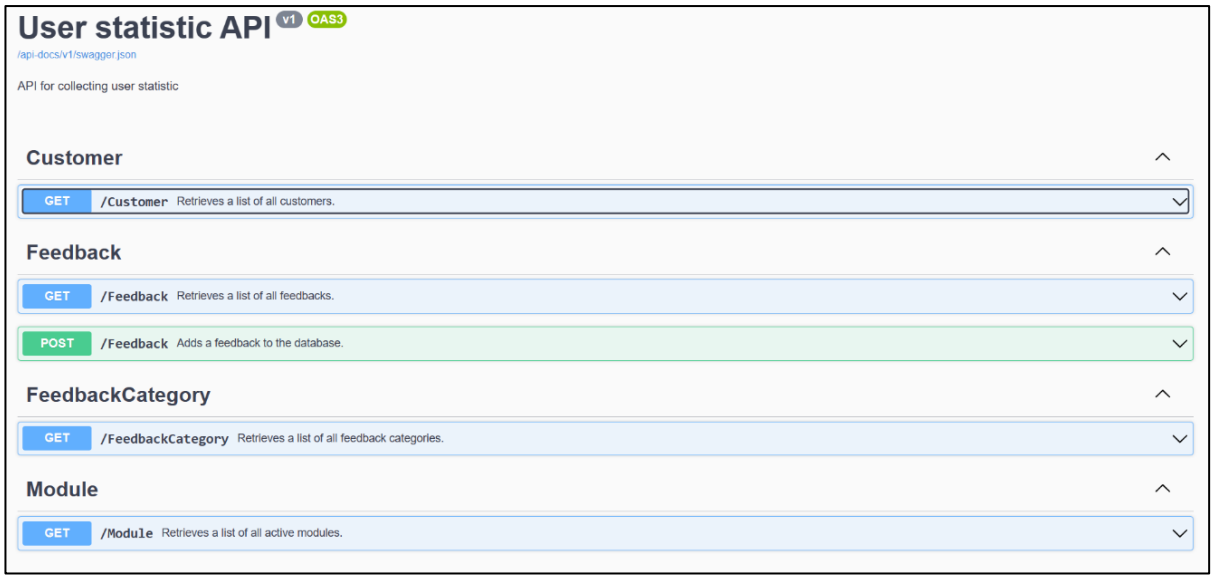
GET-metodene serialiserer returobjektet om til JSON og POST-metodene deserialiserer tilbakemeldingen fra JSON automatisk. Videre blir en responskode opprettet. Responskodene er: 200 OK, 400 Bad Request og 500 Internal Server Error. Den vanligste responskoden er 200 OK, som indikerer at en respons lyktes.

4.2.4 Data transfer object (DTO)

Før data bli sendt over til en klient er det ønskelig å forme data. Dataoverføringsobjekter er et objekt som definerer hvordan data vil bli sendt over nettverket. Hver modell har en DTO som skjuler attributter klienten ikke skal se (Anderson, 2022). DTO er også nyttig når man må gruppere verdier som skal sendes. Eksempler på attributter som blir fjernet i en DTO er unike ID-er eller sensitiv informasjon. Dataoverføringsobjektene isolerer modellene fra presentasjonsnivået. Dermed mottar klienten data som ikke er direkte tilknyttet databasen. Dette resulterer i en løs kobling (Esposito, 2009).

4.2.5 Swagger

For å legge til Swagger i programvaregrensesnittet var det nødvendig å laste ned pakken Swashbuckle. Swashbuckle er en implementasjon av Swagger brukt for å enkelt ta i bruk Swagger verktøyet for .Net (Microsoft, u. å, d). Pakken blir enkelt lastet ned i utviklingsmiljøet og importert i start-filen til applikasjonen. Når programvaren blir kjørt vil verktøyet opprette et dokument i nettleseren som tillater at brukeren interagerer med ressursene til programvaregrensesnittet. Det interaktive grensesnittet simulerer en klient (se Figur 3).



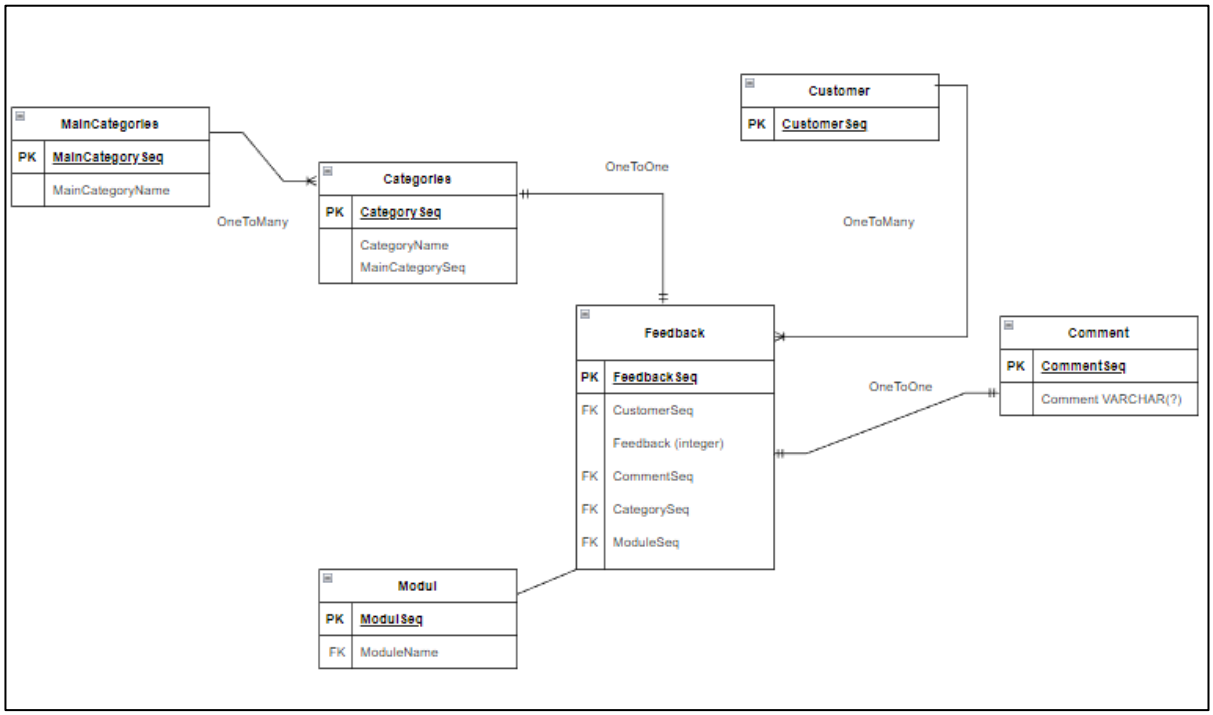
Figur 3: Swaggers interaktive grensesnitt

4.3 Databaseløsning

4.3.1 Omstrukturering av database

For langtidslagring av brukerstatistikk fikk gruppen tilgang til en Microsoft SQL database, eid av Stacc Escali. Denne databasen brukes i eksisterende applikasjoner. Databasen er stor og inneholder flere tabeller med ulike relasjoner. Det var nødvendig å legge til nye entiteter, slik at databasen kunne lagre tilbakemeldinger. Disse mottas fra API-et. De nye endringene skulle være fleksible og ta høyde for videre arbeid.

Det ble lagt til fire nye entiteter i databasen: hovedkategori, underkategori, tilbakemelding og kommentar. Hovedkategori og underkategori tar for seg de kategoriene en bruker skal gi tilbakemelding på. Et eksempel på en hovedkategori er feilmeldinger. Kritiske feil og advarsler er eksempler på underkategorier som tilhører feilmeldinger. Tilbakemeldingstabellene inneholder informasjon om hvilke kunde, modul, kommentar og kategori som er assosiert med en tilbakemelding. Kommentartabellen inneholder kommentarene som tilhører tilbakemeldingen.



