Uppdatering av Strålsäkerhetscentralens öppna datatjänst

Atte Lähepelto

Lärdomsprov

Informationsteknik

2025

Lärdomsprov

Atte Lähepelto

Uppdatering av Strålsäkerhetscentralens öppna datatjänst

Yrkeshögskolan Arcada: Informationsteknik, 2025

Uppdragsgivare:

Strålsäkerhetscentralen

Sammandrag:

Detta arbete behandlar migreringen av en webbaserad applikation från Vue.js 2 till Vue.js 3, tillsammans med förbättringar i backend och införandet av en ny funktion för visualisering av radionukliders koncentration i luften. För att genomföra migreringen framgångsrikt och förbättra applikationens prestanda och säkerhet användes en stegvis uppdatering av kodbasen, där äldre Vue.js 2-komponenter ersattes med Vue.js 3-kompatibla motsvarigheter. Syftet var att förbättra applikationens prestanda, säkerhet och användarupplevelse genom teknologisk modernisering och systemarkitekturens optimering. Eftersom Finska Meteorologiska Institutets EDR API inte var fullt fungerande, användes det äldre WFS API för backend-implementeringen. EDR API erbjuder dock flera fördelar jämfört med WFS API, såsom effektivare datahämtning, bättre stöd för tidsserier och högre precision vid dataförfrågningar. För att möjliggöra bearbetning av data om luftens radionuklider och strålningsdosvärden utvecklades ett skript för databehandling, vilket även ledde till skapandet av en ny API-tjänst för hantering och hämtning av tidsseriedata för radionuklider. På frontend-sidan förbättrades användargränssnittet och applikationens prestanda genom uppdatering av Vue-komponenter samt implementeradet av en ny kartfunktion som ger användarna möjlighet att växla bakgrundskarta mellan OpenStreetMap och andra karttjänster. För att verifiera förbättringarna genomfördes prestandamätningar baserade på laddningstider, filstorlek och renderingsprestanda, medan säkerhetsgranskningar identifierade och åtgärdade tidigare sårbarheter. Resultaten från prestandatester visade på snabbare laddningstider och en mer responsiv användarupplevelse. Säkerhetsscanningar identifierade tidigare sårbarheter vilka har åtgärdats för att förbättra applikationens säkerhet. En potentiell framtida förbättring är att byta helt och hållet till EDR API när det blir tillgängligt, samt att optimera dataflödet direkt till frontend för att minska lagringskraven och reducera systemets komplexitet.

Nyckelord: Vue.js, migrering, EDR, WFS, säkerhet, Strålsäkerhetscentralen, radionuklider, STUK

Degree Thesis

Atte Lähepelto

Update of the Radiation and Nuclear Safety Authority's open data service

Arcada University of Applied Sciences: Information Technology, 2025

Commissioned by:

Radiation and Nuclear Safety Authority

Abstract:

This thesis addresses the migration of a web application from Vue.js 2 to Vue.js 3, backend improvements, and the implementation of a new feature for visualizing airborne particulate radioactivity (APR). To successfully carry out the migration and enhance the application's performance and security, a step-by-step codebase update was conducted, replacing older Vue.js 2 components with Vue.js 3-compatible counterparts. The aim was to enhance the application's performance, security, and user experience through technology modernization and system architecture optimization. Since the Finnish Meteorological Institute's EDR API was not fully functional, the older WFS API was used for the backend implementation. However, the EDR API offers several advantages over the WFS API, including more efficient data retrieval, better support for time series, and higher precision in data queries. Data processing for APR and radiation dose values was improved by developing a script for data processing, which also led to the creation of a new API service for retrieving time series data for radionuclides. On the frontend, the user interface and performance were improved by updating Vue components and implementing a new map functionality, allowing users to switch between OpenStreetMap and other map services. To verify these improvements, performance measurements were conducted based on loading times, file size, and rendering efficiency, while security audits identified and resolved previous vulnerabilities. Performance tests showed that the new version loads content faster and provides a more responsive user experience. Security scans identified previous vulnerabilities, which were resolved after updating the dependencies, improving the application's security. A potential future improvement is transitioning fully to the EDR API once it becomes available and optimizing data retrieval directly to the frontend, which would reduce storage requirements and simplify the system's complexity.

Keywords: Vue.js, migration, EDR, WFS, security, Radiation and Nuclear Safety Authority, STUK

Opinnäyte

Atte Lähepelto

Uppdatering av Strålsäkerhetscentralens öppna datatjänst

Yrkeshögskolan Arcada: Informaatiotekniikka, 2025

Toimeksiantaja:

Säteilyturvakeskus

Tiivistelmä:

Tässä työssä käsitellään web-sovelluksen siirtymistä Vue.js 2:sta Vue.js 3:een, backendin parannuksia ja uuden ilman radionuklidien visualisointitoiminnon toteutusta. Siirtymä toteutettiin vaiheittain päivittämällä koodikanta ja korvaamalla vanhat Vue.js 2 -komponentit Vue.js 3 -yhteensopivilla vastineilla, mikä varmisti yhteensopivuuden ja vakaan käyttöönoton. Tavoitteena oli parantaa sovelluksen suorituskykyä, turvallisuutta ja käyttäjäkokemusta teknologian modernisoinnin ja järjestelmäarkkitehtuurin optimoinnin avulla. Koska Suomen Ilmatieteen laitoksen EDR API ei ollut täysin toimiva, käytettiin vanhempaa WFS API:a backend-toteutuksessa. EDR API tarjoaa kuitenkin WFS API:iin verrattuna useita etuja, kuten tehokkaamman tiedonhaun, paremman tuen aikasarjoille ja tarkemmat kyselymahdollisuudet. Ilman radionuklidien ja säteilyannosarvojen käsittelyä varten kehitettiin skripti, mikä johti myös uuden API-palvelun luomiseen radionuklidien aikasarjojen hakemista varten. Frontendissä parannettiin käyttöliittymää ja sovelluksen suorituskykyä päivittämällä Vue-komponentteja ja luomalla uuden karttatoiminnon, joka mahdollistaa taustakartan vaihdon OpenStreetMapin ja muiden karttapalveluiden välillä. Suorituskyvyn ja turvallisuuden parantumisen varmistamiseksi suoritettiin testejä, joissa analysoitiin latausaikoja, tiedostokokoja ja renderöintitehokkuutta, sekä toteutettiin haavoittuvuustarkastuksia haavoittuvuuksien tunnistamiseksi ja korjaamiseksi. Suorituskykytestit osoittivat, että uusi versio lataa sisältöä nopeammin ja tarjoaa responsiivisemman käyttäjäkokemuksen. Haavoittuvuusskannaukset tunnistivat aiempia haavoittuvuuksia, jotka tuli korjattua siirtymällä uusimpiin versioihin riippuvuuksista, mikä paransi sovelluksen turvallisuutta. Mahdollinen tuleva parannus on kokonaan siirtyminen EDR API:iin sen tullessa saataville ja datan hakemisen optimointi suoraan frontendille, mikä vähentäisi tallennustarvetta ja yksinkertaistaisi järjestelmän rakennetta.

