# Ohjelmointirajapinnat avoimen datan käytön mahdollistajina

Turun yliopisto
Tietotekniikan laitos
LuK-tutkielma
Teknillinen tiedekunta
Huhtikuu 2025
Jasmin Lehtola

TURUN YLIOPISTO
Tietotekniikan laitos

Jasmin Lehtola: Ohjelmointirajapinnat avoimen datan käytön mahdollistajina

LuK-tutkielma, 31 s.
Teknillinen tiedekunta
Huhtikuu 2025

---

Ohjelmointirajapinnat ovat keskeinen osa nykyaikaista ohjelmistokehitystä, sillä ne mahdollistavat ulkoisten palveluiden ja avoimen datan hyödyntämisen sovelluksissa. Rajapinnat toimivat yhteysväylänä järjestelmien ja palveluiden välillä ja niiden avulla voidaan hyödyntää jo olemassaolevaa koodia ja toiminnallisuuksia. Hyvin laadittu dokumentaatio on tärkeässä osassa rajapinnan omaksumisen kannalta ja vaikuttaa merkittävästi rajapinnan käytettävyyteen sekä ohjelmistokehittäjän tehokkuuteen. Tässä tutkielmassa tarkoituksena oli tutkia eroavaisuuksia kahden eri avoimen rajapinnan välillä kehittäjän näkökulmasta. Menetelmänä oli Google Mapsin ja AccuWeatherin dokumentaatioiden vertailu. Kriteereinä arvioitiin mm. dokumentaation selkeyttä, käytön helppoutta ja käyttäjätuen saatavuutta. Dokumentaation esimerkkien laajuudessa ja rakenteessa havaittiin eroja, jotka voivat vaikuttaa rajapintojen oppimisen nopeuteen ja kehittäjän kokemaan tyytyväisyyteen. Hyvä dokumentaatio edistää tehokasta kehitystyötä ja on olennainen osa rajapinnan suunnittelua.

Asiasanat: API, ohjelmointirajapinta, avoin data, käytettävyys

# Sisällys

# 1 Johdanto

Viimeisten vuosien aikana ohjelmointirajapintojen määrä on ollut valtavassa kasvussa. Niitä käytetään yhä enenevissä määrin yhdistämään internetsovelluksia ja -palveluita toisiinsa verkkoyhteyden kautta. [1] Ohjelmointirajapinnat nopeuttavat kehitysprosessia, sillä ne sisältävät ohjelmallisia toimintoja ja palveluita, joita kehittäjät voivat käyttää sovellusten rakentamisessa ja olemassa olevien järjestelmisen laajentamisessa [2]. Rajapinnat mahdollistavat myös avoimen datan käytön [1]. Pelkästään Suomen Digi- ja väestötietovirastolla on 2280 (10.4.2025) avointa tietojoukkoa kenen tahansa käytettäväksi [3].

Tämän tutkielman tarkoituksena on tarkastella ohjelmointirajapintojen hyödyntämistä avoimen datan käyttöönotossa sekä tutkia tarkemmin kahden eri avoimen rajapinnan dokumentaatioita. Kiinnostuksen kohteena on myös näiden ohjelmointirajapintojen käytön helppous kehittäjän näkökulmasta.

Tässä tutkielmassa pyritään vastaamaan seuraaviin kysymyksiin:
Tutkimuskysymys 1: Kuinka avoimet rajapinnat helpottavat sovellusten kehittämistä?
Tutkimuskysymys 2: Mitä haasteita ja ongelmia avoimissa rajapinnoissa on?
Tutkimuskysymys 3: Miten kehittäjäkokemus ja käytettävyys eroavat Google Mapsin ja AccuWeatherin rajapintojen välillä?

Tieteellisten artikkelien hakuun käytettiin Web of Science- ja IEEE-tietokantoja sekä Google Scholaria. Artikkeleita löydettiin myös muiden artikkelien lähdeviitteistä. Lisäksi hakukone Googlen avulla haettiin tietoa avoimen datan tietoportaaleista sekä Google Mapsin ja AccuWeatherin dokumentaatioista.

Hakulausekkeina käytettiin esimerkiksi seuraavia:

- ("application programming interface"OR "API") AND ("open data" OR "web API"OR "public API")
- ("API"AND "usability")
- ("application programming interface"OR "API") AND ("open data"OR "open*)
- ("application programming interface"OR "API") AND ("rest*"OR "soap")
- ("application programming interface"OR "API") AND (challenge* OR difficult* OR risk* OR issue*)

Aluksi haettiin muutamia artikkeleita tutkielmalle pohjaksi, ja näistä kirjoitettiin taustaluvut. Näitä löytyi yhteensä 7 kappaletta. Hakutuloksia rajattiin Web Of Science -tietokannassa topic-valinnalla ja vuosiksi valittiin 2010–2024 relevanttien tuloksien löytämiseksi. IEEE:ssä käytettiin ainoastaan vuosilukuja rajauksena. Tätä rajausta vanhemmat valitut artikkelit löydettiin muiden artikkelien lähdeviitteistä. Aluksi hakutuloksista valittiin kiinnostavat otsikot, sitten tarkasteltiin abstraktit ja silmäiltiin johdannot ja lopuksi niiden sekä JUFO-luokituksen perusteella valittiin tutkielman pohjaksi tulevat artikkelit.

Alla olevassa hakutaulukossa havainnollistetaan hakuprosessia. Kaikkea tarvittavaa tietoa ei löydetty hakutaulukon hakulausekkeilla, sillä laajan aiheen rajaaminen yhteen lauseeseen on vaikeaa. Tutkielman edetessä hakulausekkeita muokattiin täsmällisiksi haettavaa tietoa varten, esimerkiksi REST-arkkitehtuuria ja haasteita varten tarvittiin omat hakulausekkeensa.

Taulukko 1.1: Tutkielman pohjaa varten käytetyt hakulausekkeet.

| **Hakulauseke** | **Käytetty tietokanta** | **Hakutulosten määrä** | **Nimen perusteella valitut** | **Abstraktin lukemisen jälkeen valitut** | **Lopulliset valitut lähteet** |
|---|---|---|---|---|---|
| ("application programming interface"OR "API") AND ("open data"OR "web API"OR "public API") | Web Of Science | 686 | 37 | 6 | 4 |
| ("application programming interface"OR "API") AND ("open data"OR "web API"OR "public API") | IEEE | 698 | 42 | 8 | 4 |
| ("API"AND "usability") | Web Of Science | 448 | 33 | 5 | 3 |
| ("API"AND "usability") | IEEE | 478 | 31 | 7 | 2 |

Tähän aiheeseen päädyttiin, koska avoin data, sen käyttäminen rajapintojen kautta ja yleisesti rajapinnoista oppiminen tuntui kiinnostavalta. Vertailtaviksi rajapinnoiksi valikoitiin Google Maps ja AccuWeather, sillä ne ovat molemmat hyvin tunnettuja ja laajasti käytettyjä, mutta kuitenkin erityyppisiä ja erilaisia käyttökohteiltaan. Molemmat ovat käytössä monissa kaupallisissa ja avoimissa sovelluksissa ja niillä on laaja käyttäjäkunta sekä kattava dokumentaatio.

Tutkielman rakenne on seuraava: luvussa 2 tarkastellaan ohjelmointirajapintoja yleisesti sekä perehdytään tarkemmin REST- ja SOAP -arkkitehtuureihin sekä rajapintojen käytettävyyteen ja niiden haasteisiin. Luku 3 keskittyy avoimeen dataan, datan läpinäkyvyyteen ja FAIR-periaatteisiin. Luvussa 4 tutkitaan Google Mapsin ja AccuWeatherin dokumentaatioiden eroavaisuuksia kehittäjän näkökulmasta

seitsemän eri katsantokannan kautta. Lopuksi luku 5 sisältää yhteenvedon sekä pohdintaa aiheesta.

# 2 Ohjelmointirajapinnat

Ohjelmointirajapinnat (application programming interface, API) ovat ohjelmallisia käyttöliittymiä ohjelmistokomponenteille tai -palveluille [1]. Ne sisältävät esimerkiksi kirjastoja, kehitystyökaluja ja ohjelmistokehyksiä. Rajapinnat antavat kehittäjille tehokkaan tavan hyödyntää olemassa olevaa koodia ja toiminnallisuuksia, jolloin niitä ei tarvitse aina rakentaa uudestaan alusta. Näin nopeutetaan kehittämistyötä ja lisätään tuottavuutta. [4]

Ohjelmointirajapinnoilla on keskeinen tehtävä ohjelmistokehityksessä. Ne mahdollistavat verkkosovellusten yhdistämisen ja vuorovaikutteisuuden verkon välityksellä, toimivat väylänä avoimen datan käyttöönotolle ja tukevat ohjelmistokomponenttien uudelleenkäyttöä ja siten taas modulaarisuutta. [1] Modulaarisuus tarkoittaa ohjelmiston jakamista eri osiin eli moduuleihin, jolloin jokainen moduuli on itsenäinen osa, joka suorittaa tietyn tehtävän ja toimii muiden moduulien kanssa. Modulaarisuus helpottaa ohjelmiston kehittämistä ja ylläpitoa, koska muutokset yhdessä moduulissa eivät välttämättä vaikuta koko järjestelmään. [5]

Rajapinnat voidaan myös ajatella kahden osapuolen välisinä sopimuksina: jokainen kehittäjä voi työskennellä itsenäisesti, koska rajapinta määrittelee selkeästi, mitä toinen osapuoli tarjoaa ja mitä toinen osapuoli odottaa saavansa. Tämä mahdollistaa ohjelmistokehityksen tehokkaan koordinoinnin ilman, että kaikkien tarvitsee tuntea toistensa koodin yksityiskohtia. Rajapinnat siis helpottavat yhteistyötä toimimalla selkeinä sopimuksina, jotka määrittelevät rajapinnan, mutta jättävät toteutuksen

vapaasti kehitettäväksi ilman, että se häiritsee muiden työskentelyä. [5]