Figur 4: ER-Modell av databaseløsning

4.3.2 Visning

En visning er en delmengde av en database og er basert på en spørring fra én eller flere tabeller [(IBM, 2021).](#) For å opprette visningen ble kolonnene fra ulike tabeller kombinert ved hjelp av nøkkelordet 'INNER JOIN'. På denne måten velges rader med like verdier i begge tabellene. Det er nødvendig med en relasjon mellom tabellene for at dette skal fungere.

```
CREATE VIEW [dbo].[vwFeedbackInformation]
AS
SELECT
    cu.CustomerName, mfc.MainFeedbackCategoryName, fc.FeedbackCategoryName, f.FeedbackNumber, m.ModuleName, c.CommentText, f.FeedbackSince
FROM
    dbo.Feedbacks AS f
    INNER JOIN dbo.Comments AS c                    ON c.CommentSeq = f.CommentSeq
    INNER JOIN dbo.Customers AS cu                  ON cu.CustomerSeq = f.CustomerSeq
    INNER JOIN dbo.FeedbackCategories AS fc         ON fc.FeedbackCategorySeq = f.CategorySeq
    INNER JOIN dbo.MainFeedbackCategories AS mfc    ON mfc.MainFeedbackCategorySeq = fc.MainFeedbackCategorySeq
    INNER JOIN dbo.Modules AS m                     ON m.ModuleSeq = f.ModuleSeq
```

Figur 5: Første del av spørringen henter ut relevante kolonner for kundetilbakemeldinger

Spørringen vist i figur 5 henter ut alle relevante kolonner som trengs for implementering av dashbordet. For å øke brukervennligheten skal alle kunder og datoer være i denne visningen. UNION-nøkkelordet ble brukt i spørringen for å oppnå dette. Nøkkelordet kombinerer resultater fra flere spørringer. Dermed er det mulig å velge en spesifikk kunde og dato, selv om de ikke er assosiert med en tilbakemelding. Radene som kun inkluderer kunder, datoer, kategorier og moduler kan enkelt filtreres bort når man teller faktiske tilbakemeldinger. Dette gir en mer omfattende visning av data og forbedrer brukervennligheten uten å påvirke ytelsen til databasen. Dette er fordi en visning er en virtuell tabell, noe som betyr at den ikke opptar plass i databasen (Microsoft, 2023, b).

Figur 5 kombineres med Figur 6 og danner den komplette spørringen for å lage visningen.

```
UNION

SELECT
    C.CustomerName, '' AS MainFeedbackCategoryName, '' AS FeedbackCategoryName, 0 AS FeedbackNumber,
    ''  AS ModuleName, '' AS CommentText,  CA.CalendarDate AS FeedbackSince
FROM
    Customers C
    CROSS JOIN Calendar CA
WHERE
    CA.CalendarDate <= getdate() AND CA.CalendarDate >= '1/1/2023'

UNION

SELECT
    '' AS CustomerName,  MFC.MainFeedbackCategoryName, FC.FeedbackCategoryName, 0 AS FeedbackNumber,
    ''  AS ModuleName, '' AS CommentText,  CA.CalendarDate AS FeedbackSince
FROM
    MainFeedbackCategories MFC
    INNER JOIN FeedbackCategories FC ON FC.MainFeedbackCategorySeq = MFC.MainFeedbackCategorySeq
    CROSS JOIN Calendar CA
WHERE
    CA.CalendarDate <= getdate() AND CA.CalendarDate >= '1/1/2023'


UNION

SELECT
    '' AS CustomerName,  '' AS MainFeedbackCategoryName, '' AS FeedbackCategoryName, 0 AS FeedbackNumber,
    M.ModuleName, '' AS CommentText,  CA.CalendarDate AS FeedbackSince
FROM
    Modules M
    CROSS JOIN Calendar CA
WHERE
    CA.CalendarDate <= getdate() AND CA.CalendarDate >= '1/1/2023'
```

Figur 6: Andre del av spørringen bruker union for mer omfattende datavisning

4.4 Visualisering av data

4.4.1 Innhenting av data

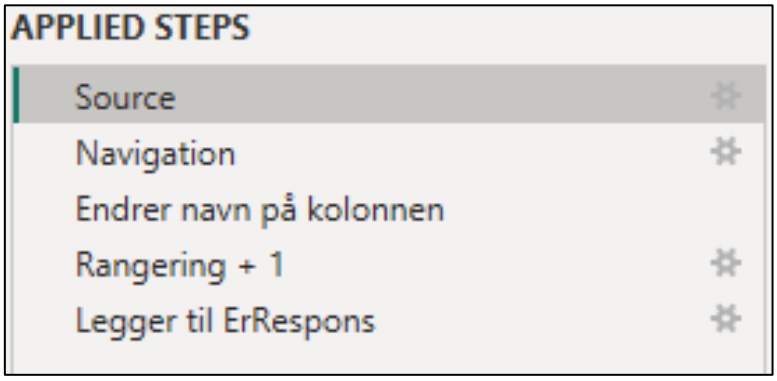
Som beskrevet i Kapittel 4.3 ble en visning opprettet i databasen. Power BI ble gitt tilgang til databasen og visningen ble valgt for importering. Visningen ble deretter transformert til et Power BI datasett. Datasettet må oppdateres manuelt for å hente endringer fra de underliggende datakildene.

I dette kapittelet brukes ordet 'tilbakemelding' til en kundes respons. Dette bør ikke forveksles med hovedkategorien tilbakemeldinger. Ordet respons brukes på dashbordet for å unngå denne forvekslingen, etter ønske fra oppdragsgiver. Videre i kapittelet vil det også være figurer som inneholder falske data som kun er ment til illustrasjonsformål.

4.4.2 Behandling av data

Etter visningen har blitt importert og transformert til et Power BI datasett, blir det behandlet, slik at det er mer lesbart for brukeren. Et eksempel er navngiving av kolonner. Navnet på kolonnen vil i noen tilfeller bli vist til brukeren. Dermed vil det være bedre å bruke et konsist norsk ord. Et eksempel er at kolonnen CustomerName endrer navn til Kunde. Dette ble gjort gjennom Power Query funksjonen. Denne funksjonen gjør at datasettet følger et fast sett med regler hver gang det lastes inn på nytt. Power Query ble også brukt til å legge til én til alle verdiene i vurderings-kolonnen. Et vurderingssystem fra 1 til 5 er mer intuitivt enn fra 0 til 4.

Ettersom datasettet inneholdt flere rader som ikke var assosiert med en tilbakemelding, ble en ny kolonne opprettet kalt ErRespons. Denne kolonnen er binær, der verdien 1 blir gitt dersom raden er en tilbakemelding.



Figur 7: Stegene som blir gjort i Power Query

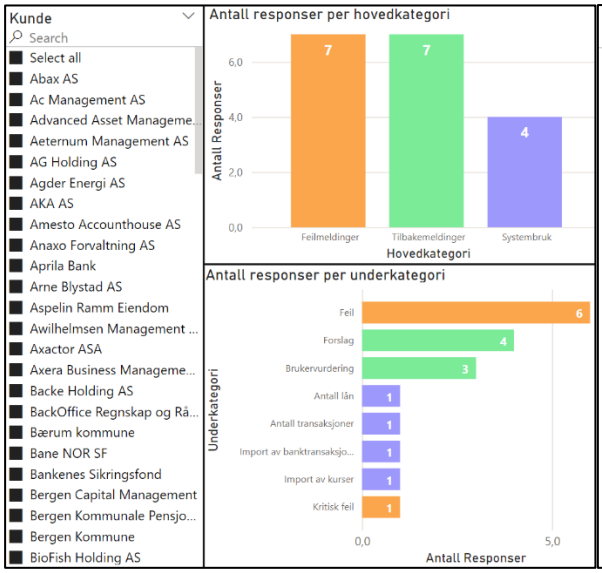
4.4.3 Presentasjon av data

Dashbordet er en sammensetning av diagrammer og grafer, som representerer datasettet. Tilbakemeldinger skal kunne filtreres på kunder og datoer.