Avainsanat: Vue.js, migraatio, EDR, WFS, turvallisuus, suorituskyky, Säteilyturvakeskus, STUK

Förkortningar

API (Application Programming Interface) – Ett gränssnitt som gör det möjligt för olika program och system att kommunicera med varandra genom definierade regler och protokoll.

DOM (Document Object Model) – Ett programmeringsgränssnitt som representerar strukturen av en HTML- eller XML-sida i ett trädformat och gör det möjligt för skript att dynamiskt ändra innehåll och struktur.

EDR (Environmental Data Retrieval) – Ett nätverksprotokoll som används för att överföra filer mellan en klient och en server på ett nätverk. Används ofta för att ladda upp och ner filer från webbservrar.

FTP (File Transfer Protocol) – Ett programmeringsgränssnitt som representerar strukturen av en HTML- eller XML-sida i ett trädformat och gör det möjligt för skript att dynamiskt ändra innehåll och struktur.

JSON (JavaScript Object Notation) – Ett lättviktigt dataformat som används för att strukturera och utbyta information, särskilt i webbaserade tjänster och API:er.

ReDoS (Regular Expression Denial of Service) – En typ av cyberattack där en ineffektiv reguljäruttrycksberäkning leder till extremt hög CPU-belastning och kan göra en tjänst otillgänglig.

Vue.js – Ett progressivt JavaScript-ramverk som används för att bygga användargränssnitt, särskilt inom frontend-utveckling.

WFS (Web Feature Service) – En webbstandard för att hämta, redigera och hantera geografiska objekt i realtid över internet.

WMTS (Web Map Tile Service) – En standardiserad tjänst för att tillhandahålla kartbilder i form av förgenererade kakel (tiles), vilket förbättrar prestanda vid kartvisning.

WMS (Web Map Service) – En tjänst som genererar dynamiska kartbilder från geografiska databaser och gör det möjligt att visa och kombinera kartlager online.

Innehåll

1 Inledning

Webbutveckling är en föränderlig process som kräver regelbundna uppdateringar för att säkerställa att tjänster är både effektiva och säkra. Strålsäkerhetscentralens tjänst för öppna data är en viktig tjänst, som tillhandahåller information om luftens strålningsnivåer. Eftersom det handlar om allmänhetens hälsa och säkerhet är det kritiskt att tjänsten fungerar korrekt och uppdateras med den senaste tekniken.

Läsaren bör ha god kännedom om avancerade webbutvecklingsprinciper och moderna teknologier, inklusive JavaScript-ramverk som Vue.js, samt Python.

1.1 Bakgrund

Strålsäkerhetscentralen har en tjänst för öppna data, som visualiserar luftens strålningsnivåer[1]. Tjänsten är byggd med Vue.js 2, ett JavaScript-ramverk för att skapa användargränssnitt, och den använder ett föråldrat WFS API (Web Feature Service) för att hämta geospatial data via webben. Genom att uppdatera tjänsten från en äldre version av Vue.js till en nyare version skulle man säkerställa att allmänheten har tillgång till pålitlig och uppdaterad information. Stödet för Vue.js 2 upphörde den 31 december 2023, vilket innebär att inga nya funktioner, uppdateringar eller säkerhetsfixar längre tillhandahålls för denna version.

Att byta från WFS API till EDR API (Environmental Data Retrieval), en modern gränssnittslösning för åtkomst och hantering av miljödata, erbjuder också en mer flexibel och effektiv tjänst. EDR API har visat sig vara mer anpassningsbart än WFS API, då det stöder olika datatyper som raster och tidsserier, samt möjliggör mer exakt datainsamling utan att ladda ner stora dataset, vilket förbättrar prestandan och minskar nätverksbelastningen. Dessutom är EDR API lättare att integrera med andra moderna system, vilket ger en större grad av interoperabilitet.

Resultaten från detta projekt kan användas inom hela yrkesområdet för att förbättra förståelsen för tekniska migrationer och integrationsprocesser. Migreringen från Vue.js 2 till

[1] https://stukfi.github.io/opendata.html

Vue.js 3 ger insikt i de tekniska utmaningar och fördelar som är förknippade med webbutveckling, inklusive kompatibilitet och prestanda.

1.2 Syfte och mål

Syftet med mitt arbete är att uppdatera och förbättra Strålsäkerhetscentralens tjänst för öppna data genom att migrera från Vue.js version 2 till den senaste versionen, Vue.js 3. Denna uppgradering syftar till att öka prestanda, säkerhet och underhållbarhet. Samtidigt planeras att utöka tjänstens funktionalitet genom att lägga till möjligheten att byta bakgrundskarta samt att integrera visualisering av radionukliders koncentration i luften, vilket ger användarna viktig information i realtid. För att uppnå detta ska även API-gränssnittet för strålningsdosdata bytas från Meteorologiska institutets WFS API till EDR API, vilket förväntas förbättra dataleveransernas effektivitet och flexibilitet.

Målen i arbetet är följande:

1. En migration från Vue.js 2 till Vue.js 3, vilket innebär att kodbasen måste analyseras för att säkerställa kompatibilitet och funktionalitet
2. En integration av luftens radionukliders visualisering, vilket kräver ytterligare analys och testning för att säkerställa datakvalitet och användarupplevelse
3. Integration av möjligheten att byta bakgrundskarta
4. API-gränssnittet för strålningsdosdata bytt till EDR API, vilket ska bidra till snabbare och mer tillförlitliga datauppdateringar

Genom att uppfylla dessa mål förväntas projektet resultera i en moderniserad tjänst som är mer användarvänlig och som erbjuder ny funktionalitet. Detta bidrar i sin tur till en säkrare och mer pålitlig tjänst för allmänheten, samtidigt som det lägger en grund för framtida utveckling och expansion.

1.3 Forskningsfrågor

Jag vill undersöka vilka problem eventuellt kan uppstå i Vue.js migrationen, hur versionerna skiljer sig, hur WFS och EDR API skiljer sig, och med vilka metoder jag kan samla in data för att bevisa att problemen är lösta.

