Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for ingeniørvitenskap
Institutt for vareproduksjon og byggteknikk

Lappegård, Embrik
Styve, Kjersti Rønhovde
Amundsen, Lars Ole Hakkim

Implementering av OGC API

Et undersøkende og sammenlignende arbeid tilknyttet tjenesteoppsett etter ny og gammel standard

Bacheloroppgave i ingeniørfag, geomatikk
Veileder: Sverre Stikbakke
Mai 2023

NTNU
Kunnskap for en bedre verden

Lappegård, Embrik
Styve, Kjersti Rønhovde
Amundsen, Lars Ole Hakkim

Implementering av OGC API

Et undersøkende og sammenlignende arbeid tilknyttet tjenesteoppsett etter ny og gammel standard

Bacheloroppgave i ingeniørfag, geomatikk
Veileder: Sverre Stikbakke
Mai 2023

Norges teknisk-naturvitenskapelige universitet
Fakultet for ingeniørvitenskap
Institutt for vareproduksjon og byggteknikk

NTNU
Kunnskap for en bedre verden

NTNU
Kunnskap for en bedre verden

Implementering av OGC API

Et undersøkende og sammenlignende arbeid tilknyttet tjenesteoppsett etter ny og gammel standard

Embrik Lappegård
Kjersti Rønhovde Styve
Lars Ole Hakkim Amundsen

Gradering: Åpen

Bachelor i ingeniørfag - geomatikk
Innlevert: mai 2023
Veileder: Sverre Stikbakke

Norges teknisk-naturvitenskapelige universitet
Institutt for vareproduksjon og byggteknikk

NTNU
Institutt for vareproduksjon
og byggteknikk

Oppgavens tittel: Implementering av OGC API	Dato: 16.05.2023 Antall sider: 54			
	Masteroppgave:		Bacheloroppgave	x
Navn: Embrik Lappegård, Kjersti Rønhovde Styve, Lars Ole Hakkim Amundsen				
Veileder: Sverre Stikbakke				
Eventuelle eksterne faglige kontakter/ veiledere:				

OGC API er en ny familie av standarder fra Open Geospatial Consortium (OGC), lansert som en videreutvikling av WxS-familien. De nye standardene baserer seg på moderne teknologi for datautveksling over internett, og etterligner i større grad hvordan dette blir gjort i IT-verdenen generelt. Oppgaven undersøker hvordan én av disse nye standardene, OGC API Features, vil påvirke en geomatikers arbeidsgang for tjenesteoppsett på etablerte serverplattformer sammenlignet med dennes tilsvar blant WxS-standardene, Web Feature Service (WFS). Det blir også undersøkt hvilke muligheter overgangen til OGC API eventuelt vil gi.

For å undersøke dette har vi delt denne oppgaven i to deler, hvorav en del inneholder en teoretisk gjennomgang av standardene, og den andre en mer praktisk utprøving av tjenesteoppsett i henhold til både ny og gammel standard. For den praktiske undersøkelsen ble det satt opp tjenester på tre ulike plattformer, GeoServer, ArcGIS Server og pygeoapi. Samtlige har støtte for OGC API Features, men kun de to førstnevnte har støtte for WFS. GeoServer og pygeoapi ble satt opp på en ekstern Linux/Ubuntu server. ArcGIS Server ble satt opp lokalt på en Windows-datamaskin.

Arbeidsgangen ved oppsett av tjenester etter ny og gammel standard på ulike plattformer blir presentert, hvor installasjon og tjenesteoppsett blir beskrevet trinnvis. Det blir også framvist skjermdumper av eksempler på kall. Innholdet her, teori og ulike aspekter med programvaren blir videre utforsket og analysert.

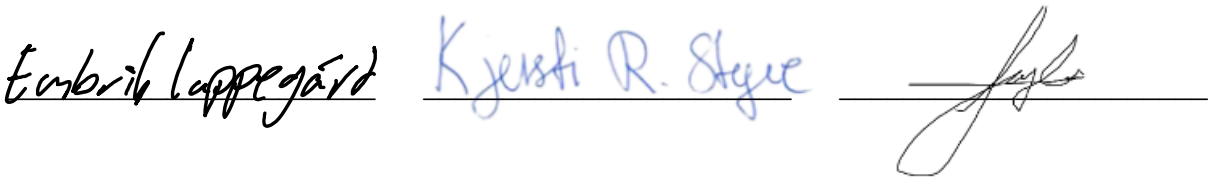
Standardene representer to grunnleggende forskjellige måter å bygge opp tjenester på. Den eldre WFS-standarden bruker sammen med resten av WxS-familien HTTP med SOAP som overføringsmetode for XML-baserte filer, mens OGC API-standardene er basert på nyere REST-arkitektur og er rettet mot datarepresentasjoner, og åpner opp for data på flere formater. Overgangen vil gjøre det mulig å gå bort fra XML-baserte filer til å kunne gi ut de filene som selv er ønsket.

Plattformenes forskjeller ligger i brukervennlighet og muligheter. ArcGIS Server oppleves brukervennlig både på installasjon, tjenesteoppsett og dokumentasjon. Den har derimot begrensede utvidelsesmuligheter, er proprietær og krever lisens for bruk. GeoServer vurderes som brukervennlig etter gjennomført installasjon, har et etablert bruker/-utviklermiljø, og har god dokumentasjon. GeoServer har åpen kildekode, men har også noen utfordringer, spesielt rundt installasjonsprosessen. pygeoapi har også åpen kilde, oppleves krevende å sette opp, har et forholdsvis lite bruker/-utviklermiljø og krever en større grad av teknisk forståelse rundt REST-teknologi for bruk. Til gjengjeld kan man ved beherskelse av pygeoapi sette opp avanserte, framtidsrettede og gode tjenester i henhold til den nye standarden.

Stikkord:

OGC API
Features
Standarder
Web Feature Service
REST
GeoServer
pygeoapi
ArcGIS Server
Tjenesteoppsett

Ål/Oslo 16. mai 2023



Embrik Lappegård Kjersti Rønhovde Styve Lars Ole Hakkim Amundsen

Forord

Vi presenterer herved vår bacheloroppgave, som er resultatet av utallige timer med hardt arbeid og noen få timer fjas. Vår trio av forfattere er en sammensatt gjeng – en 23-åring fra Ål, en 27-årig Vossing og en mann på 30 fra Oslos østkant. Til tross for våre ulikheter i alder, geografisk plassering og bakgrunn, har vi klart å jobbe godt nok sammen til å ferdigstille denne oppgaven. Vi hadde imidlertid ikke klart dette uten stødig veiledning fra vår veileder Sverre Stikbakke – tusen takk! Vi vil også rette en takk til de andre ansatte ved geomatikk-utdanningen på NTNU Gjøvik, spesielt Rune Strand Ødegård, som var en pådriver for denne oppgaven i oppstartsfasen.

By og land – hand i hand!

Abstract

OGC API is a new family of standards from the Open Geospatial Consortium (OGC), released as a development of the existing WxS-standards. These new standards are based on modern technology for data exchange over the internet and more closely resembles how this is done in the IT industry in general. This study examines how one of these new standards, OGC API Features, will affect a geomatics engineers' workflow for service setup on established server platforms compared to its counterpart among the WxS standards, Web Feature Service (WFS). The study also investigates what opportunities the transition to OGC API might provide.

To investigate this, we have divided this study into two parts, one of which contains a theoretical review of the standards, and the other a more practical testing of service setup according to both the new and old standards. For the practical investigation, services were set up on three different platforms: GeoServer, ArcGIS Server, and pygeoapi. All three support OGC API Features, but only the first two support WFS. GeoServer and pygeoapi were installed on an external Linux/Ubuntu server. ArcGIS Server was installed locally on a Windows computer.

The workflow for setting up services according to the new and old standards on different platforms is presented, with installation and service setup described step-by-step. Examples of operations are also presented. The content of which, in addition to theory and various aspects of the software are further explored and analyzed.

The standards represent two fundamentally different ways of building services. The older WFS standard, together with the rest of the WxS family, use HTTP with SOAP as the transmission method for XML-based files, while the OGC API standards are based on newer REST architecture aimed towards data representations, opening for data in multiple formats. The transition will make it possible to move away from XML-based files, enabling data to be delivered in multiple formats.

The platforms differ in usability and capabilities. ArcGIS Server is user-friendly when it comes to both installation and service setup and has good quality documentation. However, it

has limited extension possibilities, is proprietary, and requires a license to be used. GeoServer is considered user-friendly after installation, has an established community, and has good quality documentation. GeoServer is open source but also has some challenges, especially regarding the installation process. pygeoapi is also open source, but it is challenging to set up, has a relatively small community, and requires a greater degree of technical understanding of REST technology. At the same time, using pygeoapi allows for the creation of advanced, future-oriented services in compliance to the new standard.

Innholdsfortegnelse

Figurliste

Forkortelser

API	Application Programming Interface
CSV	Comma Separated Values
FOSS	Free and Open-Source Software
FOSS4G	Free and Open-Source Software for Geospatial
GeoJSON	Geographic JavaScript Object Notation
GIS	Geografiske informasjonssystemer
GML	Geography Markup Language
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
OGC	Open Geospatial Consortium
REST	Representational State Transfer
RPC	Remote Procedure Call
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery and Integration
W3C	World Wide Web Consortium
WFS	Web Feature Service
WMS	Web Map Service
WMTS	Web Map Tile Service
WSDL	Web Service Description Language
XML	Extensible Markup Language
YAML	YAML Ain't Markup Language

1 Innledning

Open Geospatial Consortium (OGC) sine åpne standarder har siden oppstarten på 1990-tallet hatt stor betydning for hvordan geografisk informasjon fremdeles i dag blir delt over internett. De eldre standardene, som gjerne går under betegnelsen WxS-standardene, beskriver ulike tjenestetyper for blant annet visualisering og overføring av geografiske data.

I nyere tid startet OGC arbeidet med å utvikle en ny generasjon standarder: OGC API. Den nye generasjonen av standarder ønsker å modernisere hvordan geografiske data blir delt på nettet, ved å spesifisere REST API-er som følger OpenAPI-spesifikasjonen. Målet med de nye standardene er å kunne tilby moderne API-er som bygger på allerede veletablerte metoder og teknologier for utveksling av data over internett, og bidra til at OGC kan oppfylle sitt mandat om å gjøre geodata FAIR – *Findable*, *Accessible*, *Interoperable* og *Reusable* (OGC, u.å.-a).

Oppgaven vil undersøke følgende problemstilling:

Fra en geomatikers perspektiv, hva blir konsekvensene som følge av innføringen av OGC API med hensyn på tjenesteoppsett og hvilke muligheter fører dette med seg?

Med muligheter menes i denne sammenheng hvorvidt innføringen vil ha noe å si for arbeidsgangen ved tjenesteoppsett, og om det for eksempel må opparbeides ny kompetanse tilknyttet overgangen til OGC API-er. For å skape en tydelig ramme vil vi avgrense undersøkelsene innen tjenesteoppsett til OGC API Features, og dets tilsvar i den gamle familien av standarder, Web Feature Service (WFS).

For å kunne sammenligne standardene vil vi sette opp nedlastningstjenester i henhold til både WFS og OGC API Features på serverplattformer med åpen og proprietær kildekode, henholdsvis GeoServer og pygeoapi, og ArcGIS Server. Oppsett av tjenestene vil primært bli vurdert ut ifra en helhetlig vurdering av arbeidsgangen tilknyttet installasjonsprosessen og tjenesteoppsett, samt andre aspekter slik som dokumentasjon og bruker-/utviklermiljø tilknyttet plattformen. Sammenligningen av OGC API Features og WFS vil bli gjort basert på de teoretiske beskrivelsene av de ulike standardene, samt ut ifra erfaringen som ble gjort undervegs i oppsettingen av tjenestene.

I teorikapittelet vil det bli presentert et teoretisk grunnlag for oppgaven. Dette grunnlaget vil hovedsakelig bestå av en presentasjon av standardene fra OGC og de ulike

serverplattformene, i tillegg til mer generell begrepsavklaring for standarder, webtjenester og API-er. Videre vil metodekapittelet inneholde en beskrivelse av datagrunnlaget som har blitt brukt i testing av tjenester, samt hvordan tjenesteoppsett på plattformene skal sammenlignes, før selve oppsettet av tjenestene på de ulike plattformene blir presentert i resultatkapittelet og tilhørende vedlegg. Avslutningsvis vil diskusjonsdelen vurdere de ulike plattformene opp mot hverandre, i tillegg vil det bli diskutert hvordan Features skiller seg fra WFS på et mer teknisk nivå.

Verdenssamfunnet står i dag overfor store klima- og miljøutfordringer. FNs bærekraftsmål presenterer en helhetlig oversikt over ulike områder som det må arbeides med, samt delmål relatert til bærekraft sett fra både et økonomisk, samfunnsmessig og miljømessig perspektiv. Satt i en større sammenheng kan vår undersøkelse av de nye standardene fra OGC være et bidrag til oppnåelse av bærekraftsmål nummer 9, «Industri, innovasjon og utvikling» (FN-sambandet, 2023). Dette målet retter søkelyset på bærekraftig og robust infrastruktur, utbygging og industrialisering, og oppmuntrer til innovasjon. Gjennom bruk av OGC API-er kan man forenkle tilgangen til geografiske data, som igjen kan fungere som beslutningsstøtte for bærekraftig utbygging og ressursutnyttelse. Samfunnsmessig kan dette legge til rette for bedre planlegging og forvaltning av ressurser, for eksempel innen landbruk, energiproduksjon og infrastrukturutbygging, ved at data tilgjengeliggjøres for flere.

2 Teori

2.1 Geografiske data

I Nasjonal geodatastrategi blir geografisk informasjon definert som «[…] informasjon om objekter (vann, hus, veger, fyr, osv.), hendelser og forhold der posisjonen (sted på jorda) er en vesentlig del av informasjonen.» (Kommunal- og moderniseringsdepartementet, 2018, s.28). Videre blir det satt likhetstegn mellom begrepene geografiske data (geodata) og stedfestet informasjon (stedsdata).

I engelsk litteratur som omhandler geografiske data og GIS, finner man gjerne begrepet *Geospatial* – som de Smith, Goodchild og Longley (2018) mener er et mer presist begrep sammenlignet med geografisk – da det ikke nødvendigvis trenger å være en grafisk fremstilling tilknyttet dataene. Uavhengig av valg av begrep, er det en egenskap ved geodata som gjør det mulig å skille de fra andre datatyper; de er bygd opp av objekter eller fenomener som er stedfestet og har en plassering som refererer til et geografisk referansesystem (Kraak og Ormeling, 2020; Yeung og Hall, 2007).

Geodata blir hovedsakelig presentert som vektor eller raster. Vektorformatet beskriver geodata i form av geografiske objekter, som er «abstrakt representasjon av et virkelig fenomen knyttet til et bestemt sted eller geografisk område» (Geodataforskriften, 2012, §3), presentert som punkter, linjer eller polygoner (Yeung og Hall, 2007). Begrepet objekt samsvarer med det engelske *features* – som i denne oppgaven blir presentert i forbindelsene med standardene for Web Feature Service og OGC API – Features.

Brunsdon og Comber (2021) argumenter for en økende andel av digital data kan kalles romlige, fordi dataene har blitt samlet inn fra en gitt plassering. Dette synet støttes også av Kotsev *et al.* (2020), som mener at stedfestet data ikke lenger kan sees på som noe spesielt og særegent. Dette blir blant annet begrunnet med at det har vært et skifte fra at geodata hovedsakelig ble administrert og innsamlet av det offentlige, mens innsamlingen i dag også skjer i det private; for eksempel av bedrifter og av den generelle befolkningen. I tillegg blir det i dag produsert store mengder geodata, godt hjulpet av for eksempel satellittbaserte overvåkningsprogram, slik som Copernicus og teknologi tilknyttet Tingenes internett (IoT). Geodata kan derfor sies å ha blitt mindre spesielt, fordi kunnskapen om det ikke lenger er

begrenset til visse kretser og fagområder, samtidig som en stadig økende andel av dataene som produseres nettopp kan kalles geodata. Kotsev *et al.* (2020) mener derfor at man i større grad enn i dag bør se til de metodene og den teknologien som benyttes for lagring og distribusjon av data ellers i IT-verdenen, og ta i bruk disse for geodata også.

2.2 Om standardisering

Standarder kan defineres som veiledninger på hvordan produkter, systemer og tjenester kan brukes og utvikles på best mulig måte (Yeung og Hall, 2007; Standard Norge, 2021). Standarder finnes overalt i ulike sektorer og fagområder i samfunnet, og det arbeides kontinuerlig med å utvikle nye eller revidere eksisterende standarder som respons på nye eller endrede behov i markedet. En sentral organisasjon innenfor standardisering, er den internasjonale standardiseringsorganisasjonen ISO, som utvikler standarder innen teknologi, ledelse og produksjon (ISO, u.å.). Andre internasjonale organisasjoner er gjerne mer rettet mot et bestemt fagfelt, slik som W3C, som retter seg mot utvikling av standarder og protokoller for internett (W3C, u.å.-b), som kommunikasjonsprotokollen HTTP.

Selv om standarder er ment å skulle ha en viss varighet, vil det som nevnt være behov for å revidere dem – både for å holde tritt med endrede brukerbehov, men også den teknologiske utviklingen (Kralidis, 2008). På denne måten gjenstår det fremdeles et viktig arbeid etter publisering av standarder, i form av revisjon og revidering (Yeung og Hall, 2007).

2.2.1 Åpne og proprietære standarder og programvarer

Åpenhet er et begrep som favner bredt, men i begreper som åpne data, programvarer og kildekoder referer ‘åpen’ til noe som er fritt tilgjengelig, og der brukere har tillatelse til å bruke, endre og dele det slik de ønsker (Open Knowledge Foundation, 2023).

Standarder kan klassifiseres som åpne eller proprietære ut ifra hvordan de forvaltes og hvordan de har blitt utviklet (Yeung og Hall, 2007). Krechmer (2005) definerte i alt 10 ulike krav til en åpen standard, sett fra både et produsent-, implementerings- og brukerperspektiv. Relatert til utvikling og forvaltning, vil en oppsummering av disse 10 punktene definere en åpen standard som en standard hvor utvikling og revidering er åpne og konsensusbaserte prosesser, samt at både standarden og tilhørende dokumenter er tilgjengelige og kan brukes av alle. En alternativ måte å forstå åpne standarder på, er som en motsetning av proprietære

standarder – altså en lukket standard som ikke er fritt tilgjengelig for allmenheten, og hvor rettighetene og kunnskapene for å utvikle et bestemt produkt eller tjeneste i utgangspunktet forblir hos utviklerne utviklerne (Bårdgård, 2022; Yeung og Hall, 2007). Selskapet som har utviklet og eier standarden kan dermed kontrollere utviklingen, implementeringen og lisensieringen av den.