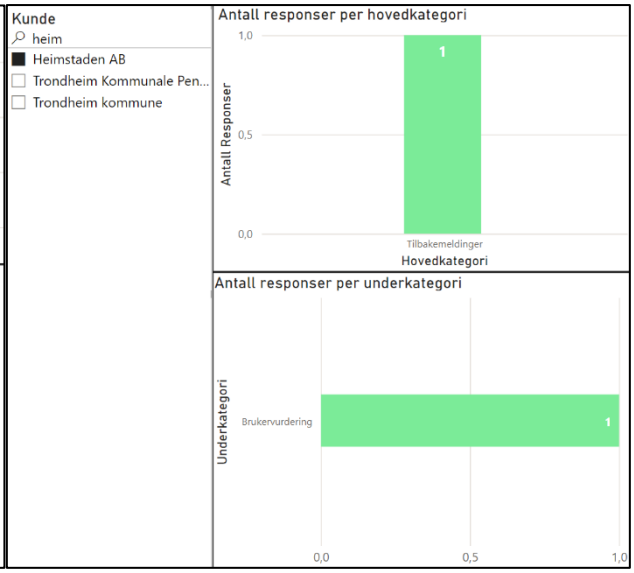
For å filtrere tilbakemeldinger etter kunder, ble verktøyet Slicer brukt. Slicer lar brukere filtrere hvilke elementer som skal vises [(Microsoft, 2023, c).](#) Kundelisten inneholder et stort antall elementer, derfor måtte sliceren være kompakt. En vertikal liste hadde kapasitet til å vise flest navn samtidig, og ble valgt for kundefiltrering. Videre ble en søkefunksjon inkludert for å få enkel tilgang til en spesifikk kunde (se Figur 9).

Det ble lagt til diagrammer som viser antall tilbakemeldinger fordelt i ulike hovedkategorier, underkategorier og moduler. Diagrammet for tilbakemeldinger per hovedkategori ble vist i et stående søylediagram. Underkategorier og moduler hadde flere elementer. For å forbedre lesbarheten til disse ble de vist i et liggende søylediagram.

For å tydeliggjøre forholdet mellom hovedkategorier og underkategorier, ble underkategoriene farget i samme farge som hovedkategorien de tilhører. Antall tilbakemeldinger per modul ble farget i en annen farge for å unngå forvirringer. Diagrammene ble deretter koblet til kunde-sliceren. Dermed kan brukeren filtrere på en eller flere kunder og diagrammene vil oppdatere seg. Diagrammene fungerer også som et filter. Ved å klikke på en søyle, vil de andre visuelle elementene vise informasjon som kun er relatert til den aktuelle søylen.



Figur 8: Samspill mellom kundeliste og diagrammer



Figur 9: Oppdaterte diagrammer ved bruk av søkefunksjon og valg av én kunde

Det ble opprettet et visuelt element for å vise det totale antallet tilbakemeldinger etter filtrering. I tillegg ble det lagt til et måle-element for å vise gjennomsnittvurderingen av svarene. Disse elementene finnes under Vedlegg C – Dashbord-elementer.

For å kunne lese kommentaren til en bestemt tilbakemelding ble kunde, modul og kommentar lagt til i et tabellelement. Til slutt ble det laget en ny slicer for filtrering etter dato. Dette lar brukeren velge en start- og sluttdato for å filtrere datasettet, noe som gjør det mulig å presentere informasjon fra ulike perioder. Dette øker dashbordets fleksibilitet.

4.4.4 Bruk av dashbordet

Power BI tilbyr visning av dashbord i sin web-applikasjon. For å bruke denne nettsiden er det et krav om at brukeren har en Microsoft-konto. Dersom dette kriteriet er oppfylt, kan brukeren laste inn dashbordet lagret som en .pbix fil. Da får brukeren tilgang på dashbordet og datasettet i web-applikasjonen. For å holde datasettet oppdatert kan man ved hjelp av innstillingene godkjenne datakilden ved hjelp av databasens brukernavn og passord. Da vil brukeren kunne etterspørre oppdateringer på datasettet for å ha den senest tilgjengelige informasjonen i rapporten. Nedenfor er resultatet av det endelige dashbordet presentert fra en skjermdump.

Figur 10: Det endelige dashbordet

5 RESULTATER

5.1 Evalueringsmetode

Som nevnt i Kapittel 3.5 er enhetstesting, integrasjonstesting, systemtesting og brukertesting evalueringsmetodene som er brukt i prosjektet. Metodene ble brukt for å verifisere at systemet fungerte slik oppdragsgiver ønsket og for å validere idéen til produktet.

Enhetstesting ble gjort i programvaregrensesnittet under utvikling. Hovedformålet var å teste logikken til de ulike komponentene. Dette sikret at hver del i programvaren fungerte slik den skulle.

Når enhetstestene hadde blitt verifisert var det enklere å starte med integrasjonstesting. Dette gikk ut på å få flere instanser i systemet til å fungere sammen. Det ble gjort integrasjonstesting på ulike deler av systemet under utviklingen. Integrasjonstestene ble verifisert ved å teste og bygge systemet. En vellykket test ble utført da tilbakemeldinger ble lagret i databasen.

Systemtesting ble utført etter integrasjonstestene. Hele systemet ble testet og det ble verifisert at alle instansene fungerte sammen.

Brukertesting ble utført for å sikre at sluttproduktet var i tråd med forventningene til oppdragsgiveren. En brukertest ble utført under prosjektet der hele systemet ble evaluert. Brukertestingene ble gjennomført av oppdragsgiver i arbeidslokalet til Stacc Escali.

Under brukertesten skulle testpersonen opprette en tilbakemelding ved hjelp av testverktøyet og visualisere den i Power BI-dashbordet. Testpersonene var usikker på hvordan en tilbakemelding skulle sendes i JSON-format. Dermed måtte gruppen gi en demonstrasjon. Etter testen ble produktet evaluert ved at testperson ga tilbakemelding på systemets brukervennlighet og funksjon. Gruppen hadde også forberedt et evalueringsskjema som inneholdt spørsmål relatert til de funksjonelle kravene. Spørsmålene skulle besvares med et tall mellom 1 (svært uenig) og 5 (svært enig).

5.2 Evalueringsresultat

Resultatene fra evalueringsmetodene påvirket utformingen av systemet. Metodene bestod av enhetsteser, integrasjonstester, en systemtest og tilbakemeldinger fra personen som utførte brukertestingen.

5.2.1 Enhetstesting

Testresultatene fra enhetstestingen av programvaregrensesnittet ga gruppen en indikasjon om at logikken til produktet var utviklet riktig.

5.2.2 Integrasjonstesting

Testresultatet fra integrasjonstestingen som ble gjort under prosjektet.

Test	Instans som testes	Hensikt med testen	Resultat
1	API	Få brukt testverktøyet til å hente og lagre data fra hovedminne-database.	**1)** Vellykket test.
2	API og Database	Utvidelse av test 1. Få brukt API til å hente og lagre data fra tildelt databasen.	**1)** Problem med forbindelse mellom API og Database. **2)** Vellykket test. Løst ved å endre på tilkoblingsstrengen.
3	API, Database og Power BI	Utvidelse fra test 2. Importere visningen til Power BI og få ønsket data i et datasett.	**1)** Problem med visning. Fikk ikke ønsket datasett. **2)** Vellykket test. Løst ved ny visning.

Tabell 1: Integrasjonstester

Test 1: Første trinnet var å teste funksjonaliteten til programvaregrensesnittet. Testverktøyet sendte tilbakemeldinger til en hovedminne-database. Tilbakemeldingene ble deretter presentert i konsollen til utviklingsmiljøet.

