Høgskulen på Vestlandet

BACHELOROPPGAVE

Presentasjon og Optimalisering av Romlig Informasjonsstrøm for Bergen brannvesen

Presentation and Optimization of Spatial Information Flow for Bergen Fire Department

Prosjektdeltakere: *Kristine Bellen Wear, Frida Landro Johansen, Olav Markus Stiegler Folgerø*

Bachelor i landmåling og eiendomsdesign
Fakultet for teknologi, miljø- og samfunnsvitenskap
Institutt for bygg, miljø- og naturvitenskap

Veileder: *Stig Frode Samnøy*
Innleveringsdato: *21.05.2024*
Antall ord: 19504

Vi bekrefter at arbeidet er selvstendig utarbeidet, og at referanser/kildehenvisninger til alle kilder som er brukt i arbeidet er oppgitt, jf. Forskrift *om studium og eksamen ved Høgskulen på Vestlandet, § 12-1.*

Forord

Vår bacheloroppgave representerer avslutningen på et treårig studieløp innen landmåling og eiendomsdesign på Høgskulen på Vestlandet ved Fakultetet for teknologi, miljø- og samfunnsvitenskap. Bacheloroppgaven ble ferdigstilt våren 2024 og tilsvarer 20 studiepoeng. Arbeidet med oppgaven har gitt oss bedre innsikt og en dypere forståelse for bruken av GIS innen nødetater og de potensielle fordelene som open source kan tilby. Prosjektgruppen består av Kristine Bellen Wear, Frida Landro Johansen og Olav Markus Stiegler Folgerø.

Vi ønsker å rette en takk til Bergen brannvesen for samarbeid og alle bidrag til prosjektet. Videre vil vi takke vår veileder Stig Frode Samnøy for veiledning og faglig innsikt gjennom hele utviklingsarbeidet.

Bergen, mai 2024

Oppsummering

I dette prosjektet har vi utviklet et dashbord som viser hvordan geografiske informasjonssystemer (GIS) kan brukes til å optimalisere informasjonsstrømmen for Bergen brannvesen. Ved å forbedre GIS i etatens daglige arbeid er målet å forbedre bruken av kartdata i utrykningssituasjoner gjennom bedre håndtering og visualisering av romlig data.

Vår tilnærming har involvert bruk av open source-programvare som QGIS og PostgreSQL, samt programmering i R for å utvikle et tilpasset dashbord. Gjennom oppgaven har vi også sammenlignet noen av fordelene og ulempene ved open source- og proprietær programvare, som ArcGIS.

Resultatene viser at implementering av GIS kan gi betydelige forbedringer for brannvesenet, da det kan gi raskere og mer presis informasjon. Dashbordet som ble utviklet i løpet av prosjektet tillater tilgang til oppdatert og detaljerte kart. Dashbordet er utviklet for at man lett kan bruke det gjennom et brukervennlig grensesnitt som er enkelt å forstå.

Oppgaven konkluderer med at bruk av open source-programvare og moderne databaseteknologier kan representere en kostnadseffektiv og skalerbar løsning. Ved å videreutvikle kartløsningene, kan brannvesenet ytterligere forbedre sin evne til å håndtere nødsituasjoner.

Abstract

In this project, we developed a dashboard that demonstrates how Geographic Information Systems (GIS) can be utilized to optimize the flow of information for Bergen Fire Department. By improving GIS into the agency's daily operations, the goal is to enhance the use of map data in emergency response situations through improved management and visualization of spatial data.

Our approach involved the use of open-source software such as QGIS and PostgreSQL, as well as programming in R to develop a customized dashboard. Throughout the task, we also compared some of the advantages and disadvantages of open-source and proprietary software, such as ArcGIS.

The results indicate that implementing GIS can lead to significant improvements for the fire department, as it can provide faster and more accurate information. The dashboard developed during the project allows access to updated and detailed maps. The dashboard is designed to be easily used through a user-friendly interface that is simple to understand.

The thesis concludes that the use of open-source software and modern database technologies can represent a cost-effective and scalable solution. By further developing these mapping solutions, the fire department can further improve its ability to handle emergencies.

Innholdsfortegnelse

Begrepsforklaring

API står for Application Programming Interface og fungerer som en bro som gir programvare tilgang til data og funksjoner i et annet datasystem. Dette gjør det enklere for ulike systemer eller tjenester å kommunisere (Rossen & Nätt, 2024).

Brannstasjon er en fysisk bygning eller et anlegg der brannmannskap, utstyr og kjøretøy er stasjonert for å kunne raskt respondere på brannalarmer og andre nødsituasjoner (Rimstad, 2023).

Beredskap skal organiseres i vaktberedskap etter antall innbyggere (*Brann- og redningsvesenforskriften*, 2022). Vaktberedskapet omfatter brann og redningstjenester. Avdelingen håndterer også vedlikeholdsoppgaver, øvelser og undervisningsaktiviteter (*Beredskapsavdelingen*, u.å.).

Brannvesen er et kommunalt organ med hovedoppgave å forebygge og slokke branner. Det opererer i tråd med forskrifter og retningslinjer fastsatt av Direktoratet for samfunnssikkerhet og beredskap (DSB) (Rimstad, 2023).

Brukergrensesnitt eller User Interface handler om å fokusere på valg av utforming. Det inkluderer valg av farger, skrifttyper og generell visuell estetikk (Aaberge, 2021).

Brukeropplevelse eller User Experience er en tilnærming til brukeropplevelse. Den fokuserer på hvordan bruker navigerer og finner fram. Målet er å skape en opplevelse av flyt, der brukeren føler seg naturlig ledet gjennom systemet uten unødvendige hindringer eller stopp (Aaberge, 2021).

Dashbord viser nøkkelinformasjon og målinger på ett oversiktlig panel. Det integrerer data fra flere kilder og presenterer dem slik at bruker enkelt kan forstå og samhandle med informasjonen. Dashbordet kan effektivisere kommunikasjon og beslutningstaking (Changiz, 2024).

Datadefinisjonsspråk (DDL) brukes til å sette opp relasjonsdatabase, lage tabeller og definere kolonnene i tabellene. Definisjonene av tabellene og kolonnene kalles ofte for skjemaet til relasjonsdatabasen (Bratbergsengen et al., 2023).

Datamanipulasjonsspråk (DML) brukes til å stille spørsmål (query), sette og slette data og endre på data i databasen (Bratbergsengen et al., 2023).

FKB står for felles kartdatabase og representerer et omfattende datasett som inneholder noen av de mest presise og detaljerte kartleggingsdataen tilgjengelig i Norge. Ansvaret for spesifikasjonene til FKB ligger hos Geovekst-samarbeidet, som er organisert og koordinert gjennom Geovekst-forum (Granum, 2023).

Forebyggende avdeling må brann- og redningsvesenet bestemme hva som utgjør tilstrekkelig forebyggende arbeid for å oppfylle de oppgavene de er satt til å håndtere. Dette arbeidet er

avgjørende for å minimere risiko gjennom forebyggende tiltak (*Brann- og redningsvesenforskriften*, 2022).

GIS står for geografisk informasjonssystem og er en teknologi designet for å samle inn, organisere, lagre, analysere og visualisere geografisk informasjon. Oppbyggingen av systemet inkludert data, brukere, maskinvare, programvare og metodikk (Ørstavik & Mæhlum, 2023).

IMRaD er en skrivemodell som oversettes til Introduksjon – Metode – Resultater – og – Diskusjon. Det er en mal for hvordan en vitenskapelig artikkel kan være bygget opp (*IMRaD-modellen*, 2024).

Kartgrensesnitt refererer til det visuelle brukergrensesnittet eller den digitale plattformen som tillater bruker å samhandle med kartdata. Det er det verktøyet eller grensesnittet som lar bruker visualisere og utforske geografiske data brukervennlig måte (Bratbergsengen & Bothner-By, 2023).

KI er forkortelsen for Kunstig intelligens. Det dreier seg om å skape datasystemer som kan lære å løse vanskelige oppgaver på egen hånd (Astrup, 2020).

Offentlig domene refererer til verk som ikke er beskyttet av opphavsrett eller hvis opphavsrettigheter har utløpt, blitt avstått eller er ugyldige. Verk i det offentlige domenet kan brukes av hvem som helst uten restriksjoner (Liseter, 2023).

Open source-programvare er programvare med åpen kildekode som alle kan inspisere, modifisere og distribuere (Gramstad, 2023).

PgAdmin er en gratis åpen plattform. Programvaren tilbyr en brukervennlig måte å behandle databaser i PostgreSQL. Programmet fungerer som et brukergrensesnitt, der man kan gjøre spørringer til databasen. Det gir muligheter for å opprette databaser lokalt på pc-en, men man kan også koble den opp til en ekstern database (*pgAdmin*, u.å.).

PostgreSQL er et utvidbart, open source relasjonelt databasehåndteringssystem. PostgreSQL gir støtte til både SQL (relasjonelle databaser) JSON (for ikke-relasjonelle forespørsler), noe som gjør den svært fleksibel for ulike typer dataarbeid (Introduction to PostgreSQL, u.å.) (PostgreSQL Tutorial, u.å.).

Programmeringsspråket R er spesialisert for statistisk databehandling og visualisering. R er et tolket språk med kommandolinjegrensesnitt. R's funksjonalitet kan utvides gjennom bruk av pakker. Pakkene dekker et bredt spekter av statistiske teknikker, inkludert anvendelig for romlige analyser (Smith & Venables, 2024).

Proprietær kildekode refererer til programvarekoden som er eiet og kontrollert av enkeltpersoner, selskaper eller organisasjoner. Denne koden er ikke åpen for offentligheten, og bruker har ikke tillatelse til å endre, dele eller studere koden uten tillatelse fra eieren (*Proprietær vs Åpen Kildekode*, u.å.).

QGIS er en gratis open source programvare. Med QGIS har man muligheten til å se, behandle og redigere romlig data. I QGIS kan man lage detaljerte kart og gjøre geografiske analyser ved å hente data fra ulike kilder. Vi kan også bruke QGIS til å importere og eksportere romlig data inn i PostgreSQL databaser («QGIS», 2020).

Relasjonsdatabaser er en type database som organiserer data i tabeller, eller "relasjoner", hvor hver tabell består av rader og kolonner (Bratbergsengen, 2023).

Romlig data er data som inneholder et geografisk element, som koordinater eller polygoner, og brukes i GIS for kartlegging og analyse. Denne typen data kan være vektorbasert, representert ved punkter, linjer og polygoner, eller rasterbasert, hvor hver celle representerer et spesifikt område på jorden (Rød & Dick, 2023).

RStudio er et integrert utviklingsmiljø for programmeringsspråk som R, Python og SQL. RStudio er et brukergrensesnitt som inkluderer kodeeditor, verktøy for plotting, historikk, debugging og prosjektforvaltning. R støtter også et bredt bibliotek av pakker som eksempelvis: *Shiny*, *Sf* og *Leaflet* (Kosourova, 2022)

Sanntidsdata refererer til informasjon som samles inn og umiddelbart gjøres tilgjengelig for bruk eller analyse straks etter at den er oppnådd. Sanntidsdata muliggjør øyeblikkelig beslutningstaking og interaksjon basert på de aller nyeste opplysningene (Arnøy, 2018)

110-sentralen er mottak for innringing for brann- og ulykkesmeldinger. På sentralen koordineres utrykningsenhetene og sentralen er ansvarlig for å alarmere innsatsstyrken (*110-sentralen*, u.å.).

Shapefil er et geospatialt vektor dataformat for GIS programvare. Den ble utviklet av ESRI, og er en av de mest brukte formatene for å lagre geografisk data som kart, grenser, og landemerker (*Shapefiler*, u.å.)**.**

Web applikasjon er et dataprogram som kjører på en webserver og samhandler med bruker via nettlesere. Den bruker en kombinasjon av server-side skript for å håndtere lagring og databehandling, og klient-side skript for å presentere informasjon til bruker (Wikstøm, 2023).

Widget kan legges inn og kjøre på et brukergrensesnitt, som en nettside eller et dashbord. Widgets er designet for å gi enkel tilgang til hyppig brukte funksjoner og vise informasjon i sanntid, som værmeldinger (*What Is a Widget*, 2020)

XML er et formateringsspråk som brukes for å lagre og transportere data. Markeringsspråket er en fleksibel og strukturert måte å representere data på. En egenskap XML har, er dens evne til å organisere informasjon på en måte som er lesbar for både mennesker og maskiner. XML gir en standardisert måte for applikasjoner å kommunisere med hverandre på (Rossen & Dvergsdal, 2023).

1. Introduksjon til oppgaven

På Eiendomskonferansen, arrangert av Geoforum i Bjørnafjorden kommune høsten 2022, fikk vi inspirasjon til hvilke tema vi ønsker å undersøke gjennom bachelorprosjektet vårt. På dette arrangementet deltok Bergen brannvesen med en presentasjon om bruken av GIS i deres operative kontekst. I presentasjonen belyste de begrensninger og mangler i deres kartsystem. Brannvesenets deltakelse på Eiendomskonferansen vekket interesse for å utforske hvordan en eventuell bacheloroppgave kunne belyse deres problemstillinger.

Gjennom oppstartsfasen planla vi oppgaven etter hva som er gjennomførbart med tanke på tid og kompetansegrunnlag. Videre i oppstartsfasen satt vi fokus på hvilke kartdata og programvare som kan anvendes for å oppnå våre ambisjoner og krav. I denne oppgaven har vi utviklet en prototype som gjennom programmering og visualisering, skal kunne bli et brukervennlig dashbord for nødetatene. Dette dashbordet skal fungere som en prototype for visualisering av data. Vi ønsker å vise hva som er mulig med open source-programvarer, samtidig som vi har undersøkt mulighetene med proprietær.

For å illustrere en aktuell bruk av dashbordet, testes det gjennom en case for å illustrere hvordan dashbordet kan brukes i et nødstilfelle for Bergen brannvesenet. Casen innebærer en innringing om brann- og røykutvikling i Inndalsveien 26, 5063 Bergen. Innringer befinner seg i bygning K2 på rom M515 og finner ikke veien ut av bygget. Det er meldt om økt vindstyrke den kommende timen og er dermed fare for spredning av røyk. Casen er designet for å demonstrere hvordan vår løsning fungerer i praksis og for å vise funksjonaliteten av dashbordet. Dette blir presentert i kapittel 3.3 s.67.

1.1 Bakgrunn for valg av oppgave

Brannvesenet utgjør en viktig del av samfunnets beredskapsapparat og representerer en sentral institusjon i beskyttelsen av liv, eiendom og miljø mot brann- og ulykkesrelaterte trusler. Brann- og eksplosjonsvernloven jf. § 1 fastsetter sitt formål som verner om liv, helse, miljø og materielle verdier mot brann, eksplosjoner, farlig gods og farlige stoffer, samt mot andre akutte ulykker og uønskede tilsiktede hendelser.

Bergen brannvesen fungerer som den offisielle brann- og redningstjenesten til Bergen kommune. I tillegg har de ansvarsområde som inkluderer forebyggende avdeling mot brann og ulykker (Sommerlade, 2023).

110 Vest er organisert med en arbeidsstyrke bestående av 21 ansatte, og har ansvar for koordinering av nødtelefoner og ressursutplassering i 38 kommuner i Vestland fylke (*Dette er Bergen brannvesen*, u.å.). Ettersom Bergen brannvesen har ansvar for 110 Vest og en befolkning på omtrent 600 000 innbyggere, betjener de totalt 33 brannvesen med til sammen 94 brannstasjoner. Drift og administrasjon av sentralen er basert på Bergen brannstasjon (*Dette er Bergen brannvesen*, u.å.).

Dagens bruk av kartdata er en viktig del av Bergen brannvesen sitt arbeid. Gjennom tre ulike studiedager hos 110-sentralen observerte vi hvor hyppig GIS ble brukt. Ettersom brannvesenet deles opp i beredskap, sentralen og forebyggende avdeling, er bruken forskjellig. I beredskap er de avhengig av kart som er brukervennlig og med god kartografi. I en utrykningssituasjon er tiden begrenset, og man er avhengig av et best mulig situasjonsbilde. Med et godt datagrunnlag, vil beredskap få muligheten til dette. På samme tid vil sentralen få en innringer som forblir på linjen til beredskap ankommer stedet. Dette gir de muligheten til å raskere finne frem relevant kart til den enkelte situasjon. For eksempel, ved fare for gasslekkasje, kan sentralen sette opp en buffersone for å vurdere hvem som kan bli berørt og må evakueres. Forebyggende avdeling jobber i dag med programvaren ArcGIS for å lage kart som kan være nyttig i en brann- og ulykkerelatert situasjon. Et eksempel på forebyggende arbeid, er å undersøke høyde på bygninger og veidimensjon slik at man kan vurdere innkjøp av nye brannbiler og stiger. En annen tilnærming forebyggende avdeling bruker kartdata til, kan være å kartlegge farlig gods.

I brannvesenets presentasjon på Eiendomskonferansen ga de uttrykk for at dagens systemhåndtering ikke er tilstrekkelig. Data brukes på forskjellige måter i etaten og fra ulike karttjenester. Ettersom Bergen brannvesen opererer i flere ulike tjenester, resulterer det i betydelig fragmentering av datahåndtering, noe som kan skape utfordringer med koordinering og øke risikoen for forvirring og ineffektivitet. Det vil dermed være en naturlig tilnærming å forslå en løsning på dette problemet.

Med utgangspunkt i den beskrevne bakgrunnen, sikter vår bacheloroppgave mot å forbedre håndteringen av kartdata. Det finnes muligheter for å løse det med ulike fremgangsmåter og verktøy. Vi vil gjennomføre det med ulike open source-programvarer, samtidig som vi ser på muligheten med proprietær programvare.

I oppgaven vil leseren bli introdusert for dagens systemhåndtering, problemstilling, relevant teori og mulige forbedringer. Formålet med oppgaven er å identifisere, analysere og løse mangler og potensielle forbedringer. Et delmål vi har satt oss, innebærer sammenligningen av open source- og proprietær programvare. Det overordnede målet med oppgaven er å levere et akademisk bidrag og en prototype av et dashbord. Dette skal svare på problemstillingen og samtidig presenterer en mulig forbedring av GIS i brannvesenet.

1.2. Problemstilling

Omfanget av oppgavene som brannvesenet står overfor, trenger effektive og presise løsninger. I nødssituasjoner er hastighet og nøyaktighet avgjørende faktorer, og bruken av GIS spiller en viktig rolle. GIS tilbyr muligheten til å analysere romlig data i sanntid, og kan dermed gi brannvesenet verdifull innsikt som kan bidra til bedre ressursutnyttelse i nødsituasjoner.

Vår oppmerksomhet retter seg mot spørsmålet om hvordan GIS kan benyttes og bidra i nødsituasjoner. Gjennom en grundig utforskning av potensialet innen GIS og hvordan

brannvesenet bruker ulik data, søker dette forskningsprosjektet å identifisere løsninger som kan bidra til å forbedre og styrke etatens evne til å håndtere nødssituasjoner på en rask, effektiv og målrettet måte.

Vår problemstilling er todelt: Første del omhandler hvordan open source GIS-teknologier kan tilpasses brannvesenets behov, med sikte på tilgjengeliggjøring av romlig data. Andre del sammenligner mulighetene ved proprietær programvare.

1.3 Teori

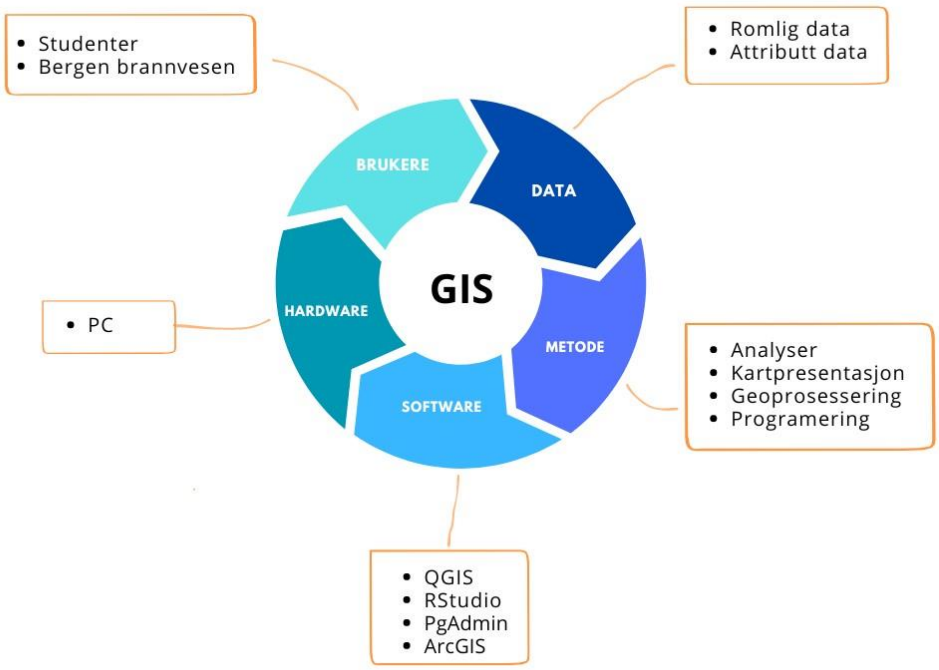
Gjennom kapittelet ønsker vi å presentere relevant teori som bygger oppunder våre beslutninger gjennom prosjektet. Dette inkluderer en gjennomgang av sentrale konsepter og modeller som er relevante for prosessen og vår forskning.

1.3.1 En innføring i Geografiske Informasjonssystemer

GIS håndterer, analyserer og visualiserer geografisk data. GIS kan koble sammen data fra ulike kilder og formater, som kan vises på ett og samme kart. Dette gir en bedre forståelse av geografisk data med dens tilhørende informasjon (What is GIS?, u.å.). Dette er relevant for vår oppgave, hvor målet er å innhente geografisk data og deretter visualisere den ved hjelp av open source-programvare.

GIS brukes i mange forskjellige felt og bransjer, enten det er arealplanlegging, naturressursforvaltning eller kartlegging. Systemet kan hjelpe til med å forstå mønstre, relasjoner og sette data i en geografisk kontekst (What is GIS?, u.å.). Geografisk data har som oftest til felles å beskrive jordoverflaten eller deler av den (Rød, 2023). GIS er i stand til å integrere data fra ulike kilder som forsterker lagringskompleksiteten. Dette gir et behov for sammenhengende og korrekt data gjennom analysene (Reddy, 2018). Samlet sett er den tekniske siden av GIS en komplisert balanse mellom datahåndtering, prosessering og analyse. For å maksimere GIS' potensial, er det viktig å forstå denne kompleksiteten og anvende riktig teknologi og metodikk for å oppnå ønskede resultater.

GIS består av fem komponenter: Data, Metode, Software, Hardware og Bruker. Figur 1 tar for seg hva komponentene kan innebære i dette bachelorprosjektet. Komponentene samarbeider for å etablere en plattform som tillater effektiv forvaltning, analyse og visuell representasjon av romlig data (*Essential Components of GIS*, 2023).



Figur 1: De fem komponentene i GIS. Deles i bruker, data, metode, software og hardware.

Komponenten om data refererer til den dataen som brukes i programvaren. Data deles opp i to kategorier: romlig-, og ikke-romlig data. Romlig data representerer geografiske attributter og lokasjoner som er visualisert på et kart. Denne type data kan inneholde informasjon om fysiske kjennetegn som høyde, terreng og arealbruk. Dataen lagres som koordinater og topologi (Reddy, 2018).

På andre siden vil ikke-romlig data, også kalt attributtdata, fungere som et bindeledd for romlig data. Denne typen data inneholder informasjon om variabler som befolkning og infrastruktur, som ikke nødvendigvis er representert på et kart, men som kan assosieres med geografiske kjennetegn (*Essential Components of GIS*, 2023). Attributtdata lagres vanligvis i form av tabeller, der hver rad og kolonne representerer en geografisk egenskap (Reddy, 2018). Dataen blir lagret i en database som i vårt tilfelle er PostgreSQL. Ved å koble PostgreSQL og R opp mot hverandre, kan vi hente ut data med ulike spørringer.

Den andre komponenten er metode. Metodikken referere til diverse analytiske og prosesseringsmetoder som anvendes for å tolke og manipulere geografisk data. Evnen til å oppnå innsikt og å utvikle løsninger, er avhengig av metodikk. Det finnes et mangfold av metoder, inkludert romlig analyse, kartografi og datavisualisering, for å nevne noen (*Essential Components of GIS*, 2023). Gjennom oppgaven har vi brukt ulike metoder, noe som blir nærmere forklart i kapittel 2 s.28.

Programvare utgjør en sentral komponent i GIS, og brukes til å håndtere, analysere og visualisere geografisk data. Applikasjonene muliggjør utførelsen av romlige analyser,

kartproduksjon og andre GIS-relaterte oppgaver for brukeren (*Essential Components of GIS*, 2023). Det finnes en bred variasjon av tilgjengelig programvare, inkludert open source, som er noe vi har fokusert på i utarbeidelsen av vårt dashbord. Med open source programvare tillater det alle å inspisere, modifisere og forbedre programvarens kildekode. Open source-programvare eliminerer ofte behovet for lisens (*Hva er åpen kildekode?*, u.å.). Eksempler på programvarer uten lisens inkluderer blant annet QGIS, pgAdmin og RStudio. På andre siden finnes det også programvare med proprietær kildekode, som for eksempel ArcGIS. Dette refererer til en type programvare hvor utvikleren beholder eksklusiv kontroll over kildekoden (*Proprietær vs Åpen Kildekode*, u.å.). Fordelen med slike programvarer er tilgjengeligheten av omfattende dokumentasjon, men en ulempe kan være høye kostnader forbundet med bruk (*Essential Components of GIS*, 2023). Det er derfor lagt vekt på anvendelse av open source-programvarer i oppgaven.

Maskinvare, eller hardware, refererer til de fysiske verktøyene og utstyret som er nødvendige for innsamling, lagring og behandling av geografiske data. Dette kan inkludere datamaskiner, servere og GPS-enheter. Maskinvaren skal være i stand til å håndtere store datamengder og kjøre programvareapplikasjoner uten forsinkelser (*Essential Components of GIS*, 2023). I vårt prosjekt har vi brukt våre bærbare datamaskiner, men i en eventuell videreutvikling av arbeidet hadde det vært gunstig med større kapasitet på maskinvaren.

Brukeren har en sentral rolle i arbeidet med innsamling og validering av geografiske data. Dette muliggjør utførelsen av analyser, tolkning av resultater og utledning av konklusjoner basert på undersøkelsene. Uten brukerens innsats ville det være utfordrende å forstå betydningen av resultatene. Bruker kan være den som er engasjert i prosjektets gjennomføring, samt den som benytter det endelige produktet (*Essential Components of GIS*, 2023). I vår oppgave er det vi som er bruker, da vi la ned innsats ved å samle inn og analysere data. Likevel utvikler vi et produkt for Bergen brannvesen, så de vil være en potensiell fremtidig bruker.

1.3.2 Kart- og webapplikasjoner

GIS har en sentral rolle i kart- og webapplikasjon. En kartapplikasjon er en type programvare som er spesialisert for å vise geografisk informasjon på et kartgrensesnitt. Applikasjonene er designet for å gi bruker muligheten til å utforske geografiske data (Nottveit, 2009). Kartapplikasjoner kan variere fra mindre avanserte verktøy som viser en enkelt kartvisning, til mer avanserte systemer som tilbyr funksjoner som geoprosessering, lagring og deling av kartdata (What are Application Maps?, u.å.).

Webapplikasjon er et verktøy som kjøres på en webserver via en nettleser. Webapplikasjoner kan ha ulike formål, som eksempelvis visualisering og analyse av data. Forskjellen mellom en kart- og webapplikasjon ligger hovedsakelig i fokusområdet og funksjonaliteten. Mens en kartapplikasjon fokuserer på å vise karttjenester, er en webapplikasjon et bredere begrep

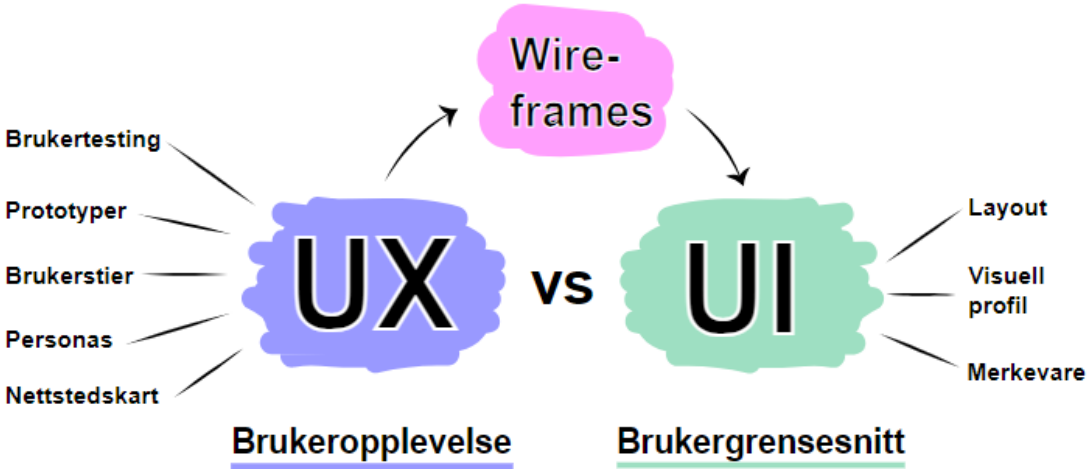
som inkluderer ulike tjenester og funksjoner tilgjengelig gjennom en nettleser (*What is Web Application*, u.å.).

I dette prosjektet er kartapplikasjon brukt til å vise ulike typer geografiske data og funksjonaliteter, som for eksempel muligheten til å zoome inn og ut, legge til kartlag, og å utføre geoprosessering. Kartapplikasjonen blir en sentral del av dashbordet, som brukes til å visualisere og manipulere data. I prosjekt brukes en webapplikasjon for å gjøre kartapplikasjonen tilgjengelig på tvers av forskjellige plattformer, noe som sikrer at alle brukere enkelt kan få tilgang til systemet. Fordelen med å bruke en webapplikasjon for å lage et kartgrensesnitt, er at det er tilgjengelig og fleksibelt for bruker (Wikstøm, 2023). Figur 2 viser Norgeskart som er et eksempel på en kartapplikasjon som brukes via en webapplikasjon. Applikasjonen er levert av Kartverket. På Norgeskart oppleves det som lett å orientere seg, da det er et oversiktlig brukergrensesnitt.

*Figur 2: Utklipp fra Norgeskart for å illustrere kart- og webapplikasjon (*Norgeskart*, 2024).*

Ved utvikling av webapplikasjoner er brukeropplevelse og brukergrensesnitt relevant. Brukeropplevelse, også kalt User Experience Design (UX-design), handler om å utvikle gode digitale løsninger. Jo enklere det er å navigere seg i dashbordet, jo bedre helhetsinntrykk vil man sitte igjen med. Brukergrensesnitt er det som gjør det mulig å kommunisere med en enhet via menyer, ikoner eller knapper. Dette er også kalt User Interface Design (UI-design). Hvordan en nettside er visuelt utformet har mye å si. Komponentene er avgjørende for å sikre at webapplikasjonen fungerer pålitelig og effektivt, samtidig som den tilbyr en brukervennlig opplevelse for sluttbrukeren (Aaberge, 2021).