For programvarer som har åpen kildekode (*open-source*), er kildekoden tilgjengeliggjort for brukerne slik at de blant annet har mulighet til å studere den, rette eventuelle feil og legge til egen kode for å tilføre ny funksjonalitet til programvaren (Gramstad, 2021). Selv om hensikten med åpen kildekode er at endringen skal tilbakeføres til felleskapet slik at de blir åpent tilgjengelig for alle brukere, kan enkelte lisensieringer av åpne kildekode-programvarer også tillate publikasjon av egne, proprietære versjoner basert på de endringene eller forbedringene man selv har gjort.

Fri programvare (*Free software*) er nært relatert til åpen kildekode, men der åpen kildekode i større grad handler om de praktiske fordelene knyttet til tilgjengeliggjøring av koden, handler fri programvare mer om de prinsipielle idéene tilknyttet frihet – at en bruker skal ha frihet til å gjøre det de vil med koden, alt fra å kjøre, endre og videredistribuere den (Nätt 2023; Stallman, 2022). Et begrep som dekker begge konseptene, er FOSS – *Free and Open Source Software* (Moreno-Sanchez, 2012). For programvarer som håndterer geografiske data, benyttes alternativt FOSS4G (*Free and Open Source Software for Geospatial*).

Å distribuere programvare som åpen kildekode kan fungere som en kvalitetssikring, dersom tilstrekkelig antall personer bidrar med forbedringer og feilrettinger til programvaren (Nätt, 2023). Et sentralt og viktig aspekt tilknyttet FOSS/FOSS4G, er derfor det nettverket eller fellesskapet av personer som bidrar med kunnskap og kode til utvikling av programvaren (Coetzee *et al.*, 2020). Disse fellesskapene, som gjerne består av både profesjonelle utviklere og andre brukere av programvaren, er avgjørende for at et FOSS/FOSS4G-prosjekt skal få gjennomslag i markedet, og holde seg bærekraftig over tid. Videre trekker Coetzee *et al*. (2020) frem at et godt fellesskap kjennetegnes ved høy grad av aktivitet, der bidragsyterne samarbeider om oppgaver slik som uttesting og å lage dokumentasjon og opplæringsmateriell. Disse felleskapene går gjerne under betegnelse bruker-/utviklermiljø.

2.3 Open Geospatial Consortium (OGC)

Open Geospatial Consortium (OGC) er en sammenslutning av ulike bedrifter, offentlige etater, universiteter og andre interessenter, som utvikler åpne standarder spesielt tilpasset geografiske data ved å bygge videre på veletablerte standarder utgitt av organisasjoner som ISO og W3C (Kralidis, 2008). Overordnet sett beskriver disse standardene metoder for hvordan geografiske data kan overføres via internett, eller hvilket format de kan overføres på (Bocher og Neteler, 2012).

Siden etableringen på 1990-tallet, har OGC gitt ut en rekke standarder som fortsatt har stor betydning for hvordan geografisk data i dag blir delt, tilgjengeliggjort og fremstilt (Blanc *et al.*, 2022). OGC har blant annet utviklet standarder for oppsett av webtjenester spesielt tilpasset geografiske data – en samling som gjerne kalles WxS-standardene. Disse standardene beskriver mange ulike tjenestetyper med ulik funksjonalitet. Når det kommer til overføring, tilgjengeliggjøring og visning av geografiske data, er det primært tjenestetypene WFS (Web Feature Service), WMS (Web Map Service) og WMTS (Web Map Tile Service) som har gjort seg gjeldene.

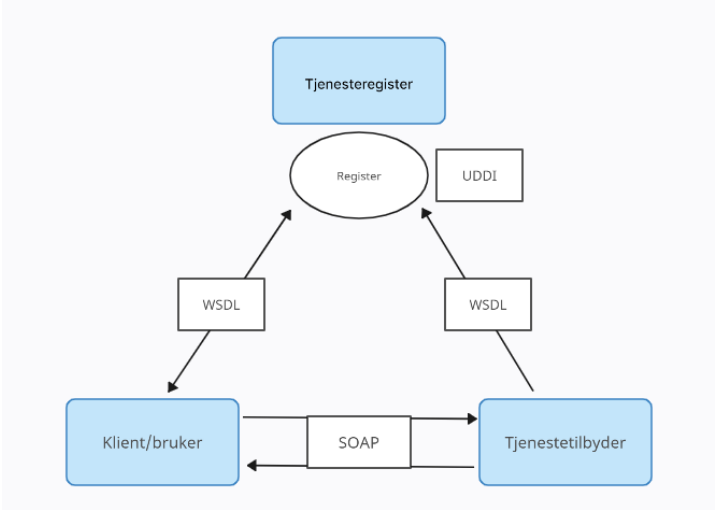
2.4 Webtjenester

Webtjenester kan defineres på ulike måter. Yeung og Hall (2007) definerer det som del av en programvare som er tilgjengelig over internett via standardiserte protokoller, mens Snell, Tidwell og Kulchenko (2002) definerer det som et XML-baserte grensesnitt som gir tilgang til noe av funksjonaliteten til en applikasjon. Essensen i webtjenester – uavhengig av hvilken definisjon som brukes – er at de muliggjør kommunikasjon mellom programvarer ved bruk av åpne internettstandarder slik som HTTP, SOAP og XML (Sahin og Gumusay, 2008; Davis Jr. og Lacerda Alves, 2008). Det at webtjenester baserer seg på åpne standarder, gjør at programvarer kan bruke de til å interagere med hverandre, uavhengig av faktorer som programmeringsspråk og operativsystem (Mitchell, 2005).

Webtjenester er gjerne assosiert med begrepet tjenesteorientert arkitektur (*service oriented architectur*) (Chappell og Jewell, 2002). Dette kommer av at webtjenester både kan betraktes som en implementasjon av tjenesteorientert arkitektur (IBM, 2022), alternativt at tjenesteorientert arkitektur er en arkitekturstil for webtjenester – altså en beskrivelse av hvordan webtjenesten kan designes på best mulig måte (Subramanian og Raj, 2019).

Den tjenesteorienterte arkitekturen beskriver hvordan en fullstendig programvare kan bygges opp ved å kople sammen uavhengige, nettverkstilgjengelige tjenester med ulike funksjoner (Sahin og Gumusay, 2008; W3C, 2004). En tjeneste kan i denne sammenhengen forståes som en selvstendig kodeblokk som utfører én eller flere aktiviteter (OGC, 2016). For at en tjeneste skal kunne brukes må den være realisert av en tjenestetilbyder (*service provider*) (W3C, 2004). I tillegg til tilbyderen, er arkitekturen også avhengig av et tjenesteregister (*service broker*) og en bruker (*service requester*) (Chappell og Jewell, 2002). Disse samhandler med hverandre ved hjelp av grunnleggende operasjoner som publisering, oppdaging og tilkobling. Typisk vil interaksjonen mellom komponentene være at en tilbyder definerer og sender en beskrivelse av sine tjenester til registeret. En bruker kan dermed lete gjennom registeret etter en tjeneste som dekker sitt behov, for deretter å kunne koble seg opp mot tjenesten – altså sende en forespørsel til tilbyderen ved hjelp av beskrivelsen gitt av registeret (Snell, Tidwell og Kulchenko, 2002; Chappell og Jewell, 2002). Oppsummert vil altså en tjeneste bli tilbudt av en tilbyder, og brukt av en bruker, mens et register vil hjelpe dem med å finne hverandre.

Som tidligere referert til, er oppbyggingen av webtjenester basert på et utvalg teknologier eller protokoller som bygger lagvis på hverandre i en webtjeneste-arkitektur (W3C, 2004). Det finnes ulike måter å illustrere oppbyggingen på, men i Sahin og Gumusay (2008) sin modell, er det inkludert 4 lag – oppdagelse, beskrivelse, meldingsutveksling og transport – hvor hvert lag er relatert til en åpen internettstandard/-protokoll, henholdsvis UDDI, WSDL, SOAP, og HTTP. Figur 1 viser hvordan de ulike protokollene, foruten HTTP, er koplet sammen med de overnevnte komponentene fra den tjenesteorienterte arkitekturen, som webtjenestearkitekturen også bygger på.



Figur 1. Sammenhengen mellom komponentene i en tjenesteorientert arkitektur og internettprotokollene WSDL, SOAP og UDDI. Inspirert av figur i Subramanian og Raj (2019).

Kort forklart vil UDDI blir brukt som register over tilgjengelige tjenester, som både konsumenten og tilbyderen kontakter for å henholdsvis finne og tilgjengeliggjøre en tjeneste. WSDL beskriver tjenestene i registeret, slik som hvilke type meldinger den aksepterer, mens SOAP angir formatet som blir brukt til å overføre meldinger når konsumenten kontakter tilbyderen over HTTP (Sahin og Gumsuay, 2008; Subramanian og Raj, 2019).

En sentral teknologi i webtjeneste-arkitekturen, er XML (W3C, 2004). XML er kort forklart et menneske- og maskinlesbart format, som ble utviklet for å lagre og transportere data (W3Schools, u.å.). XML fungerer som selve grunnlaget for webtjenester, da både blir brukt til å beskrive, lagre og overføre data, i tillegg til at de sentrale protokollene som brukes, er basert på XML (Newcomer, 2002).

2.4.1 Web Feature Services (WFS)

Web Feature Service (WFS) er en type webtjeneste spesifisert i standarden WFS fra OGC (OGC, 2014). WFS-tjenesten er basert på en tjenesteorientert arkitektur, som var gjeldene på den tiden disse standardene ble lansert (OGC, 2017). Tjenesten tilbyr et grensesnitt for overføring av geografiske data på vektorformat med tilhørende geometri og egenskapsdata (Norge Digitalt, 2019). Standardformatet for dataene som skal overføres er GML; et XML-basert format definert av OGC spesielt tilpasset geografiske objekter (OGC, 2014).

I tillegg til å tilby ren dataoverføring, vil en WFS-tjeneste også kunne tilby ulike former for manipulasjon av dataene (OGC, 2014). Versjonen WFS 2.0 har i alt støtte for 11 ulike operasjoner som en bruker kan kalle på. 3 av disse – GetCapabilities, DescribeFeatureType og GetFeature – kan anses som grunnleggende operasjoner, hvor GetCapabilities returnerer metadata om serveren, DescribeFetureType beskriver objekttyper i datasettet og GetFeature returnerer et utvalg objekter (OGC, u.å.-e). Typisk vil kommunikasjon mot en WFS-tjeneste utføres med kall i denne rekkefølgen (Norge digitalt, 2019). I tillegg til å beskrive generelle metadata, vil GetCapabilities også inneholde blant annet en oversikt over tilgjengelige operasjoner og hvilke objekttyper tjenesten tilbyr.

WFS, i likhet med de resten av standardene i WxS-familien, er basert på konseptet med RPC, nærmere bestemt RPC over HTTP (OGC, 2023). RPC (*Remote Procedure Calls*) beskriver en generell kommunikasjonsmetode hvor en bruker sender et kall til en server og får utført en spesifikk operasjon (funksjon/metode) (Jin, Shevat og Sahni, 2018). RPC gjør det derfor

mulig for programvarer som kjører på forskjellige systemer å samhandle med hverandre over nettet (Snell, Tidwell og Kulchenko, 2002).

SOAP er en XML-basert protokoll for utveksling av strukturert informasjon mellom programvarer over internett (Dvergsdal og Mæhlum, 2021), som kan brukes til å utføre RPC-basert kommunikasjon (Snell, 2002). SOAP er ikke tilknyttet en spesiell transportprotokoll, likevel er HTTP mest brukt (Cerami, 2002). SOAP, sammen med HTTP GET og HTTP POST, utgjør til sammen *service bindings*-ene som *WFS*-standarden tillater – altså tillatte kommunikasjonsprotokoller for overføring av meldinger mellom WFS-tjeneren og brukeren (OGC, 2014). SOAP-protokollen gjør det mulig å definere hvilken type data som skal overføres, hvordan det skal uttrykkes i XML, og hvordan informasjonen skal overføres (Snell, Tidwell og Kulchenko, 2002).

2.5 API

API (*Application Programming Interface*), eller programmeringsgrensesnitt, kan noe forenklet beskrives som grensesnitt som brukes av programvarer for å kommunisere med hverandre ved hjelp av et felles språk som begge forstår (Patni, 2017; Jacobson, Brail og Woods, 2011). API er ikke et nytt fenomen, men forbindes i dag gjerne i dag med det som kalles for Web API-er, som er API-er som er bygd opp og basert på teknologier knyttet til internett (OGC, 2023).

Jin, Shevat og Sahni (2018) viser til at det over årenes løp har eksistert ulike paradigmer for API-design og oppbygging. I dag er det stor grad et utvalg etablerte standarder som brukes for oppbygging av API-er, som for eksempel REST, SOAP og RPC (Nätt og Rossen, 2022). Blant disse blir REST ansett som en av de mest populære, som en arvtaker til SOAP, som tidligere var gjeldende (Chen *et al.*, 2017).

2.5.1 REST-API

REST (*Representational State Transfer*) er en arkitekturstil for oppbyggingen for distribuerte systemer, hvor selve oppbyggingen skal følge et sett med designkriterier: *Client-server*, *uniform interface*, *layered system*, *cache*, *stateless* og *code-on-demand* (Masse, 2011; Subramanian og Raj, 2019). Uten å gå nærmere innpå forklaringen av hver av disse kriteriene, er målet med REST å skape en arkitektur som etterligner måten internett er bygd opp på, og følgelig verdsette egenskaper slik som skalerbarhet, enkelhet, synlighet og ytelse.

REST er sentrert rundt konseptet ressurser, som kan defineres som en fysisk komponent som gjerne kan lagres på en datamaskin, og som har en unik adressering tilknyttet seg, slik som URL (Uniform Resource Locator) (Richardson og Amundsen, 2013). Hver ressurs kan representeres på ulike måter, noe som gir mulighet for brukeren å etterspørre spesifikke representasjoner av en ressurs – og på den måten manipulere en ressurs gjennom representasjoner (Massé, 2011). Generelt sett er en representasjon av en ressurs et dokument som beskriver tilstanden (*state*) til ressursen på et maskinlesbart format – slik som et XML-dokument eller et JSON-objekt (Richardson og Amundsen, 2013). Det er altså representasjonen som blir overført mellom klient og tjener, og ikke ressursen i seg selv.

Det grunnleggende prinsippet til REST, er at komponentene i det distribuerte systemet skal kommunisere ved hjelp av teknologiene og protokollene til internett (som HTTP og URI), og på den måten tilby en måte å bygge opp webtjenester og API-er som ikke er avhengig av tilleggsprotokoller slik som SOAP og WDSL (Subramanian og Raj, 2019; Richardson og Ruby, 2007). Et web-API som følger REST-arkitekturstilen, blir kalt et REST API (Masse, 2011) og vil være en samling av ulike ressurser som er knyttet til hverandre.

REST API-er eksponerer altså data som ressurser, og bruker HTTP-metodene til å utføre grunnleggende operasjoner på dem (Jin, Shevat og Sahni, 2018). HTTP-protkolloen har i alt 8 ulike metoder for hvordan klienten og serveren kan kommunisere med hverandre, hvorav de mest brukte til å manipulere ressurser er: GET, DELETE, POST, PUT (Richardson og Amundsen, 2013). Disse metodene referer henholdsvis til henting eller lesing av ressurs, sletting, oppretting av ny ressurs og å oppdatere eksisterende ressurs.

Selv om REST API-er kan returnere responsen på flere ulike formater, er bruken av JSON utbredt i moderne REST API-er (Jin, Shevat og Sahni, 2018; Lange, 2016). Sammenlignet med XML, er JSON betraktet som et lettere format – både med tanke på lesbarhet, men også

fordi det har en lettere struktur, som gjør at JSON er lettere å prosessere og transportere over HTTP (Patni, 2017; Jacobson, Brail og Woods, 2011)

I sin sammenligning av SOAP-baserte og REST API-er trekker Patni (2017) frem fordeler og ulemper ved begge. Noen av fordelene med REST, er at de kan levere data i flere formater, og at API-et er lett å implementere og vedlikeholde. SOAP bruker derimot mer båndbredde ved dataoverføring og blir ansett som relativt tungvint å implementere. På en annen side har SOAP fordelene med at det kan fungere over hvilken som helst kommunikasjonsprotokoll, og at det har innebygd sikkerhet og autorisering som del av protokollen.

2.5.2 API-er og webtjenester

API-er og webtjenester er to begreper som det er vanskelig å finne entydige definisjoner på, da ulike kilder definerer dem ulikt avhengig av kontekst og formål. Selv om begge konseptene har det samme hovedformålet – å muliggjøre kommunikasjon mellom programvarer i et nettverk – har hver av dem likevel noen særegne aspekter ved seg, som tilsier at begrepene ikke bør brukes fritt om hverandre.

Sammenhengen de imellom kan derfor i noen tilfeller være uklar, men en vanlig oppfatning er at webtjenester som konsept i større grad hører fortiden til, blant annet grunnet forbindelsen mot SOAP-basert arkitektur (Hygraph, 2022). En årsak til dette er at SOAP-baserte tjenester med sin tilknytning til XML ofte blir ansett som et tungt overføringsformat, spesielt sammenlignet med tjenester bygget på REST og JSON (W3C, u.å.-a) som overfører informasjon raskere og krever mindre nettverkskapasitet (Osman, 2022). Noe forenklet kan man si at alle webtjenester er en form for API, men ikke alle API-er er webtjenester. Hva som kan kalles for hva vil blant annet være bestemt av arkitektur, overføringsformat og transportprotokoll (Juviler, 2022).

2.6 OGC API-standardene

OGC API er en ny generasjon standarder fra OGC, som har som mål å gjøre det enklere å implementere romlige data i moderne systemer og applikasjoner (Kotsev *et al.*, 2020). Disse nye standardene er basert på OpenAPI-spesifikasjonen (OAS), som er en åpen standard for å beskrive REST API-er på et språk- og plattformuavhengig format (Mainas, Petrakis og Sotiriadis, 2017).

API-er som er beskrevet i henhold til OAS-standarden har dokumentasjon på et format som både er menneske- og maskinlesbart (OpenAPI Initiative, 2021), noe som blant annet vil gjøre det lettere for utviklere og brukere å forstå hva tjenesten tilbyr, samt hvordan man skal interagere med den. Det at API-et er dokumentert på et maskinlesbart format, åpner også for at programvarer automatisk kan tolke noe av funksjonaliteten til API-et, noe som både kan minimere menneskelig feil, samt effektivisere arbeidet med å med å utvikle programvare som skal kommunisere med API-et (OpenAPI Initiative, 2023) Sekundært vil bruken av OAS også gjøre API-tjenestene mer tilgjengelige, fordi de blir lettere å oppdage via tradisjonelle søkemotorer slik som Google og Bing (Kotsev *et al.*, 2020).

OAS blir i økende grad brukt til å beskrive REST API (Karlsson, Čaušević og Sundmark, 2020), noe som trekker i riktig retning når det gjelder OpenAPI sitt mål om å standardisere hvordan REST API-er blir beskrevet. På nåværende tidspunkt, har de nye standardene fra OGC også kun støtte for å bruke Open API sin spesifikasjon (OGC, 2023)

Standarden «OGC API – Common (OGC,2023) beskriver hvordan de nye standardene skal implementeres som et Web API, blant annet ved å spesifisere regler for bruken av HTTP-protokollen og URI, som igjen skal sikre at konsistent bruk av HTTP-protokollen, og hvordan URI-ene skal lages, og hvordan URI-spørringene (*queries*) skal brukes.