Test 2: Programvaregrensesnittet kobles til en relasjonsdatabase og tilbakemeldinger blir lagret permanent. Tilbakemeldingene ble visualisert i administrasjonsverktøyet for databasen.

Test 3: Systemet kan nå oppdatere Power BI rapporten og nye tilbakemeldinger blir lagt til og presentert på dashbordet.

5.2.3 Systemtest

Da alle instansene var på plass, ble det utført en test av hele systemet. Testen var vellykket, og alle instansene fungerte sammen. Systemet lagret tilbakemelding, feilmeldinger og advarsler til databasen. Innsamling og lagring av brukerstatistikk var gjennomført. Brukerstatistikken ble fordelt ut på to entiteter i databasen. Tilbakemeldingsentiteten og kommentarentiteten. Power BI oppdaterte datasettet og nye data ble presentert på dashbordet.

Resultatet ble at alle instansene samhandlet riktig og brukerstaistikken lagret på riktig plass. Testene bidro til å løse problemstillingen og oppå målet som ble nevnt i Kapittel 1.

5.2.4 Brukertest

Etter brukertesten fikk gruppen tilbakemeldinger som ble brukt for evaluering. Kort sammendrag av tilbakemeldingen:

Testperson var fornøyd med programvaregrensesnittet og testverktøyet. Han uttalte at det interaktive grensesnittet til testverktøyet var oversiktlig og forenklet testingen. Han stilte spørsmål angående å sende tilbakemelding i JSON format. Etter en rask demonstrasjon fra gruppen ble det klart hvordan det skulle gjøres. Han var tilfreds med dashbord-løsningen, spesielt hvor oversiktlig det var. I tillegg var det positivt med de ulike filtreringene dashbordet tilbydde. Videre sa han det meste av videreutviklingen vil være å legge til funksjonalitet i porteføljesystemet.

Evalueringsskjemaet som ble besvart under brukertesten:

Nr.	Spørsmål	Svar
	Funksjonelle krav	
1	Synes du henting og lagring av brukerstatistikk er gjort på en god måte?	5
2	Er du tilfreds med omstruktureringen av databasen?	5
3	Inneholder visningen all nødvendig informasjon?	5
4	Hvor fornøyd er du med presenteringen av data på dashbordet?	5
5	Hvor fornøyd er du med testverktøyet for løsningen?	5
	Generelle spørsmål	
7	Hvor stor sannsynlighet er det for at Stacc Escali gjør videre arbeid på produktet?	5
8	Hvordan vil du vurdere kvaliteten av det leverte produktet?	5

Tabell 2: Evalueringsskjema

Oppdragsgiver kommenterte følgende på spørsmål 8:

"Mitt inntrykk er at det har vært en velfungerende gruppe som har jobbet godt og selvstendig. Produktet følger alle de funksjonelle kravene som ble fastsatt i starten. Bra levert!"

5.3 Prosjektresultat

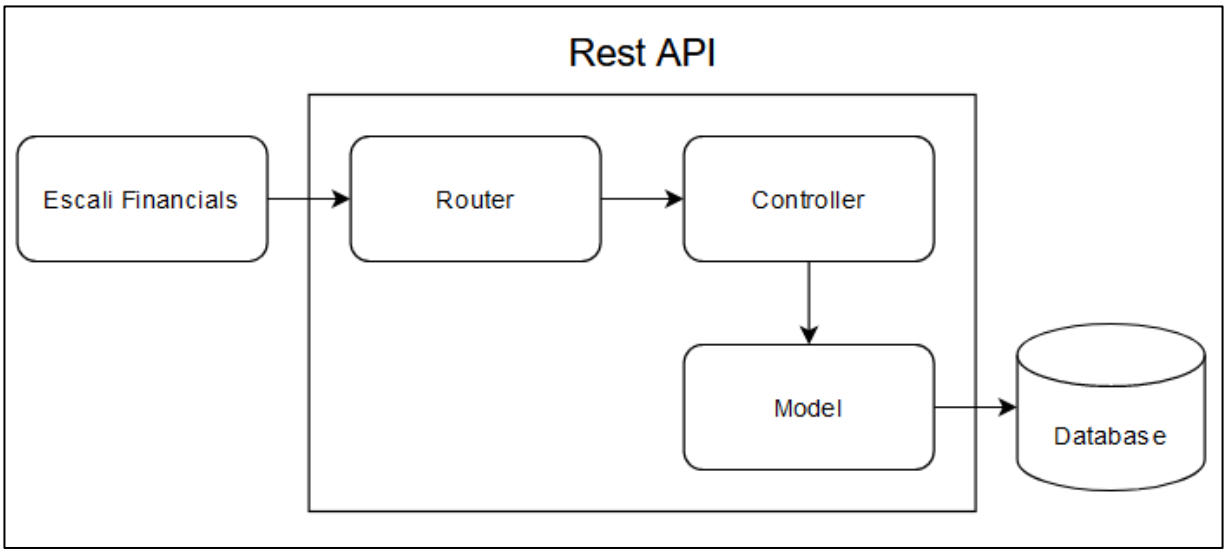
Resultatene fra evalueringsmetodene bidro til å løse problemstilling til prosjektet. Sluttproduktet hentet data ved hjelp av et testverktøy, lagret tilbakemeldinger i databasen og presenterte det i en Power BI rapport. Målsetningen for prosjektet er oppnådd ved å levere et system som møter kravene til oppdragsgiveren.

Enhetstestene, integrasjonstestene og systemtesten har verifisert at sluttproduktet tilfredsstiller kravene til oppdragsgiveren. Brukertesten indikerte at sluttproduktet har blitt utviklet korrekt og at det har potensiale for videre utvikling.

6 DISKUSJON

6.1 Fra initiell idé til endelig produkt

I startfasen av prosjektet hadde gruppen en initiell idé over hvordan sluttproduktet skulle bli utformet i henhold til problemstillingen. Den avviker i stor grad fra hva det endelige produktet kom til å bli.



Figur 11: Tidlig design av API

6.1.1 Programvaregrensesnitt

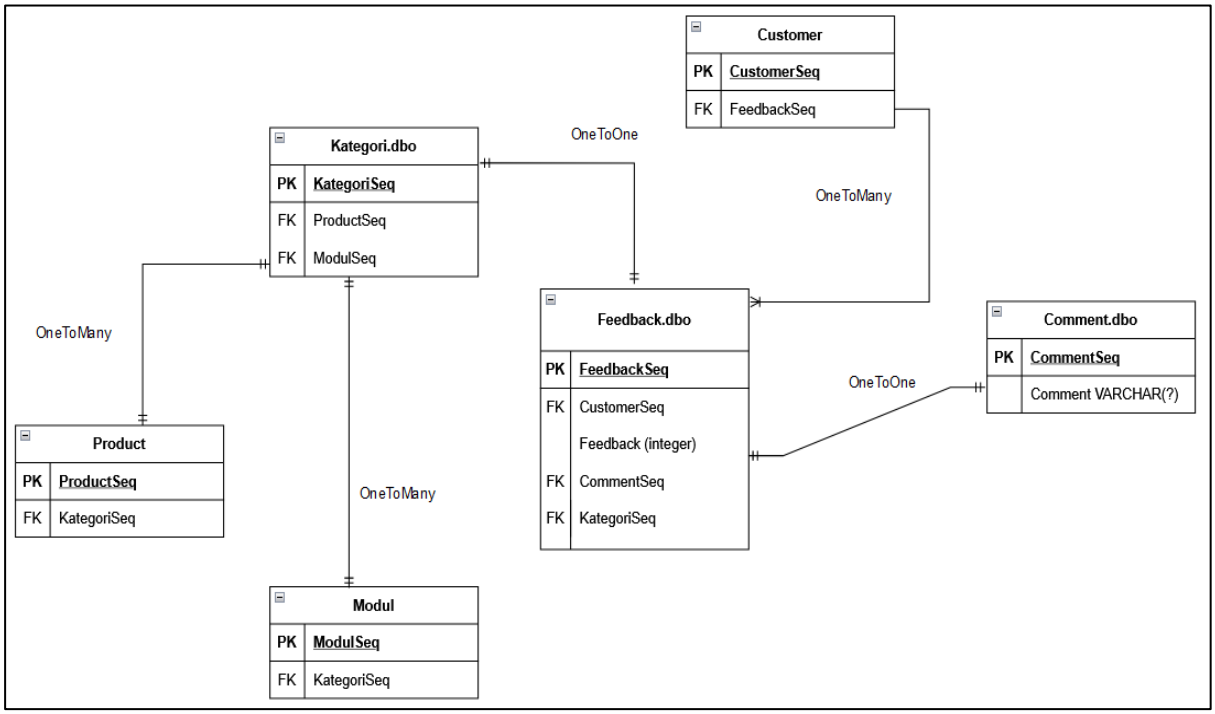
Det første utkaste av programvaregrensesnittet besto originalt av tre komponenter: Ruter, Kontroller og Model. Utkastet var basert på et eksisterende API som var utviklet i typescript og Node.js. Som nevnt i Kapittel 3.3 skulle gruppen ta i bruk .Net som rammeverk for programvaren. Rammeverket bruker en mellomvare som matcher URL-er til innkommende forespørsler og dirigerer dem til en handlingsmetode (Nowak, Larkin og Anderson, 2023). Dermed var det ikke behov for konvensjonell ruting i en klasse.