Figur 3 illustrerer UX kontra UI. UX fokuserer på hvordan bruker navigerer og beveger seg rundt på nettsiden, med mål om å gi en sømløs opplevelse og unngå hindringer. Mens ved UI vektlegges fremstillingen til nettsiden, inkludert valg av farger, skrifttyper og bilder. Det er viktig å sikre at designet på nettsiden er i tråd med formålet til oppgaven, slik at man formidle et helhetlig inntrykk (Aaberge, 2021).



Figur 3: Bildet illustrerer UX og UI (Grolid & Albertine, u.å.).

Det er mulig å lage en kartapplikasjon inne i en webapplikasjon. Vårt dashbord kan åpnes i en nettleser, hvor bruker får tilgang til ulike kart. Ved å integrere et kartgrensesnitt i en webapplikasjon kan man utforske og visualisere geografiske data direkte fra nettleseren. Ved å kombinere kart- og webapplikasjon kan man dra nytte av fordelene ved begge tilnærmingene.

1.3.3 Strukturering av romlige databaser

Ettersom oppgaven bygger på en database i PostgreSQL, vil den bakenforliggende struktur for dataen være viktig for effektiv bruk. En database består av en sammensetning av ulike filtyper. Filene er organisert innen et felles system som er spesifikt tilpasset dets formål. Variabler som antall brukere, størrelse og organisasjonstype kan variere betydelig mellom ulike databaser (*Hva er en geodatabase?*, u.å.).

Databasen er hovedsakelig basert på prinsippene for relasjonsdatabaser. Tabeller og attributt-typer anvendes for å lagre skjemaer, regler, baser og romlige attributtdata. Med bruk av programmeringsspråk som SQL, er det mulig å skape, modifisere og utføre spørringer mot de ulike tabellene i databasen (*Arkitekturen til en geodatabase*, u.å.).

Hver geografisk entitet er strukturert som en tabell, hvor hver rad i tabellen symboliserer én spesifikk funksjon. Når geodatabasen lagres i PostgreSQL, blir de romlige representasjonene av vektor- eller raster-data lagret. Dette realiseres gjennom bruk av SQL.

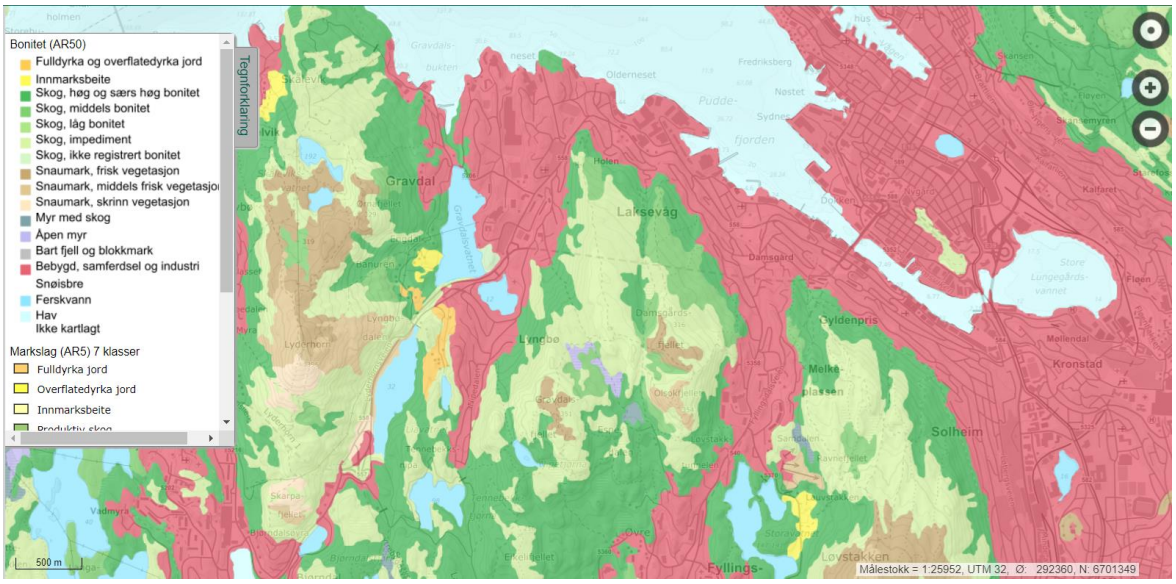
Som nevnt i kapittelet 1.3.1 s.12, er geografisk data strukturert i to hovedkategorier: romlig- og ikke-romlig data. Attributtdata inneholder beskrivende informasjon om geografiske elementer. Romlig data, derimot, inneholder posisjonsrelatert informasjon, gitt ved koordinater og geometriske former som punkter, linjer eller polygoner (Rød, 2023).

Attributtdata kan være enten kvalitative eller kvantitative. Kvantitative data opererer på tre ulike målenivåer: ordinal, intervall og forhold. Med ordinalnivået blir verdier rangert, men det finnes ingen klar avstand mellom klassene slik det gjør med intervallnivå. Intervallnivået blir

også gjenkjent ved at det ikke er et fast nullpunkt. Dette har derimot forholdstallsnivået, som også inkluderer et bestemt forhold mellom verdiene (Grønmo, 2024). Dataen tilfører kontekst og detaljer til de romlige elementene i datasettet. Koblingen mellom attributtdata og romlig data er avgjørende i GIS, da den muliggjør utførelse av analyser og visualiseringer som gir dypere innsikt i geografiske fenomener(Rød, 2023).

En annen viktig sammenheng mellom romlig- og egenskapsdata ligger i forståelsen av topografiske kart ved hjelp av erfaring eller kartforklaring. For eksempel, vanligvis er vann representert med blå farge. Denne konvensjonen tillater oss å intuitivt forstå sammenhengen mellom geografisk data og den virkelige verden, og gjør det lettere å tolke og analysere data (Rød, 2023).

Kvalitativ data, også kjent som nominaldata, innebærer å beskrive forskjeller mellom ulike elementer på et kart. Nominaldata tillater kategorisering av områder med kun én klassifisering om gangen. For eksempel kan et skogområde bli klassifisert som en grønn flate, mens en innsjø kan representeres som en blå flate. Figur 4 er et eksempel på en slik klassifisering fra Nibios Gårdskart. Kategoriene kan videre deles inn i polygoner, hvor ulike kvaliteter, som for eksempel bonitet, kan tilskrives forskjellige deler av området (Rød, 2023).



*Figur 4: Utklipp fra Nibios Gårdskart som viser ulike klassifiseringer (*Gårdskart - NIBIO*, 2024).*

1.3.4 Kartografiske designprinsipper

Kartografiske designprinsipp har vært et sentralt konsept gjennom utviklingen av en applikasjon. Det er viktig med god balanse mellom designprinsippene ettersom vi skal utvikle et dashbord som må fremstå brukervennlig. I dashbordet ønsker vi å presentere ulike kart og funksjoner som skal gi god innholdsforståelse av data. Ved kartografiske designprinsipp skilles det primært i fem prinsipper: lesbarhet, visuell kontrast, figurorganisering, hierarkisk organisering og balanse.

Prinsippene samvirker for å skape et system som muliggjør identifisering og forståelse av innholdet på kartet. Fravær av prinsippene kan føre til mislykket situasjonsforståelse. Det er viktig å merke seg at prinsippene ikke opererer isolert, men heller utfyller hverandre. Samlet sett bistår de kartografer i å utvikle kart som effektivt formidler geografisk informasjon (Aileen, Buckley, 2012). Hovedformålene med prinsippene er å dele informasjon, fremheve strukturer og prosesser, og illustrere et sammenhengene resultat (Longley et al., 2011). Nedenfor vil det bli gitt nærmere teori om de fem designprinsippene.

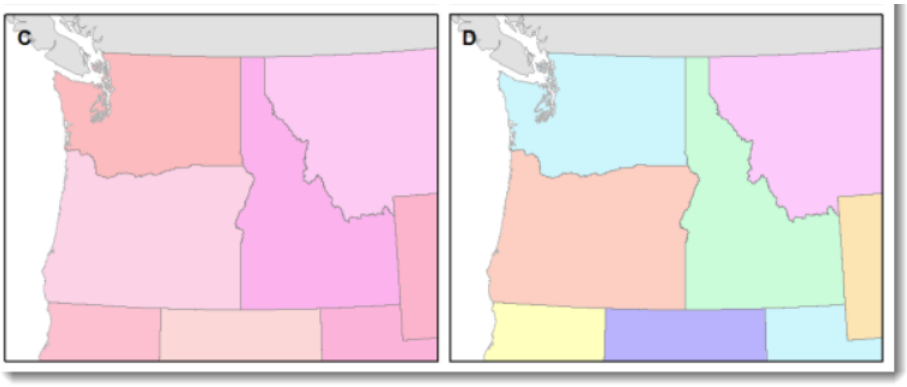
Lesbarhet defineres som "evnen til å bli sett og forstått" (Buckley, 2011). Kartinnholdet og sideelementene skal være tydelige og synlige, men det er likevel avgjørende at de også er forståelige. Valg av symboler er viktig med tanke på lesbarheten. Passende størrelser og form gjør resultatene lettere å forstå. Geometriske symboler er generelt lettere å lese når de er av mindre størrelser, mens mer komplekse symboler krever større plass for å være lettleste (Buckley, 2011), (Longley et al., 2011).

Et annet prinsipp som er viktig, er visuell kontrast. Det refererer til hvordan kartfunksjoner og sideelementer skiller seg ut fra hverandre og bakgrunnen (Longley et al., 2011). For å illustrere prinsippet, kan man vurdere evnen til å se klart i mørke omgivelser. Da mottar øynene begrenset reflektert lys, noe som resulterer i liten visuell kontrast mellom ulike objekter. Ved tilførsel av mer lys, øker imidlertid kontrasten mellom funksjonene og bakgrunnen (Buckley, 2011).

Prinsippet om visuell kontrast er også anvendelig innen kartografi. Et godt utformet kart med høy visuell kontrast kan gi et klart, tydelig og skarpt bilde. Jo større kontrast det er mellom funksjonene, desto mer fremtredende vil visse elementer være. Omvendt kan et kart med lav visuell kontrast benyttes for å skape et mer subtilt inntrykk, der funksjoner med mindre kontrast vil oppleves å være mer sammenhengende (Buckley, 2011).

Figur 5 er et eksempel på hvordan visuell kontrast kan påvirke et kart. I eksempel C er det brukt variasjoner i én fargetone for å skape visuell kontrast, men i eksempel D er det brukt flere ulike fargenyanser. Eksempelet viser hvordan visuell kontrast kan påvirke oppfatningen av et kart og tydeliggjøre informasjon (Buckley, 2011).

Visuell kontrast og lesbarhet danner grunnlaget for synliggjøring. Utover evnen til å skille funksjoner fra hverandre og fra bakgrunnen, må funksjonene også være tilstrekkelig store til å bli sett og forstått. Videre kan visuell kontrast og lesbarhet også benyttes for å støtte de andre designprinsippene (Buckley, 2011).



Figur 5: Bildet viser et eksempel på bruk av et kartografisk designprinsipp: visuell kontrast. Dette kan bidra til tydeliggjøring av kart for leser. Bildet er hentet fra ESRI.

Hierarkisk organisering på kart viser plasseringen av ulike fysiske og kulturelle trekk som terreng, veier, grenser og bosetninger. På kartet bør ikke funksjonene fremstå som viktigere enn andre, og dermed bør de visuelt sett ligge på omtrent samme nivå (Longley et al., 2011). Hierarkiet på kart er vanligvis mer subtilt, og kartleseren bringer elementene i forgrunnen ved å rette oppmerksomheten mot dem. På tematiske kart er temaet viktigere enn den geografiske konteksten som kartet bygger på (Buckley, 2011).

Balanse omfatter organiseringen av kartet og andre komponenter, som tittelfelt, nord-pil og tegnforklaring. En balansert kartside etterlater et inntrykk av harmoni og likevekt. Balanse brukes for å gi et mer naturlig inntrykk og oppstår som et resultat av to hovedfaktorer: visuell vekt og retning. Forestill deg at midten av kartet er som et punkt der alt holder seg i balanse. Hva som får kartet til å lene seg i en retning, avhenger av elementenes plassering, form, størrelse, og tilhørende tema (Buckley, 2011).

Figurbakgrunn er det femte designprinsippet. Det er den naturlige separasjonen av figuren i forgrunnen fra en uklar bakgrunn. Kartografer bruker dette designprinsippet for å hjelpe kartlesere med å fokusere på et bestemt område av kartet. Det er flere måter å fremme figur-bakgrunn på, for eksempel ved å legge til detaljer på kartet eller bruke en lysning, en skygge eller fjæringsteknikk på bakgrunnen (Buckley, 2011).

Figur 6 illustrerer viktigheten av en figurbakgrunn gjennom fire ulike eksempel. På eksempel A ser vi at det er brukt lukkede former, som gjør det vanskeligere å lese av kartet. Mens i eksempel B er det brukt hvit vask for å fremheve kartet fra bakgrunnen. I eksempel C er det brukt skygge for å skape en illusjon av dybde i kartet, mens eksempel D tar i bruk fjæring. Fjæring i kartografi refererer til en teknikk der kantene på figurer gjøres mykere eller mindre synlige for å skape en mer harmonisk overgang mellom figuren og bakgrunnen. Dette gjør kartet lettere å lese og reduserer distraksjoner ved å minske kontrasten mellom figuren og bakgrunnen (Aileen, Buckley, 2012).

Figur 6: Bildet viser et eksempel for figurbakgrunn (Aileen, Buckley, 2012).

1.3.5 Programvare for GIS-analyse

I utviklingsarbeidet har valg av programvare vært viktig. Vi har primært valgt å jobbe med open source-programvarer, men har parallelt med dette arbeidet undersøkt muligheter med proprietære programvarer i kapittel 2.4 s.51. Dette er på bakgrunn av lisenser, kostnader og tilgjengelighet.

Bruk av open source-programvarer gir mange muligheter. Vi har brukt QGIS, R/RStudio og pgAdmin. QGIS gir mulighet til å bearbeide og visualisere romlig data, som gjennom oppgaven har vært fundamental for å få data inn i PostgreSQL. Ved uthenting av data har vi fått ulike filformater, men vi har primært brukt shapefiler (*Hva er åpen kildekode?*, u.å.).

Databasen er opprettet i PostgreSQL som er et database-håndteringssystem for både romlige og ikke-romlige data. PgAdmin er et verktøy som håndterer romlig data og er fleksibel for ulike typer databehov (*Introduction to PostgreSQL*, u.å.). Ved å kunne jobbe med databasen i PostgreSQL med pgAdmin som brukergrensesnitt, blir det lettere anvendelig gjennom utarbeidelsen av applikasjonen.

En annen open source-programvare vi har brukt, er RStudio. RStudio er et ideelt verktøy for denne oppgaven fordi det tilbyr et integrert utviklingsmiljø hvor vi kan benytte programmeringsspråket R. RStudio fungerer som et brukergrensesnitt hvor vi bruker R for å opprette selve dashbordet. R støtter rammeverket Shiny, som lar oss bygge interaktive webapplikasjoner direkte fra R. Det er nyttig for oppgaven for å presentere komplekse dataanalyser på en tilgjengelig måte. R har verktøy for datahåndtering som enkelt kan integreres med databaser som PostgreSQL. Vi har valgt å bruke R fremfor Python, på bakgrunn av R sitt rike univers for å håndtere romlig data. Med et stort bibliotek av pakker og en aktiv brukerbase, tilbyr R støtte for romlige analyser via pakker som *sf* og *leaflet*. Derfor ble R et fornuftig valg for å effektivt utføre og presentere analyser i tekniske prosjekter som

involverer avansert datainnsikt (Kosourova, 2022). Programvarene som er brukt har til felles at de er kostnadsfrie og gir pålitelig datahåndtering. Dette gjør det også lettere å videreutvikle ved en senere anledning.

1.3.6 Programmeringsspråk

Ved valg av programmeringsspråk er det relevant å undersøke hvorfor R er gunstig for vårt arbeid. R er et programmeringsspråk som er spesielt utviklet for statistisk analyse og grafisk fremstilling av data, noe som gjør det til et godt valg for oppgaven, som krever databehandling og komplekse analyser. Det rike biblioteket av pakker, som *ggplot2* for avanserte visualiseringer og *dplyr* for datamanipulasjon, tilbyr verktøy som er gunstig for å håndtere og analysere store datasett. Dette er spesielt relevant i GIS og andre områder som krever databehandling, hvor trender typisk avdekkes gjennom bruk av avanserte statistiske modeller og visualiseringer (Smith & Venables, 2024).

I tillegg er R også høyst relevant for sin evne til å integrere med andre verktøy og plattformer. Dette er noe som har vært relevant for oppgaven ettersom vi har brukt flere ulike programvarer. R kan enkelt kobles til ulike databaser og GIS-verktøy, som PostgreSQL og PostGIS, noe som muliggjør en smidig arbeidsflyt der data kan flyte fritt mellom analyse og lagringssystemer. Dette gjør det mulig for oss å utføre sanntidsanalyser og håndtere romlig data (Smith & Venables, 2024).

Programmeringsspråket R er særlig gunstig for vår oppgave fordi det tilbyr pakker som Shiny. Shiny kan brukes til å utvikle interaktive webapplikasjoner, noe som er ideelt for å presentere geografiske data på en brukervennlig måte. Videre er R gunstig for analyse og datahåndtering, særlig for å behandle og analysere store og komplekse datasett (Smith & Venables, 2024).

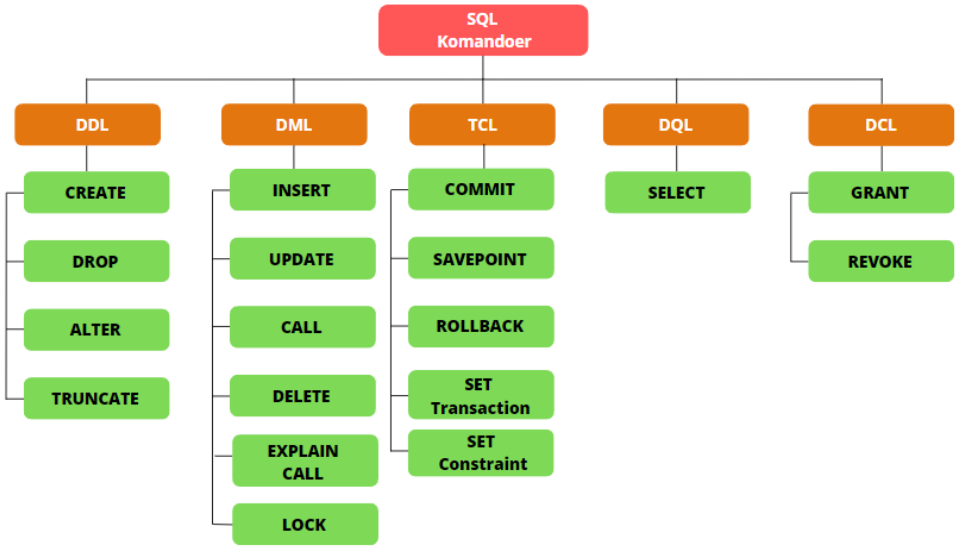
Structured Query Language (SQL) er et programmeringsspråk som brukes til å hente og oppdatere informasjon fra relasjonsdatabaser (Bratbergsengen et al., 2023). SQL har fem ulike underspråk som har hver sine funksjoner for hvordan man manipulerer og undersøker data (*SQL-kommandoer*, 2023).

Data Definition Language (DDL) brukes til å definere databasens skjema. Det håndterer beskrivelsen av databasestrukturen og brukes til å opprette, endre og slette strukturen til databaseobjektet, men ikke selve dataen (*SQL-kommandoer*, 2023).

Data Query Language (DQL) blir brukt til å utføre spørringer på data i databasen. Formålet med kommandoen er å hente data basert på spørringen. SELECT er kommandoen som brukes for å hente ønsket data fra databasen. (*SQL-kommandoer*, 2023). Vi har gjort spørringer direkte i PostgreSQL eller i R ved hjelp av pakkene *DBI* og *RPostgres* (Wickham & Muller, u.å.).

Data Manipulation Language (DML) tillater utførelse av operasjoner som setter inn, sletter og oppdaterer data i en eksisterende tabell. Data Control Language (DCL) administrerer tilgangstillatelser for bruker eller roller, og opprettholder sikkerheten i databasen. Transaction Control Language (TCL) administrerer transaksjoner i databasen, og inkluderer blant annet kommandoer for å starte og angre transaksjoner (*MySQL*, u.å.).

Valget av riktig kommando avhenger av oppgaven man ønsker å utføre i databasen. Figur 7 er en oversikt over kommandoene, og deres tilhørende funksjoner.



*Figur 7: Oversikt over SQL kommandoer. Inspirasjon for figur hentet fra (*SQL Commands*, u.å.).*

1.3.7 Dataformater: Lagring og utveksling av geografisk data

Et filformat definerer strukturen for hvordan data lagres for en bestemt applikasjon. En applikasjon kan støtte flere typer filformater, men det er ikke garantert at det kan åpne alle. Det er også mulig for et program å lese deler av en fil, som eksempelvis teksten, men ikke formateringen. Det er derfor essensielt at man finner data tilpasset sin bruk og programvare (*Lær om filformater*, u.å.).

Man skiller mellom åpne og proprietære formater. Åpne formater er tilgjengelige for allmennheten, og muliggjør at informasjon om lesing og skriving av data er transparent. Dette tillater bruker å konstruere applikasjoner som direkte kan håndtere formatet. På den andre side vil proprietære formater være mer lukkede. Det eksisterer vanligvis restriksjoner angående tilgang og datahåndtering i formatet. Konvertering av filer uten manuell kopiering av informasjonen mellom applikasjoner kan være vanskelig. Videre kan det oppstå risiko for tap av data hvis støtte for den spesifikke lagringsapplikasjonen opphører. Proprietære filformater er dermed avhengige av enkeltpersoner eller organisasjoner som opprettholder standarden. Tilgang til slike formater er vanligvis underlagt en lisensavtale (Nätt & Liseter, 2023).

Gjennom forskningsprosjektet har vi brukt filformatene Shapefil, DATEX og SQLite. Alle formatene håndterer romlig data, men på ulike måter.

Shapefil er et vektordatalagringsformat for lagring av plassering, form og attributter med geografiske egenskaper. Shapefiler kan representere geografiske funksjoner som punkter, linjer eller polygoner. De kan eksempelvis beskrive lyktestolper, veier og innsjøer. Hver form i en shapefil har vanligvis attributter som gir mer informasjon, for eksempel navn, temperatur og areal (*Shapefiler*, u.å.). Formatet består av en samling filer med en felles stamme, lagret i samme katalog. De tre obligatoriske filene har filutvidelse .shp, .shx og .dbf. Den faktiske shapefilen refererer spesifikt til .shp-filen, men alene er den ufullstendig for distribusjon, da de andre støttende filene også er nødvendige. Det er mulig å lese og skrive geografiske datasett ved hjelp av shapefile-formatet med en rekke forskjellige programvarer (*Shapefiler*, u.å.).

Data Exchange between traffic and travel information centres (DATEX) er en europeisk standard for trafikk- og reiseinformasjon mellom ulike aktører. Denne standarden baserer seg på XML-meldinger (*Hva er DATEX?*, u.å.). DATEX muliggjør effektiv deling av trafikkrelatert informasjon, inkludert værforhold, reisetider, kamerabilder og trafikkmeldinger (*User guide, DATEX*, u.å.). Bruk av DATEX krever tilgang. Vi forespurte tilgang via Statens vegvesen sine sider. Når man får tilgang kan man ta i bruk publikasjonene som tilbys. Tjenesten er gratis og tilgjengelig for ulike distributører av trafikkdata og andre aktører som ønsker å benytte informasjon til egen tjeneste (*User guide, DATEX*, u.å.). Bruken av DATEX er regulert av norsk lisens for offentlig data, utarbeidet av Digitaliseringsdirektoratet (*NLOD, 2.0*, 2023).

SQLite er en innebygd SQL-database med mange applikasjoner. Databasen er spesielt designet for å være brukervennlig og krever ingen separat serverprosess. I stedet leser og skriver den direkte til vanlige diskfiler. Koden er et offentlig domene, som betyr at den er gratis for bruk til ethvert formål (*SQLite*, 2023). Dens format eliminerer behovet for en separat serverprosess og gjør direkte skriving til vanlige diskfiler mulig, i motsetning til flere andre SQL-databaser (*SQLite*, 2023). Ettersom en SQLite-database ikke har et behov for administrasjon, er den godt egnet for enheter som må operere uten behov for personell med høy kompetansebakgrunn. Med SQLite er det enkelt å overføre innhold fra et system til et annet, hvor mottaker kan benytte SQL for å hente ut innhold etter behov (*Appropriate Uses For SQLite*, 2024).

Mer informasjon om hvordan vi har anvendt formatene og dens påvirkning på oppgaven kommer vi tilbake til i kapittel 2.1.3 s.31.

1.3.8 Anvendelse og potensial for KI-teknologi innen Geografiske Informasjonssystemer

Ved å bruke kunstig intelligens som Copilot og ChatGPT i utviklingsprosesser, spesielt i arbeid med R og dashbordkonstruksjon, åpnes det for en mer dynamisk og effektiv utviklingsprosess.

KI-verktøyenes evne til å analysere, foreslå syntaksrettelser og tilby forbedringsforslag, kan effektivisere utviklingsarbeidet. Systemene kan ikke bare identifisere feil, men også foreslå

mer effektive kodestrukturer og algoritmer, som kan øke kvaliteten og ytelsen på den endelige løsningen. Slik bidrar KI til en mer intuitiv utviklingsprosess, hvor vi får verktøyene til å raskt forbedre prosjektet (*GitHub Copilot*, 2024) (*Introducing ChatGPT*, 2022).

Gjennom oppgaven har KI bidratt til bedre forståelse innen programmering. Nærmere om dette i kapittel 2.2.3 s.37. Ved å benytte naturlig språkbehandling, er verktøyene i stand til å forstå brukerspørsmål og foreslå kodeeksempler, feilsøkingstips og hjelp innen programmering. Denne direkte tilgangen til kunnskap og løsninger sparer verdifull tid og ressurser (*Introducing ChatGPT*, 2022). Det er likevel viktig å merke seg at kunstig intelligens skal kun fungere som et støtteverktøy. Det er ikke en kunnskapsbase, og de løsningene som KI presenterer er det viktig å kontrollere opp mot en ekstern kilde.

Selv om kunstig intelligens er et nyttig verktøy for hjelp, er det viktig å merke seg at man er avhengig av å ha en kritisk tilnærming.

1.3.9 Romlig indeksering

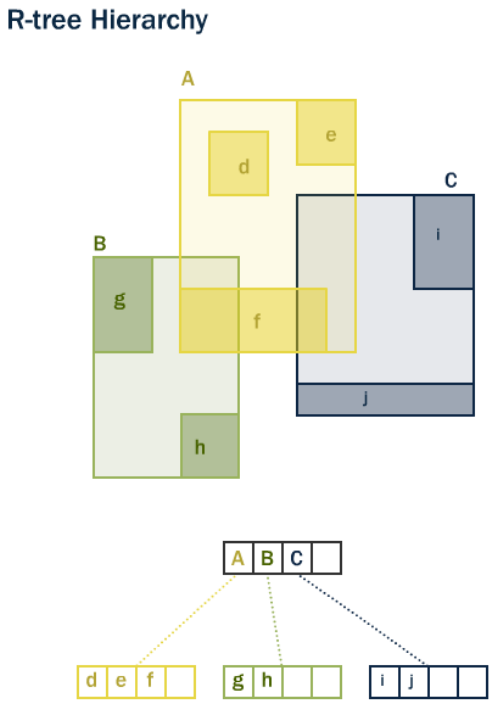
I løpet av dette forskningsprosjektet har teorien om romlig indeksering vært av vesentlig betydning for utviklingen av dashbordet. Ved implementering av PostgreSQL i kombinasjon med romlige data, har det blitt nødvendig å utvide databasens funksjonalitet gjennom integrasjon av PostGIS. PostGIS er en utvidelse for romlig dataprosessering (*PostGIS*, u.å.). Mer om hvordan vi har brukt PostGIS i kapittel 2.2.1 s.35 og 2.3.7 s.50. Det muliggjør håndtering av tabeller som inneholder romlige attributter. For å øke effektiviteten av våre databasespørringer, har vi implementert romlig indeksering av tabellene.

En database uten indeksering kan sammenlignes med å søke på et ord i en bok uten register. Søket vil da ta for seg side for side. Ved å indeksere databasen kan vi se for oss at boken nå har en innholdsliste. Dette muliggjør at man kan søke i ett kapittel istedenfor i hele boken (Worboys & Duckham, 2004b).

Når vi indekserer ikke-romlige databaser, bruker vi en teknikk kjent som B-trær. Dette innebærer at vi organiserer tabellen i flere nivåer. For eksempel, hvis en tabell inneholder 1 000 000 rader, kan indekseringen dele denne opp slik at de første 100 000 radene utgjør ett segment, de neste 100 000 et annet segment, og så videre opp til en million. Hvis vi da ønsker å finne rad nummer 109 534, kan vi med indeksering begrense søket til segmentet fra 100 001 til 200 000 rader i stedet for å søke gjennom hele millionen. Indeksering gjør spørringen raskere ved å effektivisere lagring, prosessering og visualisering av data.

Romlig indeksering fungerer på en lignende måte som tradisjonell indeksering, men fokuserer på de geometriske egenskapene til dataen i en tabell. For eksempel, hvis vi ønsker å finne ut hvor mange punkter som er innenfor en bestemt polygon, lar romlig indeksering oss avgrense søket for å effektivt identifisere punktene som overlapper med polygonen. Uten romlig indeksering ville databasen sjekket alle punkter mot alle polygoner, noe som er mindre effektivt. Romlig indeksering sparer derfor ressurser ved å være mer målrettet i sine søk, noe

som er svært nyttig i større databaser med mange brukere. I PostgreSQL benyttes databaseutvidelsen PostGIS som bruker «GiST» (Generalized Search Tree), en type R-tre. R-treet er en balansert trestruktur hvor hver node representerer en bounding box, et rektangel, som effektivt kan redusere omfanget av søket ved å eliminere grener som ikke berører det aktuelle området (*Romlig indeksering*, u.å.). Figur 8 er en illustrasjon av R-tre hierarkiet.