Ohjelmistokehitysprojektien koordinoinnin lisäksi rajapinnoilla on myös tärkeä rooli viestinnän välineenä. Ne toimivat ohjelmistokehittäjille yhteisenä kielenä, jonka avulla voidaan keskustella selkeästi ja yksiselitteisesti ohjelmiston toiminnallisuuksista. Koska rajapinnat määrittelevät, miten eri komponentit voivat olla vuorovaikutuksessa keskenään, ne luovat yhteisen perustan kehittäjille. Rajapinnat siis toimivat rajapintoina paitsi kooditasolla myös ihmisten välillä, helpottaen yhteistyötä ja ymmärrystä eri kehittäjien ja tiimien kesken. [5]

Rajapinnoilla on sekä suunnittelijoita että käyttäjiä. Suunnittelija on ohjelmoija, joka päättää, mitä ominaisuuksia rajapintaan tulee, kuinka niitä käytetään ja kuinka nämä ominaisuudet dokumentoidaan. Rajapinnan käyttäjä voi olla tavallinen datasta kiinnostunut henkilö, mutta yleensä kuitenkin ohjelmointitaitoinen kehittäjä, joka tarvitsee rajapinnan sisältämiä valmiita ominaisuuksia. [1]

Rajapinnat voidaan jakaa kahteen päätyyppiin: suljettuihin (sisäisiin) ja avoimiin (ulkoisiin) [6]. Avoimet rajapinnat, joita kutsutaan myös nimellä Open API, ovat käytettävissä kenelle tahansa, kuten toisille yrityksille ja yksittäisille kehittäjille. Avoimet rajapinnat tukevat uusien palveluiden kehitystä ja ovat keskeisiä nykyaikaisissa liiketoimintaympäristöissä. [6][7] Sisäiset rajapinnat (engl. internal APIs) taas ovat käytössä vain organisaation sisällä helpottaen sisäisten tiimien toimintaa ja monialaista yhteistyötä. Ulkopuolisilla ei ole pääsyä niihin. [7] Sisäisten rajapintojen avulla yritykset voivat integroida järjestelmiään tehokkaammin, mikä lisää tuottavuutta ja tehostaa liiketoimintaprosesseja [6].

## 2.1 RESTful API

Kehittäjän on tunnettava palvelun rajapinta, jotta hän voi sujuvasti käyttää kolmannen osapuolen tekemää palvelua [8]. Viime vuosina internet-sovelluksissa on käytetty laajalti REST (REpresentational State Transfer) -menetelmää. REST on arkkiteh-

tuurityyli, jonka pääajatus on käyttää HTTP-mekanismia sovellusten yhdistämiseen. Se soveltuu web-palveluiden yksinkertaiseen toteuttamiseen, jolloin HTTP:tä tunteva pystyy helposti ymmärtämään sen periaatteet. REST keskittyy ensisijaisesti resurssien hallintaan ja niiden tilan muokkaamiseen, toisin kuin vaikkapa SOAP-pohjaiset verkkopalvelut, joissa painopiste on viestien lähettämisessä ja vastaanottamisessa. [9]

RESTillä on neljä periaatetta, joihin sen arkkitehtuurityyli pääasiassa perustuu. Ensimmäisenä periaatteena on resurssien tunnistaminen yksilöllisten URI-osoitteiden (Uniform Resource Identifier) kautta. Nämä osoitteet muodostavat osoiteavaruuden resurssien ja palveluiden löytämistä varten. [10]

Toisena periaatteena on yhtenäinen käyttöliittymä, eli jokainen REST-resurssi käsitellään neljän perustoiminnon avulla [10][9]:

- GET hakee resurssin
- POST muuttaa olemassa olevan resurssin tilaa
- PUT luo tai päivittää resurssin
- DELETE poistaa resurssin kokonaan

Näitä neljää toimintoa kutsutaan myös nimellä CRUD. Lyhenne tulee sanoista Create, Read, Update ja Delete. [11]

Kolmantena REST-resurssit luokitellaan esitystapansa mukaan, jotta niitä voidaan hyödyntää eri formaateissa, kuten XML (Extensible Markup Language), JSON (JavaScript Object Notation) ja pelkkä teksti. Resurssit sisältävät myös metadataa, jota voidaan käyttää esimerkiksi välimuistin hallintaan, lähetysvirheiden tunnistamiseen ja toiston hallintaan. [10]

Viimeisenä periaatteena on tilallinen vuorovaikutus hyperlinkkien avulla. Jokainen resurssin kanssa tapahtuva vuorovaikutus on tilaton, eli jokainen pyyntö on itse-

näinen. Tilallisuus perustuu eksplisiittiseen tilansiirtoon, ja sen hallintaan voidaan käyttää useita erilaisia tekniikoita, kuten URI-uudelleenkirjoitusta tai evästeitä. [10]

RESTin perustuminen HTTP:hen ja JSON-viestimuotoon tekee siitä erittäin kevyen sekä vasteajoiltaan että kaistanleveyden käytöltään. Tämä tekee siitä ihanteellisen valinnan erityisesti verkko- ja mobiilisovelluksille. RESTin haittapuolena on kuitenkin standardien puute. [9] RESTissä ei esimerkiksi ole vakiomallinnuskieltä, jolloin rajapintojen suunnittelussa joudutaan usein turvautumaan suunnittelijan omaan tulkintaan siitä, mitä REST periaatteessa tarkoittaa. Tämä voi johtaa toteutusten väliseen epäyhtenäisyyteen ja vaikeuttaa rajapintojen käytettävyyttä. [12]

## 2.2 SOAP API

SOAP (Simple Object Access Protocol) on XML-pohjainen protokolla, jota käytetään sovellusten väliseen tiedonsiirtoon. Se toimii yleisimmin HTTP-protokollan avulla. SOAP-viestejä voi ajatella kirjekuorina, joihin sovellus liittää lähetettävät tiedot. Kirjekuori sisältää kaksi keskeistä osaa: valinnaisen otsikkoelementin (engl. header) ja pakollisen runkoelementin (engl. body), joka sisältää varsinaisen viestisisällön. Otsikko sisältää SOAP-moottoriin liittyviä tietoja sekä tietoa siitä, kuinka viesti tulee käsitellä. Runkoelementti sisältää SOAP-sanoman keskeiset päästä päähän -tiedot XML-dokumentin muodossa. [9]

SOAPin merkittävä etu on sen yhteensopivuus, sillä se mahdollistaa eri alustoilla ja ohjelmointikielillä kehitettyjen sovellusten välisen tiedonvaihdon. Tämä tekee siitä joustavan ratkaisun verkkopalveluiden kehittämiseen. Toinen keskeinen vahvuus on SOAPin tuki erilaisille viestinvaihtomalleille – perinteisestä pyyntö-vastausmallista aina yleislähetyksiin ja monimutkaisiin viestikorrelaatioihin. [9] SOAP vaatii kuitenkin RESTiä enemmän kaistanleveyttä, sillä sen lähettämät pyynnöt ovat suurempia [7].

SOAP soveltuu erityisesti monimutkaisiin toteutuksiin. SOAP-pohjainen verk-

kopalvelupino rakentuu useista standardeista, kuten WSDL (Web Services Description Language), WS-Transaction (Web Services Transaction) ja WS-Security (Web Services Security). [9] WSDL on automaattisesti tuotettu rajapintakuvaus [13], WS-Transaction on yhteensopivuusstandardi, jota käytetään verkkopalveluiden välisten liiketoimintatransaktioiden hallintaan [14] ja WS-Security on verkkopalvelujen turvastandardi, joka mahdollistaa viestien salauksen, allekirjoituksen ja autentikoinnin [15]. SOAPin käyttöönotto edellyttää siis vähintään perustason ymmärrystä näistä standardeista. SOAP on kuitenkin näiden standardien ansiosta turvallisempi kuin REST, ja sitä käytetäänkin esimerkiksi useimpien pankkipalveluiden toteutukseen. [9] Esimerkiksi WSDL-tiedostoa hyödynnetään usein SOAP-rajapinnan integroinnissa. WSDL antaa kattavan kuvauksen käytettävissä olevista metodeista ja kutsurakenteista, ja se on automaattisesti luotu ja formaalisti määritelty. REST-pohjaisista rajapinnoista puuttuu vastaava standardoitu kuvaustapa, jolloin kehittäjä on riippuvainen manuaalisesti tuotetusta dokumentaatiosta, jonka kattavuus ja ajantasaisuus voi vaihdella huomattavasti. [13]

## 2.3 Käytettävyys

Ohjelmointirajapintojen hyvä käytettävyys on tärkeä ominaisuus kehittäjälle, sillä se vaikuttaa koettuun tyytyväisyyteen ja tuottavuuteen [2]. Rajapinnat koetaan usein vaikeina käyttää ja niiden opetteluun käytetään runsaasti aikaa. Käytön hankaluus ja epäselvä dokumentaatio aiheuttavat virheitä ja jopa turvallisuusongelmia. [4] Rajapintojen käyttäjät valitsevat mieluiten sellaisia rajapintoja, joiden avulla on helppo päästä tietoon käsiksi, joiden dokumentaatio on hyvää ja selkeää ja jotka ovat helppoja oppia. Rauf ym. tekemän tutkimuksen *A Systematic Mapping Study of API Usability Evaluation Methods* mukaan opittavuus mainitaan useissa tutkimuslähteissä käyttäjien keskuudessa käytettävyyden tärkeimpänä kriteerinä. [1]