Gruppen hadde misforstått bruksområdet til modeller. Planen var opprinnelig å gjøre all databaselogikk i modellene. Etter grundig undersøkelser på konseptet bak modeller var det klart at dette var feil. Som beskrevet i Kapittel 4.2.1 skulle modellene kun holde på data og representere entiteter fra databasen. Handlers ble tatt i bruk for separering av kode og for databaselogikk. Dette ga bedre struktur og modellene ble brukt slik de skulle.

Arkitekturen i figur 11 eksponerer databaseentiteter direkte til klienten. For å skjule attributter klienten ikke trengte å se, ble dataoverføringsobjekter tatt i bruk. I motsetning til den første versjonen ble den endelige løsningen mer fleksibel for endringer.

6.1.2 Databaseløsning

Omstrukturering av database var det første som ble diskutert og endret på. Strukturen i figuren nedenfor var det første forslaget for omstrukturering. Designet var basert på andre databasediagrammer som inneholdt tilbakemeldinger og kommentarer som entiteter.



Figur 12: Skisse av databaseløsning

Den første løsningen for omstrukturering var ikke var kompatibel med den tildelte databasen. Grunnen var at modul og produkt ikke skulle ha en direkte kobling til kategori. Videre var det også vanskelig å skille mellom overordnede og underordnede kategorier. Dermed var det nødvendig med en re-modellering og gruppen endte opp med løsning som er presentert i Kapittel 4.3 Den endelige løsningen var mer skalerbar, oversiktlig og lagret brukerstatistikk på en bedre måte.

6.1.3 Visning

I løpet av utviklingen, ble visningen stadig endret før den var klar til å brukes i Power BI. Under oppbyggingen av dashbordet ble det oppdaget at det ikke var mulig å sortere etter dato, ettersom det ikke eksisterte en datokolonne i tilbakemeldingstabellen. Selv om dette ikke var et formelt krav, anså gruppen denne funksjonen som viktig for å kunne oppsummere data over en bestemt periode. Dermed ble en datokolonne opprettet i tilbakemeldingstabellen.

Ved en senere anledning ble det avdekket at det ikke var mulig å velge enkelte datoer som filtreringsalternativ i dashbordet. Dette skyldtes at Power BI kun inkluderte datoer fra datasettet. Kundelisten var heller ikke optimal, ettersom den bare viste kunder som hadde gitt tilbakemeldinger. Oppdragsgiver ønsket at kundelisten skulle være statisk, og inneholde alle kundenavn, uavhengig av om de hadde gitt tilbakemeldinger eller ikke.

For å adressere disse problemene ble den opprinnelige visningen kombinert med tabellene for kunder, moduler, samt under- og hovedkategorier. Disse ble deretter kryss-multiplisert med alle datoer, fra og med 1. januar 2023. Da ble det mulig å filtrere på alle datoer, og kundene ble med selv om de ikke hadde gitt noen tilbakemeldinger. Denne endringen forbedret brukervennligheten til dashbordet.

En ulempe med denne løsningen er at kryssproduktet danner en stor visning. I tillegg vil antall rader i visningen øke over tid. Grunnen er at nye datoer legges til i visningen. Dermed blir importeringen av data ressurskrevende for Power BI. Dette vil påvirke brukeropplevelsen når datamengden øker.

Et alternativ kunne vært å importere alle nødvendige tabeller i Power BI. Da kan Power Query-funksjonen brukes til å opprette et datasett lik visningen i databasen. Det ble delvis forsøkt, men forkastet. Hovedgrunnen er at gruppen har mer kompetanse med visninger, kontra Power Query. I tillegg vil Power Query være like ressurskrevende som visningen fra databasen.

6.2 Drøfting

Dette delkapittelet inneholder en diskusjon av det endelige produktet opp mot de funksjonelle kravene som ble spesifisert i starten av prosjektet. Videre diskuteres det om løsningen besvarer problemstillingen.

I henhold til Kapittel 2.1.2 Initielle krav, ble følgende kriterier gitt av oppdragsgiver:

- En måte å hente inn brukerstatistikk.
- En måte å samle brukerstatistikken.
- Database skal være i stand til å motta brukerstatistikk
- Lage struktur i form av en visning.
- Visualisere informasjon på et dashbord.
- En måte å teste løsningen.

Ved en brukertest og demonstrasjon av sluttproduktet, kan gruppen konkludere med at produktet oppfyller kravene til oppdragsgiver på en god måte. I tråd med SCRUM-metodikken har prosjektgruppen hatt møte med oppdragsgiver hver fredag. Hensikten med disse møtene har vært å presentere arbeidet som er utført, og gjøre eventuelle tilpasninger for å oppfylle de funksjonelle kravene. Møtene har vært avgjørende for å identifisere og løse problemer som kunne oppstå underveis. Videre har møtene vært viktig for å koordinere utviklingsprosessen mellom gruppens medlemmer og oppdragsgiver. Dette sikrer at alle parter er enige om den endelige løsningen.

I Kapittel 1.4.1 Problembeskrivelse blir problemstillingen for prosjektet definert: "Hvordan utvikle et dynamisk system som henter tilbakemeldinger, brukerstatistikk og feilmeldinger, og deretter presentere dette."

Produktet oppfyller de tidligere nevnte funksjonelle kriteriene, og det kan konkluderes med at brukerstatistikk blir hentet, lagret og presentert. Gjennom prosjektet har det blitt klargjort at tilbakemeldinger og feilmeldinger er underkategorier av brukerstatistikk. De ulike kategoriene av brukerstatistikk blir presentert på dashbordet. Dette resulterer i at delen "Hente tilbakemeldinger, brukerstatistikk og feilmeldinger, og deretter presentere dette." av problemstillingen oppfylles.

Den første delen av problemstillingen spør: "Hvordan utvikle et dynamisk system…" Dette kan besvares med hvordan gruppen har utviklet programvaregrensesnittet og omstrukturert databasen. Programvaregrensesnittet tar i bruk en kommunikasjons-arkitektur og bruker et designmønster som organiserer koden. Databasen har blitt modifisert på en ryddig og oversiktlig måte til at den kan motta tilbakemeldinger. Sammen gjør de sluttproduktet skalerbart, enklere å vedlikeholde og videreutvikle. Dermed er systemet dynamisk og tar høyde for nye endringer. Med dette er problemstillingen besvart.