*Figur 8: Bildet illustrerer R-tre hierarkiet inne romlig indeksering (*R-tree Hierrarchy*, u.å.).*

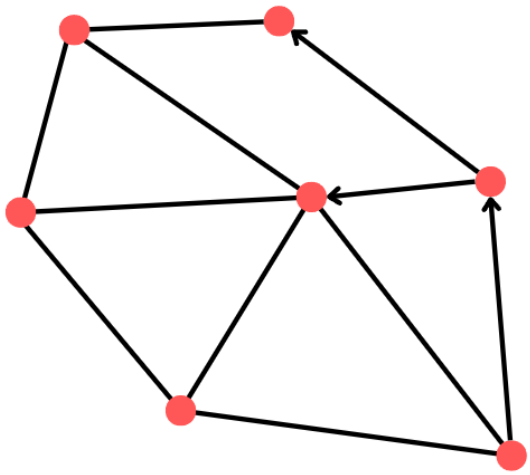
1.3.10 Romlige analyser

Romlig analyse refererer til arbeidet med å forstå geografisk plassering og mønstre ved hjelp av romlig data. En viktig del er å forstå hvordan ulike geografiske enheter er plassert i forhold til hverandre. Dette kan bestå av punkter, linjer og polygoner (Vijay, 2022). Gjennom oppgaven har vi vært innom ulike romlige analyser, som blant annet nettverksanalyse, overlayanalyse og bufferanalyse.

Nettverksanalyse er en metode for å studere relasjonene mellom enheter i et nettverk. Det innebærer å analysere forbindelsene mellom enhetene og tilhørende attributter. Nettverksanalyse kan blant annet brukes til å finne korteste vei fra ett sted til et annet (*What Is Network Analysis?*, 2023).

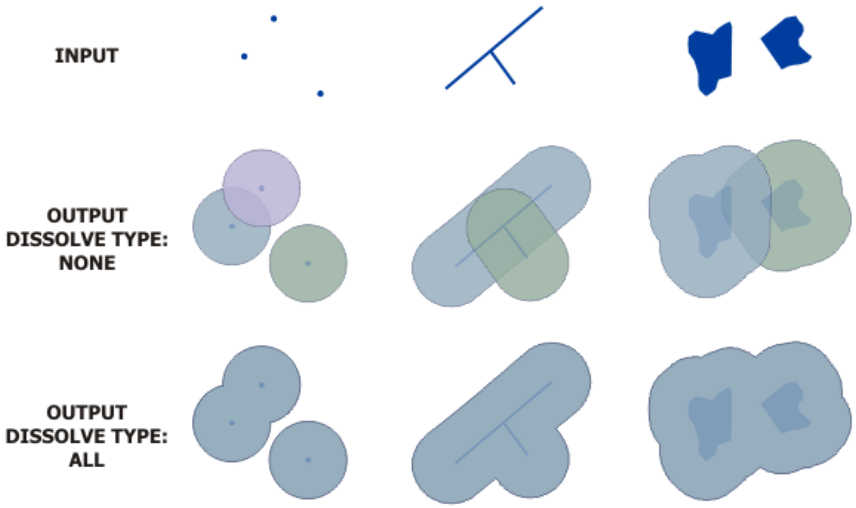
Et nettverk eller en graf er bygget opp på en samling av noder, og kanter som kobler nodene sammen (Eilertsen, 2023). Grafen kan inneholde rettede og urettede kanter. Urettede kanter kan sammenlignes med veier som kan kjøres i begge retninger, mens rettede kanter kan sammenlignes med enveiskjørte veier (*What Is Network Analysis?*, 2023). I en

veinettverksanalyse er hensikten å finne korteste rute fra én adresse til en annen. Hvis deler av den foreslåtte ruten består av rettede kanter og dermed ikke kan benyttes i begge retninger, må analysen finne en alternativ vei for å komme fra start til slutt. Dette konseptet er blant annet relevant for trafikkplanlegging. Figur 9 viser en illustrasjon av et nettverk. De røde punktene illustrerer node, linjene med pil illustrerer rettede kanter og de rette linjene urettede kanter.



Figur 9: Illustrasjon av et nettverk. Rød prikk illustrerer noder og linjene illustrerer grafer.

Bufferanalyse er bestemmelse av en sone rundt et geografisk objekt som for eksempel en vei. Man bruker bufferanalyse for å visualisere og analysere områder som ligger innenfor en viss avstand fra det enkelte geografiske objekt (*Bufferanalyse*, u.å.). En buffer blir ofte representert med en polygon slikt som vist på Figur 10.



Figur 10: Utklipp fra ArcGIS, viser hvordan en buffer er bygd opp rundt punkt, linje eller polygon.

Overlayanalyse representerer en metodisk tilnærming for å identifisere optimale områder for ulike formål. Denne analysen innebærer sammenslåing av flere geografiske kartlag. Gjennom denne integrasjonen kan vi oppnå dyp innsikt ved å avdekke sammenhenger, identifisere

mønstre og utforske komplekse relasjoner mellom ulike geografiske faktorer. Implementeringen av overlayanalyse innebærer bruk av ulike geometriske operasjoner, som for eksempel *union*, *intersect* og *buffer*. Operasjonene muliggjør sammenføyning av kartlagene og genererer dermed et nytt datalag som gir et mer omfattende og helhetlig bilde for det ønskede resultat (*Understanding overlay analysis*, u.å.).

1.4 Omfang og avgrensninger

Etter å ha gjennomført tre studiedager hos 110-sentralen Vest, har vi fått et innblikk i hvordan de ulike plattformene fungerer. Dette ga oss en oversikt over hva omfanget innebar. En av de sentrale plattformene i dagens drift er Locus Emergency 110. Dette systemet brukes til å motta varsler om utløst alarm fra automatiske brannalarmanlegg, som sendes til sentralen og brannbilene. Parallelt med arbeidet som foregår i Locus, arbeides det også i andre plattformer.

Omfanget av oppgaven er bredt, men utviklingsarbeidet vil fokusere på utarbeidelsen av en prototype av et dashbord. I dag opererer Bergen brannvesen med flere separate applikasjoner. Utviklingen av en prototype vil gi tilgang til relevant informasjon på en samlet plattform. Omfanget av oppgaven vil fokusere på utarbeidelsen av dashbordet, med fokus på teori, sammen med programvarene som er brukt. For å besvare oppgaven best mulig, har vi gjort avgrensinger i oppgaven. Det er på bakgrunn av tid, ressurskapasitet og tilgang på programvarer.

En avgrensning som vi har valgt å gjøre, er basert på datastørrelse. Ved å begrense datasettet til kun Bergen kommune, muliggjør det raskere generering av resultater sammenlignet med en tilnærming som omfatter hele Vestland fylkeskommune. Dette vil redusere mengden data som programvaren må håndtere. Dette på bakgrunn av serverstørrelse og datakapasitet som vår maskinvare klarer å håndtere. På grunn av begrensninger i serverkapasitet, hindrer det muligheten for større søk på fylkesnivå. Lokale databaser kan ha begrensninger knyttet til lagringsplass, tilgjengelighet og sikkerhet. Dette kan føre til dårligere ytelse, begrenset kapasitet og økt risiko for nedetid. Det kan også være vanskelig med å oppskalering etter behov (Bratbergsengen & Mallaug, 2024).

Andre avgrensninger som er relevant for oppgaven er knyttet til rettigheter. I noen tilfeller kreves det rettigheter for å få tilgang til ulike datasett. Dette gjelder blant annet ulike sjøkart, togtrafikk eller automatiske brannalarmer i offentlige bygg. Det setter en begrensing i oppgaven ettersom det kunne vært datasett med god verdi. Vi har likevel klart å hente datasett og widgets fra andre offentlige sider. Vi har blant annet satt inn url-nøkler fra meteorologisk værinstitutt Yr, Statens vegvesens vegkart og MazeMap, som tilbyr planløsning på en rekke offentlige bygg.

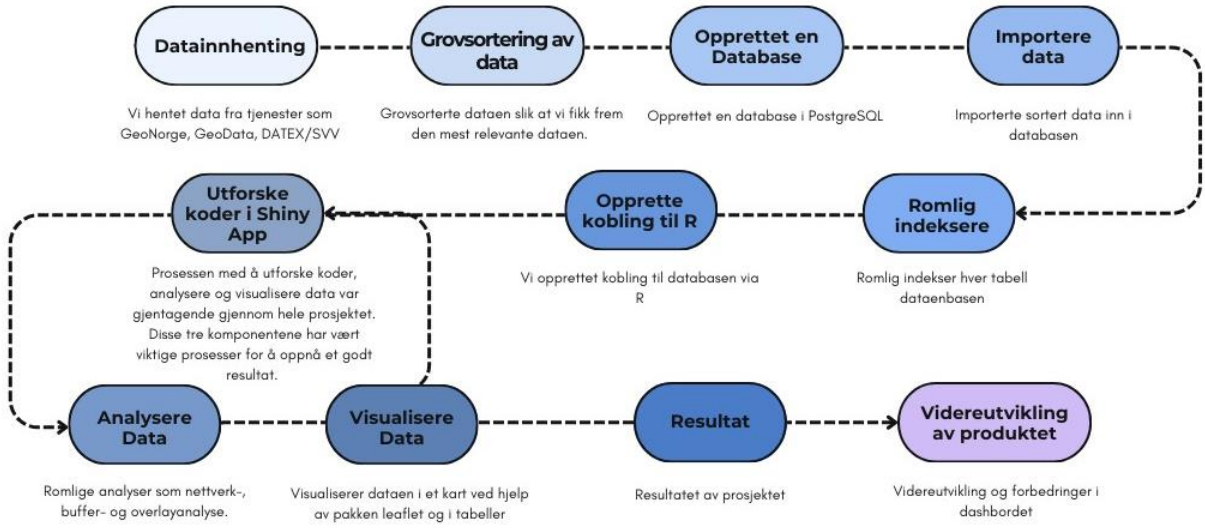
På bakgrunn av avgrensingene vil oppgaven forme seg rundt mulighetene som ligger i grunn, ikke begrensningene.

2. Metode

Kapittelet presenterer vår tilnærming til hvordan vi gjennomfører forskningsprosjektet vårt. Deler av oppgaven har gått ut på å samle inn og anvende data. I kapittelet vil vi beskrive og begrunne våre valg, og gi en oversikt over prosesser vi har benyttet oss av. Figur 11 er en forenklet fremstilling av utviklingsprosessen vår.

Utviklingsprosessen startet med datainnhenting, der data ble samlet fra ulike tjenester. Deretter ble dataen grovsortert for å filtrere ut den mest relevante informasjonen, og en database ble opprettet i PostgreSQL. Etter import av den sorterte dataen ble det opprettet en kobling til R for å kunne utføre analyser og kjøre databasespørringer. For å forbedre ytelsen ved geografiske spørringer ble hver tabell i databasen romlig indeksert.

Videre utforsket vi koder i Shiny App gjennom en iterativ prosess med å skrive kode, analysere og visualisere data. Denne prosessen inkluderte romlige analyser som nettverks-, buffer- og overlay-analyser. Visualisering av data ble med bruk av *Leaflet*-pakken i kart og i tabeller. Resultatet av prosjektet ble presentert i et dashbord, hvor dataen var tilgjengelige gjennom forhåndsdefinerte spørringer, inkludert adressesøk-funksjonen som viste det geometriske punktet for en adresse i et kart.



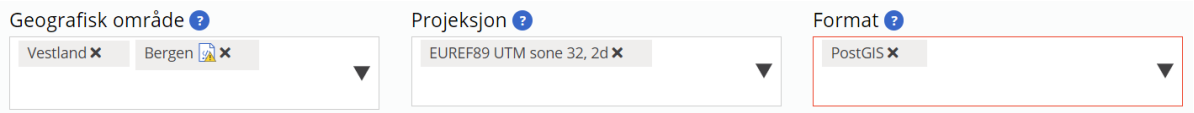
Figur 11: Utviklingsprosess fra start til resultat

2.1 Datainnsamling, vurdering av kvalitet og integrering

Vi startet prosjektet med å finne relevant data til dashbordet vi har utviklet. Ved innsamling brukte vi tjenester som Geonorge, Statens vegvesen, meteorologisk institutt, MazeMap og Geodata. På Geodatas avanserte uttrekk kunne vi filtrere ut ønsket data innen eksempelvis eiendom, befolkning, kulturminner, samferdsel og samfunnssikkerhet.

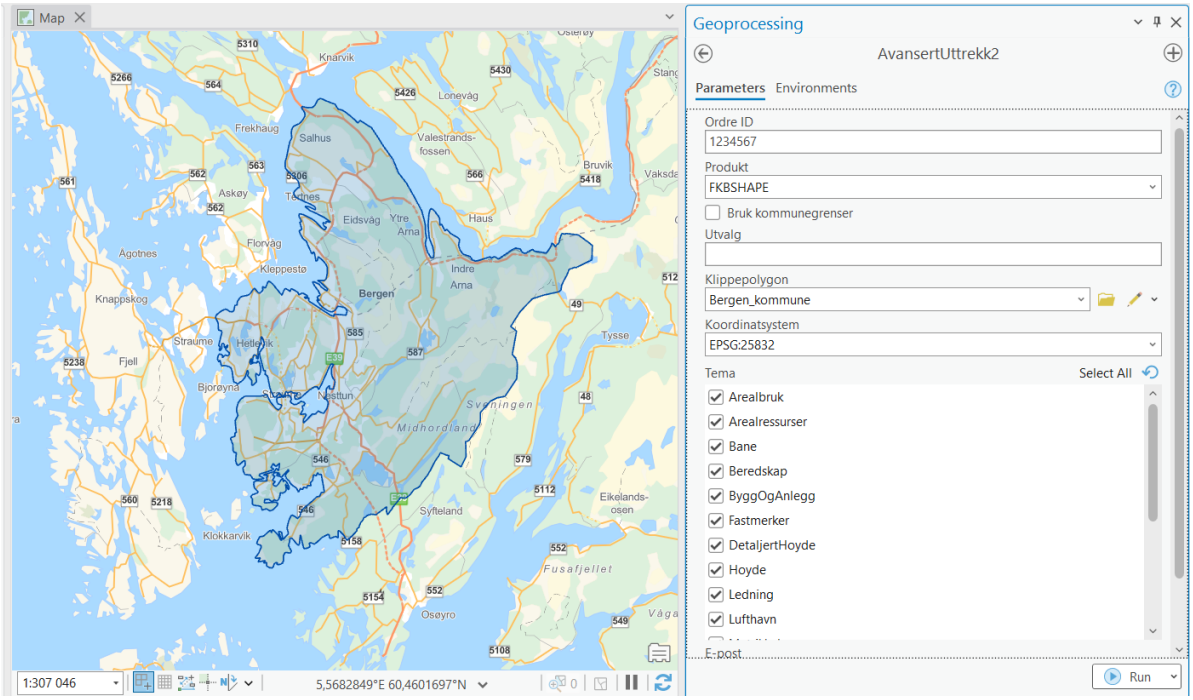
Data betraktes som en ferskvare i mange teknologiske og vitenskapelige sammenhenger, noe som understreker viktigheten av oppdatering og vedlikehold. Dette sikrer at informasjonen som brukes er nøyaktig og relevant. I vårt forskningsprosjekt er datainnsamling fra diverse kilder spesielt utfordrende. Vi integrerte data fra et bredt spekter av tjenester og institusjoner, som værdata fra meteorologiske institutter, trafikkdata fra Statens vegvesen, og beredskapsdata som flomsone og skredfare. Datakildene har egne formater, noe som krever kontinuerlig validering for å sikre at informasjonen som formidles gjennom vårt system, er oppdatert («Datakvalitet», 2019).

Gjennom Geonorges kartkatalog kan man laste ned offentlig kartdata. Her brukte vi parameterinnstillingene: Vestlandet, Bergen for geografisk område, EUREF89 UTM sone 32, 2d som projeksjon og PostGIS eller shapefil som format. På bakgrunn av teori presentert i kapittel 1.4 s.27, var det fornuftig for å minske datakapasitet, for bedre ytelse og resultat. Figur 12 er et eksempel på hvordan parameterinnstillingene ved nedlastning av datasett ville sett ut.



Figur 12: Bildet illustrere parameterinnstillinger for nedlastning av data fra Geonorge.

Geodata er leverandør av Esri og ArcGIS i Norge og tilbyr en rekke geografisk data (*Geodata*, u.å.). Figur 13 illustrerer parameterinnstillingene for uttrekket. Vi har avgrenset *polygonen* til kommunegrensen for Bergen. *Produkt* refererer til ønsket filformat for uttrekket. Vi valgte shapefil-formatet, da dette er et allsidig format som støttes av flere programvarer (Dempsey, 2023). *Koordinatsystemet* er satt til EPSG:25832 (*EUREF89 UTM sone 32)*, da dette er koordinatsystemet som anbefales for Bergen (*Kartprojeksjoner og koordinatsystemer*, u.å.). Under *tema* kan man velge ønsket datasett. I vårt tilfelle valgte vi å hente ut all tilgjengelig data, slik at vi senere kunne filtrere bort irrelevante data.



Figur 13: Geodata sin avanserte uttrekk i ArcGIS.

Samlet sett ga nedlastningen fra Geonorge og Geodata oss et datagrunnlag for videre bruk til utarbeidelsen av dashbordet. Etter innsamling av data var det viktig å undersøke kvalitet.

2.1.1 Datakvalitet

Ved innsamling av data er det viktig å vurdere kvaliteten og pålideligheten til data som samles inn. Dette kan omfatte vurdering av datakilder, nøyaktighet og eventuelle begrensninger knyttet til datasettene. For geografiske data er det viktig å se på hvor dataen kommer fra, for eksempel fra kilder som Geonorge og Geodata. Det er også viktig å se på hvordan de samler inn dataen. Fra Geonorge er det mulig å undersøke hvem som leverer datasettet man laster ned. Det kan for eksempel være Kartverket eller Miljødirektoratet (Kartkatalogen, u.å.).

For å sikre høy datakvalitet og pålitelighet, kan det være nødvendig med prosesser som datavask. Dette kan innebære å bruke verktøy for å identifisere og rette opp feil og mangler i datasettet («Datakvalitet», 2019). Ved å ha en grundig tilnærming til vurdering og sikring av datakvalitet og pålitelighet, kan man øke tilliten til resultatene og bidra til mer pålitelige analyser og konklusjoner. Som nevnt er data ferskvare, da det kontinuerlig oppdateres. For å lage værvarsel-modellen har vi brukt sanntidsdata gjennom en widget. Mer om widgets blir undersøkt i kapittel 2.3.5 s.45.

Kompleksiteten av data er forsterket av behovet for å kunne behandle og analysere dataen. Det krever kvalitetskontroll for å håndtere variasjoner i dataens nøyaktighet og kontekst. Dette arbeidet er avgjørende for at systemet skal kunne levere presis informasjon til nødetatene. Datahåndteringsstrategier sikrer at vår løsning kan fungere som en pålitelig kilde til oppdatert informasjon.

2.1.2 Dataintegrasjon

Etter vi har samlet inn, undersøkt og sikret høy kvalitet, var neste steg i utviklingsarbeidet å gjøre data anvendelig. Når data er samlet inn fra ulike kilder, kan det være nødvendig å integrere og transformere datasettene for å gjøre dem anvendelige for analyse og visualisering. Dette kan innebære prosesser som georeferering, konvertering av filformater og fjerning av feilaktig data. I dette prosjektet har shapefilene blitt bearbeidet i QGIS. Dette undersøkes nærmere i kapittel 2.2.1 s.35.

For å sikre at den innsamlede dataen er anvendelige for videre analyse og visualisering, er dataintegrasjon viktig. Data fra forskjellige kilder har ofte varierende formater. Uten en integreringsprosess vil det være vanskelig å koble eller utføre analyser av datasettene. Ved å grovsortere dataen, sikrer vi at informasjonen kan brukes i sammenhengende modeller og at det gir pålitelige resultater.

Vi valgte QGIS som verktøy for dataintegrasjon ettersom det er et verktøy som håndterer romlig data. QGIS støtter et bredt spekter av filformater og tilbyr verktøy for georeferering og datakonvertering (*Hva er åpen kildekode?*, u.å.). Som nevnt i kapittel 1.3.5 s.20, har QGIS blitt brukt til å importere og eksportere romlig data inn i PostgreSQL databasen. Vi har brukt Database og DB Manager for å få det i databasen i PostgreSQL.

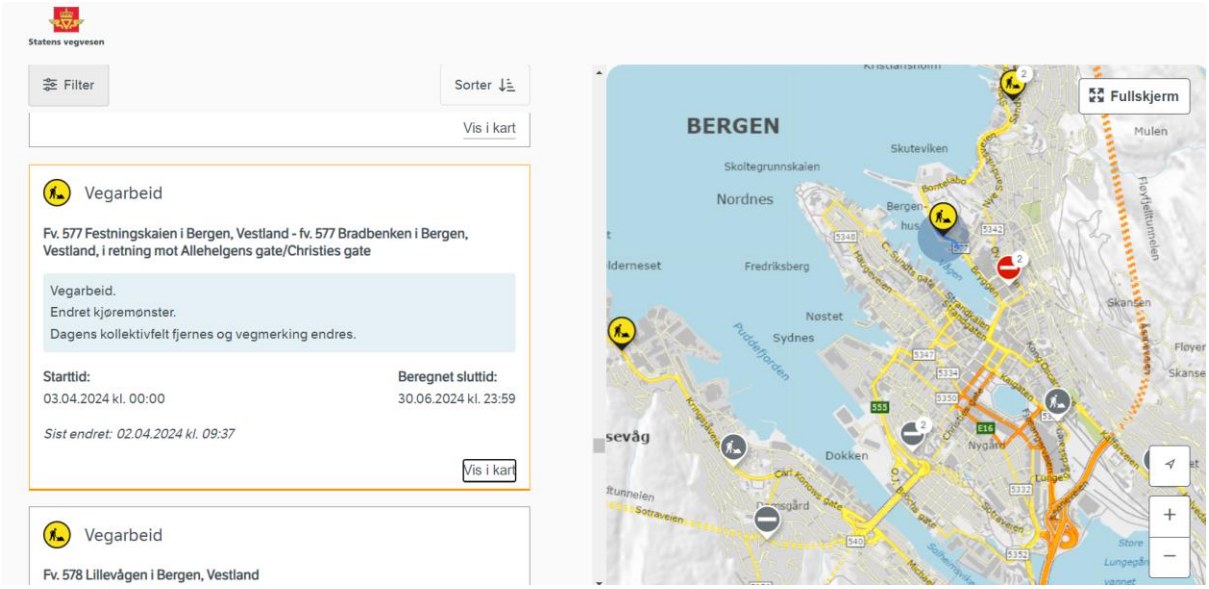
2.1.3 Formater og deres anvendelse i prosjektet

Gjennom datainnsamlingen har vi anvendt ulike formater. Utviklingen av et dashbord krever data fra ulike kilder, som ofte kommer i varierende formater. I vårt prosjekt har vi arbeidet med flere filformater, inkludert DATEX, Shapefile og SQLite. For å håndtere datainnsamling- og behandling i prosjektet, ble det nødvendig å forstå og arbeide med ulike filformater. Denne delen av metodeavsnittet fokuserer på vår tilnærming til å håndtere formatene og deres anvendelse i prosjektet.

Formatene har vært avgjørende for vår evne til å samle, analysere og presentere data på en effektiv måte. Valget av format ble bestemt basert på de programvarene vi brukte og tjenestene som leverte dataen. For å gi en bedre forståelse av filformatene, vil vi undersøke hvert format individuelt. Vi vil se på funksjonalitet og hvordan de har blitt brukt i vårt prosjekt.

De nevnte formatene spilte en viktig rolle i vår tilnærming til datainnsamling, analyse og presentasjon. DATEX-formatet ble brukt for å samle inn trafikkdata fra eksterne kilder, Shapefil- formatet ble brukt for å representere geografiske data, og SQLite-formatet ble brukt som en lokal database for veidata til nettverksanalysen.

I vår metode for å inkludere trafikkmeldinger direkte i dashbordet, valgte vi å benytte DATEX-formatet. Dette formatet gir oss tilgang til sanntidsinformasjon om veitrafikk, noe som er avgjørende ved utrykninger og andre nødsituasjoner. Vår målsetning med DATEX var å oppnå et resultat lignende Figur 14.



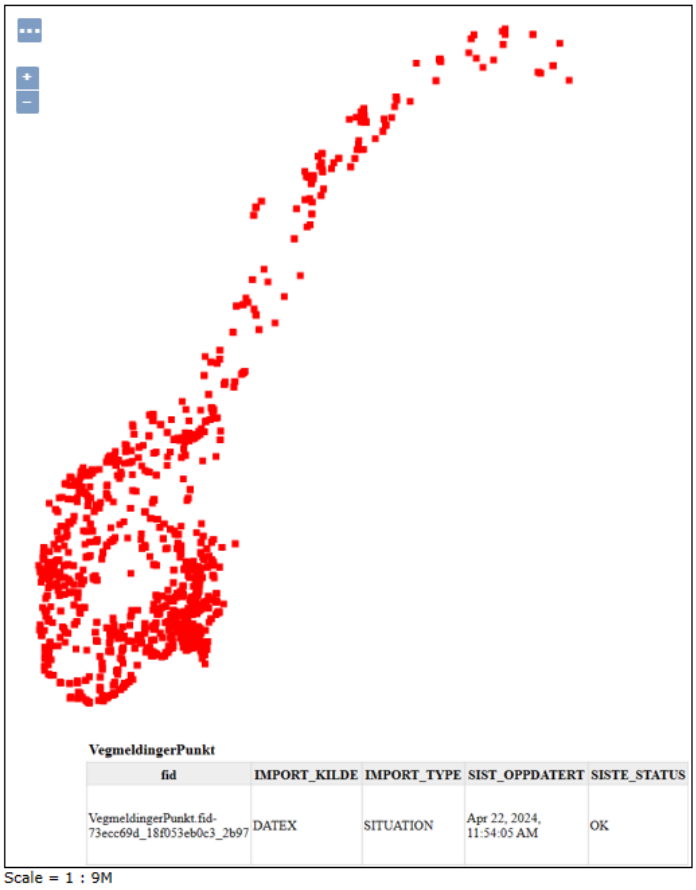
*Figur 14: Utklipp fra Statens Vegvesen (*Trafikkmeldinger | Vegvesen trafikk*, u.å.)*

Etter Statens vegvesen sin anbefaling valgte vi å bruke versjon 2.3. Dette var på grunnlag av at den nyeste versjonen 3.1 ikke var ferdig utviklet og overtar ikke før i juni 2024. Selv om vi i utgangspunktet ønsket å integrere koden direkte i vårt eksisterende script, endte vi opp med å legge inn API-nøkkelen som et *iframe* objekt i dashbordet. *Iframe* objektet bruker vi å lese inn eksternt innhold fra nettsider. For å implementere DATEX-formatet inn i dashbordet brukte vi Kode 1, her satt vi inn OGC tjenesten i serverdelen av scriptet. *Tags* er et objekt som inneholder funksjoner for å generere HTML-elementer i Shiny. *$* brukes i Shiny til å referere til bestemte UI- eller server server-objekter, slik som *output* og *input* (Grolemund et al., 2017) (*Shiny - Server Function*, u.å.).

```
server <- function(input, output, session) {

  api_url <- "https://ogckart-
sn1.atlas.vegvesen.no/datex_2_0/wms?service=WMS&version=1.1.0&request=GetMa
p&layers=datex_2_0%3AVegmeldingerPunkt&bbox=-
100000.0%2C6430000.0%2C1120000.0%2C7960000.0&width=612&height=768&srs=EPSG%
3A25833&styles=&format=text%2Fhtml%3B%20subtype%3Dopenlayers#toggle"

  res <- GET(api_url)
  if (status_code(res) == 200) {
    data <- content(res, "text")
    output$info_frame <- renderUI({
      tags$iframe(src = api_url, width = "100%", height = 600)
    })
  } else {
    output$info_frame <- renderText({
      "Kunne ikke hente informasjon"
    })
  }
}
```

Kode 1: Bruk av DATEX i dashbordet.

Som et standardformat utviklet for utveksling av trafikkinformasjon på tvers av europeiske trafikksystemer, har DATEX muliggjort mottak av sanntidsdata fra Statens vegvesen. Dette inkluderer viktig informasjon om trafikkmeldinger, veiarbeid og hendelser som påvirker trafikkflyten. Ved å integrere dataen i vårt GIS-baserte dashbord, har vi kunnet presentere presis og oppdatert trafikkinformasjon i sanntid. Dette har vært viktig for å kunne tilby et klart og aktuelt bilde av trafikksituasjonen, noe som muliggjør raskere og bedre informerte beslutninger i nødssituasjoner. Figur 15 er versjon 2.3 fra DATEX og punktene representerer tilhørende trafikkmeldinger.



Figur 15: Utklipp av versjon 2.3 DATEX.

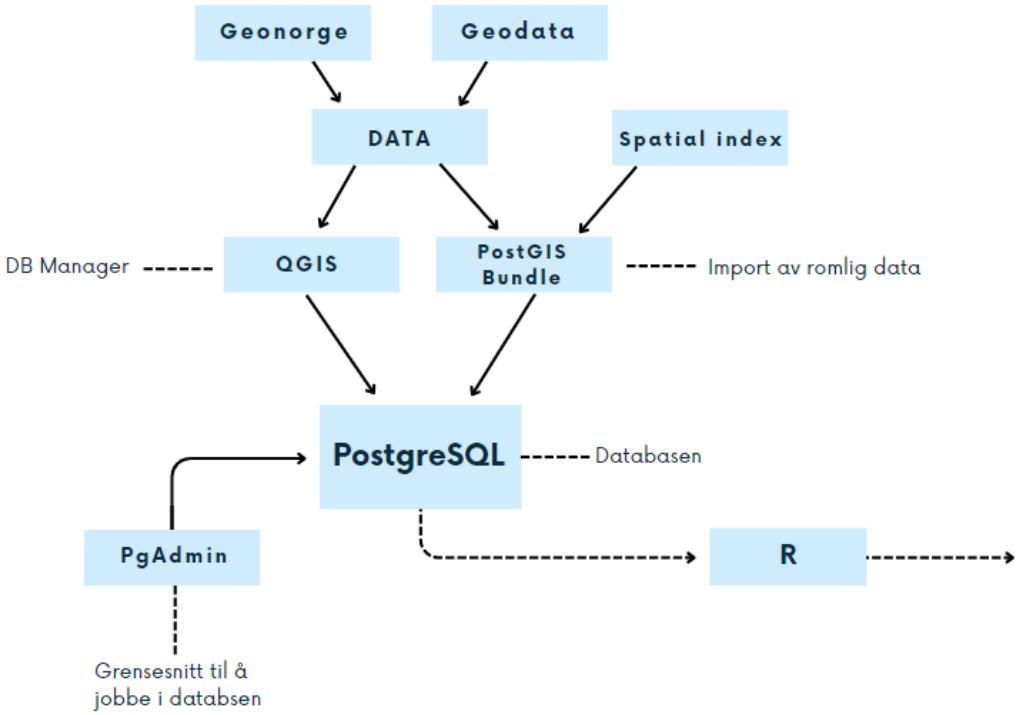
Shapefilene ble lastet ned fra Geodatas avanserte uttrekk gjennom ArcGIS Pro. Deretter ble filene bearbeidet i QGIS, før de ble integrert inn i databasen. Shapefilene danner grunnlaget for datasettene som vises på det interaktive kartet i dashbordet. Med bruk av data har vi utført ulike analyser, inkludert nettverks-, buffer- og overlayanalyse. For å hente ut data fra PostgreSQL-databasen, har vi benyttet SQL-spørringer i R.

