

Gränssnittsstandarden "OGC API - Environmental Data Retrieval" och dess användning för nedladdning av väderdata

Hanna Lundström

Civilingenjörsutbildningen i Lantmäteri
Lunds Tekniska Högskola

Institutionen för Naturgeografi och Ekosystemvetenskap
Lunds Universitet





LUNDS UNIVERSITET
Lunds Tekniska Högskola

Gränssnittsstandarden "OGC API - Environmental Data Retrieval" och dess användning för nedladdning av väderdata

EXTM05 Masteruppsats, 30 hp
Civilingenjörsutbildningen i Lantmäteri

Hanna Lundström

Handledare:

Lars Harrie

Institutionen för Naturgeografi och Ekosystemvetenskap

21 Augusti, 2024

Opponent: Axel Andersson
Examinator: Helena Elvén Eriksson

Copyright © Hanna Lundström, LTH

Institutionen för Naturgeografi och Ekosystemvetenskaper
Lunds Universitet
Sölvegatan 12
223 62 Lund

Telefon: 046-222 30 30
Fax: 046-222 03 21
Hemsida: <http://www.nateko.lu.se>

Examensarbete i geografisk informationsteknik nr 39
Tryckt av E-tryck, E-huset, 2024

Abstract

Weather forecasts are of great significance for instance in planning flight routes, military operations, and seafare. Many organizations today openly provide, not only their forecasts, but their raw data as well. Technological development and a greater availability of open data ensures that both weather- and forecast data can be used for both private and commercial purposes. However, customized application programming interfaces (APIs) and domain-specific solutions create friction between applications and can hinder interoperability. Access to geospatial data such as weather- and forecast data can therefore be made easier with uniform standardization.

The Open Geospatial Consortium (OGC) specifically develops open geospatial data standards, including standardized APIs. One such standard is the newly introduced OGC API - Environmental Data Retrieval (EDR), which aims to facilitate access to large volumes of data such as weather and forecast data. This thesis project aims to evaluate EDR in comparison with other OGC standards for the sharing of forecast data. Additionally, an EDR solution with the purpose of retrieving subsets of raw data is designed, implemented and evaluated.

The report begins with a literature review to establish the necessary background and context for both the comparison and the practical implementation. Two case studies are executed. The first case study compares EDR and seven other OGC standards, which are deemed relevant to the aim of this thesis. The comparison is then carried out by examination of the OGC specifications, as well as using the tool ChatGPT in order to gather information. It can be concluded that EDR best meets the requirements and needs related to a weather application, but that it must be noted that EDR is a new standard and thus currently lacks enough certified implementations and tools.

The second case study, the practical implementation, results in a weather application based on EDR for retrieving subsets of raw data. The implementation is carried out using `pygeoapi`, which is a reference implementation of EDR and the only to be certified by OGC for EDR at present. The case study highlights that EDR is difficult both to evaluate and to use presently, due to the limited support for the standard at this time.

In conclusion, this thesis demonstrates that the EDR standard has great potential for greater access and sharing of multidimensional data, such as weather- and forecast data, but further development and implementation are needed.

Sammanfattning

Väderprognoser är av stor betydelse för till exempel flygrutter, militära operationer och sjöfart. Flera organisationer tillhandahåller numera inte bara sina väderprognoser öppet, utan delar även med sig av sina prognosdata för nedladdning. Teknisk utveckling och ett större utbud av öppna data möjliggör användning av väder- och prognosdata för både privata och kommersiella ändamål. Egna programmeringsgränssnitt och domänspecifika lösningar medför dock friktion mellan applikationer och försämrar interoperabilitet. Åtkomst till geospatiala data såsom väder- och prognosdata kan därmed underlättas av enhetlig standardisering.

Open Geospatial Consortium (OGC) utvecklar specifikt öppna geospatiala datastandarder inklusive standardiserade programmeringsgränssnitt. En sådan standard är den nyttillkomna OGC API - Environmental Data Retrieval (EDR), som ämnar underlätta dataåtkomst för stora volymer data såsom väder- och prognosdata. Denna studie syftar till att utvärdera EDR i jämförelse med likvärdiga OGC-standarder för datadelning av prognosdata. Därtill designas, implementeras och utvärderas en EDR-lösning för hämtning av urval av rådata.

Rapporten inleds med en litteraturstudie i syfte att bidra med nödvändig bakgrund och kontext för både jämförelse och praktisk implementering. Därefter genomförs två fallstudier. Den första fallstudien jämför EDR och sju andra OGC-standarder, bedömda som relevanta för studiens syfte. Jämförelsen verkställs genom granskning av OGC-specifikationer och med användning av verktyget ChatGPT i syfte att sammanställa information. Ur jämförelsen kan konstateras att EDR bäst uppfyller krav och behov i relation till en väderapplikation, men att hänsyn måste tas till att EDR är en ny standard och än så länge utan tillräckligt med certifierade implementeringar och verktyg.

Praktisk implementering utgör den andra fallstudien och resulterar i en väderapplikation baserad på EDR för hämtning av urval av rådata. Implementering sker med tillämpning av *pygeoapi*, som är en referensimplementation av EDR och den enda implementeringen av EDR som är certifierad av OGC just nu. Fallstudien belyser att EDR är svår att utvärdera och använda i nuläget i och med att det ännu inte finns så stort stöd för standarden.

Sammanfattningsvis visar denna studie att standarden EDR har stor potential för att underlätta delning av multidimensionella data såsom väder- och prognosdata, men att ytterligare utveckling och implementering krävs.

Förord

Denna rapport utgör resultatet av det slutliga examensarbetet inom civilingenjörsprogrammet Lantmäteri med specialisering inom geografisk informationsteknik på Lunds Tekniska Högskola, LTH. Examensarbetet omfattar 30 högskolepoäng och är gjord under våren 2024 i samarbete med T-Kartor Geospatial Group AB.

Ett antal personer är att tacka för rapportens fullbordan. Tack först och främst till min akademiska handledare **Lars Harrie**, för det stöd och den vägledning du visat i stora som små frågor gällande rapporten. Tack för att du outtröttligt rättat alla gånger jag refererat till data i singular.

Tack även till min handledare på T-Kartor, **Patrik Sylve**. Tjugo års vänskap senare fortsätter jag att följa i dina fotspår. Tack för det engagemang och den uppmuntran du visat mig och denna studie. Vidare riktas ett stort tack till övriga på T-Kartor - tack för att ni alltid velat prata väder med mig i fikarummet.

För undertecknad är denna studie inte bara slutet på fem års studier, utan slutet på totalt nio år tillägnat Lund och dess studentliv. Utöver nära och kära, vänner och bekanta, vill jag därmed slutligen tacka mig själv, för att jag nu äntligen kan kalla mig själv både litteraturvetare och civilingenjör. Jag har examen (x2).

Lund 2024-08-21

Hanna Lundström

Innehållsförteckning

I Inledning	1
1 Introduktion	2
1.1 Bakgrund.....	2
1.2 Problemformulering.....	3
1.3 Syfte.....	4
1.4 Övergripande genomförande.....	4
1.5 Avgränsning.....	4
1.6 Disposition.....	5
II Teoretisk bakgrund	6
2 Teoretisk bakgrund	7
2.1 Väderdata och väderprognoser.....	7
2.2 Väderapplikation.....	10
2.3 Standarder och standardisering.....	13
2.3.1 Open Geospatial Consortium (OGC).....	15
2.3.2 OGC Web Map Service (WMS).....	15
2.3.3 OGC Web Feature Service (WFS).....	18
2.3.4 OGC Web Coverage Service (WCS).....	19
2.3.5 OGC Sensor Observation Service (SOS).....	20
2.3.6 OGC API - Environmental Data Retrieval (EDR).....	21
2.3.7 OGC API - Features (Features).....	23
2.3.8 OGC API - Records (Records).....	24
2.3.9 OGC Web Processing Service (WPS).....	25
III Fallstudier	27
3 Kravställning	28
3.1 Användarfall.....	28
3.1.1 Användarfall - Hämtning av data.....	28
3.1.2 Användarfall - Interoperabilitet mellan T-Kartors tjänster.....	29
3.2 Kravsammanställning.....	30
3.2.1 Utvecklingskrav.....	30
3.2.2 Användarkrav.....	30
4 Fallstudie 1 - Teoretisk jämförelse	32

4.1 Val av standarder.....	32
4.2 Metod.....	33
4.3 Resultat.....	35
5 Fallstudie 2 - Praktisk implementering	40
5.1 Systemarkitektur.....	40
5.2 Val av verktyg.....	41
5.2.1 Server.....	41
5.2.2 Klient - applikationsramverk.....	42
5.2.3 Klient - användargränssnitt.....	42
5.3 Implementering.....	42
5.3.1 Server.....	42
5.3.2 Klient.....	43
5.3.3 Fillagring.....	43
5.4 Utvärdering.....	43
5.5 Distribuering av kod.....	45
IV Diskussion och slutsatser	46
6 Diskussion	47
6.1 Fallstudie 1 - Teoretisk jämförelse.....	47
6.2 Fallstudie 2 - Praktisk implementering.....	49
6.3 Sammanställning och generell diskussion.....	50
7 Slutsatser	52

Ordlista

API	<i>Application Programming Interface</i>	Applikationsprogrammeringsgränssnitt
EDR	<i>OGC API - Environmental Data Retrieval</i>	OGC-standard för hämtning av geospatiala data
ECMWF	<i>European Centre for Medium-Range Weather Forecasts</i>	Europeisk samarbetsorganisation som bland annat producerar globala väderprognoser
Features	<i>OGC API - Features</i>	OGC-standard för att skapa, redigera och fråga efter geospatiala data online
GeoJSON	<i>Geographic JavaScript Object Notation</i>	JSON-baserat format för att beskriva geografiska objekt
GML	<i>Geography Markup Language</i>	XML-baserat språk för att beskriva geografiska objekt
GRIB	<i>GRIdded Binary eller General Regularly-distributed Information in Binary form</i>	Binärt dataformat i form av ett gridnät där kod kombineras med externa metadata
ISO	<i>International Organization for Standardization</i>	Internationell organisation som utvecklar och publicerar standarder
IFS	<i>Integrated Forecasting System</i>	ECMWF-modell för deras operativa väderprognoser
JSON	<i>JavaScript Object Notation</i>	Textbaserat dataformat för informationsutbyte
MET	<i>Meteorologisk institutt</i>	Norsk motsvarighet till SMHI

NetCDF	<i>Network Common Data Form</i>	Uppsättning programvarubibliotek och maskinoberoende dataformat
OGC	<i>Open Geospatial Consortium</i>	Internationellt konsortium som tar fram geodatastandarder
OWS	<i>OGC Web Services</i>	Samlingsnamn för OGC-standarderna WMS, WFS, WCS med flera
Records	<i>OGC API - Records</i>	OGC-standard för att skapa, redigera och fråga efter metadata online
SIS	<i>Svenska Institutet för Standarder</i>	Standardiseringsorganisation som representerar Sverige i europeiska och globala sammanhang
SMHI	<i>Sveriges Meteorologiska och Hydrologiska Institut</i>	Statlig expertmyndighet inom meteorologi, hydrologi, oceanografi och klimatologi
SOS	<i>OGC Sensor Observation Service</i>	OGC-standard för att hämta observationsdata från sensorer
WCS	<i>OGC Web Coverage Service</i>	OGC-standard för tillgång till multidimensionella data online
WFS	<i>OGC Web Feature Service</i>	OGC-standard för att hämta, redigera och uppdatera geospatiala data online
WMS	<i>OGC Web Map Service</i>	OGC-standard för publicering av kartor online.

WPS	<i>OGC Web Processing Service</i>	OGC-standard för standardisering av geospatiala beräkningstjänster
W3C	<i>World Wide Web Consortium</i>	Konsortium som utvecklar standarder för webben
XML	<i>Extensible Markup Language</i>	Textbaserat märkspråk för dataöverföring

Del I
Inledning

1 Introduktion

1.1 Bakgrund

Från religion och vidskepelse, till vetenskap, har önskan om att kunna förutspå vädret stått som följeslagare genom mänsklighetens historia. Vädret är ett ständigt inslag i varje människas liv - ett återkommande samtalsämne i fikarummet, ett evigt planerande av kläder efter väder. Även i större bemärkelse är vädret en viktig faktor att ta hänsyn till i fråga om till exempel flygrutter, militära operationer eller sjöfart. Att kunna beakta och förbereda utifrån vädrets förutsättningar, framförallt när det rör sig om kraftigare väderlek, är en stor anledning till att väderprognoser fortsatt är till sådan nytta idag.

Avgörande för tillförlitliga väderprognoser är tillgång till ett konstant flöde av väderobservationer [SMHI, 2021a]. Internationellt utbyte av meteorologiska och oceanografiska data har pågått sedan mitten av 1800-talet och understryker vikten av geospatiala standarder för att uppnå interoperabilitet. Friktion mellan domänspecifika lösningar ökar i takt med att teknisk utveckling och digitalisering möjliggör för vem som helst att utnyttja geospatial programvara för egna ändamål [Seib et al., 2018]. Statliga myndigheter såsom norska Meteorologisk Institutt (MET) och svenska Sveriges Meteorologiska och Hydrologiska Institut (SMHI) erbjuder nedladdning av väderdata utan kostnad samt tillhandahåller egna programmeringsgränssnitt, det vill säga API:er (engelskans *Application Programming Interface*), för delning av prognosdata. Ett gemensamt, och därmed standardiserat, API hade dock möjliggjort ett mer effektivt samarbete och utbyte mellan applikationer i fråga om åtkomst till data.

Väderdata är ofta multidimensionella - latitud och longitud, tid och höjd - och med flera möjliga parametrar såsom vind, tryck och temperatur. På grund av sin komplexitet och volym är väderdata svåra att hantera och det är inte alltid implementering av standarder fungerar felfritt. Termen väderdata innefattar i denna studie både meteorologiska och oceanografiska data. Meteorologi och oceanografi är tätt sammanlänkade i studier av jordens klimat. För denna studie anses därmed väderdata vara en tillämpbar, generell term för multidimensionella, geospatiala data av meteorologisk och oceanografisk karaktär. Behandlade väderdata benämns sedan prognosdata.

Open Geospatial Consortium (OGC) är den ledande organisationen internationellt för utvecklingen av specifikt geospatiala datastandarder inklusive standardiserade API:er. Dessa standarder syftar till att förenkla utbytet av geospatial information och på så vis tillgängliggöra geospatiala data på internet [OGC, 2024a]. Flera av de klassiska OGC-standarderna är uppåt 20 år gamla och är idag inte de mest tillämpbara gränssnitten för väder- och prognosdata. Standarder som ändå allmänt godtagits för användning till väder- och

prognosdata är till exempel OGC:s Web Map Service (WMS), Web Feature Service (WFS), Web Coverage Service (WCS) och Sensor Observation Service (SOS) [Schubert & Bamberger, 2019].

OGC jobbar aktivt med att uppdatera och förnya standarder i enlighet med rådande webbarkitektur. I syfte att bygga vidare från de traditionella standarderna nämnda ovan skapades OGC API-standarder för ett modernare tillvägagångssätt. 2021 tillkom standarden OGC API - Environmental Data Retrieval (EDR), som är en standard ämnad att underlätta åtkomst till ett större spektrum av data genom ett enhetligt, väldefinierat gränssnitt samt möjliggöra datareduktion så att användare endast behöver interagera med de specifika data som efterfrågas. Standarden är tänkt att vara tillämpbar för väderprognoser eller andra fall med stora volymer data, där en användare endast är intresserad av ett specifikt urval [OGC, 2023a].

Företaget T-Kartor Geospatial Group AB (hädanefter T-Kartor) tillhandahåller en väderapplikation som visualiserar olika typer av prognosdata. Prognosdata genereras av en WPS-server (implementerad i enlighet med OGC-standarderna Web Processing Service) och visualiseras av en WMS-server. WPS-servern har förprogrammerade processer för att omvandla rådata till bilder baserat på ingående parametrar. Parametrarna väljs av användaren och kan till exempel vara intervaller av tid, område, prognosmodell etc. Användare av väderapplikationen är bland annat det norska försvaret. För närvarande erbjuds användaren endast att ladda ned kartbilder och inte rådata direkt i tjänsten, men för att T-Kartor ska kunna erbjuda hämtning av rådata i framtiden krävs ett API med rätt funktionalitet. Denna studie ämnar således utvärdera EDR i relation till behov och krav för en sådan väderapplikation.

1.2 Problemformulering

Väder- och prognosdata är i sin storlek och omfattning svåra att hantera, kopiera eller distribuera inom rimliga tidsramar. För att underlätta delning och hämtning av data krävs ett enkelt och standardiserat API. EDR är en nyttillkommen standard som ska bidra med ett lättförståeligt gränssnitt för hämtning av delmängder av geospatiala data, något som efterfrågas i analys av väderprognoser.

Eftersom EDR-standarderna först godkändes som standard av OGC 2021 är den ännu inte fullt etablerad och har inte hunnit implementeras så mycket. Det är därmed skralt med litteratur och projekt att jämföra med och svårt att avgöra hur implementering av standarderna presterar i verkligheten. Det är av intresse att undersöka hur denna nya standard står sig gentemot tidigare, väletablerade standarder. Av specifikt intresse är att utvärdera EDR för om standarderna är tillämpbara för väderapplikationer.

1.3 Syfte

Det övergripande syftet med denna studie är att teoretiskt utvärdera API-standarden EDR samt praktiskt implementera en väderapplikation baserat på EDR. Utvärdering sker i jämförelse tillsammans med andra, likvärdiga OGC-standarder. Mer specifikt har studien som syfte att:

- Utvärdera standarden EDR tillsammans med relevanta OGC-standarder för datadelning av prognosdata. Med prognosdata åsyftas processade meteorologiska och oceanografiska data, det vill säga rasterdata av temperatur, vind mm.
- Designa, implementera och utvärdera en EDR-lösning för att hämta ut urval av rådata från ovanstående prognosdata.

1.4 Övergripande genomförande

Studien är uppdelad i en teoretisk och en praktisk del. Den teoretiska delen omfattar ett bakgrundskapitel som syftar till att grundläggande redogöra för definitioner kopplade till studiens praktiska del, däribland väder- och prognosdata samt uppbyggnad av en väderapplikation. Den teoretiska bakgrunden beskriver även allmänt vad standardisering innebär för geospatiala data och några av de gränssnittsstandarder som OGC tillhandahåller.

Den praktiska delen består av två fallstudier, där den ena jämför OGC-standarder utifrån en kravställning med utgångspunkt i implementering av en väderapplikation. Vid genomförande av den första fallstudien används AI-verktyget ChatGPT i syfte att sammanställa information. Informationen kontrolleras gentemot OGC-specifikationer samt andra tillförlitliga källor. Den andra fallstudien implementerar en EDR-lösning för en väderapplikation, grundad i ovanstående kravställning. Resultatet av fallstudierna utvärderas och diskuteras sedan gemensamt.

1.5 Avgränsning

Denna studie utgår från OGC-standarden EDR och studiens teoretiska bakgrund samt fallstudier är centrerade till denna standard. Premissen för studien är val av API för en väderapplikation och därmed är den jämförande fallstudien avgränsad så att EDR jämförs tillsammans med relevanta, liknande OGC-standarder för detta användningsområde. I syfte att avgränsa ytterligare undersöks endast OGC-standarder, med motiveringen att OGC är den ledande organisationen för geospatiala datastandarder och deras standarder mest framträdande inom ämnet. Jämförelsen är även baserad på en kravställning som framtas utifrån krav och behov i relation till användning av en väderapplikation.

Den praktiska implementeringen utgår också ifrån kravställningen, men tar enbart hänsyn till EDR och av de andra jämförda standarderna görs ingen implementering.

1.6 Disposition

Denna studie utgörs av fyra delar. Den första delen, *Inledning*, introducerar studiens grundläggande premisser med bakgrund, problemformulering och syfte, genomförande, avgränsning samt förevarande avsnitt. I följande del, *Teoretisk bakgrund*, presenteras teoretisk bakgrund och förutsättningar för fallstudierna. Kapitlet tar upp väderdata och generella väderapplikationer, samt standardisering och ett antal OGC-standarder. Det praktiska arbetet som genomförts beskrivs sedan i rapportens tredje del, *Fallstudier*, där även resultat presenteras. Slutligen avslutas studien med dess fjärde del, *Diskussion och slutsatser*, som knyter resultatet tillbaka till studiens syfte och problemformulering.

Del II
Teoretisk bakgrund

2 Teoretisk bakgrund

Detta kapitel ämnar att kortfattat presentera nödvändig bakgrund och kontext inför jämförelseanalysen som genomförs i studiens första fallstudie. Vidare introduceras förutsättningar för den praktiska implementeringen i studiens andra fallstudie. Kapitlet innefattar en generell beskrivning av väderdata och väderprognoser, en väderapplikations uppbyggnad, standardisering och ett antal standarder av OGC.

2.1 Väderdata och väderprognoser

Väderprognoser är beroende av stora mängder multidimensionella data, specifikt observationer av väder och hav som samlas in kontinuerligt från mätstationer, flygplan, fartyg och satelliter. Observationer av väder kan till exempel vara av nederbörd, luftfuktighet, lufttryck, temperatur, vindhastighet och vindriktning. Havsobservationer kan till exempel vara av våghöjd och havstemperatur. Data av detta slag kallas väderdata.

Utifrån insamlade väderdata kan sedan prognoser göras genom olika modeller. Internationellt utbyte av meteorologiska och oceanografiska data bidrar till globala väderprognoser. Både svenska SMHI och norska MET använder sig av globala väderprognoser från den europeiska samarbetsorganisationen European Centre for Medium-Range Weather Forecasts (ECMWF). ECMWF har sedan 1979, med hjälp av medlemsländernas samlade resurser, producerat globala väderprognoser - idag bland annat baserade på ECMWF Integrated Forecasting System (IFS). För mer information kring denna modell hänvisas till ECMWF [ECMWF, u.å.-a; u.å.-b].

Från ECMWF får de anslutna medlemsorganisationerna dagligen tillgång till globala väderprognoser upp till 10 dygn framåt. För mer lokala väderprognoser, upp till 12 timmar framåt, används av SMHI en så kallad mesoskalig analysmodell (MESAN). MESAN kartlägger mer detaljerat den aktuella vädersituationen i Sverige och väger sedan samman alla tillgängliga data till en väderkarta. En ytterligare modell som används för mesoskaliga väderprognoser, upp till 2 dygn framåt, kallas AROME och är resultatet av ett samarbete mellan ett tjugotal länder i Europa. AROME-modellen utgör grunden i SMHI:s prognostjänst och används även av MET [SMHI, 2021b; MET, 2021].

De färdiga väderprognoserna består av väderdata behandlade utifrån en tillämpad modell såsom AROME eller IFS. Sådana prognosdata och dess väderprognoser har åtskilliga användningsområden. Wavhale et al. (2020) listar prognosdata som avgörande för kritiska beslut gällande katastrofberedskap, lavinprognoser, skogsbrandsrespons med mera. Fathi et al. (2022) understryker värdet av väderprognoser för bland annat jordbruk, bevattning och sjöfartshandel. De listar även väderprognoser som tillämpbara inom industri, transport och

energihantering. Det är också en viktig faktor i planeringen av militära operationer.

ECMWF förser prognosdata till samtliga av medlemsländernas nationella meteorologiska tjänster, men de tillgängliggör även en del data öppet till allmänheten. Öppna data är en produkt bestående av deras prognos för medellång räckvidd och säsongsbaserad väderprognos som är nedladdningsbar i GRIB2-format utan kostnad. GRIB2-formatet är ett specifikt dataformat som används just för lagring av prognosdata. GRIB är ett binärt dataformat i form av ett gridnät där kod kombineras med externa metadata. GRIB2 är den senaste versionen av GRIB och definieras av World Meteorological Organisation (WMO) [SMHI, 2022]. Data är sedan fritt tillgängligt under förutsättning att användaren följer licensen *Creative Commons CC-4.0-BY* och ECMWF:s egna användarvillkor. De data som tillhandahålls uppdateras en gång i timmen och är i rasterformat med parametrar såsom temperatur, vind och tryck [ECMWF, u.å.-c; u.å.-d].

Exempel på data ur ECMWF:s öppna data ges nedan i tabell 2.1. Tabellen visar önskvärda parametrar i en väderprognos, såsom vindkomponenter i olika riktningar, temperatur, fuktighet och havsvågor samt deras förkortning.

Tabell 2.1: Utdrag ur ECMWF:s öppna data med förkortning till vänster och förklarande namn till höger [ECMWF, u.å.-d]

Short name	Long name
10u	10 metre U wind component
10v	10 metre V wind component
2t	2 metre temperature
msl	Mean sea level pressure
mp2	Mean zero-crossing wave period
mwd	Mean wave direction
mwp	Mean wave period
pp1d	Peak wave period
swh	Significant height of combined wind waves and swell
ro	Runoff
tp	Total Precipitation
sp	Surface pressure
st	Soil temperature
tcwv	Total column vertically-integrated water vapour
d	Divergence
gh	Geopotential height
q	Specific humidity
r	Relative humidity
t	Temperature
u	U component of wind
v	V component of wind
vo	Vorticity (relative)

MET delar också vidare dess data och produkter, både observationer och prognoser, i olika format. Tillsammans med Norsk rikskringkasting (NKR), tillhandahåller MET vädertjänsten Yr. Tjänsten beräknas användas av upp till 11 miljoner människor per vecka och uppskattas vara en av de mest tillförlitliga väderprognoserna just nu, framförallt inom Norge men även över 12 miljoner platser globalt [Yr, u.å.-a]. Yr tillgängliggör prognosdata nedladdningsbart i GRIB-format, samt erbjuder utvecklare ett öppet API för väder kallat MET Weather API.

Exempel på nedladdningsbara prognosdata från Yr ges nedan i tabell 2.2. Tabellen visar några av de parametrar MET och Yr tillhandahåller öppet, däribland havsströmmar och vind. Här beskrivs även vilket område parametern tillhör, hur stor datafilen är, dess filnamn och hur hög upplösning det är.

Tabell 2.2: Utdrag ur vädertjänsten Yr:s öppna data [Yr, u.å.-b]

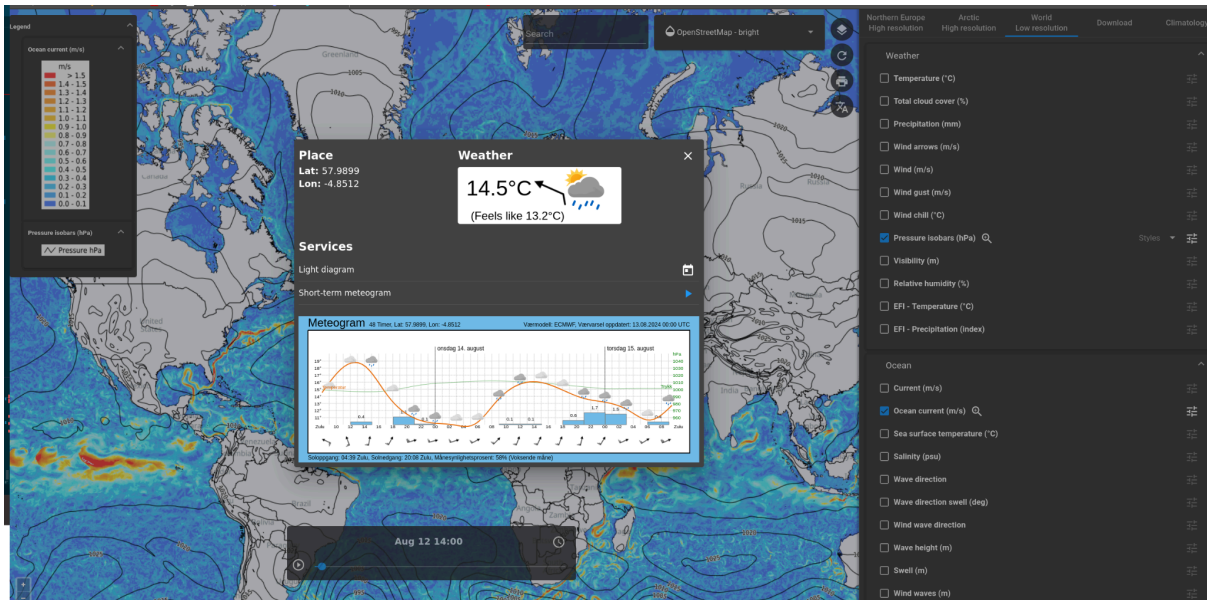
Content	Area	File size	File name	Resolution
Current	Skagerrak	420 kB	norkyst800m_weatherapi_skagerrak.grb	1 hour, 800 m
Current	Oslofjorden	428 kB	norkyst800m_weatherapi_oslofjord.grb	1 hour, 800 m
Current	Sørlandet	1.2 MB	norkyst800m_weatherapi_sorlandet.grb	1 hour, 800 m
Current	Vestlandet	1.1 MB	norkyst800m_weatherapi_west_norway.grb	1 hour, 800 m
Current	Nordsjøen	414 kB	nordic4km_weatherapi_n-northsea.grb	1 hour, 4 km
Current	Nordland	445 kB	norkyst800m_weatherapi_nordland.grb	1 hour, 800 m
Current	Troms/Finmark	304 kB	norkyst800m_weatherapi_troms-finnmark.grb	1 hour, 800 m
Wind	Skagerrak	1.1 MB	meps_weatherapi_skagerrak.grb	1 hour, 2.5 km
Wind	Oslofjorden	560 kB	meps_weatherapi_oslofjord.grb	1 hour, 2.5 km
Wind	Sørlandet	2.0 MB	meps_weatherapi_sorlandet.grb	1 hour, 2.5 km
Wind	Vestlandet	4.3 MB	meps_weatherapi_west_norway.grb	1hour, 2.5 km
Wind	Nordsjøen	310 kB	meps_weatherapi_n-northsea.grb	1 hour, 2.5 km
Wind	Nordland	1.3 MB	meps_weatherapi_nordland.grb	1 hour, 2.5 km
Wind	Troms/Finmark	780 kB	meps_weatherapi_troms-finnmark.grb	1 hour, 2.5 km
Waves	Skagerrak	719 kB	mywavewam4_weatherapi_skagerrak.grb	1 hour, 4 km
Waves	Oslofjorden	89 kB	mywavewam4_weatherapi_oslofjord.grb	1 hour, 4 km

Att data och prognoser tillgängliggörs som öppna data har stor betydelse. Värdet av öppna data har analyserats år 2020 av DAMVAD Analytics. Analysen skedde på uppdrag av Lantmäteriet utifrån det regeringsuppdrag som tilldelades Lantmäteriet i maj 2019 rörande öppna data och vidareutnyttjande av information från offentlig sektor. I rapporten som framställdes, noteras att meteorologiska data som görs tillgänglig som öppna data "leder till omfattande användning av informationen". De rapporterar även ett stort allmänintresse för just meteorologiska data och meteorologiska tjänster [Ekström, D. & Johannesson, J., 2020].