2.6.1 OGC API – Features

Inndelingen av standardene i den nye samlingen av OGC-standarder, er basert på ressurstype (OGC, u.å.-b). Standarden OGC API – Features (heretter kalt Features) spesifiserer derfor oppbyggingen for API-er som deler geografiske vektordata i form av objekter.

Features-standarden er igjen splittet opp i flere deler som skal publiseres trinnvis (OGC, u.å.-b). Per mars 2023, er det kun de to første delene som har blitt publiserte – del 1 «Core» og del 2 – «Coordinate Reference System by Reference». Del 2 gir mulighet for å bruke flere koordinatsystem enn bare WGS84, som er det eneste gjeldende koordinatsystemet i del 1 (OGC 2023). Del 3–5 er også under utvikling, med henholdsvis navnene «Filtering and the Common Query Language (CQL)», «Create, Replace, Update and Delete» og «OpenAPI 3.1» (OGC, u.å.-d). De kommende delene vil tilføre ekstra funksjonalitet, slik som flere og mer avanserte former for filtrering.

En Features-tjeneste er tenkt å levere data i primært fire forskjellige formater; HTML, GeoJSON/JSON, og to ulike versjoner av GML (OGC, 2022). Ingen av disse er direkte påkrevd gjennom Features-standarden, men det er anbefalt at enhver Features-tjeneste bør levere data i HTML og GeoJSON. Anbefalingen om HTML kommer i hovedsak av de utfordringene GeoJSON og GML har med å være søkemotoroptimalisert, som de to formatene i liten grad er. Med GeoJSON og GML kan det i tillegg det være krevende å vise data direkte i en nettleser uten tilleggsprogramvare, som er enklere med HTML ved at for eksempel attributtabeller kan representeres i tabellform rett i nettleseren. I prinsippet er det dermed ikke bare data som kan representeres i de nevnte formatene, selve tjenesten kan også beskrives i for eksempel JSON, hvor de ulike tjenestedelene kan vises som objekter i format-notasjonen.

En vesentlig del av en Features-tjeneste er tjenestens forside, i utviklerterminologien kalt landingsside (*landing page*) (OGC, 2022). Denne har som formål å gi en forenklet oversikt over tjenestens innhold, metadata og funksjonalitet, og er tjenestens utgangspunkt, med utforming og innhold bestemt gjennom OGC API Commons (OGC, 2023). På denne siden finnes det i hovedsak lenker til tre vesentlige bestanddeler av tjenesten; konformitetsklasser (*Conformance Classes*), API-definisjon og objektsamling (*Feature Collections*) (OGC, 2022).

I konformitetsklasse-oversikten finner man hvilke bestemmelser i *Conformance*-delkapitlene i OGC API Commons og -Features som er fulgt, med URI-er til de aktuelle konformitetsklassene i standardene (OGC, 2022). API-definisjonene har i denne sammenheng formål om å gi en oversikt over tjenestens egenskaper og bruksområder. Her finnes beskrivelser om tjenestens oppbygging, metadata og innhold, som er kompatible med API-struktur slik den er definert i OAS-standarden (Swagger, 2022). I objektsamlingen finner man informasjon om selve objektene som finnes i datasettene tjenesten leverer, og det er også her mulighetene for å laste ned datasett på et tilgjengelig representasjonsformat gjerne ligger (OGC 2022).

2.7 Serverplattformer

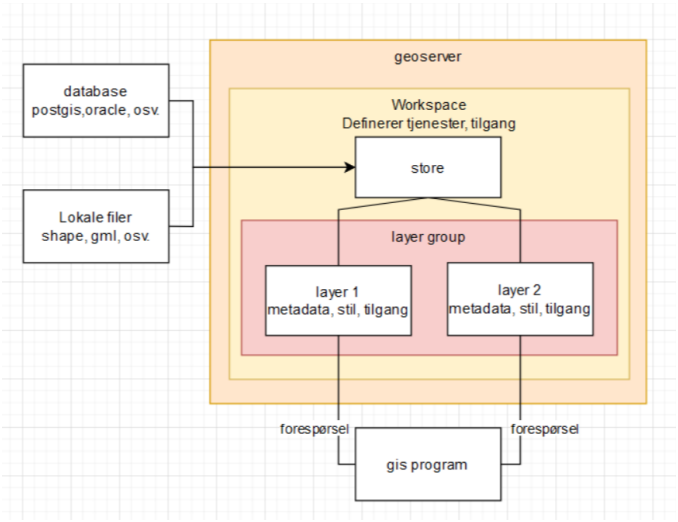
Begrepet plattform kan ha flere betydninger, men i denne oppgaven brukes plattform om applikasjoner som gjør det mulig å sette opp servere som kan fungere som tjener for overføring eller representasjon av geodata. En plattform vil i denne sammenheng tillate

oppsett av tjenester bygget på standarder slik som WFS og Features, og kan forståes som en programvare som installeres på en server for å tilby et grensesnitt for oppbygging og bruk av slike tjenester. Plattformene kan være bygget opp etter både proprietære- og FOSS-prinsipper, og i de påfølgende avsnittene vil plattformene benyttet i denne oppgaven bli presentert.

2.7.1 GeoServer

GeoServer er en åpen kildekode-programvare for å levere kart og geografisk informasjon på internett, bygget på Java. GeoServer har røtter tilbake til 2001 og det offentlige amerikanske nonprofit-prosjektet TOPP (TheOpenPlanningProject), som hadde som formål om å gjøre informasjon om byråkratiske byplanleggingsprosesser mer transparent ved tilgjengeliggjøring over internett (GeoServer, 2020). Plattformen har fra starten av vært basert på åpne standarder fra OGC, og er bygd opp rundt et nettleserbasert grensesnitt for kommunikasjon mot backend-delen av tjenesten, som muliggjør oppsett av tjenester rett i nettleseren etter installasjon (GeoServer, u.å.-b). Med GeoServer har man gode muligheter for integrasjon mot andre åpen kildekode-prosjekter, støtte mot alle WxS-standarden, og har gjennom en nedlastbar tilleggsutvidelse mulighet for oppsett av OGC API-er (foreløpig kun del 1 for Features). Utvidelsen er laget av bruker-/utviklermiljøet tilknyttet GeoServer, som i nåværende versjon (2.22.x) ikke er en offisiell del av programvaren, men er derimot ventet å bli en del av den offisielle dokumentasjonen i framtidige versjoner (GeoServer, u.å.-a).

Enkelt forklart er GeoServer en plattform som lytter etter forespørsler fra klienter, leverer geografiske data lagvis eller samlet når forespørsel er gjort, og kan slik sett brukes av alle som vet hvordan forespørsler etter OGC-standardene fungerer (Miller, 2020). Ved bruk og oppsett av tjenester i GeoServer må man opprette en *workspace.* Det er i en workspace man kan definere hva slags tjeneste som skal settes opp, og hvilke datatyper og datakilder som inngår. Datakilder håndteres gjennom *stores*, hvor man blant annet kan opprette en kobling mellom en workspace og en PostGIS-database eller andre datakilder (Iacovelli og Youngblood, 2013). Figur 2 illustrerer skjematisk hvordan for eksempel en GIS-programvare kan hente data fra en database via tjenester satt opp med GeoServer.



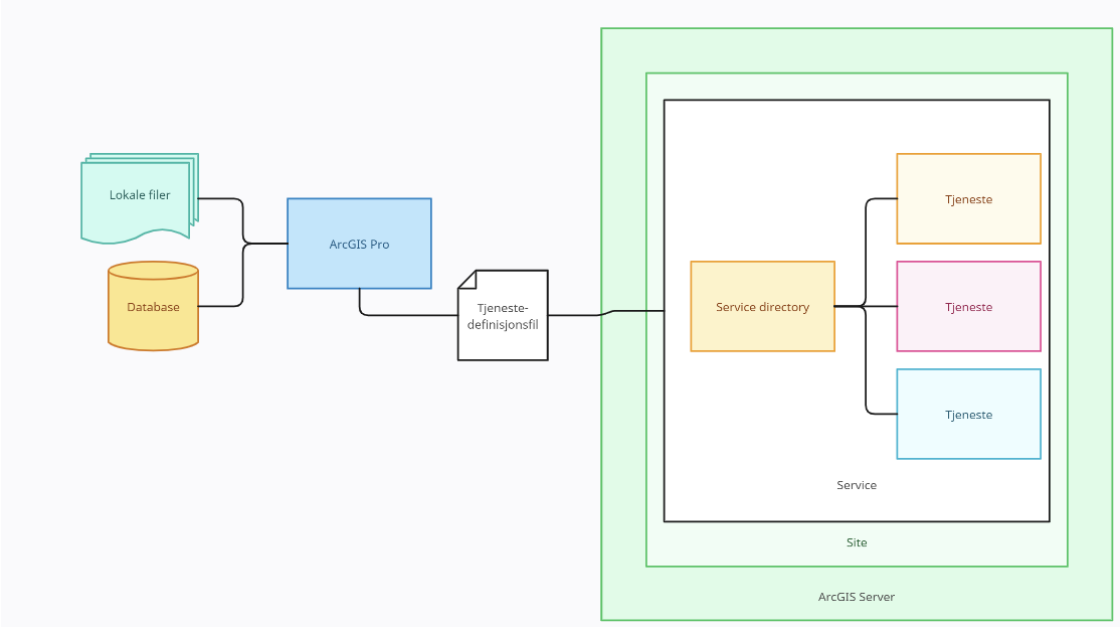
Figur 2: Skjematisk presentasjon av GeoServers komponenter og virkemåte. Egen illustrasjon

2.7.2 ArcGIS Server

ArcGIS Server er et proprietært alternativ til distribusjon og administrasjon av geografiske data over internett, utviklet av Esri (Esri, u.å.-b). Programvaren er primært en del av ArcGIS Enterprise, en portal for samhandling mellom de ulike komponentene i Esris programvarepakke, men kan også brukes separat (Geodata, u.å.). I tillegg til publiseringsfunksjonalitet har det også en del anvendelser innen analyse og prosessering, ved integrering opp mot andre Esri-produkter (Esri u.å.-b). Plattformen støtter alle WxS-standarder fra OGC, og har fra versjon 11 (utgitt juli 2022) også støtte for å publisere tjenester i henhold til Features (del 1), som foreløpig eneste av OGC API-standardene (Esri 2022b). ArcGIS Server er tilgjengelig for Windows og Linux/Ubuntu. Ved installasjon av programvaren vil man, uavhengig av operativsystem, følge en klassisk trinnvis installasjonsveiviser gjennom et grafisk brukergrensesnitt (Esri, 2022a).

Når programvaren er installert på en lokal maskin eller ekstern server, skjer videre tjenesteoppsett gjennom et nettleserbasert grensesnitt, *Server Manager*. Her har man muligheten til å sette opp og definere *sites* og *services*, to sentrale begreper i programvaren (Esri, 2017). En *site* er i denne sammenheng en beskrivelse av én eller flere servere hvor tjenestene og geografisk data er fysisk plassert, og kan beskrives som selve grensesnittet der tjenesteoppsettet gjøres. Dette er dermed det første som må opprettes inne i programvaren etter fullført installasjon, og må gjøres før man kan sette opp tjenester. En *service* er selve tjenesten, og opprettes i Service Manager på en *site*, og er i brukergrensesnittet synlig som ulike mapper med en hierarkisk struktur. Man kan opprette flere *service*-mapper på samme *site*, og med dette ha flere ulike tjenester i samme service-mappe (Esri, 2022c). Man har muligheter til å benytte data som ligger lokalt på server/datamaskin, oppkobling mot sky, og

databasetilkobling, blant annet mot PostGIS (Esri u.å.-a). Når en *site* og *service* er klargjort, kan man sette opp tjenesten ved å opprette et ArcGIS Pro-prosjekt med de ønskede dataene lagt inn som lag, og tilkoblinger mot datasett definert i prosjektet, for eksempel lokale filer på serveren eller en database. Prosjektet må da eksporteres som en tjenestedefinisjonsfil (*Service definition-file/.sd-file*), hvor det ved eksport kan hukes av bokser for hvilken type tjenester som ønskes og definisjonsfilen da vil omfatte. Eksporten forholder seg til enkelte begrensninger på datatyper, som for eksempel at et rasterlag ikke kan eksporteres som WFS. Ved eksport vil ArcGIS Pro kontrollere at datasettet er kompatibelt med tjenestene som er huket av. Dersom datasettet har mangler vil dettet bli listet opp i en feillogg, og man vil ikke få eksportert før dette er utbedret (Esri, u.å.-c). Etter eksport kan tjenestedefinisjonsfilen lastes opp i den ønskede Service-mappen i Server Manager, programvaren vil da opprette tjenesten automatisk (Esri, u.å.-d). Arbeidsgangen for dette framkommer også av figur 3, som viser et skjematisk forløp ved tjenesteoppsett. Som vist i figuren blir det også opprettet en REST-basert side for den aktuelle *Service*-en, av programvaren kalt *Service Directory*, en slags forside med oversikt over datasett og lenker til alle tilknyttede tjenester. Inne på denne siden kan man blant annet se på dataene i en selvstendig webkartløsning som blir automatisk opprettet, samt finne enkelte metadata og URI-er til de tilgjengelige tjenestene med det aktuelle datasettet (ArcGIS Developer, 2022).



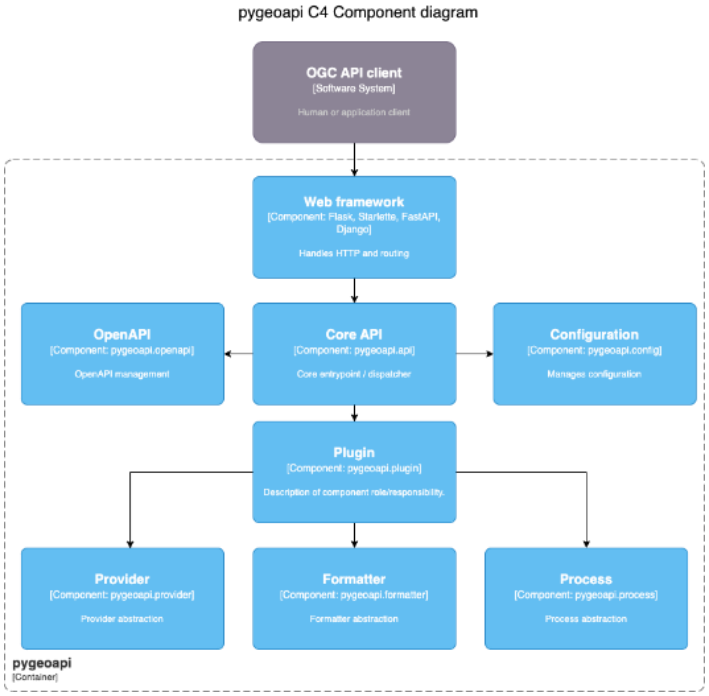
Figur 3: Skjematisk presentasjon av ArcGIS Servers arbeidsgang ved tjenesteoppsett. Egen Illustrasjon.

2.7.3 pygeoapi

pygeoapi er en forholdvis ny åpen kildekode-serverplattform for leveranse av geografisk data. Plattformen er fortsatt i utvikling etter å ha blitt lansert i 2018, og skiller seg noe ut blant andre plattformer da den primært er bygget på støtte mot OGC API-er, og ikke standarder fra OGC generelt. Som et resultat av dette støtter den heller ikke noen av de nåværende WxS-standardene, men en rekke OGC API-er, deriblant del 1 av Features (osGEO, u.å.-a). pygeoapi er basert på Python, og tillater å sette opp REST API-er ved bruk av OpenAPI, GeoJSON-data og et HTML-grensesnitt (pygeoapi, 2023a). Den er tilgjengelig for bruk både lokalt og gjennom server, kan settes opp på både Windows, Mac og Linux/ Ubuntu; med spesielt god støtte for sistnevnte ved oppsett på server.

Etter at pygeoapi er lastet ned lokalt eller på server, blir tjenestene definert gjennom en YAML-fil (pygeoapi 2023d). Denne er sammenlignbar med en XML-fil ved at den lister opp objekter hierarkisk og beskriver tilhørende egenskaper, men i YAML blir det brukt innrykk og kolonner til å strukturere informasjonen, framfor start- og sluttagger (Ben-Kiki, Evans og Net, 2021). I filen ligger det informasjon om filstier, databasetilkoblinger, koordinatsystemer, metadata og annen vesentlige data som er nødvendig for oppsett, og utgjør dermed en beskrivelse av hva tjenesten som settes opp gjør, lesbart for både mennesker og datamaskiner (Reini, 2022). Med tjenestens grunnleggende informasjon definert i konfigurasjonsfilen vil man gjennom pygeoapis nettleserbaserte landingsside få tilgang til tjenestene som er satt opp, og slik ha mulighet til å laste ned data i formatene JSON, GeoJSON, HTML, og CSV (OSGeo, u.å.-b)

Plattformen har også muliggjort å automatisk opprette API-dokumentasjon via OpenAPI med utgangspunkt i konfigurasjonsfilen, som fører til at man kan få tjenestespesifiserte definisjoner svært enkelt (pygeaoapi 2023e). Som illustrert i figur 4 utgjør konfigurasjonsfilen og OpenAPI-dokumentet kjernen av plattformen, som et bindeledd mellom brukergrensesnittet og *plugins*. Sistnevnte har av de ansvarlige bak pygeoapi blitt presentert som begrepet som brukes om ulike datakilder, -tilbydere og -adaptere, og er dermed funksjonaliteten som tilgjengeliggjør datasett eller informasjon for plattformen (Kralidis, 2022). Som et resultat av at pygeoapi per april 2023 ikke har støtte mot del 2 av Features er det kun mulig å sette opp tjenester i WGS84 (pygeoapi, 2023b).



Figur 4: Sentrale komponenter i pygeoapi og hvordan disse henger sammen. Hentet fra pygeoapi (2023c)

3 Metode

3.1 Sammenligning av WFS og Features

For å kunne gjøre en sammenligning av de to standardene, ble relevant litteratur og dokumentasjon undersøkt. Dette inkluderte å oppsøke og analysere offisielle standarddokumenter, forskningsartikler og publikasjoner fra anerkjente kilder og aktører innen aktuelle forskningsfelt, først og fremst geomatikk-, informatikk- og standardiseringsfeltet. En forståelse av dette materialet ble presentert i teorikapittelet, og videre bearbeidet og realisert i diskusjons- og konklusjonskapittelet av denne oppgaven. Den direkte sammenligningen har i stor grad vært litteraturbasert, og standardene har ikke blitt testet direkte utover slik dette blir beskrevet i kapittel 3.2.

3.2 Oppsett av tjenester

Etter en vurdering av aktuelle serverplattformer kom gruppen fram til å sette opp tjenester i henhold til WFS og Features med GeoServer og ArcGIS Server, som har støtte for begge. I tillegg ble det bestemt å sette opp en Features-tjeneste med pygeoapi, som kun har støtte for OGC API-er. I denne sammenheng representerer GeoServer og pygeoapi åpen-kildekode. ArcGIS Server representerer det proprietære alternativet, som en del av Esris programvare-pakke som krever lisens for bruk.