Käytettävyys on laadullinen ominaisuus ja termi, jonka määritelmä riippuu käyt-

täjästä. On kuitenkin olemassa erilaisia käytettävyysstandardeja, joilla käytettävyyttä pystytään määrittelemään. [1] Esimerkiksi ISO-standardit (ISO 9241-11:2018) määrittelevät käytettävyyden niin, että käyttäjät voivat saavuttaa tavoitteensa tuloksellisesti, tehokkaasti ja tyydyttävällä tavalla käyttökonteksti huomioon ottaen [16]. Jakob Nielsen, joka on yksi maailman tunnetuimmista käytettävyysasiantuntijoista, on todennut, että käytettävyys ei ole yksiulotteista. Hän on tarkentanut käytettävyyden viidellä eri määritteellä: opittavuus, tehokkuus, muistettavuus, virheettömyys ja miellyttävyys. Käyttäjän tulisi nopeasti ymmärtää järjestelmän perusrakenne, jotta pääsee työskentelemään sen kanssa. Kun käyttäjä on oppinut järjestelmän käytön, hän pystyy olemaan erittäin tuottava. Järjestelmän tulisi olla helposti muistettava, jolloin käyttäjä pystyy palaamaan sen pariin ilman uudelleenopettelua, vaikka viime käyttökerrasta olisi kulunut aikaa. Käyttäjä ei kohtaa virheitä järjestelmää käyttäessään, ja jos kohtaa, hän selviää niistä helposti. Järjestelmää on myös miellyttävä käyttää ja käyttäjä pitää siitä. [17]

Rajapinta valitaan yleensä dokumentaation perusteella sekä kokeilemalla suorittaa sillä yksinkertaisia tehtäviä. Mikäli rajapinta ei tunnu kehittäjästä riittävän helppokäyttöiseltä ja miellyttävältä, sitä ei yleensä valita käyttöön tai sen käyttö lopetetaan kokeilun jälkeen. Rajapinnan huolellinen valinta on tärkeää, sillä väärin valittu rajapinta heikentää kehitystyön tehokkuutta, lisää resurssien kulutusta sekä viivästyttää projektin etenemistä. [1]

Brad A. Myers ja Jeffrey Stylos ovat todenneet tutkimuksessaan *Improving API usability* [4], että rajapinnat ja niiden käytettävyys vaikuttavat kolmeen keskeiseen sidosryhmään, joita ovat:

1. Rajapintojen suunnittelijat, jotka kehittävät uusia rajapintoja ja laativat niiden dokumentaation. Heidän tavoitteenaan on maksimoida rajapinnan käyttöaste ja minimoida kehityskustannukset.
2. Rajapintojen käyttäjät eli kehittäjät, jotka hyödyntävät sovellusliittymiä luodak-

seen nopeasti toimivia ja virheettömiä ohjelmia.

3. Loppukäyttäjät, jotka odottavat helppokäyttöistä ja luotettavaa ohjelmistoa. Vaikka he eivät suoraan käytä rajapintoja, niiden laatu vaikuttaa epäsuorasti lopputuotteen toimivuuteen ja käyttökokemukseen.

## 2.4 Haasteita

Ohjelmointirajapintojen haasteet liittyvät rajapinnan monimutkaisuuteen, tietoturvaan ja integrointiongelmiin, jotka kaikki kytkeytyvät jossain määrin rajapinnan käytettävyyteen.

Rajapinnat, joilla on huono käytettävyys, heikentävät kehittäjän tuottavuutta ja aiheuttavat virheitä [1]. Rajapinnan käytettävyyteen olennaisimmin vaikuttava tekijä on tutkimusten mukaan sen opittavuus. Esimerkiksi metodien ja luokkien nimeämisten tulisi vastata rajapinnan käyttäjän odotuksia, ja rajapinnan tulisi ohjata käyttäjää oikeaan käyttöön esimerkiksi selkeiden oletusarvojen avulla. Riittävän hyvä dokumentaatio on erittäin tärkeää, ja käytettävyysongelmia voidaankin usein helpottaa parantamalla dokumentaatiota ja siinä olevaa esimerkkikoodia. [4]

Rajapinnan käyttöönotto ja alkuun pääseminen voi olla haastavaa etenkin aloittelevalle kehittäjälle. Rauf ym. luettelevat tutkimuksessaan *Perceived Obstacles by Novice Developers Adopting User Interface APIs and Tools* aloittelevien kehittäjien yleisimpiä haasteita rajapintojen käyttöönotossa. Ensimmäinen haaste liittyy kehitysympäristön käyttöönottoon: aloitteleville kehittäjille oikeanlaisen ohjelmointiympäristön ja konfiguraation määrittäminen voi muodostua esteeksi, joka pahimmillaan estää rajapinnan käytön aloittamisen kokonaan. Toisena haasteena on mahdollisuus suorittaa osittain valmista koodia saadakseen palautetta työn etenemisestä. Kyvyttömyys ottaa käyttöön sovelluksia ja suorittaa osittain valmista koodia johtaa kehittäjän turhautumiseen, ja on siten tärkeä ominaisuus rajapinnan ja sii-

hen liittyvien työkalujen suunnittelussa. Kolmantena haasteena ovat heikkolaatuiset kehitystyökalut, jotka voivat turhauttaa kehittäjää ja johtaa käytön lopettamiseen. Neljäntenä haasteena ovat täyttymättömät odotukset rajapinnan käyttöönotossa. Kehittäjällä on rajapinnasta tiettyjä odotuksia pohjautuen aiempiin kokemuksiin ja annettuihin lupauksiin, ja mikäli nämä odotukset eivät täyty, se voi johtaa tyytymättömyyteen. Viimeisenä haasteena on oppimateriaalien saatavuus. Mikäli käyttäjä ei löydä helposti tarkoituksenmukaisia oppimateriaaleja, se voi muodostua merkittäväksi esteeksi rajapinnan omaksumiselle. Helpolla oppimisella on vahva yhteys käyttäjätyytyväisyyteen. [2]

Ohjelmointirajapinnat muuttuvat ja kehittyvät uusien päivitysten ja versioiden myötä [13][18], mikä voi vaikuttaa sekä lukuisiin rajapinnan käyttäjiin että niiden lopputuotteiden käyttäjiin [18]. Näiden muutosten ennakoiminen ja niihin varautuminen voi vähentää kehitysprosessiin liittyviä haasteita [18]. Koska julkaistun rajapinnan muuttaminen on haastavaa laajan käyttäjäkunnan vuoksi, ja lisäksi rajapintaa voidaan käyttää tavoilla, joita niiden alkuperäinen suunnittelija ei ole osannut ottaa huomioon, käytettävyyden huolellinen arviointi ennen julkaisua on olennaista [1]. Myös arkkitehtuurisuunnittelu on ensisijaisen tärkeää, jotta kolmansien osapuolten, kuten kirjastojen, muutokset vaikuttavat vain pieneen osaan tiedostoja ja ovat helposti muutettavissa. Suuret sovellusversiopäivitykset voivat pahimmillaan hajottaa koko ohjelmiston ja rajapinnan palvelutarjoaja saattaakin pakottaa kehittäjän päivittämään koko ohjelmiston. [13]

Rajapinnan huono suunnittelu voi aiheuttaa tietoturvariskejä. Myös rajapinnan väärinkäyttö saattaa johtaa virheisiin ja sitä kautta tietoturvaongelmiin. [4] Julkisissa rajapinnoissa on enemmän turvallisuushaasteita kuin suljetuissa, sillä rajapinnan käyttäjiä voi olla tuhatmäärin enemmän kuin niiden kehittäjiä [4], ja käyttäjät eivät ole osallistuneet rajapinnan suunnitteluun eivätkä siten tunne rajapinnan rakennetta yhtä hyvin kuin sen alkuperäiset suunnittelijat [1].

OWASP on maailmanlaajuinen, voittoa tavoittelematon järjestö, joka pyrkii tietoturvan parantamiseen, ja se on tunnettu erityisesti sovellusturvallisuuteen liittyvistä listoistaan [19]. OWASP:n API Security Top 10 vuodelta 2023 [20] listaa rajapintojen 10 kriittisintä riskiä:

1. Rikkoutunut objektitason valtuutus (Broken Object Level Authorization): Hyökkääjä pääsee käsiksi resursseihin, joiden ei pitäisi olla käytettävissä. Tämä tapahtuu manipuloimalla pyynnön sisällä lähetetyn objektin tunnusta.
2. Rikkoutunut todennus (Broken Authentication): Todennus on toteutettu väärin, jolloin hyökkääjä pääsee käsiksi todennustunnuksiin tai voi hyödyntää toteutusvirheitä ottaakseen toisen käyttäjän identiteetin.
3. Rikkoutunut objektin ominaisuustason valtuutus (Broken Object Property Level Authorization): Puutteellinen tai virheellinen valtuutuksen tarkistaminen objektien ominaisuustasolla, mikä voi johtaa tietovuotoihin tai luvattomaan tiedon manipulointiin. Rajapinnat palauttavat usein kaikki objektin ominaisuudet, erityisesti REST-rajapinnoissa, mikä altistaa ne tietoturvariskeille.
4. Rajoittamaton resurssien kulutus (Unrestricted Resource Consumption): Useiden samanaikaisten pyyntöjen avulla tehty palvelunestohyökkäys, sillä rajapinta ei rajoita asiakkaiden vuorovaikutusta tai resurssien kulutusta.
5. Rikkoutunut toimintotason valtuutus (Broken Function Level Authorization): Monimutkaiset käyttöoikeusmallit, joissa on erilaisia hierarkioita, ryhmiä ja rooleja, voivat johtaa valtuutusvirheisiin. Hyökkääjä voi hyödyntää näitä puutteita saadakseen luvattoman pääsyn muiden käyttäjien resursseihin tai hallinnollisiin toimintoihin.
6. Rajoittamaton pääsy arkaluonteisiin liiketoimintaprosesseihin (Unrestricted Access to Sensitive Business Flows): Haavoittuvat rajapinnat mahdollistavat liiketoimintaprosessien, kuten lippujen ostamisen tai kommenttien julkaisemi-

sen, automatisoidun väärinkäytön ilman riittäviä suojatoimia. Tämä voi esimerkiksi estää laillisia käyttäjiä ostamasta tuotetta tai vääristää pelin sisäistä taloutta.