SQLite-filen lastet vi ned fra Geonorge. Vi har brukt data til å gjennomføre nettverksanalyse. Veidataen inneholder informasjon om veistrukturer, trafikkflyt og andre relevante detaljer som er avgjørende for å utføre nøyaktige analyser. Ved å integrere denne veidataen i SQLite-filen, har vi kunnet utføre avanserte nettverksanalyser som inkluderer beregning av reisetider og ruteoptimalisering.

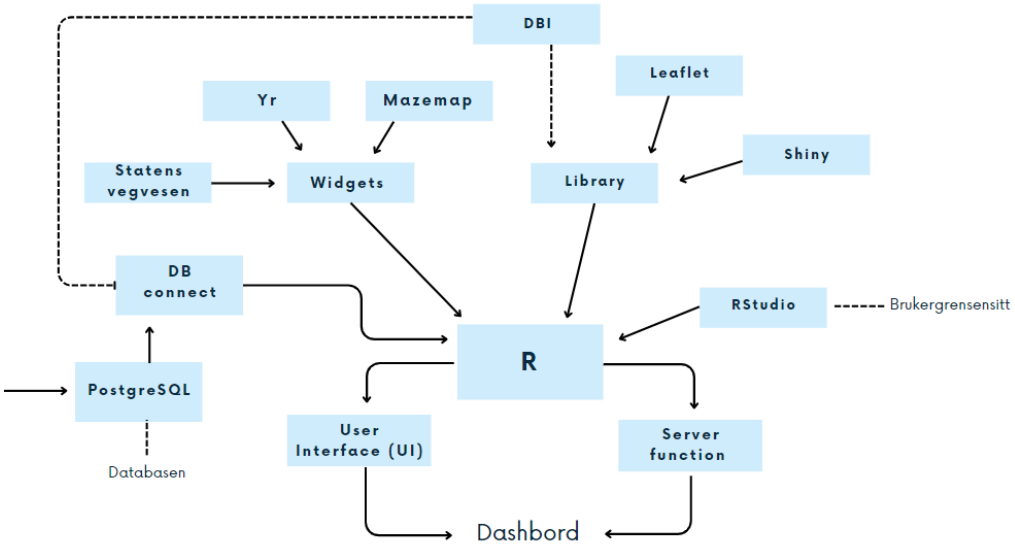
2.2 Datahåndtering

Etter datainnsamlingsprosessen var gjennomført, startet arbeidet i de ulike programvarene. Gjennom forskningsprosjektet ble det brukt programvarer som QGIS, pgAdmin og RStudio. I dette kapittelet undersøkes det hvordan data blir håndtert, og veien fra data til applikasjon.

Figur 16 illustrerer veien fra innsamlet data til en database. Her startet datainnsamlingen i Geonorge og Geodata, som ga oss et datagrunnlag, som deretter ble tilpasset i QGIS og PostGIS Bundle, med romlig indeksering. Vi har jobbet i pgAdmin som et brukergrensesnitt mot databasen i PostgreSQL. Mens Figur 17 illustrerer flyten fra data til et ferdig dashbord. Det gir en oversikt over widgets, valg av *Library* og programmeringsspråk. Pakken *DBI* åpner muligheten for å kjøre SQL spørringer i R. Sammen utgjør figur 16 og 17 en helhetlig prosess fra data til ferdig dashbord. Nærmere om hvordan vi har anvendt de ulike programvarene presenteres i kapittel 2.2.1 s.35.



Figur 16:Bildet er en forenklet Illustrasjon av informasjonsflyten til databasen.

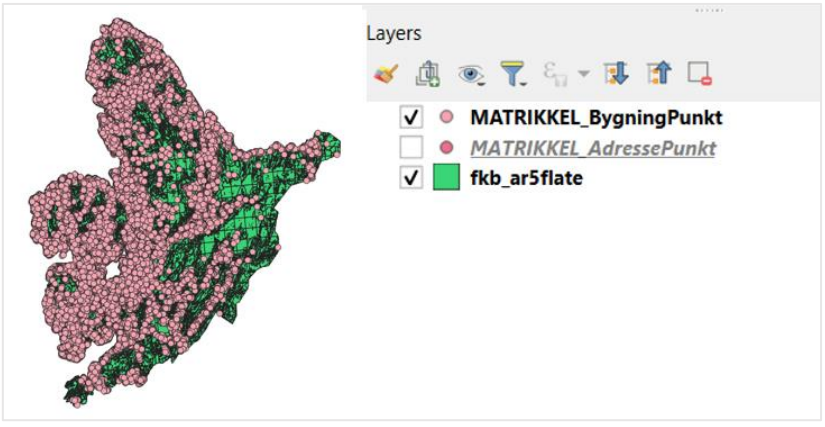


Figur 17: Bildet illustrerer en forenklet modell av hvordan programmeringsspråket R og de ulike leddene er brukt for å få ut et ferdig dashbord.

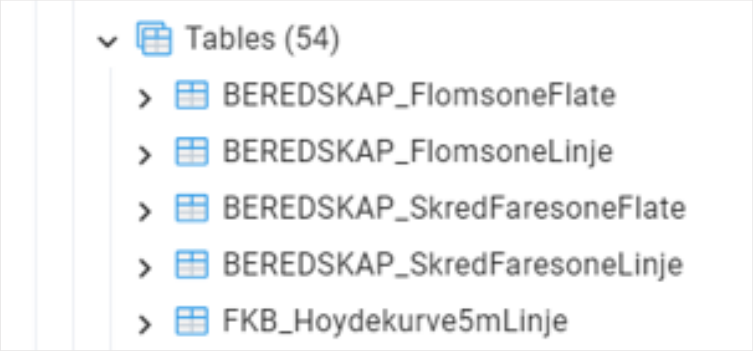
2.2.1 Data til applikasjon

Utviklingsarbeidet fra data til applikasjon startet med arbeid i QGIS, deretter PostgreSQL og videre R i RStudio.

I vårt utviklingsarbeid har vi brukt QGIS til å utforske og grovsortere data. Vi har sett gjennom ulike datasett og filtrert det som er relevant. I QGIS har man muligheten til å rense data for eventuelle feil og mangler. Dette ble nevnt kort innledningsvis under kapittel 2.1.2 s.31. I QGIS lastet vi opp datasett for å sjekke at det var riktig geografisk data for Bergen kommune. Denne grovsorteringen forenklet arbeidet for å kvalitetssikre data. Figur 18 viser hvordan data kan se ut i QGIS. *Layers* viser hvilke datalag som er i bruk. I QGIS koblet vi opp til databasen vi opprettet i PostgreSQL ved hjelp av verktøy *Database* og *DB Manager*. Denne sammenkoblingen gjorde at vi fikk data inn i PostgreSQL databasen, under Tables i pgAdmin. Figur 19 viser et utklipp i pgAdmin over hvordan *Tables* i databasen ser ut. Vi har totalt 54 tabeller.

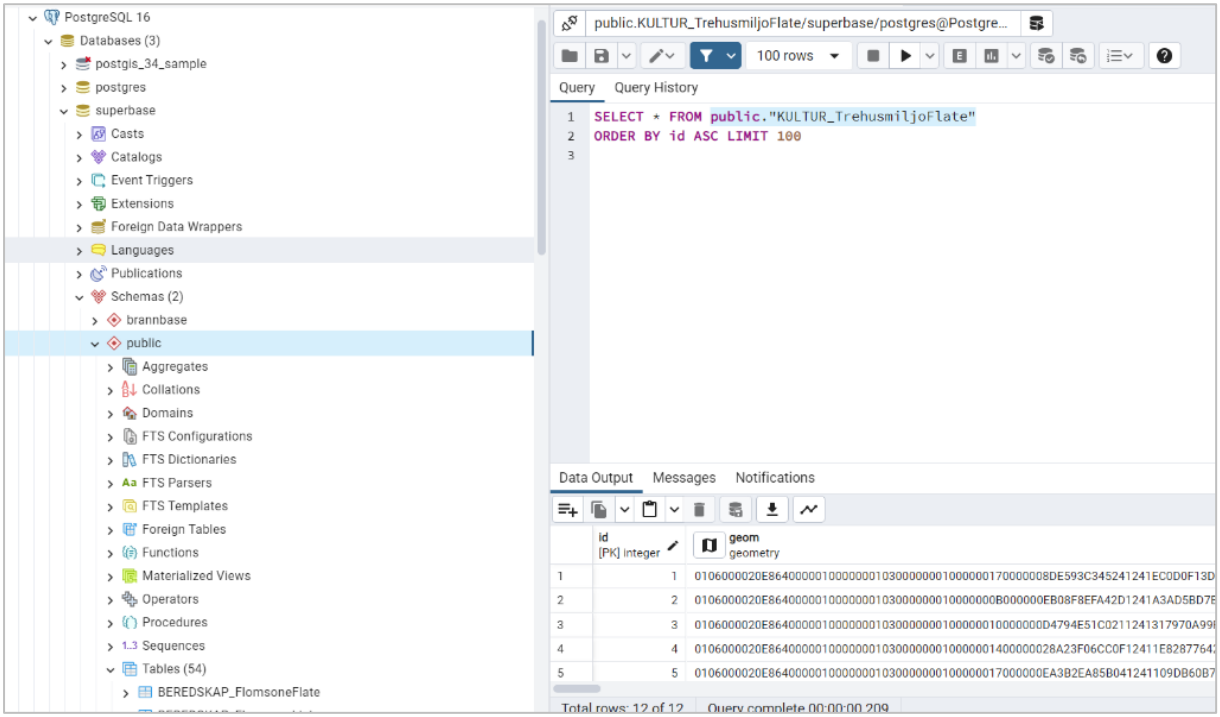


Figur 18: Bildet illustrerer hvordan data kan se ut i QGIS. Her ser man arealflate og bygningspunkt.



Figur 19: Bildet er et utklipp fra pgAdmin i Tables.

Videre i arbeidet lagde vi en database i PostgreSQL med et brukergrensesnitt i pgAdmin. Der plasseres all data vi henter fra Geonorge og Geodata. For at databasen skal kunne lagre og behandle romlig data, må vi utvide databasen med PostGIS. PostGIS utvider funksjonaliteten til databasen ved å legge til støtte for lagring, indeksering og spørring av romlig data (*Creating a Spatial Database*, u.å.). Det muliggjør lagring av ulike typer romlige data som punkter, linjer, polygoner, og multi-geometrier i 2D og 3D (PostGIS, u.å.). Romlig data importeres via PostGIS Bundle eller QGIS, inn i databasen vår i PostgreSQL. I vårt forskningsprosjekt har vi kalt databasen *superbase*. Figur 20 viser et utklipp av databasen som er opprettet i PostgreSQL, men visualisert i pgAdmin.



Figur 20: Bildet er et utklipp fra pgAdmin som viser en oversikt over databasen fra PostgreSQL.

Etter at data var importert, kunne vi fortsette arbeidet i R med RStudio. I R kan vi hente ut data fra databasen og utføre spørringer. Vi kan også knytte tabeller sammen og slette overflødig data vi ikke har behov for.

Selve dashbordet oppretter vi med pakken *Shiny*. Pakken brukes for å lage interaktive webapplikasjoner. Det muliggjør å analysere data på en brukervennlig måte (*Shiny*, u.å.). Vi bruker *leaflet* til å få fram et interaktivt kart, hvor vi kan visualisere data fra databasen (*Leaflet*, 2023).

Vi ønsker også å få koblet opp API-nøkler fra ulike tjenester som trafikkmeldinger til Statens vegvesen. Dette kan også bygges videre med marinetrafikk og eCall, som er SOS-knappen i biler. Ved hjelp av denne framgangsmåten kan man få «alt i en» hvor man kan hente inn og kombinere ulike tjenester i ett og samme kartlag.

2.2.2 Kobling mellom PostgreSQL, QGIS og R

For å effektivt håndtere, analysere og visualisere romlig data, har selve koblingen mellom programvarene vært sentral. Det etableres en kobling mellom PostgreSQL, QGIS og R. Vi startet med å opprette en romlig database i PostgreSQL, hvor vi bruker pgAdmin til å kommunisere med databasen. Databasen fungerer som ryggraden i vårt dataøkosystem. Ved hjelp av PostGIS, som er en romlig utvidelse for PostgreSQL, kan vi lagre komplekse datasett og utføre romlige analyser.

Med databasen som det sentrale knutepunktet, starter videre arbeid i QGIS, som tillater direkte kobling til databasen. Denne tilkoblingen muliggjør en interaktiv utforskning og vi kan få visualisert data. Vi bruker også DB-Manager til å koble oss til databasen hvor vi kan både importere og eksportere data.

Vi kobler opp postgreSQL-databasen vår til R, gjennom *Rpostgres*- pakken og *DBI* -pakken, slik at vi får en databaseforbindelse og kan hente og manipulere data (Wickham et al., 2023). R har pakker som retter seg mot romlige analyser, som *simple features*. Denne pakken gir muligheten til å utføre statisk bearbeiding og modellering av data. Den integrerte tilnærmingen forbedrer vår evne til å tolke romlige data (Pebesma, u.å.). Kode 2 er et utklipp av hvordan vi oppretter en tilkobling til databasen vår i R.

```
con <- dbConnect(RPostgres::Postgres(),
                 dbname = "superbase",
                 host = "localhost",
                 port = 5432,
                 user = "postgres",
                 password = "postgres")
```

Kode 2: Koden viser hvordan man oppretter en kobling til databasen.

2.2.3 Kunstig intelligens

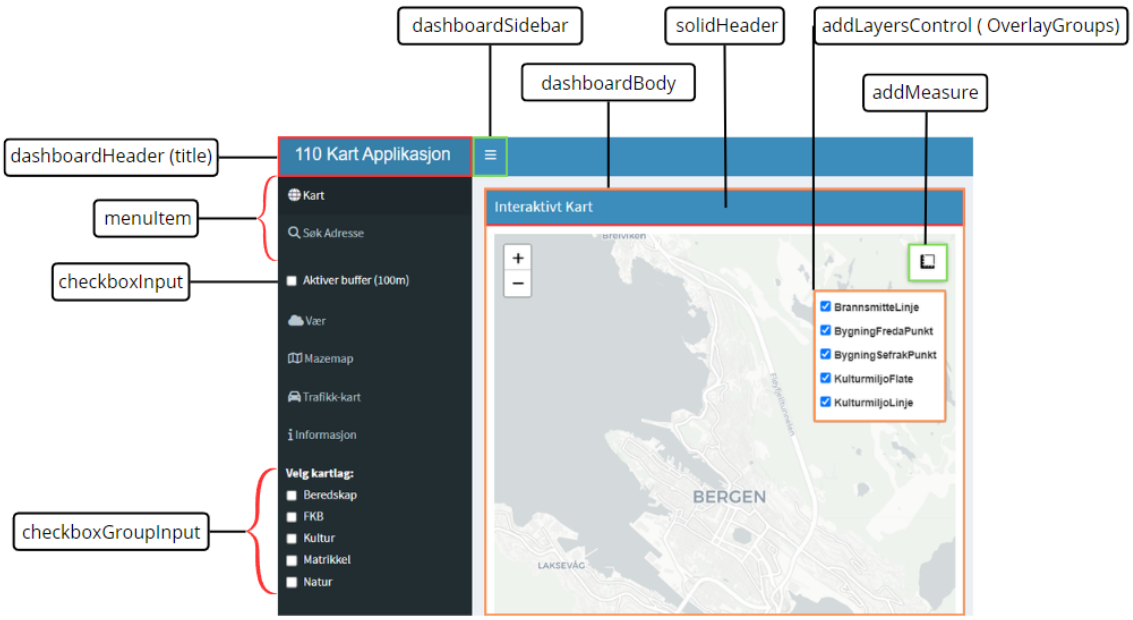
Nevnt innledningsvis i kapittel 1.3.8 s.23, åpner kunstig intelligens for en mer dynamisk og effektiv feilsøkingsprosess. Copilot gir oss mulighet til å få programmeringsstøtte direkte i RStudio. Vi bruker også OpenAI sin ChatGPT4 til både koding og andre typer spørringer.

Gjennom ChatGPT med OpenAI har vi utviklet en avansert versjon. Denne versjonen er en chatbot vi selv har programmert i OpenAI til å tolke vår oppgave. Under oppsettet av denne KI-modellen ga vi inndata i form av script vi har laget, datagrunnlag og selve oppgaveteksten. Vi lagde den til en teknisk chatbot, som kan løse problemer innen GIS og generelt om tekniske spørsmål.

Chatbotter er likevel ikke en selvstendig kunnskapsbase, men brukes som et verktøy for å bygge på informasjon fra eksterne kilder. Gjennom arbeidet har kunstig intelligens blitt primært brukt til feilsøking og optimalisering av koder. Verktøy som Copilot og ChatGPT tilbyr støtte gjennom kodeutviklingsprosessen. Det vektlegges likevel at gjennom hele arbeidet er KI blitt brukt med en kritisk tilnærming, hvor vi har sjekket svar opp mot eksterne kilder og gitt egne begrunnelser for valg av kode.

2.3 Dashbordets anatomi

Dashbordet er laget i Shiny app, som er en R-pakke, og er bygd opp av tre ulike deler. *User Interface* (UI), *Server function* (server) og *ShinyApp* som kobler sammen *UI* og *server.* Figur 21 illustrerer dashbordets anatomi og forklarer de spesifikke elementene den består av. Navnene på komponentene i figuren, slik som *dashboardHeader*, *menuItem*, og *checkboxInput*, er direkte hentet fra koden i skriptet som definerer hver funksjon og layout i dashbordet. Det viser en sammenheng mellom koden og de visuelle komponentene.



Figur 21: Dashbordet med forklaring referert til koden om dashbordets anatomi

UI bestemmer hvordan appen ser ut. Her lages layouten og elementene som visualiseres i dashbordet, samt hvor vi legger inn valg av knapper og menyer. Her ønsker vi å vektlegge brukervennlighet med gode designprinsipp (*Shiny - Getting Started*, u.å.). Kode 3 er en forenklet versjon av UI-delen vår. Her kan man se hvordan dashbordet vårt er bygget opp av sidemeny, med ulike *menuItems* og *checkboxInputs*. *dashboardBody* bestemmer vi hvordan hvert panelvindu skal se ut når vi har trykket på et *menuItem*.

```
ui <- dashboardPage(
  dashboardHeader(title = "110 Kart Applikasjon Dashboard"),
  dashboardSidebar(
    sidebarMenu(
      menuItem("Kart", tabName = "kart", icon = icon("globe")),
      menuItem("Søk Adresse", tabName = "sokAdresse", icon =
icon("search")),
      checkboxInput("bufferToggle", label = "Aktiver buffer (100m)", value
= FALSE),
      menuItem("Vær", tabName = "vaer", icon = icon("cloud")),
    )
  ),
  dashboardBody(
    tabItems(
      tabItem(tabName = "kart",
              fluidRow(
                box(width = 12, solidHeader = TRUE, status = "primary",
                    title = "Interaktivt Kart",
                    leafletOutput("leafletMap", height = "600px")
                )
              )),
      tabItem(tabName = "sokAdresse",
              fluidRow(
                DTOutput("adresseTable")
              ),
              leafletOutput("kart", height = "95vh"),
              absolutePanel(top = 50, right = 20,
                            textInput("adresse", label = NULL, placeholder
= "Skriv inn adresse, nummer, og bokstav"),
                            actionButton("sok", "Søk"),
                            id = "searchPanel"),
              tags$script(HTML("
                $(document).on('shiny:inputchanged', function(event) {
                  if (event.name === 'adresse') {
                    $('#adresse').on('keyup', function (e) {
                      if (e.keyCode === 13) {
                        $('#sok').click();
                      }
                    });
                  }
                });
              "))
      ),
      tabItem(tabName = "vaer",
              fluidRow(
                tabBox(
                  id = "vaerTabs",
                  height = "100vh",
                  tabPanel("Værvarsel",
                           box(title = "Værvarsel", status = "primary",
solidHeader = TRUE,
```

```
tags$iframe(src="https://www.yr.no/en/content/1-
92416/meteogram.svg?mode=dark", height = "95%", width = "100%")
                                  collapsible = TRUE, collapsed = FALSE)
                  )
      )
    )
  )
)
```

Kode 3: UI-delen av dashbordet.

Kode 3 viser hvordan UI-delen er bygget opp av en sidemeny med ulike *menu-items*. Koden er enkelt bygget opp, noe som gjør at man lett kan legge til flere *menu-items*. Verdiene som slutter med *output* refererer videre til analysene vi gjør i serverfunksjonsdelen av appen (*Shiny - User Interface (UI)*, u.å.).

I serverdelen, illustrert i Kode 4, ligger logikken som appen bruker. Det er her funksjonene til de ulike *items* ligger. Spørringer gjør vi til databasen for å hente opp widgets eller lenker, slik at vi kobler opp til *leaflet*. Det er her vi håndterer de reaktive forbindelsene mellom inndata og utdata. Vi definerer hva som skal utregnes og hvordan det skal presenteres i appen.

I Kode 4 kan vi se hvordan vi henter opp en API og legger inn i et *iframe*.

```
server <- function(input, output, session) {

  api_url <- "https://ogckart-
sn1.atlas.vegvesen.no/datex_2_0/wms?service=WMS&version=1.1.0&request=GetMa
p&layers=datex_2_0%3AVegmeldingerPunkt&bbox=-
100000.0%2C6430000.0%2C1120000.0%2C7960000.0&width=612&height=768&srs=EPSG%
3A25833&styles=&format=text%2Fhtml%3B%20subtype%3Dopenlayers#toggle"

  res <- GET(api_url)
  if (status_code(res) == 200) {
    output$info_frame <- renderUI({
      tags$iframe(src = api_url, width = "100%", height = 600)
    })
  } else {
    output$info_frame <- renderText({
      "Kunne ikke hente informasjon"
    })
  }
}
```

Kode 4: Utklipp av serverdelen til dashbordet.

Shiny app er stedet vi kobler User Interface og server funksjonene sammen, slik som Kode 5. Vi får opp den fullstendige applikasjonen, altså dashbordet. Appen kan kjøres videre direkte i RStudios innebygde vindu eller i en ekstern nettleser.

```
   # Kjører Shiny-appen
   shinyApp(ui, server)
```

Kode 5: Kobler UI- og server-delen sammen.

Hovedkomponentene for å lage en app i Shiny er UI, server function og Shiny app som kobler appen sammen (*Shiny - Share Your Apps*, u.å.). Appen kan kjøres lokalt eller via tjenester som *Posit Connect* (*Shiny - Server Function*, u.å.). Vi har primært fokusert på romlige analyser og brannvesenets behov under utviklingen av dashbordet. For fullstendig script se vedlegg, appendiks.

Kode 6 er en forenklet visning av hvordan vi har lagt inn kartlag i kartet. Vi oppretter en kobling til databasen. Kartlagene vi legger inn knyttes opp til «action buttons» som vi har laget i UI-delen av scriptet. Dette kan gjøres med alle romlige tabeller i databasen, men vi har valgt å avgrense til et par tabeller. Vi opplever at flere kartlag fører til dårligere ytelse.

```
    romlige_data <- reactive({
    req(input$layers)
    data_list <- list()

    if("lag1" %in% input$layers) {
      data_brann <- st_read(con, query = "SELECT * FROM
public.\"KULTUR_BrannsmitteLinje\"")
      data_list[["BrannsmitteLinje"]] <- st_transform(data_brann, crs =
4326)
    }

    if("lag1" %in% input$layers) {
      data_trafikk <- st_read(con, query = "SELECT * FROM
public.\"KULTUR_BygningFredaPunkt\"")
      data_list[["BygningFredaPunkt"]] <- st_transform(data_trafikk, crs =
4326)
    }
    return(data_list)
  })
  output$leafletMap <- renderLeaflet({
    leaflet() %>%
      addProviderTiles("CartoDB.Positron") %>%
      setView(lng = 5.325, lat = 60.3945, zoom = 13) %>%
      addMeasure(primaryLengthUnit = "meters", primaryAreaUnit =
"sqmeters", activeColor = "#0aaff1", completedColor = "#c70a20") %>%
      addLayersControl(
        overlayGroups = c("BrannsmitteLinje", "BygningFredaPunkt"),
        options = layersControlOptions(collapsed = FALSE)
      )
  })
  observe({
    data <- romlige_data()
    leafletProxy("leafletMap") %>%
      clearShapes()

    if (!is.null(data$BrannsmitteLinje)) {
      leafletProxy("leafletMap") %>%
        addPolylines(data = data$BrannsmitteLinje, color = "#ff6200",
weight = 4, opacity = 0.5, group = "BrannsmitteLinje")
    }

    if (!is.null(data$BygningFredaPunkt)) {
      leafletProxy("leafletMap") %>%
        addCircleMarkers(data = data$BygningFredaPunkt, color = "red",
radius = 3, group = "BygningFredaPunkt")
    }
```

```
    # Oppdater addLayersControl med de nye lagene
    leafletProxy("leafletMap") %>%
      addLayersControl(
        overlayGroups = c("BrannsmitteLinje", "BygningFredaPunkt"),
        options = layersControlOptions(collapsed = FALSE)
      )
  })
```

Kode 6: Kartlag i dashbordet

I vårt arbeid med dashbord har vi valgt å iverksette romlige analyser og verktøy som kan være nyttige for brannvesenet. Vi har gjennomført analyser for forebyggende arbeid, men også som kan være nyttige i en nødsituasjon. Analysene vi har valgt i dashbordet er: adressesøk, overlay-, buffer- og nettverksanalyse. Til forebyggende har vi gjort overlay spørringer som for eksempel viser områder i ras- og flomsoner, og hvilke adresser som går under trehusbebyggelse. Målet med analysene er å gjøre innsamlet data om til nyttig informasjon (*Introduction to Spatial Analysis*, u.å.). I tillegg har vi implementert verktøy som kan være nyttige i brannvesenets ulike operasjoner. Dette inkluderer måleverktøy og widgets.

2.3.1 Adressesøk i kart

I dashbordet har vi laget et adressesøk i kartet vårt. Sluttbruker kan da søke på adresser i databasen og som resultat få representert adressen i kartet. Vi har lagt til rette for å kunne søke på adresser med og uten bokstav. Da tillater scriptet å søke på adresser som «Inndalsveien 28» og «Fabrikkgaten 7C». Vi søker da på adressen i databasen, og henter ut «GEOM» kolonnen til adressen. Adressen vil vises i kartet via *leaflet*. Kode 7 er et utklipp av adressesøk.

```
    # Analyser inndata for å skille ut adressenavn, nummer og bokstav
    parts <- strsplit(input$adresse, " ")[[1]]
    adressenav <- paste(parts[-length(parts)], collapse = " ")
    nummer_bokstav <- parts[length(parts)]

    # Fjerner alt unntatt tall
    nummer <- gsub("[^0-9]", "", nummer_bokstav)

    # Fjerner tall, etterlater bokstaver
    bokstav <- gsub("[0-9]", "", nummer_bokstav)

    # Justere spørringen basert på om bokstav er spesifisert eller ikke
    query <- if(nchar(bokstav) > 0) {
      sprintf("SELECT ST_Y(ST_Transform(geom, 4326)) AS latitude,
ST_X(ST_Transform(geom, 4326)) AS longitude FROM
public.\"MATRIKKEL_AdressePunkt\" WHERE adressenav = '%s' AND nummer = %s
AND bokstav = '%s'",
              adressenav, nummer, bokstav)
    } else {
      sprintf("SELECT ST_Y(ST_Transform(geom, 4326)) AS latitude,
ST_X(ST_Transform(geom, 4326)) AS longitude FROM
public.\"MATRIKKEL_AdressePunkt\" WHERE adressenav = '%s' AND nummer = %s
AND (bokstav = '' OR bokstav IS NULL)",
              adressenav, nummer)
    }
```

```
    # Utfør spørringen
    result <- dbGetQuery(pool, query)
    dbDisconnect(pool)

    # Vis resultat på kart hvis funnet
    if(nrow(result) > 0) {
      lat <- result$latitude[1]
      lon <- result$longitude[1]
```

Kode 7: Koden viser en del av koden for å gjøre adressesøk i dashbordet.

2.3.2 Bufferanalyse

Som nevnt i kapittel 1.3.10 s.25, innebærer bufferanalyse å skape en definert sone rundt et gitt punkt. I vårt prosjekt har vi integrert bufferanalyse i adressesøk, ved å bruke tilleggsspørringer. Når bufferanalysen aktiveres, markeres sonen, og det genereres en tabell over adresser innenfor denne sonen. Denne analysen gir raskt en oversikt over potensielt berørte adresser i en nødsituasjon. Kode 8 illustrerer utviklingen av denne analysen.

For å lage bufferen bruker vi *ST_Buffer.* Vi finner adresser inni bufferen ved å bruke *ST_Within,* dataen henter vi fra «MATRIKKEL_AdressePunkt». Adressene som er innenfor bufferen, blir presenteres i en tabell i dashbordet, ved bruk av *datatable*.

```
    # Sjekker om bufferToggle er aktivert og inkluderer bufferlogikken
    if(input$bufferToggle && nrow(adresseResult) > 0) {
      bufferQuery <- sprintf("
        WITH adressepunkt AS (
            SELECT geom, adressenav, nummer
            FROM public.\"MATRIKKEL_AdressePunkt\"
            WHERE adressenav = '%s' AND nummer = '%s'
        ), buffer AS (
            SELECT ST_Buffer(ST_Transform(adressepunkt.geom,
4326)::geography, 100)::geometry AS geom
            FROM adressepunkt
        )
        SELECT a.adressenav, a.nummer, ST_Y(ST_Transform(a.geom, 4326)) AS
latitude, ST_X(ST_Transform(a.geom, 4326)) AS longitude
        FROM public.\"MATRIKKEL_AdressePunkt\" a, buffer b
        WHERE ST_Within(ST_Transform(a.geom, 4326), b.geom)", adressenav,
nummer)

      bufferResult <- dbGetQuery(dbPool, bufferQuery)
      # Viser adressene innenfor bufferen i en tabell
      output$adresseTable <- renderDT({
        if(nrow(bufferResult) > 0) {
          datatable(bufferResult, options = list(pageLength = 5, autoWidth
= TRUE), rownames = FALSE)
        } else {
          datatable(data.frame(Informasjon = "Ingen adresser funnet
innenfor bufferen"), options = list(pageLength = 5, autoWidth = TRUE),
rownames = FALSE)
        }
      })
    }
  })
```

Kode 8: Bufferanalyse.