Gruppen ble 19. februar 2023 gitt tilgang til en Linux/Ubuntu-webserver på NTNUs domene som kunne brukes til tjenesteoppsett. For ArcGIS Servers del fikk gruppen kun tilgang til lisenser for en Windows-kompatibel versjon. Dette førte til tjenesteoppsett ikke kunne gjøres på NTNU-serveren, men måtte gjøres på egne datamaskiner gjennom *localhost*, og førte til at tjenestene ikke ville bli tilgjengeliggjort for andre enn gruppen. Selve installasjonsoppsettet ville derimot forløpt seg likt ved bruk av en Windows-server, og nokså likt på Linux/Ubuntu. Arbeidsgangen ved installasjon ble derfor vurdert til å fortsatt være sammenlignbar mot de andre plattformene.

All installasjon på NTNU-serveren var kommandobasert, og ble gjort fra egne datamaskiner gjennom et Git Bash-kommandovindu, med tilgang til serveren gjennom .ssh-nøkkelparing. Både GeoServer og pygeoapi er gjennom statusen som FOSS4G-progamvare kompatibel mot

Linux/Ubuntu på åpen lisens, og kunne installeres og settes opp på serveren som ble tildelt. For datahåndtering ble pgAdmin – et administrasjonsgrensesnitt for det åpne databasesystemet PostgreSQL – benyttet, installert på serveren med utvidelsen PostGIS for støtte mot geografiske data. En beskrivelse av fremgangsmåte for installasjon og oppsett av både PostGIS og de tre ulike plattformene framkommer i resultatdelen av oppgaven, samt gjennom trinnvise fremgangsmåter liggende som vedlegg A, B og C.

3.2.1 Datagrunnlag

Ved tjenesteoppsett ble det besluttet å bruke samme dataene på alle plattformer. Gruppen valgte seg ut to datasett til utprøving. Det ene var «*Administrative enheter fylker*», et forholdvis lite datasett med polygoner og linjer som representerer fylker, fylkesgrenser, samt territorial- og riksgrense, publisert åpent på Geonorge av Kartverket. Det andre var «*Naturvernområder*», et datasett med riksdekkende oversikt over områder og enkeltobjekter med vernevedtak i henhold til ulike naturvernslovverk, publisert åpent på Geonorge av Miljødirektoratet. Datasettet består hovedsakelig av polygoner, men også linje- og punktobjekter, gjennom de tre objekttypene *Naturvernområde*, *Naturverngrense* og *Teiggrensepunkt*. De to datasettene var relativt sett ganske forskjellige, spesielt i størrelse. Der fylkesdatasettet var lite, var naturvernområder mye større både i objekt- og attributtantall, og fysisk filstørrelse. Begge datasett var i koordinatsystemet WGS84 (EPSG:4326). Datasettene ble deretter lagt inn i PostGIS-databaser etablert på serveren ved bruk av QGIS og FME Workbench, og videre brukt til tjenesteoppsett.

3.3 Sammenligning av plattformer

For å sammenligne de ulike plattformene kom gruppen fram til fire vurderingsområder, som ble valgt med bakgrunn i de erfaringene som ble gjort ved oppsett og testing, og reflekterer hvordan gruppen vurderer de ulike plattformene helhetlig. Vurderingsområdene og definisjon av disse framkommer i punktene 3.3.1 til 3.3.4.

Til presentasjon av de ulike tjenestene i resultatdelen av oppgaven ble det fastslått å legge inn skjermdumper av noen utvalgte vesentlige detaljer for én av plattformene, først og fremst for å illustrere forskjellen mellom WFS og Features, her ble GeoServer brukt. I tillegg ble det bestemt å nøye beskrive arbeidsgangen for installasjon og tjenesteoppsett, ved å gjengi dette

trinnvis i respektive vedlegg for hver plattform, som også ville inneholde skjermdumper av tjenesteeksemepler.

Det var ulike egenskaper som ble anslått å være relevante å ha med, både for plattformene seg imellom og for tjenester satt opp i henhold til Features og WFS. Et viktig poeng her var å kunne vise til kall/spørringer som hadde lik funksjon, eller viste den samme informasjonen for de respektive tjenestetypene, for å tydeliggjøre forskjellen dem imellom. For Features-tjenestene ble det derfor besluttet å vise en representasjon av objektene slik det framstår i GeoJSON-format, som er direkte sammenlignbar mot kallet GetFeature til WFS-tjenesten med samme datasett. Det ble også bestemt å vise Features-tjenestenes landingsside i HTML eller JSON mot WFS-tjenestenes GetCapabilities-kall, med samme bakenforliggende tankegang. Dette ville også gi en oversikt over tjenestenes egenskaper, eventuelle ulikheter i resultatet av samme kall, samt som en dokumentasjon på at tjenestene faktisk fungerer.

Videre ble en tabell utarbeidet for å gi en oversikt over deler av funksjonaliteten til de ulike plattformene, med formål om å kunne gi en leser av denne oppgaven et enkelt overblikk over mulighetene som gis ved hver plattform.

3.3.1 Brukervennlig installasjon

Ved vurdering av installasjonsprosessen ble det lagt til grunn hvorvidt denne var enkel å gjennomføre. I denne sammenheng har gruppen definert installasjon som alt som må gjøres *før* man har mulighet til å sette opp tjenester på plattformen, altså at plattformen er klar til bruk. Det har blitt undersøkt om plattformene var tilgjengelig for nedlastning som ferdigkompilerte pakker gjennom f.eks. Docker eller andre, eller om de måtte bygges opp «fra bunnen av» på serveren. I tillegg ble det gjort en vurdering av hvordan det er å følge installasjonsveiviserne i den offisielle dokumentasjonen. Det har også blitt undersøkt om det er noen systemkrav som må innfris på servermaskinen for at plattformen kan installeres.

3.3.2 Sette opp tjenester

Her har gruppen sett på mulighetene og brukervennligheten plattformene har når det skal settes opp tjenester i henhold til de to standardene. Vurderingene som er lagt til grunn baserer seg på hvor enkelt det var å sette opp egne tjenester gjennom plattformene på NTNU-serveren/ lokal installasjon. Framgangsmåte ved oppsett ble vurdert gjennom en

sammenligning av anbefalt metodikk slik dette framgår i offisiell dokumentasjon. Det ble også vurdert hvor enkelt det er å endre på en tjeneste når den først er satt opp. I vurderingen har det også blitt tatt høyde for om plattformen trenger en tilleggsutvidelse for å støtte Features, og eventuelt hvor komplisert det er å finne og installere denne. I tillegg har det blitt vurdert muligheten til å generere egne API-dokumenter fra bl.a. OpenAPI under oppsett, eller om dette måtte gjøres på en alternativ måte.

3.3.3 Aktivt bruker-/utviklermiljø

For dette vurderingsområdet er det undersøkt om miljøene og utviklergruppene rundt plattformene er aktivt og bidrar til en progressiv utvikling av plattformen. Ved tjenesteoppsett og installasjon var slike ressurser i hyppig bruk, og erfaringene fra denne prosessen har derfor vært vektlagt ved sammenligning. På egne nettsider har plattformene oppgitt lenker til forum, enten på eget domene eller offisielle tråder hos ressurssider som gis.StackExchange eller Stack Overflow. Det har ved vurdering blitt undersøkt hvor mye informasjon som finnes i disse kanalene, om plattformene selv bruker dette til kommunikasjon inn mot miljøet eller om ressurssidene er utelukket brukerdrevet uten offisiell tilknytning til plattformen eller leverandør. Det har også blitt sett på om de som står bak plattformene aktivt benytter disse ressursene til brukerstøtte, og da med en rimelig responstid. For plattformer bygget på åpen-kildekode har det også blitt vektlagt positivt om utviklermiljøet bidrar med egenutviklede programtillegg eller -utvidelser til plattformen.

3.3.4 Brukervennlig dokumentasjon

Gruppen har utført en vurdering av den offisielle dokumentasjonen ved å se på strukturen og informasjonen som en helhet. Med offisiell dokumentasjon menes dokumentasjonen tilgjengelig på offisielle nettsteder tilknyttet de respektive plattformene. Dersom det finnes alternativ dokumentasjon skrevet av tredjeparter som ikke er publisert på offisielt nettsted, har denne ikke inngått i vurderingen. For å gjøre en grundig vurdering, har gruppen valgt ut noen fokusområder betraktet som viktigere enn andre. Disse inkluderer lesbarhet, innhold og informasjon, vurderingen har blitt gjort med kunnskapen som en geomatiker ville hatt som utgangspunkt.. Gruppen har også vurdert hvordan et aktivt bruker-/utviklermiljø rundt plattformen kan påvirke den offisielle dokumentasjonen. Ved å benytte en slik metodikk ønsker gruppen et helhetlig bilde av kvaliteten på dokumentasjonen.

4 Resultater

4.1 PostGIS

For å sikre at alle tjenester brukte de samme dataene uten konflikter, og for å etterligne industrien, ble PostGIS installert og aktivert på servermaskinen. Installasjonsprosessen for PostGIS var basert på Ellingwood og Drake (2018) sin framgangsmåte, men med en enkel modifisering for å kunne opprette nye databaser uten måtte opprette nye brukere. I stedet for å bygge programvaren fra bunnen av med originale binærfiler fra PostgreSQL, ble pakkeinstallasjonen på Linux-systemet brukt.

For å administrere PostgreSQL-databaser i et grafisk brukergrensesnitt, og dermed mer brukervennlig måte, ble pgAdmin4-web installert på NTNU-serveren. Den offisielle metoden beskrevet i pgAdmins dokumentasjon ble benyttet her. Dette ga bedre kontroll og enklere administrasjon av databasene uten å måte ha programvare installert på egen pc.

PostGIS ble installert som en nødvendig utvidelse til PostgreSQL for tilrettelegging av bruk med geografiske data. Her ble den innebygde pakkebehandleren i Linux benyttet for nedlastning, før den ble aktivert på serveren gjennom kommandoer i pgAdmin. Dette ble gjort for de databasene der det var nødvendig å lagre og håndtere geografisk informasjon.

For å kunne legge inn data med bruk av for eksempel QGIS eller FME ble det også åpnet for å kunne koble til PostgreSQL fra IP-adresser som ikke var lokale på servermaskinen. Dette ble gjort ved å endre på konfigurasjonsfilen postgresql.conf i en teksteditor. I filen ble linjen

`"listen_addresses = 'localhost'"` endret til `"listen_addresses = '*'"`

Det ble deretter lagt inn en nødvendig endring i en annen konfigurasjonsfil, pg_hba.conf, som tilsa å sette IP-adressen oppført under `"ipv4 local connections"` til `0.0.0.0/0`

Etter ferdig installasjon ble det lastet opp data ved hjelp av databasehåndterings-verktøyet i QGIS. Det ble opprettet en respektiv database for hver av de to datasettene. Gruppen hadde nå tilgjengelig to databaser med PostGIS-utvidelsen installert, administrerbar gjennom en nettside på serveren.

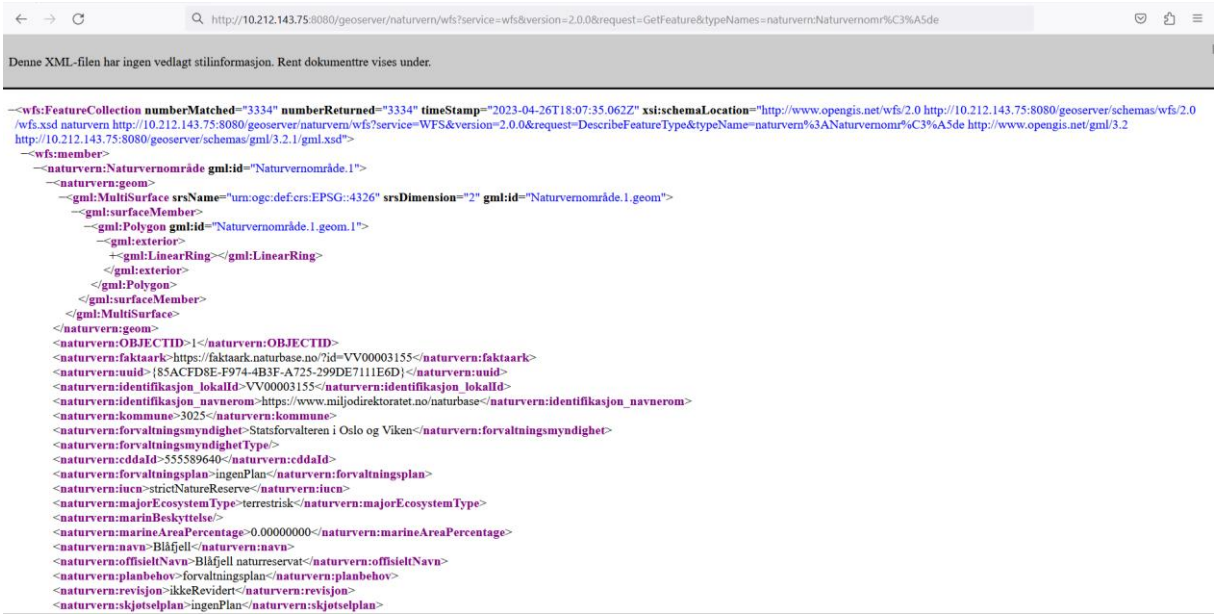
4.2 GeoServer

Forutsetningene før installasjon var at gruppen hadde en tom Linux/Ubuntu-webserver til rådighet. Som forklart i teoridelen er GeoServer bygget på Java, som derfor måtte installeres på serveren før selve programvaren.

Det ble fulgt en tredjeparts installasjonsveileder publisert av Jethva (2018) ved installasjon av programvaren, samtidig som GeoServers egen metode for utvidelsesinstallasjon ble fulgt for å installere Features-tillegget. Som en del av installasjonen måtte det opprettes en systemfil, dette ble gjort med startkommandoen `nano /usr/lib/systemd/system/geoserver.service`, med påfølgende kommandoer vist som trinn 7 i vedlegg A. Når dette var fullført var GeoServer klar til bruk, men ikke med OGC API-muligheter. Denne utvidelsen ble hentet fra en side tilknyttet GeoServers offisielle utvikler-/brukermiljø med kommandoen `wget https://build.geoserver.org/geoserver/2.22.x/community-2023-03-30/geoserver-2.22-SNAPSHOT-ogcapi-plugin.zip`, og pakket ut i en egenspesifisert mappe for utvidelser. Etter en omstart var GeoServer dermed klar til bruk med OGC API-er. En trinnvis fremgangsmåte for både programvare- og tilleggsinstallasjon, samt den nødvendige installasjon av Java finnes i vedlegg A. Med GeoServer og OGC API-tillegg ferdig installert ble videre tjenesteoppsettet gjort gjennom det nettleserbaserte brukergrensesnittet, og den kommandobaserte delen av oppsettet var nå ferdig.

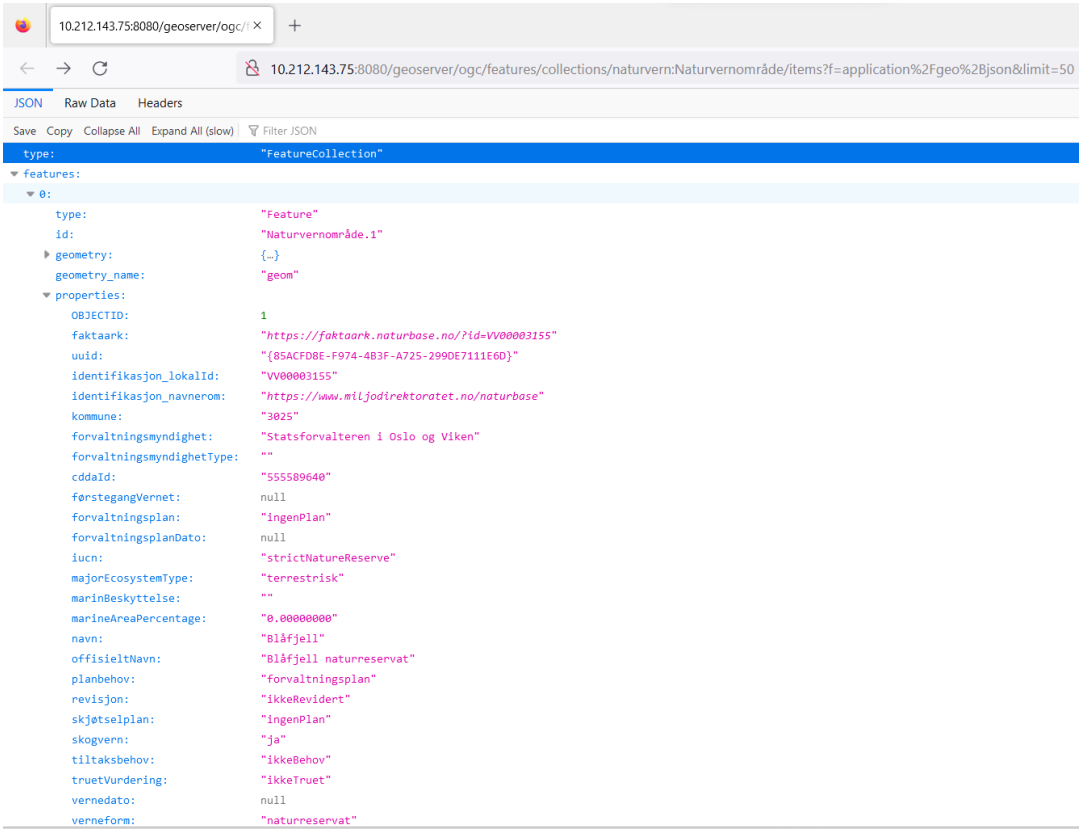
Når det gjelder oppsettet av tjenestene ble det opprettet en *workspace* for henholdsvis fylker og naturvernområder, som ved å definere en *store* med tilkobling til de respektive databasene i PostGIS fikk tilgang til datasettene. Det ble publisert tre lag fra naturverndatasettet og fem lag fra fylkesdatasettet, som ved enkle operasjoner i brukergrensesnittet ble tilgjengeliggjort som WFS- og OGC API-tjenester. Tjenestene ble testet med forskjellige kall, samt at det ble gjort en kobling mot tjenesten QGIS, for kontroll om datasettene framstod som tiltenkt.

I figur 5 vises et GetFeature-kall mot WFS-tjenesten naturvern, mot objekttypen Naturvernområde. Kallet viser tjenestens objekter i GML-format, i figuren er kun objektet med objekt-ID = 1 synlig (Blåfjell naturreservat), resten av objektene er å finne lenger ned i filen.