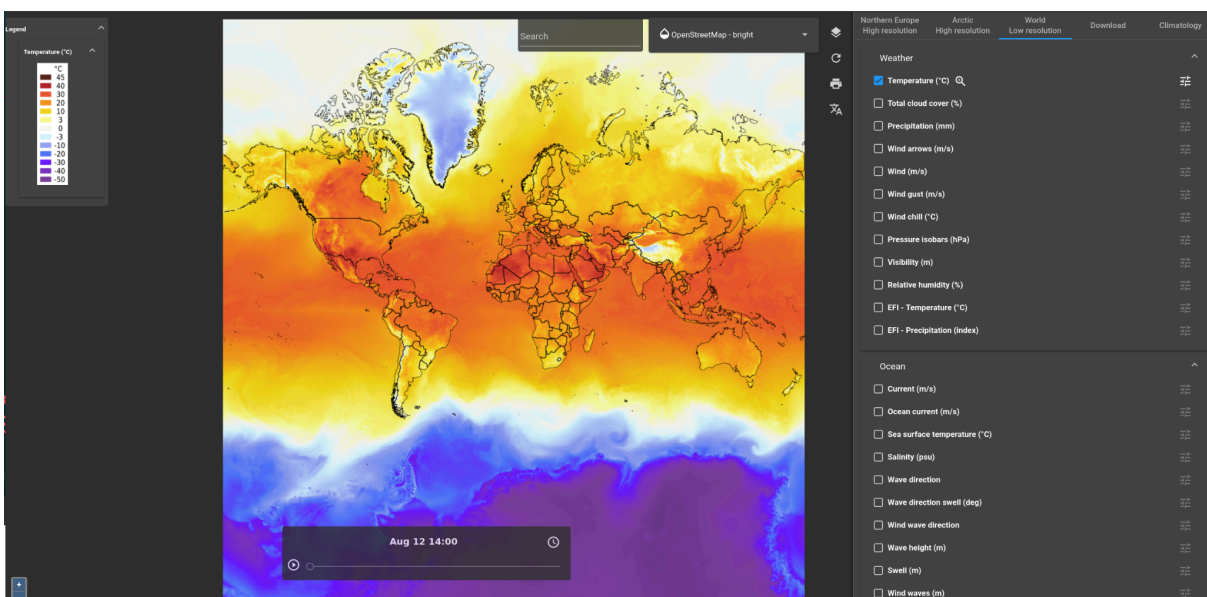
2.2 Väderapplikation

Med väderapplikation åsyftas en applikation för att visa, exportera och/eller beskriva det nuvarande och framtida, men även det historiska, vädret på en önskad plats. T-Kartor tillhandahåller en väderapplikation för kommersiellt bruk, som visar olika typer av prognosdata. Användare kan välja parametrar av, till

exempel, intervaller av tid, område, prognosmodell, för visning av prognosdata. Figur 2.1 och 2.2 är exempel på olika val av parametrar. Figur 2.1 visar en kartbild över tryck och havsströmmar i världen, med överlagd information rörande en specifik plats med vädret just nu och platsens tillhörande meteogram. Figur 2.2 visar en kartbild över temperaturen i världen.



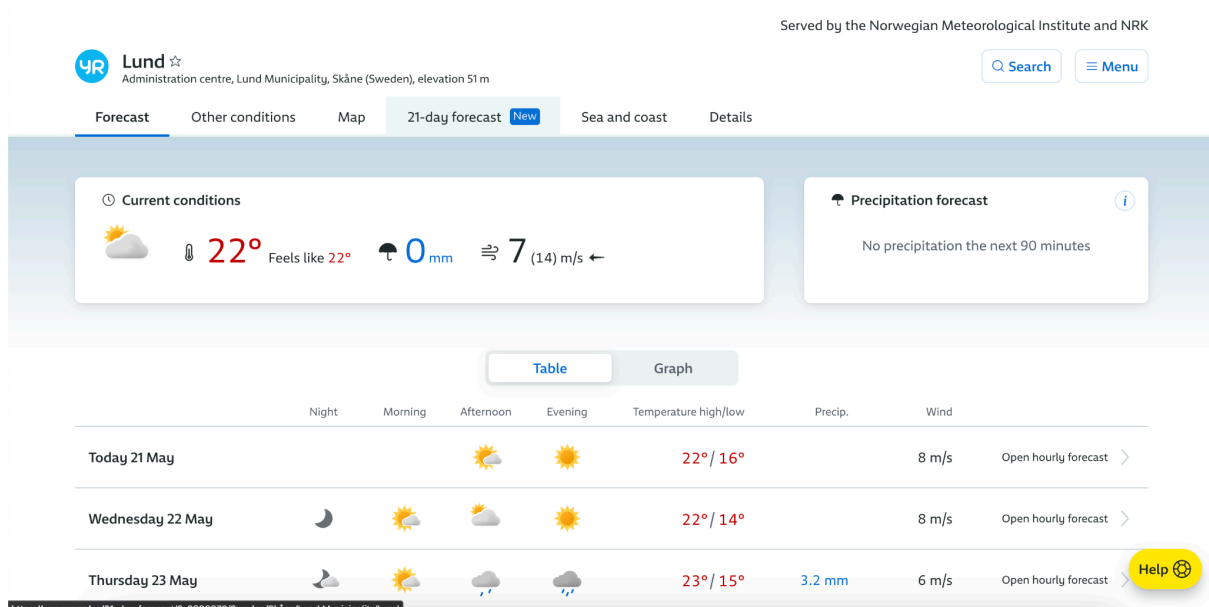
Figur 2.1: Vädret just nu och tillhörande meteogram för en specifik plats i T-Kartors väderapplikation



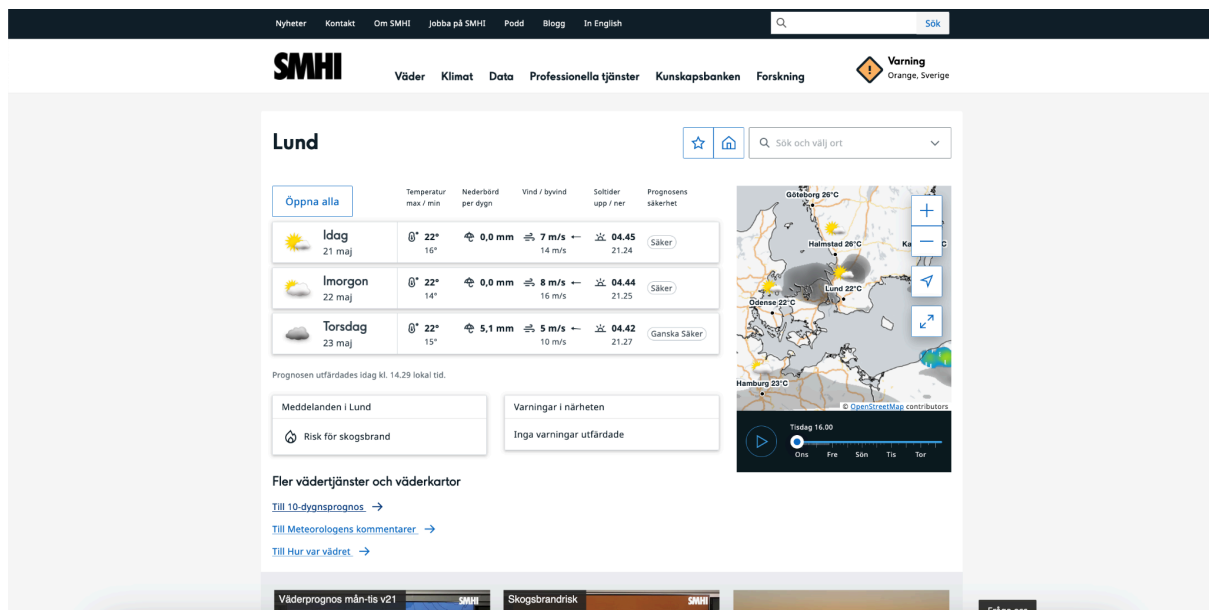
Figur 2.2: Temperaturkarta över världen från T-Kartors väderapplikation

SMHI och Yr är istället två exempel på statligt underhållna väderapplikationer och figur 2.3 respektive 2.4 återger exempelbilder för prognoser av den önskade platsen Lund, Skåne. I deras öppna tjänster finns möjlighet att söka efter en specifik plats och se både aktuellt väder vid sökningens tidpunkt samt prognosen

för de närmsta dagarna. SMHI erbjuder en väderkarta för platsen i visningsläget direkt medan det på Yr:s hemsida finns som ett alternativ att se väderprognosen i helbildskarta.



Figur 2.3: Väderprognos för den sökta platsen Lund, Skåne i Yr.no:s öppna vädertjänst [Yr, u.å.-c]

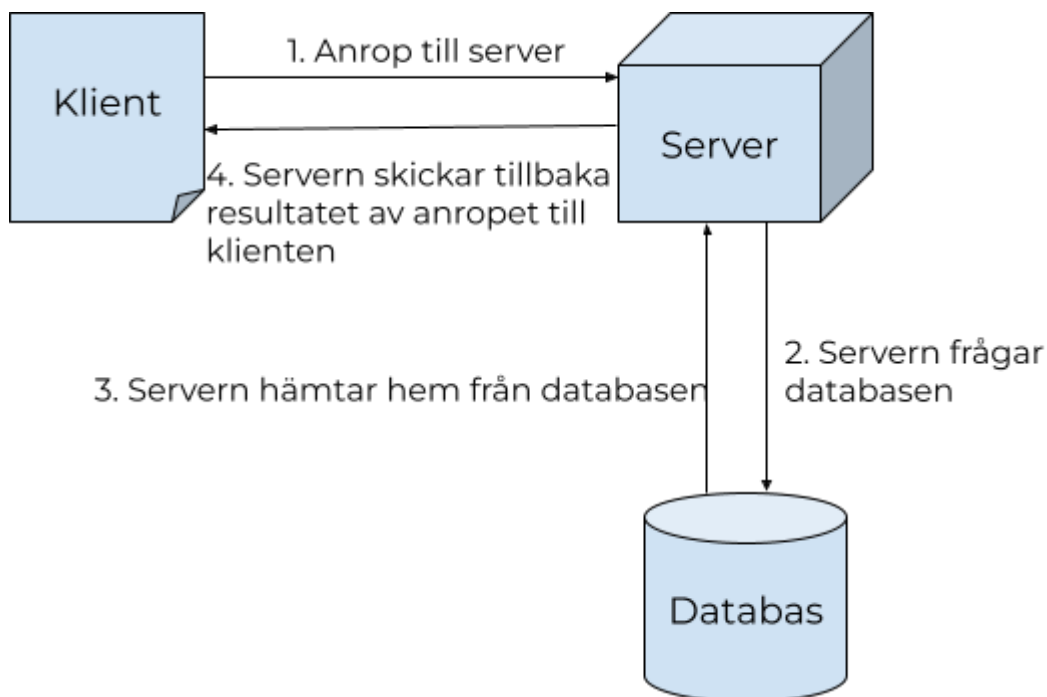


Figur 2.4: Väderprognos för den sökta platsen Lund, Skåne i SMHI.se:s öppna vädertjänst [SMHI, u.å.]

Som exemplen ovan visar är en väderapplikation vanligen en webbaserad visningstjänst. Webbarkitekturen består generellt av en klient och en server, där klienten - en webbläsare eller mobilapp - skickar ett HTTP-anrop till servern som laddar in begärda data från en databas. Geospaciala visningstjänster använder ofta en WMS-server som returnerar en karta över önskade data.

Väderapplikationen i denna studie är en nedladdningstjänst. Fortfarande skickas ett HTTP-anrop med valda parametrar från en klient, men servern erbjuder nedladdning av önskade data istället för visning. Servern i denna studie ska följa OGC-standarden EDR. Att kunna ladda ner specifika prognosdata i råformat som NetCDF eller GRIB istället för visualisering i bildrepresentation, kan vara intressant för en användare som önskar använda data till fortsatta analyser i andra applikationer eller behöver data offline. NetCDF (Network Common Data Form) är ett generellt format för att dela data, som ofta används för lagring av multidimensionella data. NetCDF har en uppsättning mjukvarubibliotek och maskinberoende dataformat, som stödjer skapande, åtkomst och delning av multidimensionella data [Unidata, u.å.].

Figur 2.5 visar arkitekturen bakom en väderapplikation samt processen i fyra steg, med HTTP-anrop från klient till server som första steg, följt av serverns eget anrop till databasen och därefter hämtning av det önskade materialet från databas till server och därefter från server tillbaka till klienten. Det är hur anrop hanteras och i vilken representation som resultatet är som skiljer en ned- och visningstjänst åt.



Figur 2.5: Uppbyggnad av systemarkitektur i en generell väderapplikation med komponenter klient, server och databas

2.3 Standarder och standardisering

Svenska Institutet för Standarder (SIS) definierar en standard som “en gemensam lösning på ett återkommande problem”. Målet med standarder är unisona rutiner som är enhetliga och transparenta. Det ger ett gemensamt språk och utgångspunkt - standarder höjer på så sätt kvaliteten och minskar risken för

missförstånd [SIS, u.å.-a]. För geospatiala data är standarder av stor vikt för utbyte av data mellan olika system. Percival (2010) betonar särskilt delning och kombinerande av data inom geovetenskap för att genom gemensamma datakällor, modeller och processer kunna studera globala frågor. För att åstadkomma detta sätts höga krav på interoperabilitet. Ett fungerande utbyte av geospatiala data mellan applikationer kräver i sin tur standarder, både för hur data ska registreras och för hur systemet ska se ut [Percivall, 2010].

Lantmäteriet delar in standarder inom geodata i tre områden: generella standarder, domänstandarder och teknikstandarder. Fokus för denna studie är de tekniska standarderna. Enligt Lantmäteriets definition, syftar de tekniska standarderna till förenkling för "att skapa geodatatjänster och tillhandahålla geodata, vilket gör det möjligt för olika system att arbeta tillsammans genom standardiserade gränssnitt" [Lantmäteriet, u.å.-a]. Tekniska standarder för så kallat tjänstebaserat geodatautbyte är specifikationer av gränssnitt, för att ha en gemensam grund för både servrar och klienter till basjänster ämnade att tillgängliggöra geodata [SIS, u.å.-b].

Ett antal organisationer utvecklar geodatastandarder, däribland SIS, ISO (International Organization for Standardization), W3C (World Wide Web Consortium) och OGC. SIS är som nämnt ovan den svenska standardiseringsorganisationen och representerar Sverige i europeiska och globala standardiseringsorganisationer. ISO är en internationell organisation som i utvecklingen av serien ISO 19100 lagt fram ungefär 80 geodatastandarder, varav de flesta är tekniska standarder. W3C tar fram mer generella standarder för webben, vilka är basen för flera geodatastandarder. OGC är den ledande organisationen internationellt för framtagning av öppna och allmänt tillgängliga geodatastandarder [Lantmäteriet, u.å.-b]. I enlighet med studiens avgränsning fokuseras följande avsnitt endast på OGC-standarder. OGC presenteras ytterligare nedan i avsnitt 2.3.1, därefter följer ett antal av OGC:s tekniska standarder i avsnitt 2.3.2-2.3.8.

Utöver standardiseringsorganisationer finns regler och riktlinjer som ger struktur till geodata. Ett av de mer omfattande ramverk för hantering av geospatiala data är INSPIRE (Infrastructure for Spatial Information in Europe) [INSPIRE, u.å.]. INSPIRE är ett EU-direktiv som syftar till att effektivisera utbytet av geospatiala data och göra data mer lättillgängliga. Direktivet innehåller "bestämmelser för att inrätta en infrastruktur för geodata i Europa" och i Sverige ansvarar Lantmäteriet för samordning av genomförandet [Lantmäteriet, u.å.-c]. Direktivet trädde i kraft 2007 och har tidigare krävt komplexa tillägg till existerande standarder för att de ska bedömas som i linje med direktivet. I modern tid noteras standarder i befintligt skick, utan tillägg, som INSPIRE *Good Practice* och därmed godkända för användning av direktivet. Ett antal ISO-standarder och en del OGC API-standarder bedöms av INSPIRE som *Good Practice*-standarder [Lantmäteriet, u.å.-b; Chester, 2021].

2.3.1 Open Geospatial Consortium (OGC)

OGC utgörs av över 500 företag, myndigheter, forskningsorganisationer och universitet som tillsammans utvecklar öppna geospatiala standarder på global nivå. En öppen standard är tillgänglig för allmänheten utan kostnad och vem som helst har rätt att implementera den. OGC:s standarder syftar till att göra geospatial information FAIR - sökbar, tillgänglig, interoperabel och återanvändbar (engelskans *Findable, Accessible, Interoperable, Reusable*). De involverade aktörerna i framtagningen av standarder arbetar utefter konsensus för att öppna upp för medlemmarnas specifika behov och möjliggöra innovation [Lantmäteriet, u.å.-b; OGC, 2024a]. Lantmäteriet rekommenderar ett urval av standarder för användning i Sverige, där bland annat ett antal standarder från OGC förekommer: OGC API - Features, WMS, WFS och WCS (se respektive avsnitt nedan för beskrivning av standarderna i mer detalj).

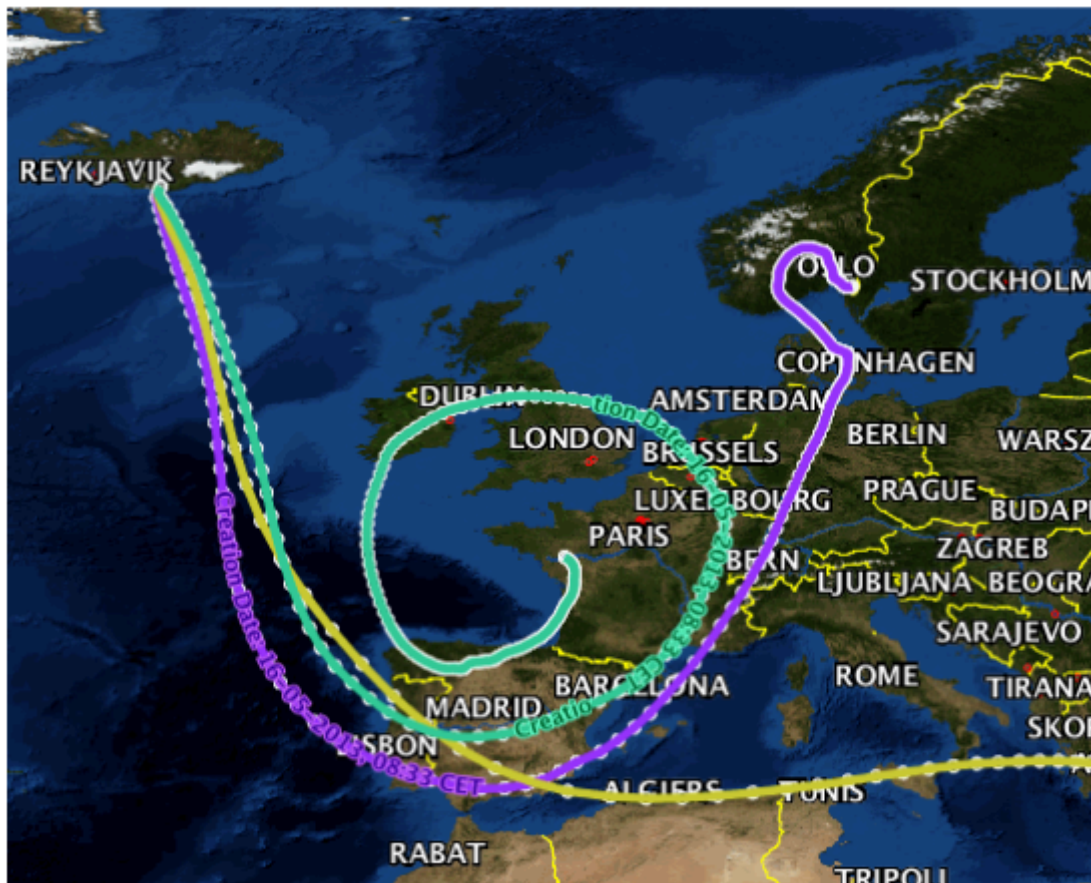
Produkter som implementerar OGC-standarder listas på OGC:s hemsida genom ett *Compliance Program*. Programmet testar så att produkterna efterföljer standarderna. De produkter som listas är verifierade i att de uppfyller kraven i den specificerade standarden, samt att produkten stödjer interoperabilitet mellan system som implementerar samma standard. OGC listar där även antalet referensimplementationer, vilket är specifika implementationer av mjukvara som demonstrerar hur en bestämd standard ska implementeras. OGC listar separat produkter och mjukvara som inte är certifierade genom *Compliance Program* men är utvecklade baserat på OGC-standarder, där alla implementationer listas enligt specifikation. För följande avsnitt noteras dock endast certifierade produkter.

2.3.2 OGC Web Map Service (WMS)

WMS är en gränssnittsstandard som används för att leverera kartor över geospatiala data dynamiskt från geografisk information. "Karta" innebär visualisering av geografisk information i form av en digital bildfil bäst lämpad för visning på en datorskärm. WMS producerar oftast kartor i rasterformat såsom PNG, GIF eller JPEG [OGC, 2006]. Standarden är ett av de mest välkända gränssnitten för rendering av digitala kartor och är i enlighet med INSPIRE-direktivet den gränssnittsstandard som ska användas för visningstjänster. WMS är den standard som används för olika myndigheters visningstjänster, såsom SGU (Sveriges geologiska undersökning), SMHI, Trafikverket, Naturvårdsverket och Lantmäteriet. Gränssnittet är även användbart för visning av väderprognoser. En förlängning till WMS, kallat WMS-T, möjliggör hantering av temporala data. Denna förlängning är fördelaktig vid tidsseriedata som väderdata eller annat historiskt bildmaterial.

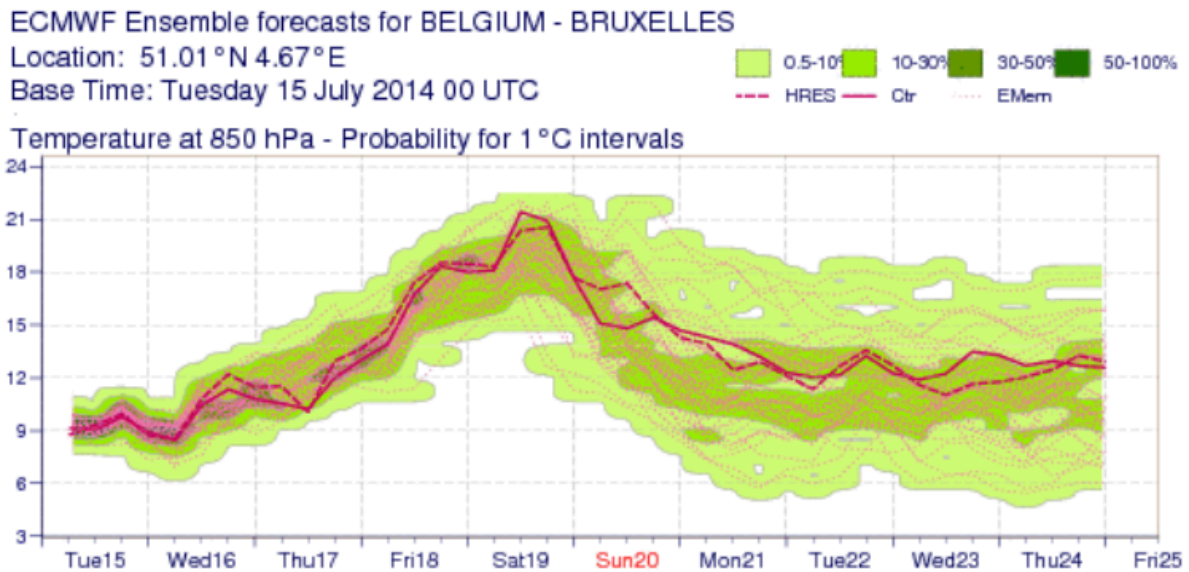
OGC beskriver praxis och riktlinjer för användandet av standarder i *OGC Best Practice*-dokument. För användning av WMS har OGC framställt sådana dokument bland annat för tids- eller höjdberoende data och för

Sammanställningar av prognosdata. Båda dessa exempel riktar sig mot användandet av WMS för visualisering av meteorologiska data och väderprognoser. För tidsberoende data kan WMS till exempel användas för att visualisera beräknade framtida banor för vulkanaska. Figur 2.6 visar ett exempel på visualisering för tre möjliga prognoser av vulkanaska som släpps ut i den övre troposfären från vulkanen Katla på Island [Voidrot-Martinez et al., 2014].



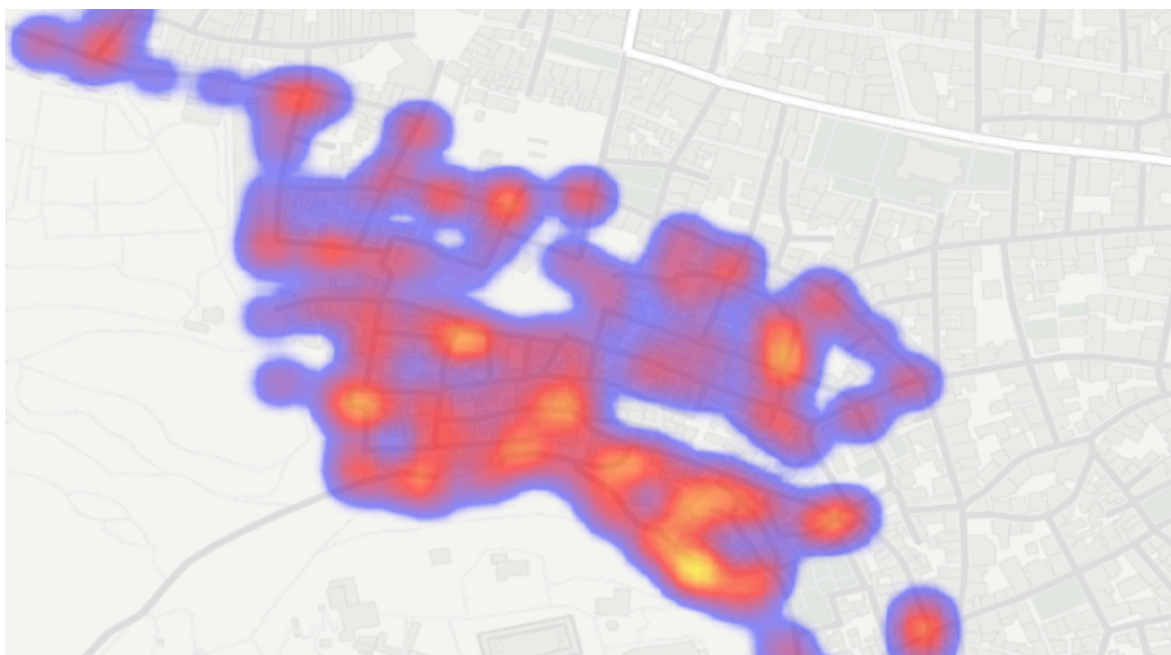
Figur 2.6: Banor av aska beräknade från vulkanen Katla [Voidrot-Martinez et al., 2014]

Sammanställning av prognosdata innebär att ett större antal väderprognoser tillsammans kan estimerar det mest sannolika utfallet i vädret samt uppskatta osäkerheten i en väderprognos. Figur 2.7 visar ett exempel på visualisering av en 10-dagars temperaturprognos för Bryssel [Seib et al., 2018].