Med vår utvikling av bufferanalysen, ønsket vi at det skulle være mulig å fastsette sikkerhetssoner rundt en hendelse. Analysen er særlig nyttig ved situasjoner hvor det er kritisk å evakuere befolkningen fra området, eksempelvis ved lekkasje av farlig gods. Ved å bruke bufferanalysen til å kartlegge plasseringen av nødvendige ressurser, sikrer brannvesenet at de har tilstrekkelig informasjon til rask respons.

Vår bufferanalyse kan også bidra i planleggings- og forebyggingsstadiene. Denne teknikken kan brannvesenet bruke til å vurdere dekningsgraden til brannstasjoner over et gitt geografisk område. Dette kan hjelpe til med strategisk plassering av nye ressurser eller forbedring av eksisterende infrastruktur. Denne tilnærming sikrer at brannvesenet er bedre forberedt på å håndtere nødsituasjoner på en organisert og effektiv måte.

2.3.3 Overlayanalyser

Det er mange spørringer vi kan gjøre i databasen som kan være nyttige i forbyggende situasjoner. Vi kan eksempelvis gjøre overlayanalyser ved å kombinere flomsoner og adresser eller fredete bygninger. I spørringen i Kode 9 har vi kombinert fredede bygninger med flom og skredsoner.

```
# SQL query
query <- "
SELECT 'flood' as type, k.id AS building_id, k.navn AS building_name,
       f.flomsoneid::text AS zone_id, f.flomsonena AS zone_name, k.geom AS
geom
FROM public.\"KULTUR_BygningFredaPunkt\" k
JOIN public.\"BEREDSKAP_FlomsoneFlate\" f ON ST_Intersects(k.geom, f.geom)
UNION ALL
SELECT 'landslide' as type, k.id AS building_id, k.navn AS building_name,
       s.id::text AS zone_id, s.skredstats::text AS zone_status, k.geom AS
geom
FROM public.\"KULTUR_BygningFredaPunkt\" k
JOIN public.\"BEREDSKAP_SkredFaresoneFlate\" s ON ST_Intersects(k.geom,
s.geom);
"

# Fetch the data from PostgreSQL database
data <- dbGetQuery(con, query)
```

Kode 9: Viser til overlayanalyse med SQL spørringer.

Med Kode 10 kan vi finne ut hvilke adresser som er innenfor trehusbebyggelsen i Bergen. Dette kan være en fornuftig analyse ettersom det er områder med høy spredningsfare.

```
SELECT
    a.id AS adresse_id,
    a.adressenav,
    a.nummer,
    a.bokstav,
    a.postnr,
    a.postnavn,
    a.kommunenum,
    a.gnr,
    a.bnr,
    a.fnr,
```

```
    a.snr,
    t.id AS trehusmiljo_id,
    t.navn AS trehusmiljo_navn,
    t.kommunenav AS trehusmiljo_kommune,
    t.fylkesnavn AS trehusmiljo_fylke
FROM
    public."MATRIKKEL_AdressePunkt" a
JOIN
    public."KULTUR_TrehusmiljoFlate" t ON ST_Within(a.geom, t.geom)
    ORDER BY
    a.adressenav ASC;
```

Kode 10: SQL spørring: Join for adresser innen trehusmiljø.

2.3.4 Måleverktøy

For å implementere et måleverktøy i kartet, har vi også lagt til *addMeasure* funksjonen fra *leaflet.extra* pakken som er markert i Kode 11. Dette gjør at vi kan måle avstand mellom punkt og få opp arealet i en polygon. Verktøyet kan være nyttig for å raskt måle i kartet slik at man kan få en bedre stedsforståelse (Karambelkar & Schloerke, 2022).

```
  output$leafletMap <- renderLeaflet({
    leaflet() %>%
      addProviderTiles("CartoDB.Positron") %>%
      setView(lng = 5.325, lat = 60.3945, zoom = 13) %>%
      addMeasure(primaryLengthUnit = "meters", primaryAreaUnit =
"sqmeters", activeColor = "#0aaff1", completedColor = "#c70a20") %>%
      addLayersControl(
        overlayGroups = c("BrannsmitteLinje"),
        options = layersControlOptions(collapsed = FALSE)
      )
  })
```

Kode 11: Måleverktøy.

2.3.5 Widgets

I vår applikasjon har vi tatt i bruk eksterne tjenester fra Yr, Statens vegvesen og MazeMap. Tjenestene er lagt inn som widgets, som vil gi sanntidsdata direkte i applikasjonen. Dette sikrer koblinger til nøyaktig og oppdatert informasjon. Ved å integrere tjenestene i appen, gir vi sluttbruker et bredere spekter av relevant data.

For værinformasjon har vi lagt til en widget fra Yr, en tjeneste fra Meteorologisk institutt. Tjenesten gir detaljerte værprognoser og visuelle meteogrammer. Dette er nyttig i beredskapssammenheng der værdata kan påvirke beslutningsprosesser. Værdata kan være nyttig for nødetaten ettersom vindstyrke og retning er faktorer som kan endre situasjonsbildet. Dataen er lagt i et *iframe*- element som peker direkte til Yr sin tjeneste. Dette sikrer at data alltid er oppdatert (Hecking & Cialfi, u.å.).

Vi la også inn MazeMap som en widget. Denne tilbyr detaljerte plantegninger og navigasjonsmuligheter innenfor noen offentlige bygg og campusområder. Dette verktøyet kan hjelpe beredskap til å navigere i ukjente bygninger.

En annen widget som kunne gi relevant informasjon var Statens vegvesen. Widgeten gir sanntids oppdateringer om trafikkforhold, veiarbeid og stengte veier. I kartet kan man også tegne og måle opp polygoner.

Samlet sett bidrar eksterne widgets'ene til å gjøre vår applikasjon til et sentralt sted for informasjon. Ved å samle denne informasjonen på ett sted, kan bruker ta bedre og mer informerte beslutninger basert på oppdatert data. Dette bidrar til at man kan prioritere hvilken informasjon som er relevant ut ifra situasjonen.

I Kode 12 kan vi se et utklipp av hvordan vi kobler til widgets. Output refererer til det spesifikke outputelementet i UI. *Tags$iframe* skaper et *iframe* HTML-element, slik at vi kan bruke tredjepartstjenester i dashbordet vårt. HTML element er strukturen på en nettside. Vi bruker *iframe* i pakken *Shiny* til å hente inn eksternt innhold fra andre nettsteder.

```
  # Implementer MazeMap-widget
  output$mazemapWidget <- renderUI({
    tags$iframe(style = "width:100%; height:600px;", src =
"https://use.mazemap.com/#v=1&zlevel=1&center=5.347446,60.369517&zoom=10.7&
campusid=9")
  })
```

Kode 12: Eksempel på implementering av widgets i dashbordet, her MazeMap-widget.

2.3.6 Nettverksanalyse

Som en annen romlig analyse har vi valgt å gjennomføre en nettverksanalyse. Veinettverksanalyse muliggjør identifisering av den korteste ruten mellom to spesifiserte adresser eller noder. Nettverksanalyse representerer et analytisk verktøy som tillater utforskning av ulike scenarier, inkludert beregning av kortest vei og evaluering av hvor mange adresser som kan nås av brannvesenet innenfor en gitt tidsramme. Dette verktøyet er ikke bare nyttig for forebyggende tiltak, men spiller også en rolle i håndteringen av nødsituasjoner. Som beskrevet i kapittel 1.3.10 s.25 involverer nettverksanalyse kompleks bruk av grafer, som bidrar til en forståelse av romlig data (Hecking & Cialfi, u.å.).

For å kunne utføre en nettverksanalyse trenger vi data til å lage veinettverket. I vårt tilfelle har vi brukt veinettverksdataen fra Geonorge, publiser at Statens vegvesen (*NVDB*, u.å.).

I R har vi brukt pakken *igraph* til å lage graf og finne korteste vei i en analyse. Vi har også brukt pakkene *Simple features sf*, *leaflet* og *dplyr*. *Sf* gjør at vi enkelt kan jobbe med romlig data i R. Vi bruker denne pakken til å lese inn filer og transformere data mellom forskjellige referansesystem (Pebesma, u.å.). Med *leaflet* kan vi lage et bakgrunnskart som man kan navigere og zoome rundt i (*Features*, u.å.). Vi bruker *dplyr* til å effektivt kunne manipulere data. Vi bruker funksjoner som «filter», «summarise» (Wickham et al., u.å.).

På forhånd har vi filtrert veinettet med Bergens kommunenummer i tabellen, som er 4601. Dermed jobber vi bare med veinettsdata tilhørende Bergen kommune. Videre bruker vi *st_read* til å lese inn SQLite-filen. For å sørge for at veinettsdataen er i samme

referansesystem som adressetabellen i PostgreSQL, bruker vi *st_transform*. I Kode 13 er et utklipp som illustrerer.

```
veinett <- st_set_crs(veinett, 25833)
veinett <- st_transform(veinett, 4326)
```

Kode 13: Transformerer til koordinat referanse system (CRS) 4326.

Datasettet forberedes ved å lage en rettet graf. Geometrien settes som null, slik at vi lettere kan manipulere datasettet som en dataframe, uten å behandle det som romlig data. Det er illustrert i Kode 14. Vi velger kolonene «fromnode», «tonode» og «drivetime_fw». Vi filtrerer «drivetime_fw» for å sørge for at det ikke er noen negative verdier og for å unngå feil i grafen. Vi sørger også for at «fromnode» og «tonode» blir lest inn som heltall (*integer*). Dette gjør vi for å sikre korrekt indeksering og effektiv behandling av dataen. Til slutt bruker vi i*graf* til å opprette grafen *g*.

```
edges <- veinett %>%
  st_set_geometry(NULL) %>%
  dplyr::select(fromnode, tonode, drivetime_fw) %>%
  filter(drivetime_fw >= 0) %>%
  mutate(fromnode = as.integer(fromnode), tonode = as.integer(tonode))

g = graph_from_data_frame(edges, directed = TRUE)
```

Kode 14: Setter geometri til NULL.

Videre opprettes funksjonen *get_adress_geometry* for å kunne søke opp to adresser fra databasen i PostgreSQL. Vi har konstruert spørringene slik at man kan søke på adresser både med og uten bokstav på slutten, som forklart i kapittel 2.3.1 s.42 og illustrert i Kode 15.

```
get_address_geometries <- function(start_adressenav, start_nummer,
start_bokstav = NULL,
                                   end_adressenav, end_nummer, end_bokstav
= NULL) {
  # SQL query start addresse
  if (is.null(start_bokstav)) {
    start_query <- sprintf("SELECT geom AS start_geom FROM
public.\"MATRIKKEL_AdressePunkt\" WHERE adressenav = '%s' AND nummer = %s",
                           start_adressenav, start_nummer)
  } else {
    start_query <- sprintf("SELECT geom AS start_geom FROM
public.\"MATRIKKEL_AdressePunkt\" WHERE adressenav = '%s' AND nummer = %s
AND bokstav = '%s'",
                           start_adressenav, start_nummer, start_bokstav)
  }

  # SQL query slutt adresse
  if (is.null(end_bokstav)) {
    end_query <- sprintf("SELECT geom AS end_geom FROM
public.\"MATRIKKEL_AdressePunkt\" WHERE adressenav = '%s' AND nummer = %s",
                         end_adressenav, end_nummer)
  } else {
    end_query <- sprintf("SELECT geom AS end_geom FROM
public.\"MATRIKKEL_AdressePunkt\" WHERE adressenav = '%s' AND nummer = %s
AND bokstav = '%s'",
                         end_adressenav, end_nummer, end_bokstav)
  }
```

```
  # Utfør og les geometry
  start_geometry <- st_read(con, query = start_query)
  end_geometry <- st_read(con, query = end_query)

  # Transform 4326
  start_geometry <- st_transform(start_geometry, 4326)
  end_geometry <- st_transform(end_geometry, 4326)

  return(list(start_geometry = start_geometry, end_geometry =
end_geometry))
}
```

Kode 15: Søker på adresser i PostgreSQL.

Vi lager også en funksjon slik at vi kan finne nærmeste noder til adresser. Det gjør vi ved å regne ut distansen mellom geometrien og veinettet, og videre finne hvilke noder som har minimal distanse.

```
get_nearest_node_info <- function(geometry, veinett) {
  # Finner distanse til alle
  distances <- st_distance(geometry, veinett)
  # Find nærmeste
  nearest_index <- which.min(distances)

  nearest_node_info <- veinett[nearest_index, c("fromnode", "tonode")]

  return(nearest_node_info)
}
```

Kode 16: Nærmeste node til adresse.

Vi utvider Kode 16 med logikk for å finne nærmeste sluttnode for både start- og sluttpunkt basert på adresser. I Kode 17 kaller vi opp «get_adress_geometries», hvor vi henter geometriene til adressene og bruker «get_nearest_node_info» for å finne nærmeste node. Funksjonen tar inn adresser og bestemmer start og sluttnoder til lokasjonene.

```
get_address_geometries_and_nearest_nodes_info <- function(start_adressenav,
start_nummer, start_bokstav = NULL,
                                                          end_adressenav,
end_nummer, end_bokstav = NULL) {
  # Hente adresse geometri
  geometries <- get_address_geometries(start_adressenav, start_nummer,
start_bokstav,
                                       end_adressenav, end_nummer,
end_bokstav)

    # Hente nærmeste node info for start geometri
  nearest_start_node_info <-
get_nearest_node_info(geometries$start_geometry, veinett)

  if (nrow(geometries$end_geometry) > 1) {
    nearest_end_node_info <- lapply(1:nrow(geometries$end_geometry),
function(i) {
      get_nearest_node_info(geometries$end_geometry[i, ], veinett)
    })
  } else {
    nearest_end_node_info <- get_nearest_node_info(geometries$end_geometry,
veinett)
  }
```

```
  return(list(nearest_start_node_info = nearest_start_node_info,
nearest_end_node_info = nearest_end_node_info))
}
```

Kode 17: Definerer start og sluttnode.

Kode 18 er en utvidelse av Kode 16 og Kode 17. Koden søker opp adresser og finner nærmeste node. Videre sjekker vi om de finnes i grafen «g» og deretter beregnes korteste ruten med *shortest_paths* fra pakken *igraf*.

Vi vekter analysen med «drivetime_fw». Da vil den finne den raskeste veien i tid, og ikke nødvendigvis distanse. Videre konverterer vi ruten til en liste over node ID-er, og filtrerer kantdataen fra «veinett» for å inkludere kanter som en del av ruten. Vi beregner også total kjøretid ved å summere alle «drivetime_fw» verdiene i ruten.

Med *mode* bestemmes hvilken retning kantene skal følges i for å finne den korteste ruten i nettverket. *Vpath* danner en liste som inneholder sekvensen av noder langs den korteste veien. Vi har valgt å bruke *Dijkstra*, som algoritme, for å finne korteste vei. Dette gjør vi ved å koble sammen alle nodene slik at avstanden fra startnoden til enhver annen node er kortest mulig. Kantenes vekter må ha positive verdier for at algoritmen kan fungere (Worboys & Duckham, 2004a). Kode 18 viser korteste vei med vekting kjøretid.

```
# Regner ut korteste vei, med vekting kjøretid
route <- shortest_paths(g, from = V(g)[name == start_node], to = V(g)[name
== end_node], mode = "all",weights = edges$drivetime_fw,algorithm =
"dijkstra")$vpath[[1]]

# Konverterer ruten til en nodeID liste
route_ids <- V(g)$name[route]

# Inkluderer kantdataen
route_edges <- veinett[veinett$fromnode %in% route_ids & veinett$tonode
%in% route_ids, ]

# Regner og printer total kjøretid
total_driving_time <- sum(route_edges$drivetime_fw, na.rm = TRUE)
print(total_driving_time)
```

Kode 18: Korteste vei.

Videre visualiserer vi ruten på to ulike måter. Første tilnærming bruker plotfunksjonen til å plotte ruten i veinettverket. Dette blir en statisk måte å presentere ruten på. Den andre tilnærmingen var å plotte ruten i et kart med bruk av *leaflet*. Her har vi brukt *st_simplify* for å lettere håndtere dataen. I Kode 19 presenteres metodiske valg i scriptet, mens resultatet blir presentert i kapittel 3.1.8 s.61.

```
# Plot
plot(st_geometry(veinett), col = 'grey', main = "Ruteberegning")
if (nrow(route_edges) > 0) {
  plot(st_geometry(route_edges), col = 'red', add = TRUE)
} else {
  message("No edges found for the calculated route.")
}
# Forenkling av rutegeometrien
route_edges <- st_simplify(route_edges, preserveTopology = TRUE)
```

```
leaflet(data = route_edges) %>%
  addTiles() %>%  # Adds the default OpenStreetMap tiles
  addPolylines(color = 'red')  # Adds the routes as red polylines
```

Kode 19: Forenkling av rutegeometri.

Nettverksanalysen vår kan i tillegg til drivetime, forbedres ved å vektlegge kolonnen «oneway». Cellene i kolonnen er kodet med «FT», «TF» og «B». Kodene forteller oss om kantene er fra-til, til-fra eller begge veier. Refererer til Figur 9 i kapittel 1.3.10 s.25. Ved å håndtere kantene på denne måten, sørger vi for at ruteberegningen tar hensyn til kjøreretning. Kode 20 viser et forsøk på hvordan vi har prøvd å utvikle dette. Vi tar for oss dette temaet videre i kapittel 3.1.8 s.61 Nettverksanalyse og om forbedringer i kapittel 4 s.70. diskusjon.

```
# Forbered data for grafkonstruksjon
edges <- veinett %>%
  st_set_geometry(NULL) %>%
  dplyr::select(fromnode, tonode, drivetime_fw, oneway) %>%
  filter(drivetime_fw >= 0) %>%
  mutate(
    fromnode = as.integer(fromnode),
    tonode = as.integer(tonode),
    oneway = as.character(oneway)
  )
# Eksplisitt håndtere enveiskjøringer
edges_ft <- filter(edges, oneway == "FT")
edges_tf <- filter(edges, oneway == "TF")
edges_b <- filter(edges, oneway == "B")

# Samle alle kantene til en enkelt dataramme
all_edges <- bind_rows(edges_ft, edges_tf, edges_b)
# Opprett en graf
g <- graph_from_data_frame(all_edges, directed = TRUE)
```

Kode 20: I dette kodeutklippet viser hvordan vi har prøvd å konstruere grafen med hensyn på kjøreretning.

2.3.7 Romlig indeksering

For å kunne gjøre kompliserte romlige spørringer effektivt i databasen, så har vi lagt til romlige indekser. Slike indekser er nyttige for spørringer som utfører romlige analyser. Som omtalt i kapittel 1.3.9 s.24 indekseres data i et søketre som raskt kan krysses for å finne en bestemt post (*Romlig indeksering*, u.å.).

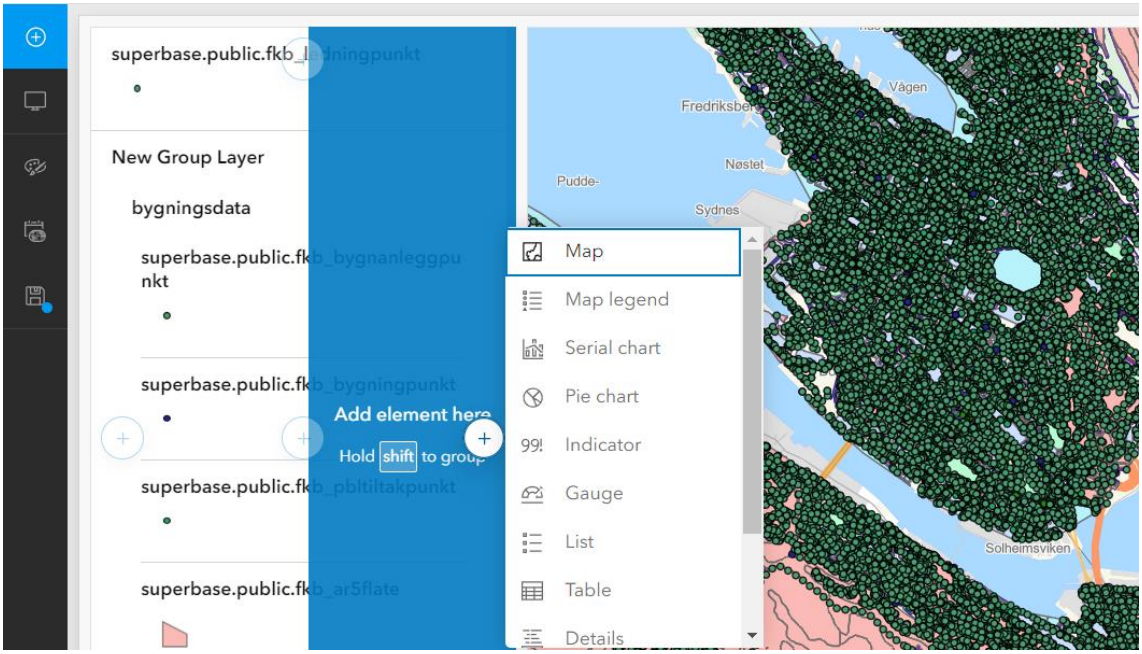
Standard databaseindekser lager et hierarkisk tre basert på verdiene til kolonnen som indekseres. Romlige indekser åpner opp for muligheten til å bryte ned rommet i hierarkiske strukturer, som rektangler og bokser. Innenfor PostGIS benyttes en R-tre indeksstruktur. Denne indekstypen justerer seg selv for å effektivt håndtere variasjoner i datatetthet og størrelsen og overlappingen av objekter (*Romlig indeksering*, u.å.). Kode 21 viser hvordan vi romlig indekserer tabellene i PostgreSQL.

```
CREATE INDEX ON public."FKB_HoydekurveLinje" USING GIST (geom);
```

Kode 21: Romlig indeksering av tabeller ved hjelp av PostGIS-pakken

2.4 Muligheter ved proprietær programvare: ArcGIS

Som en del av vår sammenligning av open source verktøy i R, har vi utforsket mulighetene som finnes i proprietær programvare ved å bruke ArcGIS. ArcGIS, utviklet av ESRI, inkluderer et bredt spekter av forhåndsdefinerte verktøy som kan tas i bruk med lite til ingen forhåndsoppsett (*Web GIS Mapping Software*, u.å.). Det vil si at ArcGIS tilbyr maler for eksempelvis dashbord som illustrert nedenfor, med ArcGIS Dashboards.



Figur 22: Utklipp av noen maler som tilbys via ArcGIS Dashboard.

I vårt prosjekt har vi opprettet et kart i ArcGIS Pro og koblet dette til vår PostgreSQL-database. Dette lar oss utnytte alle forhåndsdefinerte analyser og verktøy som ArcGIS tilbyr, slik som romlig analyse, datahåndtering og avansert kartografi. Funksjonene gjør det mulig for oss å bearbeide og visualisere dataen på en måte som er både effektiv og intuitiv. Figur 22 viser et utklipp av ArcGIS Dashboard med muligheter for å benytte seg av ulike funksjoner.

En betydelig fordel med ArcGIS er dets innebygde funksjonalitet for samarbeid og deling. Med ArcGIS Online, kan kart og prosjekter enkelt lastes opp og deles med andre medlemmer av gruppen. Dette gjør at man kan arbeide på samme prosjekt samtidig og se hverandres endringer (*ArcGIS Online Implementerings-Guide*, 2022).

Videre kan vi integrere kartet i et ArcGIS Dashboard. Dette er en plattform som gjør det mulig å opprette interaktive dashbord, som kan inneholde kart, målere, grafer og andre visualiseringsverktøy. Dashbordet kan tilpasses og konfigureres for å vise sanntidsdata, noe som er ideelt for å følge med på dynamiske og endrende datasett (*ArcGIS Dashboards*, u.å.).

Selv om det er mange fordeler med ArcGIS, er det viktig å vurdere potensielle ulemper som kostnader og mindre fleksibilitet sammenlignet med open source-alternativer, som gir muligheter for tilpasning og utvidelse ved hjelp av ulike pakker.

3. Resultat

I dette kapittelet vil vi presentere hva vi har funnet ut på bakgrunn av teorien som er gjennomgått og utviklingsarbeidet i metode. Vi vil presentere beregninger, illustrasjoner og observasjoner fra arbeidet.

Resultatet vårt presenteres i et dashbord, som skal være en prototype for et ferdigprodukt som brannvesenet skal kunne ta i bruk. Dashbordet har ulike funksjoner som vil bli presentert. Vi har også laget et underkapittel som vil vise casen som ble introdusert innledningsvis, her illustrerer vi funksjonene og hvordan de kan anvendes i praksis.

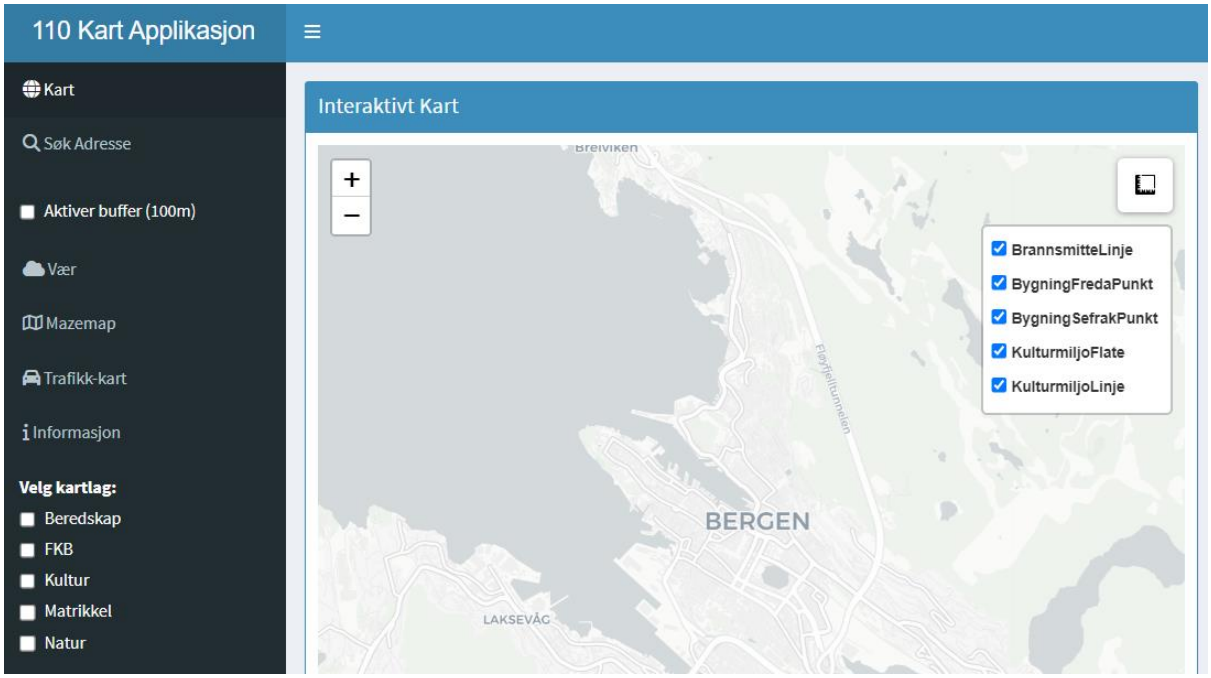
For å sikre en mer sammenhengende forståelse av våre analyser og verktøy, har vi utviklet en video kalt "Video Dashbord". Videoen gir en mer detaljert gjennomgang av vårt resultat, og gir seerne et helhetlig innblikk i funksjonaliteten og nytteverdien av dashbordet vi har utviklet.

3.1 Ferdig produkt: Shiny APP

Gjennom arbeidet har vi designet en app for Bergen brannvesen med et dashbord som komponent. Dette har resultert i et verktøy som kan bidra til bedre håndtering og visualisering av kartdata. Dashbordet tilbyr oppdatert kartinformasjon i et brukervennlig grensesnitt.

3.1.1 Dashbord

Dashbordet vi har utviklet er utformet for å være intuitivt og brukervennlig, slik at man raskt kan navigere og hente ut informasjon i pressede situasjoner. Den sentrale funksjonen i dashbordet er det interaktive kartet, som gir en umiddelbar visuell representasjon av ulike datakilder. Brukergrensesnittet tillater enkel tilgang til forskjellige datakategorier, noe som sikrer at relevant informasjon er lett tilgjengelig. Nedenfor vil vi presentere dashbordet og funksjonene. Vi har delt inn dashbordet i interaktivt kart, adressesøk, værvarsel, MazeMap, Statens vegvesen kart, bufferanalyse, nettverksanalyse og verktøysmeny.



Figur 23: Illustrasjon av dashbord.

Figur 23 er forsiden til dashbordet. Når appen starter åpnes det interaktive kartet. Appen er bygget opp med en sidemeny som kan skjules eller vises. I kartet har vi koblet opp data fra databasen. Vi har delt opp data i ulike kartlag som Beredskap, Felleskartdatabase (FKB), Kultur, Matrikkel og Natur. Bruker kan umiddelbart visualisere og analysere data som er relevante for deres spesifikke oppdrag.

Hver kategori og underkategori av data er nøye utvalgt for å gi en omfattende oversikt over forhold på et gitt sted. For eksempel, i en redningsoperasjon kan beredskapslaget raskt identifisere nødvendige ressurser og hindringer i området. Bruken av underkategorier som linjer, punkter og flater gir ytterligere detaljer som kan hjelpe i planlegging og utførelse av innsatser. Figur 23 er dette illustrert til høyre. Denne modulære tilnærmingen til datahåndtering sikrer at bruker ikke bare har tilgang til omfattende informasjon, men også kan håndtere denne informasjonen på en måte som reduserer kompleksitet og potensiell forvirring under kritiske situasjoner.

Selve oppbyggingen og organiseringen av dashbordet er designet for å maksimere brukervennlighet og effektivitet i krisesituasjoner. Sidemenyen, som er en sentral funksjon, tilbyr enkel tilgang til de forskjellige kartlagene og andre funksjoner. Denne fleksibiliteten tillater bruker å tilpasse visningen etter behov. Dette er spesielt nyttig i situasjoner hvor rask tilgang til spesifikk informasjon kan være avgjørende.

3.1.2 Adressesøk

Adressesøk funksjonen i dashbordet er utviklet for å gi rask og nøyaktig lokasjonsinformasjon, noe som er essensielt i nødsituasjoner hvor hvert sekund teller. Denne funksjonen lar bruker enkelt søke etter og lokalisere adresser på et interaktivt kart ved å benytte detaljert informasjon som gateadresser. Det intuitive brukergrensesnittet tillater inntasting av adresser, og systemet prosesserer denne informasjonen for å hente geografiske koordinater fra databasen. Med adressesøk kan vi søke i databasen vår, og få opp en markør hvor adressen blir visualisert i kartet.