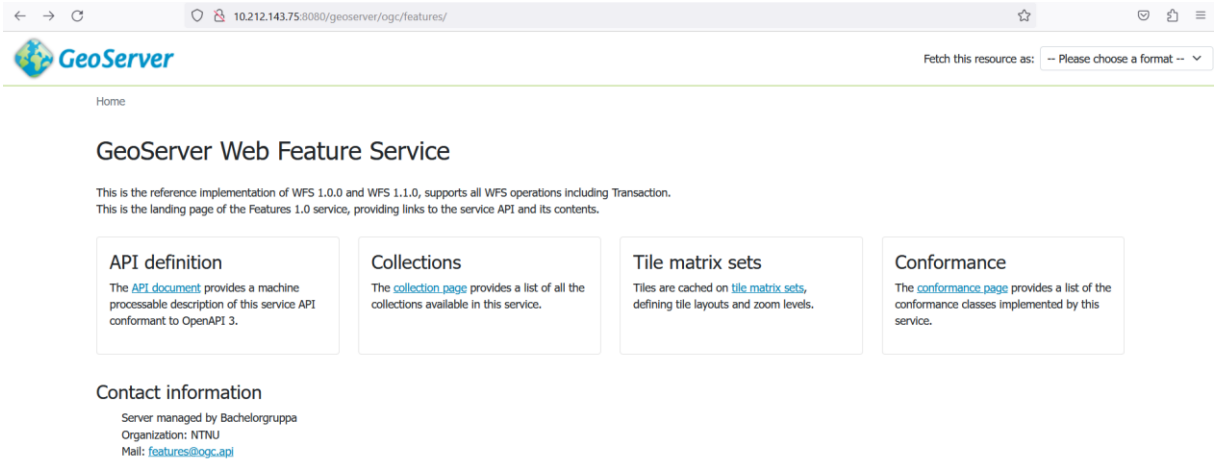
Figur 5: GetFeature-kall mot tjeneste i GeoServer

En GeoJSON-representasjon av objektsamlingen i Features-tjenesten med samme datasett vises i figur 6, man kan her finne tilsvarende informasjon som i GetFeature-kallet mot WFS-tjenesten.



Figur 6: Objektsamling i Feature-tjeneste satt opp med GeoServer

I figur 7 vises landingssiden på HTML-format, som framstår lik for begge tjenestene. Man kan med enkelhet trykke seg inn på de ulike sidene og finne informasjonen man måtte ønske om tjenesten. Merk at GeoServers landingsside er standardisert utformet med en tilleggsside som gir oversikt over cachede kartbilder for en OGC API Tiles-tjeneste, som ikke er aktuell for vår tjeneste.

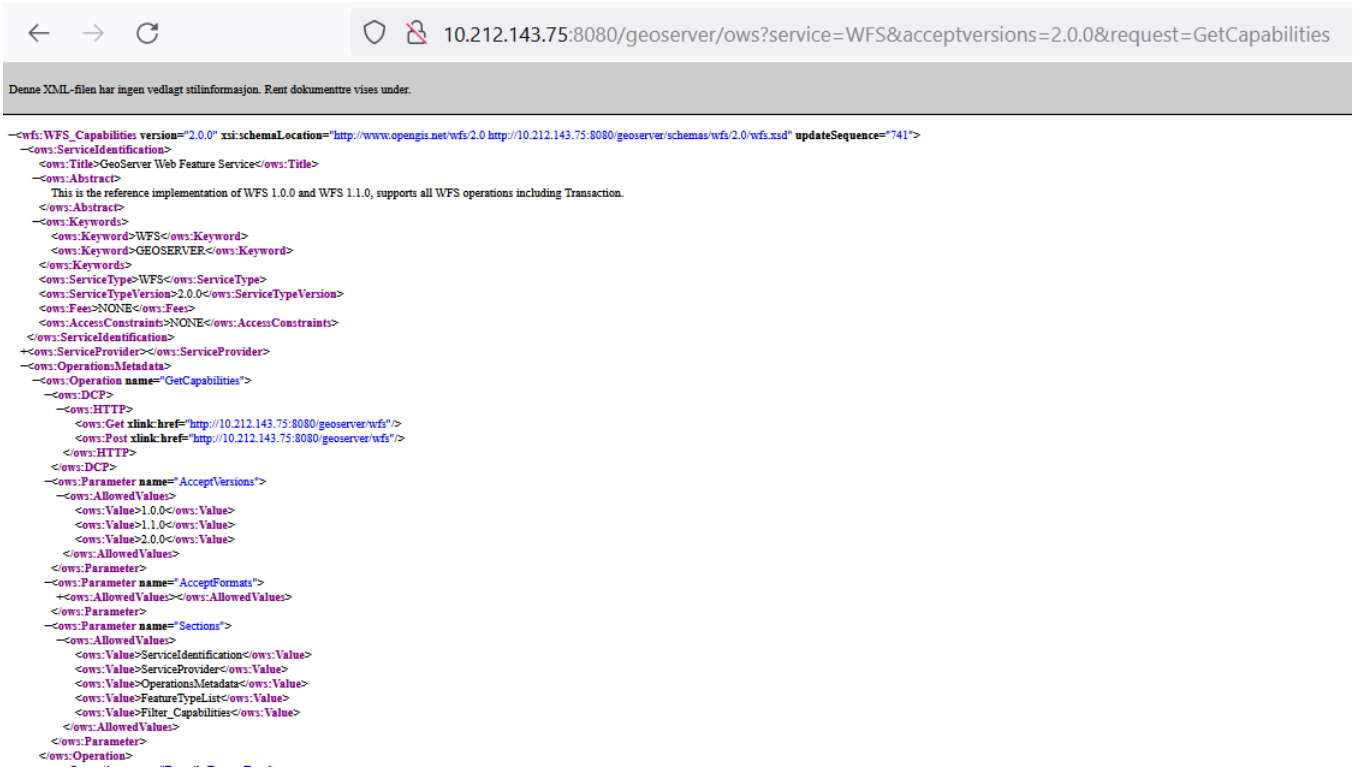


Figur 7: Landingsside i GeoServer på HTML-format

Om ønskelig kan man få landingssiden representert i JSON (figur 8), ved å velge dette i nedtrekks menyen «Please choose a format», som vist i øvre høyre hjørne på figur 7.

Figur 8: Landingsside i GeoServer på JSON-format

Figur 9 viser et GetCapabilities-kall mot WFS-tjenesten med fylkesdatasett. Her framkommer mye av den samme informasjonen som i landingssiden, merk blant annet kontaktinformasjonen som vises nederst på figur 7 også framkommer i GetCapabilities-kallet.



Figur 9: GetCapabilites-kall mot WFS-tjeneste satt opp med GeoServer

4.3 ArcGIS Server

Forutsetningen før installasjon var at det ikke var mulig å sette opp ArcGIS Server på webserveren som var tildelt da man ikke fikk tilgang til Linux/Ubuntu-kompatible lisenser. Programvaren måtte derfor installeres lokalt på en datamaskin med tjenester tilgjengeliggjort gjennom localhost. Gruppen fikk tilgang til installasjons- og autorisasjonsfil til ArcGIS Server v.11.0 via NTNUs avtale med distributør.

Programvaren ble lastet ned og installert med den offisielle Windows-installasjonsveiviseren, det ble her benyttet anbefalt framgangsmåte publisert av Esri for serveroppsett på én maskin (2022). Installasjon foregikk uten problemer og det ble opprettet en *Site* i Server Manager ved

første gangs oppstart. For tydelig strukturering ble det opprettet to Service-undermapper, egne mapper for tjenester tilhørende de to ulike datasettene som ble brukt.

I ArcGIS Pro ble det opprettet to prosjekter, hvor de to datasettene ble importert inn i hvert sitt respektive prosjekt. I prosjektene ble datasettene videre eksportert som tjeneste-definisjonsfiler, ved eksport av disse ble det huket av for opprettelse av WFS- og Features-tjenester. Dataene ble validert og godkjent av ArcGIS Pro, på datasettet med Naturvernområder i WGS84 måtte prosjekt-koordinatsystem defineres på nytt for å bli godkjent, dette trengtes ikke på Administrative enheter fylker i UTM sone 33. På Service Manager ble det i hver sin spesifiserte undermappe opprettet tjenester ved bruk av *Publish Service*-verktøyet, hvor tjenestedefinisjonsfilene ble lastet opp. Alle tjenester var fungerende rett etter opplasting av filen. En fullstendig fremgangsmåte av installasjon og tjenesteoppsett, samt skjermdumper som viser tjenestetesting framkommer i vedlegg B.

4.4 pygeoapi

Plattformen er bygget på Python, det er dermed en forutsetning at dette er installert i forkant av programvareinstallasjon. Python var i dette tilfellet allerede installert på serveren, dette ble dermed ikke gjort av gruppen i forbindelse med pygeoapi-installasjon. En trinnvis framgangsmåte for installasjon og eksempler på tjenestetesting finnes i vedlegg C. Konfigurasjonsfilen for oppsett finnes i vedlegg D.

Programvaren ble installert i henhold til offisiell framgangsmåte tilgjengelig i dokumentasjon. Det var her flere mulige tilnærmelser og installasjonsmetoder, men hvor det ble bestemt å benytte en pakke fra UbuntuGis (2022) linket fra den offisielle veilederen. Denne hentet med kommandoen `sudo add-apt-repository ppa:ubuntugis/ubuntugis-unstable`, etterfulgt av installasjonskommandoen `apt-get install python3-pygeoapi`. Metoden benyttet et virtuelt Python-miljø til installasjon, som noe forenklet betyr at det opprettes et isolert-fungerende Python-miljø med tilhørende biblioteker, uten å skape konflikt med eventuelle andre Python-miljøer allerede eksisterende på serveren. Som vist ved trinn 7 i vedlegg C, ble det ved bruk av kommando opprettet en konfigurasjonsfil i YAML-format. Det ble i første omgang kopiert inn en eksempelmal fra pygeoapis hjemmeside, all nødvendig informasjon i denne kunne enkelt tilpasses egne tjenester og dermed brukes som utgangspunkt ved oppsett. Det ble blant annet definert databasetilkobling, som vist i linjene 92-103

(naturvern, Teiggrensepunkt) og 179-190 (Fylker, polygoner) i vedlegg D. Begge tjenester kunne defineres i samme konfigurasjonsfil, adskilt med navn og innrykks plassering i filen. Når dette var gjort ble det opprettet et OpenAPI-dokument basert på konfigurasjonsfilen. Dette skjer automatisk med kommandoene vist i trinn 9-11 i vedlegg C, og man får da tjenestetilpassete API-definisjoner. Tjenester var fungerende med en gang etter oppsett.

4.5 Funksjonalitetstabell

Tabell 1.

Oversikt over sentrale egenskaper tilknyttet plattformenes funksjonalitet

	GeoServer	ArcGIS	pygeoapi
Støtte for dynamisk kartvisning av data direkte i tjenesten	*Nei*	*Ja**	*Ja*
Støtte for representasjonsformater utover de i standarden	*Ja*	*Ja*	*Ja*
Mulighet for å sette opp WxS-tjenester	*Ja*	*Ja*	*Nei*
Mulighet for å sette opp OGC API-tjenester	*Ja**	*Ja*	*Ja*
Avhengig av annen programvare for tjenesteoppsett	*Nei*	*Ja*	*Nei*
Støtte for individuell tilpasning gjennom utvidelser	*Ja*	*Nei*	*Ja*

5 Diskusjon

5.1 Sammenligning av WFS og Features

Selv om oppgaven kun har undersøkt Features-standarden, vil mye av diskusjonen rundt forskjellen mellom WFS og Features omhandle mer generelle forskjeller, som også vil gjelde for eventuell sammenligning mellom andre standarder fra WxS- og OGC API-samlingen.

På bakgrunn av at OGC gikk bort fra å lansere en oppdatert versjon av WFS, til å lansere en ny type standard i form av Features, forventet vi å finne noen betydelige forskjeller mellom de to. Områder som har blitt undersøkt i denne oppgaven er hovedsakelig arkitektur, presentasjon og dokumentasjon, og hvilke formater som tillates for dataene som overføres. Nøkkelbegreper i den forbindelse er REST, GeoJSON og OpenAPI.

Når det gjelder arkitektur, vil innføringen av de nye standardene representere et skifte fra tjenesteorientert til ressursorientert arkitektur i form av REST. Både REST og den tjenesteorienterte arkitekturen til WFS, baserer seg på bruk av HTTP-protokollen for utveksling av informasjon mellom tjenere og klienter. Det vil likevel være noe forskjellig hvordan de to tjenestetypene utnytter HTTP. Der WFS bare bruker HTTP som ren overføringsmetode – begrenset til metodene GET og POST –vil Features som et REST-API kunne benytte seg av samtlige av HTTP-metodene, og ikke være avhengig av sekundære protokoller slik som SOAP. Forskjellen mellom en tjenesteorientert og ressursorienter oppbygging, gjenspeiles også i selve URL-en som brukes for å interagere med tjenesten/API-et. Ved å sammenligne URL-ene fra Figur 5 og 6 – som presenterer objektet Blåfjell naturreservat fra henholdsvis WFS-tjenesten og Features-tjenesten – er det en strukturell forskjell i oppbyggingen. Hos Features ser man hvordan man har kommet frem til objektet stegvis ut ifra landingssiden (*landingsside*/features/collections/naturvern:Naturvernområder), mens hos WFS er URL-en har en mer ustrukturert oppbygging, sentrert rundt operasjonen (*GetFeatures*), med spesifikasjoner for å komme frem til ønsket objekt gitt av en rekke nøkkel-verdi par (*key-value pairs*), slik som *typeNames=naturvern*.

En webtjeneste eller et API kan dermed vurderes som mer eller mindre RESTful avhengig av hvor godt de oppfyller de satte designkriteriene til REST. Selv om de første utgavene av WxS-standardene ble utgitt før begrepet REST ble lansert, innehar flere av de noe REST-funksjonalitet (se for eksempel OGC, 2016), som vil si at de i noen grad oppfyller et eller flere av designkriteriene til REST. Selv om ofte hovedfokuset i omtalelser av den nye generasjonen med standarder fra OGC gjerne fokusert rundt REST, har noen av de gamle standardene allerede hatt en viss grad av REST-funksjonalitet. Det at REST dermed blir fokusert på som et av de nye aspektene med de nye standardene, dreier seg derfor trolig mer om at Features og de resterende standardene i større grad kan betraktes som fullverdige REST API-er.

Når det gjelder formater, så kan Features-standarden sies å bryte med den tette tilknytningen til XML-formatet på flere måter. Det vil for eksempel ikke lenger være krav om at GML skal være standardformat ved overføring av data, da Features ikke legger opp til at tjenesten skal være knyttet til et bestemt format, selv om støtte for HTML og GeoJSON blir anbefalt. Ellers i IT-verdenen er GeoJSON et utbredt format for geodata, så anbefalingen om å ha støtte for GeoJSON kan kalles en slags tilpasning til dette, i tillegg til man trolig ønsker API-er som kan overføre data på en rask måte, noe som det enkle og mindre ordrike filformatet til GeoJSON kan bidra med. Anbefalingen om HTML er mer knyttet til å gjøre selve dataene i API-et mer tilgjengelig for søkemotorer.

Videre så vil ikke tjenester som er satt opp i henhold til Features være tjenesteorienterte, og dermed heller ikke være bygd opp av XML-baserte protokoller slik som SOAP og WSDL. Det at Features beveger seg vekk fra XML, er i tråd med å målet om at de den nye generasjonen av standardene skal benytte de metodene som blir brukt ellers ved oppbygging av tjenester og API-er. Dette fordi XML i mange sammenhenger blir sett på som et noe utdatert format, da det ofte får en ordrik og tung struktur, som igjen krever mer av nettverket ved overføring mellom tjenere og klienter.

Overgangen fra WFS til Features innebærer også endringer i hvordan brukeren får tjenesten presentert. Som nevnt i introduksjonen, starter gjerne interaksjonen med en WFS-tjenesten med et *GetCapabilites*-kall, hvor man får returnert et GML-dokument som vist i figur 9. GML-dokumentet er ment å skulle gi brukeren en oversikt over tjenesten ved å presentere metadata slik som dataeier og kontaktinfo, i tillegg til informasjon om operasjoner tjenesten støtter, og hvilke objekttyper selve datasettet består av.

Til sammenligning kan landingssiden til en typisk Features-tjeneste, som eksemplifisert i figur 7, sies å være mer oversiktlig. På landingssiden er informasjonen presentert som en HTML-side, hvor brukeren/klienten får informasjonen presentert oppdelt i kategorier – slik som *definition, contact info* og *collections*. Til sammen utgjør informasjonen på landingssiden alt det brukeren trenger for å finne frem til enhver ressurs i datasettet, og brukeren kan enkelt klikke seg videre fra landingssiden for å finne frem til ønskede data.

På den måten kan landingssiden til Features-tjenesten være et mer brukervennlig alternativ til GML-dokumentet, spesielt for brukere som ikke er vant med GML-oppbyggingen av et dokument, men også for en geomatiker som er kjent med strukturen til *GetCapabilites-dokument* fra før. En annen viktig egenskap med HTML-landingssiden, er at gjør det lettere for tradisjonelle søkemotorer å oppdage datasettet, samt hver av objekttypene og objektene, siden disse også har sine egne HTML-dokumenter. Til sammenligning så er WFS-tjenestene i den geografiske infrastrukturen i stor grad tilgjengelige via Geonorge sin kartkatalog, og tradisjonelle søkemotorer har vanskelig for å frem til objekttyper og objekter siden de ligger i et GML-dokument som man først finner ved å sende et *GetFeatures*-kall.

Til sist referer nøkkelordet OpenAPI til hvordan Features-tjeneste nå skal bli beskrevet og dokumentert. Ved å bruke OAS sin standard for oppbygging av web API-er, får de nye standardene flere ønskede egenskaper. Siden denne måten å spesifisere et API på er en allerede utbredt metode, vil det både kunne effektivisere arbeidet med å sette opp tjenester, og også kunne være med å senke terskelen for å sette opp tjenester i henhold til OGC sine standarden. Bruken av OAS åpner også mer opp for at tjenesteutviklere som ikke er vant med å jobbe med geodata lettere tar i bruk standarden.

Det kan argumenteres for at de nye standardene både kom som en noe forsinket respons på endret teknologi eller metodikk innen utvikling av API-er, men at også endringer innen hvem de typiske tilbyderne og brukerne av slike standarder er. Som nevnt i teorikapittelet kan stadig mer av de dataene som i dag blir samlet inn/generert, kalles geografiske data, noe som videre kan bety at gruppen av både brukere og tilbydere av API-er for geografiske data har utvidet seg. Videre vil de nye standardene fra OGC kunne støtte opp om nye brukere, når de når ligner mer på hvordan API-er generelt blir laget, samt blir spesifiserte ved hjelp av OAS, som er en allerede veletablert spesifikasjon for beskrivelse av web API-er.