Gruppen er fornøyd med resultatet og sammen med oppdragsgiver kan det konkluderes at målet er nådd. Sluttproduktet er enkelt å ta i bruk, automatiserer og forenkler innsamling av brukerstatistikk. I fremtiden vil det være et nyttig verktøy for ledelsen i Stacc Escali. Systemet vil enkelt kunne tilpasses slik at det kan brukes når funksjonaliteten i porteføljesystemet er implementert.

6.3 Oppsummering av prosjektet

Prosjektet startet på en god måte der mange timer gikk til forberedende arbeid. Gruppen satte opp planer for hvilke dager som skulle være på kontoret og en felles kalender ble opprettet for viktige frister. I startfasen ble gruppen enig med oppdragsgiver om at vi skulle ha SCRUM-møte hver uke på fredager. Det var et godt rammeverk rundt prosjektet og en samarbeidsvillig og engasjert oppdragsgiver.

I startfasen oppsto det utfordringer i den forstand av at det var vanskelig å designe et produkt som skulle være kompatibelt med et system gruppen ikke kjente til. Oppgaven virket også overveldende og det tok tid før gruppen forstod hva som skulle gjøres. Et av de største spørsmålene til gruppen var "Hvordan skal brukerstatistikk inneholde informasjon om feilmeldinger og advarsler?" Det tok litt tid før alt falt på plass. I mellomtiden gikk mye av tiden til å tilegne seg informasjon om API-design og Power BI, mens gruppen ventet på svar. Det hadde vært en fordel om gruppen fikk redegjort uklarheten tidligere i oppstartsfasen.

Omstruktureringen av databasen var også en utfordring for gruppen. Det tok flere forsøk før en løsning var i samsvar med oppdragsgiverens ønsker. Dette var fordi det var vanskelig å forstå hvordan entitetene i databasen skulle relateres og hvilke attributter som skulle inkluderes i den endelige tilbakemeldingen.

Flere av medlemmene hadde deltidsjobber og andre fag, noe som førte til bekymring rundt forpliktelsene og hvordan dette kunne påvirke prosjektet negativt. Løsningen ble å jobbe med oppgaven tre faste dager i uken. I eksamensperioden var tiden knapp. Dermed ble det mindre arbeid med oppgaven enn planlagt. Gruppen kunne fordelt og planlagt arbeidet utover en lengre periode. Det tapte arbeidet ble likevel tatt igjen mot slutten.

7 KONKLUSJON OG VIDERE ARBEID

7.1 Konklusjon

Sluttproduktet består av et system som henter og lagrer brukerstatistikk ved hjelp av et programvaregrensesnitt, og deretter presenterer dette på et dashbord. Systemet er dynamisk og tar høyde for at innhenting av statistikk kan endre seg over tid.

Målsetningen for prosjektet har vært å utvikle et verktøy som kan skaffe ledelsen i Stacc Escali oversikt over bruken av deres systemer. Ved å koble på sluttproduktet med porteføljesystemet, har prosjektet nådd målet.

Evalueringsresultatene fra 5.2 Evalueringsresultat indikerer at de opprinnelige målene fra Kapittel 1 var nådd. Systemtesten bekreftet at sluttproduktet kunne hente og lagre tilbakemeldinger i databasen. Videre ble brukerstatistikk presentert på et dashbord i form av en Power BI rapport. Brukertesten ga gruppen en indikasjon over brukervennligheten til systemet fra perspektivet til en ny bruker. Den bekreftet også at produkt-idéen var korrekt. Sluttproduktet møter alle kravene til oppdragsgiver og tilbakemeldingene har vært meget positive.

Kravene til oppdragsgiver er møtt. Dermed kan problemstillingen besvares: "Hvordan utvikle et dynamisk system som henter tilbakemeldinger, brukerstatistikk og feilmeldinger, og deretter presentere dette." Svaret på problemstillingen beskrives i rapporten. Det står oppført hvilke teknologier som blir brukt, hvordan de blir implementert og hvordan de fungerer. Rapporten tar for seg hvordan alle delene i systemet fungerer sammen for å skape en applikasjon for brukerstatistikk fra porteføljesystemet.

7.2 Videre arbeid

Sluttproduktet er utviklet på en måte som gjør det kompatibelt med porteføljesystemet og tilrettelegger for videre arbeid. Gruppen har overført eierskap av GitHub arkivet til oppdragsgiver. Kodedokumentasjon, API-dokumentasjon, instruksjoner for nedlastning og bruk er godt forklart.

Fra evalueringsskjemaet i Kapittel 5.2, var oppdragsgiver innstilt på å videreutvikle systemet. Videre arbeid innebærer å legge til funksjonalitet i porteføljesystemet og koble på programvaregrensesnittet. I tillegg må programvaren bli kjørt i Azure App Service. Alternativt kan programvaregrensesnittet bli lagt til og kjørt på virtuell maskin. Databaseløsningen og programvaregrensesnittet tar høyde for eventuelle endringer i brukerstatistikken.

For å redusere overhead, forventes det at spørringen som oppretter visningen justeres noe. I den nåværende løsningen øker antall rader som returneres daglig. Videre arbeid kan innebære å la data bli foreldet, slik at de ikke lenger inkluderes i datasettet.

8 REFERANSER

Frøslie, K.F., & Bjørnstad, J. (2021). *Statistikk,* i Store norske leksikon. Tilgjengelig fra: https://snl.no/statistikk (Hentet: 14. februar 2023).

Microsoft (u. å, a). *What is OneDrive for work or school?* Tilgjengelig fra: https://support.microsoft.com/en-us/office/what-is-onedrive-for-work-or-school-187f90af-056f-47c0-9656-cc0ddca7fdc2 (Hentet: 19. april 2023).

Postman (2021). *What is Postman?* Tilgjengelig fra: https://www.postman.com/ (Hentet: 7. mars 2023).

SmartBear (u. å.). *About Swagger*. Tilgjengelig fra: https://swagger.io/about/ (Hentet: 7. mars 2023).

Microsoft (u. å, b). *What is Power BI.* Tilgjengelig fra: https://powerbi.microsoft.com/en-us/what-is-power-bi/ (Hentet: 21. april 2023).

Google (u. å.) *Looker Studio.* Tilgjengelig fra: https://support.google.com/looker-studio/answer/6283323?hl=en (Hentet: 21. april 2023).

Microsoft (u. å, c*). Build. Test. Deploy.* Tilgjengelig fra: https://dotnet.microsoft.com/en-us/ (Hentet: 3. mai 2023).

Glasspaper (u. å.). *Hva er egentlig Scrum?* Tilgjengelig fra: https://www.glasspaper.no/artikkel/hva-er-egentlig-scrum/ (Hentet: 7. mars 2023).

Kanbanize (u. å.). *What is Kanban?* Tilgjengelig fra: https://kanbanize.com/kanban-resources/getting-started/what-is-kanban (Hentet: 7. mars 2023).

Gantt.com (u. å.). *Hva er et Gantt-diagram?* Tilgjengelig fra: https://www.gantt.com/no/ (Hentet: 7. mars 2023).

Aven, T. (2017). *Risikoanalyse,* i Store norske leksikon. Tilgjengelig fra: https://snl.no/risikoanalyse (Hentet: 5. mai 2023).

Microsoft (2023, a). *RESTful web API Design*. Tilgjengelig fra: https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design (Hentet: 5. mai 2023).

Codecadamy (u. å.). *What is Rest?* Tilgjengelig fra: https://www.codecademy.com/article/what-is-rest (Hentet: 3. mai).

Smith, S. (2022). *Overview ASP.NET Core MVC*, i Microsoft Learn. Tilgjengelig fra: https://learn.microsoft.com/nb-no/aspnet/core/mvc/overview?WT.mc_id=dotnet-35129-website&view=aspnetcore-6.0/ (Hentet: 28. april 2023).