I detta arbete syftar jag till att svara på följande forskningsfrågor:

1. Hur kan migrationen från Vue.js 2 till Vue.js 3 genomföras framgångsrikt för att förbättra applikationens prestanda och säkerhet?
2. Vilka fördelar förekommer med EDR API jämfört med WFS API?
3. Hur kan jag samla in data för att mäta prestanda- och säkerhetsförbättringar?

2 Teoretisk referensram

2.1 Vue.js Ramverket

Vue.js is är ett JavaScript ramverk för att bygga användargränssnitt. Vue.js bygger på standard HTML, CSS, och JavaScript och tillhandahåller komponentbaserade programmeringsmodeller som hjälper till att utveckla effektivt användargränssnitt av olika komplexitet.

Vue.js möjliggör till exempel självdeklarativ renderering och reaktivitet. Vue utökar standard-HTML med en mallsyntax som låter HTML deklarativt beskrivas baserat på JavaScript tillstånd. Vue.js spårar också automatiskt ändringar i JavaScript tillstånd och uppdaterar effektivt DOM när ändringar sker (Vue.js, u.å.-c).

2.1.1 Historik

Vue.js är skapat av Evan You år 2014 som ett personligt sidoprojekt (Vue.js, u.å.-b). Han bestämde sig att skapa ett nytt ramverk efter att han jobbat på Google och använt AngularJS, ett JavaScript-baserat webbramverk för att bygga ensidiga webbapplikationer. Evan ville skapa något som hade likadan funktion som han tyckte om i Angular, men något lättvikt (Wikipedia, u.å.). Idag är Vue.js aktivt underhållet av ett team med både anställda och frivilliga medlemmar runt om hela världen, lett av Evan You.

Vue.js har tre olika versioner (Versionlog, u.å.):

Vue.js 1.0 var publicerat i december 2015, versionen är den så kallade ”The Evangelion Release”. Det här var början för ett helt nytt ramverk. Strax efter version 1 kom förbättrade versionen Vue.js 2 (Vue.js, 2015).

Vue.js 2.0, även känd som "Ghost in the Shell", släpptes i september 2016. Denna version innebar en betydande uppgradering från den tidigare versionen och introducerade flera nya funktioner och förbättringar som gjorde ramverket ännu mer kraftfullt och flexibelt. Vue.js 2 har nått End of Life 31 december 2023. Versionen får inte längre kritiska buggfixar eller säkerhetsuppdateringar (Vue.js, 2016).

Vue.js 3.0, kallad "One Piece" lanserades i september 2020 och representerar den senaste och mest avancerade versionen av ramverket. Denna version introducerade flera betydande förbättringar och nya funktioner så som förbättrad prestanda, skalabilitet, och TypeScript / IDE stöd för att möta moderna utvecklingsbehov. Vue.js 3 innehåller ”breaking changes”, vilket gör att version 2 och 3 är inte kompatibla med varandra (Vue.js, u.å.-a).

Vue.js compat, kallad ”The migration build” publicerades vid Vue.js 3 utkomst. Versionen är skapad för att hjälpa till i migrationsprocessen från Vue.js 2 till Vue.js 3. Versionen kör på Vue.js 3, men är också kompatibel i viss grad med Vue.js 2. Compat versionen ger varningar i konsolen av utfasade ändringar mellan versionerna, vilket hjälper till att upptäcka problemområden i koden (Vue.js, u.å.-d).

2.1.2 Vue.js 2 vs Vue.js 3

Vue.js 3 introducerar tydliga förbättringar och ändringar jämfört till Vue.js 2.

1. Kompositions-API:
- I Vue.js 2 används Options API för att definiera data, metoder och beräknade egenskaper inom en komponent. Vue.js 3 introducerar Composition API, som gör det möjligt att organisera logiken i separata funktioner. Detta gör koden mer modulär och lätt att återanvända mellan olika komponenter.
2. Reaktivitetssystem:

- Vue.js 2 bygger sitt reaktivitetssystem på `Object.defineProperty`, vilket kan leda till vissa begränsningar och sämre prestanda. Vue.js 3 använder ett Proxy-baserat reaktivitetssystem, vilket ger bättre prestanda och färre restriktioner.

3. Teleport-funktion:

- Vue.js 2 saknar en inbyggd funktion för att flytta ett element till en annan del av DOM. Denna funktion introduceras i Vue.js 3, vilket gör det möjligt att flytta element smidigt.

4. Fragmentstöd:

- Vue.js 2 kräver att varje komponentmall har ett enda root-element. Vue.js 3 tillåter att flera root-element används i en och samma mall, vilket minskar behovet av onödiga omslutande element.

5. Optimerad prestanda:

- Vue.js 3 har genomgått många optimeringar för att förbättra prestandan, inklusive en snabbare virtuell DOM, effektivare komponentuppdateringar och förbättrad hantering av statiska träd, vilket minskar renderingsbelastningen.

6. Förbättrat TypeScript-stöd:

- Stödet för TypeScript i Vue.js 2 är begränsat. Vue.js 3 erbjuder bättre integration och typkontroll, vilket gör det enklare för utvecklare att arbeta med TypeScript.

7. Registrering av anpassade direktiv:

- I Vue.js 2 registreras anpassade direktiv globalt med `Vue.directive`. I Vue.js 3 används `app.directive` för att registrera direktiv, vilket gör dem mer modulära och begränsade till specifika applikationsinstanser.

8. Scoped Slots förbättringar:

- Vue.js 2 har vissa begränsningar när det gäller scoped slots, vilket gör det svårt att överföra flera värden. Vue.js 3 introducerar `v-slots`, vilket ger större flexibilitet och kraft till scoped slots.

9. Tree shaking:

- Vue.js 3 möjliggör för byggverktyg att ta bort oanvänd kod under byggprocessen, vilket resulterar i bättre prestanda och mindre filstorlekar.

10. API-förändringar:

- Vue.js 3 har infört förändringar i vissa API. Till exempel har nextTick-funktionen ersatts med en Promise-baserad version, vilket förenklar hanteringen av asynkrona operationer.

(Tinker, 2023)

2.1.3 Utmaningar och risker vid migration

Migrationen från version 2 till version 3 kommer med många utmaningar och risker. Mycket har förändrats, och om man tänker sig att stora delar av koden måste byggas om, så kan det förekomma oväntade buggar. Även de minsta ändringarna som man är tvungen att göra, kan påverka andra delar av hela webbsidans kod negativt (Monterail, 2023, 19:39). En hel del av ändringarna är sådana som kanske kräver att man måste hitta på ett helt nytt sätt att implementera den gamla funktionaliteten.