Figur 2.7: Sammanställning av väderprognoser för lufttemperaturen i Bryssel under en 10-dagarsperiod [Seib et al., 2018]

Ytterligare ett exempel på användning av WMS ges av Sofos et al. (2017) som presenterar en datamodell i syfte att tillgängliggöra sensorobservationer. Datamodellen följer OGC-standarden SOS för sensorobserverade mätningar och använder en WMS-tjänst för att tillhandahålla "obearbetade observationsvyer som lyfter fram aspekter av kvalitet och produktivitet" [Sofos et al., 2017]. Ett exempel på detta kan vara visualisering av antalet tillgängliga mätningar genom rendering av en värmekarta över relativ mättäthet, se figur 2.8 nedan.



Figur 2.8: Värmekarta över relativ mättäthet för visualisering av antalet tillgängliga mätningar [Sofos et al., 2017]

WMS-standarden är över tjugo år gammal och är idag inte användarvänligt när det sker snabba förändringar i önskat område eller användbart för rendering av kartor där det önskas möjligt att panorera, zooma eller inspektera data. Allen (2020) recenserar WMS som ett gränssnitt bäst lämpat för statiska kartor, där varje förfrågan enbart ger en bild, även om det i bilden kan kombineras flera lager vid användning av en kommaseparerad lista. Ett bättre alternativ för kartor där användaren önskar mer interaktion, är istället gränssnittsstandard Web Map Tile Service (WMTS). WMTS har samma grundprincip som WMS, men returnerar bilder i små brickor (engelskans *tiles*), ofta förrenderade så att de med fördel kan returneras snabbare till användaren. Utöver kortare svarstid förbättras även stabiliteten på servern med mindre, förrenderade brickor. WMS och WMTS är OGC:s huvudsakliga rasterbildsbaserade standarder för visningstjänster [Allen, 2020]. Ett mer modernt alternativ, OGC API - Maps, är för närvarande i konceptfasen [OGC, 2024b].

Den senaste versionen av WMS är WMS 1.3.0 från 2006 och OGC listar totalt 33 organisationer vars produkter är verifierat kompatibla med standarden. Det är en lista påsammanlagt 114 produkter där samtliga både är implementeringar av standarden och för närvarande kompatibla. 3 av produkterna är även referensimplementeringar.

2.3.3 OGC Web Feature Service (WFS)

WFS är en gränssnittsstandard som erbjuder åtkomst av geografisk information på attribut- och egenskapsnivå (engelskans *feature* och *feature property*). Det är en standard för nedladdningstjänster, som tillåter att användaren endast hämtar eller modifierar de data som är intressanta. Det finns en förlängning till standarden, WFS-T (T för engelskans *Transaction*), som möjliggör uppdatering av data. Enligt OGC representerar WFS en förändring i hur geografisk information skapas, modifieras och utbyts på internet. Detta då WFS istället för att returnera en fil med alla data - önskade data men troligen mycket mer data än så - returnerar endast de data som användaren söker efter. Data returneras i vektorformat, främst i Geography Markup Language (GML) även om andra vektorformat såsom Geographic JavaScript Object Notation (GeoJSON) också kan stödjas genom implementering [OGC, 2010].

WFS är även en ISO-standard och är den mest välkända standarden för hämtning av geospatiala data. Allen (2020) noterar WFS som användbart där användaren önskar möjlighet att både visa, analysera och redigera data, huvudsakligen i GIS-program. Till nackdel för användningen är dock att WFS, på grund av volymen data som returneras, ofta är resurskrävande både på servern och på nätverket. Likt WMS är WFS en standard från tidigt 2000-tal och har med det vissa egenheter. Exempelvis kan argumenteras att GML som format är utdaterat [Allen, 2020].

Ytterligare utvärdering av WFS som standard görs bland annat av Liu et al. (2013), som presenterar ett prestandatest (engelskans *benchmark*) för geospatiala webbtjänstplattformar som stödjer WFS, med applicering inom offentlig krishantering. WFS används i studien som stöd för att bland annat lagra och hämta olika format av geospatiala data. Ett typscenario är att en eld bryter ut på en stor bensinmack, där relevanta geospatiala data kan vara information om bensinmacken och dess exakta position, närliggande byggnader som är i farozonen och vägar runt omkring. Studien noterar särskilt att användandet av WFS motiveras av att det är en öppen standard och därmed främjar interoperabilitet och minskad duplikation av data, en stor fördel i hantering av geospatiala data [Lie et al., 2013]. Zhang et al. (2010) diskuterar också användning av WFS inom krishantering vid sökning efter geospatiala attribut, men uttrycker att WFS fallerar i att det inte stödjer automatiska sökningar från olika källor vid en tidspressad nödsituation.

WFS-standarden är ursprungligen från 2002, med den senaste versionen 2.0.2 från 2014. Det fanns planer på att modernisera WFS och uppdatera till version 3.0, bland annat för att stödja andra format än GML såsom GeoJSON, men det utvecklades istället till en egen OGC-standard - Features (se avsnitt 2.3.7). Version 2.0 från 2010 har sammanlagt 39 certifierade produkter från 9 organisationer som implementerar standarden, varav de 39 är alla för närvarande kompatibla och 4 av dessa är referensimplementationer. Utöver denna version listar även OGC tidigare versioners produkter samt verifierade produkter för WFS-T.

2.3.4 OGC Web Coverage Service (WCS)

WCS är en gränssnittsstandard som stödjer hämtning av geospatiala data som raster (engelskans *coverages*). Raster hänvisar här till digital geospatial information som representerar tid/rum-varierande fenomen, speciellt rumstemporala rutnät (både regelbundna och oregelbundna), punktmoln och generella nätstrukturer [OGC, 2018]. En direktöversättning av den engelska termen *coverages* anses ej applicerbar och hädanefter kommer således raster användas i studien med hänsyn till ovanstående definition.

WCS erbjuder åtkomst till multidimensionella raster över internet [OGC, 2018]. Det är en standard för nedladdningstjänster, även om WCS också används för visningstjänster, såsom Lantmäteriets *Markhöjdsmodell Nedladdning*. Standarden möjliggör hämtning av de data som visningstjänsten innehåller. Lantmäteriet noterar även att de flesta GIS-programvaror på marknaden har ett inbyggt stöd för standarden [Lantmäteriet, u.å.-d].

Specifikationen för WCS detaljerar endast kärnan av WCS och för implementering av standarden krävs tillägg därutöver, med ytterligare krav specificerade. Standarden är jämförbar med både WFS och WMS, där WCS i likhet med dessa standarder erbjuder användaren att hämta ut delar av data baserat på valda

parametrar. WCS returnerar dock inte en renderad karta som WMS, utan istället hämtas data tillsammans med dess detaljerade beskrivning. Detta tillåter ytterligare analyser av data och returnerar data i ursprunglig semantik istället för att enbart visualiseras som bilder. I jämförelse med WFS returnerar WCS hela raster som representerar multidimensionella data, där WFS istället returnerar diskreta geospatiala egenskaper [OGC, 2012a]. Skillnaden mellan WCS och WFS är även att WCS möjliggör returneringar i rasterdata istället för vektorformat. I övrigt påminner WCS mycket om WFS, men används mer sällan och i mer specifika fall så som för satellitbilsdata. WCS är bättre lämpat för mer detaljerad filtrering än WFS och WMS, samt extra användbar där multidimensionella data önskas i olika bildformat [Allen, 2020].

Ett exempel på användning av WCS är inom jordbruk, där fjärranalysbilder av jordbruksfält kan vara ett stöd i att bestämma status på fälten och analysera bäst lämpade grödor samt hur fälten bäst odlas och bevattnas [Sun, Di, et al. 2016]. Sun, Di, et al. (2016) presenterar en kombination av WCS och Simple Object Access Protocol (SOAP), ett XML-baserat protokoll för informationsutbyte i webbtjänster, för att underlätta och effektivisera hämtning av fjärranalysbilder för jordbruksägare. WCS understryks här vara ett interoperabelt gränssnitt som lämpar sig väl för manipulation av rasterdata i form av raster, medan kombinationen WCS och SOAP säkerställer att fjärranalysbilderna enkelt och säkert levereras från server till slutlig användare [Sun, Di, et al., 2016]. Di et al. (2010) kombinerar istället WCS med OGC Catalog Service-Web Profile (CSW) protokoll, med WCS som gränssnitt för en tjänst som tillåter användaren att anpassa data och CSW som protokoll för en tjänst som möjliggör för användaren att hitta data. Data som används i studien samlas in av Geostationary Operational Environment Satellites (GEOS) av National Oceanic and Atmospheric Administration (NOAA). NOAA GOESS-data är en del av Global Earth Observation System of Systems (GEOSS) och studien syftar till att göra dessa data både sökbara och tillgängliga i GEOSS genom ovanstående tjänster för anpassning och sökning [Di et al., 2010].

Den nuvarande versionen av WCS är 2.1 från 2018, som förlänger WCS 2.0 från 2012. OGC listar certifierade produkter för både kärnstandarderna och de olika förlängningarna, varav kärnstandarderna i version 2.0.1 har 20 verifierade produkter från 6 olika organisationer. Samtliga 20 produkter är standardimplementationer och för närvarande kompatibla. En av dessa är även en referensimplementation.

2.3.5 OGC Sensor Observation Service (SOS)

SOS definierar ett gränssnitt för webbservice, specifikt för standardisering av sensorsystem och observationer av sensorer. Det är en gränssnittsstandard för hämtning och hantering av metadata och observationer från sensorsystem. Standarden är en viktig del av OGC Sensor Web Enablement (SWE) ramverket, som ämnar att möjliggöra för applikationer och tjänster att kunna erhålla tillgång till sensorer av olika typer. SOS bidrar till ökad interoperabilitet för sensordata över

internet i och med dess standardisering av sensorer och sensorssystem. Standarden har utöver kärnstandarderna flera förlängningar, *Enhanced*, *Transactional* och *Result Handling* [OGC, 2012b].

SOS kan användas för hantering av mätobservationer från sensorer i lantmäterisyften [Sofos et al., 2017]. Standarden kan även användas för tidsseriedata, till exempel i samband med sensorobservationer av hav och marina. Bermudez et al. (2009) jämför användning av standarderna WFS och SOS för att "representera och utbyta punktdataposter från fasta in-situ marina plattformar", med syfte att kunna filtrera observationer gjorda av sensorer genom olika kriterier såsom region, plattformstyp och tid. WFS är, tack vare sin *GetFeature*-operation som möjliggör hämtning av specifika attribut, tillämpligt i detta avseende. I studien anses WFS dock slutligen vara för enkel och kräva profilering för att kunna utnyttjas effektivt. SOS är istället mer anpassad för att representera både tidsserieobservationer från sensorer, men också från sensorsystem, och betraktas alltså som det mer fördelaktiga alternativet av de två för detta område [Bermudez et al., 2009].

Den senaste versionen av SOS är 2.0 från 2012. Standarden har inga certifierade implementationer, varken för version 1.0 eller 2.0.

2.3.6 OGC API - Environmental Data Retrieval (EDR)

EDR är en gränssnittsstandard vars syfte är att tillhandahålla lätta frågegränssnitt för tillgång till spatiotemporala dataresurser. Syftet är att förenkla dataåtkomst genom att kunna begära ut data på en Position, inom en Area, i en Kub, längs med en Bana (engelskans *Trajectory*) eller genom en Korridor. Anrop kan även göras för Föremål (engelskans *Items*), Platser (engelskans *Locations*) och Instanser (engelskans *Instances*). Standarden önskar dels att underlätta åtkomsten till ett spektrum av data genom ett enhetligt, väldefinierat webbgränssnitt, dels att reducera ner data till endast efterfrågat urval och därmed kunna gömma stora delar av komplexiteten i datalagringen. OGC nämner särskilt väderprognoser som ett användningsområde där EDR kan användas för att hämta specifika urval ur stora samlingar geodata [OGC, 2023a].

EDR är en standard inom OGC API-familjen, som utvidgar kärnstandarderna OGC API - Common (Common). Common definierar den tekniska grunden för alla OGC API:er, med definitioner för reglerna för HTTP-protokoll och Uniform Resource Identifiers (URIs). API-standarderna är ett modernt alternativ till de ursprungliga OGC Web Services (OWS): WMS, WFS, WCS och så vidare [OGC, 2023b]. Syftet med EDR är, som nämnt ovan, att förenkla frågegränssnitt så att det endast finns en uppsättning möjliga sätt att begära ut data från tjänster som implementerar EDR. Det är detta som särskiljer EDR från exempelvis WCS (se avsnitt 2.3.4) eller SOS (se avsnitt 2.3.5), även om alla tre standarder med fördel kan användas till tjänster med multidimensionella data. Koncepten av ovannämnda Position, Area,

Kub, Bana eller Korridor, är konsistenta med vad som återfinns i SOS-standarden och implementationer av WCS har enkelt tillgång till EDR-dataresurser (det vill säga, multidimensionella dataset) [OGC, 2023a].

Framförallt specificerar EDR två operationer: *Discovery* och *Query*. De svenska direktöversättningarna anses här inte applicerbara och studien håller sig således till de engelska termerna. *Discovery*-operationer är operationer för att hämta metadata om tjänsten och dataresurser tillhandahållna av tjänsten, samt för att kunna förhöra API:et om dess kapacitet. *Queries* är operationer för att begära ut dataresurser utifrån användarens valda parametrar. Anrop resulterar i geospatiala data i vektorformat, bestående av deras geografiska attribut och deras egenskaper [OGC, 2024c].

EDR är från 2021 och således en högst aktuell, ny lösning för hantering av multidimensionella data. Exempelvis så hanterar EDR hämtning av raster och HTML-svar, något den tidigare standarden SOS inte gör. EDR är kompatibel med WCS-implementationer, men har till skillnad från WCS möjlighet att generera en enda förfrågan mot en samling raster. EDR är även helt kompatibel med Features del 1 (se avsnitt 2.3.7), där båda standarderna stödjer Samlingar (engelskans *Collections*) och Föremål [OGC, 2024c].

2023 gjorde British Geological Survey (BGS) ett pilotprojekt för användning av EDR. På beställning av Marine Environmental Data and Information Network (MEDIN), var projektets mål att undersöka möjligheterna att implementera EDR för att kunna leverera data externt från MEDIN:s databas Geology and Geophysics Data Archive Centre (DAC). Slutresultatet blev dock inte implementering av EDR utan implementering av Common Query Language (CQL) tillsammans med Features, som är en annan standard inom OGC API-familjen. Features kombinerat med CQL ansågs ha samma funktionalitet som EDR, men med annan syntax. Data som används i projektet var bland annat av marina provdata, geofysiska undersökningsdata, kartindex, geologiska faktorkartor och kolvätebrunnar ute till havs [Blackburn et al., 2023].

Blackburn et al. (2023) anför att projektet inte kunde fullfölja implementering av EDR då standarden upplevdes som emellanåt svår och tvetydig att implementera. Utmaningarna i implementeringen berodde bland annat på att standarden huvudsakligen är fokuserad på rasterdata, det vill säga multidimensionella data, och det därmed blev svårt att implementera i kontexten av deras typ av data. De följer upp med att både Features och EDR levererar potentiellt väldigt stora dataset som kan påverka prestandan för visualisering. För större dataset, 100+ attribut, bör istället OGC API - Tiles implementeras, som påminner mer om WMS. Då kan Features eller EDR, efter visualisering, användas för att hämta hem en mer lämplig delmängd av data.

Eugene et al. (2022) använder standardiserade geospatiala webbtjänster för att tillgängliggöra ICroplandNet, ett träningsdataset av bevattnad odlingsmark som baseras på Cropland Data Layer Data mellan 1997 och 2021 i USA. Syftet är att förenkla för odlare och beslutsfattare inom precisionsbevattningsodling att upptäcka bevattnad odlingsmark. För åtkomst av data vid parametrering, förordas användning av EDR, då EDR stödjer fördefinierade hämtningar av delmängder med hjälp av de tidigare nämnda Position, Area, Kub, Bana och Korridor [Eugene et al., 2022].

EDR är på sin första version, 1.1, och har i dagsläget endast en certifierad implementation. Denna produkt är även en referensimplementation.

2.3.7 OGC API - Features (Features)

Features är en API-standard som utvidgar Common och implementerar den tekniska grunden av OGC API-familjen. Syftet med Features är att möjliggöra för att skapa, modifiera och söka efter attribut på webben. Standarden är uppdelad i två delstandarder, där del 1, Kärnan (engelskans *Core*), specificerar en kärnstandard och dess kapaciteter. I kärnstandarden kan attribut hämtas där geometrin är representerad i koordinatsystemet WGS 84 och axelordningen är longitud/latitud. Del 2 har istället kapacitet för attribut med geometrier representerade i alla andra koordinatsystem och har således namnet Koordinatreferenssystem genom referens (engelskans *Coordinate Reference System by Reference*) [OGC, 2024d]. Del 1 publicerades 2019 och del 2 2020. Ytterligare delar, del 3 och 4, är planerade för framtiden med mer avancerade operationer. Endast del 1 är idag tillgängligt öppet.

Både Features och EDR implementerar *Discovery*- och *Query*-operationer. Features stödjer flera format för utmatning, däribland HTML, GeoJSON och GML, och föreskriver inte något format i specifikationen. Standarden påminner även om WFS, men följer ett mer moderniserat tillvägagångssätt i fråga om webbarkitektur [OGC, 2019].

Features är användbart för statliga myndigheter, privata organisationer och akademiska institut som önskar tillgängliggöra geodata i vektorformat för andra att kunna göra ytterligare analyser på data eller sammanställa nya kartor. Som nämnt ovan påminner Features mycket om WFS och var från början tänkt som en ny version av WFS (WFS 3.0), men utvecklades sedan till sin egen standard. WFS är fortfarande mer användningsbar för applikationer som endast är kompatibla med de klassiska OWS-standarderna, men i övrigt är Features bättre anpassad till modern webbarkitektur och stödjer fler format än vad WFS gör [OGC, 2024e].

Santana & Davis Jr (2023) jämför Features med WFS och utvärderar prestandan i olika implementeringsmetoder för de två standarderna. De noterar att prestanda inte enbart beror på systemarkitektur, utan att andra faktorer såsom "mjukvara,

programmeringsspråk och filformat också spelar en signifikant roll” [Santana & Davis Jr, 2023]. GeoServer är en certifierad produkt av OGC och har både en implementering för OGC API och för WFS. I jämförelse mellan dessa implementeringar konstateras att API:t har bättre prestanda. Det framförs att Features må vara fördelaktigt i enlighet med studiens resultat och dessutom ämnat att vara mera användar- och utvecklbarvänligt, men att utveckling av OGC API:erna inte ännu når mognadsgraden hos de äldre OWS-standarderna.

Studien gjord av Eugene et al. (2022) föreslår Features till användning för att kunna söka efter träningsdataset då standarden stödjer sökning efter geografiskt område, tidsintervall samt skräddarsydda parametrar - framförallt i den planerade Del 3 av standarden. Santhanavanich et al. (2023) använder Features för att tillgängliggöra 2D-data av tillhörande egenskaper till 3D-modellering av städer.

Det finns än så länge endast en version av Features, 1.0, som kom 2019. OGC listar 8 organisationer, med totalt 9 produkter, som implementerar standarden. 3 av dessa ses som referensimplementationer.

2.3.8 OGC API - Records (Records)

Records är ytterligare en API-standard som kommer att tillhöra OGC API-familjen, dock i skrivande stund är standarden fortfarande i konceptfasen. Idéen med standarden är att det ska vara möjligt att skapa, modifiera och söka efter metadata på internet och därmed standardisera sättet som kataloger av metadata är exponerat. Records standardiserar även sättet som register exponeras och hanteras, så att det blir möjligt att upptäcka och dela relaterade dataresurser från geospatiala dataresurser eller från deras tillhörande metadata [OGC, 2024f].

Syftet med Records är att definiera ett API med moderna API mönster och kodningar för att förenkla åtkomsten till geodataresurser på internet, samt förenkla upptäckten av metadata och funktionalitet av hämtningsoperationer jämförbart med den äldre OGC-standarderna Catalogue Service (CSW) [OGC, 2024g]. Standarden är således huvudsakligen för sortering och tillgängliggörande av data, där Records är tänkt att underlätta åtkomsten till geodata genom att göra metadata mer sökbara.

Eugene et al. (2022) nämner att Records kan med fördel användas för sökning efter träningsdataset då Records stödjer sökning filtrerad på geografiskt område, tidsintervall och fritext. Det noteras dock att Records ännu fortfarande bara är i konceptfasen [Eugene et al. 2022]. Simoes & Cerciello (2022) undersöker OGC API-alternativen till OWS-standarderna i relation till The eMOTIONAL Cities-projektet. Projektet ämnar att “förstå hur den naturliga och byggda miljön kan forma känslor hos dem som upplever den” och gör så genom att bland annat samla in dataset som karaktäriserar både den känslomässiga samt den byggda

miljön [Simoes & Cerciello, 2022]. De standarder som jämförs är framförallt WFS gentemot Features, WMTS mot OGC API - Tiles och CSW mot Records. Där framkommer att Records redan har server-implementationer såsom Python-biblioteket *pygeoapi* och stöds på klient-sidan i QGIS genom *Metasearch core plugin*, trots att det ännu inte är en accepterad standard hos OGC [Simoes & Cerciello, 2022].

Records är, i skrivande stund, endast i konceptfasen och har således ännu inga certifierade implementeringar.

2.3.9 OGC Web Processing Service (WPS)

WPS är en gränssnittsstandard med syfte att förenkla bearbetning av geodata genom ett standardiserat gränssnitt för beräkningstjänster via webben. Eftersom geospaciala data generellt behöver behandlas och processas innan informationen kan användas, behövs enhetliga regler för hur indata och utdata ska hanteras. WPS möjliggör för användaren att bestämma hur utförandet av en process ska ske, antingen genom direkt bearbetning eller genom asynkron bearbetning. Direkt bearbetning rekommenderas för uppgifter som inte kräver speciellt mycket tid medan asynkron bearbetning är användbart i de mer komplexa och tidskrävande fallen [OGC, 2018b].

WPS används för många av de processer som finns i GIS-mjukvara och för mer specialiserade processer för spatio-temporal modellering och simulering. Standarden är designad främst för spatial behandling, såsom exekveringen av databehandling som kombinerar rasterdata, vektordata och/eller raster för att producera nya rasterdata, vektordata, och/eller raster. Med raster åsyftas definitionen från avsnitt 2.3.4. WPS går dock likväl att använda i andra webbtjänstmiljöer och kan alltså erbjuda icke-spatiala indata, behandlingar och utdata. Gränssnittet är konfigurerbart och WPS möjliggör att algoritmer, beräkningar och modeller kan göras mer tillgängliga [OGC, 2018b; 2024h].

Cannata et al. (2014) belastningstestar en geoportal, inom HELIDEM-projektet, som integrerar flera OGC-standarder såsom WMS, WFS, WCS och WPS. Syftet med portalen är att "valorisera huvudprojektets utdata som är en gränsöverskridande digital terrängmodell" och underlätta användningen av data [Cannata et al., 2014]. Genom WPS-standardens erbjuder HELIDEM-portalens ett antal processer såsom dataextraktion, koordinatomvandling och hämtning av konturlinjer i en höjdmodell. Portalens server-sida är uppdelad i två komponenter, där WPS, WMS och WFS implementeras på en server för databearbetning och leverering av resultat. WCS implementeras däremot på en annan server för leverans av digitala terrängmodeller. Belastningstestet visar att systemet "har en god robusthet och god kvalitet på service" även om svarsfrekvensen ökar med mer komplicerade och komplexa processer [Cannata et al., 2014].

Andra användningsområden för WPS är till exempel som bas för miljösystem-modellering, då WPS kan användas för att designa tidsberoende modeller som webbtjänster såsom i en studie av Castronova et al. 2013.

WPS är för närvarande i version 2.0.1 som publicerades 2015. Genom OGC listas endast en organisation med en implementering av standarden, som även räknas som en referensimplementering.

Del III
Fallstudier

3 Kravställning

Studiens praktiska del består av två fallstudier. Fallstudie 1 avser en teoretisk jämförelse av standarden OGC API - Environmental Data Retrieval (EDR) tillsammans med andra OGC-standarder. I fallstudie 2 implementeras EDR praktiskt. Fallstudierna genomförs i syfte att diskutera och utvärdera EDR. Bakgrunden till fallstudierna är T-Kartors väderapplikation, en visningstjänst med flera källor till prognosdata. T-Kartor ser ett ökat behov av ett välfungerande gränssnitt som möjliggör interoperabilitet och effektiv hämtning av prognosdata. Val av gränssnittsstandard är således viktigt i den framtida utvecklingen av deras applikation.

Fallstudierna verkställs utifrån T-Kartors perspektiv och behov. I dialog med T-Kartor har användarfall och kravställning arbetats fram. Det framtagna materialet bedöms kunna appliceras på en allmän väderapplikation med syfte att tillgängliggöra prognos- och väderdata. Kravställningen ligger till grund för båda fallstudierna och är därmed utgångspunkt i både den teoretiska jämförelsen och den praktiska implementeringen. Användarfall och kravställning presenteras i avsnitt 3.1 respektive 3.2. Därefter följer fallstudie 1 i kapitel 4 och fallstudie 2 i kapitel 5. Samtliga resultat diskuteras i kapitel 6.

3.1 Användarfall

Kravställningen syftar till att presentera grundläggande krav för utvärderingen av EDR. Vid framtagning av kravställningen har två användarfall studerats samt hänsyn tagits till utvecklarbehov. Användarfallen är baserade på potentiella användare av väderapplikationen, både externt och internt i en organisation. De studerade användarfallen presenteras mer detaljerat nedan i avsnitt 3.1.1 och 3.1.2. Kraven har utarbetats från dessa användarfall och sedan delats in i två underkategorier: utvecklare och användare.

3.1.1 Användarfall - Hämtning av data

En av T-Kartors kunder och användare av deras väderapplikation är det norska försvaret. En anställd på det norska flygvapnet planerar en operation där ett kompani ska transporteras över en specifik rutt. Den anställde vill analysera förutsättningarna för vädret längs rutten och därmed kunna filtrera data endast över den tid och rymd som rutten anger. Planeringen kräver exakta och tillförlitliga väderförhållanden och den anställde önskar ta reda på information kring ruttens överliggande data såsom var väderprognosen kommer ifrån, vilket område hela prognosen är gjord över, vilken prognosmodell som använts och vilken tidsperiod som prognosen syftar till. Den anställde vill även kunna jämföra olika prognoser från olika datakällor. Rutten är angiven i lokala koordinatsystem och väderdata måste således läsas in med korrekt position i samma koordinatsystem för att vara användbar för den specifika rutten.

Den anställde öppnar en webbklient för att visualisera den önskade rutten och filtrerar fram området av intresse. Endast en webbläsare är nödvändig för att tjänsten ska fungera på den anställdes dator. Tjänsten antas vara säker så att ingen förutom användaren och utvecklare kan få åtkomst till sökningen. Den anställde önskar ha möjlighet att spara den aktuella sökningen till historiken, utan att det kompromissar säkerheten. Vid hastigt ändrade rutter vill den anställde kunna göra en ny sökning utan fördröjning och tjänsten antas vara tillräckligt tidseffektiv för den anställde att göra flera sökningar på kort tid.

Efter visualisering önskar den anställde kunna använda rådata av det framtagna området till ytterligare analyser i andra program. Det önskas vara möjligt att ladda ned rådata i NetCDF så att det finns en lokal kopia av sökningens resultat på den anställdes dator.

3.1.2 Användarfall - Interoperabilitet mellan T-Kartors tjänster

T-Kartors webbtjänst använder idag en WPS-server med förprogrammerade processer för att omvandla rådata till bilder baserat på olika parametrar (tid, område, prognosmodell etc.). Bilderna levereras via en WMS-server. Protokollet för WPS är mycket öppet och specificerar endast hur integration ska ske, men inte vilka processer som används. WPS-standarden definierar endast tre operationer, där en av operationerna initierar en process. Vad för processer som implementeras regleras inte och operationen kan därmed innebära lite vad som helst beroende på implementation. I praktiken betyder det att en klient som ska kopplas upp mot servern måste vara utformad efter serverns specifika processer och en WPS-server kan innebära att man förlorar interoperabilitet då varje klient måste skräddarsys mot serverns tjänster.