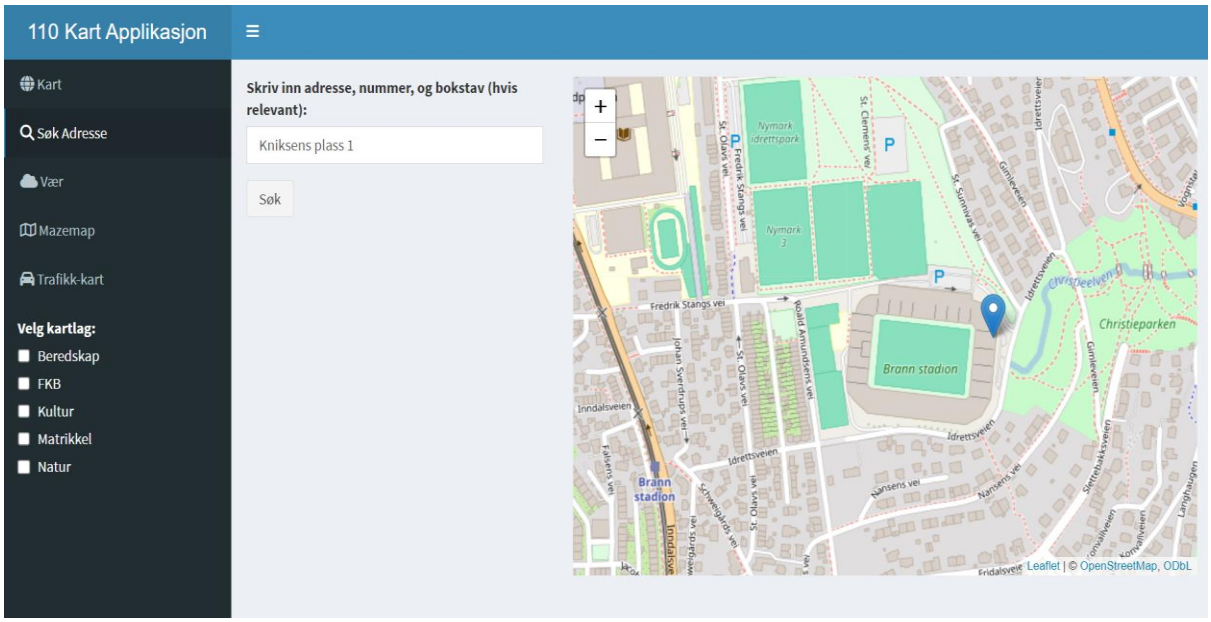
Funksjonen benytter R-skript for å analysere og skille adressenavn, nummer og eventuelle bokstaver. Dette er kritisk for å håndtere adresser hvor bokstaver kan skille bygninger, leiligheter eller innganger innenfor samme bygningsnummer. Avhengig av om en bokstav er inkludert i adressen eller ikke, tilpasses SQL-spørringen for å hente ut de eksakte geografiske koordinatene, latitude og longitude. Koordinatene blir hentet fra PostgreSQL-databasen ved hjelp av PostGIS-funksjoner. Denne fleksibiliteten i spørringslogikken sikrer at resultatene er så presise som mulig, noe som er avgjørende for nøyaktig posisjonering i kartet. Kode 22 viser adressesøk funksjonen.

```
    # Analyser inndata for å skille ut adressenavn, nummer og bokstav
    parts <- strsplit(input$adresse, " ")[[1]]
    adressenav <- paste(parts[-length(parts)], collapse = " ")
    nummer_bokstav <- parts[length(parts)]

    nummer <- gsub("[^0-9]", "", nummer_bokstav) # Fjerner alt unntatt tall
    bokstav <- gsub("[0-9]", "", nummer_bokstav) # Fjerner tall, etterlater
bokstaver

    # Justere spørringen basert på om bokstav er spesifisert eller ikke
    query <- if(nchar(bokstav) > 0) {
      sprintf("SELECT ST_Y(ST_Transform(geom, 4326)) AS latitude,
ST_X(ST_Transform(geom, 4326)) AS longitude FROM
public.\"MATRIKKEL_AdressePunkt\" WHERE adressenav = '%s' AND nummer = %s
AND bokstav = '%s'",
              adressenav, nummer, bokstav)
    } else {
      sprintf("SELECT ST_Y(ST_Transform(geom, 4326)) AS latitude,
ST_X(ST_Transform(geom, 4326)) AS longitude FROM
public.\"MATRIKKEL_AdressePunkt\" WHERE adressenav = '%s' AND nummer = %s
AND (bokstav = '' OR bokstav IS NULL)",
              adressenav, nummer)
    }
    # Utfør spørringen
    result <- dbGetQuery(pool, query)
    dbDisconnect(pool)
    # Vis resultat på kart hvis funnet
    if(nrow(result) > 0) {
      lat <- result$latitude[1]
      lon <- result$longitude[1]

      output$kart <- renderLeaflet({
        leaflet() %>%
          addTiles() %>%
          setView(lng = lon, lat = lat, zoom = 15) %>%
```

```
          addMarkers(lng = lon, lat = lat, popup = paste(adressenav,
nummer_bokstav))
      })
    } else {
      output$kart <- renderLeaflet({
        leaflet() %>%
          addTiles() %>%
          setView(lng = 10.7522, lat = 59.9139, zoom = 5) %>%
          addPopups(10.7522, 59.9139, "Ingen resultater funnet")
      })
    }
  })
```

Kode 22: Adressesøk.

Når adressen er funnet, vises resultatene direkte i kartet. En markør plasseres på den eksakte lokasjonen, og et pop up-vindu viser adressenavnet sammen med nummer og bokstav, noe som gir en klar visuell bekreftelse på søkeobjektets plassering. Dette er spesielt nyttig for nødetater som trenger å forstå hvor ressursene skal settes inn. Hvis adressen ikke blir funnet, blir bruker informert med melding «ingen resultater funnet», noe som umiddelbart indikerer at adressen kanskje må skrives om eller spesifiseres. Figur 24 er et utklipp av dashbordet hvor man kan benytte seg av adressesøk. Kode 22 er tilpasset slik at man kan søke på adresser både med og uten bokstav.



Figur 24: Bilde viser utklipp av dashbordet for funksjonen for adressesøk.

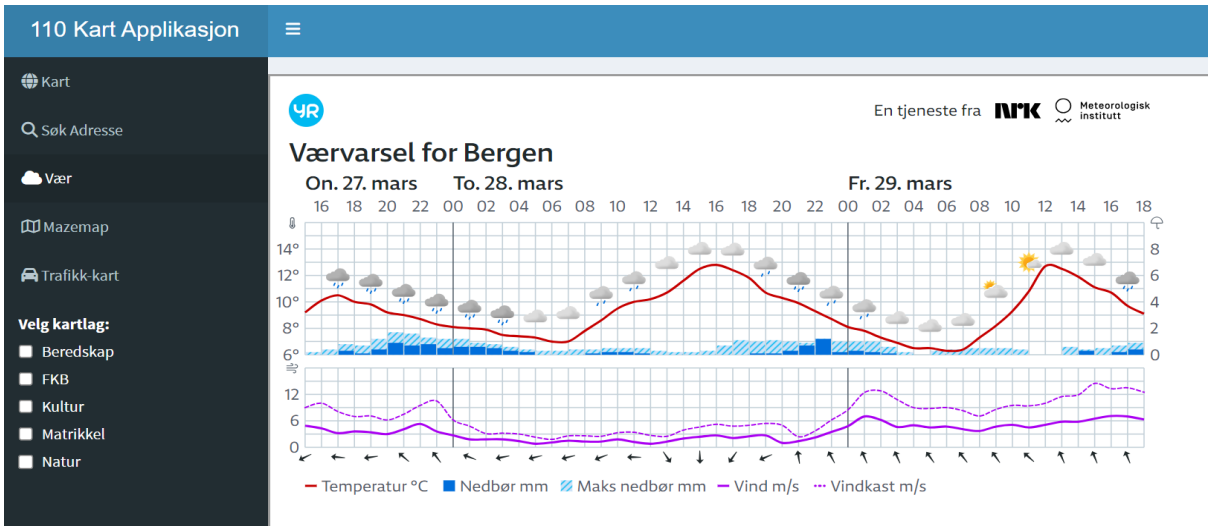
Adressesøk funksjonen gjør spørringer i kolonnene adressenavn, nummer og bokstav som er koloner i «MATRIKKEL_AdressePunkt» tabellen. Videre henter adressesøk-funksjonen matrikkeldata når en adresse er søkt opp. Vi henter ut geometrien i «MATRIKKEL_AdressePunkt» tabellen. Funksjonen er vist i Kode 22. Samlet sett er adressesøk-funksjonen på dashbordet et verktøy som kombinerer avansert databehandling med brukervennlig teknologi for å støtte rask og effektiv navigasjon og planlegging.

3.1.3 Værvarsel

Værvarselsfunksjonen i vårt dashbord støtter oppunder beslutningsprosesser og planlegging. Denne funksjonen gir bruker tilgang til oppdaterte og nøyaktige værdata direkte på brukergrensesnittet. Det kan være kritisk i tilfeller hvor værforhold kan ha stor innvirkning. Værinformasjonen, inkludert temperatur, nedbør, vindstyrke og vindretning, presenteres i et klart og forståelig format, noe som muliggjør raske og informerte beslutninger under tidskritiske operasjoner.

Vår integrerte vær-widget henter data i sanntid fra Meteorologisk institutt. Funksjonen er designet for å gi en kontinuerlig strøm av værdata, noe som sikrer at informasjon som vises er aktuell. Dette er spesielt viktig i scenarier hvor værforholdene stadig skifter og hvor vinden kan påvirke røykutvikling. Et annet eksempel er tørke og høye temperaturer som over lengre tid øker sannsynlighet for skogbrann. Widget'en viser værdata på en måte som er enkel å forstå, og den er tilpasset for å fremheve de værfaktorene som er mest relevante for nød- og redningstjenester, som ekstrem nedbør, stormvarsler og spesifikke vindforhold.

Når vi trykker på «Vær» hentes vær-widgeten for Bergen inn. Etter samtale med de som sitter på sentralen, forsto vi at det ikke var nødvendig med et «live værkart», men mer nyttig med en lett leselig værmelding. Med denne widgeten får vi temperatur, nedbør (mm), maks nedbør (mm), Vind m/s, vindkast (m/s) og vindretning. Figur 25 viser hvordan værvarselet blir presentert, både kommende dager og time for time.



Figur 25: Utklipp fra dashbordet som viser hvordan værvarsel for Bergen blir presentert.

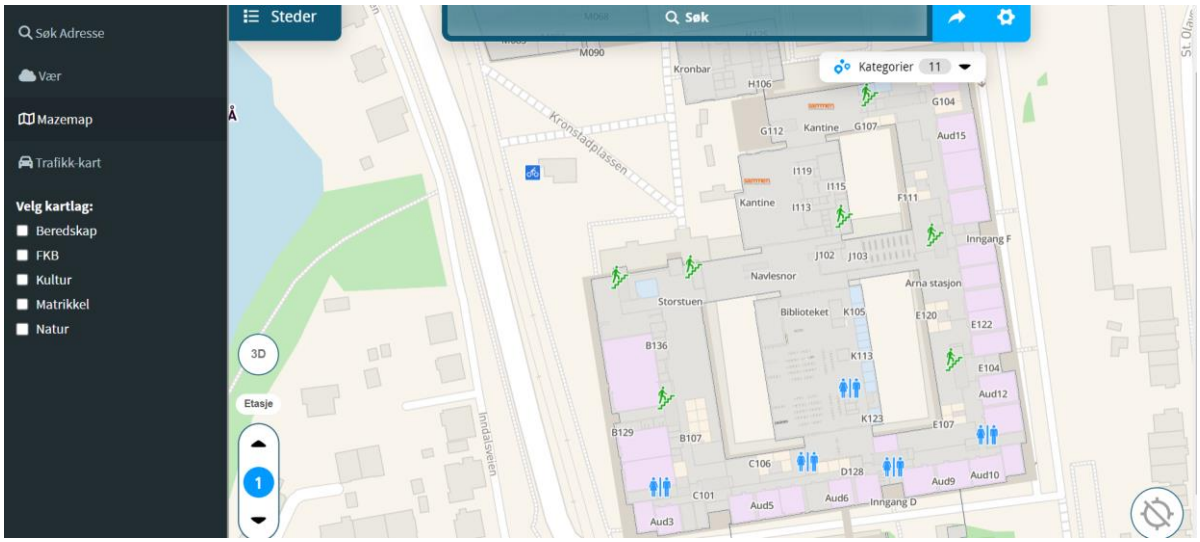
I tillegg til å presentere nåværende forhold, kan vær-widgeten også gi prognoser som hjelper til i planleggingsprosesser. Denne fremtidsrettete kapasiteten gjør det mulig for sentralen å tilpasse strategier basert på forventede værforhold. Ved å integrere informasjonen direkte inn i operasjonelle dashbord, kan brukeren enkelt konsultere værdata uten å måtte veksle mellom forskjellige applikasjoner eller plattformer, noe som sparer tid og kan redusere risikoen for feil.

Samlet sett er værvarselsfunksjonen i dashbordet en ressurs for nød- og redningstjenestene. Den gir dem muligheten til å forstå og reagere på værrelaterte risikoer med større presisjon og sikkerhet. Ved å tilby nøyaktig og tilgjengelig værinformasjon styrkes både planleggingen og utførelsen av innsatser, noe som bidrar til å beskytte liv og eiendom i situasjoner der værforhold spiller en kritisk rolle.

3.1.4 MazeMap

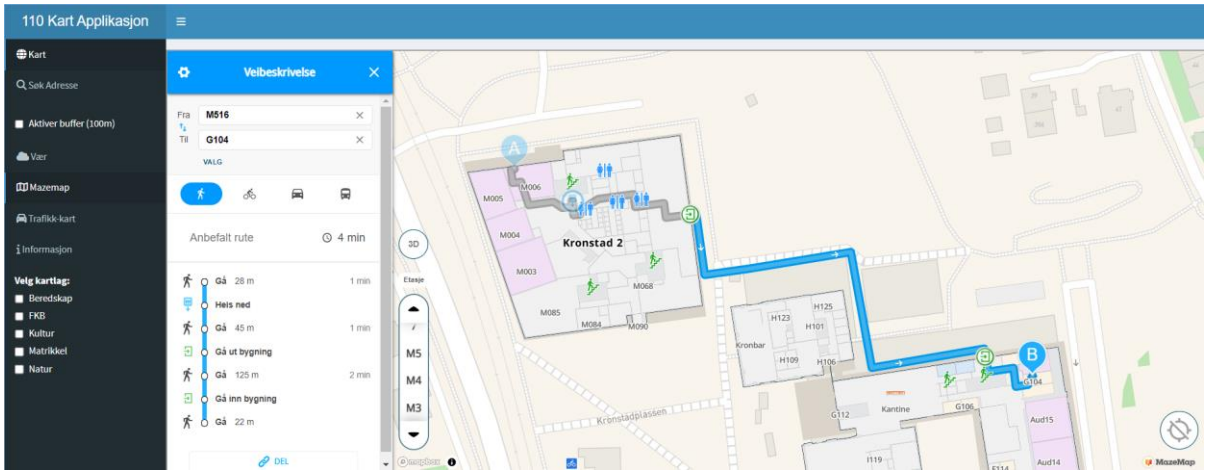
MazeMap-widgeten i dashbordet er et verktøy som tilbyr detaljerte plantegninger og veibeskrivelser i noen offentlige bygninger. Denne widgeten har vi implementert for å forbedre situasjonsbevisstheten og navigasjonsevnen til nødetatene under operasjoner, som i komplekse bygningsmiljøer inkludert skoler, sykehus og offentlige kontorer.

Dette kan spare verdifull tid for redningstjenester som navigerer i ukjente eller komplekse planløsninger. Figur 26 er Høgskulen på Vestlandet sin plantegning for bygning K1, 1. etasje.



Figur 26: Viser en oversikt over MazeMap funksjonen i dashbordet

I tillegg støtter MazeMap integrasjon av sanntidsdata, noe som ytterligere forbedrer brukerens evne til å gjøre informerte beslutninger basert på oppdaterte forhold. Dette bidrar til å sikre at de operative enhetene kan reagere mer presist og effektivt under press.



Figur 27: Viser en oversikt over MazeMap verktøyet i dashbordet, med fokus på veibeskrivelse.

Figur 27 er veibeskrivelse fra grupperom «M516» til «G104». Dette kan være et nyttig verktøy for raskere lokalisering av personer og farekilder. Denne funksjonen tilbyr detaljerte, trinn-for-trinn rutebeskrivelser. Tilgangen til nøyaktige veibeskrivelser kan reduserer tiden det tar å nå frem til ulykkesstedet. Ved å presentere klare og enkle veibeskrivelser, sikrer MazeMap at redningsmannskapet kan handle raskt og effektivt. MazeMap kan også gjøre det enklere for innringer å oppgi en mer presis lokasjon til sentralen.

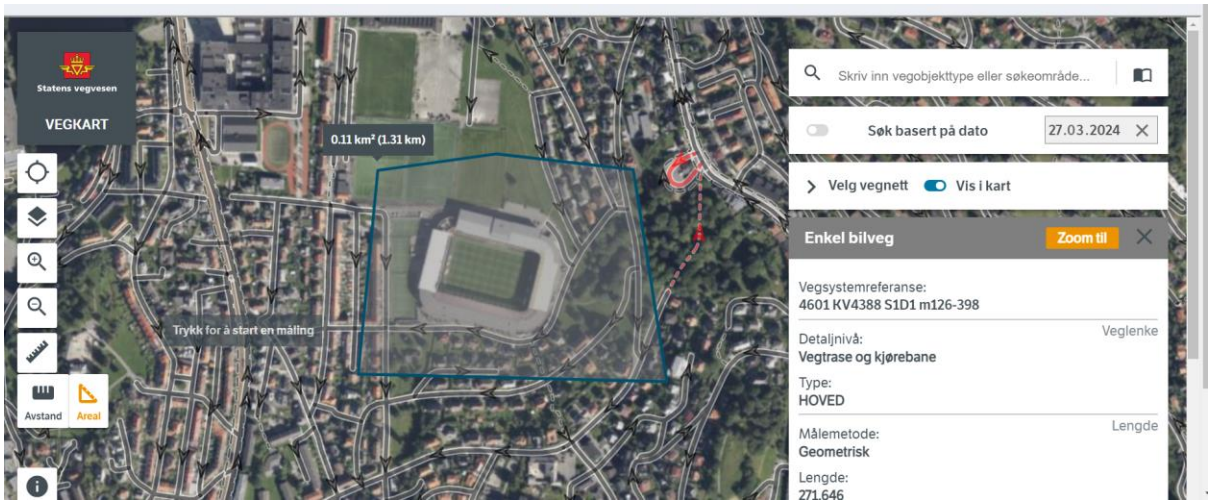
Samlet sett forsterker MazeMap-widgeten beredskapsstyrkens kapasiteter ved å tilby detaljert og pålitelig navigasjonsstøtte der den er mest nødvendig. Denne funksjonen gir informasjon om innendørs navigasjon, noe som kan hjelpe nødetatene med å planlegge og gjennomføre operasjoner med større sikkerhet.

3.1.5 Statens vegvesens

Statens vegvesens vegkart er et nyttig verktøy for nødetater som krever oppdatert og detaljert informasjon om veiforhold. Dette verktøyet gir brukeren tilgang til omfattende veikart som inneholder informasjon om alt fra trafikkflyt og veiarbeid til eventuelle hindringer eller ulykker som kan påvirke navigasjonen.

Ved å integrere vegkartet direkte på dashbordet, kan operatører i nødtjenestene raskt vurdere og planlegge de mest effektive rutene til et innsatsområde. Dette inkluderer muligheten til å skifte mellom ulike kartvisninger som Norgeskart, flyfoto og gråtonekart, noe som gir en bedre forståelse av geografien og topografien i området. Verktøyet støtter også muligheten til måling av avstand og areal direkte i kartet, noe som er nyttig for logistisk planlegging og for å forstå skalaen av et innsatsområde. Denne integrasjonen er viktig for å sikre at nødetatene har nøyaktig og tilgjengelig informasjon i sanntid.

Figur 28 viser Statens vegvesen sitt vegkart. Denne plugin-en har mange nyttige verktøy. Man kan søke på områder og stedsnavn, men ikke adresser. For eksempel kan man søke på Kronstad, men ikke «Inndalsveien 28».



Figur 28: Bildet er et utklipp av dashbordet som viser Statens vegvesen sitt vegkart.

Statens vegvesen sitt trafikkart er bygget på DATEX-formatet, som gir en visning av trafikkmeldinger i Norge. Som illustrert i Figur 15, benytter dette kartet versjon 2.3 av DATEX-formatet. Kartet viser kun røde punkter innenfor en kontur av Norge. Det ville vært optimalt å inkludere bakgrunnskart for å gi bedre kontekst til trafikkmeldingene. Når man klikker på et rødt symbol, får man tilgang til tilhørende attributtdata. Trafikkmeldingene kan inkludere informasjon om veiarbeid, stengte veier og plasseringen av fotobokser. Attributtdataen som vises inkluderer blant annet ID, dato, sted, type trafikkmelding og en beskrivelse av situasjonen.

Figur 29 viser en mer detaljert attributtinformasjon for et spesifikt punkt. På grunn av lengden på attributtabellen, ønsker vi å fremheve den informasjonen vi anser som mest relevant og interessant.

VegmeldingerPunkt

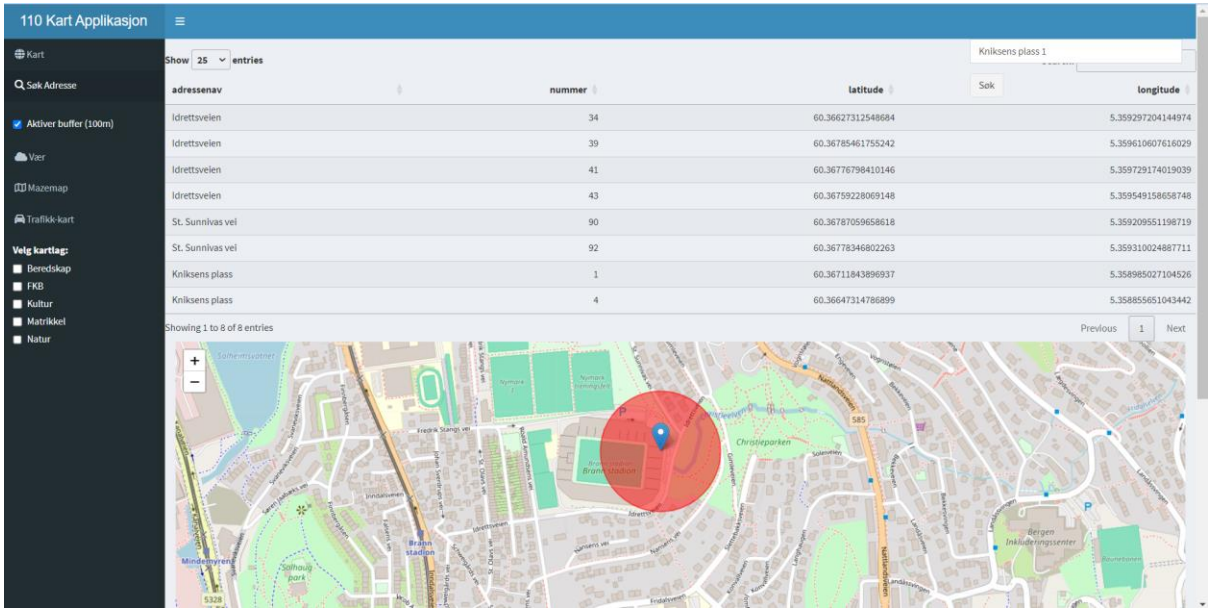
fid	IMPORT_KILDE	SIST_OPPDATERT	SITUATION_ID	DESCRIPTION	CREATION_TIME	LOCATION_DESCRIPTION	ROAD_NUMBER
VegmeldingerPunkt.fid--22a241a4_18f7233c2be_-1d31	DATEX	May 13, 2024, 3:42:05 PM	NPRA_HBT_08-05-2024.82291	Vegarbeid.|Manuell dirigering.|Vent på ledebil. Utrykningskjøretøy kan passere.	May 8, 2024, 5:34:23 PM	E16 Stanghelle i Vaksdal, Vestland - E16 Klenslåttunnelen i Vaksdal, Vestland	E16
VegmeldingerPunkt.fid--22a241a4_18f7233c2be_-1d30	DATEX	May 13, 2024, 3:42:05 PM	NPRA_HBT_08-05-2024.82291	Vegarbeid.|Manuell dirigering.|Vent på ledebil. Utrykningskjøretøy kan passere.	May 8, 2024, 5:34:23 PM	E16 Stanghelle i Vaksdal, Vestland - E16 Klenslåttunnelen i Vaksdal, Vestland	E16

Figur 29: Utklipp av VeimeldingerPunkt.

3.1.6 Bufferanalyse

Bufferanalysen i vårt dashbord representerer et verktøy for nødetatene ved planlegging og risikohåndtering. Denne funksjonen gjør det mulig for brukeren å opprette en visuell buffersone på eksempelvis 100 meter rundt en gitt adresse. Ved søk på en adresse, utføres en SQL-spørring mot vår PostgreSQL-database ved hjelp av PostGIS-funksjoner, som beregner de geografiske koordinatene for den angitte adressen. Med denne informasjonen opprettes en buffersone med bruk av *ST_Buffer-funksjonen*. For at bruker kan se hvor buffersonen er,

oppretter vi en sirkel i kartet rundt den aktuelle adressen. Buffersonen i kartet lager vi direkte med *leaflet* funksjonen *AddCircle*.



Figur 30: Bildet illustrerer bufferanalyse og mulig berørte adresser i buffersonen.

Figur 30 viser hvordan denne visualiseringen gir brukeren en klar og umiddelbar indikasjon på området som kan bli berørt. Det kan være avgjørende ved evakuering og andre kritiske innsatser. Videre bruker vi *ST_Within* -funksjonen for å identifisere alle adresser innenfor den opprettede bufferen. Resultatene vises både som en liste i en tabell og visuelt på kartet. Denne innsikten er spesielt nyttig for å koordinere på tvers av nødetater og bedre kommunikasjonen med berørte parter.

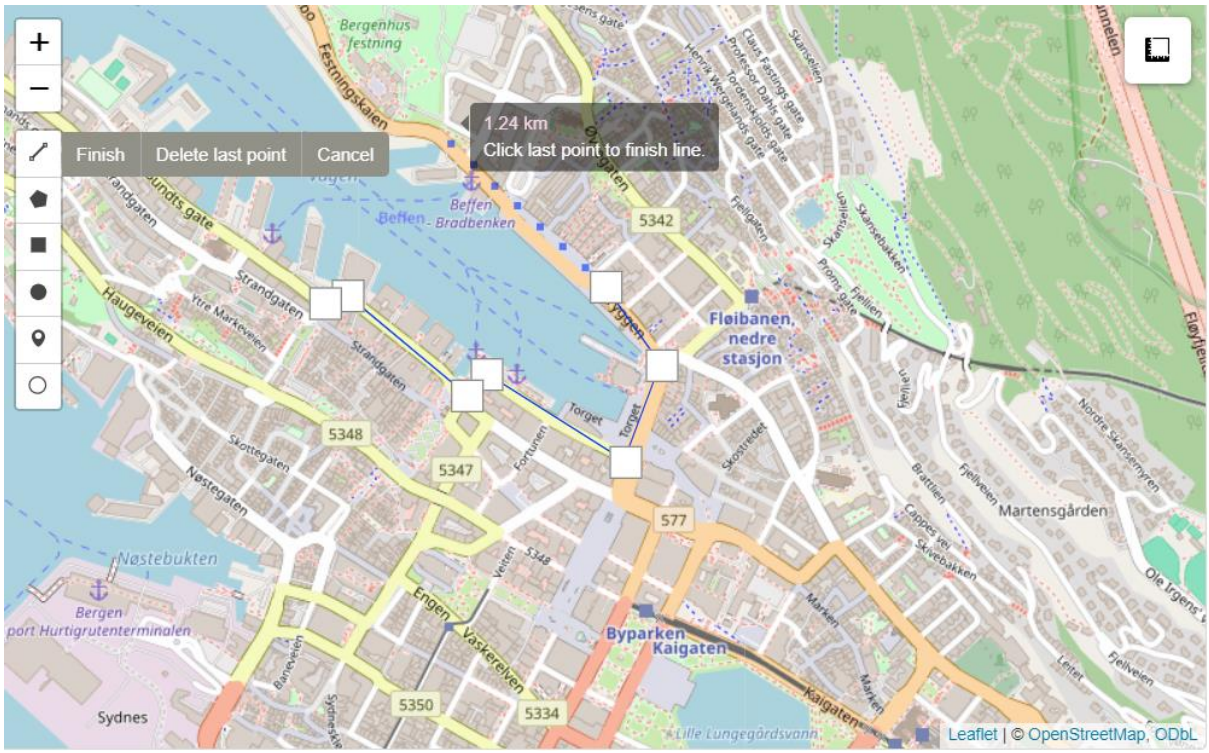
En forbedring av denne bufferanalysen inkluderer potensialet for integrasjon av miljødata, slik som vindretning og vindstyrke, noe som kan være avgjørende for å forstå og planlegge for spredning av røyk, kjemikalier eller andre farlige stoffer. Dette blir videre omtalt i kapittel 4. s.70.

3.1.7 Måleverktøy

Verktøysmenyen i kartet inneholder en rekke verktøy som er designet for å assistere nødetatene med å utføre målinger, identifisere geografiske koordinater og skape tilpassede sikkerhetssoner direkte på kartet. Hver funksjon i verktøysmenyen er nøye tilpasset behovene som oppstår i kritiske situasjoner.

For eksempel, måleverktøyet i kartet tillater brukeren å nøyaktig måle avstanden mellom to punkter eller beregne arealet av et spesifikt område. Ved å bruke dette verktøyet kan innsatsstyrkene få en bedre forståelse av de fysiske dimensjonene til en hendelse. Figur 31 viser en oversikt over dette verktøyet. Verktøyet kan brukes til å markere viktige steder, som tilgangspunkter, for eksempel brannhydranter, og potensielle hindringer eller områder som krever spesiell oppmerksomhet.

Samlet sett forbedrer verktøysmenyen i kartet nødetatenes kapasitet til å utføre oppgaver med presisjon og effektivitet. Med tilgang til brukervennlige verktøy kan dashbordet bidra til en mer koordinert og sikker håndtering av nødsituasjoner.



Figur 31: Utklipp fra menykartet som viser avstand, i dette tilfellet 1.24 km.

3.1.8 Nettverksanalyse

Nettverksanalysen er utformet slik at brukeren kan oppgi adresser for å finne den korteste ruten mellom to steder. Figur 32 viser veibeskrivelsen mellom Bergen brannstasjon og Brann Stadion. Analysen er vektet med «drivetime_fw» og «all» som mode. Dersom analysen utføres uten vekting, prioriteres den korteste distansen fremfor andre faktorer som tid og veiretning. Resultatet av en slik analyse kan avvike fra faktiske forhold, ettersom den ofte favoriserer mindre veier og unngår større hovedveier.