Alla ändringar som har uppkommit i Vue.js versionsuppdateringen är inte kompatibla med den gamla versionen. Sådana ändringar är som följande:

1. Global API:
   - Det globala Vue API har ändrats till att använda en applikationsinstans.
   - Globala och interna API har omstrukturerats för att vara tree-shakable.
2. Template Directives:
   - `v-model`-användning på komponenter har omarbetats och ersätter `v-bind.sync`.
   - `key`-användning på `<template v-for>` har ändrats.
   - v-if och v-for har förändrad företrädesordning när de används på samma element.
   - `v-bind="object"` är nu ordningskänslig.
   - `v-on:event.native`-modifieraren har tagits bort.
3. Components:
   - Renderfunktions-API har ändrats.
   - `$scopedSlots`-egenskapen har tagits bort och alla slots exponeras via `$slots` som funktioner.
   - `$listeners` har tagits bort och slås samman med `$attrs`.

- `$attrs` inkluderar nu klass- och stilattribut.

4. Custom Elements:
   - Kontroller av anpassade element utförs nu under template-kompilering.
   - Specialattributet is får endast användas på den reserverade `<component>`-taggen.
5. Andra små ändringar:
   - Livscykelalternativet `destroyed` har bytt namn till `unmounted`.
   - Livscykelalternativet `beforeDestroy` har bytt namn till `beforeUnmount`.
   - Props standardfabrikfunktion har inte längre tillgång till `this`-kontexten.
   - Anpassade direktiv-API har ändrats för att anpassas till komponentens livscykel och `binding.expression` har tagits bort.
   - data-alternativet ska alltid deklareras som en funktion.
   - data-alternativet från mixins sammanfogas nu ytligt.
   - Attributkoercionsstrategin har ändrats.
   - Några övergångsklasser har bytt namn.
   - `<TransitionGroup>` renderar nu inget omslagselement som standard.
   - När man övervakar en array, utlöses callback endast när arrayen ersätts. För att utlösa vid mutation måste deep-alternativet anges.
   - `<template>`-taggar utan speciella direktiv (`v-if/else-if/else, v-for, eller v-slot`) behandlas nu som vanliga element och resulterar i ett inbyggt `<template>`-element i stället för att rendera dess inre innehåll.
   - En monterad applikation ersätter inte längre elementet som den är monterad på.
   - Livscykelhändelseprefix ändrades till `vnode-`.
6. Removed APIs
   - keyCode-stöd som v-on-modifierare.
   - Instansmetoderna `$on, $off och $once`.
   - Filter.
   - Attribut för inline-templates.
   - Instansegenskapen $children.
   - propsData-alternativet.

- Instansmetoden `$destroy`. Användare bör inte längre hantera livscykeln för individuella Vue-komponenter manuellt.
- Globala funktioner set och delete, samt instansmetoderna `$set och $delete`. De behövs inte längre med proxybaserad ändringsdetektion.

(Vue.js, u.å.-a)

2.2 WFS och EDR API

2.2.1 WFS

WFS API representerar en modern metod för att hantera geografisk information över internet, jämfört med äldre metoder som att dela data via FTP, ett protokoll för att överföra filer. Med WFS får man direkt och detaljerad åtkomst till geografiska data på en nivå som fokuserar på enskilda funktioner och deras egenskaper (OGC, u.å.-b).

2.2.2 EDR API

EDR API är en specifikation som erbjuder enkla och effektiva gränssnitt för att hämta miljödata. Varje resurs som hanteras av ett EDR API följer ett fastställt frågemönster. Specifikationen för EDR API beskriver de resurser som kan nås, fastställer krav och definierar hur API för miljödata ska fungera (OGC. u.å.-a).

2.2.3 WFS vs EDR API

WFS API och Environmental Data Retrieval (EDR) API är båda standarder för att få tillgång till och hantera geografisk och miljödata, men de har olika fokus och funktioner. Här är en översikt över de viktigaste skillnaderna och likheterna mellan de två API:na:

Tabell 1. Skillnaderna mellan WFS och EDR API

Egenskap	WFS API	EDR API
Syfte	Tillhandahålla detaljerad åtkomst och manipulation av geospatiala data.	Erbjuda ett enkelt gränssnitt för att hämta miljödata.
Standard	Open Geospatial Consortium (OGC)	Open Geospatial Consortium (OGC)
Dataformat	GML (Geography Markup Language), XML-baserat	JSON, ofta enklare och lättare att integrera
Frågeoperationer	GetFeature, GetPropertyValue	Hämtar datan baserat på enkla urvalskriterier
Prestanda	Kan vara tungt för stora datamängder	Optimerat för snabb åtkomst
Flexibilitet	Erbjuder omfattande funktioner och detaljerad manipulation av data	Fokuserad på enkelhet och effektivitet
Användbarhet	Bra för komplexa geospatiala applikationer men kan vara mer komplex att implementera	Enklare att integrera i moderna applikationer

3 Metoder

I projektet användes en kombination av litteraturstudier, praktisk implementering och mätning av prestanda för att besvara forskningsfrågorna och uppnå projektets mål. Metoderna delades in i följande steg:

3.1 Litteraturstudie

För att få en djup förståelse för de tekniska förändringarna mellan Vue.js 2 och Vue.js 3 samt skillnaderna mellan WFS API och EDR API genomfördes en litteraturstudie. Denna innefattade den officiella Vue.js-dokumentationen (Vue.js, 2015) (Vue.js, 2016) (Vue.js, u.å.-a) (Vue.js, u.å.-b) (Vue.js, u.å.-c) (Vue.js, u.å.-d) där förändringar och migrationsstrategier beskrivs i detalj. För ytterligare perspektiv granskades även en bloggartikel från SHIFT ASIA (Tinker, 2023) och en video från Monterail (Monterail, 2023), ett företag som genomfört migrationer. Vidare studerades OGC:s officiella dokumentation (OGC, u.å.-a) (OGC, u.å.-b) för att analysera skillnaderna mellan WFS API och EDR API.

Slutligen gav Wikipedia (Wikipedia, u.å.) och Versionlog (Versionlog, u.å.) en historisk översikt över Vue.js-utvecklingen.

3.2 Praktisk implementering

Den praktiska delen av arbetet genomfördes i flera steg. Först migrerades applikationen från Vue.js 2 till Vue.js 3. Arbetet inleddes med en grundlig analys av den befintliga koden i Vue.js 2 för att identifiera potentiella problemområden. Vue.js's officiella migreringsguide och verktyg användes för att säkerställa en smidig övergång, och nödvändiga kodändringar samt uppdateringar av beroende bibliotek och moduler genomfördes för att uppnå kompatibilitet med Vue.js 3. Även erfarenheter från utvecklargemenskapen och diskussioner i Vue.js-forum beaktades vid problemlösning och optimering.