5.2 Sammenligning av platformer

5.2.1 Brukervennlig installasjon

Ved en vurdering av installasjonsprosessen for plattformene, er det viktig å ta høyde for at det er vesensforskjell mellom de to åpne Linux/Ubuntu-baserte installasjonene av GeoServer og pygeoapi, og Windows-installasjonen av ArcGIS Server. Der både pygeoapi og GeoServer ble satt opp på NTNU-serveren via Git Bash-kontroll fra egne maskiner, ble ArcGIS Server installert med et grafisk installasjonsgrensesnitt på en Windows-maskin. Sistnevnte er en installasjonsmetode som er kjent for de fleste som bruker datamaskiner, og følger en trinnvis og 'klassisk' Windows-framgangsmåte for installasjon som antagelig er gjennomførbar med et minimum av dataferdigheter. Som proprietær programvare, skiller ArcGIS Server seg ut ved at man må ha liggende en autorisasjonsfil på maskinen, som man må angi bane til i installasjonsgrensesnittet for å kunne fullføre installasjonen. Denne er kun tilgjengelig ved ervervelse av lisens, man kan dermed ikke installere programvaren uten. Man er også avhengig av å ha tilgang til en fysisk installasjonsfil, som det antas man får tilgang til ved kjøp av programvaren fra distributør; det ble ikke funnet noen installasjonsfil åpent tilgjengelig på Esri eller Geodata AS (norsk enedistributør) sine nettsteder.

En vanlig metode for installasjon av slike tjenester vil være med bruk av programvaresystemet Docker. Denne metoden ble valgt bort tidlig da kunnskapsnivået rundt Dockers funksjonalitet ikke var så utbredt blant gruppemedlemmene. Det har i denne sammenheng også vært et poeng å gjøre både installasjon og tjenesteoppsett i henhold til det plattformtilbyderne selv anser som beste metode (*best practice)*. Dette var gjennomførbart for pygeoapi og ArcGIS Server, men ikke GeoServer, som blir videre diskutert senere i delkapittelet.

GeoServer og pygeoapi ble installert med kommandoer på serveren via Git Bash, blant annet *apt-get* og *wget*, som vist i vedlegg A og C. Begge plattformene var tilgjengelige som ferdigkompilerte pakker, som i stor grad forenklet prosessen. Det ble brukt en pakke fra UbuntuGIS for pygeoapi, og SourceForge for GeoServer, men lenker til flere alternative pakker var også tilgjengelige på de offisielle nettstedene**.** Installasjonsveivisere ble fulgt, som bidro til en forenkling. Ved ferdig installasjon måtte det allikevel gjøres noe ekstra trinn for å kunne bruke plattformene slik gruppen ønsket. På GeoServer måtte det opprettes både bruker(e) og en service-fil, alt ved bruk av kommandoer i Git Bash. I tillegg måtte man

installere utvidelsen for støtte mot OGC API-er gjennom en *wget*-kommando. For pygeoapis del måtte det lages en YAML-konfigurasjonsfil og et OpenAPI-dokument, som også ble opprettet og endret på gjennom kommandoer.

Det har hele veien vært en grunntanke å følge plattformene offisielle installasjonsguider ved installasjon. Dette har blitt gjort for pygeoapi og ArcGIS Server, som begge hadde enkle veivisere publisert på eget nettsted. For GeoServers del ble installasjonen derimot to-delt. Installasjonen av selve programvaren ble gjort ved å følge en veiviser fra en tredjepart tilknyttet utviklermiljøet rund plattformen, samtidig som installasjon av OGC API-utvidelsen ble gjort ved å følge den offisielle fremgangsmåten for utvidelsesinstallasjon tilgjengelig i dokumentasjonen. Denne måten ble foretrukket da det ble oppdaget enkelte mangler i GeoServers egne installasjonsveiviser for Linux/Ubuntu, som ble forsøkt fulgt uten at den fungerte skikkelig. Det ble i denne forbindelse bemerket at tredjeparts-guiden hadde betydelig flere trinn, og var også mye mer detaljert enn den offisielle. GeoServer fungerte omgående etter installasjon på den alternative måten, også med OGC API-støtte etter bruk av den offisielle installasjonsmetode for utvidelser.

GeoServer og pygeoapi er bygget på henholdsvis Java og Python, og siden gruppen ble gitt en «tom» server må disse installeres i tillegg. Det viste seg at Python allerede var installert på serveren fra før, men ikke Java. Installasjonen av Java inngår dermed i den trinnvise fremgangsmåten som vist i vedlegg A. Ved installasjon av ArcGIS Server vil installasjonsgrensesnittet gi brukeren mulighet til nedlastning og installasjon av det som eventuelt måtte mangle av nødvendige kompilatorer eller programmeringsspråk direkte i grensesnittet. Ellers var det ikke nevneverdige systemkrav på noen av plattformene, utover tilstrekkelig tilgjengelig minne.

For gruppens del gikk installasjonsprosessen fint på samtlige plattformer, og møtte heller ikke på noen nevneverdige utfordringer knyttet til installasjon av tillegg eller de konfigurasjonsfil-opprettelsene som måtte gjøres. Det bør allikevel påpekes at man ved installasjon av pygeoapi eller GeoServer bør inneha grunnleggende til middels kunnskap om hvordan kommandobasert installasjon og databehandling foregår i Linux/Ubuntu. Dette er det ikke alle som har, og selv om gruppen opplevde en problemfri installasjonsprosess kan sannsynligvis mange ha nytte av ArcGIS Servers enkle og velkjente installasjonsmetode, som ikke krever denne type kunnskap.

Det finnes også andre installasjonsmetoder for GeoServer og pygeoapi på blant annet Windows, som gir tilnærmet ferdig programvare etter enkel installasjon, som også kan være nyttig for mange. Ulempen er at disse ofte kan være veldig restriktive i hvordan man kan tilpasse plattformene uten inngående kunnskap om flere ulike datasystemer. Dette er blant annet gjenkjennbart i ArcGIS Server, hvor man er veldig låst til en bestemt fremgangsmåte med mindre tilpasningsmuligheter, for eksempel om man ønsker å installere utvidelser som ikke er en del av programvaren som standard. Formålet med å gjøre alt dette fra «bunnen av» var at dette ville gi gruppen et grunnlag for å vurdere arbeidsgangen ved tjenesteoppsett, ved å gjennomgå hele prosessen fra en «tom» server til tjenesten er oppe og går. Dette ga også mer innsikt i de ulike programmenes virkemåte, uten at dette ble for avansert ved installasjon med binærfiler.

5.2.2 Sette opp tjenester

Arbeidsgangen ved oppsett av tjenester var forholdvis ulik de tre plattformene imellom. På GeoServer ble alt gjort via det nettleserbaserte brukergrensesnittet, på ArcGIS Server ble en tjenestene definisjonsfil opprettet i ArcGIS Pro, og i pygeoapi ble tjenester definert gjennom ulike elementer i en konfigurasjonsfil i YAML-format.

I GeoServer ble tjenester konfigurert gjennom en nettleser. Dette kan betraktes som en forholdvis enkel måte å sette opp tjenester på, da alt foregår i et grafisk brukergrensesnitt med relativt intuitive funksjoner, menyer og knapper. Det ble opprettet en Workspace for hver tjeneste, en tilkobling til datasettene i PostGIS ble definert i hver sin respektive Store, og de ulike lagene ble publisert. Veiledninger for tjenesteoppsett var tilgjengelig i dokumentasjon, og ble fulgt uten videre utfordringer. Overordnet var det forholdvis problemfritt å sette opp tjenester i GeoServer, og oppsettet var likt for både WFS og Features ved at det med Features-utvidelsen installert kan hukes av for dette ved publisering av lag. Det var også mulig å endre på tjenester etter oppsett ved å for eksempel kun endre på en Store i en Workspace.

For ArcGIS Servers del, hvor det også var likt oppsett for både WFS og Features, ble det ved eksport av tjenestedefinisjonsfil huket av for ønskede tjenestetyper i valg for dette. Tjenestedefinisjonsfilen ble så lastet opp i Server Manger, og tjenestene var oppe med en gang. Metodikken her kan beskrives som svært brukervennlig og enkel, fra man legger inn et datasett i ArcGIS Pro til man har en standard-kompatibel tjeneste oppe og går er det snakk om få minutters arbeid. Det var allikevel enkelte mindre alvorlige observasjoner som måtte tas

hensyn til ved oppsett. Blant flere ting var dette at objekter i datasettene automatisk ble indeksert på nytt med en egen «ESRI:objectID», lagt til som en ekstra kolonne i attributtabellen ved tjenestedefinisjonsfil-eksport. Verdiene som ble tildelt hvert enkeltobjekt i denne kolonnen var identiske med de i objekt-ID-kolonnen som fantes i datasettene fra før. Dette kan i mange sammenhenger være en uønsket funksjon, og dersom det allerede eksisterer en objekt-identifikator med unike verdier burde det være mulig å bruke denne, framfor å legge inn redundant informasjon i datasettet. Dette kan være spesielt negativt om det skal publiseres datasett med et stort objekt- og attributtantall i utgangspunktet.

En annen bemerkning er at det oppstod enkelte problemer ved eksport av tjeneste-definisjonsfil hvor bokstavene *Æ*, *Ø* eller Å var en del av tjeneste- eller filnavn. I et tidlig forsøk på tjenesteoppsett ble både tjeneste og fil navngitt *Naturvernområde*, hvor bokstaven Å inngår. Filen ble lastet opp via Service Manager og tjenesten ble opprettet etter vanlig prosedyre. Både WFS og Features-tjenestene brukte derimot svært lang tid på å respondere på kall, og fungerte generelt dårlig. I påfølgende forsøk ble, som eneste endring fra *Naturvernområde*, fil- og tjenestenavn forandret til *VernetNatur*. Alle tjenestene virket som tiltenkt etter denne endringen. Samme effekt oppstod når det ble lagt inn data i ArcGIS Pro som ikke befant seg på rot-nivå på C-disken på datamaskinen tjenestene ble satt opp på. Også her forsvant disse utfordringene etter utbedring, som var å opprette en egen mappe til serverdata på rot-nivå. Å endre på en eksisterende tjeneste er heller ikke mulig uten å eksportere en ny definisjonsfil fra ArcGIS Pro, det er eventuelt kun navn som kan endres direkte i Server-programvaren. En siste observasjon tilknyttet tjenesteoppsett med ArcGIS Server er at dette er den eneste av de tre plattformene som krever en tilleggsprogramvare for å sette opp tjenester, nemlig ArcGIS Pro. I de fleste sammenhenger er nok dette en ikke-eksisterende problemstilling, da ArcGIS Pro og ArcGIS Server trolig er populære tilbud fra distributører i en pakkeløsning, bl.a. gjennom ArcGIS Enterprise. Sannsynligvis vil også en aktuell bruker av ArcGIS Server ha kjennskap til Esris programvarepakker fra før av, hvor ArcGIS Pro kan betraktes som en sentral del.

Oppsett av tjenester med pygeoapi skiller seg vesentlig ut fra de to andre, ved at dette foregår med tjenestedefinisjoner i en YAML-basert konfigurasjonsfil framfor i et grafisk brukergrensesnitt. Denne metoden kommer med enkelte fordeler og ulemper. Man må i stor grad skrive og definere innholdet i denne filen selv, i gruppens tilfelle ble det tatt utgangspunkt i en offisiell mal. Dette gir en større grad av kontroll og innsikt i hvilke

bestanddeler som må med for å få på plass en fungerende tjeneste, blant annet hjulpet av YAML-formatets struktur og lesbarhet. I tillegg kan det antas at et dokument i YAML-formatet, som i enkel forstand bare er en type tekstfil, vil kreve langt mindre diskplass enn det et grafisk brukergrensesnitt ville gjort. En utfordring er derimot at man bør inneha en viss forståelse av hvordan syntaks til YAML-formatet er bygget opp, som kan være krevende for uerfarne brukere. I tillegg vil man med denne metoden ikke ha noen visuell feedback på om YAML-dokumentet inneholder feil i syntaks eller struktur, den erfaringsmessig beste måten å sjekke dette på var ved ren testing av tjenestene. I prinsippet finnes det en valideringskommando man kan bruke for dette formålet, denne ble testet ved oppsett. Erfaringen ble at den fungerer fint for kontroll av struktur og at alle påkrevde bestanddeler inngår i filen, men ikke for kontroll av innhold. Dersom det for eksempel er en skrivefeil for et tabellnavn i definisjon av databasetilkobling, vil ikke valideringsfunksjonen nødvendigvis oppdage dette. Tjenester satt opp med pygeoapi er forholdvis enkle å endre på når de først er oppe og går, ved at man allerede har en eksisterende konfigurasjonsfil å jobbe i. Dersom det for eksempel må byttes datakilde kan dette enkelt gjøres i konfigurasjonsfilen uten å påvirke tjenestene som allerede er oppe og går underveis i endringen.

Av de tre plattformene har antagelig pygeoapi den mest kompliserte arbeidsgangen for oppsett av tjenester. Muligheten til å opprette en tjenestespesifisert API-dokumentasjon med utgangspunkt i konfigurasjonsfilen er derimot en funksjon som forenkler prosessen rundt tjenesteoppsett en god del. For GeoServers del er tjenesteoppsett forholdsvis brukervennlig, men det blir vurdert som negativt at det ikke er mulig å sette opp en Features-tjeneste uten ha installert en utvidelse, som for flere kan være et kompliserende ledd. Med overgangen mot OGC API-er vil det allikevel være ventet at funksjonalitet for dette inngår i framtidige versjoner av GeoServer. Det merkes også at man hverken med ArcGIS Server eller GeoServer har muligheten til å opprette tjenestespesifikk API-dokumentasjon. For GeoServers del sier offisiell dokumentasjon at dette blir opprettet automatisk, men man har tilsynelatende ingen enkel metode å verifisere eller endre på denne. Lignende erfaringer er gjort i ArcGIS Server, hvor det også blir automatisk opprettet uten mulighet for verifisering. ArcGIS Server har et svært enkelt forløp ved tjenesteoppsett. Det ble bemerket under oppsett at det i grunnen er svært lite arbeid som gjøres i *programvaren* ArcGIS Server; i hovedtrekk fungerer denne mer som «sentral» man kan laste opp tjenestedefinisjonsfiler til, og som tjenester blir publisert gjennom. Størstedelen av arbeidet i forkant av publisering gjøres i ArcGIS Pro, hvor man

også har enkelte utfordringer, blant annet med at data blir lagt til i datasettene ved eksport av tjenestedefinisjonsfil.

5.2.3 Aktivt bruker-/utviklermiljø

Med vurderingen av hvor aktivt et bruker-/utviklermiljø er må det tas hensyn til at plattformene har ulike utgangspunkt i hvordan brukere eventuelt kan bidra eller være aktive. For en åpen kildekode-programvare vil miljøet rundt kunne bidra med utvidelser, utbedringer og endringer, og står veldig fritt til å gjøre som ønsket med kildekoden. For en proprietær programvare som ArcGIS Server har man ikke denne muligheten, men det betyr ikke at det ikke finnes et brukermiljø.

Esri administrerer et eget brukerforum på nett, *Esri Community*, hvor de ulike programmene til selskapet har egne underforum. Her diskuteres gjerne temaer som funksjonalitet, utfordringer, teknisk støtte og GIS generelt. Diskusjonstråder tilknyttet ArcGIS Server ligger i underforumet til ArcGIS Enterprise, hvor det pr. mai 2023 finnes i overkant av 5000 tråder med ArcGIS Server tagget som tema. Et viktig poeng er at ansatte og andre tilknyttet Esri i stor grad er aktive her, og ikke bare programvarebrukere. Man kan derfor få betraktninger og bistand fra offisielt hold, og responstiden virker å være kort. I tillegg har Esri opprettet et forum kalt *ArcGIS Ideas*, som står litt på siden av det tradisjonelle brukerforumet. I dette forumet kan man komme med forslag tilknyttet funksjonalitet og utbedringer til eksisterende programmer, men også idékonsepter til ny programvare.

GeoServer har ikke et offisielt brukerforum på samme måte, men har offisielle tråder på ressurssidene gis.Stackexchange og Stackoverflow, samt et eget område på GitHub hvor kildekoden er tilgjengelig. Inntrykket er at disse er aktivt i bruk, både av de ansvarlige bak plattformen og et større antall brukere, og at en får svar ved spørsmål. Dersom en ønsker å komme med forslag til endringer i kildekoden eller utvikle egne utvidelser gjøres dette primært via GitHub, hvor det pr. mai 2023 står oppført 318 aktive bidragsytere. GeoServer har i den forbindelse publisert noen retningslinjer som bør følges dersom utvidelsen eller endringen skal bli en del av den offisielle programvaren, sannsynligvis for å sikre en viss grad av konformitet på ny kildekode. Det finnes en rekke uoffisielle utvidelser tilgjengelig med utspring fra utviklermiljøet, som GeoServer gir oversikt over på egen hjemmeside. GeoServer kommer som regel i ny versjon to ganger i året, hvor eventuelle feilrettelser, nye utvidelser, funksjoner og forbedringer som regel inngår.

pygeoapi har et relativt lite bruker-/utviklermiljø, sammenlignet med de to andre. Det finnes også her offisielle forum og GitHub-områder, men det bemerkes at disse har langt færre deltagere enn tilsvarende løsninger for ArcGIS Server og GeoServer. På forumet er det knapt med informasjon, og foreløpig kun ett innlegg fra 2023 pr. 13. mai. På GitHub er det oppført 60 bidragsytere. Dette kan ha flere årsaker, en sannsynlig grunn er at pygeoapi relativt sett er mye yngre enn de to andre, som begge har eksistert i over 20 år og dermed har hatt tid til å bygge opp en større bruker- og utviklergruppe. Den manglende støtten mot WxS kan også påvirke her, og det kan antas at man ville favnet om flere om denne muligheten var til stede, samtidig har noe av grunntanken bak pygeoapi nettopp vært å kun tilby oppsett av OGC API-er. Inntrykket er at denne spesialiseringen mot OGC API-er er noe de er forholdvis alene om blant plattformtilbydere.

Antallet bidragsytere i miljøet til en åpen kildekode-plattform kan fungere som et mål på graden av kvalitetssikring. pygeoapi har et lite miljø, og det kan være en svakhet at såpass få personer deltar, selv om man ikke nødvendigvis har noe grunnlag for å betvile disse personenes ferdigheter. Teknisk støtte er også en sentral del av et brukermiljø, og for åpen kildekode-plattformene vil dette i stor grad være basert på frivillighet, med ressurser tilgjengelig via diverse forum. Det kan dermed være en forutsetning at man har et veletablert brukermiljø for å få hjelp til eventuelle problemer. For proprietær programvare vil det her være et kommersielt hensyn, det kan antas at distributører av ArcGIS Server kan ta betalt for supporttjenester, eller at dette inngår som en del av prisen som betales for lisenser. ArcGIS Server har et stort frivillig brukermiljø i tillegg, så her får man litt av begge verdener. Med en proprietær programvare kan en forvente at problemer eller forespørsler blir behandlet innen rimelig tid, noe man ikke har garantier for med åpen kildekode. Samtidig antas det, at med et aktivt og etablert utviklermiljø, kan det gå fortere å få rettet opp feil eller lagt til nye funksjoner på åpen kildekode-programvare framfor proprietært. Dette grunnet i at flere kan være på saken raskere enn om det skal løses av selskapet bak en proprietær programvare.