7. Palvelinpuolen pyyntöjen väärennys (Server Side Request Forgery): Etäresurssin hakeminen vahvistamatta käyttäjän toimittamaa URI:tä. Tämä voi johtaa esimerkiksi tietojen paljastumiseen tai palomuurien ohittamiseen.
8. Virheellinen tietoturvakonfiguraatio (Security Misconfiguration): Virheellisten tietoturva-asetusten johdosta hyökkääjä yrittää löytää rajapinnasta puutteita, yleisiä päätepisteitä, turvattomilla oletuskokoonpanoilla toimivia palveluita tai suojaamattomia tiedostoja saadakseen luvattoman pääsyn järjestelmään.
9. Puutteellinen API-inventaarinhallinta (Improper Inventory Management): Hyökkääjä voi saada luvattoman pääsyn vanhojen versioiden tai korjaamattomien päätepisteiden avulla, jotka käyttävät heikompia tietoturvakäytäntöjä. Vanhentunut dokumentaatio vaikeuttaa haavoittuvuuksien löytämistä.
10. Sovellusliittymien turvaton käyttö (Unsafe Consumption of APIs): Hyökkääjä etsii integroituja kolmannen osapuolen palveluita, joiden kautta hyökätä varsinaisen rajapinnan sijaan.

# 3 Avoin data

Avoin data on maksutonta dataa, joka on saatavilla kaikille ja joka on koneluettavassa muodossa. Kuka tahansa voi selata tietoaineistoja ja ladata niitä omaan käyttöönsä tai käyttää niitä pohjana uusien palveluiden luomiseen, kunhan lähde mainitaan. [21] Esimerkiksi reittioppaat hyödyntävät avoimia kartta-aineistoja ja ilmanlaadun seurantasovellukset avoimia sääaineistoja. Avointa dataa voi löytää erilaisista palveluista, esimerkiksi Digi- ja väestötietoviraston ylläpitämästä Avoindata.fi-palvelusta [3] sekä Yhdysvaltojen hallituksen Data.gov-palvelusta [22].

Ohjelmointirajapinnat ovat käytetyimpiä lähestymistapoja avoimen datan käyttämiseen [23]. Rajapintojen avulla kehittäjät pääsevät käsiksi avoimeen dataan [24].

## 3.1 Datan läpinäkyvyys ja tietojen saatavuus

Avoimella datalla on tärkeä merkitys valtioiden ja julkisen sektorin läpinäkyvyyden takaamisessa, sillä avoin data on tietoa, joka on vapaasti kaikkien käytettävissä. Julkisen sektorin läpinäkyvyys tarkoittaa sitä, että organisaatioita ja niiden toimintaa koskeva tieto on helposti saatavilla, jolloin ulkopuoliset toimijat voivat seurata ja arvioida organisaation suorituskykyä. Läpinäkyvyyttä voidaan edistää nykyaikaisilla tietojärjestelmillä, mikä auttaa maailmanlaajuisesti valtioita vähentämään korruptiota ja vahvistamaan vastuuvelvollisuutta kansalaisia kohtaan. Yksi keskeinen tapa lisätä hallinnon avoimuutta on julkisen hallinnon avoin data (Open Government Data, OGD). OGD on muovannut merkittävästi hallinnon pyrkimyksiä edistää

avoimuutta ja vastuullisuutta julkisten resurssien käytössä. Avoimet dataportaalit toimivat tässä keskeisenä välineenä, sillä ne mahdollistavat kansalaisille pääsyn julkisiin tietoihin. Näiden portaalien on välitettävä käyttäjille toiminnallisuuksia, joiden avulla tietoa voi helposti löytää, hyödyntää ja analysoida. [25]

Datan läpinäkyvyydellä tarkoitetaan myös tietoa siitä, mitä henkilötietoja kerätään, säilytetään ja jaetaan eteenpäin kolmansille osapuolille. Tämä periaate on myös tietosuoja-asetusten, kuten GDPR:n (General Data Protection Regulation) taustalla. Nämä säännökset määräävät tietosuojaperiaatteita, joita on noudatettava. [26] Yleinen tietosuoja-asetus GDPR asettaa tarkkoja vaatimuksia yrityksille ja organisaatioille niiden henkilötietojen hallinnointia koskien. Asetuksen tavoitteena on suojata yksilöiden oikeuksia ja varmistaa, että henkilötietojen käsittely on läpinäkyvää, turvallista ja perusteltua. Henkilötietoja ovat kaikki tiedot, josta henkilö voidaan suoraan tai epäsuoraan tunnistaa, kuten nimi, osoite, IP-osoite ja terveystiedot. Tietojen käsittely on sallittua vain, jos sille on laillinen peruste, kuten suostumus, sopimus tai lakisääteinen velvoite. Suostumuksen on oltava vapaaehtoinen, tietoinen ja yksiselitteinen, ja sen voi peruuttaa milloin tahansa. Lisäksi henkilötietojen käsittelystä on tiedotettava rekisteröidyille selkeästi ja ymmärrettävästi. [27]

Läpinäkyvyyden varmistamiseksi organisaatioiden on viestittävä avoimen datan käytöstään selkeästi ja ymmärrettävästi. Läpinäkyvyys ei tarkoita täydellistä avoimuutta kaikkien taustajärjestelmien, algoritmien tai tietojen käsittelyprosessien osalta. Sen sijaan se tarkoittaa ymmärrettävän ja saavutettavan tiedon jakamista avoimen datan tavoitteista, tuotantoprosesseista ja käyttötavoista. Organisaatiot voivat viestiä näistä asioista eri tavoin varmistaakseen, että käyttäjät ymmärtävät ja voivat hyväksyä sen, kuinka heidän tietojaan käytetään. [28]

## 3.2 FAIR-periaatteet

Avointa dataa tulisi pystyä uudelleenkäyttämään mahdollisimman paljon. Siksi vuonna 2016 onkin julkaistu kansainväliset FAIR-periaatteet laadukkaamman avoimen datan julkaisemiseksi. [29] Nämä periaatteet ovat perustana tieteellisen datan jakamisessa ja niiden tarkoituksena on tehdä datasta hyödyllistä myös muille kuin alkuperäiselle tekijälle. Lyhenne FAIR tulee sanoista Findable (Löydettävyys), Accessible (Saavutettavuus), Interoperable (Yhteentoimivuus) ja Re-usable (Uudelleenkäytettävyys). FAIR ei kosketa ainoastaan avointa dataa, vaan säännökset ohjeistavat datan jakamisessa erilaisissa yhteysprotokollissa. [30]

FAIR-periaatteissa esitetään 10 mittauskriteeriä datalle neljässä kategoriassa. *Löydettävyyden* varmistamiseksi jokaisella tietoaineistolla on oltava pysyvä ja yksilöllinen tunniste (PID), joka toimii maailmanlaajuisesti ja linkittyy suoraan metatietoihin. Lisäksi metatietojen tulee olla kattavat ja standardoitujen periaatteiden mukaiset, jotta ne voidaan rekisteröidä ja indeksoida hakujärjestelmiin. [29][30]

*Saavutettavuus* puolestaan edellyttää, että sekä metatiedot että tietoaineisto ovat haettavissa avoimen, standardoidun ja maailmanlaajuisesti käytettävän protokollan avulla. Mikäli tietojen käyttö edellyttää tunnistamista ja valtuuttamista, prokollan on tuettava näitä toimintoja. Lisäksi metatiedot tulee säilyttää myös siinä tapauksessa, että itse tietoaineisto poistetaan, jotta viittaukset ja yhteydet muihin resursseihin säilyvät. [29][30]

Tietoaineiston *yhteentoimivuuden* varmistamiseksi sen tulee hyödyntää standardoituja, laajasti käytössä olevia ja rakenteellisesti yhtenäisiä tapoja tiedon esittämiseen. Metatietojen on perustuttava FAIR-periaatteiden mukaisiin sanastoihin ja lisäksi niiden tulee sisältää määriteltyjä viittauksia muihin metatietoihin. [29][30]

Lopuksi, *uudelleenkäytettävyyden* takaamiseksi metatietojen on oltava mahdollisimman monipuolisia, tarkkoja ja osuvia, jotta ne tukevat aineiston tehokasta hyödyntämistä eri käyttötarkoituksiin. Tietoaineisto ja sen metatiedot on julkaistava

selkeän lisenssin alaisina, jotta niiden käyttöehdot ovat yksiselitteiset. Lisäksi tietoaineiston alkuperän tulee olla tarkasti määritelty. Uudelleenkäytettävyyttä tukee myös se, että sekä metatiedot että tietoaineisto noudattavat oman aihepiirinsä parhaita käytäntöjä ja ohjeistuksia, mikä edistää niiden laadukasta ja kestävää käyttöä. [29][30]

# 4 Google Mapsin ja AccuWeatherin dokumentaatioiden eroavaisuudet

Vaikka avoimet rajapinnat mahdollistavat tehokkaan ja modulaarisen ohjelmistokehityksen, niiden onnistunut hyödyntäminen riippuu paljolti dokumentaation selkeydestä ja käytettävyydestä. Hyvin laadittu dokumentaatio tukee kehittäjien työskentelyä, loiventaa oppimiskäyrää ja vähentää virheiden mahdollisuutta. [24][31]

Tässä vertailussa tutkitaan, kuinka helppokäyttöisiä Google Mapsin ja AccuWeatherin rajapinnat ovat sovelluskehittäjälle, sekä mitä eroavaisuuksia Google Mapsin ja AccuWeatherin dokumentaatioissa on. Ominaisuuksia analysoidaan seitsemän eri näkökulman kautta, jotka valittiin aiempaan tutkimuskirjallisuuteen ja käytettävyyden arviointikriteereihin perustuen.