Därefter byttes WFS API till EDR API för strålningsdosdata. Detta innebar att EDR API integrerades och testades i backenden för att säkerställa att systemet effektivt kunde hämta och hantera data.

Slutligen implementerades visualiseringen av radionukliders koncentration i luften, och funktionen för kartbytande lades till. Detta inkluderade planering av visualiseringens utformning, modifiering av tjänstens datahämtning för att inkludera information om luftens radionuklider, implementering av kartbytarfunktionen samt skapandet av nya komponenter i Vue.js 3-projektet.

3.3 Mätning och utvärdering

För att bekräfta att projektets mål hade uppnåtts samlades data in och analyserades med hjälp av olika metoder. Nedanstående verktyg och mätmetoder valdes efter att alternativa tillvägagångssätt hade analyserats och bedömts utifrån deras relevans för applikationens prestanda och säkerhet.

Vid val av prestandamätning övervägdes även Lighthouse[2] och Google PageSpeed Insights[3], men WebPageTest[4] valdes då det ger mer detaljerad data och möjligheten att simulera olika nätverkshastigheter. För säkerhetsanalys beaktades även SonarQube[5] och Veracode[6], men Snyk[7] valdes för dess omfattande databas över kända sårbarheter.

Med WebPageTest fokuserades på att mäta:

- Initial laddningstid: Hur lång tid det tar för sidan att börja visa innehåll.
- Fullständig laddningstid: Den totala tiden det tar för alla resurser att laddas och för sidan att bli helt klar.
- Tiden till interaktivitet: Den tid som går innan sidan är fullt interaktiv och användaren kan börja interagera med den.
- Resursbelastning: Den totala mängden data som laddas.
- First Contentful Paint: Den tidpunkt då den första biten av innehåll, så som text eller bild, visas för användaren.
- Largest Contentful Paint: Den tidpunkt då det största synliga innehållet laddas.
- Speed Index: En mätning av hur snabbt innehållet på sidan blir synligt.

Av dessa är *Tiden till interaktivitet* och *Largest Contentful Paint* de mest avgörande för användarupplevelsen, då de direkt påverkar hur snabbt applikationen upplevs som responsiv.

3.4 Säkerhetsanalys med Snyk

För att kunna bedöma säkerheten används Snyk, som identifierar och analyserar sårbarheter i beroenden. Jämförelser görs mellan Vue.js 2 och Vue.js 3 för att analysera förbättringar i säkerheten.

[2] https://github.com/GoogleChrome/lighthouse
[3] https://pagespeed.web.dev/
[4] https://www.webpagetest.org/
[5] https://www.sonarsource.com/products/sonarqube/
[6] https://www.veracode.com/
[7] https://snyk.io/

En mer omfattande analys kunde ha genomförts genom att kombinera Snyk med verktyg som Semgrep eller SonarQube, men Snyk valdes för att få en snabb och direkt identifiering av de mest kritiska sårbarheterna i Vue.js-miljön. Vid framtida analyser kan en kombination av flera verktyg ge en djupare säkerhetsbedömning.

4 Utgångsläget

Utgångsläget är det läge i vilket projektet befinner sig innan några förändringar har genomförts. Projektet är uppdelat i två delar: frontend och backend. Backend-delen hanterar datainsamling och lagring, medan frontend-delen ansvarar för visualisering av data från backend.

4.1 Frontend

Frontend-delen av applikationen är byggd med Vue.js 2, och testerna är skrivna med Jest.

Den nuvarande strukturen och de nuvarande teknologierna inkluderar:

- Komponentstruktur: Applikationen är uppbyggd med flera Vue-komponenter vilka är ansvariga för olika delar av användargränssnittet.
- dose-rate.js hämtar data från filstrukturen.
- State Management: Vuex används för att hantera applikationens tillstånd och dataflöden mellan komponenterna.
- Styling: CSS och SCSS används för styling av komponenterna.
- Byggverktyg: Webpack används som byggverktyg för att paketera och optimera frontend-koden. Webpack hanteras av Vue CLI.
- Testning: Jest används för att skriva och köra enhetstester för att säkerställa att komponenterna fungerar korrekt.
- Beroenden: Applikationen har de följande beroenden:
- Filstruktur: Komponenterna är indelade i olika mappar enligt deras placering

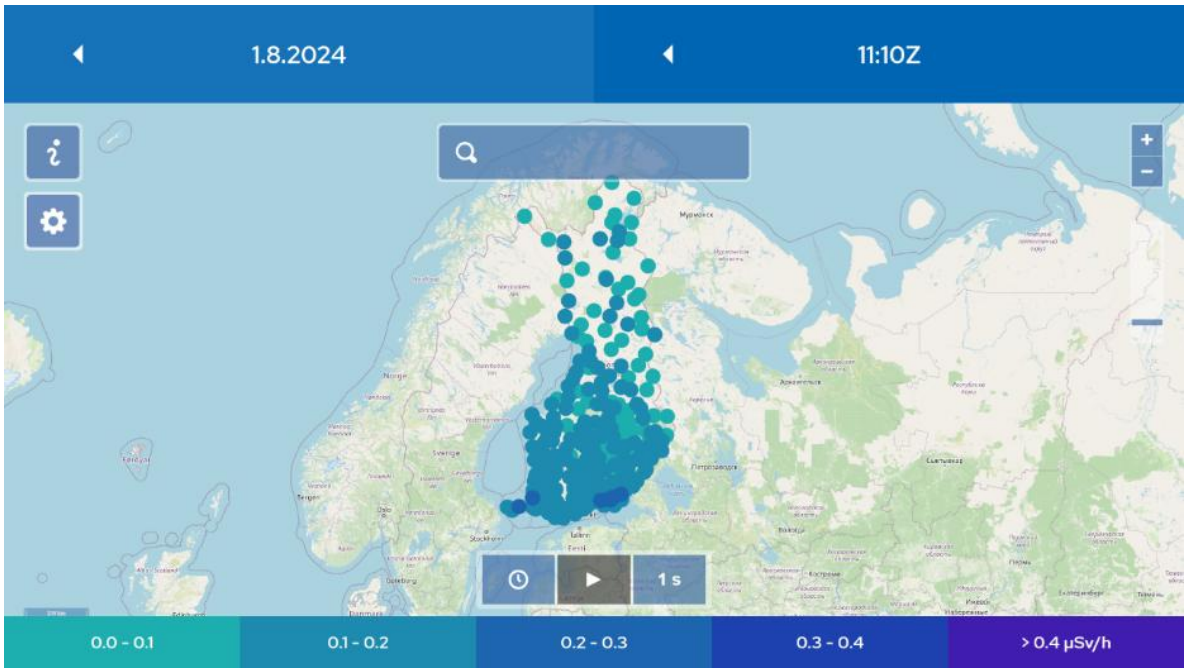
Tabell 2. Applikationens beroenden

Beroende	Version
caniuse-lite	^1.0.30001442
loadash	^4.17.21
ol	^5.3.3
plotly.js	^2.17.1
vue	^2.7.14
vue-click-away	^2.2.2
vue-i18n	^8.22.2
vue-progressbar	^0.7.5
vue-resource	^1.5.3
vuejs-datepicker	^1.6.2
vuex	^3.6.0