Figur 32: Bildet viser veibeskrivelse fra Bergen brannstasjon til Brann Stadion.

Ved nærmere gjennomgang av ruten blir det klart at den trenger ytterligere justeringer. Vi har en teori om at dette problemet kan løses ved å benytte kolonnen «oneway» under konstruksjonen av grafen. I denne kolonnen er kantene markert som «FT» for fra-til, «TF» for til-fra, og «B» for begge retninger. Vi har ikke klart å konstruere Kode 20 slik at dette fungerer i praksis, men viser til forsøket presentert i 2.3.6 Nettverksanalyse s.46. Figur 33 illustrerer hvordan ruten plasserer seg i det motgående kjørefelt. Den blå linjen viser hvordan ruten faktisk skal gå. Dermed kan vi justere kantene i datasettet før grafen konstrueres. Dette blir videre diskutert i kapittlet 4. Diskusjon s.70. Man kan også se scriptet hvor vi vekter «oneway» i appendikset. Vi har sammenlignet ruten i vår analyse og Google Maps sin. Google Maps sin kartfunksjon anbefaler samme rute, men ikke kjøring i motsatt kjørefelt.

Figur 33: Utklipp av ruten vår vises i rødt. Blå viser korrekt rute.

Figur 34 illustrerer hvordan ruten er plottet inn i veinettet, presentert på en statisk måte som ikke tillater interaktiv navigering eller orientering i kartet. Ruten er fra Lungegårdskaien 40 til Kniksen plass 1. Det grå linjene representerer nettverket, mens den røde linjen er ruten.

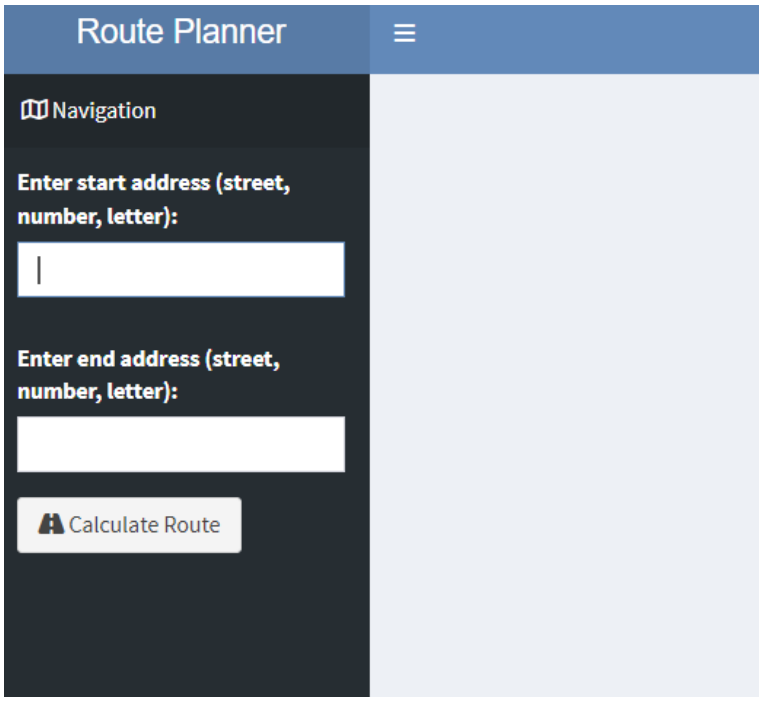
Figur 34: Statisk måte å vise veibeskrivelsen i veinettet.

Vi har også regnet ut reisetiden mellom punktene ved å summere opp drivetime_fw kollonen i route dataen vi lager. Figur 35 viser den estimerte tiden mellom Lungegårdskaien 40 og Kniksens plass 1. Tiden presenterer kun kjøretiden med hensyn til noden. Den tar ikke hensyn til trafikk, trafikklys, enveiskjørtegater eller andre forhold som kan påvirke tiden.

```
st_as_s2(): dropping Z and/or M coordinate
st_as_s2(): dropping Z and/or M coordinate
[1] 5.912269
st_as_s2(): dropping Z and/or M coordinate
```

Figur 35: Estimert tid. Summert kjøretid fra Lungegårdskaien 40 til Kniksens plass 1.

Nettverksanalysen er foreløpig ikke implementert i dashbordet, men brukergrensesnittet er designet som vist i Figur 36. Brukeren oppgir start- og sluttadresse, og ved å trykke på «Calculate Route» vises ruten på kartet.



Figur 36: Interface til dashbordet.

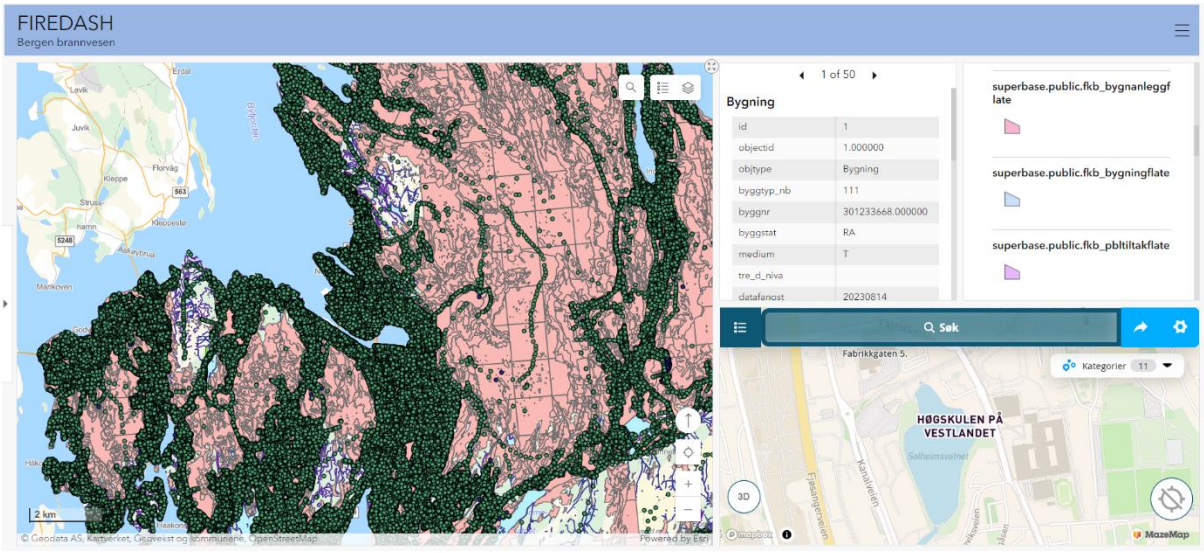
3.2 ArcGIS

I dette underkapittelet vil vi hovedsakelig utforske og presentere resultatene oppnådd gjennom bruk av ArcGIS Dashboards. Vi vil se på funksjonaliteten og effektiviteten av ArcGIS. Fokuset vil være på å vurdere og fremheve de spesifikke fordelene som tilbys med et særlig blikk på brukervennlighet, tilgjengelighet av analytiske verktøy og integrasjonsevner.

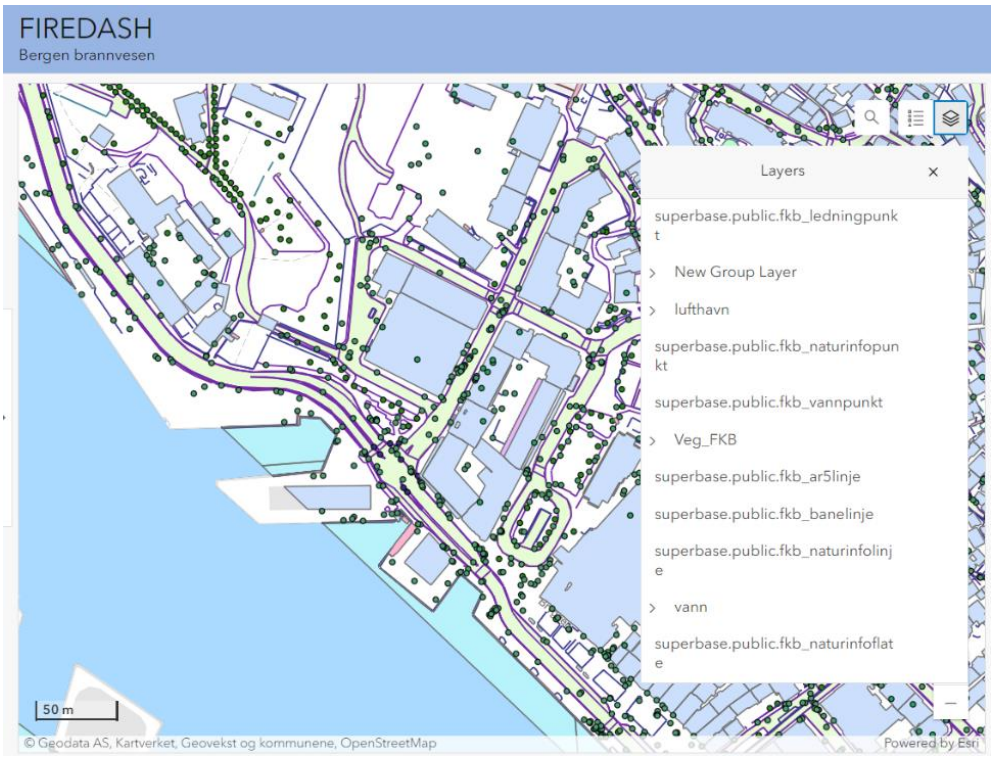
Vi ønsket å utforske hva ArcGIS kunne tilby oss. Det er på bakgrunn av at vi ønsket å undersøke forskjellene mellom open source og proprietær kildekode. Ved å utarbeide et dashbord i ArcGIS ville det gi oss en bredere forståelse av forskjellene, og fordeler og ulemper med de ulike programvarene. Samtidig gir det større læringsutbytte og øker vår kunnskapsbase. Vi har valgt å bruke R gjennom RStudio, som er en open source-programvare, mens ArcGIS er en proprietær programvare. Proprietær programvare er programvare hvor utvikleren beholder kontrollen over kildekoden og vanligvis er tilgangen begrenset (*Proprietær vs Åpen Kildekode*, u.å.).

Gjennom ArcGIS har vi utviklet et dashbord kalt *FIREDASH*. Figur 37 viser et utklipp fra dashbordet, som er organisert i flere seksjoner for brukervennlighet. Den første delen inneholder et kart som viser geografisk data, hvor brukeren kan velge hvilke kartlag de ønsker å vise. På siden av kartet finner man en attributt-tabell som gir detaljert informasjon om det valgte kartlaget eller det punktet man klikker på skjermen. Det ser man et enkelt utklipp av på Figur 38. Der er det tydelig hvordan man kan justere kartlag under *Layers*. Videre har vi integrert MazeMap i dashbordet, som tillater brukeren å arbeide på en måte som ligner funksjonaliteten i Shiny app. MazeMap er et eksempel på at det er mulig å sette inn Widgets

på lik linje som i Shiny. Til slutt inneholder dashbordet en *legend*, som brukeren kan velge å skjule etter behov. ArcGIS Dashboards har en innebygd funksjon som utvider og lukker de ulike panelvinduene.



Figur 37: Bildet er et utklipp av dashbord i ArcGIS for å teste åpen kontra lukket kildekode.

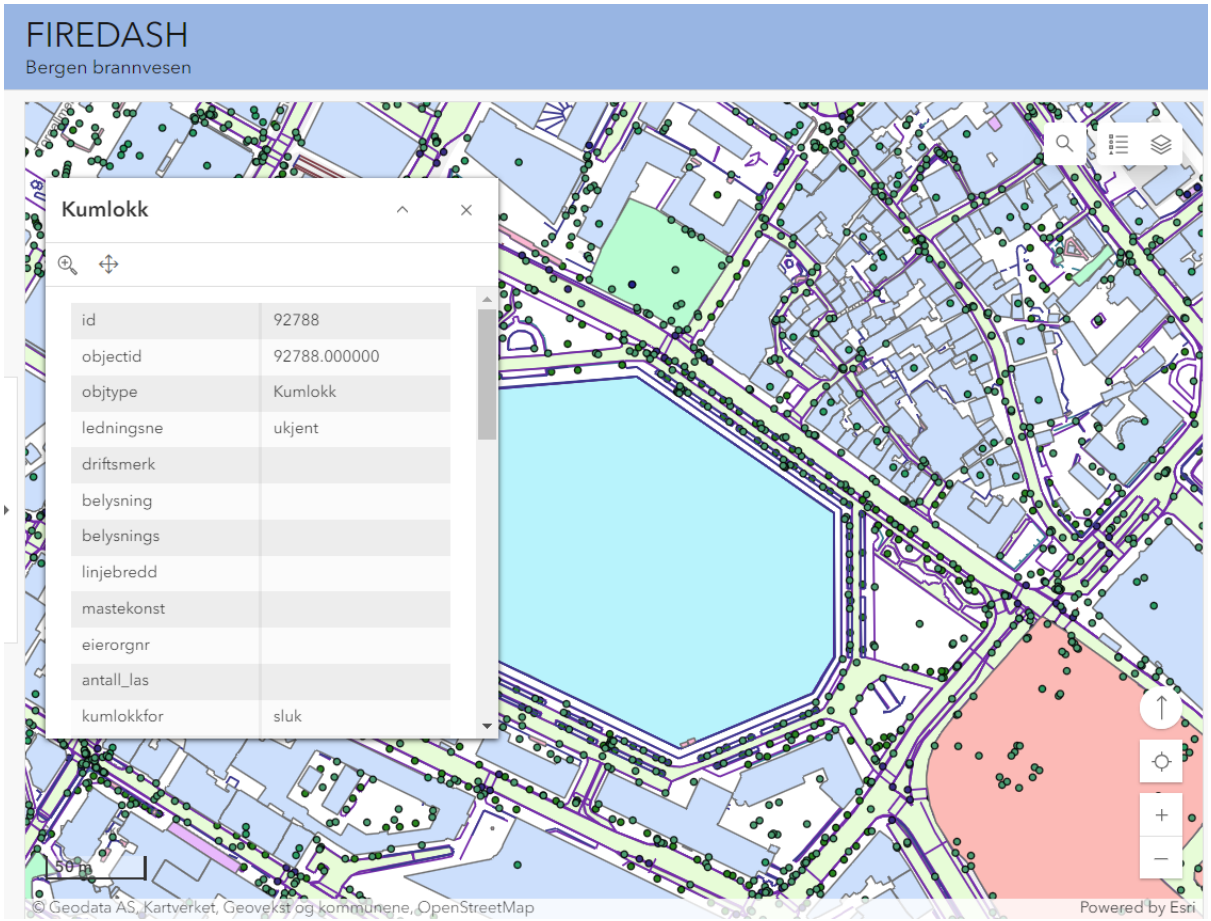


Figur 38: Bildet er et utklipp av FIREDASH i ArcGIS hvor man kan redigere i Layers.

Vi har lagt inn data om eksempelvis kumlokk ettersom det vil være fornuftig for brannvesenet å kunne lokalisere. Ved å trykke på symbolet får man opp mer om punktet og ytterligere informasjon.

Figur 39 ser vi et detaljert kart som viser kumlokk lokalisert rundt byen. Dataen er verdifulle for brannvesenet da rask og nøyaktig lokalisering av ulike objekter kan være kritisk i

nødsituasjoner. Når man klikker på et punkt på kartet, åpnes et informasjonsvindu som gir ytterligere detaljer om det spesifikke punktet, som identifikasjonsnummer (ID), objekttype, og andre relevante attributter som kan være nyttige. Kartet inneholder funksjoner som lar brukeren tilpasse visningen ved å velge hvilke data-lag som skal vises, noe som øker interaktiviteten og brukervennligheten. På siden av kartet vises attributt-tabellen, som gir en oversikt over og lett tilgang til all tilgjengelig data knyttet til hvert punkt som er valgt eller trykket på.



Figur 39: Bildet er et utklipp av FIREDASH i ArcGIS som viser detaljer rundt markert punkt i kartet.

Selv om det har blitt gjort en rask og grunnleggende utvikling av dashbordet i ArcGIS for å utforske programmets potensial, har vi valgt å ikke fokusere på å forbedre detaljene. Hensikten var først og fremst å øke vår forståelse av mulighetene som ligger i proprietær kildekode. Dette viser at symbolikken ikke er brukt korrekt. For eksempel, i Figur 39 representerer ikke alle markørene nødvendigvis et kumlokk. De markerer en rekke ulike objekt-IDer. Likevel er det mulig å kategorisere objekt-IDene slik at de kan tilordnes passende symboler.

Avslutningsvis understreker våre resultater betydningen av å integrere avanserte GIS-funksjoner og sanntidsdata i operasjonelle dashbord for nødetater. Ved å utnytte teknologiene har vi demonstrert hvordan nøyaktig og tilgjengelig geografisk informasjon forbedrer planleggings- og responsprosesser. Funksjoner som adressesøk, værvarsler,

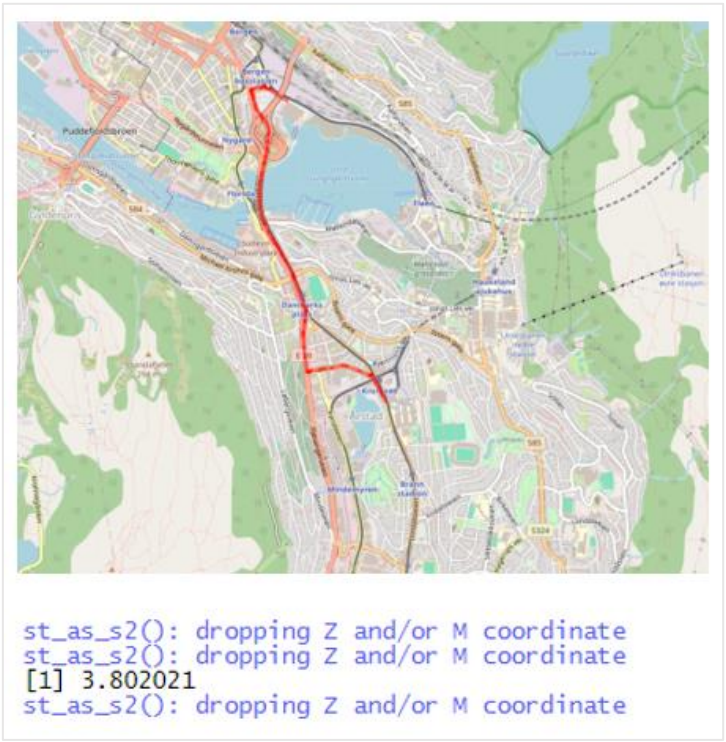
MazeMap-integrasjoner, og verktøysmenyer har vist seg å kunne være viktig i å støtte raskere og mer målrettet innsats under kritiske forhold.

På bakgrunn av resultatene vi har oppnådd, er det noen svakheter i arbeidet som vi har kommet over. Det er likevel noe som er mulig å forbedre eller jobbe videre med. Vi vil kjøre en kort sammenligningsstudie med Shiny App under kapittelet 4. Diskusjon s.70, hvor vi sammenligner de for å gi et overblikk over deres styrker og svakheter.

3.3 Case

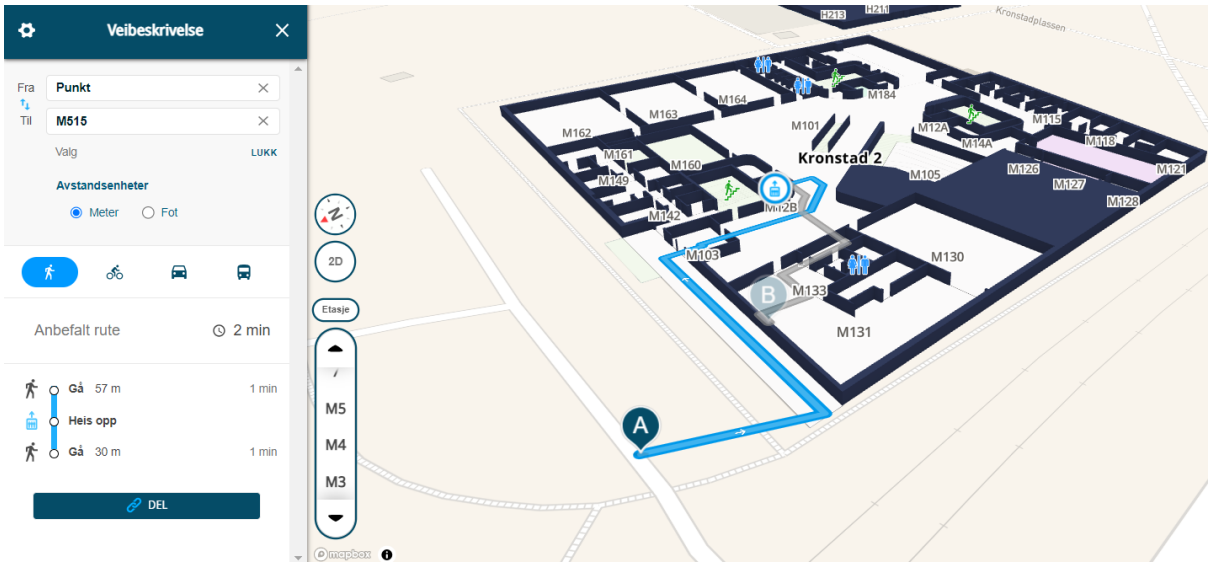
I introduksjonen nevnte vi en case for å illustrere hvordan dashbordet er funksjonelt for brannvesenet. Vi har valgt å simulere en nødsituasjon for å vise dashbordets kapasiteter og funksjonaliteter. Dette eksemplet vil gi innsikt i hvordan teknologiske verktøy og dataintegrasjon kan forbedre beslutningstaking, noe som understreker viktigheten av vår utviklede løsning. Ved å illustrere et utrykningsscenario, håper vi å belyse de praktiske implikasjonene av vårt arbeid og fremheve potensialet for videreutvikling.

Det kommer en telefon til 110-sentralen om brann- og røykutvikling i Inndalsveien 26, 5063 Bergen. Sentralen får en oversikt over tilgjengelige brannbiler som er klar for utrykning. På kjøreturen til Inndalsveien 26 bruker de *Statens vegvesens trafikkart* i dashbordet for å sjekke trafikkmeldinger om veiarbeid og stengte veier. De ser at det er veiarbeid på en sidevei og planlegger ruter deretter. I nettverksanalysen får man opp veibeskrivelse til Inndalsveien 26 og ved å kombinere den informasjonen med Statens vegvesen vegkart, kan det bidra til økt situasjonsforståelse. Figur 40 gir veibeskrivelse og estimert tid.



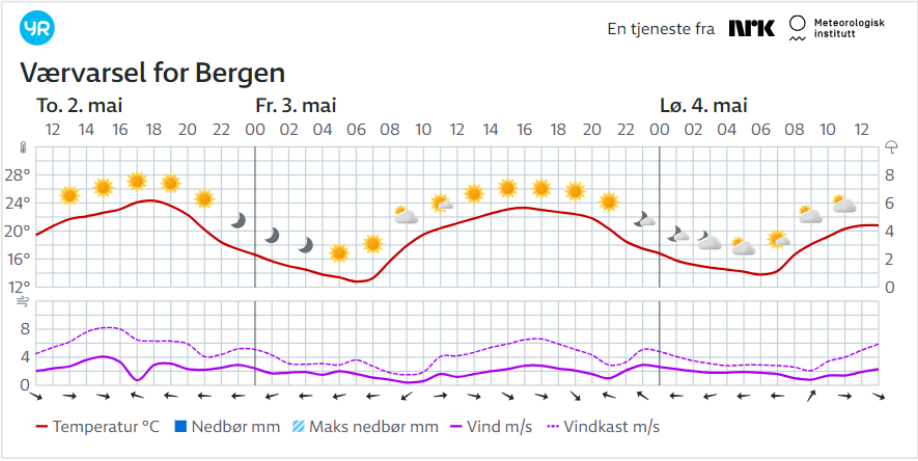
Figur 40: Veibeskrivelse fra Bergen brannstasjon til Inndalsveien 26 og estimert tid.

I bilen på vei til hendelsesstedet bruker de *adressesøk-funksjonen* for å se hvor Inndalsveien 26 er. De ser deretter at det er adressen til Høgskulen på Vestlandet og vet at det er et offentlig bygg som bruker MazeMap. Sentralen lokaliserer innringer i bygning K2 på rom M515. Ved å bruke *MazeMap funksjonen* kan beredskap på stedet lokalisere innringer og lettere nå frem til de, eller veilede dem til nærmeste utgang. Figur 41 viser MazeMap og kan gi bedre situasjonsforståelse, samt bidra til evakueringsplan.



Figur 41: Veibeskrivelse i MazeMap fra gitt punkt utenfor K2 til rom M515.

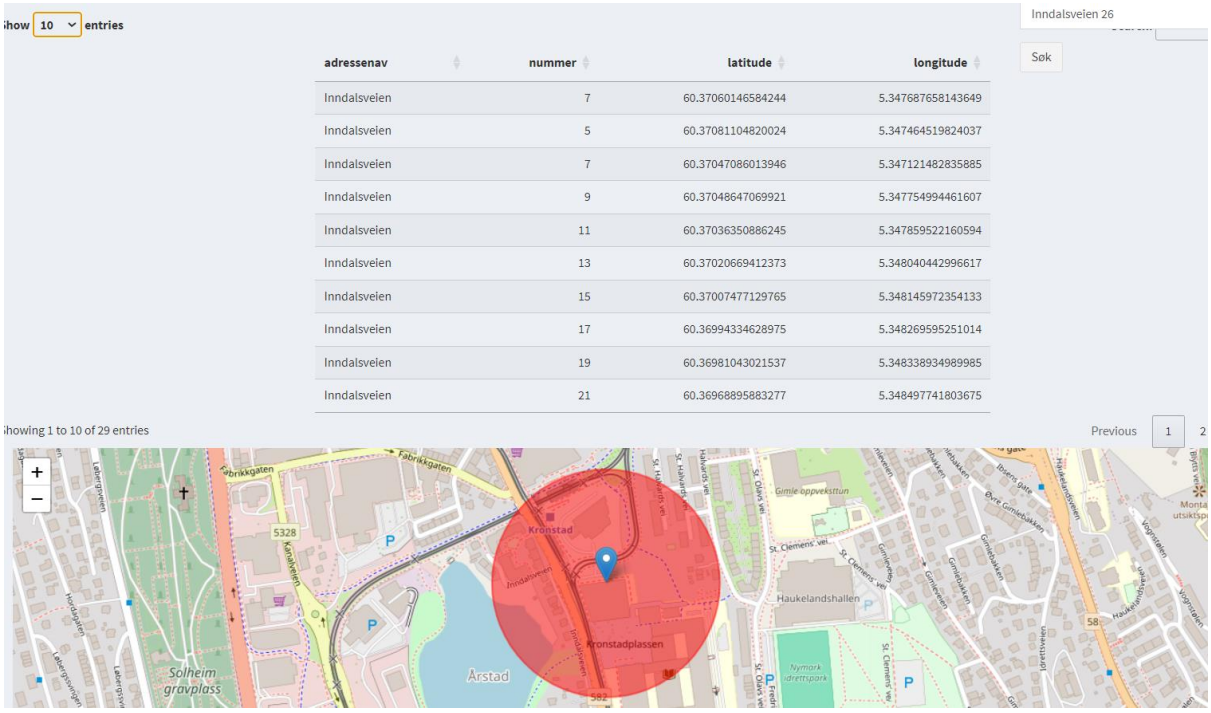
Det er meldt om røykutvikling og på bakgrunn av dette er det fordelaktig å undersøke værvarselet. I vår case har det i løpet av timen blitt meldt om vind med opptil 17 m/s. Dette endrer situasjonen slik at det vil være nødvendig å lage en evakueringsplan for området rundt. Figur 42 viser *værvarsel-widget* med sanntidsdata.



Figur 42: Værvarsel - Bildet kan ikke justeres til casen sin situasjon.

Vi antar at en vanlig buffersone i mildt vær ligger på ca. 50 meter, mens i sterk vind velger beredskapsleder å øke buffersonen til 150 meter. De kan da bruke *bufferanalysen* i dashbordet for å lage en buffersone i ønsket antall meter. I denne buffersonen får vi en oversikt over rammede adresser. Figur 43 viser en buffersone på 150 meter og adressene som

berøres av denne sonen. Som vist i tabellen, identifiseres totalt 29 adresser. Det er imidlertid viktig å bemerke at dette antallet ikke nødvendigvis samsvarer med antallet bygninger. En betydelig andel av enhetene deler samme adresse. Dette fenomenet antyder en mulig hypotese om at flere leiligheter eksisterer under én og samme adresse, dog med ulike leilighetsnummer. I kapittel 4 s.70 vil vi utforske dette avviket nærmere.



Figur 43: Bufferanalyse med sone på 150 meter og berørte adresser.

4. Diskusjon

Gjennom prosjektet har vi møtt på både styrker og svakheter med arbeidet mot et dashbord for Bergen brannvesen. En sentral del av vår oppgave var å samle og analysere data for å utvikle en løsning som kunne styrke deres arbeid. I lys av første del av problemstillingen om hvordan open source GIS-teknologier kan tilpasses brannvesenets ulike behov, med sikte på tilgjengeliggjøring av romlig data, skal vi diskutere resultatet. Vi vil også ta for oss mulighetene ved proprietær programvare og måler opp funn mot de resultatene vi har oppnådd med open source.

Etter å ha reflektert over prosjektet, innså vi at noen aspekter overgikk forventningene, mens andre avdekket områder som kunne forbedres. En funksjon som spesielt utmerket seg, var deler av bufferanalysen. Et forventet resultat ved analysen var muligheten til å velge størrelse på buffersonen. I tillegg til dette klarte vi å koble sammen buffersonen til adresser, slik at vi fikk en tabell med alle potensielt berørte adresser. Dette er en analyse som vi anser som viktig ettersom nytteverdien kan være høy. Analyse kan være viktig ved røykutvikling og eventuelle farer ved lekkasje av farlig gods. Man kan også legge til leilighetsnummer på berørte adresser i buffersonen, slik at man får et enda mer detaljert søkeresultat og som styrker dashbordets funksjonalitet ytterligere.

En annen faktor som styrket prosjektet, var innføring av romlig indeksering. Denne teknikken muliggjør mer effektive og presise søk i databasen, ved at den tillater søk med hensyn til geometrisk data. Dette fører til at kun relevant data selekteres ut fra forespørselen, uten å måtte gjennomgå hele tabellen, noe som gjør spørringen mindre ressurskrevende.

Vi har likevel identifisert rom for forbedringer som kan øke dashbordets funksjonalitet. Forbedringene inkluderer begrensninger i søkefunksjonalitet, datahåndtering og fleksibilitet i analyser.

En begrensning i dashbordet er mangel på dynamisk tilpasning i bufferanalysen når det kommer til faktorer som vindretning. Det kan være avgjørende for å håndtere situasjoner med røykutvikling eller spredning av farlig gods. For å forbedre dette, kan vi implementere sanntids meteorologiske data for å automatisk justere buffersonen i henhold til vindforhold. Dette tiltaket kan øke presisjonen i risikovurderinger. Med mer tid og kompleks programmering, kan dette være innen rekkevidde for videreutvikling.