Google Maps [32] mahdollistaa sijaintipohjaisten rajapintojen käytön, joiden avulla käyttäjät voivat hyödyntää maantieteellisesti tarkkoja ilma-, satelliitti- ja katunäkymiä. Rajapinnat kattavat useita eri osa-alueita, kuten karttojen esittäminen, reittidatan hakeminen ja liikennetietojen hyödyntäminen. AccuWeather [33] on sääpalvelu, jonka rajapinnat välittävät käyttäjälle ajankohtaista säädataa, sääennusteita, erilaisia sääindeksejä sekä satelliittikuvia. AccuWeatherin rajapinnat keskittyvät erityisesti sääolosuhteiden reaaliaikaiseen seurantaan ja ennustamiseen.

### 1. Aloittaminen ja käytön aloituskynnys

Tarkasteltavien rajapintojen käyttöönotossa keskeisiä vaiheita ovat tilin luominen, laskutuksen aktivointi ja tarvittavien sovellusliittymien tai avainten hankkiminen. Google Maps- ja AccuWeather-rajapintojen aloitusprosessit noudattavat samankaltaisia perusperiaatteita, mutta niissä on myös eroja.

**Google Maps** Koska rajapintavaihtoehtoja on useita, tässä kohdassa keskitytään tarkastelemaan karttarajapinnan Maps JavaScript API käyttöönottoa. Rajapinnan käyttö edellyttää ensin projektin luomista Google Cloud Console -ympäristössä. Käytön aloittamiseksi on aktivoitava laskutus, vaikka rajapinnan käyttö pysyy maksuttomana niin kauan kun kuukausittainen maksuton kiintiö ei ylity. Lisäksi projektiin on otettava käyttöön tarvittavat sovellusliittymät ja ohjelmistokehityspaketit (SDK:t), jotka mahdollistavat haluttujen toimintojen hyödyntämisen.

Seuraavaksi luodaan yksilöllinen API-avain, joka toimii tunnisteena ja todentaa projektiin liittyvät pyynnöt sekä mahdollistaa laskutuksen hallinnan ja luvattoman käytön eston. API-avain tulee sisällyttää jokaiseen rajapintakutsuun, jotta Google voi tunnistaa sovelluksen ja myöntää pääsyn palveluun.

Lopuksi on ladattava rajapinta Maps JavaScript API, ja tähän on useita eri vaihtoehtoja. Ensimmäinen tapa on käyttää dynaamista kirjaston tuontia, joka mahdollistaa tarvittavien kirjastojen lataamisen ajon aikana. Tämä voi parantaa suorituskykyä, sillä vain tarvittavat osat ladataan sovelluksen käyttötilanteen mukaan. Toinen vaihtoehto on hyödyntää komentosarjan suoralataustunnistetta, joka lataa kaikki tarvittavat kirjastot kerralla heti, kun komentosarja suoritetaan. Tämä menetelmä voi olla hyödyllinen, jos sovellus tarvitsee useita eri kirjastokomponentteja välittömästi käytettäväksi. Kolmas vaihtoehto on käyttää NPM-pakettihallintaa, jossa rajapinta asennetaan yksinkertaisella komennolla: "npm install @googlemaps/js-api-loader".

Käytön aloittamiseen on saatavilla kattavat ohjeet, vaikka dokumentaation laajuus voi alkuvaiheessa tuntua haastavalta rajapintojen suuren määrän vuoksi. Oppimiskäyrä voi olla jyrkkä etenkin aloittelijalle. Kehittäjän on syytä perehtyä dokumentaatioon huolellisesti ennen käyttöönottoa.

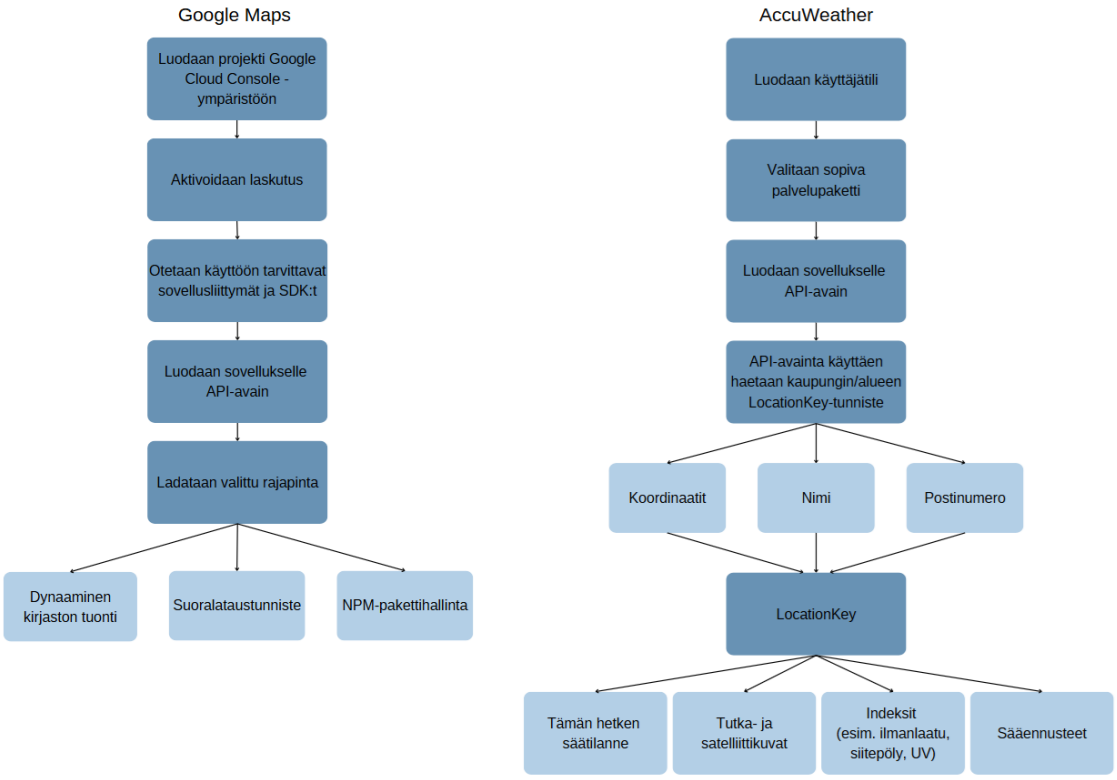
**Accuweather** Rajapinnan käyttöönotto alkaa käyttäjätilin luomisella AccuWeatherin palveluun. Tämän jälkeen valitaan sopiva palvelupaketti, jonka myötä laskutus aktivoituu. Tarjolla on sekä maksuttomia että maksullisia toimintoja.

Myös AccuWeatherissa on luotava tunnisteena toimiva API-avain, jotta sovellus voi tehdä API-kutsuja. Sijaintipohjaisten säätietojen hakemiseksi käytetään ensin Locations-APIa, joka palauttaa yksilöllisen LocationKey-tunnisteen. Tämä tunniste haetaan joko koordinaattien, postinumeron tai kaupungin nimen perusteella. Esimerkiksi tekstipohjaisella haulla kaupungin nimen perusteella voidaan hakea tietyn kaupungin tiedot. Saadun LocationKey:n avulla voidaan suorittaa jatkokutsuja esimerkiksi sääennusteiden tai ajankohtaisen säätilan hakemiseksi. Esimerkiksi tietyn sijainnin päivittäinen ennustetieto voidaan hakea suoralataustunnisteella "http://dataservice.accuweather.com/forecasts/v1/daily/1day/LocationKey".

Käyttöönotto on varsin suoraviivainen ilman ylimääräisiä vaiheita. Aloitusprosessin jälkeen rajapintojen käyttö keskittyy pääasiassa LocationKey-tunnisteen hyödyntämiseen.

AccuWeatherilla on myös erillinen kehittäjäsivusto AccuWeather Enterprise, joka on tarkoitettu pääasiassa yrityksille liiketoimintakäyttöä varten [34]. Tällä sivustolla on lueteltu kaikki rajapintaparametrit ja vastausparametrit selkeästi ja sivusto sisältää enemmän tietoa kuin rajoitetumpi, kaikille saatavilla oleva julkinen rajapinta.

Alla olevassa kuvassa havainnollistetaan käytön aloittamisen eroja visuaalisella kaaviolla.




(a) Käytön aloittaminen Google Mapsilla ja Accuweatherilla.

### 2. Dokumentaation selkeys ja kattavuus

Rajapinnan dokumentaation laajuus ja selkeys vaikuttavat huomattavasti sen käytettävyyteen ja kehittäjäkokemukseen. Google Maps- ja AccuWeather-rajapintojen dokumentaatioissa on eroja eritoten sisällön kattavuuden ja laajuuden osalta.