En av T-Kartors kunder, det norska försvaret, som använder webbtjänsten tillhandahåller data som kan vara intressanta för andra organisationer. I samarbetet mellan organisationer krävs att webbtjänsten är utformad så att exempelvis USA:s försvar kan koppla upp sig mot Norges dataserver. Det är önskvärt att utformningen är gjord efter en öppen, väletablerad standard med tillgång till mycket verktyg, för att möjliggöra för enkel utveckling av webbtjänsten. Tjänsten ska även vara utformad på så sätt att den är lätt att förstå för alla potentiella användare.

En användare på plats i USA önskar koppla upp sin klient mot Norges dataserver. Användaren önskar använda befintlig klient som är implementerad efter OGC:s standarder, utan extra anpassning efter WPS-servern. Det ska vara möjligt för en användare i USA och en användare i Norge att ha åtkomst till webbtjänsten samtidigt. Vid ett annat tillfälle önskar en annan användare i USA koppla en egen server mot Norges server för att få tillgång till deras data. Även här krävs interoperabilitet mellan system för sömlös kommunikation och datadelning.

3.2 Kravsammanställning

Följande kravsammanställning baseras på användarfallen presenterade i avsnitt 3.1. Kraven är indelade i två kategorier. Första kategorin, utvecklingskrav (U), är krav framtagna ur perspektivet av utvecklare av tjänsten. Den andra kategorin, användarkrav (A), utgår från användarens perspektiv. Några av kraven kan tillhöra båda kategorierna, men har här valts att endast ingå i en.

3.2.1 Utvecklingskrav

U.1 Öppen standard

Tjänsten ska vara utformad efter en öppen standard.

U.2 Etablerad, enkel implementering av standard

Standarden ska vara etablerad med många godkända verktyg, för fortsatt utveckling av tjänsten.

U.3 Webbaserad

Tjänsten ska kunna användas direkt i en webbläsare.

U.4 Interoperabilitet

Tjänsten ska vara utformad enligt väldefinierad specifikation så att sömlös interaktion är möjlig mellan olika system.

U.5 Stöd för simultana, tillförlitliga källor

Tjänsten ska kunna stödja flera datakällor. Med källor avses olika data från exempelvis SMHI, Yr eller ECMWF.

U.6 Tidseffektiv

Tjänsten ska inte vara för tidskrävande. Stöd ska finnas för format och teknologier som är aktuella och välfungerande.

U.7 Hantera flera anrop samtidigt

Tjänsten ska kunna hantera stora datamängder och ett stort antal anrop samtidigt.

3.2.2 Användarkrav

A.1 Stöd för hämtning av prognosdata

Användaren ska kunna hämta prognosdata i NetCDF-format eller GRIB direkt i tjänsten.

A.2 Stöd för filtrering och definiering av delmängd

Användaren ska kunna filtrera och definiera delmängder av rådata, utifrån geometriska punkt-, linje- och ytobjekt.

A.3 Lokaliserade rådata

Inläsning av data ska ske med korrekt position. Koordinatsystem som ska stödjas är EPSG:3034, EPSG:32661 och EPSG:3857.

A.4 Lätt att använda

Tjänsten ska följa principer i webbdesign och det grafiska gränssnittet ska vara enkelt att förstå.

A.5 Information

Metadata kring tjänsten ska vara lätt att tillgå. Med metadata avses information som datakälla, geografiskt område, tillgängliga prognosmodeller, tidsperiod och vilka parametrar som finns (exempelvis vind, tryck, etc.).

4 Fallstudie 1 - Teoretisk jämförelse

Fallstudie 1 syftar till att teoretiskt utvärdera standarden EDR tillsammans med andra OGC-standarder. Samtliga standarder presenteras grundläggande i litteraturstudien i avsnitt 2.3 och motiveras för fallstudien nedan i avsnitt 4.1. Fallstudien baseras på kravställningen (se kapitel 3). Standarderna bedöms utifrån hur väl de uppfyller kraven - huruvida de uppfyller, uppfyller delvis eller inte alls uppfyller varje krav. Genomförande av fallstudien beskrivs i detalj i avsnitt 4.2. Resultat redovisas i avsnitt 4.3.

4.1 Val av standarder

Totalt jämförs åtta standarder. Jämförelse sker av EDR och ytterligare sju standarder med liknande användningsområden, med avseende främst på väder- och prognosdata. Samtliga standarder är OGC-standarder. OGC är den ledande organisationen inom geospatiala datastandarder och deras öppna standarder möjliggör större interoperabilitet mellan system. Att endast jämföra OGC-standarder motiveras även av att standarderna därmed genomgått samma granskningsförfarande och är likvärdigt utformade.

Utöver EDR, undersöks OGC Web Map Service (WMS), OGC Web Feature Service (WFS), OGC Web Coverage Service (WCS), OGC Sensor Observation Service (SOS), OGC API - Features (Features), OGC API - Records (Records) och OGC Web Processing Service (WPS).

WMS och WFS är två av de mest väletablerade standarderna i OGC:s arsenal och därav av intresse för fallstudien. WMS tillämpas i de flesta visningstjänster och är användbar för att visualisera väder- och prognosdata, dock endast statiska eller förproducerade kartor. WMS används idag i T-Kartors webbtjänst och är således av betydelse trots att visualisering skiljer sig från resterande standarders användningsområden. WFS används till att hämta geografiska dataobjekt över webben, inklusive väder- och prognosdata, där risken dock finns att prestanda försämras ju större volym data har.

WCS är något mindre etablerat än WMS och WFS, men den mesta GIS-mjukvaran har inbyggt stöd för WCS. WCS är designad för att tillgängliggöra multidimensionella geospatiala data, men prestandan begränsas beroende på serverns kapacitet. SOS är en standard för hantering av data från sensorer, såsom väderobservationer. Det är en mer specifik standard än WMS, WFS, WCS och således mindre allmänt använd, men med mer fokus på väderdata. WPS tillhör de äldre OWS-standarderna (WMS, WFS, WCS etc.), dock mindre välkänd än de andra, men mycket flexibel i hantering av data.

Features är en nyare standard än de andra, med ett modernare tillvägagångssätt för dataöverföring. Features, EDR och Records utgår alla från OGC API - Common

och anses således ha liknande fördelar. Records är dock fortfarande i konceptfasen och därmed ännu inte accepterad av OGC som officiell standard.

4.2 Metod

Fallstudien utgår ifrån kravställningen presenterad i avsnitt 3.2. För varje krav avgörs huruvida standarderna uppfyller, uppfyller delvis eller inte alls uppfyller kravet. Kraven bedöms vara likvärdiga. Granskning sker av OGC:s tekniska specifikationer och annat tillgängligt material såsom OGC:s självstudiemoduler (OGC e-Learning) samt vetenskapliga artiklar. Därtill nyttjas verktyget ChatGPT, i syfte att sammanställa och fastställa hur standarderna förhåller sig till de specificerade kraven.

ChatGPT tillämpas genom att varje krav omformuleras till en fråga och ställs med avseende på hur väl det uppfylls av OGC-standarderna WMS, WFS, WCS, SOS, Features och WPS. Vid fallstudiens genomförande var ChatGPT senast uppdaterad januari 2022 och kunde ej besvara frågor angående de två nyare standarderna: EDR och Records. De uppskattas dock ha liknande förutsättningar som Features och vissa krav bedöms således uppfyllas likvärdigt av EDR och Records.

Kommunikation med ChatGPT återfinns i Appendix A. De omformulerade kraven ställs med avseende på samtliga standarder i samma fråga, formulerat såsom "Are the OGC standards WMS, WFS, WCS, SOS, OGC API - Features and WPS defined as "open standards"?" [*Är OGC-standarderna WMS, WFS, WCS, SOS, OGC API - Features och WPS definierade som "öppna standarder"?*]. Denna prompt motsvarar krav U.1 i kravställningen och besvaras av ChatGPT som följer:

Yes, the Open Geospatial Consortium (OGC) standards you mentioned — WMS (Web Map Service), WFS (Web Feature Service), WCS (Web Coverage Service), SOS (Sensor Observation Service), OGC API - Features, and WPS (Web Processing Service) — are considered "open standards." [OpenAI, 2024; Se Appendix A för fullständig transkription]

[*Ja, Open Geospatial Consortium (OGC) standarderna du nämnde - WMS (Web Map Service), WFS (Web Feature Service), WCS (Web Coverage Service), SOS (Sensor Observation Service), OGC API - Features, och WPS (Web Processing Service) - anses alla vara "öppna standarder."*]

All information som hämtas från ChatGPT jämförs med andra tillförlitliga källor, i synnerhet OGC:s specifikationer, i syfte att styrka ChatGPT:s påståenden. OGC:s standarder är alla öppna standarder, vilket bekräftas av OGC:s hemsida [OGC, 2024a]. Krav U.2, om etablering och godkända verktyg, ger först ett sammanställt svar för samtliga standarder och kräver ytterligare uppmaning för ChatGPT att bedöma standarderna individuellt. OGC:s standarder är välkända inom den geospatiala sfären och både antalet certifierade implementeringar samt flertalet

studier vittnar om detta faktum. De nyare standarderna inom OGC API-familjen är uttalat av OGC att de bygger vidare på de äldre standardernas tradition med syfte att vara mer moderna och enklare att implementera [OGC, 2024b]. ChatGPT:s svar till krav U.3, om webbaserad, styrks av att samtliga standarder definieras av OGC som standarder för webbaserade gränssnitt.

Kravet om interoperabilitet, U.4, ger först en sammanställd lista med gemensamma faktorer som kan påverka. För att avgöra huruvida ChatGPT listar olika faktorer beroende på standard ställs sedan frågan separat för varje standard. Emellertid noteras att ChatGPT i mycket upprepar samma påverkansfaktorer för samtliga standarder, vilket rationaliseras av att det är generella faktorer som till viss grad alltid påverkar. OGC listar inte dessa faktorer någonstans, men ett antal studier stödjer ChatGPT:s påstående.

Krav om stöd för flera datakällor, U.5, står inte uttryckligen i standarderna, men beskrivs översiktligt i OGC:s självstudiemoduler. Här används ChatGPT:s svar som förtydligande att hänvisningen i självstudiemodulerna refererar till att varje standard kan stödja flera datakällor. Krav U.6, om tidseffektivitet och aktuella format, uttrycks inte heller i klartext i standardernas officiella dokumentation och vilka format som kan stödjas specificeras inte. Vissa av standarderna har minimikrav, exempelvis WMS rekommenderar PNG och WFS har som minimikrav GML [OGC, 2006; 2010]. ChatGPT listar vilka format som kan och vanligen stöds av varje standard och huruvida de anses aktuella, samt ifall standarderna anses tidseffektiva. Studier såsom Giuliani et al. (2013) bedöms styrka ChatGPT:s svar. Standarderna specificerar inte heller huruvida de kan hantera simultana anrop eller stora datamängder, krav U.7, men till exempel ovannämnda Giuliani et al. hänvisar till simultana anrop i sina tester av prestanda. ChatGPT:s svar bedöms därmed tillförlitligt, även i fråga om Features, i och med att OGC API:erna är designade för att vara mer i linje med modern webbarkitektur än de tidigare [OGC, 2024b]. Utifrån det material som tillämpats i denna studie har inte kunnat fastställas huruvida SOS äger dessa egenskaper, men med ChatGPT:s svar som tillförlitligt för resterande standarder bedöms det också svara för SOS.

Sammanställning för stöd för hämtning av rådata, krav A.1, underlättas av ChatGPT som tydliggör vilka standarder som kan tillämpas för nedladdning. Detta bekräftas även av OGC:s specifikationer och implementationer online. Standardernas tekniska specifikation noterar inte i klartext vad för filtrering som stöds, krav A.2, men exempelvis självstudiemodulerna styrker ChatGPT:s påståenden. För krav A.3, om koordinatsystem, noteras att flera av standarderna inte specificerar vilka koordinatsystem som stöds utan endast kräver att det ska finnas stöd för åtminstone ett koordinatsystem. Utifrån detta tolkas ChatGPT:s påstående om att stöd för specifika koordinatsystem beror på implementering av server, och att samtliga standarder därmed har möjlighet att uppfylla krav A.3, som tillförlitligt [OpenAI, 2024]. Däremot noterar ChatGPT samma sak för Features, som enligt specifikationen för del 1 (Kärnstandarden) endast stödjer

WGS 84 (EPSG:4326). Det är det enda tillfället som ChatGPT tydligt tycks ha fel under fallstudiens genomförande, men som bedöms bero på att ChatGPT inkluderar Features del 2 i sitt svar.

Krav A.4, om att vara lätt att använda, bedöms uppfyllas fullt ut av alla standarderna baserat på ChatGPT:s svar, då OGC-standarder är väl implementerade. Standarderna specificerar väldigt lite obligatoriska metadata, krav A.5, men ChatGPT redogör för de vanligaste för varje standard, med betoning på att vad för metadata som finns tillgänglig är upp till implementering.

4.3 Resultat

Tabell 4.1 presenterar en sammanställning av de jämförda standarderna och hur väl de uppfyller kraven definierade i avsnitt 3.2.

Tabell 4.1: Sammanställning av standarder och kravuppfyllnad för kravlistan

Grön = Uppfyller kravet. Röd = Uppfyller inte kravet. Gul = Uppfyller delvis kravet.

** = Uppgift från ChatGPT som bedömts trovärdigt. ** = Bedöms likvärdigt med liknande standarder.*

Krav	WMS	WFS	WCS	SOS	EDR	Features	Records	WPS
U.1								
U.2								
U.3								
U.4	*	*	*	*	**	*	**	*
U.5							**	
U.6					**		**	*
U.7				*	**	*	**	
A.1								*
A.2								
A.3	*	*	*	*	*		*	*
A.4	*	*	*	*	**	*	**	*
A.5	*	*	*	*	**	*	**	*

Samtliga OGC-standarder är öppna standarder och uppfyller därmed krav U.1 [OGC, 2024a]. De äldre OWS-standarderna är väletablerade med många certifierade verktyg, bland annat *GeoServer*, *MapServer* och *QGIS*. De uppfyller därmed till fullo krav U.2, om etablering och enkel implementation. SOS och WPS

bedöms uppfylla kravet till viss del då de är etablerade, men med hänvisning till det låga antalet certifierade implementeringar tillhörande respektive standard (se avsnitt 2.3.5 och 2.3.9) bedöms de inte uppfylla kravet fullt ut. EDR och Features uppskattas också endast delvis uppfylla kravet. De tillhör OGC API-familjen som är designade för att vara enklare att implementera än de äldre standarderna i syfte att underlätta för vem som helst att leverera och använda geospatiala data på webben [OGC, 2024b]. Dock är de fortfarande relativt nya med få implementeringar och OGC API:erna anses ännu inte vara lika väletablerade som de äldre standarderna [Santana & Davis Jr, 2023]. Records är ännu inte färdigställd och anses därmed inte uppfylla kravet alls, trots att det redan finns implementerat stöd för standarden enligt Simoes & Ceciello (2022). Krav U.3, om webbaserad, uppfylls av alla standarder [OpenAI, 2024].

Kravet om interoperabilitet, U.4, bedöms inte uppfyllas fullt ut av en enda standard. ChatGPT listar ett antal faktorer som kan påverka interoperabiliteten mellan en server och en klient som implementerar samma standard. Däribland hur väl standardens specifikationer åtföljts, vilken version som implementerats och vilka operationer samt dataformat som stöds. Säkerhetsmekanismer som auktorisering och autentisering kan också påverka interoperabiliteten. För WCS kan även en faktor vara vilken utvidgning av kärnstandarderna som implementerats, medan WPS påverkas av att processbeskrivningar måste överensstämma mellan klient och server [OpenAI, 2024]. Vanmeulebrouk et al. (2009) hävdar att interoperabilitet lättare nås i system där både klient- och serversida utvecklas av samma organisation, bland annat då det då går att implementera och förlita sig på tillvalsfunktioner från standarden. Giuliani et al. (2013) noterar också att interoperabilitet kan påverkas av ifall implementation skiljer sig mellan klient och server. Det stämmer således med påståendet från ChatGPT om att interoperabilitet försvåras av att server och klient kan ha implementerats olika noggrant eller i olika versioner. EDR, Features och Records har för närvarande inga andra versioner och kan argumenteras att de därmed i högre grad uppfyller kravet men har här valts att ändå markeras som delvis uppfyllt.

Alla standarder uppfyller kravet om stöd för flera datakällor, krav U.5, dock begränsat efter vilka format varje standard är anpassad för. WMS kan leverera kartbilder från flera datakällor, antingen som separata kartbilder eller som lager i en enda kartbild. Det är även möjligt att integrera data från flera WMS-servrar till en kartbild [OGC, 2024i]. WFS tillåter att användaren begär ut data från flera WFS-servrar och tillåter även användaren att begära ut data oavsett vilken datakälla det kommer ifrån [OGC, 2024j]. Även WCS möjliggör för klienten att begära ut data från flera WCS-servrar, samt hämta data över ett nätverk av datakällor [OGC, 2024k]. SOS tillåter hämtning av data från olika sensorer och sensorsystem [OGC, 2024l]. Både Features och EDR (och därmed antas även Records), tillåter klienter att begära ut data från flera implementationer av API:et och stödjer flera datakällor genom Samlingar [OGC, 2024c; 2024e]. WPS kan

hämta data från flera datakällor i syfte att kombinera eller på annat sätt transformera data [OGC, 2024h].

Krav om tidseffektivitet och aktuella format, U.6, bedöms endast uppfyllas till fullo av de nyare standarderna. Standarderna i OGC API-familjen följer en modernare webbarkitektur och stödjer mer aktuella format. Jämförelse mellan Features och WFS resulterar i klar fördel för API:t i fråga om prestanda, där aktuella filformat är en vägande faktor [Santana & Davis Jr, 2023]. Enligt ChatGPT kan samtliga standarder anses tidseffektiva, men att resurskrävande och komplexa data framförallt påverkar svarsfrekvens och prestanda negativt hos de äldre standarderna WMS, WFS, WCS och SOS. Giuliani et al. (2013) testar prestandan hos WMS, WFS och WCS och rekommenderar bland annat effektivisering av data och nedskalning av antal anrop som sker simultant för att säkerställa bättre prestanda. Prestandan hos WPS påverkas av hur komplexa processer som krävs av servern [Cannata et al., 2014].

WMS stödjer bildformat såsom PNG, JPEG och GIF, vilka är fortsatt aktuella inom bildrendering [OGC, 2006]. WFS stödjer XML-format såsom GML, som inte längre anses vara ett aktuellt filformat [Allen, 2020]. WCS stödjer olika moderna format som varierar beroende på implementering, men där GeoTIFF och NetCDF är vanliga format [OpenAI, 2024; Thredds Data Server, 2020]. SOS stödjer de XML-baserade formaten Observation and Measurements (O&M) och Sensor Model Language (SensorML), båda vilka är fortsatt användbara inom sensordata, men som inte klassas som aktuella utanför detta område [OpenAI, 2024; OGC, 2024i]. WPS stödjer vanligen XML-baserade format, men det regleras inte i standarden utan beror på implementering [OpenAI, 2024; OGC, 2018b].

WMS, WFS, WCS och WPS kan alla hantera flera simultana anrop och stora datamängder, krav U.7, men prestandan blir sämre och det rekommenderas att begränsa både anrop och datamängd om möjligt [Giuliani et al., 2013; Sun, Liu et al., 2016]. ChatGPT rapporterar ovanstående även för SOS, men hävdar att Features moderna arkitektur är designad för stora volymer av data och hantering av simultana anrop och därmed bättre uppfyller kravet [OpenAI, 2024]. Detsamma antas därmed gälla för EDR och Records.

Krav A.1, stöd för hämtning, uppfylls till fullo av WCS och EDR [Thredds Data Server, 2020; OGC, u.å.]. WMS erbjuder inte nedladdning av rådata [OGC, 2024i]. WFS specificerar endast nedladdning i GML, även om stöd för andra vektorformat såsom GeoJSON kan implementeras. Features stödjer också endast nedladdning i vektorformat [pygeoapi, 2024a; OpenAI, 2024]. SOS kan implementeras så att NetCDF stöds, men görs det vanligen inte [OpenAI, 2024; Sorg & Kunkel, 2015]. ChatGPT klassar även WPS som möjlig att, med implementering, stödja hämtning i NetCDF, men att WPS främst används för att processa data snarare än nedladdning. Records är designad för sökning efter metadata och uppfyller därmed inte kravet [OGC, 2024g].

Stöd för filtrering och definiering av delmängder utifrån geometriska punkt-, linje- och ytoobjekt, krav A.2, bedöms inte uppfyllas av WMS då standarden är designad för att leverera förproducerade kartor - det är dock möjligt att hämta information kring en specifik punkt genom operationen *GetFeatureInfo* [OGC, 2006] och även att implementera filtrering baserat på geografiskt område [Allen, 2020]. WFS, Features och EDR uppfyller kravet till fullo - WFS behöver att användaren filtrerar data då det annars kan anstränga servern för mycket och Features samt EDR stödjer filtrering genom *Query*-operationer [Allen, 2020; OGC, 2023a; 2019]. WCS stödjer filtrering baserat på "spatiala begränsningar och andra kriterier", men för filtrering på punkt-, linje- och ytoobjekt måste specifik implementering göras [OGC, 2012a; OpenAI, 2024]. SOS stödjer både spatial och temporal filtrering, men likt WCS måste implementering göras för att kunna filtrera på geometriska objekt [OGC, 2012b; OpenAI, 2024]. Records erbjuder filtrering för metadata och alltså inte rådata [OGC, 2024g]. WPS kan filtrera data ifall implementering har gjorts av en sådan process [OGC, 2018b; OpenAI, 2024].

Lokaliserade rådata, krav A.3, uppfylls enligt ChatGPT av samtliga standarder, där vilka koordinatsystem som stöds beror på implementering av server. OGC specificerar inte i standarderna vilka koordinatsystem som måste stödjas utan, till exempel för WMS, regleras det att i alla fall ett koordinatsystem måste stödjas [OGC, 2006]. Undantaget är Features, där kärnstandard (alltså Del 1) explicit endast stödjer WGS 84 (EPSG:4326) [OGC, 2019]. ChatGPT:s påstående stämmer således inte för Features.

Syftet med OGC:s standarder är att förenkla utbytet av geospatial information över webben. Implementering av standarderna kräver viss teknisk kunskap, men i övrigt är alla OGC:s standarder designade för att vara lätta att använda. De uppfyller därmed alla krav A.4 [OpenAI, 2024]. Krav om metadata, A.5, bedöms inte uppfyllas till fullo av någon standard, då ingen av standarderna specificerar tillgängliga prognosmodeller eller parametrar som krav i sin metadata. Det kan dock implementeras vid behov. Datakälla, geografiskt område och tidsperiod är mer förekommande metadata än vad prognosmodeller och parametrar är [OpenAI, 2024].

Sammanfattningsvis, klargör tabell 4.1 att EDR bäst uppfyller kravställningen med flest antal krav till fullo och alla krav delvis eller till fullo uppfyllda. WCS, WPS och SOS har heller inga krav som inte uppfylls, men WCS uppfyller fler krav till fullo än WPS och SOS med sju respektive fem helt uppfyllda krav. WFS uppfyller också fler krav till fullo än WPS och SOS, med sju stycken uppfyllda och fyra krav delvis, men har ett krav som inte uppfylls alls (A.1). Features och Records är jämbördiga med WCS och WFS i antalet till fullo uppfyllda krav, men fallerar i att de har två krav som inte uppfylls alls. Sämst uppfyller WMS kravställningen, med sex krav uppfyllda till fullo och fyra delvis, men två krav som inte alls uppfylls. En

sammanfattning av antalet krav som uppfylls, uppfylls delvis eller inte uppfylls alls presenteras i tabell 4.2 nedan.

Tabell 4.2: Sammanställning av standarders uppfyllnadsgrad av kravlistan

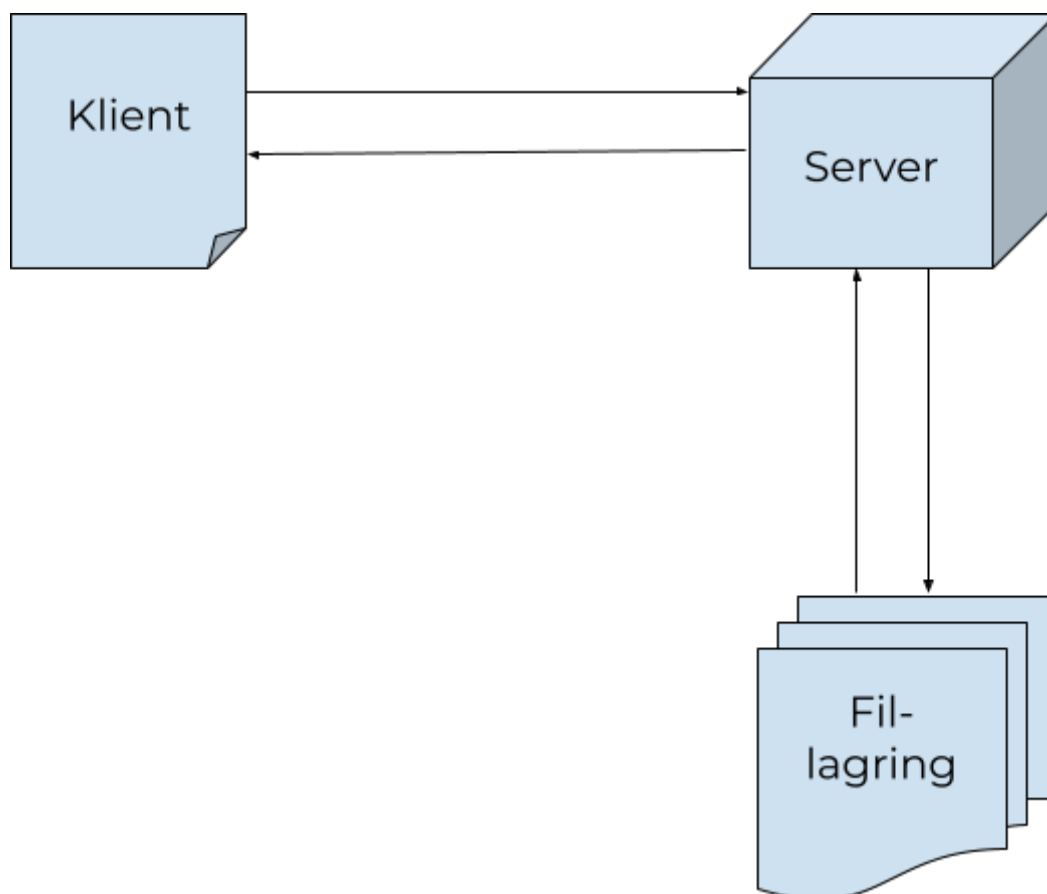
Krav	WMS	WFS	WCS	SOS	EDR	Features	Records	WPS
Uppfyllda till fullo	6	7	7	5	9	7	7	5
Uppfyllda delvis	4	4	5	7	3	3	3	7
Inte uppfyllda alls	2	1	0	0	0	2	2	0
Totalt	12							

5 Fallstudie 2 - Praktisk implementering

Fallstudie 2 syftar till att praktiskt implementera EDR. Implementering görs av en EDR-klient för hämtning av urval av rådata från en server. Fallstudien baseras på användarfall och kravställning presenterade i kapitel 3. Applikationens systemarkitektur beskrivs i avsnitt 5.1. Avsnitt 5.2 motiverar val av verktyg, därefter redovisas implementering i avsnitt 5.3. Implementeringen bedöms på hur väl applikationen motsvarar användarkraven (se avsnitt 3.2.2) och utvärderas i avsnitt 5.4. Fallstudien avslutas med val av licens för fortsatt distribution av implementerad kod i avsnitt 5.5.

5.1 Systemarkitektur

Systemarkitekturen har till grund en väderapplikation såsom presenteras i avsnitt 2.2. Applikationen är uppdelad i tre komponenter: en klient, en server samt lokal fillagring (istället för en kopplad databas). Figur 5.1 visar applikationens arkitektur.



Figur 5.1: Systemarkitektur för en väderapplikation utformad efter standarden EDR

Klienten är en applikation som kan köras i en webbläsare. Syftet med klienten är att leverera ett användarvänligt gränssnitt för att filtrera ut och ladda ned urval av

rådata. Design för klienten väljs efter detta ändamål för fallstudiens genomförande.

Vald design implementerar ett grafiskt användargränssnitt med en världskarta. Kartvyn syftar till att visualisera det geografiska området efter val av datasamling samt möjliggöra för användaren att rita upp önskat område av data direkt i kartan. En informationspanel till vänster om kartan presenterar en lista med de tillgängliga datasamlingarna för nedladdning från servern. Användare erbjuds, efter val av samling, att visa och ladda ned tillgängliga metadata, ladda ned alla data tillhörande den valda samlingen eller filtrera på önskat område, tidsintervall och/eller parametrar. Klienten bygger på EDR och utvecklas i programmeringsspråket *TypeScript*.