DATEX er et datasett som inneholder informasjon om veimeldinger som stengte veier og pågående arbeid. Det gir bruker nødvendig informasjon for effektiv navigering. En utfordring vi har erfart er integreringen av DATEX i vårt dashbord. Foreløpig har vi ikke lykkes med å vise informasjonen i kart, og dataen representeres i stedet som røde punkter i dashbordet. Dette fremhever behovet for videreutvikling for å forbedre symbolikken, noe som vil tillate en mer detaljert og informativ visuell framstilling av forskjellige veihendelser.

Utarbeidelsen av nettverksanalysen viste seg å være en lang og krevende prosess, som var mer kompleks enn opprinnelig antatt. En utfordring i prosjektet var mangelen på en visuell

fremstilling av analysen; i stedet ble resultatet presentert i en tabell, noe som gjorde det lite brukervennlig. Da vi til slutt lyktes med å visualisere nettverksanalysen ved å konkretisere dataen som «integer», og ikke «character», forenklet dette videre arbeid og beriket resultatdelen.

Som beskrevet i kapittel 3.1.8 s.61, viste nettverksanalysen veibeskrivelser med kjøring i motsatt kjørefelt, noe som er et avvik. Vi har utarbeidet en kode for å adressere dette avviket. Selv om koden ennå ikke er fullført, burde videre arbeid fokusere spesifikt på å korrigere kjøreretningen. Dette vil involvere en grundig gjennomgang av «one way»-kolonnen, kodet med «FT», «TF», og «B», som vist i Kode 20 i kapittel 2.3.6 s.46. Til tross for at koden ikke er fullført, er det potensial for forbedringer.

Vi støtte også på en utfordring med bruk av widgets, som er hentet fra eksterne kilder. En mulig forbedring kan være å selv integrere dataen inn i dashbordet, ved å benytte seg av HTML-koden tilhørende widgeten, eller deler av denne. En begrensning ved noen av widgets'ene, som for eksempel fra Statens vegvesen, er at de kun tillater søk etter stedsnavn og ikke spesifikke adresser. Man kan dermed søke «Kronstad», men ikke «Inndalsveien 28». Det er viktig å bemerke seg at en rekke leverandører bruker proprietær kildekode som kan sette en begrensing i uthenting av data. Dette gjør det mer utfordrende å anvende og tilpasse dataen etter eget ønske.

En annen utfordring vi har møtt på er knyttet til databaser. I stedet for å operere fra en felles database, har vi benyttet individuelle, lokale databaser på hver vår enhet. Vi benyttet oss av samme datasett og integreringsteknikker, men hadde selvstendig kontroll over vår egen data og database. Selv om dette var en tilstrekkelig løsning, opplevde vi likevel små ulikheter, noe som skapte komplikasjoner ved tilpasning av koder.

En mer samkjørt tilnærming til databasene kunne potensielt redusert utfordringene. For eksempel ved bruk av en skybasert løsning, som Microsoft Azure, kunne dette potensielt forbedret samarbeidet og effektiviteten i datahåndteringen. Bruk av lokale databaser kan minske kostnader og gi bedre kontroll over data, men kan også resultere i ulike dataversjoner blant teammedlemmer. Denne løsningen kan kreve økt administrasjon, noe som potensielt kan gjøre samarbeidet mer komplekst. På andre siden ville en løsning som Azure samlet dataen og muliggjøre sanntidsoppdateringer, noe som forbedrer samarbeidet. Bruken av Microsoft Azure ville imidlertid innebære ekstra kostnader, som er den primære årsaken til at vi har valgt å fortsette med lokale databaser (*Datalagring i Azure*, u.å.).

4.1 Sammenligning mellom open source og proprietær programvare

Gjennom vårt prosjekt har vi undersøkt forskjellene mellom open source- og proprietær programvare, med RStudio og ArcGIS for hver tilnærming. Denne sammenligningen gir et grunnlag for å vurdere fordeler og ulemper.

Vårt valg mellom open source- og proprietær programvare ble påvirket av forskjellene i deres kildekode og funksjonalitet. Vi har investert tid og ressurser i å utvikle løsninger med R i RStudio, og med vårt kunnskapsgrunnlag og verktøy som kunstig intelligens for feilsøking, har vi oppnådd betydelige fremskritt. Likevel erkjenner vi behovet for en dypere forståelse av dataintegrasjonsteknikker.

En faktor å vurdere er brukers ferdighetsnivå og erfaring. Med programmering i R kan det være en bratt læringskurve for nybegynnere som ikke er kjent med programmeringsspråk. I utviklingsarbeidet var programmeringsferdigheter- og erfaringer relevant for resultatet. Dette førte til at utfordringer i programmeringsarbeidet kunne ta lengre tid for tilsynelatende enkle feil. Det understreker behovet for grundig forarbeid slik at teoretisk kunnskap kan gjennomføres praktisk, for eksempel gjennom koding i R. På en annen side kan proprietær programvare tilby enklere brukergrensesnitt og verktøy, som kan være mer tilgjengelig. Mens utvikling i R krever koding fra bunnen av med pakker som «Shiny», tilbyr ArcGIS ulike forhåndsoppsett, blant annet ArcGIS Dashboards. I kapittel 3.2 s.64 viser vi vårt forsøk på ArcGIS Dashboard, som potensielt kan spare tid og ressurser.

Videre er det viktig å vurdere spørsmål knyttet til kostnad og lisens. Open source programvare som RStudio er gratis å bruke, mens proprietær programvare som ArcGIS kan kreve betalt lisens. Open source-programvare fremmer kunnskapsdeling gjennom åpen tilgang til kildekoden, mens proprietær programvare kan bidra til bedre samarbeid gjennom enklere deling av prosjekter. ArcGIS tilbyr blant annet ulike kurs fra ESRI som kan øke brukers kunnskap i programmet.

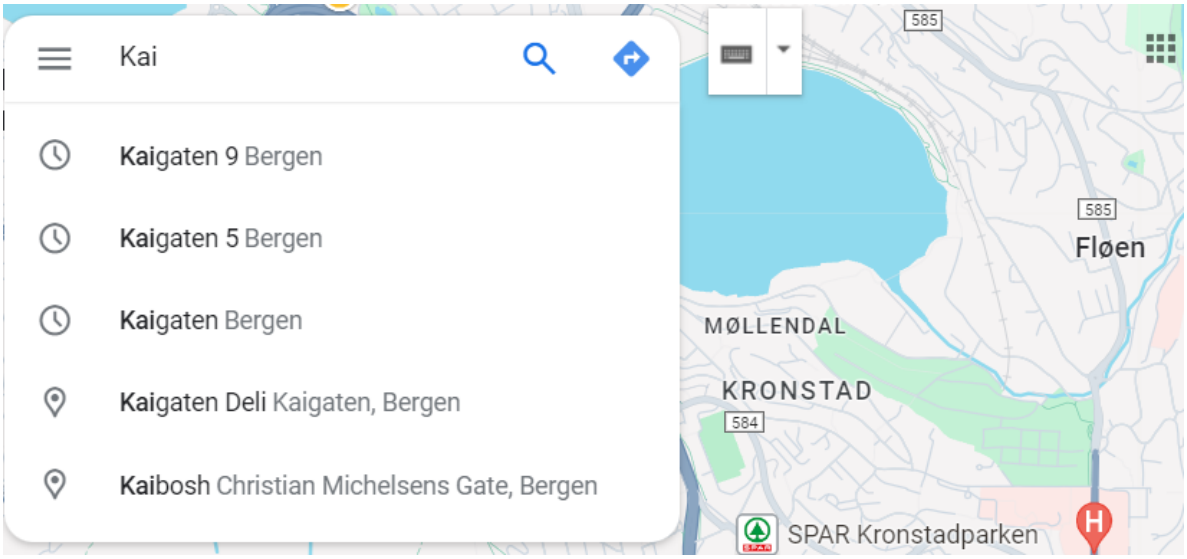
Hovedsakelig har vi valgt å bruke R for utviklingen av vårt dashbord på grunn av fleksibiliteten og tilpasningsmulighetene som open source tilbyr. Dersom prosjektet skal videreutvikles, vil det være gunstig å ha prosjektet basert på open source-programvare. Likevel var vi interessert i å utforske mulighetene som ArcGIS kunne tilby med sin proprietære kildekode. Denne tilnærmingen resulterte i klare forskjeller i utviklingsprosessen. Vi la merke til variasjoner, inkludert tidsforbruk, behov for mindre forkunnskaper, ArcGIS sitt brukervennlige grensesnitt og søkefunksjonalitet. ArcGIS har innebygde analysemuligheter, for eksempel muligheten til å utføre en bufferanalyse med bare et par klikk, mens i R må man skrive kode for å oppnå det samme resultatet.

Gjennom vårt prosjekt har vi fått innsikt i både fordelene og ulempene ved open source og proprietær programvare. Vi har på bakgrunn av kostnader og videreutvikling, valgt å utvikle dashbordet vårt med bruk av open source via R.

4.2 Videre satsing

Etter diskusjon rundt resultatene om styrker, svakheter og eventuelle avvik, er det hensiktsmessig å rette fokuset mot videre satsing. Videreutviklingen bør omhandle hvordan man kan forbedre prosjektet for å oppnå økt kvalitet og funksjonalitet. Dette omfatter både utvikling av programmeringsferdigheter, videreutvikling av analyser, optimalisering av grensesnitt og å opprette en oversikt over ressurser for brannvesenet. I dette underkapittelet vil vi utforske områdene nærmere og presentere mulige strategier og tiltak for videreutvikling.

Bergen brannvesen kom med et ønske om å få alias i adressesøk funksjonen. Alias kan forbedre brukervennlighet og nøyaktighet i søkeprosesser. Systemet innebærer innsamling og integrering av alternative navn eller vanlige feilstavelser av adresser. Selv om bruker inntaster feilaktige betegnelser, kan systemet identifisere den korrekte adressen gjennom en dynamisk nedtrekksmeny som foreslår alternativer basert på brukerinput. En slik funksjon kan redusere tidsbruken i kritiske situasjoner ved å tillate rask korreksjon og bekreftelse av adresser. Alias åpner også muligheten til å søke på punkt av interesse som «HVL» eller «Grieghallen». Figur 44 av Google Maps er et eksempel på bruk av nedtrekksmeny for alias.



Figur 44: Utklipp av Google Maps som illustrerer nedtrekksmeny for alias.

En annen tilnærming som ville løftet dashbordet er å kunne splitte og minimere skjermen etter behov. Dette ville både øke brukervennligheten og forbedret brukers evne til å multitaske. Ved å tilpasse visningen etter personlige preferanser eller oppgavespesifikke krav, kan bruker enklere håndtere og sammenligne flere datakilder samtidig. Videre kan et slikt tilpasningsdyktig grensesnitt bidra til å redusere visuell overbelastning og å tillate bruker å fokusere på mer relevant data.

Videreutvikling av dashbordet bør inkludere en oversikt over tilgjengelig ressurser som utstyr og kjøretøy. Dette er ikke blitt gjennomført i vårt dashbord på bakgrunn av tilgang på data, men er et viktig element for brannvesenet. Det vil tillate sentralen å raskt lokalisere tilgjengelige kjøretøy og ressurser på en effektiv måte, slik at nærmeste og passende ressurs

for oppdraget ankommer innsatsstedet. Integrering av sanntidsdata for ressurstilgjengelighet kan forbedre logistikk og dermed styrke responskapasiteten. Selv om brannvesenet allerede benytter denne funksjonen, vil det være hensiktsmessig å integrere en slik løsning i dashbordet. Ved å samle informasjonsflyten blir dataen lettere tilgjengelige fra én enkelt plattform.

Vi ønsket å tilføye nye funksjoner i dashbordet som kunne vurderes for fremtidig utvikling. En slik funksjon er Google Street View, som kan gi sentralen en visuell representasjon av området og terrenget. Imidlertid støtte utviklingen på utfordringer knyttet til lisensavtaler som regulerer bruken av Google Street View-data. Dette førte til at vi besluttet å ikke prioritere ressurser til denne integrasjonen på det tidspunktet. Likevel, for fremtidig utvikling, kan en slik integrasjon være svært nyttig i mange tilfeller.

En annen nyttig funksjon ville vært et samarbeid med entreprenører for å sikre tilgang til bygningers planløsninger. Ved å integrere detaljerte planløsninger i dashbordet, kan vi tilby sentralen grundig innsikt i bygningenes struktur. Dette kunne vært nyttig under redningsoperasjoner hvor presis informasjon om planløsningen er avgjørende. I tillegg vil slik informasjon kunne styrke avdelingen for forebyggende arbeid ved å gi dem bedre forståelse av bygningenes konstruksjon og potensielle risikoer.

Planene for videreutvikling kan bidra til å forbedre funksjonaliteten og nytteverdien av vår løsning. Det finnes også andre løsninger som kan heve produktet, som eksempelvis marinekart, gridkart eller slope analyse. I lys av vår forskning har GIS et stort potensial for å forbedre nødetatenes arbeid. Potensialet krever realisering med en omfattende tilnærming som løser både teknologiske og organisatoriske utfordringer. Vi poengterer samtidig at dette er et forskningsprosjekt hvor resultatet er oppnådd over en kort tidsperiode.

5. Konklusjon

Vår utviklingsprosess med å utvikle et dashbord for Bergen brannvesen har gitt oss verdifull innsikt i styrker og svakheter med vår tilnærming, samt muligheter for forbedringer.

Gjennom oppgaven har vi utforsket første del av problemstillingen om *hvordan open source GIS-teknologier kan tilpasses brannvesenet ulike behov, med sikte på tilgjengeliggjøring av romlig data*. På bakgrunn av det rike biblioteket som tilbys i open source løsninger, som R, gir det muligheten til å utvikle skreddersydde løsninger for brannvesenets ulike behov med bruk av romlig data. Vi konkluderer med at vårt dashbord representerer et skritt i riktig retning, innen GIS-teknologi.

Ved å utforske andre del av problemstillingen har vi konkludert med at vår forskning i dette prosjektet gir fordeler med open source-programvare. Konklusjonen trekkes på bakgrunn av fordelene med open source, særlig tilpasningsmulighetene og kostnadseffektivitet. Vi konkluderer dog med at proprietær programvare vil være hensiktsmessig for bedrifter som ønsker større satsing innen GIS, med bakgrunn av mulighetene for opplæring og støtte.

Vi kan på bakgrunn av oppgavens helhet konkludere med at GIS spiller en viktig rolle i Bergen brannvesen. Valg av tilnærming og våre funn er ikke bare relevant for Bergen brannvesen, men også for nødetater over hele landet som står overfor lignende utfordringer.

6. Vedlegg

6.1 Vedlegg _bachelor

- *6.1.1 Appendiks*
- *6.1.2 Dashbord*
- *6.1.3 Datagrunnlag_SQL*
- *6.1.4 ER_superbase*
- *6.1.5 Script*
- *6.1.6 Veinettverksanalyse+sqlite*
- *6.1.7 Video_dashbord*

Referanser

1. *110-sentralen*. (u.å.). Hentet 25. april 2024, fra https://abbr.no/om-abbv-2/110-sentralen/
2. Aileen, Buckley. (2012). *Make maps people want to look at*. ESRI. https://www.esri.com/news/arcuser/0112/make-maps-people-want-to-look-at.html
3. *Appropriate Uses For SQLite*. (2024, april 3). https://sqlite.org/whentouse.html
4. *ArcGIS Dashboards*. (u.å.). Hentet 2. mai 2024, fra https://www.esri.com/en-us/arcgis/products/arcgis-dashboards/overview
5. *ArcGIS Online Implementerings-Guide*. (2022). ESRI. https://www.esri.com/content/dam/esrisites/nb-no/media/pdf/implementation-guides/implement-arcgis-online.pdf
6. *Arkitekturen til en geodatabase*. (u.å.). Hentet 5. mars 2024, fra https://pro.arcgis.com/en/pro-app/latest/help/data/geodatabases/overview/the-architecture-of-a-geodatabase.htm
7. Arnøy, M. K. (2018, juni 20). *Sikrere samfunn med sanntidsdata*. Geodata. https://geodata.no/blogg/2018/06/20/sikrere-samfunn-med-samfunnsdata
8. Astrup, N. (2020, januar 14). *Nasjonal strategi for kunstig intelligens* [Plan]. Regjeringen.no; regjeringen.no. https://www.regjeringen.no/no/dokumenter/nasjonal-strategi-for-kunstig-intelligens/id2685594/
9. *Beredskapsavdelingen*. (u.å.). Hentet 25. april 2024, fra https://abbr.no/om-abbv-2/beredskapsavdelingen/
10. *Brann- og redningsvesenforskriften*. (2022, oktober 5). Direktoratet for samfunnssikkerhet og beredskap. https://www.dsb.no/lover/brannvern-brannvesen-nodnett/veiledning-til-forskrift/veiledning-til-forskrift-om-organisering-bemanning-og-utrustning-av-brann--og-redningsvesen-og-nodmeldesentralene/
11. Bratbergsengen, K. (2023). Relasjonsdatabase. I *Store norske leksikon*. https://snl.no/relasjonsdatabase

12. Bratbergsengen, K., & Bothner-By, H. (2023). Grensesnitt. I *Store norske leksikon*. https://snl.no/grensesnitt
13. Bratbergsengen, K., & Mallaug, T. (2024). Database. I *Store norske leksikon*. https://snl.no/database
14. Bratbergsengen, K., Tøssebro, E., & Mallaug, T. (2023). SQL. I *Store norske leksikon*. https://snl.no/SQL
15. Buckley, A. (2011, oktober 28). Design principles for cartography. *ArcGIS Blog*. https://www.esri.com/arcgis-blog/products/arcgis-pro/mapping/design-principles-for-cartography/
16. *Bufferanalyse*. (u.å.). Hentet 30. april 2024, fra https://pro.arcgis.com/en/pro-app/latest/tool-reference/analysis/buffer.htm#S_GUID-95A22984-0A53-4C08-A13E-95C7A93EE17B
17. Changiz, H. (2024, februar 2). *Fire gode grunner til å ha et Dashboard*. https://bas.no/blogg/fire-gode-grunner-til-a-ha-et-dashboard
18. *Creating a Spatial Database*. (u.å.). Hentet 1. mai 2024, fra https://postgis.net/workshops/postgis-intro/creating_db.html
19. Datakvalitet. (2019, april 24). *Digital Opptur*. https://digitalopptur.no/blogg/analyse/datakvalitet-er-det-pa-tide-med-en-datavask
20. *Datalagring i Azure*. (u.å.). Hentet 22. april 2024, fra https://azure.microsoft.com/nb-no/explore/global-infrastructure/data-residency/
21. Dempsey, C. (2023, august 30). *Shapefil-visere*. https://www.geographyrealm.com/shapefile-viewers/
22. *Dette er Bergen brannvesen*. (u.å.). Bergen kommune - Dette er Bergen brannvesen. Hentet 7. mai 2024, fra https://www.bergen.kommune.no/omkommunen/avdelinger/bergen-brannvesen/om-oss/dette-er-bergen-brannvesen
23. Eilertsen, A. (2023). Grafteori. I *Store norske leksikon*. https://snl.no/grafteori

24. *Empower Your Mapping Skills: Essential Components of GIS*. (2023, mai 15). https://www.spatialpost.com/components-of-gis/
25. *Geodata*. (u.å.). Geodata. Hentet 10. april 2024, fra https://geodata.no/om-oss
26. *GitHub Copilot*. (2024). GitHub. https://github.com/features/copilot
27. Gramstad, T. (2023). Åpen kildekode. I *Store norske leksikon*. https://snl.no/%C3%A5pen_kildekode
28. Granum, N. (2023). FKB. I *Store norske leksikon*. https://snl.no/FKB
29. Grolemund, G., Cheng, J., & Cetinkaya-Rundel, M. (2017, august 9). *Shiny—Customize your UI with HTML*. Shiny. https://shiny.posit.co/r/articles/build/html-tags/index.html#tags
30. Grolid, S. Å., & Albertine, A. (u.å.). *UX og UI*. ndla.no. Hentet 8. mai 2024, fra https://ndla.no/image/56211
31. Grønmo, S. (2024). Målenivå. I *Store norske leksikon*. https://snl.no/m%C3%A5leniv%C3%A5
32. *Gårdskart—NIBIO*. (2024). https://gardskart.nibio.no/landbrukseiendom/4601/3/5/0?gardskartlayer=ar5kl7
33. Hecking, T., & Cialfi, D. (u.å.). *Exploring Spatial Network Dynamics Across Multiple Levels*. Hentet 24. april 2024, fra https://www.frontiersin.org/research-topics/62555/exploring-spatial-network-dynamics-across-multiple-levels
34. *Hva er DATEX?* (u.å.). Statens vegvesen. Hentet 17. april 2024, fra https://www.vegvesen.no/om-oss/om-organisasjonen/apne-data/et-utvalg-apne-data/datex/hva-er-datex/
35. *Hva er en geodatabase?* (u.å.). Hentet 5. mars 2024, fra https://pro.arcgis.com/en/pro-app/latest/help/data/geodatabases/overview/what-is-a-geodatabase-.htm
36. *Hva er åpen kildekode?* (u.å.). Hentet 9. april 2024, fra https://opensource.com/resources/what-open-source
37. *IMRaD-modellen*. (2024, februar 13). https://www.sokogskriv.no/skriving/imrad-modellen.html

38. *Introducing ChatGPT*. (2022, november 30). https://openai.com/blog/chatgpt
39. *Introduction to PostgreSQL*. (u.å.). Hentet 9. april 2024, fra https://www.w3schools.com/postgresql/postgresql_intro.php
40. Karambelkar, B., & Schloerke, B. (2022). *Extra Functionality for «leaflet» Package*. https://cran.r-project.org/web/packages/leaflet.extras/leaflet.extras.pdf
41. *Kartkatalogen*. (u.å.). Hentet 10. april 2024, fra https://kartkatalog.geonorge.no/
42. *Kartprojeksjoner og koordinatsystemer*. (u.å.). Hentet 23. april 2024, fra https://www.geonorge.no/aktuelt/om-geonorge/slik-bruker-du-geonorge/kartprojeksjoner-og-koordinatsystemer/
43. Kosourova, E. (2022). *RStudio Tutorial*. https://www.datacamp.com/tutorial/r-studio-tutorial
44. *Leaflet*. (2023, mai 18). https://leafletjs.com/
45. *Leaflet—Features*. (u.å.). Features. Hentet 5. mai 2024, fra https://leafletjs.com/
46. Liseter, I. M. (2023). Public domain. I *Store norske leksikon*. https://snl.no/public_domain
47. Longley, P. A., Goodchild, M. F., Maguire, D. J., & Rhind, D. W. (2011). *Geographic Information System & Science* (3. utg.). Wiley.
48. *Lær om filformater*. (u.å.). Hentet 11. april 2024, fra https://support.microsoft.com/nb-no/office/l%C3%A6r-om-filformater-56dc3b55-7681-402e-a727-c59fa0884b30
49. *MySQL*. (u.å.). Hentet 22. april 2024, fra https://www.w3schools.in/mysql/ddl-dml-dcl
50. *NLOD, 2.0*. (2023, juli 3). https://data.norge.no/nlod/no/2.0#_lisensavtalens_innledning
51. *Norgeskart*. (2024). https://norgeskart.no/#!?project=seeiendom&zoom=12&lat=6734081.97&lon=-33449.44&markerLat=6739000.7055063&markerLon=-51194.68966301&panel=Seeiendom&showSelection=true&p=searchOptionsPanel&layers=1002,1013,1014,1015&sok=Koparhaugen

52. Nottveit, T. (2009). *Digitale kart - en vei til fortiden: Utvikling og evaluering av en kartapplikasjon for historiske bilder* [Master thesis, The University of Bergen]. https://bora.uib.no/bora-xmlui/handle/1956/3922
53. *NVDB*. (u.å.). Hentet 1. mai 2024, fra https://kartkatalog.geonorge.no/metadata/nvdb-ruteplan-nettverksdatasett/8d0f9066-34f9-4423-be12-8e8523089313
54. Nätt, T. H., & Liseter, I. M. (2023). Filformat. I *Store norske leksikon*. https://snl.no/filformat
55. Pebesma, E. (u.å.). *Simple Features for R*. Hentet 5. mai 2024, fra https://r-spatial.github.io/sf/articles/sf1.html
56. *pgAdmin*. (u.å.). Hentet 9. april 2024, fra https://www.pgadmin.org/docs/pgadmin4/8.4/index.html
57. *PostGIS*. (u.å.). PostGIS. Hentet 4. mars 2024, fra https://postgis.net/
58. *PostgreSQL Tutorial*. (u.å.). Hentet 9. april 2024, fra https://www.w3schools.com/postgresql/index.php
59. *Proprietær vs åpen kildekode*. (u.å.). Digitroll. Hentet 9. april 2024, fra https://www.digitroll.no/produkter/nettbutikk/proprietaer-vs-apen-kildekode-nettbutikklosning/
60. QGIS. (2020, september 29). *Hordaland*. https://fnf-nett.no/hordaland/velkommen-til-qgis-kurs/
61. Reddy, G. P. O. (2018). Spatial Data Management, Analysis, and Modeling in GIS: Principles and Applications. I G. P. O. Reddy & S. K. Singh (Red.), *Geospatial Technologies in Land Resources Mapping, Monitoring and Management* (s. 127–142). Springer International Publishing. https://doi.org/10.1007/978-3-319-78711-4_7
62. Rimstad, N. Ø. (2023). Brannvesen. I *Store norske leksikon*. https://snl.no/brannvesen
63. *Romlig indeksering*. (u.å.). Hentet 22. april 2024, fra https://postgis.net/workshops/postgis-intro/indexing.html
64. Rossen, E., & Dvergsdal, H. (2023). XML – IT. I *Store norske leksikon*. https://snl.no/XML_-_IT

65. Rossen, E., & Nätt, T. H. (2024). API. I *Store norske leksikon*. https://snl.no/API
66. *R-tree Hierrarchy*. (u.å.). Hentet 6. mai 2024, fra https://postgis.net/workshops/postgis-intro/_images/index-01.png
67. Rød, J. K. (2023). *GIS Verktøy for å forstå verden* (2. utgave). Fagbokforlaget. https://issuu.com/fagbokforlaget/docs/gis_2._utgave_9788245045222_
68. Rød, J. K., & Dick, Ø. B. (2023). Geografiske data. I *Store norske leksikon*. https://snl.no/geografiske_data
69. *Shapefiler*. (u.å.). Hentet 17. april 2024, fra https://enterprise.arcgis.com/en/portal/latest/use/shapefiles.htm
70. *Shiny*. (u.å.). Shiny. Hentet 4. mars 2024, fra https://shiny.posit.co/r/getstarted/shiny-basics/lesson1/index.html
71. *Shiny—Getting Started*. (u.å.). Shiny. Hentet 22. april 2024, fra https://shiny.posit.co/r/getstarted/build-an-app/hello-shiny/getting-started.html
72. *Shiny—Server function*. (u.å.). Shiny. Hentet 23. april 2024, fra https://shiny.posit.co/r/getstarted/build-an-app/hello-shiny/server-function.html
73. *Shiny—Share your apps*. (u.å.). Shiny. Hentet 5. mai 2024, fra https://shiny.posit.co/r/getstarted/shiny-basics/lesson7/
74. *Shiny—User interface (UI)*. (u.å.). Shiny. Hentet 22. april 2024, fra https://shiny.posit.co/r/getstarted/build-an-app/hello-shiny/user-interface.html
75. Smith, D. M., & Venables, W. N. (2024). *An Introduction to R*. https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf
76. *SQL Commands*. (u.å.). Hentet 19. april 2024, fra https://media.geeksforgeeks.org/wp-content/uploads/20210920153429/new.png
77. *SQLite*. (2023, oktober 10). https://sqlite.org/about.html
78. *SQL-kommandoer*. (2023, oktober 27). GeeksforGeeks. https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/

79. *Trafikkmeldinger | Vegvesen trafikk*. (u.å.). Statens vegvesen. Hentet 18. april 2024, fra https://www.vegvesen.no/trafikk/trafikkmeldinger
80. *Understanding overlay analysis*. (u.å.). Hentet 30. april 2024, fra https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/understanding-overlay-analysis.htm
81. *User guide, DATEX*. (u.å.). Hentet 17. april 2024, fra https://docs.datex2.eu/general/index.html
82. Vijay, K. (2022, august 17). What is Spatial Analysis? Definition and Examples. *Spiceworks Inc*. https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-spatial-analysis/
83. *Web GIS Mapping Software*. (u.å.). Hentet 2. mai 2024, fra https://www.esri.com/en-us/arcgis/products/arcgis-online/overview
84. *What are Application Maps?* (u.å.). Avi Networks. Hentet 9. april 2024, fra https://avinetworks.wpengine.com/glossary/application-maps/
85. *What is a widget*. (2020, januar 28). https://www.arimetrics.com/en/digital-glossary/widget
86. *What is GIS?* (u.å.). Hentet 4. mars 2024, fra https://www.esri.com/en-us/what-is-gis/overview
87. *What is Network Analysis*. (2023, juli 14). Great Learning Blog: Free Resources What Matters to Shape Your Career! https://www.mygreatlearning.com/blog/what-is-network-analysis/
88. *What is Web Application*. (u.å.). Software Quality. Hentet 9. april 2024, fra https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app
89. Wickham, H., François, R., Henry, L., Müller, K., & Vaughan, D. (u.å.). *Dplyr*. Hentet 5. mai 2024, fra https://dplyr.tidyverse.org/index.html
90. Wickham, H., & Muller, K. (u.å.). *DBI*. Hentet 19. april 2024, fra https://dbi.r-dbi.org/
91. Wickham, H., Ooms, J., & Müller, K. (2023). *Package 'RPostgres'*. https://cran.r-project.org/web/packages/RPostgres/RPostgres.pdf

92. Wikstøm, M., M. (2023, juni 27). *Kort om webapplikasjoner*. https://increo.no/blogg/kort-om-webapplikasjoner
93. Worboys, M., & Duckham, M. (2004a). 5.7.3 Shortest path. I *GIS A Computing Perspective* (2. utgave, s. 214. – 216).
94. Worboys, M., & Duckham, M. (2004b). Structures and access methods chapter 6. I *GIS: A Computing Perspective Second Edition* (2. utgave, s. 221–258). CRC PRESS.
95. Ørstavik, E., & Mæhlum, L. (2023). Geografisk informasjonssystem – GIS. I *Store norske leksikon*. https://snl.no/geografisk_informasjonssystem_-_GIS
96. Aaberge, A. (2021, mars 17). *Brukeropplevelse og brukergrensesnitt*. ndla.no. https://ndla.no/nb/subject:a6b56b7e-2149-4216-92b6-3095feb870f3/topic:ed3a465c-4ced-4c80-9f16-0f9b76d22248/resource:1:160302

Figurliste

Kodeliste