**Google Maps** Google Mapsin dokumentaatio on hyvin kattava. Se sisältää askel askeleelta -ohjeita, käytännön esimerkkejä, kuvia ja havainnollistuksia eri rajapinnoista. Koska Google Maps sisältää laajan valikoiman erilaisia rajapintoja ja toimintoja, dokumentaatio sisältää valtavan määrän tietoa, mikä voi olla paitsi hyödyllistä myös haastavaa hahmottaa. Hakutoimintoa on mahdollista käyttää tietoa etsiessä.

Dokumentaatio sisältää API Picker -ominaisuuden, joka helpottaa oikean rajapinnan valintaa kehittäjän tarpeiden mukaan. API Picker sisältää Google Mapsin rajapintavaihtoehdot, kuten kartan lisäämisen sovellukseen, osoitelomakkeen au-

tomaattitäytön ja nopeusrajoitusdataa. Erilaisia rajapintoja löytyy 51 kappaletta. Rajapintojen avulla on saatavilla osittain samaa dataa kuin AccuWeatherin kautta, kuten siitepöly- ja ilmanlaatutietoja.

Eri rajapinnat sisältävät oman hakemiston kaikista kyseisen rajapinnan funktioista, parametreista ja palautusarvoista. Nämä näyttävät hieman erilaisilta riippuen siitä, mitä rajapintaa tarkastellaan. Esimerkiksi Maps JavaScript APIn hakemisto on luokiteltu pääotsikon mukaan, kuten "Coordinates"ja "Errors", ja otsikoiden alla on listattuna luokat ja luokkien sisällä metodit. Dokumentti on kehittäjälle siisti ja helppolukuinen.

Google Mapsin dokumentaatio on kattava ja hyvin laadittu, antaen runsaasti esimerkkejä ja käyttäjätukea kehittäjälle. Kuitenkin juuri tämän laajuuden vuoksi dokumentaatiosta voi olla haastavaa löytää nopeasti tarvittavaa tietoa, etenkin uusilla käyttäjillä.

**Accuweather** AccuWeatherin dokumentaatio on yksinkertaisempi, mikä johtuu siitä, että rajapintoja on vähemmän ja niiden ominaisuudet ovat suppeammat verrattuna Google Mapsiin. Dokumentaation rakenne voi aluksi vaikuttaa hieman sekavalta, sillä aloitusohjeet löytyvät ainoastaan etusivulla olevasta linkistä, eikä navigaatiopalkissa ole ohjeeseen linkkiä lainkaan. Kuitenkin, kun tarvittavat aloitusohjeet löytyvät, ne osoittautuvat varsin suoraviivaisiksi. AccuWeatherilla ei ole yhtä hienostunutta hakutoimintoa kuin Googlella, mutta sillä pystyy kuitenkin suorittamaan perushakuja dokumentaatiosta.

Eri rajapintojen kutsut on lueteltu Reference-sivulla, johon on luokiteltu erikseen Locations API ja muut ennusteita tuottavat rajapinnat. Sijaintihaku on monivaiheinen, sillä kehittäjän on ensin tehtävä erillinen kutsu location APIin saadakseen locationKey-avaimen säädatan hakemiseen. Kaikki säätietoa palauttavat rajapintapyynnöt vaativat locationKey-avaimen. Tämä voi olla hämmentävää ellei kehittäjä ole perehtynyt ensin aloitusohjeeseen. Tämä myös lisää yhden ylimääräisen vaiheen

datan hakemiseen.

### 3. API:en tekniset ominaisuudet

Rajapinnoissa on teknisiä eroja. Rajapintasuunnittelussa keskeisiä näkökulmia ovat esimerkiksi käytetty arkkitehtuuri, vasteajat, tietomuodot sekä datan optimointimenetelmät.

**Google Maps** Google Maps on ensisijaisesti REST-pohjainen, mutta se hyödyntää myös gRPC-teknologiaa (open source high performance Remote Procedure Call), joka mahdollistaa tehokkaamman tiedonsiirron erityisesti suurten tietomäärien käsittelyssä.

Google Maps tukee reittihakujen yhteydessä käytettävää kenttämaskia (engl. field mask), joka rajaa palautettavaa dataa. Mikäli kenttämaskia ei ole määritetty lainkaan, rajapinta palauttaa virheviestin. Tämä lähestymistapa estää tarpeettoman datan palauttamisen ja tehostaa resurssien käyttöä, mikä on tärkeää, sillä tietomäärä vaikuttaa rajapinnan kustannuksiin.

Virheidenkäsittelyyn voi käyttää ns. eksponentiaalista peruutusta (engl. exponential backoff). Mikäli sovellus yrittää tehdä liikaa pyyntöjä lyhyen ajan sisällä, kyselyiden tahtiin lisätään viivettä, jolloin kysely palautetaan takaisin ilman virhettä.

**Accuweather** AccuWeather perustuu yksinomaan REST-arkkitehtuuriin. Datamäärää säädellään GZIP-pakkauksella (engl. GZIP Compression), joka on menetelmä HTTP-vastausten pakkaamiseen ennen niiden lähettämistä asiakkaalle. Tämä vähentää siirrettävän datan määrää ja nopeuttaa API-kutsujen vasteaikoja. Tiedot palautetaan hyvin jäsennellyssä JSON-muodossa.

**4. API-kutsujen yksinkertaisuus ja loogisuus**

Google Maps- ja AccuWeather-rajapintojen toimintaperiaatteet eroavat toisistaan erityisesti tiedon hakemisen ja palauttamisen toteutustavoissa.

**Google Maps** Google Mapsin API-kutsut ovat pääosin loogisia ja hyvin dokumentoituja. Ne ovat selkeästi nimettyjä ja noudattavat selkeää rakennetta, joskin jotkin kutsut ovat melko pitkiä. Kuitenkin, koska Google Maps sisältää monipuolisia ja kehittyneitä toiminnallisuuksia, API-kutsujen monimutkaisuus kasvaa erityisesti edistyneempiä ominaisuuksia käytettäessä. Dokumentaatiossa on paljon esimerkkejä, mutta suuren tietomäärän vuoksi halutun tiedon löytäminen voi viedä aikaa. Kutsuissa on myös AccuWeatheria monimutkaisemmat parametrit, kuten useita koordinaattipisteitä tai eri liikennemuotoja reitityksessä.

**Accuweather** AccuWeatherin API-kutsut ovat Google Mapsia yksinkertaisempia, sillä kaikki kutsut perustuvat GET-pyyntöihin, ja ne tarvitsevat vain paikkatunnisteen ja palauttavat sen perusteella tarvittavan sääennusteen. Kutsut on nimetty johdonmukaisesti siten, että niiden rakenne vastaa haettavaa tietoa ja tämän vuoksi ne voivat olla pitkiä. Esimerkiksi 10 päivän sääennusteen päiväaikaisen maksimaalisen kosteusprosentin hakemiseen käytetään kutsua *DailyForecasts.Day.RelativeHumidity.Maximum*, kun taas auringonnousuajan hakemiseen kutsua *DailyForecasts.Sun.Rise*. Kutsujen pituudesta huolimatta looginen nimeämiskäytäntö helpottaa rajapinnan käyttöä ja parantaa sen ennakoitavuutta kehittäjälle.

Päätepisteet (endpointit) on jaoteltu selkeästi, esimerkiksi ennusteiden pyytäminen eri ajanjaksoihin (1 päivä, 5 päivää, 12 tuntia). Rajapinnan palauttamaa vastausta on mahdollisuus tarkentaa, oletusarvoisesti palautetaan tiivistetty versio ennustedatasta, mutta yksityiskohtaisemmat tiedot ovat saatavilla lisäämällä URL-osoitteeseen parametri *details=true*.

**5. Tuen ja kehittäjäresurssien saatavuus**

Rajapinnan käytettävyyteen vaikuttaa saatavilla oleva tuki ja kehittäjäresurssit, kuten esimerkit ja yhteisön apu. Google Maps- ja AccuWeather-rajapinnat eroavat huomattavasti tässä asiassa.

**Google Maps** Google Mapsilla on laaja tukijärjestelmä kehittäjille. Dokumentaatio sisältää runsaasti esimerkkejä API-kutsujen toteuttamisesta, mikä helpottaa rajapinnan käyttöönottoa ja integrointia. Lisäksi Google Maps Platform Support and Resources -osiossa on usein kysytyt kysymykset (FAQ), yleisimpiin ongelmiin liittyviä ratkaisuja sekä pääsy aktiiviseen kehittäjäyhteisöön, Stack Overflow -foorumiin. Tarvittaessa käyttäjillä on myös mahdollisuus ottaa yhteyttä Google Mapsin asiakastukeen.

**Accuweather** AccuWeatherin rajapinnan tukiresurssit ovat selvästi rajallisemmat. Sillä ei ole varsinaista käyttäjätukea tai asiakaspalvelua, vaan ainoastaan usein kysytyt kysymykset (FAQ).

**6. Rajoitukset**

Rajapintojen käyttöön liittyy usein erilaisia rajoituksia, jotka vaikuttavat niiden soveltuvuuteen ja tehokkuuteen eri käyttötapauksissa. Nämä rajoitukset liittyvät esimerkiksi pyyntömääriin, hinnoitteluun, suorituskykyyn ja tietojen saatavuuteen ja käyttäjäehtoihin.

**Google Maps** Rajapinnan käyttö maksaa, kun tietty kuukausikiintiö tulee täyteen. Ensimmäiset 10 000 tapahtumaa ovat maksuttomia, minkä jälkeen hinnoittelu perustuu yksikkökohtaisiin kustannuksiin, jotka määräytyvät dollareina 1 000 tapahtumaa kohden.