Serverkomponenten har till syfte att hantera och publicera tillgängliga datasamlingar. Servern lyssnar på inkommande förfrågningar från klienten och läser in datafiler som motsvarar förfrågan. Dessa konfigureras och mappas av servern och levereras sedan till klienten. Servern utvecklas i programmeringsspråket *Python*.

Applikationens data består av lokalt lagrade filer. Samspel mellan fillagring och server sker genom inläsning av filsystemet. Mellan server och klient sker kommunikation över HTTP-protokoll.

5.2 Val av verktyg

Detta avsnitt ämnar att kortfattat motivera de verktyg som används i fallstudie 2.

5.2.1 Server

För serversidan implementeras *pygeoapi*. *pygeoapi* är en *Python*-baserad HTTP-server som implementerar OGC API-familjen. Det är ett projekt med öppen källkod under en MIT-licens, som startades 2018 i nära relation med OGC API. *pygeoapi* är verifierat av OGC som en referensimplementation för både Features, EDR och Tiles [pygeoapi, 2024b]. Projektet är populärt för hantering av geospatiala data och med ett antal aktiva utvecklare (103 så kallade "medverkande") i GitHub-kanalen [Geopython, u.å.]. Bland annat Simoes & Cerciello (2022) använder *pygeoapi* i sin studie, med slutsatsen att det är "en bra lösning för att implementera flera OGC API-standarder" tack vare dess flexibilitet och lätta användning. Även Blackburn et al. (2023) använder *pygeoapi* i deras MEDIN-projekt för implementering av EDR.

pygeoapi är för närvarande den enda certifierade implementeringen för EDR och därmed motiverad att använda i denna fallstudie.

5.2.2 Klient - applikationsramverk

Klienten ansvarar för kommunikation mellan klient och server samt ska möjliggöra för användare att filtrera data genom ett formulär. För implementering av denna funktionalitet i klienten används *Angular*. *Angular* är ett webbapplikationsramverk för närvarande i version 18 [Angular, 2024]. Det bygger på kodspråket *TypeScript*, som är en *JavaScript*-dialekt med mera stöd för typer. *Angular*-ramverket öppnar upp för en komponent-rik applikation med stor skalbarhet. Det är ett av de populäraste ramverken som används idag och har fördelar av att vara anpassningsbar och med god prestanda. Användningsområden för *Angular* är bland annat användargränssnitt, enkelsidiga applikationer, företagslösningar och storskaliga projekt [Taylor, 2023]. *Angular* är öppen källkod under en MIT-licens, utvecklat främst av Google med möjlig kontribution till källkoden av andra genom GitHub. 1 950 utvecklare står där som registrerade "medverkande" och 3,6 miljoner står på GitHub som användare av ramverket [Angular, u.å].

Ett antal ramverk för webbapplikationer kan nyttjas för denna fallstudie, däribland *React* som är ett JavaScript-bibliotek med mycket flexibilitet. Motivering för användning av *Angular* kommer framförallt av goda personliga erfarenheter av detta verktyg och att det därmed är ett välbekant verktyg med kortare inlärningsperiod.

5.2.3 Klient - användargränssnitt

OpenLayers är ett verktyg för interaktiva kartor på nätet. Genom *OpenLayers* kan funktionalitet som till exempel animering, in- och utzoomning samt modifiering av attribut implementeras och skapa dynamiska kartor kodat i JavaScript. Det är fritt tillgängligt och öppen källkod under en BSD-2 Clause licens, med 371 "medverkande" på GitHub [OpenLayers, u.å].

OpenLayers används i fallstudien i syfte att skapa en användarvänlig och grafiskt tillfredsställande karta för visualisering av det geografiska området av applikationens tillgängliga dataresurser.

5.3 Implementering

Detta avsnitt ämnar att kortfattat beskriva den implementering som gjorts av valda verktyg. Implementering av applikationens komponenter sker lokalt på samma dator, men är tänkt att fungera i en distribuerad miljö. Kommunikation mellan klient och server sker via HTTP-protokoll lokalt på datorn.

5.3.1 Server

Implementering och konfigurering av *pygeoapi*-servern sker enligt instruktioner på *pygeoapis* hemsida [pygeoapi, 2024c]. Konfigurationsfilen,

example-config.yml, uppdateras för att lägga till testdata och korrekta kontaktuppgifter. Tillagda dataresurser beskrivs nedan i avsnitt 5.3.3. För att publicera data till EDR för *pygeoapi* krävs installation av en plug-in EDR-leverantör, *Xarray-edr*. *Xarray-edr* beror på *Python*-paketet *Xarray* som hämtas från *Xarrays* hemsida [pygeoapi, 2024d; Xarray, 2024].

5.3.2 Klient

Angular används för att bygga klientsidan av applikationen. Klienten består av komponenter som ansvarar för kartvyn (inklusive en funktion för användaren att rita upp ett geografiskt område) samt för kommunikation till server och användarinmatning. Komponenten för karta och ritning i karta inspireras av kod från *OpenLayers*. Kod för denna komponent återfinns i Appendix B.

Interaktion mellan klient och server sker med hjälp av en service som ansvarar för skapandet av korrekta anrop till servern. Komponenten använder dessa anrop i enlighet med användarens begäran om antingen visning av metadata, nedladdning av metadata, nedladdning av en hel datasamling eller filtrering av data. Metoder i denna komponent återfinns i Appendix C.

5.3.3 Fillagring

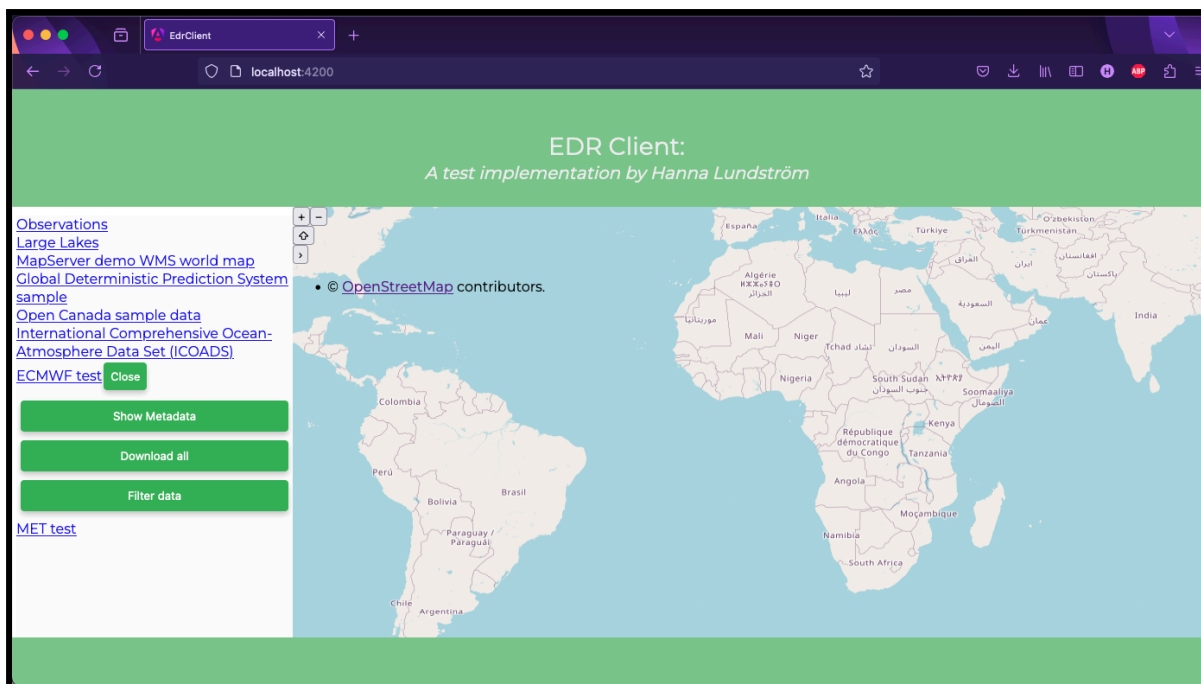
Fallstudiens server testas gentemot lokalt lagrade filer. Testdata i NetCDF-format, bestående av observationer av världshavens yttemperatur, lufttemperatur, zonvind och meridional vind, hämtas från *pygeoapis* GitHub-kanal utifrån exempel på hantering av EDR-dataresurser [Geopython, u.å.].

Därefter laddas öppna data från ECMWF:s webbplats ned [ECMWF, u.å.-d]. De testdata består av data från ECMWF:s IFS-modell, specifikt vågdata såsom genomsnittlig vågriktning, genomsnittlig vågperiod och högsta vågperiod. Data från ECMWF laddas ned i GRIB-format och konverteras till NetCDF genom ECMWF ecCodes mjukvara [Koehler, 2024].

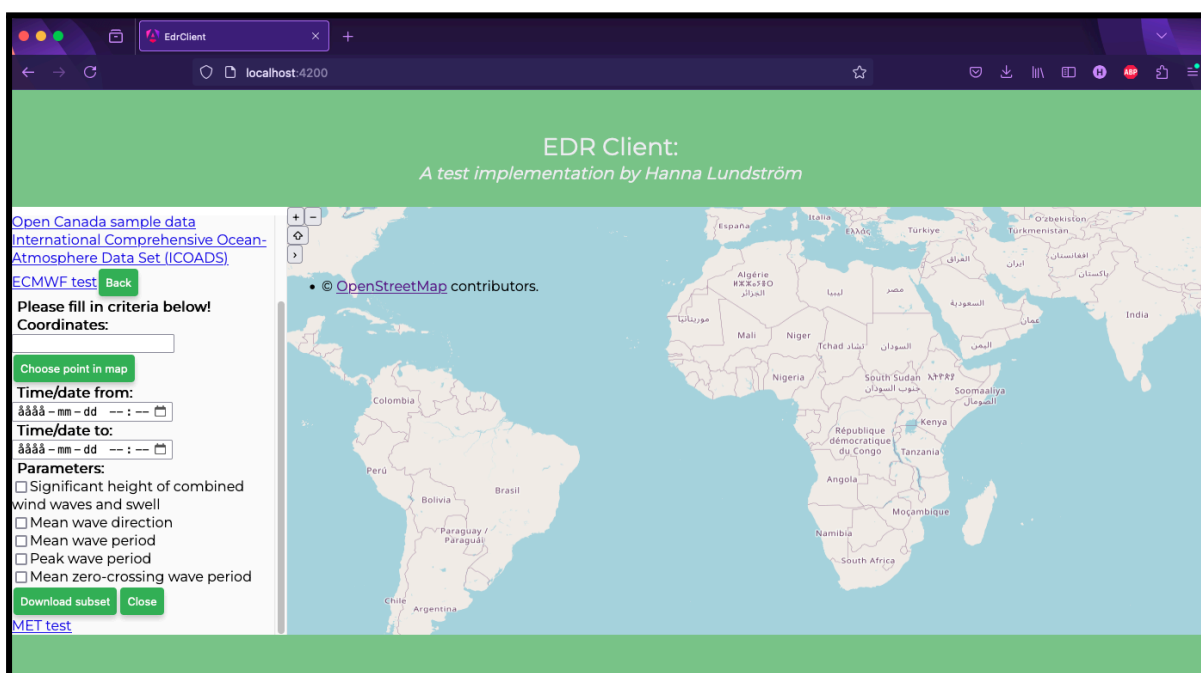
Ytterligare testdata hämtas från MET, specifikt data från deras AROME-modell för Arktis. Data består av sannolikhetsprognoser för extremväder [MET, u.å.].

5.4 Utvärdering

Applikationens slutliga design presenteras nedan. Figur 5.2 visar klientens tillgängliga val för användaren för vald datasamling (här har *ECMWF test* valts av användaren). Figur 5.3 visar filtreringsmöjligheter utifrån den valda datasamlingen efter val av specifikt en Position-query. Liknande filtreringsmöjligheter finns vid val av en Kub-query, men då ombeds användaren välja ett geografiskt område och inte koordinater för en punkt.



Figur 5.2: Användarens alternativ vid val av datasamlingen *ECMWF test* i implementerad väderapplikation



Figur 5.3: Användarens alternativ för filtrering vid en *Position*-query i implementerad väderapplikation

Applikationen utvärderas gentemot kravställningen i avsnitt 3.2.2. Av de ställda användarkraven, bedöms endast kravet om att vara lätt att använda, krav A.4, uppfyllas till fullo. Krav A.1, om stöd för hämtning av prognosdata, är beroende av serverns kapaciteter. Applikationen är utformad för att uppfylla kravet om nedladdning i NetCDF eller GRIB, men *pygeoapi* stödjer i skrivande stund inte

hämtning i något annat format än JSON, HTML eller JSONID. Krav A.2, om filtrering, är även det implementerat på klientsidan, men begränsas av att *pygeoapi* för närvarande endast implementerat anrop för EDR i form av metadata, Position och Kub. Linjeobjekt är alltså inte möjligt att filtrera på. EDR:s standard stödjer ytterligare anrop för Area, Bana och Korridor, men detta är ännu inte implementerat av *pygeoapi*.

Krav A.3, om koordinatsystem, beror även det på *pygeoapis* serverimplementation, som för närvarande endast stödjer WGS 84 (motsvarande EPSG:4326). Krav A.5, om metadata, bedöms uppfyllas delvis då den önskade informationen är möjlig att implementera men beror på tillgängliga metadata från testdata. Krav A.1, A.2 och A.3 noteras att de ej uppfylls till fullo på grund av förutsättningarna hos *pygeoapi*, men att det enligt OGC:s standarddokumentation för EDR bör vara möjligt att implementera.

En ytterligare begränsning i fallstudien är att *pygeoapi* endast har en leverantör av data för EDR, *Xarray-edr*, som inte stödjer något annat format än NetCDF. Öppna väder- och prognosdata är för det mesta i GRIB- eller GRIB2-format och till exempel ECMWF:s öppna data kunde således först användas efter konvertering från GRIB till NetCDF. Försök gjordes att testa öppna data från MET, men gick inte att konfigurera då servern misslyckades att läsa vissa attribut. Hämtning fungerade endast vid nedladdning av specificerade attribut i en Position-query. Både *pygeoapis* och ECMWF:s testdata är oceanografiska och inte strikt meteorologiska, men ingår i definitionen av väderdata och är i storlek och volym komplexa och multidimensionella likt meteorologiska data.

Sammanfattningsvis bedöms applikationen begränsas av funktionaliteten i *pygeoapi*, men *pygeoapi* är i dagsläget fortfarande den enda certifierade implementeringen av EDR på OGC:s hemsida.

5.5 Distribuering av kod

Kod som implementeras i fallstudie 2 distribueras öppet under en MIT-licens på GitHub¹. En MIT-licens innebär att användaren tillåts fritt nyttja, kopiera, ändra, sprida och publicera mjukvara. Kod under en MIT-licens kan distribueras vidare under valfri licens, så länge som den ursprungliga MIT-licensen och copyrightmeddelandet inkluderas där kod använts. Val av licens motiveras av att både *pygeoapi* och *Angular* delar sin kod under en MIT-licens. *OpenLayers* distribuerar sin kod under en BSD 2-Clause licens, som även den är en relativt enkel licens och tillåter användaren fri tillgång till mjukvara så länge som en kopia av licensen inkluderas. Kod som är under en BSD-2 Clause licens tillåts att ingå i ett program under MIT-licens, så länge BSD-2 Clause licensen är inkluderad i de filer där koden används.

¹ <https://github.com/ha6544lu/EDR-Client-Test>

Del IV
Diskussion och slutsatser

6 Diskussion

Diskussionen är uppdelad i tre delar. Först diskuteras resultatet av fallstudie 1, som efterföljs av diskussion rörande resultatet av fallstudie 2. Därefter avslutas kapitlet med en sammanställd, generell diskussion.

6.1 Fallstudie 1 - Teoretisk jämförelse

Resultatet från fallstudie 1 visar att EDR bäst uppfyller kravställningen. Ur tabell 4.1 och 4.2 kan konstateras att EDR uppfyller flest antal krav till fullo jämfört med de andra OGC-standarderna och att EDR uppfyller alla krav helt eller delvis. Fallstudie 1 syftar till att teoretiskt jämföra EDR med likvärdiga standarder för datadelning, främst av prognosdata. Krav A.1, stöd för hämtning av prognosdata, relaterar specifikt till detta syfte och uppfylls till fullo endast av en standard, WCS, utöver EDR. SOS och WPS bedöms uppfylla kravet delvis, efter implementering av sådan funktionalitet. Standarder som WFS och Features stödjer nedladdning av rådata, men inte i här önskat format.

Samtliga krav bedöms här vara likvärdiga och ha samma prioritet. I relation till fallstudiens syfte kan dock diskuteras ifall krav A.1 bör vara av högre eller högsta prioritet och att de standarder som då inte uppfyller kravet direkt kan avskrivas. Viktning hade kunnat göras av kravlistan för att markera krav A.1:s betydelse. Kraven hade kunnat tilldelas olika prioriteter och därmed tydliggöra vad i kravställningen som är ett behov och vad som är en önskan. Det hade besvarat frågor som till exempel ifall det är acceptabelt att förlora tidseffektivitet (krav U.6) i utbyte mot att det är möjligt att filtrera små delmängder (krav A.2). Argument kan göras för att krav som då A.1, men även krav U.2 (om etablering och verktyg), krav U.4 (om interoperabilitet) och krav A.2 (om filtrering), är av större vikt för den typ av väderapplikation som utvärderas för här. En viktning av kravlistan hade även gett mer betydelse till resultatet av jämförelsen. Ifall EDR fortsatt hade bäst uppfyllt kraven, även efter viktning, hade det gett en större trovärdighet till EDR som bäst lämpad för användning i en väderapplikation för delning av väder- och prognosdata.

Med hänsyn till uppfyllnad av krav A.1, synes WCS som bäst jämförbar med EDR. WCS skiljer sig från EDR i krav U.2, U.6, U.7 och A.2, där endast krav U.2, om etablering, står till WCS:s fördel. EDR uppfyller detta krav endast delvis. WCS är mer etablerad och har därmed större tillgång till verktyg än vad EDR har, vilket kan vara en avgörande faktor beroende på ändamål för implementering. Krav U.6 och U.7 är båda relaterade till prestanda och kapacitet. Studier som Giuliani et al. (2013), Sun et al. (2016b) och Santana & Davis Jr (2023) rapporterar att den moderna arkitekturen hos OGC API presterar bättre än den traditionella arkitekturen hos exempelvis WCS. De ursprungliga OWS-standarderna är fortsatt funktionella, men långsammare och mer resurskrävande på servern än de nyare API:erna. EDR är mer anpassad till modern webbarkitektur än WCS och

uppskattas därmed vara bättre lämpad för fortsatt utveckling. Det kan argumenteras att resultatet i fallstudie 1 pekar på EDR som en framtida investering och WCS som ett välbeprövat och säkrare alternativ för närvarande.

WFS och Features erbjuder nedladdning, men i fel format. De är inriktade på vektordata, medan kravställningen är fokuserad till rasterformat. Efter EDR och WCS kan argumenteras att WFS och Features bäst uppfyller kravställningen, som ett alternativ för nedladdning av väderdata i vektorformat. De påminner mycket om både WCS och EDR, och påvisar samma val mellan det traditionella och det moderna alternativet.

SOS och WPS uppfyller delvis kravet om nedladdning, A.I. SOS är dock en standard specifikt riktad mot sensorer och WPS är en standard för databearbetning. Båda dessa standarder har möjlighet att uppfylla fallstudiens syfte om datadelning, men är inte huvudsakligen utarbetade för nedladdning och underlättar inte nämnvärt nedladdning av prognosdata i större skala. Det kan därmed konstateras att med nedladdning som huvudsyfte i en applikation, är inte det mest lämpliga valet implementering av SOS eller WPS. Däremot kan det vara ett alternativ för en applikation som erbjuder nedladdning vid sidan av. Det bör dock noteras att både SOS och WPS endast delvis uppfyller krav U.2 i och med att SOS inte har några certifierande implementeringar och WPS endast en.

WMS och Records erbjuder inte nedladdning av rådata överhuvudtaget. De uppfyller därmed inte det grundläggande kravet i kravställningen och det kan diskuteras om dessa standarder då är likvärdiga nog med EDR för att vara bra jämförelsematerial. I synnerhet WMS och Records kan argumenteras för att de är för skilda åt från EDR i syfte, men även de andra standarderna kan ifrågasättas om de är rätt standarder för en tillförlitlig jämförelse. Vidare jämförs endast OGC-standarder med motiveringen att OGC är ledande i att skapa geospatiala standarder. Några av OGC:s standarder, exempelvis WMS och WFS, har en motsvarande tillhörighet hos ISO. ISO utvecklar ett antal geodatastandarder i serien ISO 19100 som hade kunnat vara ett alternativ till de standarder som jämförts i denna studie. Det finns dock ingen annan organisation eller företag som skapar öppna och etablerade geospatiala standarder på den nivå OGC gör.

För fallstudiens genomförande har verktyget ChatGPT använts. Användning av ChatGPT är komplicerat då det är svårt att säkra trovärdigheten hos ChatGPT som källa. All information som ChatGPT har producerat för fallstudien har dock jämförts och bearbetats med andra källor. Sedan fallstudiens genomförande har ChatGPT även genomgått en uppdatering och har numera möjlighet att besvara frågor gällande EDR och Records. En snabb genomgång av kravlistan i relation till EDR och Records tyder på att de antaganden som gjorts i fallstudie 1 stämmer överens med vad ChatGPT:s senaste uppdatering säger. En generell jämförelse mellan svaren använda i fallstudie 1 och svaren som fås av den uppdaterade versionen anses allmänt vara tillräckligt lika för att ge ytterligare stöd till

resultaten för fallstudie 1. Det är dock fortfarande osäkert hur mycket formuleringen av frågor spelar in i vad för svar som erhålls av ChatGPT. För användningen av ChatGPT har kraven försökts formuleras så sakligt och objektivt som möjligt. Det kan emellertid argumenteras att kraven är formulerade så detaljerat att ChatGPT kan gissa sig till vad för svar som eftersöks och således erbjuda just det svaret. Trots bearbetning av ChatGPT:s svar måste det tas i beaktning att en annan formulering hade kunnat ändra svaren och potentiellt även resultatet i fallstudie 1.

Fallstudiens genomförande och resultat är slutligen till viss mån en tolkningsfråga, där standardernas uppfyllnadsgrad av kraven kan argumenteras - ett exempel är huruvida endast certifierade implementeringar ska räknas till krav U.2, eller om det finns andra sätt att bedöma en standards etablering och ifall godkända verktyg finns att tillgå.

6.2 Fallstudie 2 - Praktisk implementering

I utvärderingen av fallstudie 2 konstateras att EDR inte uppfyller alla användarkrav ställda på applikationen. Varken krav A.1, A.2 eller A.3 är möjligt att implementera till fullo på grund av begränsningar hos servern. Fallstudien kan därmed endast utvärdera EDR i relation till vad *pygeoapi* stödjer. För att kunna diskutera EDR i sin helhet bedöms att en egen server hade behövt implementeras. Omfattningen av att implementera en egen server uppskattas i denna studie inte vara genomförbart, både på grund av bristande programmeringserfarenhet och studiens tidsrymd. Att trots detta använda *pygeoapi* motiveras dock av att det är den enda referensimplementation av EDR verifierad av OGC för närvarande. Begränsningarna i fallstudiens utförande kopplade till *pygeoapi* kan diskuteras vara typiska för standardens etablerande i skrivande stund och därmed motiverade för studiens syfte. Detta överensstämmer även med Blackburn et al. (2023):s svårigheter att fullfölja implementering av EDR och deras slutsats att standarden kräver investering av tid och resurser.

OGC listar på EDR:s GitHub-kanal ett antal implementationer som användes i utvecklingen av standarden. Utöver *pygeoapi*, nämns implementationer gjorda av UK Met Office, US National Weather Service, IBL Software Engineering, Wuhan University, ESRI Image Server facade och US Geological Survey. Ytterligare implementationer gjorda av Unidata och NASA väntas. Det nämns även att organisationer såsom ECMWF inväntar att standarden ska mogna innan de avsätter resurser [OGC, u.å.]. Trots att OGC för närvarande endast certifierat en implementation är det tydligt att intresset för standarden är stort. Utöver ovanstående ingår även Météo-France, Meteorological Service of Canada, Finnish Meteorological Institute och NASA i arbetsgruppen för EDR-standardens framtagande. Dessa intressenter tyder på att standarden kommer att fortsätta utvecklas framöver, framförallt för användning inom väder- och prognosdata.

För fallstudiens resultat ger *pygeoapis* server, trots dess begränsningar, en idé om hur EDR kan tillämpas i praktiken. I dagsläget stöds endast anrop för metadata, Position och Kub. Under fallstudiens genomförande noterades att anropen kräver att användaren är intresserad av ett specifikt område för att kunna hämta data. Att exempelvis hämta ut platser där temperaturen beräknas gå under noll, genom filtrering av data, är alltså inte möjligt utan att först bestämma vilken typ av objekt-anrop (Position eller Kub) som ska göras och därefter i sådant fall skicka förfrågan över samlingens hela geografiska område. Ett visst värdeintervall eller tidsintervall är inte möjligt att direkt filtrera på. Standarden möjliggör för anrop av typen Position, Area, Kub, Bana, Korridor, Föremål, Platser och Instanser, men har i den praktiska implementeringen endast kunnat testas för Position och Kub.

Det noterades även att anrop inte går att skicka över flera Samlingar, exempelvis om man är intresserad av prognosdata för en viss plats från olika datakällor. Det kan argumenteras huruvida detta är önskvärt för standarden i sig själv att stödja och hur ett svar på ett sådant anrop skulle se ut. Ett mer lämpligt alternativ kan vara att flera anrop till separata Samlingar skickas ut samtidigt vid användarinmatning.

Fallstudiens genomförande begränsades ytterligare av att *pygeoapi* endast stödjer en EDR-leverantör, *Xarray-edr*. Ursprungligen ämnades data från flera datakällor testas i applikationen. Detta försvårades av att *Xarray-edr* kräver att data är NetCDF-format medan tillgängliga väderdata är i GRIB-format.

6.3 Sammanställning och generell diskussion

Teoretisk och praktisk utvärdering av EDR avviker från varandra i resultat. Från fallstudie 1 kan konstateras att EDR är bäst lämpad för datadelning av prognosdata i relation till andra likvärdiga OGC-standarder. Fallstudie 2 visar dock att den praktiska implementeringen inte inkluderar hela standarden. I jämförelse med andra standarder noteras att EDR fortfarande är relativt ny och ännu inte särskilt etablerad. En avvägning mellan etablering och anpassning till modern webbarkitektur med möjlighet till bättre framtida utveckling måste således göras. Detta reflekteras i den praktiska implementeringen då tillämpning av den enda referensimplementering OGC certifierat just nu (*pygeoapi*) begränsar vad som går att praktiskt utvärdera för EDR.

För närvarande bedöms EDR vara svår att utvärdera praktiskt och för ytterligare utvärdering framåt krävs implementering utan *pygeoapi* där istället annan implementering görs, alternativt inväntan av att *pygeoapi* ska implementera större funktionalitet för EDR. Fallstudie 2:s resultat tydliggör att EDR må vara ett framtida alternativ med modern webbarkitektur, men att större stöd för standarden i stort krävs. Likväl tyder standardens intressenter på att fortsatt utveckling kommer att stödjas.

Den teoretiska jämförelsen noterar WCS som ett mer traditionellt alternativ till EDR. Relationen mellan WCS och EDR hade varit intressant att undersöka ytterligare med testning av de båda standarderna mer praktiskt. En fördjupning i fördelar och nackdelar mellan vektor- och rasterdata hade även kunnat inkluderas i en större undersökning av när WFS eller Features är bättre lämpat än WCS eller EDR. Resultatet av fallstudie 1 jämför standarderna endast översiktligt och diskussion kan föras kring om standarderna behöver praktisk testning för att bättre jämföras. Krav och resultat som framkommit ur denna studie kan även ifrågasättas baserat på hur de tolkats och bedömts.

7 Slutsatser

Detta examensarbete syftar till att teoretiskt utvärdera standarden OGC API - Environmental Data Retrieval (EDR) samt praktiskt implementera en väderapplikation baserat på EDR. Denna studie genomförs med hjälp av två fallstudier. Den första fallstudien ämnar teoretiskt utvärdera EDR i jämförelse med andra likvärdiga standarder. Totalt jämförs åtta standarder, där samtliga standarder är OGC-standarder. OGC är den ledande organisationen inom geospaciala datastandarder och studien har avgränsats till att fokusera endast på OGC:s standarder med motiveringen att deras standarder är likvärdigt utformade och de mest etablerade standarderna för geospaciala data.