5.2.4 Brukervennlig dokumentasjon

ArcGIS har betydelig mer detaljert dokumentasjon enn de andre plattformene som er utforsket. Esri har lagt opp til at «alle» skal kunne gjennomføre installasjonsprosessen og operasjoner i programvaren, ved at ting er enkelt og detaljert forklart. Det er også godt forklart hvordan man får eventuelle andre underprosesser, filer og databaser til å passe inn i

systemet slik at man kan publisere en stor mengde ulike datakilder. Sammen med det at Esri er eneste utvikler og vedlikeholder av programvaren må man derimot anta at all informasjon er korrekt. Informasjonen er skrevet på en slik måte at en person uten geomatikkfaglig bakgrunn skal kunne forså og gjennomføre ulike operasjoner.

Som bruker har en ikke direkte mulighet til å endre på dokumentasjonen, ei hellermulighet til å forenkle eller forklare eventuelle uklarheter bedre uten å legge det ut på et område til knyttet tredjepart eller Esri Communities. Dette gjør at man enten må stole på utgiver eller søke informasjon fra tredjepartskilder eller brukerforum.

GeoServer har en dokumentasjon som er flerdelt, med en del som fokuserer på oppsett og installasjon for brukere og en som er ment for utviklerbidrag. GeoServers dokumentasjon for brukere var mer relevant i denne sammenhengen, det er den som har blitt brukt ved tjenesteoppsett og videre vurdert. Denne er igjen delt inn i flere ulike deler, som vi grovt kan dele inn i fire. En av disse delene kan betraktes som en manual som dekker det meste av programvare/oppsett svært grundig, og som sannsynligvis vil kreve noen forkunnskaper for å forstå. De andre delene virker å ha mer søkelys på nye brukere uten å gå i dybden og er rettet mot den mer grunnleggende bruken som å koble seg til databaser og publisere data. En annen del går igjennom hvordan man gjennomfører mer avanserte prosesser med publisering av spesifikke objekter og kan sies å være mer avansert, men dekker allikevel grunnleggende operasjoner i programvaren, og er antagelig rettet mot nye brukere. Den siste av disse fire delene omhandler installasjon og bruk av ulike utvidelsesmoduler som har egne instrukser, forskjellig fra en ren GeoServer-installasjon. Siden utvikler-/brukermiljøet har en direkte mulighet til å påvirke dokumentasjonen er det gjerne skrevet for å favne om med litt bredde blant ulike type brukere og brukstilfeller, med forholdsvis generell tematikk enkelte steder. En observasjon er at språk og struktur ikke oppleves gjennomgående konformt, det kommer forholdsvis tydelig fram at enkelte deler av dokumentasjonen er forskjellig avhengig av hvem som har skrevet den.

Plattformen pygeoapi har en dokumentasjon som tar for seg en veldig teknisk beskrivelse av programvarens funksjonalitet og virkemåte. Den tar utgangspunkt i at man som bruker har en god teknisk forståelse av ulike datatyper og serverprogramvare. En observasjon knyttet til dette er at dokumentasjonen har en tendens til å beskrive minimalt av informasjon rundt enkelte detaljer, dette er var spesielt tydelig i de delene som omhandlet OpenAPI og

generering av et API-dokument. Dette kan føre til forvirring om man ikke er kjent med disse konseptene, eller hvordan man jobber med denne typen programvare.

Det er lagt opp til at programvaren er veldig fleksibel i hvordan man ønsker utseende og hvordan man legge inn data med koblinger til metadata. Dokumentasjonen har spesielt et område som er svært detaljert og viser flere eksempler enn resten, delen *Publishing data*, som dekker hvilke datakilder man kan publisere. Her ligger det også en del informasjon om hvilke muligheter som gis etter valg av datakilde, og som kan være med på å påvirke hvilke valg man ønsker å ta. Det er også her mange eksempler på hvordan man kan konfigurere tjenestene. Samlet sett er dokumentasjon veldig teknisk med lite eksempler. For en person som ikke har god kjennskap til programvare vil det være tidvis vanskelig å løse eventuelle utfordringer som kan oppstå, og den oppleves lite brukervennlig på enkelte områder. Man blir da veldig avhengig av eventuell egenkunnskap og tredjepartsressurser. Dokumentasjonen virker ellers konform i framstillingen. Bruker-/utviklermiljøet har mulighet til å foreslå

5.2.5 Sammenstilling

Alt i alt har de tre plattformene ulike egenskaper som kan vektlegges forskjellig ut ifra en brukers ferdigheter, erfaring og formål med tjenestene som skal settes opp. I funksjonalitets-tabellen er det presentert seks plattform-egenskaper som kan danne et overordnet bilde knyttet til dette. Tabellen gir en oversikt over plattformenes støtte for de aktuelle egenskapene, og er tiltenkt å være et supplement til et plattformvalg for ulike brukstilfeller. De fleste av disse egenskapene er drøftet gjennom ulike deler av oppgaven, men muligheten for dynamisk kartvisning av tjenestens data inngår også som en tabellegenskap, og ble ikke vektlagt så mye i diskusjonen eller resultatene ellers. Ved valg av plattform kan dette være en viktig funksjon for flere, og kan bidra til en bedre opplevelse for de som skal bruke tjenestene og gi en forståelse av tjenesteinnhold. Med pygeoapi er dette bygget inn i selve grensesnittet til tjenestene, og vises som en del av objektsamlingen i HTML-format, som vist i figur 2 vedlegg C. På ArcGIS Server er dette i utgangspunktet ikke en funksjon, men det er mulig å få inspisert dataene visuelt i et ArcGIS Online-lag, som det ligger lenker til på Service Directory-siden til tjenesten, og opprettes automatisk ved tjenesteoppsett. Hvordan dette ser ut i praksis framkommer av figur 12 og 13 i vedlegg B. GeoServer har ingen direkte kartvisingsfunksjon i sitt grensesnitt, og heller ikke lenker til alternative metoder for dette.

ArcGIS Server er en plattform som tilbyr gode løsninger og funksjonaliteter, har en enkel og brukervennlig dokumentasjon, og et grensesnitt som kan gjøre den tilgjengelig for mange med ulike i ferdighetsnivåer. Det er derimot viktig å merke seg at programvaren kommer med en kostnad. For å få tilgang til både ArcGIS Pro og Server må man ha lisen til pakken *GIS Professional Basic* i ArcGIS Enterprise, som ifølge TrustRadius (u.å.) har en kostnad på $ 2750 pr. år pr. lisens. Denne pakken inkluderer riktignok en del annen programvare, verktøy og funksjonalitet, samt brukerstøtte, men kostnaden kan allikevel være en barriere for mange. Den vil nok i større grad være rettet mot et profesjonelt marked, og muligens firmaer hvor GIS/geomatikk ikke er en del av den daglige driften, og enkelt og brukervennlig grensesnitt vil være fordelaktig.

Pr. i dag kan GeoServer betraktes som en god løsning for de aller fleste, er relativt enkel å installere, og selv om gruppen ikke fikk til dette med den offisielle veilederen, var det enkelt å finne og buke tredjepartsveiledere. Den er også ganske enkel å bruke, har en åpen kildekode, og bruker-/utviklersamfunnet er aktivt og veletablert. Den kan også betraktes som en stabil og pålitelig plattform, blant annet grunnet antallet bidragsytere. Brukerne kan også dra nytte av diskusjonsforum, dokumentasjonsressurser, opplæringsmateriell og tilpassede utvidelser som springer ut av bruker-/utviklermiljøet.

pygeoapi har gode tilpasningsmulighet og har en mer spisset løsning for en Features-tjeneste sammenlignet med ArcGIS Server og GeoServer. Den har imidlertid en bratt læringskurve, og kan være krevende å bruke for personer uten erfaring med GIS/geomatikk, og spesielt informatikk. Plattformen har et mindre bruker-/utviklersamfunn sammenlignet med de andre plattformene, og dette påvirker tilgjengeligheten og kvalitet på brukerstøtte og dokumentasjon. Dette kan igjen være en av grunnene til at det er færre brukere av pygeoapi, men for de som har ferdigheter til å utnytte plattformens fulle potensial kan den levere avanserte og framtidsrettede tjenester.

5.3 Metode

Arbeidet med denne oppgaven begynte først rundt semesterstart i midten av januar 2023. På det tidspunktet var kun et fåtall av OGC API-standardene lanserte; del 1 og 2 av OGC API-Features (publisert 11. mai 2022), samt del 1 av OGC API-Tiles (publisert 10. november 2022); tilsvaret til WMTS i WxS-familien. Som nevnt innledningsvis, ble oppgaven likevel avgrenset til å kun sammenligne WFS og Features, for å tilpasse arbeidsmengden til den satte

tidsrammen for oppgaven. WFS og Features ble valgt fremfor WMTS og Tiles fordi vi ønsket å teste ut tjenester knyttet til overføring av vektordata, og ikke kartbilder.

Det blir i innledningen påpekt at et undersøkende arbeid knyttet til denne oppgavens tematikk kan bidra til oppnåelse av FNs bærekrafts mål nummer 9. Oppgaven har presentert et utvalg plattformer som kan brukes til oppsett av ny generasjon standarder, som i større grad gjør geodata tilgjengelig og konformt med hvordan data blir utvekslet ellers over internett. Dette samsvarer spesielt godt med delmål 9.c, som har som mål om å «Øke tilgangen til informasjons- og kommunikasjonsteknologi betydelig […]» (FN-sambandet, 2023). I lys av dette kan en dypere forståelse av de mulighetene en overgang til Features gir, bidra til å fremme mer effektiv bruk av geodata og informasjonsteknologi i tråd med FNs målsetninger.

Når det gjelder valg av hvilke plattformer som skulle inkluderes i oppgaven, ønsket vi opprinnelig å ha med MapServer, da dette er en plattform som vi har benyttet oss av tidligere, og som kan sies å ha et relativt stort bruker-/utviklermiljø med god dokumentasjon. Selv om de seneste versjonene til MapServer skal ha støtte for å sette opp tjenester i henhold til Features (fra og med versjon 8.0.0), dukket det opp utfordringer som gjorde at vi ikke lyktes å sette opp tjenester i henhold til Features. En sentral utfordring var at den gjeldende versjonen av MapServer – til forskjell fra tidligere versjoner – kun var tilgjengelig gjennom en oppbygging av binærfiler, noe som var en ukjent metode for oss, og som vi ikke lyktes med å få til. Det er også verdt å nevne at ingen av de foreløpig lanserte testtjenestene for OGC API på Geonorge (2023) har benyttet MapServer, noe som gir inntrykk av at også andre har hatt problemer med å sette opp på MapServer-plattformen.

Videre når det gjelder valg av metode og fremgangsmåte, kunne oppgaven med fordel ha inkludert mer praktisk uttesting av forskjellen mellom WFS og Features, i tillegg til den hovedsakelig teoretiske sammenligningen som foreligger nå. Alternativt kunne man ha testet ut tjenestene ved hjelp av programvarer slik som MapBox eller OpenLayers.

I forhold til bruk av begrepene «API» og «Tjeneste», var det utfordrende å finne entydige definisjoner på dem begge, da begrepene noen ganger ble brukt om hverandre i litteraturen, samtidig som noe litteratur viste til enkelte forskjeller mellom de to. Det skal riktignok sies at selv på Kartverket sine nettsider blir det gjort forskjell mellom tjenester og API-er, som peker på at det kan være verdt å holde begrepene adskilt, slik som gjort i oppgaven.

Til tross for overnevnte rom for forbedringer, gjør til oppgaven likevel forsøk på å bli kjent med OGC API-ene, både ved å teste ut oppsetting av tjenester i henhold til Features-standarden, samt ved å sammenligne den opp mot WFS.

Til framtidig arbeid ville det blant annet vært interessant å ha inkludert både flere ulike plattformer i undersøkelsen av oppsett av OGC API Features, samt å ha undersøkt oppsett av flere andre standarder, slik som Tiles og Maps, og vurdert de opp mot sine respektive WxS-standarder, henholdsvis WMTS og WMS.

6 Konklusjon

Innledningsvis blir det vist til at mye av de dataene som blir samlet inn i dag, har en plassering tilknyttet til seg, og derfor kan sees på som geodata. Dette kan i seg selv være et argument for at standardene som i stor grad brukes til å dele og distribuere geografiske data på internett, bør følge de samme prinsippene for hvordan data ellers deles på internett.

Standardisering handler grunnleggende sett om at man ønsker at majoriteten av tilbydere av en tjeneste eller et produkt skal være samstemte i måten de bygger opp produktet eller tjenesten sin på. Med lanseringen av OGC API-Features – og de resterende standardene i OGC API-familien – forsøker OGC å tette gapet mellom hvordan geodata og andre data blir distribuert og delt på internett. Ved å basere seg på teknologi og standarder som allerede er i utbredt bruk blant moderne web API-er – slik som REST, GeoJSON, og Open API-spesifikasjonen – vil de nye standardene bidra til at geodata blir mer tilgjengelig. Dataene blir lettere å oppdage via tradisjonelle søkemotorer – som er et viktig startpunkt for de som letter etter informasjon på internett – i tillegg til at tjenestene som distribuerer geodata kan bli lettere å ta i bruk for personer uten geomatikkfaglig bakgrunn også. Det at standardene også blir beskrevet ved Open API-spesifikasjonen vil øke graden av interoperabilitet som standarden har med annen programvare, noe som kan forenkle prosessen med å opprette API-er i henhold til Features.

Sett bort i fra at hovedfunksjonen har forblitt den samme i overgangen fra WFS til OGC API-Features, er det ellers grunnleggende forskjeller i hvordan de er bygd opp, og i måten de utnytter HTTP for å overføre vektordata. Med hensyn på tjenesteoppsett, har det vist seg å være relativ liten forskjell i framgangsmåten for oppsett av WFS og Features, i hvert fall hos de undersøkte plattformene ArcGIS Server og GeoServer. Ved å ta utgangspunkt i disse to plattformene kan det tyde på at overgangen til OGC API-er ikke vil påvirke geomatikeren i særlig grad når det kommer til selve oppsettet av tjenesten.

De store forskjellene mellom Features og OGC API kommer hovedsakelig til uttrykk for brukeren av tjenesten; men som geomatiker vil man også hente inn data fra eksterne kilder, og dermed måtte interagere med tjenester fra brukersiden også. Til forskjell fra WFS vil man ved interaksjon med et Features API, kunne møte datasettet via en landingsside, som har en

oppbygging kjent fra nettsider ellers, og som dermed gjør interaksjonen med et ukjent datasett mer brukervennlig sammenlignet med WFS-tjenestene man bruker i dag.

Hvilken plattform som er mest gunstig til oppsett av et Features-API, vil variere ut ifra om man skal velge plattform etter hvor enkel installasjon og tjenesteoppsett er, eller om man verdsetter et aktiv bruker/utviklermiljø og detaljert dokumentasjon. Det vil også være et aspekt knyttet til økonomi og interoperabilitet med andre programvarer, og følgelig om man ønsker å benytte seg av proprietære eller åpne alternativer.

Det kan hevdes at pygeoapi virker å ha en noe mer framtidsrettet tankegang bak eget konsept, ved at de ikke støtter gamle standarder. Dersom deres metodikk blir gjeldende i framtiden kan det tyde på at geomatikeren i større grad bør gjøre seg kjent med konsepter som tidligere har vært mer assosiert med informatikk, og at overgangen til OGC API-er vil understreke viktigheten av å utvide kompetanse utover mer tradisjonelle geomatikk-ferdigheter. Dette kan innebefatte en grunnleggende forståelse av REST API-er, deres virkemåte og struktur. Å opparbeide seg slike ferdigheter kan også by på muligheter i en større sammenheng. Gapet mellom GIS/geomatikk og den generelle IT-bransjen kan reduseres ved at man kan samarbeide tettere med et felles rammeverk – i tillegg kan det antas at geomatikk-feltet tettere integreres innenfor et utvidet IT-begrep. I sum kan man si, at mens overgangen til OGC API-er kanskje ikke vil ha en betydelig innvirkning på arbeidsflyten til de som allerede bruker GeoServer eller ArcGIS Server fra før, vil det kunne kreve en kompetanseutvidelse for å dra nytte av mer framtidsrettet teknologi som pygeoapi og en mer REST API-orientert tilværelse.

Til framtidig arbeid ville det blant annet vært interessant å ha inkludert både flere ulike plattformer i undersøkelsen av oppsett av Features, samt å ha undersøkt oppsett av flere andre standarder, slik som Tiles og Maps, og vurdert de opp mot sine respektive WxS-standarder. Et annet aspekt som ikke videre har blitt diskutert her– men som er et interessant og verdifullt tema for videre arbeid – er hvordan implementeringen av disse standardene vil påvirke den geografiske infrastrukturen, hvor de gamle standardene allerede er godt etablerte. Ikke minst kunne det vært interessant med en videre diskusjon om hvordan disse nye standardene kan hjelpe Norge med å nå visjonen om å være ledende innen bruk av geodata, jamfør geodatastrategien (Kommunal- og moderniseringsdepartementet, 2018).

Litteraturliste

ArcGIS Developer (2022) *Using the Services Directory*. Tilgjengelig fra: https://developers.arcgis.com/rest/services-reference/enterprise/using-the-services-directory.htm (Hentet: 15. mai 2023).

Ben-Kiki, O., Evans, C. og Net, I. (2021) *YAML Ain't Markup Language (YAML™) version 1.2*. Tilgjengelig fra: https://yaml.org/spec/1.2.2/ (Hentet: 15. mai 2023).

Blanc, N. *et al.* (2022) OGC API state of play–a practical testbed for the national spatial data infrastructure in Switzerland, *The international archives of the photogrammetry, remote sensing and spatial information sciences*, 48, s. 59-65.

Bocher, E. og Neteler, M. (2012) *Geospatial free and open source software in the 21st century*. Springer.

Brunsdon, C. og Comber, A. (2021) Opening practice: supporting reproducibility and critical spatial data science, *Journal of geographical systems*, 23(4), s. 477-496. https://doi.org/10.1007/s10109-020-00334-2

Bårdgård, T. (2022) *Åpne og proprietære standarder*. Tilgjengelig fra: https://ndla.no/article/23623 (Hentet: 26. mars 2023).

Cerami, E. (2002) *Web services essentials*. Beijing: O'Reilly.

Chappell, D. A. og Jewell, T. (2002) *Java Web services*. Sebastopol, Calif: O'Reilly.

Chen, X. *et al.* (2017) Restful API Architecture Based on Laravel Framework, *J. Phys.: Conf. Ser*, 910(1), s. 12016. https://doi.org/10.1088/1742-6596/910/1/012016

Coetzee, S. *et al.* (2020) Open geospatial software and data: A review of the current state and a perspective into the future, *ISPRS International Journal of Geo-Information*, 9(2), s. 90.

Davis Jr., C. A. og Lacerda Alves, L. (2008) Web Services, Geospatial, i Shekhar, S. og Xiong, H. (red.) *Encyclopedia of GIS.* Boston, MA: Springer US, s. 1270-1273.

de Smith, M. J., Goodchild, M. F. og Longley, P. A. (2018) *Geospatial Analysis: A Comprehensive Guide to Principles Techniques and Software Tools*. The Winchelsea Press.

Dvergsdal, H. og Mæhlum, L. (2021) SOAP *Store Norske Leksikon.* Tilgjengelig fra: https://snl.no/SOAP (Hentet: 15. mai 2023).