Google Mapsin käyttäjäehtojen mukaan rajapinnan käyttö tulee pääsääntöisesti olla maksutonta ja julkisesti saatavilla. Poikkeuksena ovat maksulliset mobiilisovellukset ja erilliset yrityssopimukset Googlen kanssa. Luvaton käyttö on estettävä ja rikkomuksista vastattava itse.

**Accuweather** Rajapinnan käyttö maksaa, mikäli päivässä tapahtuu yli 50 pyyntöä. Maksuttomaan pakettiin sisältyvät viiden päivän sääennusteet, 12 tunnin tuntikohtaiset ennusteet sekä erilaiset sääindeksit. Mikäli käyttöön tarvitaan lisäominaisuuksia, kuten säähälytykset, trooppisten syklonien seuranta tai karttapalvelut, niiden käyttö edellyttää erillistä kuukausimaksua.

Palautettava päivämäärä on UCT-muodossa, mistä johtuen sen käsittely voi olla monimutkaista ja vaatia lisäkäsittelyä erityisesti alueellisissa sovelluksissa.

AccuWeatherin käyttäjäehtojen mukaan rajapintoja ei saa muuttaa, myydä eikä yhdistää muihin säädatalähteisiin. Sopimustilaukset uusiutuvat automaattisesti, mikäli niitä ei peruuteta ennen uuden maksukauden alkua. Käyttäjä ei saa säilyttää rajapinnasta johdettua dataa ilman lupaa.

### 7. Todennus ja tietoturva

Rajapintojen suojaamisella estetään väärinkäytökset, kuten tietovuodot, palvelunestohyökkäykset ja luvaton käyttö. Google Mapsilla ja AccuWeatherilla on hyvin eri tasoisia ratkaisuja tietoturvaan.

**Google Maps** Luvattomalta käytöltä suojataan käyttämällä todennustietoja API-kutsuihin. Vähintään yksi API-avain on pakollista luoda todennusta ja laskutusta varten. Avainten käyttö rajoitetaan vain niihin rajapintoihin, joita sovellus tarvitsee, ja kehittäjä on taloudellisessa vastuussa rajoittamattomien avainten väärinkäytöstä. Rajoituksia voi luoda koskemaan vain tiettyjä nettisivuja, IP-osoitteita tai ainoastaan Android- tai iOS-sovelluksia.

Käyttöön voi ottaa myös digitaalisen allekirjoituksen, joka varmistaa, että kaikki API-avaimella pyyntöjä tekevät sivustot ovat valtuutettuja siihen. Allekirjoitus on yksityinen avain, jonka voi luoda Google Cloud -konsolista löytyvällä URL-allekirjoituksen salaisuudella. Allekirjoitusprosessi perustuu salausalgoritmiin, joka yhdistää URL-osoitteen ja jaetun salaisuuden.

Lisäksi Firebase App Check suojaa sovellusta estämällä liikenteen muualta kuin laillisista sovelluksista. Tämä tapahtuu ReCAPTCHAn avulla. ReCAPTCHA on kehittyneisiin riskianalyysitekniikoihin perustuva palvelu, jonka tarkoitus on erottaa ihmiset ja botit toisistaan, ja siten suojata roskapostilta ja väärinkäytöksiltä. Firebase App Check on tarkoitettu ainoastaan julkisille sovelluksille.

**Accuweather** Myös AccuWeatheriin luodaan API-avain todennusta, laskutusta ja tietoturvaa varten. Erillisiä rajoituksia, kuten Google Mapsissa, ei voi kuitenkaan avaimille asettaa.

Kehittäjäsivuston (AccuWeather Enterprise) rajapinnat tukevat HTTPS-protokollaa, mikä mahdollistaa salatun tiedonsiirron. Muita erityisiä tietoturvaominaisuuksia kehittäjää varten ei löydy.

# 5 Yhteenveto ja pohdinta

Alussa asettamamme tutkimuskysymykset olivat:
TK1: Kuinka avoimet rajapinnat helpottavat sovellusten kehittämistä?
TK2: Mitä haasteita ja ongelmia avoimissa rajapinnoissa on?
TK3: Miten kehittäjäkokemus ja käytettävyys eroavat Google Mapsin ja AccuWeatherin rajapintojen välillä?

Tutkimuskysymykseen 1 saatiin vastattua yksimielisesti. Ohjelmointirajapinnat ovat niin tärkeä osa kehittäjän arkea, että niiden hyödyllisyyttä ei kirjallisuudessa kyseenalaisteta. Niiden avulla saadaan nopeutettua kehitystyötä uudelleenkäyttämällä ohjelmistokomponentteja, yhdistettyä sovelluksia internetin kautta ja hyödynnettyä avointa dataa.

Rajapinnat tuovat kuitenkin mukanaan myös haasteita, kuten tutkimuskysymyksessä numero 2 huomattiin. Rajapinnat voivat olla kehittäjälle vaikeakäyttöisiä, joko huonon dokumentaation, vähäisten oppimismateriaalien tai heikon rajapintasuunnittelun takia. Vaikeasti opittava ja epäkäytännöllinen rajapinta jää usein käyttämättä kokonaan. Rajapinnoissa voi olla myös haavoittuvaisuuksia ja niiden päivittäminen voi olla hankalaa ja se voi pahimmassa tapauksessa rikkoa koko ohjelmiston.

Tutkimuskysymyksessä 3 pohdittiin eroavaisuuksia Google Mapsin ja AccuWeatherin rajapintojen käyttäjäkokemuksessa ja käytettävyydessä kehittäjän näkökul-

masta. Näissä rajapinnoissa oli suuria eroja niin dokumentaation laajuudessa kuin käyttäjätuessakin. AccuWeather oli huomattavasti yksinkertaisempi ja vähemmän ominaisuuksia sisältävä rajapinta, kun taas Google Maps sisälsi valtavan dokumentaation, mutta myös kattavan tuen sen käyttämiseen. Kumpaankin löytyi aloitusohje, jonka turvin kehittäjä pääsee rajapinnan käytössä alkuun. Alla olevassa taulukossa havainnollistetaan tiiviissä muodossa näitä eroavaisuuksia.

Taulukko 5.1: Google Mapsin ja AccuWeatherin dokumentaatioiden eroavaisuudet tiivistetysti.

| **Ominaisuudet** | **Google Maps** | **AccuWeather** |
|---|---|---|
| Aloittamisen helppous | Alkuun voi olla raskas suuren tietomäärän vuoksi | Selkeämpi alkuun, koska toiminnot ovat rajatumpia |
| Dokumentaation laatu | Hyvin jäsennelty, paljon esimerkkejä ja ohjeita | Selkeä, mutta vähemmän yksityiskohtainen |
| Tekniset ominaisuudet | REST- ja gRPC-pohjainen. Mahdollisuus rajata palautettavaa dataa ja käsitellä virheitä | REST-pohjainen. Datamäärää voi rajata GZIP-pakkauksella |
| API-kutsut | Loogisia, hyvin dokumentoituja, selkeästi nimettyjä. Kutsut monimutkaisia edistyneempiä ominaisuuksia käytettäessä | Johdonmukaisesti nimettyjä, hyvin dokumentoituja. Jotkin kutsut pitkiä, mutta silti loogisia |
| Käyttäjätuki ja yhteisö | Suuri käyttäjäyhteisö, virallinen tuki ja Stack Overflow | Ei yhteisöapua, enemmän yritysasiakaslähtöinen |
| Rajoitukset | 10 000 maksutonta tapahtumaa kuukaudessa | 50 maksutonta pyyntöä päivässä. Maksullisia lisäominaisuuksia |
| Tietoturva | API-avaimen luominen. Avainten käytön rajoittaminen. Digitaalinen allekirjoitus. Firebase App Check | API-avaimen luominen. HTTPS-protokolla yritysasiakkaille |

Kirjallisuudessa ohjelmointirajapintojen negatiiviset puolet tuleva helpommin esille ja niistä on tehty enemmän nykyaikaista tutkimusta kuin positiivisista puo-

lista. Tämä johtuu todennäköisesti siitä, että rajapintojen positiiviset puolet ovat niin itsestäänselviä, ettei niistä ole tehty yhtä paljon tutkimusta, tai tutkimus on tehty kauan aikaa sitten, kun rajapinnat ovat olleet vielä uusi tutkimuskohde. Rajapintojen tuomien etujen lisätutkimus tuskin on tarpeen, sillä rajapinnat ovat niin yksiselitteisesti hyödyllisiä. Myös rajapintojen käytettävyysongelmat ovat jo varsin tutkittu kohde. Sen sijaan avoin data voisi kaivata lisätutkimusta esimerkiksi avoimen datan hyödyistä ja haitoista yrityskäytössä.

# Lähdeluettelo


[1] I. Rauf, E. Troubitsyna ja I. Porres, ”A systematic mapping study of API usability evaluation methods”, *Computer Science Review*, vol. 33, s. 49–68, 1. elokuuta 2019, ISSN: 1574-0137. DOI: 10.1016/j.cosrev.2019.05.001. url: https://www.sciencedirect.com/science/article/pii/S1574013718301515 (viitattu 07. 02. 2025).

[2] I. Rauf, P. Perälä, J. Huotari ja I. Porres, ”Perceived obstacles by novice developers adopting user interface APIs and tools”, teoksessa *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, ISSN: 1943-6106, syyskuu 2016, s. 223–227. DOI: 10.1109/VLHCC.2016.7739689. url: https://ieeexplore.ieee.org/document/7739689 (viitattu 20. 02. 2025).

[3] ”Avoindata.fi”. (), url: https://www.avoindata.fi/en (viitattu 10. 04. 2025).