Utgångspunkten för jämförelsen är EDR, en nyttillkommen standard från 2021, som ämnar modernisera och effektivisera datahämtning av geospaciala data. Ett specifikt användningsområde är prognosdata och väderdata i stora volymer. Utöver EDR undersöker fallstudie 1 standarderna OGC Web Map Service (WMS), OGC Web Feature Service (WFS), OGC Web Coverage Service (WCS), OGC Sensor Observation Service (SOS), OGC API - Features (Features), OGC API - Records (Records) och OGC Web Processing Service (WPS). WMS och WFS är två av de mest välkända standarderna för geospaciala visnings- respektive nedladdningstjänster. WCS är något mindre väletablerad, men designad för att tillgängliggöra multidimensionella data och används ofta i specifika fall såsom satellitbilsdata. SOS hanterar data från sensorer och sensorsystem, bland annat väderobservationer. WPS syftar till databearbetning. Features och Records utgår från samma kärnstandard som EDR (OGC API - Commons), där Features påminner om en modern version av WFS och Records hanterar metadata.

Jämförelsen utgår ifrån en kravställning och för varje krav avgörs huruvida standarderna uppfyller, uppfyller delvis eller inte alls uppfyller kravet. Bedömningen baseras på granskning av OGC:s officiella dokument och annat tillgängligt material, vetenskapliga artiklar och studier samt användning av verktyget ChatGPT. ChatGPT nyttjas som stöd för sammanställning av information och bekräftelse av funktionalitet som inte regleras specifikt i standarderna, men som hänvisas till i studier och artiklar. Fallstudiens resultat visar att EDR bäst uppfyller kravställningen. Det konstateras att EDR är ett modernt alternativ till datadelning för rasterdata och mest jämförbar med WCS. WCS tillhör de äldre OWS-standarderna och EDR bedöms därför mer anpassad till modern webbarkitektur med löften om ytterligare framtida utveckling. WCS är dock väletablerad medan EDR är för närvarande utan stabila verktyg. För framtida studier föreslås jämförande av EDR och WCS i praktiska tester.

Den andra fallstudien implementerar EDR praktiskt i en väderapplikation för hämtning av urval av rådata. Systemarkitekturen för applikationen är uppdelad i en klient, en server och lokalt lagrade filer. Samtliga komponenter implementeras lokalt på samma dator, men är ämnade att fungera i en distribuerad miljö.

Klienten levererar ett gränssnitt för användaren att enkelt filtrera ut och ladda ned urval av rådata, utvecklat i programmeringsspråket *TypeScript* med tillämpning av ramverket *Angular* och kod från *OpenLayers*. Servern hanterar och publicerar tillgängliga datakollektioner, samt lyssnar på förfrågningar från klienten, utvecklat i programmeringsspråket *Python* med implementering av *Python*-biblioteket *pygeoapi*. Kommunikation mellan klient och server sker via HTTP-protokoll lokalt på datorn. Data använda för fallstudien lagras lokalt och samspel mellan server och fillagring sker genom inläsning av filsystemet. Testdata består av exempeldata från *pygeoapi* samt öppna data från den internationella väderorganisationen ECMWF och norska meteorologiska myndigheten MET.

Fallstudiens genomförande begränsas av funktionaliteten i *pygeoapi*. Vid fallstudiens genomförande stödjer *pygeoapi* inte andra nedladdningsformat än JSON, HTML och JSONID. De enda implementerade anropen för EDR är metadata, Position och Kub. Endast en EDR-leverantör finns för publicering av data, *Xarray-edr*, som enbart hanterar data i NetCDF-format. *pygeoapi* är för närvarande den enda OGC-certifierade implementeringen av EDR och dessutom ansedd att vara en referensimplementation. Begränsningarna kopplade till *pygeoapi* i fallstudiens genomförande påvisar att EDR ännu inte är fullt etablerad som standard och det därmed inte finns tillförlitliga verktyg att använda för implementering. EDR bedöms vara svår att utvärdera praktiskt i dagsläget och att framtida utvärdering kräver implementering självständigt från *pygeoapi* alternativt inväntan av att *pygeoapi* utvecklar vidare implementering av EDR. Framtagning av standarden har bland annat gjorts av UK Met Office, US National Weather Service, NASA och ESRI. Det antas att dessa har ett fortsatt intresse för att se standarden utvecklas och etableras.

Sammanfattningsvis kan utifrån denna studie konstateras att EDR teoretiskt är mycket väl lämpad för datadelning av multidimensionella data såsom väder- och prognosdata, men att standarden ännu är för ny och med för få verktyg för att vara av praktisk nytta.

Litteraturförteckning

[Allen, 2020] Allen, M. (7 maj 2020). *OGC Web Services: What is WMS or WFS or WMTS? And how do I use them?*. Medium.

<https://markallengis.medium.com/ogc-web-services-d3186fdc988f> [2024-08-28]

[Angular, u.å] Angular. (u.å). *Angular* (Version 18.2.1). GitHub.

<https://github.com/angular/angular> [2024-08-28]

[Angular, 2024] Angular. (2024). *What is Angular?* <https://angular.dev/overview> [2024-08-28]

[Bermudez et al., 2009] Bermudez, L., Bogden, P., Bridger, E., Cook, T., Galvarino, C., Creager, G., Forrest, D., & Graybeal, J. (2009, May). Web feature service (WFS) and sensor observation service (SOS) comparison to publish time series data. I *2009 International Symposium on Collaborative Technologies and Systems* (s. 36-43). IEEE. [10.1109/CTS.2009.5067460](https://doi.org/10.1109/CTS.2009.5067460)

[Blackburn et al., 2023] Blackburn, C., Lewis, E., Mowat, M., Nayembil, M., Richardson, A., & Stevenson, J. (2023). *Using OGC API standards for marine data delivery: MEDIN Pilot Project*. British Geological Survey. <https://nora.nerc.ac.uk/id/eprint/535011/>

[Cannata et al., 2014] Cannata, M., Antonovic, M. P., & Molinari, M. E. (2014). Load testing of HELIDEM geo-portal: an OGC open standards interoperability example integrating WMS, WFS, WCS and WPS. *International Journal of Spatial Data Infrastructures Research*, 9, 107-130. [10.2902/1725-0463.2014.09.art5](https://doi.org/10.2902/1725-0463.2014.09.art5)

[Castronova et al., 2013] Castronova, A. M., Goodall, J. L., & Elag, M. M. (2013). Models as web services using the open geospatial consortium (ogc) web processing service (wps) standard. *Environmental Modelling & Software*, 41, 72-83. <https://doi.org/10.1016/j.envsoft.2012.11.010>

[Chester, 2021] Chester, S. (17 november 2021). *INSPIRE and OGC APIs - Modernizing INSPIRE*. Open Geospatial Consortium.

<https://www.ogc.org/blog-article/inspire-and-ogc-apis-modernizing-inspire/> [2024-08-28]

[Di et al., 2010] Di, L., Yu, G., Shao, Y., Bai, Y., Deng, M., & McDonald, K. R. (2010, juli). Persistent WCS and CSW services of GOES data for GEOSS. I *2010 IEEE International Geoscience and Remote Sensing Symposium* (s. 1699-1702). IEEE. [10.1109/IGARSS.2010.5653279](https://doi.org/10.1109/IGARSS.2010.5653279)

[ECMWF, u.å.-a] European Centre for Medium-Range Weather Forecasts. (u.å.). *History*. <https://www.ecmwf.int/en/about/who-we-are/history> [2024-08-28]

[ECMWF, u.å.-b] European Centre for Medium-Range Weather Forecasts. (u.å.-b). *Integrated Forecasting System*. <https://www.ecmwf.int/en/forecasts/documentation-and-support/changes-ecmwf-model> [2024-08-28]

[ECMWF, u.å.-c] European Centre for Medium-Range Weather Forecasts. (u.å.-c). *Who we are*. <https://www.ecmwf.int/en/about/who-we-are> [2024-08-28]

[ECMWF, u.å.-d] European Centre for Medium-Range Weather Forecasts. (u.å.-d). *Open data*. <https://www.ecmwf.int/en/forecasts/datasets/open-data> [2024-08-28]

[Ekström, D. & Johannesson, J., 2020] Ekström, D., & Johannesson, J. (2020). *Värdet av öppna data: Samhällsekonomisk nyttoanalys av tillgängliggörande av särskilt värdefulla data* [PDF]. Damvad Analytics. Bilaga till Lantmäteriet rapport (Svedlund et al., 2020). https://www.lantmateriet.se/contentassets/e16a59e08cb744149c878776256560e6/bilaga-2_-vardet-av-oppna-data---samhallsekonomisk-nyttanalyys-av-vardefulla-datamangder.pdf

[Eugene et al., 2022] Eugene, G. Y., Di, L., Meyer, D. J., Zhao, P., Lin, L., Zhang, C., & Cvejovic, S. (2022, juli). ICroplandNet: An open distributed training dataset for irrigated cropland detection. I *2022 10th International Conference on Agro-geoinformatics (Agro-Geoinformatics)* (s. 1-6). IEEE. [10.1109/Agro-Geoinformatics55649.2022.9859073](https://doi.org/10.1109/Agro-Geoinformatics55649.2022.9859073)

[Fathi et al., 2022] Fathi, M., Hagh Kashani, M., Jameii, S. M., & Mahdipour, E. (2022). Big data analytics in weather forecasting: A systematic review. *Archives of Computational Methods in Engineering*, 29(2), 1247-1275. <https://doi.org/10.1007/s11831-021-09616-4>

[Giuliani et al., 2013] Giuliani, G., Dubois, A., & Lacroix, P. (2013). Testing OGC Web Feature and Coverage Service performance: Towards efficient delivery of geospatial data. *Journal of Spatial Information Science*, (7), 1-23. [10.5311/JOSIS.2013.7.112](https://doi.org/10.5311/JOSIS.2013.7.112)

[Geopython, u.å.] Geopython. (u.å.) *pygeoapi* (Version 0.17.0). GitHub. <https://github.com/geopython/pygeoapi> [2024-08-28]

[INSPIRE, u.å.] INSPIRE Knowledge Base. (u.å.). *Inspire knowledge base*. European Commission. https://knowledge-base.inspire.ec.europa.eu/index_en [2024-08-28]

[Koehler, 2024] Koehler, M. (1 juli 2024). *How to convert GRIB to netCDF*. ECMWF. <https://confluence.ecmwf.int/display/OIFS/How+to+convert+GRIB+to+netCDF> [2024-08-28]

[Lantmäteriet, u.å.-a] Lantmäteriet. (u.å.-a). *Översikt över geodatastandarder*. <https://www.lantmateriet.se/sv/geodata/standarder-for-geodata/oversikt-over-geodatastandarder/> [2024-08-28]

[Lantmäteriet, u.å.-b] Lantmäteriet. (u.å.-b). *När används geodatastandarder?*. <https://www.lantmateriet.se/sv/geodata/standarder-for-geodata/nar-anvands-geodatastandarder/> [2024-08-28]

[Lantmäteriet, u.å.-c] Lantmäteriet. (u.å.-c). *Samordning av Inspire*. <https://www.lantmateriet.se/sv/om-lantmateriet/vart-ansvar-och-uppdrag/Vart-sa-mordningsansvar/samordning-av-inspire/> [2024-08-28]

[Lantmäteriet, u.å.-d] Lantmäteriet. (u.å.-d). *Markhöjdmodell Nedladdning*. <https://www.lantmateriet.se/sv/geodata/vara-produkter/produktlista/markhojdmodell-nedladdning/> [2024-08-28]

[Lie et al., 2013] Liu, G., Li, P., & Sha, Y. (2013, oktober). Benchmark for Emergency Management Based on OGC WFS. I *2013 Ninth International Conference on Intelligent Information Hiding and Multimedia Signal Processing* (s. 350-353). IEEE. [10.1109/IIH-MSP.2013.94](https://doi.org/10.1109/IIH-MSP.2013.94)

[MET, u.å.] Meteorologisk institutt (u.å.). *AROME Arctic latest*. MET Norway Thredds Service. <https://thredds.met.no/thredds/catalog/aromearcticlatest/catalog.html> [2024-08-01]

[MET, 2021] Meteorologisk institutt. (26 mars 2021). *Data and products available from MET*. <https://www.met.no/en/free-meteorological-data/Data-and-products-available-from-MET> [2024-08-28]

[OGC, u.å.] Open Geospatial Consortium. (u.å.) *OGC API - Environmental Data Retrieval* (Version 1.1.0). GitHub. <https://github.com/engeospatial/ogcapi-environmental-data-retrieval> [2024-08-28]

[OGC, 2006] Open Geospatial Consortium. (2006). *OpenGIS® Web Map Server Implementation Specification*. <https://www.ogc.org/standard/wms/> [2024-08-28]

[OGC, 2010] Open Geospatial Consortium. (2010). *OpenGIS Web Feature Service 2.0 Interface Standard*. <https://www.ogc.org/standard/wfs/> [2024-08-28]

[OGC, 2012a] Open Geospatial Consortium. (2012). *OGC® WCS 2.0 Interface Standard- Core: Corrigendum*. <https://www.ogc.org/standard/wcs/> [2024-08-28]

- [OGC, 2012b] Open Geospatial Consortium. (2012). *OGC® Sensor Observation Service Interface Standard*. <https://www.ogc.org/standard/sos/> [2024-08-28]
- [OGC, 2018a] Open Geospatial Consortium. (2018a). *OGC Web Coverage Service (WCS) 2.1 Interface Standard - Core*. <https://docs.ogc.org/is/17-089r1/17-089r1.html> [2024-08-28]
- [OGC, 2018b] Open Geospatial Consortium. (2018b). *OGC® WPS 2.0.2 Interface Standard Corrigendum 2*. <https://docs.ogc.org/is/14-065/14-065r2.html> [2024-08-28]
- [OGC, 2019] Open Geospatial Consortium. (2019). *OGC API - Features - Part 1: Core*. <https://docs.ogc.org/is/17-069r3/17-069r3.html> [2024-08-28]
- [OGC, 2023a] Open Geospatial Consortium (2023a). *OGC API - Environmental Data Retrieval Standard*. <https://docs.ogc.org/is/19-086r6/19-086r6.html> [2024-08-28]
- [OGC, 2023b] Open Geospatial Consortium (2023b). *OGC API - Common - Part 1: Core*. <https://docs.ogc.org/is/19-072/19-072.html> [2024-08-28]
- [OGC, 2024a] Open Geospatial Consortium. (22 januari 2024a). *Understanding OGC Standards*. OGC e-Learning. <https://opengeospatial.github.io/e-learning/ogc-standards/text/services-ogc.html> [2024-08-28]
- [OGC, 2024b] Open Geospatial Consortium. (2024b). *OGC APIs - Building Blocks for Location*. <https://ogcapi.ogc.org/> [2024-08-01]
- [OGC, 2024c] Open Geospatial Consortium (22 januari 2024c). *An Introduction to OGC API - Environmental Data Retrieval*. OGC e-Learning. <https://opengeospatial.github.io/e-learning/ogcapi-edr/text/basic-main.html> [2024-08-28]
- [OGC, 2024d] Open Geospatial Consortium. (2024d). *OGC API – Features: Overview*. <https://www.ogc.org/standard/ogcapi-features/> [2024-08-28]
- [OGC, 2024e] Open Geospatial Consortium. (22 januari 2024e). *An Introduction to OGC API - Features*. OGC e-Learning. <http://opengeospatial.github.io/e-learning/ogcapi-features/text/basic-main.html> [2024-08-28]
- [OGC, 2024f] Open Geospatial Consortium. (2024f). *OGC API - RECORDS*. <https://ogcapi.ogc.org/records/> [2024-08-28]

[OGC, 2024g] Open Geospatial Consortium (2024g). *OGC API - RECORDS: Overview*. <https://ogcapi.ogc.org/records/overview.html> [2024-08-28]

[OGC, 2024h] Open Geospatial Consortium (22 januari 2024h). *WPS - Introduction*. OGC e-Learning. <https://opengeospatial.github.io/e-learning/wps/text/basic-main.html>

[OGC, 2024i] Open Geospatial Consortium (22 januari 2024i). *WMS - Operations*. OGC e-Learning. <https://opengeospatial.github.io/e-learning/wms/text/operations.html> [2024-08-28]

[OGC, 2024j] Open Geospatial Consortium (22 januari 2024j). *WFS - Introduction*. OGC e-Learning. <https://opengeospatial.github.io/e-learning/wfs/text/basic-main.html> [2024-08-28]

[OGC, 2024k] Open Geospatial Consortium (22 januari 2024k). *WCS - Introduction*. OGC e-Learning. <https://opengeospatial.github.io/e-learning/wcs/text/basic-main.html> [2024-08-28]

[OGC, 2024l] Open Geospatial Consortium (22 januari 2024l). *SOS - Main*. OGC e-Learning. <https://opengeospatial.github.io/e-learning/sos/text/main.html> [2024-08-28]

[OpenAI, 2024] OpenAI. (2024). ChatGPT 3.5 (Version mars 2024) [Språkmodell]. <https://chat.openai.com/>

[OpenLayers, u.å.] OpenLayers. (u.å.). *OpenLayers* (Version 10.1.0). GitHub. <https://github.com/openlayers/openlayers> [2024-08-28]

[Percivall, 2010] Percivall, G. (2010). The application of open standards to enhance the interoperability of geoscience information. *International Journal of Digital Earth*, 3(S1), 14–30. <https://doi.org/10.1080/17538941003792751>

[pygeoapi, 2024a] pygeoapi (2024a). *Publishing vector data to OGC API - Features*. <https://docs.pygeoapi.io/en/latest/data-publishing/ogcapi-features.html> [2024-08-28]

[pygeoapi, 2024b] pygeoapi (2024b). *Introduction*. <https://docs.pygeoapi.io/en/stable/introduction.html> [2024-08-28]

[pygeoapi, 2024c] pygeoapi (2024c). <https://pygeoapi.io/> [2024-08-28]

[pygeoapi, 2024d] pygeoapi (2024d). *Publishing data to OGC API - Environmental Data Retrieval*.

<https://docs.pygeoapi.io/en/latest/data-publishing/ogcapi-edr.html#xarray-edr>
[2024-08-28]

[Santana & Davis Jr, 2023] Santana, I. L., & Davis Jr, C. A. (2023). Comparative Performance Evaluation of OGC API and OGC Web Feature Service. I *GEOINFO* (s. 134-143).

[Santhanavanich et al., 2023] Santhanavanich, T., Padsala, R., Rosknecht, M., Dabirian, S., Saad, M. M., Eicker, U., & Coors, V. (2023). Enabling Interoperability of Urban Building Energy Data Based on OGC API Standards and Citygml 3D City Models. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 10, 97-105.

<https://doi.org/10.5194/isprs-annals-X-1-W1-2023-97-2023>

[Schubert & Bamberger, 2019] Schubert, C., & Bamberger, H. (2019). Handling continuous streams for meteorological mapping. *Service-Oriented Mapping: Changing Paradigm in Map Production and Geoinformation Management*, 251-268.

[Seib et al., 2018] Seib, J., Voidrot-Martinez, M. F., & Chris, L. (2018). *OGC Best Practice for using Web Map Services (WMS) with Ensembles of Forecast Data*. Version 1.0. Open Geospatial Consortium.

<http://docs.opengeospatial.org/is/16-086r3/16-086r3.html>

[Simoes & Cerciello, 2022] Simoes, J., & Cerciello, A. (2022). Serving geospatial data using modern and legacy standards: a case study from the urban health domain. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 48, 419-425.

<https://doi.org/10.5194/isprs-archives-XLVIII-4-W1-2022-419-2022>

[SIS, u.å.-a] Svenska Institutet för Standarder. (u.å.-a). *Vad är en standard?*

<https://www.sis.se/standarder/vad-ar-en-standard/> [2024-08-28]

[SIS, u.å.-b] Svenska Institutet för Standarder. (u.å.-b). *Tillgängliggöra geodata - så att de kan användas av andra*.

<https://www.sis.se/relevantastandarderfrgeodataenvgledning/tillgnngliggrageodat/asattdekananvndasavandra/> [2024-08-28]

[SMHI, u.å.] Sveriges meteorologiska och hydrologiska institut. (u.å.-a).

<https://www.smhi.se/q/Lund/2693678> [2024-05-21]

[SMHI, 2021a] Sveriges meteorologiska och hydrologiska institut (22 april 2021). *Utbyte och analys av väderobservationer - historik*.

<https://www.smhi.se/kunskapsbanken/utbyte-av-vaderobservationer-historik-1.107219> [2024-08-28]

[SMHI, 2021b] Sveriges meteorologiska och hydrologiska institut. (21 december 2021). *Meteorologiska modeller*.
<https://www.smhi.se/kunskapsbanken/meteorologi/meteorologiska-modeller/meteorologiska-modeller-1.5932> [2024-02-19]

[SMHI, 2022] Sveriges meteorologiska och hydrologiska institut. (11 november 2022). *GRIB-format*.
<https://www.smhi.se/data/oppna-data/information-om-oppna-data/grib-format-1.30761> [2024-08-28]

[Sofos et al., 2017] Sofos, I., Vescoukis, V., Gkegkas, A., & Tsilimantou, E. (2017). Applying OGC standards to develop a land surveying measurement model. *ISPRS International Journal of Geo-Information*, 6(3), 67.
<https://doi.org/10.3390/ijgi6030067>

[Sorg & Kunkel, 2015] Sorg, J. & Kunkel, R. (2015). Conception and implementation of an ogc-compliant sensor observation service for a standardized access to raster data. *ISPRS International Journal of Geo-Information*, 4(3), s. 1076-1096.
<https://doi.org/10.3390/ijgi4031076>

[Sun, Di, et al., 2016] Sun, Z., Di, L., Zhang, C., Lin, L., Fang, H., Tan, X. & Yue, P. (2016, July). Combining OGC WCS with SOAP to facilitate the retrieval of remote sensing imagery about agricultural fields. I *2016 Fifth International Conference on Agro-Geoinformatics (Agro-Geoinformatics)* (s. 1-4). IEEE.
[10.1109/Agro-Geoinformatics.2016.7577652](https://doi.org/10.1109/Agro-Geoinformatics.2016.7577652)

[Sun, Liu, et al., 2016] Sun, W., Liu, X., & Li, C. (2016, januari). Study of A Highly Efficient WPS Framework. I *2016 International Conference on Intelligent Control and Computer Application (ICCA 2016)* (s. 447-450). Atlantis Press.
[10.2991/icca-16.2016.107](https://doi.org/10.2991/icca-16.2016.107)

[Taylor, 2023] Taylor, H. (2 maj 2023). *Topplistan över webbutvecklingsramverk (uppdaterad lista 2023)*. WeAssemble.
<https://weassemble.se/blog/top-web-development-frameworks-updated-list-2023/> [2024-08-28]

[Thredds Data Server, 2020] Thredds Data Server. (24 augusti 2020). *TDS Web Coverage Service (WCS)*.
https://docs.unidata.ucar.edu/tds/5.5/userguide/wcs_ref.html [2024-08-28]

[Unidata, u.å.] Unidata. (u.å). *Network Common Data Form (NetCDF)*.
<https://www.unidata.ucar.edu/software/netcdf/> [2024-08-28]

[Vanmeulebrouk et al., 2009] Vanmeulebrouk, B., Bulens, J. D., Krause, A. U. M., & de Groot, H. L. E. (2009). OGC standards in daily practice: gaps and difficulties found in their use.

[Voidrot-Martinez et al., 2014] Voidrot-Martinez, M. F., Chris, L., Seib, J., Ladner, R., Custer, A., de La Beaujardière, J., Siemen, S. (2014). *OGC Best Practice for using Web Map Services (WMS) with Time-Dependent or Elevation-Dependent Data*. Version 1.0. Open Geospatial Consortium. <http://dx.doi.org/10.25607/OBP-669>

[Wavhale et al., 2020] Wavhale, V. D., Bira, S., Kumar, V., & Choudhari, V. R. (2020). Weather Data Forecast and Analytics. *Weather*, 7(08).

[Xarray, 2024] Xarray. (2024). *Installation*.
<https://docs.xarray.dev/en/latest/getting-started-guide/installing.html> [2024-08-28]

[Yr, u.å.-a] Yr. (u.å.-a). *Facts about Yr*.
<https://hjelp.yr.no/hc/en-us/articles/206550539-Facts-about-Yr> [2024-08-28]

[Yr, u.å.-b] Yr. (u.å.-b). *GRIB weather data*.
<https://hjelp.yr.no/hc/en-us/articles/360009342993-GRIB-weather-data>
[2024-08-28]

[Yr, u.å.-c] Yr. (u.å.-c). *Lund*.
<https://www.yr.no/en/forecast/daily-table/2-2693678/Sweden/Sk%C3%A5ne/Lund%20Municipality/Lund> [2024-05-21]

[Zhang et al., 2010] Zhang, C., Zhao, T. & Li, W. (2010). Automatic search of geospatial features for disaster and emergency management. *International Journal of Applied Earth Observation and Geoinformation*, 12(6), 409-418.
<https://doi.org/10.1016/j.jag.2010.05.004>

Appendix

Appendix A. Fullständig transkription av kommunikation med ChatGPT

Följande bilaga är en fullständig transkription av kommunikation med ChatGPT för sammanställning av information i syfte att bedöma huruvida standarderna OGC Web Map Service (WMS), OGC Web Feature Service (WFS), OGC Web Coverage Service (WCS), OGC Sensor Observation Service (SOS), OGC API - Features (Features), OGC API - Environmental Data Retrieval (EDR), OGC API - Records (Records) och OGC Web Processing Service (WPS) uppfyller, uppfyller delvis eller inte alls uppfyller kravlistan i avsnitt 3.2. Kraven har översatts från svenska till engelska, formulerats om till frågor och sedan ställts till ChatGPT. I vissa fall har ytterligare uppmaning till ChatGPT behövts för ett tillräckligt svar.

Kommunikation med ChatGPT genomfördes i mars 2024 och med version 3.5, som vid tillfället var senast uppdaterad i januari 2022. Standarderna EDR och Records var vid tid för kommunikation inte tillgängliga för ChatGPT att besvara frågor kring och nämns därmed inte i följande transkription.

“You” syftar på användaren, alltså undertecknad.

“ChatGPT” syftar på svaret från OpenAI:s ChatGPT version 3.5.

You

Are the OGC standards WMS, WFS, WCS, SOS, OGC API - Features and WPS defined as "open standards"?

ChatGPT

Yes, the Open Geospatial Consortium (OGC) standards you mentioned — WMS (Web Map Service), WFS (Web Feature Service), WCS (Web Coverage Service), SOS (Sensor Observation Service), OGC API - Features, and WPS (Web Processing Service) — are considered "open standards."

These standards are developed and maintained by the OGC, an international consortium dedicated to developing open standards for geospatial and location-based services. The term "open" in this context typically refers to standards that are publicly available and can be implemented by anyone without restrictions or royalties. These standards play a crucial role in enabling interoperability and compatibility among different geospatial systems and applications.

You

Are the OGC standards WMS, WFS, WCS, SOS, OGC API - Features and WPS defined as "established standards with straightforward implementation"? Are there available tools for implementation?

ChatGPT

Yes, the OGC standards you mentioned — WMS, WFS, WCS, SOS, OGC API - Features, and WPS — are indeed established standards with straightforward implementations. These standards have been widely adopted in the geospatial industry and are well-documented, making them relatively easy to understand and implement.

There are numerous tools and libraries available for implementing these standards. Many open-source and commercial GIS (Geographic Information System) software packages provide support for these standards out of the box or through plugins. Additionally, the OGC website often provides reference implementations, documentation, and resources to assist developers in implementing these standards in their own applications.

For example:

- For WMS: There are libraries like MapServer, GeoServer, and OpenLayers that provide support for creating and consuming WMS services.
- For WFS: GeoServer, MapServer, and QGIS offer tools for publishing and consuming WFS services.
- For WCS: Tools like GDAL (Geospatial Data Abstraction Library) support working with WCS for accessing and serving coverage data.
- For SOS: Implementations such as 52°North SOS and istSOS provide support for Sensor Observation Service.
- For OGC API - Features: Several libraries and frameworks are emerging to support the new generation of OGC APIs, including implementations in various programming languages like Python, JavaScript, and Java.
- For WPS: Libraries like PyWPS and 52°North WPS provide support for creating and consuming Web Processing Service.

These tools and libraries greatly simplify the implementation of OGC standards, allowing developers to focus on building applications that leverage geospatial data and services.