Anderson, R., & Larkin, K. (2023). *Tutorial: Create a web API with ASP.NET Core.* Tilgjengelig fra: https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-7.0&tabs=visual-studio (Hentet: 29. april 2023).

Anderson, R. (2022). *Create Data Transfer Objects (DTOs).* Tilgjengelig fra: https://learn.microsoft.com/en-us/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-5 (Hentet: 2. mai 2023).

Esposito, D. (2009*). Cutting Edge - Pros and Cons of Data Transfer Objects*. Tilgjengelig fra: https://learn.microsoft.com/en-us/archive/msdn-magazine/2009/august/pros-and-cons-of-data-transfer-objects (Hentet: 2. mai 2023).

Microsoft (u. å, d). *Improve the developer experience of an API with Swagger documentation.* Tilgjengelig fra: https://learn.microsoft.com/en-us/training/modules/improve-api-developer-experience-with-swagger/ (Hentet: 3. mai 2023).

IBM (2021). *Database views*. Tilgjengelig fra: https://www.ibm.com/docs/en/control-desk/7.6.1.2?topic=structure-views (Hentet: 28. april 2023).

Microsoft (2023, b). *Views.* Tilgjengelig fra: https://learn.microsoft.com/en-us/sql/relational-databases/views/views?view=sql-server-ver16 (Hentet: 28. april 2023).

Microsoft (2023, c). *Slicers in Power BI.* Tilgjengelig fra: https://learn.microsoft.com/en-us/power-bi/visuals/power-bi-visualization-slicers?tabs=powerbi-desktop (Hentet: 2. mai 2023).

Nowak, R., Larkin, K., & Anderson, R. (2023). *Routing to controller actions in ASP.NET Core*. Tilgjengelig fra: https://learn.microsoft.com/en-us/aspnet/core/mvc/controllers/routing?view=aspnetcore-6.0 (Hentet: 6. mai 2023).

9 VEDLEGG

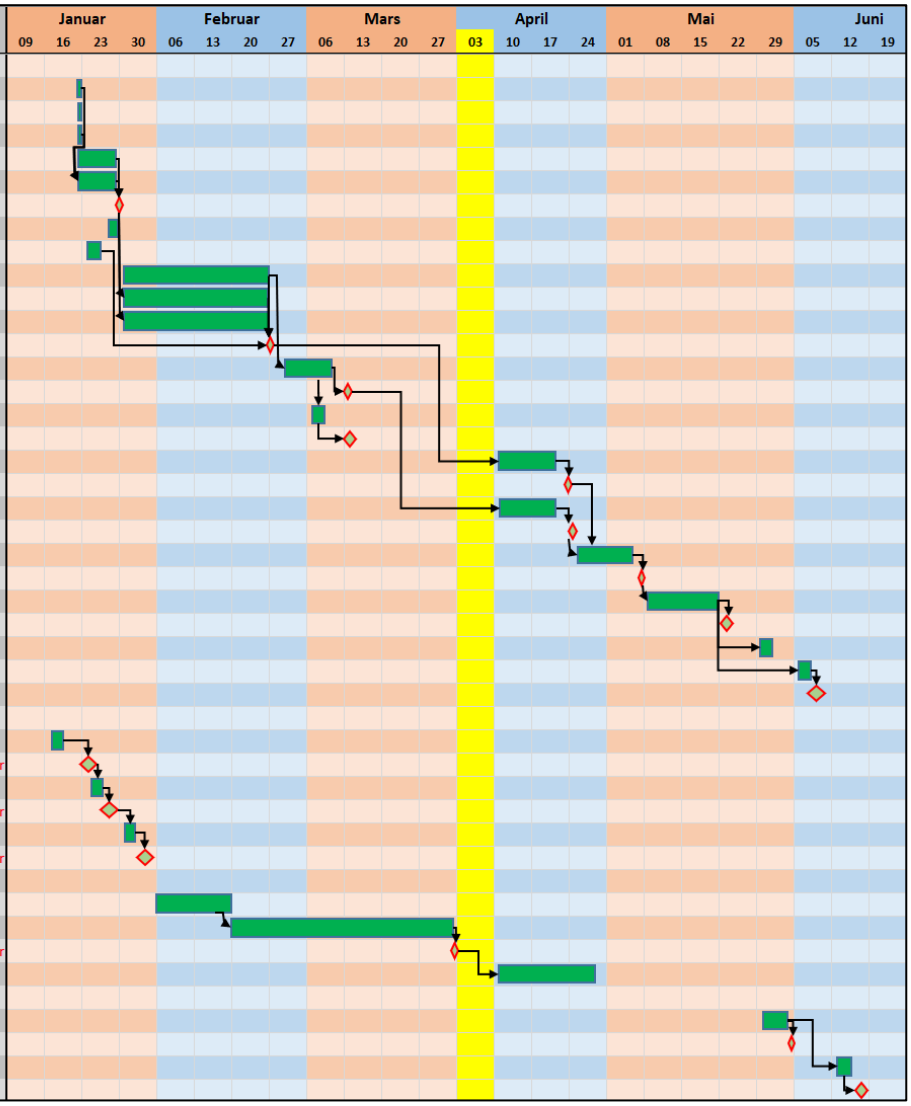
Følgende dokumenter er lagt til:

- GANTT-Diagram
- Risikoanalyse
- Dashbord-elementer

9.1 Vedlegg A – GANTT-Diagram

GANTT-Diagram Revidert 22.05.2023

ID	Utført	Oppgave	Varighet	Start	Timer	Ansvarling
1.0		**Dokumentasjon**	**139 Dager**	**20.01.2023**	**462**	**Adrian**
1.1	X	Risikoanalyse	1 Dag	20.01.2023	3	Nicolai
1.2	X	Veiledningsavtale	1 Dag	20.01.2023	1	Anders
1.3	X	GANTT Diagram	1 Dag	20.01.2023	12	Adrian
1.4.1	X	Visjonsdokument 1. Iterasjon	7 Dager	20.01.2023	16	Anders
1.4.2	X	Prosjekthåndbok 1. Iterasjon	7 Dager	20.01.2023	16	Nicolai
1.4.3	X	Godkjenning	1 Dag	29.01.2023	0	Veileder
1.5	X	Adganger og taushetserklæringe	1 Dag	27.01.2023	1	Adrian
1.6.1	X	Kravspesifikasjon 1. Iterasjon	2 Dager	24.01.2023	6	Anders
1.6.2	X	Midtveisrapport 1. Iterasjon	26 Dager	31.01.2023	60	Adrian
1.6.3	X	Visjonsdokument 2. Iterasjon	26 Dager	31.01.2023	16	Anders
1.6.4	X	Prosjekthåndbok 2. Iterasjon	26 Dager	31.01.2023	16	Nicolai
1.6.5	X	Godkjenning	1 Dag	26.02.2023	1	Veileder
1.7.1	X	Midtveisrapport	10 Dager	28.02.2023	60	Adrian
1.7.2	X	Godkjenning	1 Dag	12.03.2023	1	Veileder
1.8.1	X	Lage midveispresentasjon	2 Dager	07.03.2023	4	Nicolai
1.8.2	X	Midtveispresentasjon	1 Dag	14.03.2023	6	Veileder
1.9.1	X	Prosjekthåndbok 3. Iterasjon	10 Dager	11.04.2023	16	Nicolai
1.9.2	X	Godkjenning	1 Dag	23.04.2023	1	Veileder
1.10.1	X	Rapport Utkast 1	10 Dager	11.04.2023	72	Anders
1.10.2	X	Godkjenning	1 Dag	24.04.2023	1	Veileder
1.10.3	X	Rapport Utkast 2	10 Dager	25.04.2023	24	Anders
1.10.4	X	Godkjenning	1 Dag	07.05.2023	1	Veileder
1.11.1	X	Endelig Rapport	13 Dager	09.05.2023	100	Adrian
1.11.2		Innlevering Bacheloroppgave	14 Dager	22.05.2023	0	Adrian
1.12		Refleksjonsnotat	2 Dager	29.05.2023	9	Individuelt
1.13.1		Lage presentasjon	2 Dager	06.06.2023	16	Nicolai
1.14.1		Presentasjon	1 Dag	08.06.2023	3	Alle
2.0	X	**Modellering av Database**	**17 Dager**	**17.01.2023**	**41**	**Anders**
2.1.1	X	Forslag til omstrukturering	2 Dager	17.01.2023	8	Anders
2.1.2	X	Godkjenning	1 Dag	24.01.2023	1	Oppdragsgiver
2.2.1	X	Implementere endringer	2 Dager	25.01.2023	18	Anders
2.2.2	X	Godkjenning	1 Dag	27.01.2023	1	Oppdragsgiver
2.3.1	X	Teste endringer	2 Dager	31.01.2023	12	Nicolai
2.3.2	X	Godkjenning	1 Dag	03.02.2023	1	Oppdragsgiver
3.0	X	**Utvikling av API**	**67 Dager**	**03.02.2023**	**324**	**Nicolai**
3.1	X	Planlegging	13 Dager	03.02.2023	72	Anders
3.2.1	X	Utvikling	42 Dager	17.02.2023	180	Nicolai
3.2.2	X	Godkjenning	1 Dag	31.03.2023	0	Oppdragsgiver
3.3	X	Testing	18 Dager	11.04.2023	72	Adrian
4.0		**EXPO**	**16 Dager**	**30.05.2023**	**73**	**Nicolai**
4.1.1		EXPO Poster	3 Dager	30.05.2023	36	Anders
4.1.2		EXPO Poster innlevering	1 Dag	02.06.2023	1	Alle
4.2.1		EXPO Forberedende arbeid	2 Dager	13.06.2023	18	Nicolai
4.2.2		EXPO Bergen	1 Dag	15.06.2023	18	Alle