[4] B. A. Myers ja J. Stylos, ”Improving API usability”, *Commun. ACM*, vol. 59, nro 6, s. 62–69, 23. toukokuuta 2016, ISSN: 0001-0782. DOI: 10.1145/2896587. url: https://dl.acm.org/doi/10.1145/2896587 (viitattu 21. 02. 2025).

[5] C. R. B. de Souza ja D. F. Redmiles, ”On the roles of APIs in the coordination of collaborative software development”, *Computer Supported Cooperative Work (CSCW)*, vol. 18, nro 5, s. 445, 16. syyskuuta 2009, ISSN: 1573-7551. DOI: 10.1007/s10606-009-9101-3. url: https://doi.org/10.1007/s10606-009-9101-3 (viitattu 12. 03. 2025).

[6] O. Borgogno ja G. Colangelo, ”Data sharing and interoperability: Fostering innovation and competition through APIs”, *Computer Law & Security Review*, vol. 35, nro 5, s. 105 314, 1. lokakuuta 2019, ISSN: 0267-3649. DOI: 10.1016/j.clsr.2019.03.008. url: https://www.sciencedirect.com/science/article/pii/S0267364918304503 (viitattu 13. 03. 2025).

[7] A. Kamruzzaman, K. Thakur ja M. L. Ali, ”Cybersecurity Threats using Application Programming Interface (API)”, teoksessa *2024 International Conference on Computing, Internet of Things and Microwave Systems (ICCIMS)*, heinäkuu 2024, s. 1–6. DOI: 10.1109/ICCIMS61672.2024.10690413. url: https://ieeexplore.ieee.org/document/10690413 (viitattu 03. 04. 2025).

[8] S. Schwichtenberg, C. Gerth ja G. Engels, ”From Open API to Semantic Specifications and Code Adapters”, teoksessa *2017 IEEE International Conference on Web Services (ICWS)*, kesäkuu 2017, s. 484–491. DOI: 10.1109/ICWS.2017.56. url: https://ieeexplore.ieee.org/abstract/document/8029798 (viitattu 25. 02. 2025).

[9] R. Ramanathan ja T. Korte, ”Software service architecture to access weather data using RESTful web services”, teoksessa *Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, heinäkuu 2014, s. 1–8. DOI: 10.1109/ICCCNT.2014.6963122. url: https://ieeexplore.ieee.org/document/6963122 (viitattu 04. 03. 2025).

[10] C. Pautasso, O. Zimmermann ja F. Leymann, ”Restful web services vs. "big" web services: making the right architectural decision”, teoksessa *Proceedings of the 17th international conference on World Wide Web*, sarja WWW ’08, New York, NY, USA: Association for Computing Machinery, 21. huhtikuuta 2008, s. 805–814, ISBN: 978-1-60558-085-2. DOI: 10.1145/1367497.1367606. url: https://dl.acm.org/doi/10.1145/1367497.1367606 (viitattu 11. 03. 2025).

[11] M. Pantelelis ja C. Kalloniatis, ”Create, Read, Update, Delete: Implications on Security and Privacy Principles regarding GDPR”, teoksessa *Proceedings of the 19th International Conference on Availability, Reliability and Security*, sarja ARES ’24, New York, NY, USA: Association for Computing Machinery, 30. heinäkuuta 2024, s. 1–7. DOI: 10.1145/3664476.3670898. url: https://dl.acm.org/doi/10.1145/3664476.3670898 (viitattu 26. 03. 2025).

[12] L. Li ja W. Chou, ”Design and Describe REST API without Violating REST: A Petri Net Based Approach”, teoksessa *2011 IEEE International Conference on Web Services*, heinäkuu 2011, s. 508–515. DOI: 10.1109/ICWS.2011.54. url: https://ieeexplore.ieee.org/abstract/document/6009431 (viitattu 25. 02. 2025).

[13] T. Espinha, A. Zaidman ja H.-G. Gross, ”Web API growing pains: Stories from client developers and their code”, teoksessa *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, helmikuu 2014, s. 84–93. DOI: 10.1109/CSMR-WCRE.2014.6747228. url: https://ieeexplore.ieee.org/abstract/document/6747228 (viitattu 25. 02. 2025).

[14] ”Web Services Atomic Transaction Version 1.1”. (), url: https://docs.oasis-open.org/ws-tx/wstx-wsat-1.1-spec-os/wstx-wsat-1.1-spec-os.html (viitattu 23. 04. 2025).

[15] ”Web Services Security: SOAP Message Security Version 1.1.1”. (), url: https://docs.oasis-open.org/wss-m/wss/v1.1.1/os/wss-SOAPMessageSecurity-v1.1.1-os.html (viitattu 23. 04. 2025).

[16] ”ISO 9241-11:2018(en), Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts”. (), url: https://www.iso.org/obp/ui/en/#iso:std:iso:9241:-11:ed-2:v1:en (viitattu 04. 03. 2025).

[17] J. Nielsen, *Usability Engineering*. Morgan Kaufmann, 23. syyskuuta 1994, 382 s., Google-Books-ID: 95As2OF67f0C, ISBN: 978-0-12-518406-9.

[18] J.-P. Joutsenlahti, T. Lehtonen, M. Raatikainen, E. Kettunen ja T. Mikkonen, "Challenges and Governance Solutions for Data Science Services based on Open Data and APIs", teoksessa *2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)*, toukokuu 2021, s. 1–4. DOI: 10.1109/WAIN52551.2021.00012. url: https://ieeexplore.ieee.org/document/9474387 (viitattu 07. 02. 2025).

[19] "OWASP foundation, the open source foundation for application security | OWASP foundation". (), url: https://owasp.org/ (viitattu 10. 04. 2025).

[20] "OWASP Top 10 API Security Risks – 2023 - OWASP API Security Top 10". (), url: https://owasp.org/API-Security/editions/2023/en/0x11-t10/ (viitattu 10. 04. 2025).

[21] "Open data for citizens | avoindata.fi". (), url: https://www.avoindata.fi/en/user-guide/open-data-for-citizens (viitattu 27. 02. 2025).

[22] "Data.gov home", Data.gov. (), url: https://data.gov/ (viitattu 10. 04. 2025).

[23] C. González-Mora, I. Garrigós ja J. Zubcoff, "An APIfication approach to facilitate the access and reuse of open data", teoksessa *Web Engineering*, ISSN: 1611-3349, Springer, Cham, 2020, s. 512–518, ISBN: 978-3-030-50578-3. DOI: 10.1007/978-3-030-50578-3_36. url: https://link.springer.com/chapter/10.1007/978-3-030-50578-3_36 (viitattu 11. 02. 2025).

[24] C. González-Mora, C. Barros, I. Garrigós, J. Zubcoff, E. Lloret ja J.-N. Mazón, "Improving open data web API documentation through interactivity and natural language generation", *Computer Standards & Interfaces*, vol. 83, s. 103 657, tammikuu 2023, ISSN: 09205489. DOI: 10.1016/j.csi.2022.103657. url:

https://linkinghub.elsevier.com/retrieve/pii/S0920548922000344 (viitattu 11. 02. 2025).

[25] M. Lnenicka ja A. Nikiforova, "Transparency-by-design: What is the role of open data portals?", *Telematics and Informatics*, vol. 61, s. 101 605, 1. elokuuta 2021, ISSN: 0736-5853. DOI: 10.1016/j.tele.2021.101605. url: https://www.sciencedirect.com/science/article/pii/S0736585321000447 (viitattu 11. 03. 2025).

[26] E. Grünewald, P. Wille, F. Pallas, M. C. Borges ja M.-R. Ulbricht, "TIRA: An OpenAPI Extension and Toolbox for GDPR Transparency in RESTful Architectures", teoksessa *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, ISSN: 2768-0657, syyskuu 2021, s. 312–319. DOI: 10.1109/EuroSPW54576.2021.00039. url: https://ieeexplore.ieee.org/abstract/document/9583685 (viitattu 27. 02. 2025).

[27] "Yleinen tietosuoja-asetus (GDPR)", Your Europe. (), url: https://europa.eu/youreurope/business/dealing-with-customers/data-protection/data-protection-gdpr/index_fi.htm (viitattu 11. 03. 2025).

[28] "Trust in open data applications through transparency - Christian Wiencierz, Marco Lünich, 2022". (), url: https://journals-sagepub-com.ezproxy.utu.fi/doi/full/10.1177/1461444820979708 (viitattu 11. 03. 2025).

[29] "FAIR-periaatteet | avoindata.fi". (), url: https://www.opendata.fi/fi/opas/tietoaineiston-fair-periaatteet (viitattu 27. 02. 2025).

[30] A. Henriksen ja M. Mundt, "Sharing is caring: A practical guide to FAIR(ER) open data release", teoksessa *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Barcelona Spain: ACM, 25. elokuuta 2024, s. 6513–6522. DOI: 10.1145/3637528.3671468. url: https://dl.acm.org/doi/10.1145/3637528.3671468 (viitattu 27. 02. 2025).

[31] W. Maalej ja M. P. Robillard, ”Patterns of Knowledge in API Reference Documentation”, *IEEE Transactions on Software Engineering*, vol. 39, nro 9, s. 1264–1282, syyskuu 2013, ISSN: 1939-3520. DOI: 10.1109/TSE.2013.12. url: https://ieeexplore.ieee.org/abstract/document/6473801 (viitattu 12. 04. 2025).

[32] ”Google maps platform | google for developers”. (), url: https://developers.google.com/maps (viitattu 14. 03. 2025).

[33] ”AccuWeather APIs | home”. (), url: https://developer.accuweather.com/ (viitattu 14. 03. 2025).

[34] ”AccuWeather Enterprise API - AccuWeather Enterprise API Documentation”. (), url: https://apidev.accuweather.com/ (viitattu 03. 04. 2025).