You

Assess each standard individually according to whether they are easy to implement

ChatGPT

Sure, let's assess each OGC standard individually in terms of how easy it is to implement:

1. **WMS (Web Map Service):**
 - **Ease of Implementation:** WMS is relatively straightforward to implement, especially for basic functionality like serving pre-rendered map images. The standard defines simple HTTP requests and responses for requesting map images. However, more advanced features like layer styling and customization can add complexity.
 - **Available Tools:** There are many tools and libraries available for both serving and consuming WMS services, such as MapServer, GeoServer, OpenLayers, and Leaflet. These tools provide a range of options for implementing WMS in various programming languages and environments.
2. **WFS (Web Feature Service):**
 - **Ease of Implementation:** WFS can be moderately complex to implement compared to WMS, especially when dealing with feature querying and transactions. Understanding the GML (Geography Markup Language) data format, as used in WFS responses, can require some learning.
 - **Available Tools:** Tools like GeoServer, MapServer, and QGIS provide support for publishing and consuming WFS services. Libraries such as OGR (part of GDAL) also support working with WFS data.
3. **WCS (Web Coverage Service):**
 - **Ease of Implementation:** WCS implementation can vary in complexity depending on the level of functionality required. Basic WCS implementations for serving and accessing raster data can be straightforward, but more advanced features like subsetting and resampling may require additional effort.
 - **Available Tools:** Libraries like GDAL provide support for working with WCS for accessing and serving coverage data. Server implementations like GeoServer also support WCS.
4. **SOS (Sensor Observation Service):**
 - **Ease of Implementation:** SOS can be moderately complex to implement, especially when dealing with real-time sensor data and complex observations. Understanding the data models and encoding standards used in SOS can require some learning.
 - **Available Tools:** Implementations such as 52°North SOS and istSOS provide support for SOS. However, compared to other OGC standards, the availability of tools specifically for SOS may be more limited.
5. **OGC API - Features:**
 - **Ease of Implementation:** OGC API - Features is designed to be more lightweight and easier to implement compared to previous OGC standards like WFS. Its RESTful architecture and use of modern web standards make it more approachable for developers.

- **Available Tools:** While OGC API - Features is relatively new, there are already emerging libraries and frameworks to support its implementation in various programming languages, such as pygeoapi, GeoNode, and GeoTools.

6. **WPS (Web Processing Service):**

- **Ease of Implementation:** WPS can be moderately complex to implement, especially when dealing with processing large or complex datasets. Understanding the WPS request and response formats, as well as the execution model, may require some learning.
- **Available Tools:** Libraries like PyWPS and 52°North WPS provide support for creating and consuming WPS services. However, compared to other OGC standards, the availability of tools specifically for WPS may be more limited.

Overall, while all of these standards have varying levels of complexity in implementation, there are numerous tools, libraries, and resources available to assist developers in implementing them effectively. The choice of which standard to implement may depend on specific project requirements and the level of complexity and functionality needed.

You

Are the OGC standards WMS, WFS, WCS, SOS, OGC API - Features and WPS defined as "established standards with easy implementation"?

ChatGPT

The OGC standards you've mentioned — WMS, WFS, WCS, SOS, OGC API - Features, and WPS — are indeed established standards in the geospatial domain, but they vary in terms of implementation complexity and ease.

Among them, WMS (Web Map Service) is generally considered to have a relatively straightforward implementation compared to others. It's been around for a while, and there are many tools and libraries available to help implement it efficiently. Similarly, WFS (Web Feature Service) and WCS (Web Coverage Service) have matured over time, and while they might be more complex than WMS due to dealing with vector and raster data, respectively, there are still plenty of resources available for implementation.

SOS (Sensor Observation Service) and WPS (Web Processing Service) are considered to be more specialized and may require a deeper understanding of specific domains, such as sensor networks or geospatial processing. Implementing these standards might be more challenging, but there are still tools and frameworks available to facilitate their implementation.

OGC API - Features represents a newer generation of OGC standards designed to be more lightweight and easier to implement compared to its predecessors like

WFS. While it's relatively new, the simplicity of its RESTful architecture and use of modern web standards make it more approachable for developers.

In summary, while all of these standards are established within the geospatial community, their ease of implementation varies depending on factors such as complexity, available tools, and the developer's familiarity with the domain.

You

Are the OGC standards WMS, WFS, WCS, SOS, OGC API - Features and WPS accessible to use in a web browser?

ChatGPT

Yes, all of the OGC standards you've mentioned — WMS, WFS, WCS, SOS, OGC API - Features, and WPS — can be accessed and utilized in a web browser. Here's how each standard can typically be used:

- 1. WMS (Web Map Service):**
 - WMS is commonly used to serve map images over the web. Most modern web mapping libraries and frameworks, such as OpenLayers and Leaflet, provide built-in support for consuming WMS services. You can include WMS layers in web maps and interact with them directly within a browser.
- 2. WFS (Web Feature Service):**
 - WFS allows for the retrieval and manipulation of geographic features over the web. Web mapping applications can use WFS to retrieve vector data and display it in a web browser. Libraries like OpenLayers and Leaflet support consuming WFS services for displaying and interacting with vector features.
- 3. WCS (Web Coverage Service):**
 - WCS is used to serve and retrieve raster data over the web. Web mapping applications can use WCS to access raster datasets, such as satellite imagery or elevation data, and display them in a web browser. Similar to WMS and WFS, WCS can be integrated into web mapping libraries and frameworks.
- 4. SOS (Sensor Observation Service):**
 - SOS is used to request and retrieve sensor observation data. While less commonly used in traditional web mapping applications, SOS can still be accessed and utilized in a web browser. Web applications dealing with real-time sensor data can use SOS to retrieve and visualize sensor observations.
- 5. OGC API - Features:**
 - OGC API - Features is designed to provide a more modern and lightweight approach to accessing and manipulating geospatial data over the web. Its RESTful architecture makes it well-suited for use in web applications, as developers can make HTTP requests to interact with the API directly from a web browser.

6. **WPS (Web Processing Service):**

- WPS allows for the execution of geospatial processes over the web. While less commonly integrated directly into web mapping applications, WPS can still be accessed and utilized in a web browser. Web applications requiring geospatial processing capabilities can use WPS to execute processes and workflows.

In summary, all of these OGC standards can be accessed and utilized within a web browser, either directly through built-in browser capabilities or through integration with web mapping libraries and frameworks.

You

how much variation in interoperability can there be between each of the standard's WMS, WFS, WCS, SOS, OGC API - Features and WPS clients and servers?

ChatGPT

The level of interoperability between clients and servers for each OGC standard (WMS, WFS, WCS, SOS, OGC API - Features, and WPS) can vary depending on several factors:

1. **Adherence to Standards:** The extent to which both the client and server implementations adhere strictly to the specifications outlined by the OGC can greatly affect interoperability. Deviations or non-compliance with the standards may result in compatibility issues.
2. **Feature Support:** The level of support for various features and operations defined by the standards can impact interoperability. For example, if a WFS client expects a specific operation to be supported by the server, but the server implementation does not provide it, interoperability may be compromised.
3. **Data Formats:** The compatibility of data formats used for communication between clients and servers is crucial. If a client expects data in a particular format (e.g., GML for WFS), but the server provides data in a different format or version, interoperability issues may arise.
4. **Versioning:** Differences in versions of the standards implemented by clients and servers can lead to interoperability challenges. Newer versions may introduce changes or enhancements that older versions do not support, potentially causing compatibility issues between clients and servers.
5. **Custom Extensions:** Some implementations may include custom extensions or additional functionality beyond the core specifications of the standards. While these extensions can add value, they may also introduce interoperability issues if they are not supported by other implementations.
6. **Quality of Implementation:** The quality and robustness of both client and server implementations play a significant role in interoperability. Issues

such as bugs, performance limitations, or incomplete implementations can hinder interoperability between clients and servers.

Overall, while the OGC standards provide a framework for interoperability, the actual level of interoperability between clients and servers can vary depending on these factors. Efforts to ensure adherence to standards, support for common features, compatibility with standard data formats, and rigorous testing of implementations can help improve interoperability between different clients and servers. Additionally, standardized compliance testing suites provided by OGC can help identify and address interoperability issues.

You

How much variation in interoperability can there be between WMS clients and servers?

ChatGPT

The level of interoperability between WMS (Web Map Service) clients and servers can vary based on several factors:

1. **Adherence to Standards:** Both the WMS client and server implementations need to adhere strictly to the WMS specifications outlined by the OGC. Deviations or non-compliance with the standards may result in interoperability issues.
2. **Version Compatibility:** Differences in the versions of the WMS standard supported by the client and server can impact interoperability. Newer versions may introduce changes or enhancements that older versions do not support, potentially causing compatibility issues.
3. **Supported Operations:** WMS defines various operations for requesting map images, such as GetCapabilities, GetMap, GetFeatureInfo, etc. The level of support for these operations on both the client and server sides can affect interoperability. If a client expects a specific operation to be supported by the server, but the server does not provide it, interoperability may be compromised.
4. **Coordinate Reference Systems (CRS):** The support and handling of different CRS by both the client and server are crucial for interoperability. If the client and server use different CRS or if there are discrepancies in how CRS are handled, it can lead to misalignment or distortion in the displayed map images.
5. **Styling and Symbology:** WMS allows for styling and symbolizing map layers. The interoperability between clients and servers can be affected if there are differences in how styling information is interpreted or applied.
6. **Data Formats:** The compatibility of data formats used for communication between clients and servers, such as image formats (e.g., PNG, JPEG) and map data formats (e.g., GML), can impact interoperability. Both client and server implementations should support common data formats to ensure seamless communication.

7. **Performance and Scalability:** Differences in performance and scalability between client and server implementations can affect interoperability, especially when dealing with large datasets or high volumes of requests.

Overall, while the WMS standard provides a framework for interoperability, the actual level of interoperability between WMS clients and servers can vary based on these factors. Efforts to ensure adherence to standards, support for common operations and data formats, and rigorous testing of implementations can help improve interoperability between different WMS clients and servers. Additionally, standardized compliance testing suites provided by OGC can help identify and address interoperability issues.

You

How much variation in interoperability can there be between WFS clients and servers?

ChatGPT

The level of interoperability between WFS (Web Feature Service) clients and servers can vary based on several factors similar to those affecting WMS interoperability:

1. **Adherence to Standards:** Both the WFS client and server implementations need to adhere strictly to the WFS specifications outlined by the OGC. Any deviations or non-compliance with the standards may lead to interoperability issues.
2. **Version Compatibility:** Differences in the versions of the WFS standard supported by the client and server can impact interoperability. Newer versions may introduce changes or enhancements that older versions do not support, potentially causing compatibility issues.
3. **Supported Operations:** WFS defines various operations for querying, retrieving, and manipulating geographic features, such as GetCapabilities, DescribeFeatureType, GetFeature, etc. The level of support for these operations on both the client and server sides can affect interoperability.
4. **Data Formats:** The compatibility of data formats used for communication between clients and servers, such as GML (Geography Markup Language) for feature data, can impact interoperability. Both client and server implementations should support common data formats to ensure seamless communication.
5. **Filtering and Querying:** WFS allows for filtering and querying of feature data based on various criteria. The interoperability between clients and servers can be affected if there are differences in how filtering and querying expressions are interpreted or applied.
6. **Transaction Support:** WFS also supports transactions for adding, modifying, or deleting feature data. Interoperability can be impacted if there are differences in how transactions are supported or handled by the client and server implementations.

7. **Security Mechanisms:** If security mechanisms such as authentication and authorization are implemented differently or are not supported uniformly between clients and servers, it can lead to interoperability challenges.
8. **Performance and Scalability:** Differences in performance and scalability between client and server implementations can affect interoperability, especially when dealing with large datasets or high volumes of requests.

Overall, similar to WMS interoperability, the actual level of interoperability between WFS clients and servers can vary based on these factors. Efforts to ensure adherence to standards, support for common operations and data formats, and rigorous testing of implementations can help improve interoperability between different WFS clients and servers. Additionally, standardized compliance testing suites provided by OGC can help identify and address interoperability issues.

You

How much variation in interoperability can there be between WCS clients and servers? Is the extensions to the core standard a factor?

ChatGPT

The level of interoperability between WCS (Web Coverage Service) clients and servers can vary, and extensions to the core standard can indeed be a factor. Here are some considerations:

1. **Adherence to Standards:** As with any OGC standard, adherence to the WCS specification is critical for interoperability. Both the client and server implementations should follow the core WCS standard defined by the OGC to ensure compatibility.
2. **Version Compatibility:** Differences in the versions of the WCS standard supported by the client and server can impact interoperability. Newer versions may introduce changes or extensions that older versions do not support, potentially causing compatibility issues.
3. **Core vs. Extensions:** The WCS standard includes core functionality for serving and retrieving coverage data. However, there are also extensions to the standard that provide additional functionality, such as support for different data formats, data subsets, or processing capabilities. If a client or server relies on extensions that are not supported by the other party, interoperability may be compromised.
4. **Supported Operations:** WCS defines various operations for accessing coverage data, such as GetCapabilities, DescribeCoverage, and GetCoverage. The level of support for these operations on both the client and server sides can affect interoperability.
5. **Data Formats:** The compatibility of data formats used for communication between clients and servers, such as GeoTIFF, NetCDF, or GML, can impact interoperability. Both client and server implementations should support common data formats to ensure seamless communication.

6. **Spatial and Temporal Subsetting:** Some WCS implementations may support spatial and temporal subsetting of coverage data, allowing clients to request specific subsets of the data. Interoperability can be affected if there are differences in how subsetting capabilities are implemented or supported.
7. **Performance and Scalability:** Differences in performance and scalability between client and server implementations can affect interoperability, especially when dealing with large coverage datasets or high volumes of requests.

Overall, while the WCS standard provides a framework for interoperability, the actual level of interoperability between WCS clients and servers can vary based on these factors. Efforts to ensure adherence to standards, support for common operations and data formats, and rigorous testing of implementations can help improve interoperability between different WCS clients and servers. Additionally, awareness of any extensions used and their compatibility with other implementations is crucial for achieving interoperability.

You

How much variation in interoperability can there be between SOS clients and servers?

ChatGPT

Interoperability between SOS (Sensor Observation Service) clients and servers can vary based on several factors:

1. **Adherence to Standards:** Both the SOS client and server implementations need to adhere strictly to the SOS specifications outlined by the OGC. Any deviations or non-compliance with the standards may lead to interoperability issues.
2. **Version Compatibility:** Differences in the versions of the SOS standard supported by the client and server can impact interoperability. Newer versions may introduce changes or enhancements that older versions do not support, potentially causing compatibility issues.
3. **Supported Operations:** SOS defines various operations for querying, retrieving, and managing sensor observation data, such as GetCapabilities, DescribeSensor, GetObservation, etc. The level of support for these operations on both the client and server sides can affect interoperability.
4. **Data Formats:** The compatibility of data formats used for communication between clients and servers, such as SensorML for sensor metadata and Observations & Measurements (O&M) for observation data, can impact interoperability. Both client and server implementations should support common data formats to ensure seamless communication.
5. **Filtering and Querying:** SOS allows for filtering and querying of sensor observation data based on various criteria. The interoperability between

clients and servers can be affected if there are differences in how filtering and querying expressions are interpreted or applied.

6. **Security Mechanisms:** If security mechanisms such as authentication and authorization are implemented differently or are not supported uniformly between clients and servers, it can lead to interoperability challenges.
7. **Performance and Scalability:** Differences in performance and scalability between client and server implementations can affect interoperability, especially when dealing with large datasets or high volumes of requests.

Overall, similar to other OGC standards, the actual level of interoperability between SOS clients and servers can vary based on these factors. Efforts to ensure adherence to standards, support for common operations and data formats, and rigorous testing of implementations can help improve interoperability between different SOS clients and servers. Additionally, standardized compliance testing suites provided by OGC can help identify and address interoperability issues.

You

How much variation in interoperability can there be between OGC API - Feature clients and servers?

ChatGPT

Interoperability between OGC API - Features clients and servers is typically expected to be high due to the standard's design principles, but there can still be some variation. Here are some factors that can influence interoperability:

1. **Compliance with Standards:** OGC API - Features is designed to promote interoperability by adopting modern web standards and architectural principles, such as RESTful APIs and JSON encoding. Clients and servers need to strictly adhere to the specifications outlined by OGC to ensure compatibility.
2. **Version Compatibility:** Differences in the versions of the OGC API - Features standard supported by the client and server can impact interoperability. Newer versions may introduce changes or enhancements that older versions do not support, potentially causing compatibility issues.
3. **Supported Operations:** OGC API - Features defines a set of operations for querying, retrieving, creating, updating, and deleting geospatial features. The level of support for these operations on both the client and server sides can affect interoperability.
4. **Data Formats:** The compatibility of data formats used for communication between clients and servers, such as GeoJSON, can impact interoperability. Both client and server implementations should support common data formats to ensure seamless communication.
5. **Hypermedia Controls:** OGC API - Features relies on hypermedia controls to provide discoverability and navigability within the API. Differences in how hypermedia controls are implemented or interpreted by clients and servers can affect interoperability.

6. **Security Mechanisms:** If security mechanisms such as authentication and authorization are implemented differently or are not supported uniformly between clients and servers, it can lead to interoperability challenges.
7. **Performance and Scalability:** Differences in performance and scalability between client and server implementations can affect interoperability, especially when dealing with large datasets or high volumes of requests.

Overall, OGC API - Features is designed to promote interoperability by adopting modern web standards and architectural principles. However, the actual level of interoperability between clients and servers can still vary based on factors such as compliance with standards, version compatibility, supported operations, data formats, hypermedia controls, security mechanisms, and performance considerations. Efforts to ensure adherence to standards, support for common operations and data formats, and rigorous testing of implementations can help improve interoperability between different OGC API - Features clients and servers. Additionally, standardized compliance testing suites provided by OGC can help identify and address interoperability issues.

You

How much variation in interoperability can there be between WPS clients and servers?

ChatGPT

Interoperability between WPS (Web Processing Service) clients and servers can vary based on several factors:

1. **Adherence to Standards:** Both the WPS client and server implementations need to strictly adhere to the WPS specifications outlined by the OGC. Any deviations or non-compliance with the standards may lead to interoperability issues.
2. **Version Compatibility:** Differences in the versions of the WPS standard supported by the client and server can impact interoperability. Newer versions may introduce changes or enhancements that older versions do not support, potentially causing compatibility issues.
3. **Supported Operations:** WPS defines various operations for executing geospatial processes, such as DescribeProcess, Execute, and GetCapabilities. The level of support for these operations on both the client and server sides can affect interoperability.
4. **Data Formats:** The compatibility of data formats used for communication between clients and servers, such as XML for request and response payloads, can impact interoperability. Both client and server implementations should support common data formats to ensure seamless communication.
5. **Process Descriptions:** WPS clients need to understand the process descriptions provided by WPS servers to effectively execute processes. If

there are discrepancies or differences in how process descriptions are interpreted or applied, interoperability may be compromised.

6. **Security Mechanisms:** If security mechanisms such as authentication and authorization are implemented differently or are not supported uniformly between clients and servers, it can lead to interoperability challenges.
7. **Performance and Scalability:** Differences in performance and scalability between client and server implementations can affect interoperability, especially when dealing with complex or resource-intensive processes.

Overall, while WPS is designed to promote interoperability by providing a standardized interface for executing geospatial processes over the web, the actual level of interoperability between clients and servers can still vary based on factors such as compliance with standards, version compatibility, supported operations, data formats, process descriptions, security mechanisms, and performance considerations. Efforts to ensure adherence to standards, support for common operations and data formats, and rigorous testing of implementations can help improve interoperability between different WPS clients and servers. Additionally, standardized compliance testing suites provided by OGC can help identify and address interoperability issues.

You

Can the OGC standards WMS, WFS, WCS, SOS, OGC API - Features and WPS support several data sources at the same time?

ChatGPT

Yes, the OGC standards WMS, WFS, WCS, SOS, OGC API - Features, and WPS can all support multiple data sources simultaneously. Here's how each standard can handle multiple data sources:

1. **WMS (Web Map Service):** WMS can serve map images that combine data from multiple sources or layers. Each layer in a WMS service can represent a different dataset or data source. Clients can request map images that include specific combinations of layers, allowing them to visualize data from multiple sources in a single map.
2. **WFS (Web Feature Service):** WFS allows clients to retrieve and manipulate geographic features from multiple data sources. Clients can query and retrieve features from different feature types or datasets exposed by a WFS server. They can also perform spatial and attribute-based filtering to retrieve specific subsets of features from multiple sources.
3. **WCS (Web Coverage Service):** WCS can serve coverage data (e.g., raster data) from multiple sources. Clients can request coverage data from different coverages or datasets exposed by a WCS server. They can also specify spatial and temporal subsets to retrieve specific portions of coverage data from multiple sources.
4. **SOS (Sensor Observation Service):** SOS can retrieve sensor observation data from multiple sensors or observation platforms. Clients can query and

retrieve observations from different sensors or sensor networks exposed by an SOS server. They can also specify spatial and temporal filters to retrieve specific observations from multiple sources.

5. **OGC API - Features:** OGC API - Features is designed to provide access to geospatial feature data from multiple sources through a standardized API. Clients can query and retrieve features from different feature collections or datasets exposed by an OGC API - Features server. They can also perform spatial and attribute-based filtering to retrieve specific subsets of features from multiple sources.
6. **WPS (Web Processing Service):** WPS can execute geospatial processes that involve data from multiple sources. Clients can submit processing requests that reference input data from different sources, and the WPS server can execute the process using the specified data. The output of the process can also be generated based on data from multiple sources.

In summary, all of these OGC standards have the capability to work with multiple data sources simultaneously, allowing clients to access and manipulate geospatial data from diverse datasets and sources. This flexibility enables a wide range of applications and use cases in the geospatial domain.

You

Are the OGC standards WMS, WFS, WCS, SOS, OGC API - Features and WPS considered time efficient and with support for modern formats?

ChatGPT

The efficiency and support for modern formats can vary among the OGC standards WMS, WFS, WCS, SOS, OGC API - Features, and WPS. Here's a general overview of each standard in terms of time efficiency and support for modern formats:

1. **WMS (Web Map Service):**
 - **Time Efficiency:** WMS can be time-efficient for serving pre-rendered map images. However, generating dynamic or complex maps on the fly can be resource-intensive and may impact performance.
 - **Modern Formats:** WMS supports modern image formats like PNG, JPEG, and GIF for delivering map images. However, it may not support more modern formats for vector data or complex styling.
2. **WFS (Web Feature Service):**
 - **Time Efficiency:** WFS can be efficient for querying and retrieving vector feature data. However, the performance may vary depending on factors such as the complexity of the query, the size of the dataset, and the capabilities of the server.
 - **Modern Formats:** WFS typically uses XML-based formats like GML (Geography Markup Language) for encoding feature data, which may not be considered as modern as some JSON-based formats.
3. **WCS (Web Coverage Service):**

- **Time Efficiency:** WCS can be efficient for serving and retrieving raster coverage data, especially when accessing specific subsets of the data. However, performance may degrade when dealing with large or high-resolution datasets.
 - **Modern Formats:** WCS supports modern raster formats like GeoTIFF and NetCDF for encoding coverage data, which are widely used and considered modern in the geospatial domain.
4. **SOS (Sensor Observation Service):**
- **Time Efficiency:** SOS can be efficient for retrieving sensor observation data, especially when querying specific subsets based on time, location, or sensor type. However, performance may depend on factors such as the volume and frequency of observations.
 - **Modern Formats:** SOS typically uses XML-based formats like SensorML and Observations & Measurements (O&M), which may not be as widely adopted or considered as modern as some JSON-based formats.
5. **OGC API - Features:**
- **Time Efficiency:** OGC API - Features is designed to provide a more lightweight and efficient approach to accessing geospatial feature data over the web. Its RESTful architecture and use of modern web standards can contribute to improved time efficiency.
 - **Modern Formats:** OGC API - Features supports modern data formats like GeoJSON and GeoPackage, which are widely used and considered modern in the geospatial domain.
6. **WPS (Web Processing Service):**
- **Time Efficiency:** WPS can be time-efficient for executing geospatial processes, especially when dealing with complex or resource-intensive operations. However, the performance may depend on factors such as the complexity of the process and the capabilities of the server.
 - **Modern Formats:** WPS typically uses XML-based formats for request and response payloads, which may not be considered as modern as some JSON-based formats.

In summary, while the OGC standards provide mechanisms for accessing and manipulating geospatial data, their efficiency and support for modern formats can vary. Standards like OGC API - Features and WCS are designed with modern formats and efficiency in mind, while others may rely on older or XML-based formats that may not be considered as modern. Additionally, the time efficiency of each standard may depend on factors such as the complexity of the operation, the size of the dataset, and the capabilities of the server implementation.

You

Can the OGC standards WMS, WFS, WCS, SOS, OGC API - Features and WPS

handle large data volumes and are they able to handle several simultaneous requests?

ChatGPT

The ability of the OGC standards WMS, WFS, WCS, SOS, OGC API - Features, and WPS to handle large data volumes and simultaneous requests depends on various factors, including implementation, server capabilities, and the nature of the data being processed. Here's a general overview of each standard's capabilities in handling large data volumes and simultaneous requests:

1. **WMS (Web Map Service):**
 - **Handling Large Data Volumes:** WMS can handle large data volumes for serving map images, but the performance may degrade with extremely large datasets or complex rendering requirements. Efficient server-side caching and tile-based approaches can help improve performance.
 - **Handling Simultaneous Requests:** WMS servers can handle multiple simultaneous requests, but the level of concurrency may be limited by server capacity and configuration. Load balancing and scaling strategies can help distribute requests across multiple server instances.
2. **WFS (Web Feature Service):**
 - **Handling Large Data Volumes:** WFS can handle large data volumes for querying and retrieving vector feature data. However, performance may vary depending on factors such as the complexity of the query, the size of the dataset, and server-side optimizations.
 - **Handling Simultaneous Requests:** WFS servers can handle multiple simultaneous requests, but performance may degrade under heavy load. Server-side optimizations, such as indexing and caching, can help improve concurrency and response times.
3. **WCS (Web Coverage Service):**
 - **Handling Large Data Volumes:** WCS can handle large data volumes for serving raster coverage data, but performance may degrade with extremely large or high-resolution datasets. Efficient data storage and retrieval strategies can help mitigate performance issues.
 - **Handling Simultaneous Requests:** WCS servers can handle multiple simultaneous requests, but the level of concurrency may be limited by server capacity and configuration. Load balancing and parallel processing techniques can help improve scalability.
4. **SOS (Sensor Observation Service):**
 - **Handling Large Data Volumes:** SOS can handle large volumes of sensor observation data, but performance may depend on factors such as the volume and frequency of observations and server-side optimizations.

- **Handling Simultaneous Requests:** SOS servers can handle multiple simultaneous requests, but performance may degrade under heavy load. Server-side optimizations, such as data aggregation and caching, can help improve concurrency and response times.
5. **OGC API - Features:**
- **Handling Large Data Volumes:** OGC API - Features is designed to handle large volumes of geospatial feature data efficiently, leveraging modern web standards and RESTful principles. Servers can implement pagination and filtering mechanisms to manage large datasets effectively.
 - **Handling Simultaneous Requests:** OGC API - Features servers can handle multiple simultaneous requests efficiently, thanks to its RESTful architecture and stateless nature. Horizontal scaling and distributed processing can further improve scalability and concurrency.
6. **WPS (Web Processing Service):**
- **Handling Large Data Volumes:** WPS can handle large data volumes for executing geospatial processes, but the performance may vary depending on the complexity of the process and the size of the input data. Efficient processing algorithms and distributed computing can help manage large datasets.
 - **Handling Simultaneous Requests:** WPS servers can handle multiple simultaneous requests, but the level of concurrency may be limited by server capacity and processing requirements. Parallel processing and workload distribution techniques can help improve scalability.

In summary, while all of these OGC standards are capable of handling large data volumes and simultaneous requests to some extent, the actual performance and scalability depend on various factors, including server implementation, data volume, complexity of operations, and server configuration. Efficient server-side optimizations, such as caching, indexing, load balancing, and parallel processing, can help improve performance and scalability for handling large volumes of data and simultaneous requests.

You

Do the OGC standards WMS, WFS, WCS, SOS, OGC API - Features and WPS support download of raw data in formats such as NetCDF and GRIB?

ChatGPT

The ability to download raw data in formats such as NetCDF and GRIB varies among the OGC standards WMS, WFS, WCS, SOS, OGC API - Features, and WPS. Here's a general overview:

1. **WMS (Web Map Service):** WMS typically serves map images rather than raw data. While it can provide raster data in formats like PNG or JPEG, it

does not typically support direct download of raw data in formats like NetCDF or GRIB.

2. **WFS (Web Feature Service):** WFS serves vector feature data and is not specifically designed for serving raw raster data like NetCDF or GRIB. It typically provides access to vector data in formats like GML (Geography Markup Language) or GeoJSON.
3. **WCS (Web Coverage Service):** WCS is specifically designed for serving and accessing raster coverage data, including raw data in formats like NetCDF and GRIB. Clients can request coverage data in these formats from a WCS server for direct download.
4. **SOS (Sensor Observation Service):** SOS primarily serves sensor observation data, which may not be directly related to raw raster data formats like NetCDF or GRIB. While SOS can serve observation data in various formats, it may not support direct download of raw data in NetCDF or GRIB format.
5. **OGC API - Features:** OGC API - Features is designed for accessing geospatial feature data and is not typically used for serving raw raster data like NetCDF or GRIB. It provides access to feature data in formats like GeoJSON or GML.
6. **WPS (Web Processing Service):** WPS allows for the execution of geospatial processes, which may involve data processing or transformation, including raster data in formats like NetCDF or GRIB. While WPS itself does not serve data, it can be used to process data into different formats, including NetCDF or GRIB, as part of a processing workflow.