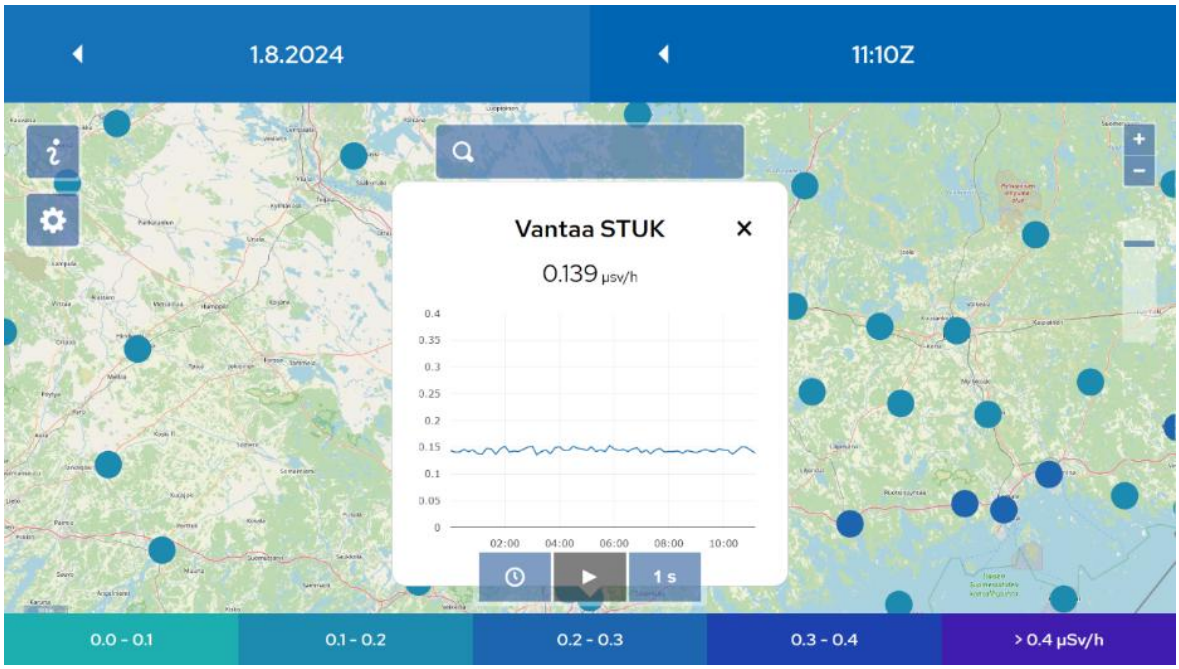
Applikationen har följande funktioner:

- Infoknapp: Öppnar en ruta med info om applikationen
- Inställningsknapp: Öppnar en ruta med inställningar, där man kan:
  - Välja språket
  - Välja datumformatet
  - Välja tidsformatet
- Datum och tidsväljare: Öppnar en kalender eller en lista med klockslag för data
- Sökfält: För att söka en mätpunkt på kartan
- Kartkontroll: Zoom verktyg
- Mediaknappar: Möjliggör punkternas animering på kartan per datum eller klockslag
- Map legend bar: Möjliggör att välja vilka värden visas på kartan
- Popup: Visar ett diagram över strålningsdosen

Nuvarande utmaningar innehåller till exempel föråldrade beroenden. Vue.js 2 och vissa tillhörande bibliotek har inte längre stöd, vilket innebär potentiella säkerhetsrisker och brister.



Figur 1. Applikationens framsida



Figur 2. Popup med diagram över strålningsdosen i Vanda

4.2 Backend

Backend-delen av projektet ansvarar för att ladda ner och bearbeta data för frontend-delens visualisering.

Anledningen till att data laddas ner till filsystemet är för att möjliggöra visualisering av andra datakällor, inte bara den information som tillhandahålls av Finska Meteorologiska Institutet. Detta gör applikationen mer flexibel och användbar för ett bredare spektrum av data.

Nuvarande funktioner och teknologier är som följande:

- Som programmeringsspråk används Python.
- En `dose-rate.js` fil i src katalogen, som är en API tjänst för att hämta data från filsystemet.
- Strålningsdosdatan laddas ner från externa källor (Finska Meteorologiska Institutets öppna datas WFS API).
- Laddade data omvandlas från XML format till GeoJSON format, och sparas i ”datasets” mappen.
- Tidslinjefiler genereras enligt tidpunkten av data och sparas i egna mappar enligt mätpunktens id.
- En metadatafil skapas eller uppdateras med alla tillgängliga tidpunkter för varje datum.

Filstrukturen och filernas uppgifter:

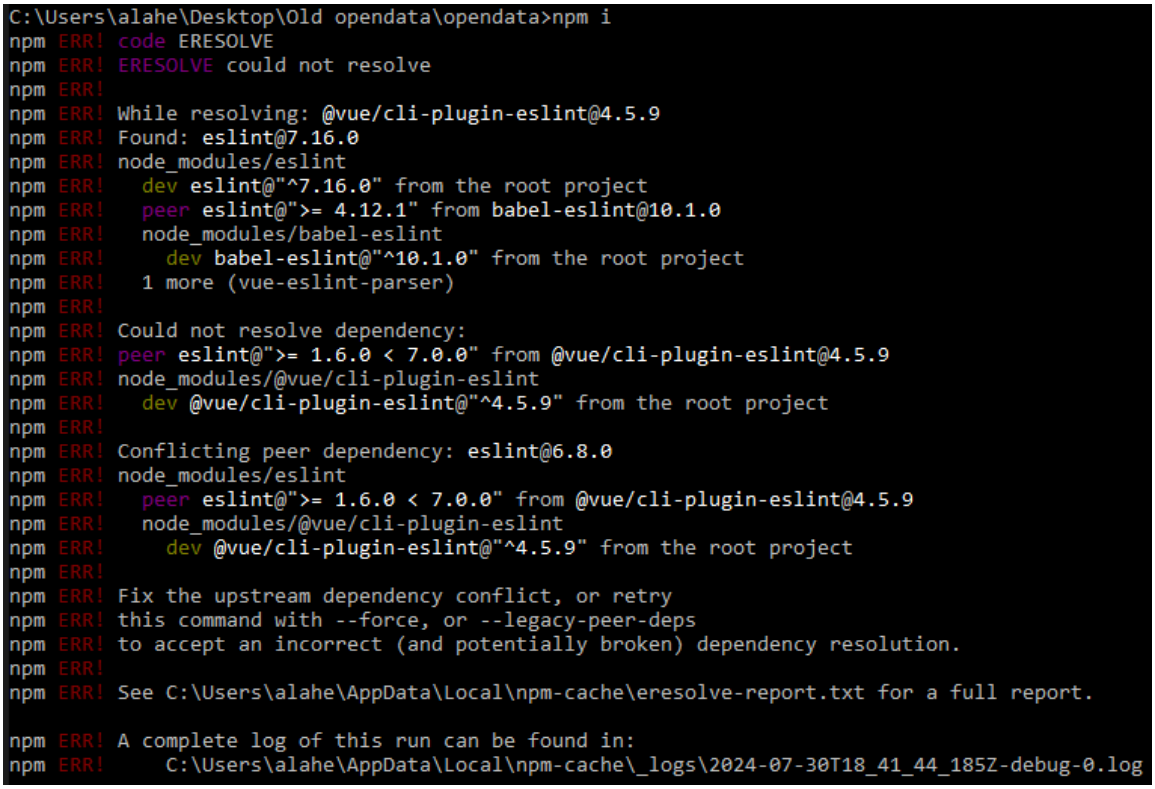
```
scripts/
│
├── dose_rates.py: Ansvarar bl.a. om att omvandla XML-data till GeoJSON-format
├── fmi_utils.py: Gör API-förfrågningar
├── get_data.py: Ansvarar för att köra hela programmet
├── metadata.py: Genererar metadatafiler
├── settings.example.json: Exempel på en inställningsfil
├── settings.py: Inställningsfil
└── time_series.py: Genererar tidsseriefiler

src/api
│
├── dose-rate.js: En API tjänst för att hämta data från filsystemet
```

5 Migrationsprocessen

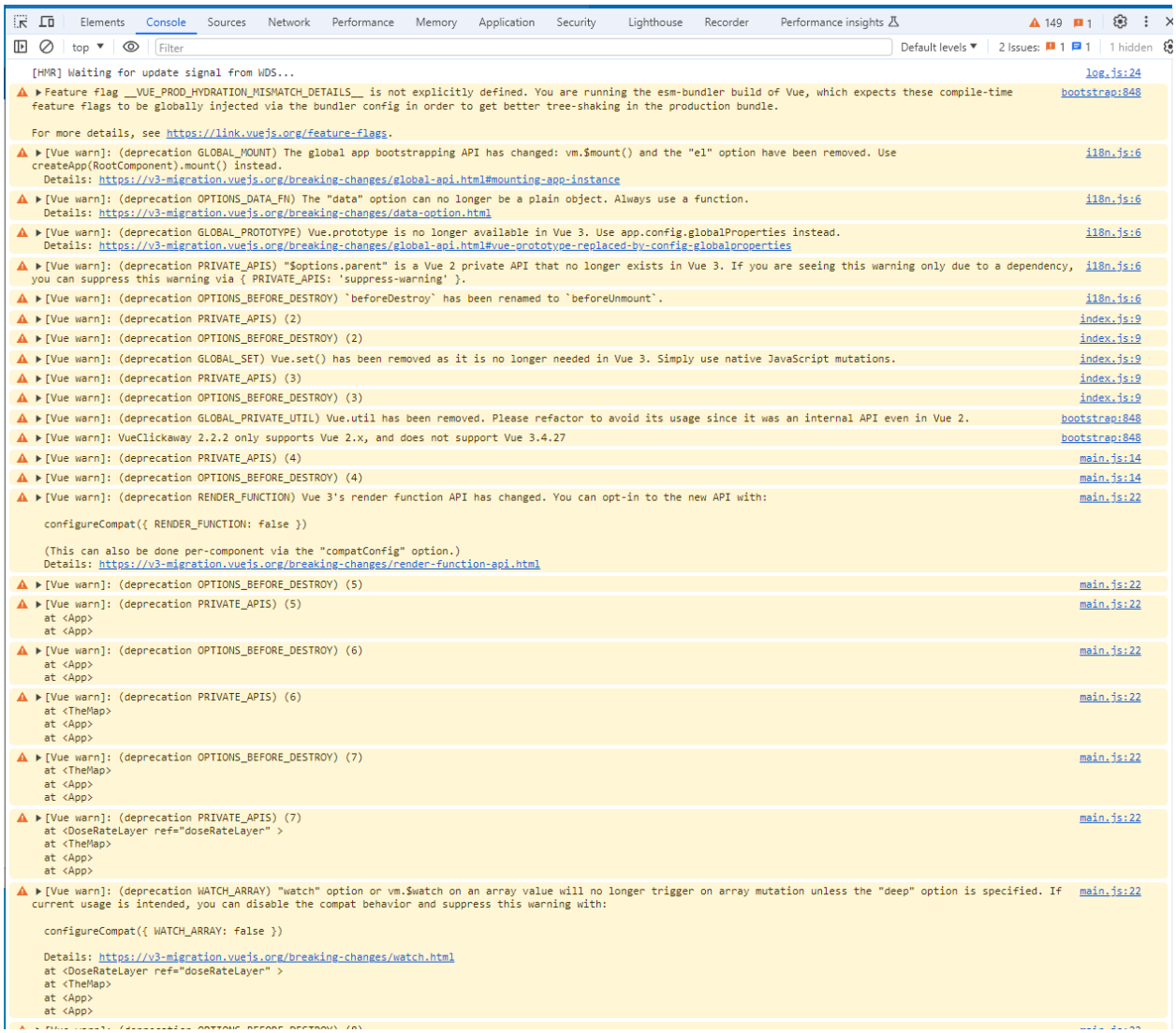
Projektets migrationsprocess började med att klona ner repositoriet från GitHub och installera det på datorn. Projektet var senast uppdaterat 27 oktober 2021. Eftersom repositoriet innehöll mycket med föråldrade beroenden kunde man inte längre installera det utan att använda `npm i -force`. Parametern `-force` tvingar paketinstalleraren npm att installera beroendena även om versionerna inte är kompatibla med varandra. Den troliga

orsaken till att det fanns inkompatibla beroenden var GitHubs Dependabot, en bot som automatiskt uppdaterar paket.