Ellingwood, J. og Drake, M. (2018) *How To Install and Use PostgreSQL on Ubuntu*. Tilgjengelig fra: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-18-04 (Hentet: 13. april 2023).

Esri (2017) *ArcGIS Enterprise: Managing ArcGIS Server*. Tilgjengelig fra: https://www.youtube.com/watch?v=J6M6vkvVbNA&ab_channel=EsriEvents (Hentet: 10. mars 2023).

Esri (2022a) *OGC support in ArcGIS Enterprise*. Tilgjengelig fra: https://enterprise.arcgis.com/en/server/latest/publish-services/linux/ogc-support-in-arcgis-server.htm (Hentet: 3. mars 2023).

Esri (2022b) *Install ArcGIS Server (Linux)*. Tilgjengelig fra: https://enterprise.arcgis.com/en/server/latest/install/linux/install-arcgis-server-on-one-machine.htm (Hentet: 15. mai 2023).

Esri (2022c) *Inside an ArcGIS Server Site*. Tilgjengelig fra: https://enterprise.arcgis.com/en/server/latest/administer/linux/inside-an-arcgis-server-site.htm (Hentet: 15. mai 2023).

Esri (u.å.-a) *What is ArcGIS Server?* Tilgjengelig fra: https://enterprise.arcgis.com/en/server/latest/get-started/linux/what-is-arcgis-for-server-.htm (Hentet: 03 mai 2023).

Esri (u.å.-b) *Make data accessible to ArcGIS Server*. Tilgjengelig fra: https://enterprise.arcgis.com/en/server/latest/manage-data/windows/making-your-data-accessible-to-arcgis-server.htm (Hentet: 15. mai 2023).

Esri (u.å.-c) *Publish service definition files*. Tilgjengelig fra: https://enterprise.arcgis.com/en/server/latest/publish-services/windows/about-service-definition-files.htm (Hentet: 15. mai 2023).

Esri (u.å.-d) *Save a service definition for a map service*. Tilgjengelig fra: https://pro.arcgis.com/en/pro-app/latest/help/sharing/overview/save-a-service-definition-for-a-map-service.htm (Hentet: 15. mai 2023).

FN-sambandet (2023) *Industri, innovasjon og infrastruktur*. Tilgjengelig fra: https://www.fn.no/om-fn/fns-baerekraftsmaal/industri-innovasjon-og-infrastruktur (Hentet: 15. mai 2023).

Geodata (u.å.) *ArcGIS Enterprise*. Tilgjengelig fra: https://www.geodata.no/produkter-og-tjenester/arcgis-fra-esri?originId=430 (Hentet: 15. mai 2023).

Geodataforskriften (2012) *Forskrift om infrastruktur for geografisk informasjon*. Tilgjengelig fra: https://lovdata.no/forskrift/2012-08-08-797 (Hentet: 15. mai 2023).

Geonorge (u.å.) *OGC APIer testtjenester*. Tilgjengelig fra: https://www.geonorge.no/verktoy/APIer-og-grensesnitt/ogc-api-er/ogc-apier-testtjenester/ (Hentet: 21. mars 2023).

GeoServer (2020) *History*. Tilgjengelig fra: https://docs.geoserver.org/stable/en/user/introduction/history.html (Hentet: 2. mars 2023).

GeoServer (u.å.-a) *OGC API Extension*. Tilgjengelig fra: https://docs.geoserver.org/latest/en/user/community/ogc-api/index.html (Hentet: 15. mai 2023).

GeoServer (u.å.-b) *What is GeoServer?* Tilgjengelig fra: https://geoserver.org/about/ (Hentet: 2. mars 2023).

Gramstad, T. (2021) åpen kildekode *Store Norske Leksikon.* Tilgjengelig fra: https://snl.no/%C3%A5pen_kildekode (Hentet: 14. mai 2023).

Hygraph (2022) *Web Service vs. API: what are they, and how are they different?* Tilgjengelig fra: https://hygraph.com/blog/web-service-vs-api (Hentet: 20. april 2023).

Iacovella, S. og Youngblood, B. (2013) *GeoServer Beginner's Guide*. Birmingham, UK.: Packt Publishing Ltd.

IBM (2022) *Web services approach to a service-oriented architecture*. Tilgjengelig fra: https://www.ibm.com/docs/en/was/9.0.5?topic=architecture-web-services-approach-service-oriented (Hentet: 15. mai 2023).

ISO (u.å.) *About us*. Tilgjengelig fra: https://www.iso.org/about-us.html (Hentet: 18. februar 2023).

Jacobson, D., Brail, G. og Woods, D. (2011) *APIs: A Strategy Guide: Creating Channels with Application Programming Interfaces*. O'Reilly Media.

Jethva, H. (2022) *How to Install GeoServer Server on Ubuntu*. Tilgjengelig fra: https://cloudinfrastructureservices.co.uk/how-to-install-geoserver-server-on-ubuntu-20-04-tutorial-step-by-step/ (Hentet: 8. mars 2023).

Jin, B., Shevat, A. og Sahni, S. (2018) *Designing web APIs : building APIs that developers love*. Sebastopol, Calif: O'Reilly.

Juviler, J. (2022) *Web Service vs. API, Explained*. Tilgjengelig fra: https://blog.hubspot.com/website/web-services-vs-api (Hentet: 16. april 2023).

Karlsson, S., Čaušević, A. og Sundmark, D. (2020) QuickREST: Property-based test generation of OpenAPI-described RESTful APIs, i *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*. IEEE, s. 131-141.

Kommunal- og moderniseringsdepartementet (2018) *Nasjonal geodatastrategi fram mot 2025 - alt skjer et sted*. Tilgjengelig fra: https://www.regjeringen.no/no/dokumenter/nasjonal-geodatastrategi---alt-skjer-et-sted/id2617560/ (Hentet: 14. mai 2023).

Kotsev, A. *et al.* (2020) From spatial data infrastructures to data spaces—A technological perspective on the evolution of European SDIs, *ISPRS International Journal of Geo-Information*, 9(3), s. 176.

Kralidis, A. T. (2008) Geospatial Open Source and Open Standards Convergences, i Hall, G. B. og Leahy, M. G. (red.) *Open Source Approaches in Spatial Data Handling*. Berlin, Heidelberg: Springer Berlin Heidelberg, s. 1-20.

Kralidis, T. (2022) *Implementing OGC APIs using Elasticsearch and pygeoapi*. Tilgjengelig fra: https://community-conference.elastic.co/session/306884 (Hentet: 2. mai 2023).

Krechmer, K. (2005) The meaning of open standards, i *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*. IEEE, s. 204b-204b.

Kraak, M.-J. og Ormeling, F. (2020) *Cartography: visualization of Geospatial Data*. London: CRC Press.

Lange, K. (2016) *The Little Book on Rest Services*. Uten forlag. Tilgjengelig fra: https://www.kennethlange.com/books/The-Little-Book-on-REST-Services.pdf.

Mainas, N., Petrakis, E. G. og Sotiriadis, S. (2017) Semantically enriched open API service descriptions in the cloud, i *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, s. 66-69

Masse, M. (2011) *REST API design rulebook: designing consistent RESTful web service interfaces*. " O'Reilly Media, Inc.".

Miller, M. (2020) *What is GeoServer?* Tilgjengelig fra: https://www.youtube.com/watch?v=OWo6IZmOaBk&t=30s&ab_channel=MikeMiller (Hentet: 28. februar 2023).

Mitchell, T. (2005) *Web mapping illustrated*. Sebastopol, Cal: O'Reilly.

Moreno-Sanchez, R. (2012) Free and Open Source Software for Geospatial Applications (FOSS4G): A mature alternative in the geospatial technologies arena (b. 16, s. 81-88): Wiley Online Library.

Newcomer, E. (2002) *Understanding Web Services: XML, Wsdl, Soap, and UDDI*. Addison-Wesley Professional.

Norge digitalt (2019) *Veileder for Web Feature Service (WFS)*. Tilgjengelig fra: https://register.geonorge.no/subregister/versjoner/nasjonale-standarder-og-veiledere/kartverket/veiledere/kartverket/wfs-veileder (Hentet: 15. mai 2023).

Nätt, T. H. og Rossen, E. (2022) API *Store Norske Leksikon.* Tilgjengelig fra: https://snl.no/API (Hentet: 14. mai 2023).

Nätt, T. H. (2023) fri programvare *Store Norske Leksikon.* Tilgjengelig fra: https://snl.no/fri_programvare (Hentet: 14. mai 2023).

OGC (2014) *OGC Web Feature Service 2.0 Interface Standard – With Corrigendum*. Tilgjengelig fra: http://docs.opengeospatial.org/is/09-025r2/09-025r2.html (Hentet: 15. mai 2023).

OGC (2016,) *OGC Testbed 11 REST Interface Engineering Report*. Tilgjengelig fra: http://www.opengis.net/doc/PER/tb11_rest_er (Hentet: 15. mai 2023).

OGC (2017). Tilgjengelig fra: http://docs.opengeospatial.org/wp/16-019r4/16-019r4.html (Hentet: 15. mai 2023).

OGC (2022) *OGC API - Features - Part 1: Core corrigendum*. Tilgjengelig fra: https://docs.ogc.org/is/17-069r4/17-069r4.html# (Hentet: 15. mai 2023).

OGC (2023) *OGC API - Common - Part 1: Core*. Tilgjengelig fra: http://www.opengis.net/doc/is/ogcapi-common-1/1.0 (Hentet: 15. mai 2023).

OGC (u.å.-a) *OGC API - Features*. Tilgjengelig fra: https://www.ogc.org/standard/ogcapi-features/ (Hentet: 15. mai 2023).

OGC (u.å.-b) *OGC API - Features: Overview*. Tilgjengelig fra: https://ogcapi.ogc.org/features/overview.html (Hentet: 15. mai 2023).

OGC (u.å.-c) *Standards roadmap*. Tilgjengelig fra: https://www.ogc.org/standards/roadmap/ (Hentet: 15. mai 2023).

OGC (u.å.-d). Tilgjengelig fra: https://www.ogc.org/ (Hentet: 15. mai 2023).

OGC (u.å.-e) *Web Feature Service*. Tilgjengelig fra: https://www.ogc.org/standard/wfs/ (Hentet: 15. mai 2023).

OpenAPI Inititative (2021) *OpenAPI Specification v3.1.0.* Tilgjengelig fra: https://spec.openapis.org/oas/latest.html (Hentet: 05. mars 2023).

OpenAPI Inititative (2023) Why the Largest Geospatial Organization in the World Uses the OpenAPI Specification *OpenAPI blog* (b. 2023). Tilgjengelig fra: https://www.openapis.org/blog/2023/02/01/why-the-largest-geospatial-organization-in-the-world-uses-the-openapi-specification (Hentet: 30. mars 2023).

OSGeo (u.å.-a) *pygeoapi*. Tilgjengelig fra: https://www.osgeo.org/projects/pygeoapi/ (Hentet: 15. mai 2023).

OSGeo (u.å.-b) *pygeoapi overview*. Tilgjengelig fra: https://live.osgeo.org/en/overview/pygeoapi_overview.html (Hentet: 15. mai 2023).

Osman, J. (2022) *JSON vs XML*. Tilgjengelig fra: https://appmaster.io/blog/json-vs-xml (Hentet: 18. april 2023).

Patni, S. (2017) *Pro RESTful APIs : Design, Build and Integrate with REST, JSON, XML and JAX-RS*. Apress : Imprint: Apress.

pygeoapi (2023a) *pygeoapi 0.14.0 documentation*. Tilgjengelig fra: docs.pygeoapi.io/en/stable/index.html (Hentet: 15. mai 2023).

pygeoapi (2023b) *Configuration*. Tilgjengelig fra: https://docs.pygeoapi.io/en/latest/configuration.html (Hentet: 15. mai 2023).

pygeoapi (2023c) *How pygeoapi works*. Tilgjengelig fra: https://docs.pygeoapi.io/en/latest/how-pygeoapi-works.html (Hentet: 15. mai 2023).

pygeoapi (2023d) *Administration*. Tilgjengelig fra: https://docs.pygeoapi.io/en/latest/administration.html (Hentet: 10. mai 2023).

pygeoapi (2023e) *Introduction*. Tilgjengelig fra: https://docs.pygeoapi.io/en/latest/introduction.html (Hentet: 15. mai 2023).

Reini, J. (2022) *How to implement OGC API Features by using pygeoapi?* Tilgjengelig fra: https://www.youtube.com/watch?v=PMCTHZu4BxI (Hentet: 21. mars 2023).

Richardson, L. og Ruby, S. (2007) *Restful Web services*. Beijing ; Sebastopol, Calif: O'Reilly.

Richardson, L. og Amundsen, M. (2013) *RESTful Web APIs*. O'Reilly.

Sahin, K. og Gumusay, M. (2008) Service oriented architecture (SOA) based web services for geographic information systems, i *XXIst ISPRS Congress. Beijing*. Citeseer, s. 625-630.

Snell, J., Tidwell, D. og Kulchenko, P. (2002) *Programming web services with SOAP*. Beijing: O'Reilly.

Stallman, R. (2022) *Why Open Source Misses the Point of Free Software*. Tilgjengelig fra: https://www.gnu.org/philosophy/open-source-misses-the-point.html (Hentet: 15. mai 2023).

Standard Norge (2021) *Standardisering*. Tilgjengelig fra: https://www.standard.no/standardisering/ (Hentet: 19. februar 2023).

Subramanian, H. og Raj, P. (2019) *Hands-On RESTful API Design Patterns and Best Practices: Design, develop, and deploy highly adaptable, scalable, and secure RESTful web APIs*. Packt Publishing Ltd.

TrustRadius (u.å.) *ArcGIS Pricing*. Tilgjengelig fra: https://www.trustradius.com/products/arcgis/pricing (Hentet: 10. mai 2023).

UbuntuGIS (u.å.) *Packages*. Tilgjengelig fra: https://launchpad.net/~ubuntugis/+archive/ubuntu/ubuntugis-unstable/+packages?field.name_filter=pygeoapi&field.status_filter=published&field.series_filter= (Hentet: 14. februar 2023).

W3C (2004) *Web Services Architecture*. Tilgjengelig fra: https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/ (Hentet: 15. mai 2023).

W3C (u.å.-a) *JSON vs XML*. Tilgjengelig fra: https://www.w3schools.com/js/js_json_xml.asp (Hentet: 20 april 2023).

W3C (u.å.-b) *About W3C*. Tilgjengelig fra: https://www.w3.org/Consortium/ (Hentet: 15. mai 2023).

W3Schools (u.å.) *XML Tutorial*. Tilgjengelig fra: https://www.w3schools.com/xml/ (Hentet: 15. mai 2023).

Yeung, A. K. og Hall, G. B. (2007) *Spatial database systems: Design, implementation and project management*. Springer Science & Business Media.

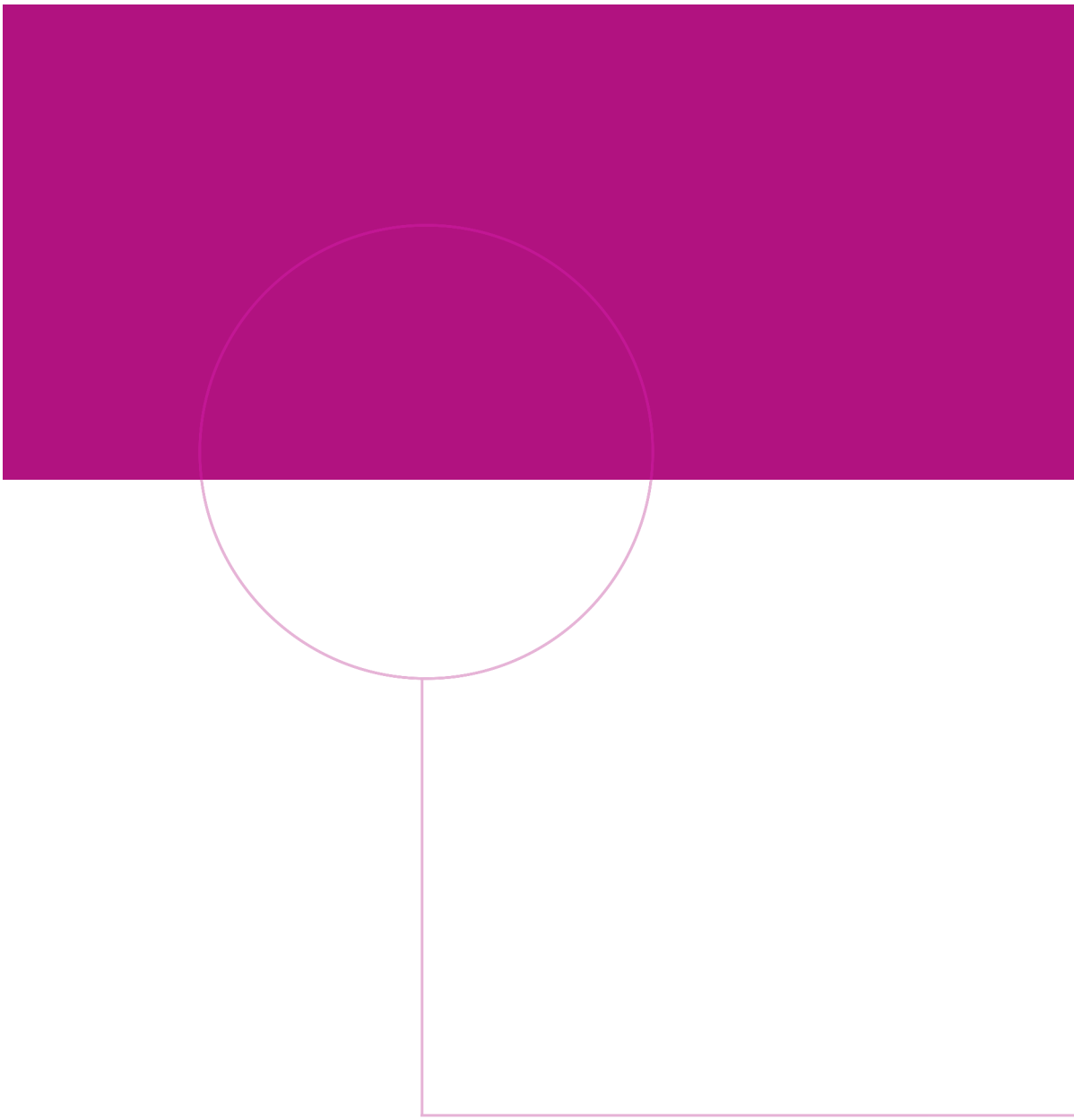
Vedleggsliste

Vedlegg A: Trinnvis fremgangsmåte for installasjon og tjenesteoppsett av GeoServer, tjenesteoppsett og eksempler på kall

Vedlegg B: Trinnvis fremgangsmåte for installasjon og tjenesteoppsett av ArcGIS Server, tjenesteoppsett og eksempler på kall

Vedlegg C: Trinnvis fremgangsmåte for installasjon av pygeoapi, tjenestetesting og eksempler på kall

Vedlegg D: Konfigurasjonsfil for pygeoapi i YAML-format



NTNU

Kunnskap for en bedre verden