In summary, among the listed standards, WCS is specifically designed to serve raster coverage data, including raw data in formats like NetCDF and GRIB. WPS can also be used to process data into these formats as part of a geospatial processing workflow. The other standards may not directly support downloading raw data in formats like NetCDF or GRIB, as they serve different types of geospatial data or focus on other aspects of geospatial information services.

You

Do the OGC standards WMS, WFS, WCS, SOS, OGC API - Features and WPS support filtering and defining subsets? As in, the user should be able to filter and define subsets of raw data based on geometric point, line, and area objects.

ChatGPT

Yes, filtering and defining subsets of raw data based on geometric objects is supported by most of the OGC standards you mentioned, with some variations in implementation details:

1. **WMS (Web Map Service):** While WMS primarily serves pre-rendered map images, some implementations may support spatial filtering based on bounding boxes. However, it does not natively support filtering based on geometric point, line, or area objects.

2. **WFS (Web Feature Service):** WFS is specifically designed for serving vector feature data and supports advanced filtering capabilities based on geometric objects such as points, lines, and polygons. Clients can specify spatial filters in WFS requests to retrieve subsets of feature data that intersect or are contained within specified geometries.
3. **WCS (Web Coverage Service):** WCS primarily serves raster coverage data, and some implementations may support spatial subsetting based on geometric objects. Clients can request subsets of coverage data using spatial bounding boxes or specific geometries.
4. **SOS (Sensor Observation Service):** SOS primarily serves sensor observation data, and some implementations may support spatial filtering based on geometric objects. Clients can specify spatial filters in SOS requests to retrieve sensor observations that intersect or are contained within specified geometries.
5. **OGC API - Features:** OGC API - Features provides a modern RESTful interface for accessing geospatial feature data. It supports advanced filtering capabilities based on geometric objects such as points, lines, and polygons. Clients can specify spatial filters in API requests to retrieve subsets of feature data that intersect or are contained within specified geometries.
6. **WPS (Web Processing Service):** WPS allows for the execution of geospatial processes, and some processes may involve spatial filtering or defining subsets based on geometric objects. Clients can specify input geometries as part of WPS requests to define subsets of data for processing.

In summary, while the level of support for filtering and defining subsets based on geometric objects may vary among the OGC standards, WFS and OGC API - Features are specifically designed to support advanced filtering capabilities for vector feature data. WCS, SOS, and some WMS and WPS implementations may also support spatial filtering to some extent, depending on the capabilities of the server implementation.

You

Do the OGC standards WMS, WFS, WCS, SOS, OGC API - Features and WPS support localized data? As in, data reading should be done with correct position. Which coordinate systems do they support? Do they support the coordinate systems EPSG:3034, EPSG:32661 and/or EPSG:3857?

ChatGPT

The ability to support localized data and various coordinate systems varies among the OGC standards WMS, WFS, WCS, SOS, OGC API - Features, and WPS. Here's a general overview:

1. **WMS (Web Map Service):**

- WMS typically serves pre-rendered map images, and while it may support localized data through spatial filtering, the support for different coordinate systems depends on the server implementation.
 - WMS can support a wide range of coordinate systems, including EPSG:3034 (ETRS89 / LAEA Europe), EPSG:32661 (WGS 84 / UPS North), and EPSG:3857 (WGS 84 / Pseudo-Mercator, commonly used by web mapping services).
2. **WFS (Web Feature Service):**
 - WFS is designed to serve vector feature data, and it typically supports localized data and various coordinate systems.
 - WFS can support a wide range of coordinate systems, including EPSG:3034, EPSG:32661, and EPSG:3857.
 3. **WCS (Web Coverage Service):**
 - WCS is specifically designed to serve raster coverage data, and it typically supports localized data and various coordinate systems.
 - WCS can support a wide range of coordinate systems, including EPSG:3034, EPSG:32661, and EPSG:3857.
 4. **SOS (Sensor Observation Service):**
 - SOS serves sensor observation data, and while it may support localized data through spatial filtering, the support for different coordinate systems depends on the server implementation.
 - SOS can support various coordinate systems, including EPSG:3034, EPSG:32661, and EPSG:3857.
 5. **OGC API - Features:**
 - OGC API - Features provides a modern RESTful interface for accessing geospatial feature data, and it typically supports localized data and various coordinate systems.
 - OGC API - Features can support a wide range of coordinate systems, including EPSG:3034, EPSG:32661, and EPSG:3857.
 6. **WPS (Web Processing Service):**
 - WPS allows for the execution of geospatial processes, and while it may support localized data through spatial filtering, the support for different coordinate systems depends on the server implementation.
 - WPS can support various coordinate systems, including EPSG:3034, EPSG:32661, and EPSG:3857.

In summary, WFS, WCS, OGC API - Features, and to some extent WMS, SOS, and WPS, support localized data and a wide range of coordinate systems, including EPSG:3034, EPSG:32661, and EPSG:3857. The actual support for specific coordinate systems may vary depending on the capabilities of the server implementation.

You

Can the OGC standards WMS, WFS, WCS, SOS, OGC API - Features and WPS be said to follow principles of web design and easy to understand?

ChatGPT

The OGC standards WMS, WFS, WCS, SOS, OGC API - Features, and WPS are designed to adhere to principles of web design and aim to provide interfaces that are relatively easy to understand. Here's how they align with these principles:

1. **WMS (Web Map Service):**
 - WMS follows basic web design principles by providing a standardized interface for serving map images over the web. Its operations and parameters are relatively straightforward, allowing clients to request map images with specific layers, styles, and spatial extents.
2. **WFS (Web Feature Service):**
 - WFS provides a standardized interface for accessing vector feature data over the web. It follows web design principles by offering operations for querying, retrieving, and manipulating feature data in a RESTful manner. While understanding the query syntax and response format may require some familiarity with the standard, WFS aims to provide clear and consistent APIs.
3. **WCS (Web Coverage Service):**
 - WCS is designed to serve raster coverage data over the web and follows web design principles by providing standardized operations for accessing coverage data in a RESTful manner. It offers clear APIs for requesting coverage data with specific spatial and temporal extents, as well as different formats and subsets.
4. **SOS (Sensor Observation Service):**
 - SOS serves sensor observation data over the web and follows web design principles by providing standardized operations for accessing observation data. While understanding the data model and query parameters may require some domain knowledge, SOS aims to provide clear and consistent APIs for accessing sensor data.
5. **OGC API - Features:**
 - OGC API - Features represents a modern approach to providing access to geospatial feature data over the web. It follows web design principles by offering a RESTful architecture with standardized APIs for querying, retrieving, creating, updating, and deleting feature data. OGC API - Features aims to be intuitive and easy to understand, with clear documentation and consistent API design.
6. **WPS (Web Processing Service):**
 - WPS allows for the execution of geospatial processes over the web and follows web design principles by providing standardized operations for submitting and executing processes. While understanding the process descriptions and parameters may require some familiarity with the standard, WPS aims to provide clear APIs for invoking geospatial processes.

Overall, while understanding and implementing these standards may require some level of technical expertise, they are designed to follow principles of web design and aim to provide clear and consistent interfaces for accessing geospatial data and services over the web. Additionally, documentation and resources provided by the OGC and other organizations can help users understand and effectively use these standards.

You

What metadata is required for the OGC standards WMS, WFS, WCS, SOS, OGC API - Features and WPS? Do they support metadata of information such as data source, geographic area, available forecasting models, time period, and which parameters are available?

ChatGPT

The OGC standards WMS, WFS, WCS, SOS, OGC API - Features, and WPS each have requirements for metadata to varying degrees. Here's a general overview of the metadata typically associated with each standard:

1. **WMS (Web Map Service):**
 - Metadata for WMS services often includes information about available layers, their styles, supported coordinate reference systems (CRS), bounding boxes, and abstracts describing the data content. It may include information about the data source, geographic area covered, and available time periods if relevant. However, detailed information about parameters and forecasting models is less common.
2. **WFS (Web Feature Service):**
 - WFS metadata typically includes information about available feature types, their attributes, supported CRS, bounding boxes, and abstracts. It may also include information about the data source, geographic area, and time period covered by the features. However, forecasting models and parameter information are typically not included in WFS metadata.
3. **WCS (Web Coverage Service):**
 - WCS metadata often includes information about available coverages, their formats, supported CRS, bounding boxes, and abstracts. It may include information about the data source, geographic area, and time period covered by the coverages. However, detailed information about forecasting models and parameters may not be included in WCS metadata.
4. **SOS (Sensor Observation Service):**
 - SOS metadata typically includes information about available procedures (e.g., sensors), observed properties, offerings, and observed areas. It may also include information about the data source, geographic area, time period covered by the observations, and available forecasting models if applicable.

5. **OGC API - Features:**

- OGC API - Features metadata often includes information about available feature collections, their attributes, supported CRS, bounding boxes, and abstracts. It may also include information about the data source, geographic area covered by the features, and time periods if relevant. However, detailed information about forecasting models and parameters may not be included in OGC API - Features metadata.

6. **WPS (Web Processing Service):**

- WPS metadata typically includes information about available processes, their inputs, outputs, parameters, and abstracts describing their functionality. It may also include information about the data source, geographic area, and time period relevant to the processes. However, detailed information about forecasting models is less common in WPS metadata.

In summary, while metadata for each OGC standard typically includes information about data sources, geographic areas, and sometimes time periods, support for detailed information such as available forecasting models and parameters may vary. Additionally, the extent and format of metadata may depend on the specific implementation and configuration of each service.

You

Which metadata is required?

ChatGPT

The OGC standards have different requirements for metadata, and what is considered "required" can vary based on factors such as the implementation, organizational policies, and the specific use case. However, I can provide some general guidance on the typical metadata elements that are often considered important or required for each OGC standard:

1. **WMS (Web Map Service):**

- Required metadata elements may include:
 - Title: Name of the service or layer.
 - Abstract: Description of the service or layer.
 - Keywords: Terms or phrases describing the content.
 - CRS (Coordinate Reference System): Supported coordinate reference systems.
 - Bounding Box: Geographic extent covered by the service or layer.
 - Format: Supported image formats.
 - Data Source: Information about the source of the data.

2. **WFS (Web Feature Service):**

- Required metadata elements may include:
 - Title: Name of the service or feature type.

- Abstract: Description of the service or feature type.
 - Keywords: Terms or phrases describing the content.
 - CRS (Coordinate Reference System): Supported coordinate reference systems.
 - Bounding Box: Geographic extent covered by the service or feature type.
 - Format: Supported data formats.
 - Data Source: Information about the source of the data.
3. **WCS (Web Coverage Service):**
- Required metadata elements may include:
 - Title: Name of the service or coverage.
 - Abstract: Description of the service or coverage.
 - Keywords: Terms or phrases describing the content.
 - CRS (Coordinate Reference System): Supported coordinate reference systems.
 - Bounding Box: Geographic extent covered by the service or coverage.
 - Format: Supported data formats.
 - Data Source: Information about the source of the data.
4. **SOS (Sensor Observation Service):**
- Required metadata elements may include:
 - Title: Name of the service or procedure.
 - Abstract: Description of the service or procedure.
 - Keywords: Terms or phrases describing the content.
 - Offering: Identifier for the observed phenomena.
 - Procedures: Identifier for the sensor or observation procedure.
 - Data Source: Information about the source of the data.
5. **OGC API - Features:**
- Required metadata elements may include:
 - Title: Name of the service or feature collection.
 - Abstract: Description of the service or feature collection.
 - Keywords: Terms or phrases describing the content.
 - CRS (Coordinate Reference System): Supported coordinate reference systems.
 - Bounding Box: Geographic extent covered by the service or feature collection.
 - Format: Supported data formats.
 - Data Source: Information about the source of the data.
6. **WPS (Web Processing Service):**
- Required metadata elements may include:
 - Title: Name of the service or process.
 - Abstract: Description of the service or process.
 - Keywords: Terms or phrases describing the content.
 - Inputs: Description of input parameters or data required for the process.

- Outputs: Description of output parameters or data produced by the process.
- Data Source: Information about the source of the data.

These are general guidelines, and the actual requirements for metadata may vary depending on factors such as the specific implementation, organizational policies, and the intended use of the service. Additionally, compliance with metadata standards such as ISO 19115 may influence the required metadata elements for some OGC standards.

Appendix B. Kod som implementerar OpenLayers

Följande bilaga består av utdrag av kod från fallstudie 2, för att tydliggöra vilken del av koden som implementerar OpenLayers och hur den ser ut. Följande kod implementerar en kartvy från OpenLayers. Metoderna för *onDrawButtonClick()* och *onPointButtonClick()* lyssnar in på om en polygon (för geografiskt område) eller en punkt ska ritas, baserat på vad användaren har klickat på för knapp i filtreringen (som regleras i en annan komponent, se Appendix C). Metoden *enableDraw()* möjliggör för användaren att rita en polygon eller en punkt i kartan och skickar sedan vidare koordinaterna tillbaka till komponenten som ansvarar för filtrering. Efter att användaren har ritat klart inaktiveras funktionen med *disableDraw()*.

```
/*BSD 2-Clause License

Copyright 2005-present, OpenLayers Contributors All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

    Redistributions of source code must retain the above copyright notice, this list
of conditions and the following disclaimer.

    Redistributions in binary form must reproduce the above copyright notice, this
list of conditions and the following disclaimer in the documentation and/or other
materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY
EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
**/

import { Component, OnInit, ChangeDetectionStrategy, Input, ElementRef,
EventEmitter, Output } from '@angular/core';
import Map from 'ol/Map';
import { Feature, View } from 'ol';
import Polygon from 'ol/geom/Polygon';
import VectorLayer from 'ol/layer/Vector';
import VectorSource from 'ol/source/Vector';
```

```

import TileLayer from 'ol/layer/Tile';
import OSM from 'ol/source/OSM';
import Draw from 'ol/interaction/Draw';
import { Fill, Stroke, Style } from 'ol/style';
import CircleStyle from 'ol/style/Circle';

@Component({
  selector: 'app-map',
  standalone: true,
  imports: [],
  templateUrl: './map.component.html',
  styleUrls: ['./map.component.css'],
  changeDetection: ChangeDetectionStrategy.OnPush,
})
export class MapComponent implements OnInit {
  @Input()
  map: Map = new Map();
  private draw: Draw | null = null;
  // @ts-ignore
  private vectorLayer: VectorLayer<VectorSource> | null = null;
  @Output() rectangleDrawn = new EventEmitter<[number, number, number, number]>();
  @Output() pointDrawn = new EventEmitter<[number, number]>();

  constructor(private elementRef: ElementRef) {}

  ngOnInit() {
    this.initMap();
  }

  private initMap(): void {
    this.map = new Map({
      view: new View({
        center: [0, 0],
        zoom: 3,
        projection: 'EPSG:4326',
      }),
      layers: [
        new TileLayer({
          source: new OSM(),
        }),
      ],
      target: this.elementRef.nativeElement,
    });
  }
}

```

```

onDrawButtonClick(): void {
  this.enableDraw('Polygon');
}

onPointButtonClick(): void {
  this.enableDraw('Point');
}

enableDraw(drawType: 'Point' | 'Polygon'): void {
  this.disableDraw();

  const source = new VectorSource();
  const vector = new VectorLayer({
    source: source,
    style: new Style({
      fill: new Fill({
        color: 'rgba(255, 255, 255, 0.2)',
      }),
      stroke: new Stroke({
        color: '#ffcc33',
        width: 2,
      }),
      image: new CircleStyle({
        radius: 5,
        fill: new Fill({
          color: 'rgba(255, 255, 255, 0.2)',
        }),
        stroke: new Stroke({
          color: '#ffcc33',
          width: 2,
        }),
      }),
    }),
  });

  if (this.vectorLayer) {
    this.map.removeLayer(this.vectorLayer);
  }
  //@ts-ignore
  this.vectorLayer = vector;
  this.map.addLayer(vector);

  this.draw = new Draw({
    source: source,
    type: drawType,
  });
}

```

```

this.map.addInteraction(this.draw);

this.draw.on('drawend', (event: { feature: any; }) => {
  const feature = event.feature;
  if (drawType === 'Polygon') {
    const extent = feature.getGeometry().getExtent();
    console.log('Bounding Box Extent:', extent);
    this.rectangleDrawn.emit(extent as [number, number, number, number]);
  } else if (drawType === 'Point') {
    const coordinates = feature.getGeometry().getCoordinates();
    console.log('Point Coordinates:', coordinates);
    this.pointDrawn.emit(coordinates as [number, number]);
  }
  this.disableDraw();
});
}

private disableDraw(): void {
  if (this.draw) {
    this.map.removeInteraction(this.draw);
    this.draw = null;
  }
}
}
}

```

Appendix C. Kod för kommunikation mellan användare och server

Följande bilaga består av utdrag av kod från fallstudie 2, för att tydliggöra hur kommunikation mellan användaren och servern har implementerats. Följande kod hanterar logik för vad som ska hända när användaren klickar på olika knappar. *onDrawButtonClick()* och *onPointButtonClick()* skickar vidare till kartkomponenten (se Appendix B) att dessa knappar har blivit klickade på, medan *onRectangleDrawn()* och *onPointDrawn()* får tillbaka koordinater från kartkomponenten och gör om det till filterkriterier för filtrering. *onFilterSubmit()* hanterar vilken typ av anrop som ska göras baserat på användarens val och skickar vidare till den *service*-komponent som konstruerar URL:en till servern. *resetFilterCriteria()* ser till så att formuläret rensas efter att användaren har skickat in det. *downloadMetadata()* och *downloadAll()* öppnar konstruerade URL:er vid användarinput för nedladdning av metadata eller nedladdning av hela datasamlingens geografiska område.

```
import { Component, EventEmitter, OnInit, Output } from '@angular/core';
import { EDRCollection } from '../EDRCollection';
import { EDRClientService } from '../services/edrclient.service';
import { HttpClientModule } from '@angular/common/http';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { MapComponent } from '../map/map.component';

@Component({
  selector: 'app-edrcomponent',
  standalone: true,
  imports: [HttpClientModule, CommonModule, FormsModule, MapComponent],
  templateUrl: './edrcomponent.component.html',
  styleUrls: ['./edrcomponent.component.css'],
  providers: []
})
export class EDRComponentComponent implements OnInit {
  collections: EDRCollection[] = [];
  selectedCollection: any | null = null;
  showButtonCollections: any | null = null;
  metadataShownForCollection: any | null = null;
  showFilterForm = false;
  filterCriteria = { area: '', coordinates: '', from: '', to: '', parameters: {} as
{ [key: string]: boolean }};
  parameters: string[] = [];
  filterType: string | null = null;
  @Output() drawButtonClicked = new EventEmitter<void>();
  @Output() pointButtonClicked = new EventEmitter<void>();
```

```

activeView: string | null = null;

constructor(private edrClientService: EDRClientService) {}

ngOnInit(): void {
  this.getCollections();
}

onDrawButtonClick(): void {
  this.drawButtonClicked.emit();
}

onPointButtonClick(): void {
  this.pointButtonClicked.emit();
}

getCollections(): void {
  this.edrClientService.getCollections()
    .subscribe(collections => {
      this.collections = collections;
      console.log('Collections:', collections);
    },
    error => {
      console.error('Error fetching collections', error);
    }
  );
}

getUniqueParameters(collection: EDRCollection): string[] {
  const parameterNames: string[] = [];
  if (collection.parameter_names) {
    Object.values(collection.parameter_names).forEach((value) => {
      parameterNames.push(value.name);
    });
  }
  return parameterNames
}

downloadMetadata(collection: EDRCollection): void {
  window.open(this.edrClientService.getDownloadMetadataUrl(collection), '_blank');
}

downloadAll(collection: EDRCollection): void {
  window.open(this.edrClientService.getDownloadAllUrl(collection), '_blank');
}

```

```

onFilterSubmit(): void {
  console.log('Filter criteria:', this.filterCriteria);
  if (this.filterType === 'position') {
    const filterUrlPosition =
this.edrClientService.generateFilterUrlPosition(this.filterCriteria,
this.showButtonCollections);
    window.open(filterUrlPosition, '_blank');
    console.log(filterUrlPosition);
  } else if (this.filterType === 'cube') {
    const filterUrlCube =
this.edrClientService.generateFilterUrlCube(this.filterCriteria,
this.showButtonCollections);
    window.open(filterUrlCube, '_blank');
  }
  this.resetFilterCriteria();
}

onRectangleDrawn(extent: [number, number, number, number]): void {
  this.filterCriteria.area = `${extent[0]}, ${extent[1]},
  ${extent[2]}, ${extent[3]}`;
}

onPointDrawn(coordinates: [number, number]): void {
  this.filterCriteria.coordinates = `${coordinates[0]} ${coordinates[1]}`;
}

resetFilterCriteria(): void {
  this.filterCriteria = {
    area: '',
    coordinates: '',
    from: '',
    to: '',
    parameters: {}
  };
}
}

```

Nedanstående kod redogör för metoderna i service-komponenten rörande konstruktion av URL:er till servern. *edrURL* är ett privat attribut bestående av strängen "<http://localhost:5050/collections>".

```

getCollections(): Observable<EDRCollection[]> {
  return this.http.get<any>(this.edrUrl)
    .pipe(
      map(response => response.collections),

```



```

        tap(_ => this.log('fetched collections')),
        catchError(this.handleError<EDRCollection[]>('getCollections', []))
    );
}

getDownloadMetadataUrl(collection: EDRCollection) {
    return this.edrUrl+'/'+collection.id+'?f=json'
}

getDownloadAllUrl(collection: any) {
    return
this.edrUrl+'/'+collection.id+'/cube?bbox='+collection.extent.spatial.bbox[0]+'&f=j
son'
}

generateFilterUrlCube(filterCriteria: any, collection: EDRCollection): string {
    const baseUrl = this.edrUrl+'/'+collection.id+'/cube';
    const params = new URLSearchParams();
    if (filterCriteria.area) {
        params.set('bbox', filterCriteria.area);
    }
    if (filterCriteria.from && filterCriteria.to) {
        const datetimePicked = `${filterCriteria.from}/${filterCriteria.to}`
        params.set('datetime', datetimePicked);
    }
    if (filterCriteria.parameters && collection.parameter_names) {
        Object.entries(collection.parameter_names).forEach(([key, value]) => {
            if (filterCriteria.parameters[value.name]) {
                params.append('parameter-name', key);
            }
        });
    }
    params.append('f', 'json');
    return `${baseUrl}?${params.toString()}`;
}

generateFilterUrlPosition(filterCriteria: any, collection: EDRCollection): string
{
    const baseUrl = this.edrUrl+'/'+collection.id+'/position';
    const params = new URLSearchParams();
    if (filterCriteria.coordinates) {
        const coordinates = `POINT(${filterCriteria.coordinates})`
        params.set('coords', coordinates);
    }
    if (filterCriteria.from && filterCriteria.to) {

```

```
const datetimePicked = `${filterCriteria.from}/${filterCriteria.to}`
params.set('datetime', datetimePicked);
}
if (filterCriteria.parameters && collection.parameter_names) {
  Object.entries(collection.parameter_names).forEach(([key, value]) => {
    if (filterCriteria.parameters[value.name]) {
      params.append('parameter-name', key);
    }
  });
}
params.append('f', 'json');
return `${baseUrl}?${params.toString()}`;
}
```

Institutionen av naturgeografi och ekosystemvetenskap, Lunds Universitet

Student-examensarbete (seminarieuppsatser) i geografisk informationsteknik.

Uppsatserna finns tillgängliga på institutionens geobibliotek, Sölvegatan 12, 223 62 LUND. Serien startade 2010. Hela listan och själva uppsatserna är även tillgängliga på LUP student papers och via Geobiblioteket (www.geobib.lu.se).

Serie examensarbete i geografisk informationsteknik

1. Patrik Carlsson och Ulrik Nilsson (2010) Tredimensionella GIS vid fastighetsförvaltning
2. Karin Ekman och Anna Felleson (2010) Att välja grundläggande karttjänst - Utveckling av jämförelsemodell och testverktyg för utvärdering
3. Jakob Mattsson (2011) Synkronisering av vägdatabaser med KML och GeoRSS - En fallstudie i Trafikverkets verksamhet
4. Patrik Andersson and Anders Jürisoo (2011) Effective use of open source GIS in rural planning in South Africa
5. Nariman Emamian och Martin Fredriksson (2012) Visualisering av bygglovsärenden med hjälp av Open Source-verktyg - En undersökning kring hur man kan effektivisera ärendehantering med hjälp av en webbapplikation
6. Gustav Ekstedt and Torkel Endoff (2012) Design and Development of a Mobile GIS Application for Municipal FieldWork
7. Karl Söderberg (2012) Smartphones and 3D Augmented Reality for disaster management - A study of smartphones ability to visualise 3D objects in augmented reality to aid emergency workers in disaster management
8. Viktoria Strömberg (2012) Volymberäkning i samhällsbyggnadsprojekt
9. Daniel Persson (2013) Lagring och webbaserad visualisering av 3D stadsmodeller - En pilotstudie i Kristianstad kommun
10. Lisette Danebjer och Magdalena Nyberg (2013) Utbyte av geodata - studie av leveransstrukturer enligt Sveriges kommuner och landstings objekttypskatalog
11. Alexander Quist (2013) Undersökning och utveckling av ett mobilt GISsystem för kommunal verksamhet
12. Nariman Emamian (2014) Visning av geotekniska provborrningar i en webbmiljö
13. Martin Fredriksson (2014) Integrering av BIM och GIS med spatiala databaser – En prestandaanalys
14. Niklas Krave (2014) Utveckling av en visualiseringsapplikation för solinstrålningsdata
15. Magdalena Nyberg (2015) Designing a generic user interface for distribution of open geodata: based on FME server technology

16. Anna Larsson (2015) Samredovisning av BIM- och GIS-data
17. Anton Lundkvist (2015) Development of a WEB GI System for Disaster Management
18. Ellen Walleij (2015) Mapping in Agricultural Development – Introducing GIS at a smallholders farmers’ cooperative in Malawi
19. Frida Christiansson (2016) Lagring av 3D - geodata - en fallstudie i Malmö Stad
20. Lisette Danebjer (2016) Methodology for creating and modifying distributed topologically structured geographical datasets
21. Jeanette Dunn Ekelund (2016) En jämförelse av algoritmer och resultat för flödesberäkning i QGIS/GRASS och ArcGIS
22. Ebba Gröndahl och Frida Thorman (2016) Verksamhetens optimala läge i staden och hur de är lokaliserade idag
23. Gunnar Rolander (2017) Data transformation using linked data ontologies
24. Måns Andersson och Moa Eklöf (2017) Stilsättning av geografiska data
25. Josefine Axelsson (2018) Automatisering av bygglovsansökningsprocessen med stöd av BIM och GIS
26. Leonard B. O. Berge (2018) Uppdatering och visualisering av stadsmodell med stöd av konverterade BIM-modeller
27. Rickard Ingesson & Gabriella Olsson (2019) Publicering av geografiska data på webben : En utvärdering av programsystem med fokus på öppen källkod
28. Alfred Hildingson & Patrik Sylve (2020) Visualisering av stadsmodeller på webben : Jämförande studie mellan CityGML och CityJSON
29. Isabelle Andersson (2020) Indoor positioning systems in office environments : a study of standards, techniques and implementation processes for indoor maps
30. Sebastian Roos & August Cnattingius (2021) Covid-19-pandemins konsekvenser på svenskt näringsliv - en Space Syntax analys : Hur har konkursutsattheten och arbetslösheten förändrats för detaljhandel, hotell och restauranger till följd av restriktioner som begränsar besöksnäring?
31. Emelie Ulin (2021) Simuleringar i planeringsprocessen med 3D-stadsmodeller
32. Alfred Hirschfeld & Christoffer Karlsson (2022) Designing and implementing a geospatial mobile application
33. Andreas Ahlström (2022) 3D-stadsmodeller för mindre kommuner - vad är behoven och vilka standarder krävs?
34. Jorun Westman (2022) Looking for shrubs in an alvar - Investigating classification of orthophotos as a way of mapping shrub species *Juniperus communis* and *Dasiphora fruticosa* on Stora Alvaret, Öland

35. Sophia Bladh & Ylva Kjellberg (2023) Enhetlig visualisering av geotekniska data och geokonstruktioner - En intervju- och implementeringsstudie
36. David Andersson & Beatrice Ekström (2023) Scenariohantering med parametrisk design i planeringsprocessen
37. Alexander Brynolf (2023) Undersökning av tillgänglighetsmått
38. Axel Andersson (2024) Kvalitetsbeskrivning och kvalitetspåverkande faktorer vid produktion av TIN-modeller
39. Hanna Lundström (2024) Gränssnittsstandarden "OGC API - Environmental Data Retrieval" och dess användning för nedladdning av väderdata