Figur A-1: GANTT-Diagram

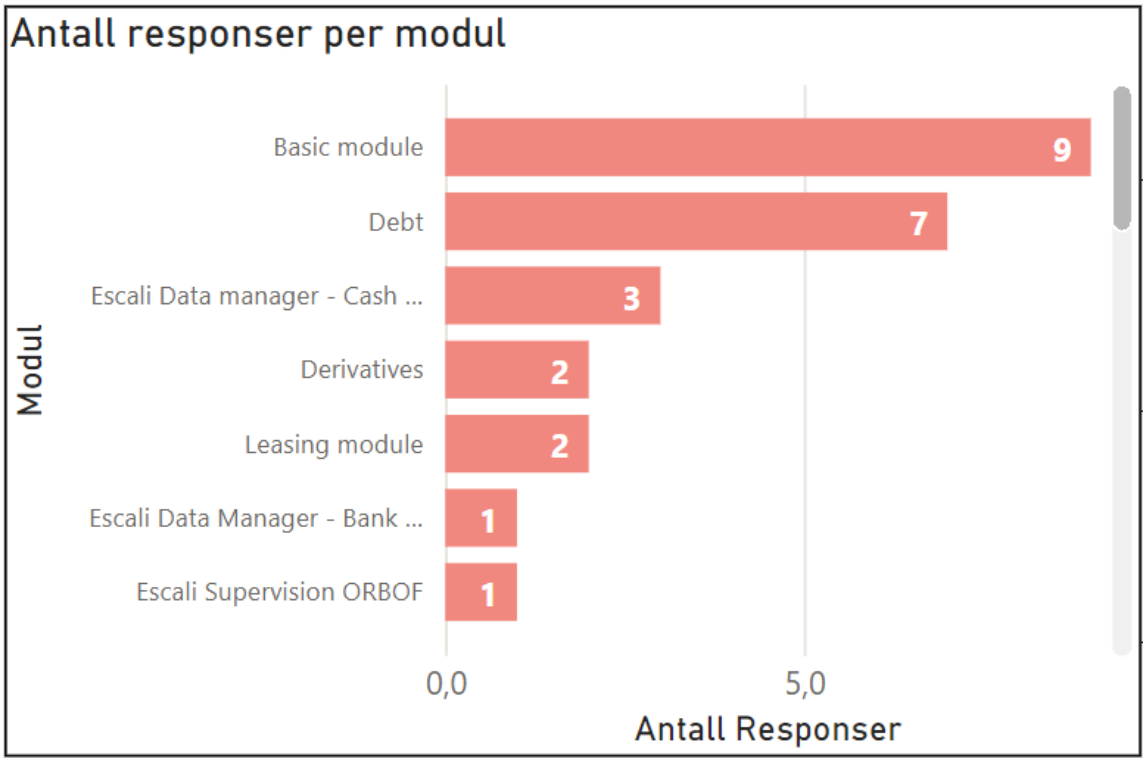
9.2 Vedlegg B - Risikoanalyse

Risiko analyse revidert 26.04.2023

	Hendelse /Risiko	Årsak	Sannsyn-lighet	Konse-kvens	Risiko-produkt	Tiltak
1	Dårlig kommunikasjon	Man snakker ikke sammen/ Dårlig bruk av kommunikasjons-kanaler.	2	3	6	Kontor er tilgjengelig hos oppdragsgiver, ha ukentlige møter der.
2	Manglende kompetanse til å utføre prosjekt	Studentene har ikke vært oppmerksomme under undervisningen.	1	3	3	Ta ansvar for å fylle eventuelle kunnskapsmangler gjennom individuell innsats.
3	Sykdom	Man blir syk	2	3	6	Gode personlige rutiner.
4	Ferie/turer	Man ønsker et avbrekk	3	3	9	Være tidlig ute slik at gruppen kan planlegge rundt en eventuell reise. Holde alle parter oppdatert. Vurder muligheten for å utføre arbeid på en fjernbasert måte.
5	Konflikt i gruppen	Uenigheter, misforståelser og/eller dårlig kommunisering	2	4	8	Sørge for et miljø, hvor hvert gruppemedlem føler seg hørt. Ta opp potensielle konflikter tidlig. Ta opp med intern veileder.
6	Begrenset arbeids-kapasitet	Deltidsjobber og andre fag.	4	4	16	God organisering og planlegging. Gode rutiner og oppfølging.
7	Tekniske utfordringer	Maskinvaresvikt eller programvaresvikt	2	3	6	Gode vedlikeholdsrutiner.

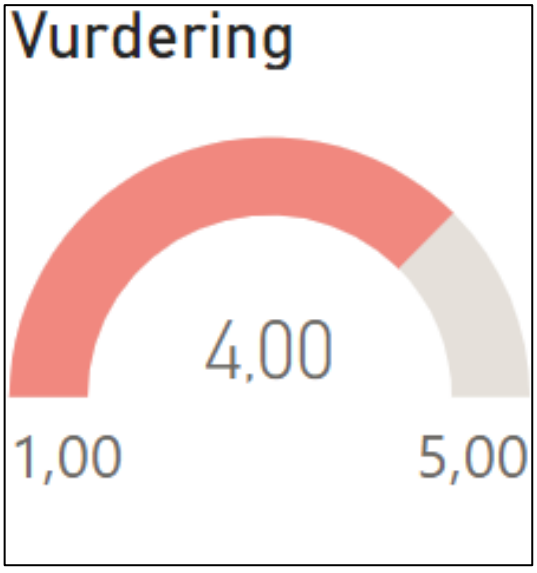
Tabell B-1: Risikoanalyse

9.3 Vedlegg C – Dashbord-elementer



Figur C-1: Liggende søylediagram for antall tilbakemeldinger per modul

Figur C-2: Kort-element som viser antall tilbakemeldinger etter filtrering



Figur C-3: Måler-element som viser gjennomsnittlig vurdering