Figur 3. Skärmbild av installationen

Vue.js migrationsversionen (@vue/compat) användes för att få reda på alla problemområden i koden. Då man första gången körde projektet med `npm run serve`, vilket startar en lokal utvecklingsmiljö, hade konsolen närmare hundra varningar för inkompatibel kod som behöver uppdateras.



Figur 4. Skärmbild av webbläsarens konsol

5.1 Användning av Vue.js 3-förbättringar

För att förbättra både prestanda och säkerhet togs i bruk följande Vue.js 3 förbättringar:

1. **Övergång till Composition API**
    - Tidigare använde projektet Options API, men för att förbättra kodstrukturen och återanvändbarheten konverterades komponenter till Composition API där det var lämpligt.
    - Exempelvis ersattes `data` och `methods` i flera komponenter med `ref` och `computed` från vue-paketet, vilket gav mer modulär och testbar kod.

2. **Optimerad reaktivitet med Proxy-baserat system**
    - Vue 2:s reaktiva system baserat på `Object.defineProperty` byttes ut mot Vue 3:s Proxy-baserade reaktivitet, vilket resulterade i snabbare förändringsdetektering och färre prestandaflaskhalsar.
    - Detta var särskilt viktigt i komponenter som hanterar stora mängder data i realtid.

3. **Tree Shaking och minskad filstorlek**
    - Vue.js 3:s stöd för Tree Shaking möjliggjorde borttagning av oanvänd kod under byggprocessen.
    - Genom att använda `import { createApp } from 'vue'` istället för att importera hela Vue-biblioteket minskades paketstorleken, vilket förbättrade laddningstiderna.

4. **Förbättrad hantering av slots och komponentstruktur**
    - Vissa komponenter som tidigare använde komplexa slot-lösningar uppdaterades för att dra nytta av v-slots, vilket förenklade koden och minskade onödiga wrapper-element.

5. **Typförbättringar med bättre TypeScript-stöd**
    - Eftersom Vue.js 3 har förbättrat TypeScript-stöd, definierades gränssnitt och typer tydligare i projektet, vilket ökade kodens robusthet och minskade risken för typrelaterade fel.
    - Dessa uppdateringar bidrog till en säkrare och mer optimerad kodbas samtidigt som applikationens prestanda förbättrades.

5.2 Migrationsprocessens steg

1. **i18n – Internationalisering:**

   En av de första större uppdateringarna jag gjorde var övergången till en ny version av i18n, vilket är ett bibliotek för hantering av flerspråkighet i applikationen. Jag flyttade alla översättningar till en separat i18n.js-fil och gjorde små ändringar i

syntaxen för att matcha den nya versionen. Ett problem som uppstod var att språket inte byttes efter att användaren klickade på språkknapparna; ändringen kom bara i kraft efter en siduppdatering. Detta fixades genom att byta ut den utfasade koden `i18n.locale = this.locale` mot `i18n.global.locale.value = this.locale`, vilket är den nya rekommenderade metoden i Vue.js 3.

2. **Vuex – Tillståndshantering:**
   Vuex, som används för tillståndshantering, krävde en mindre uppdatering i sättet man skapar en ny *store*, ett ställe för att spara tillstånd. Inga andra modifieringar var nödvändiga, vilket gjorde denna del av migreringen relativt enkel.

3. **Vue – Nytt sätt att skapa applikationer:**
   En litet syntaxförändring i Vue.js 3 är hur man skapar en ny Vue-applikation. Den gamla metoden `new Vue` har blivit utfasad och har ersatts av `createApp`.

4. **Syntaxförändringar:**
   Ytterligare syntaxändringar inkluderade att ändra `spellcheck="false"` till `:spellcheck="false"` i sökfältet, samt att uppdatera från `vue-clickaway` till `vue3-click-away` för att hantera klick utanför ett element. Dessa uppdateringar krävde små modifieringar i koden, men var nödvändiga för att säkerställa kompatibilitet med Vue.js 3.

5. **Eventbus – Ersättning av $on, $off och $emit:**
   Med övergången till Vue.js 3 har flera funktioner för händelsehantering avvecklats, inklusive `vm.$on/$once/$off()`. För att hantera detta ersatte jag dessa med TinyEmitter och skapade en `eventBus.js`-fil i `utils`-mappen. Jag importerade `eventBus` till varje komponent som använde händelser och deklarerade dessutom alla emitters i komponenterna enligt Vue.js 3 rekommendationer.

6. **Datepicker:**
   Den kalender som användes i projektet fungerade inte med Vue.js 3. Jag var därför tvungen att söka efter en helt ny kalenderkomponent som var kompatibel med Vue.js 3. Efter att jag hade testat många olika alternativ, bestämde jag mig för att

använda `Vuepic vue-datepicker`, för att dess funktioner var väldigt lika de gamla.

7. **MediaController.js – Reaktivitet:**
   En annan betydande utmaning jag stötte på var att göra konstruktorerna i MediaController.js reaktiva för att de skulle fungera korrekt med Vue.js 3. Detta var nödvändigt för att säkerställa att värdena i MediaController.js uppdaterades korrekt i komponenterna.

8. **Axios:**
   Då vue-resource inte längre underhålls, ersatte jag det med Axios för att hantera HTTP-förfrågorna för data.

9. **Tester:**
   Testerna var ursprungligen skrivna med Jest, men eftersom jag bytte från Vue CLI till Vite kunde jag i stället använda Vites egna testverktyg, Vitest, för att hantera testningen. Detta krävde en del ändringar i de befintliga testerna för att de skulle fungera korrekt med den nya testmiljön.

10. **Övriga justeringar:**
    En del övriga justeringar genomfördes för att anpassa applikationen till Vue.js 3 versionen. Jag uppdaterade bland annat OpenLayers från version 5 till 9, vilket krävde ändringar i importerna och funktionerna. Dessutom bytte jag från Vue CLI till Vite, då Vue CLI nu befinner sig i underhållsläge. Detta innebar att jag behövde uppdatera alla importer i komponenterna, och hantera problem med SVG-ikoner som inte längre visades korrekt i produktionsbygget på grund av filstrukturens ändringar.

6 Implementering av Luftens Radionukliders Visualisering

Beställaren efterfrågade en ny funktion för visualisering av radionukliders koncentration i luften. Fram till nu har tjänsten enbart visat strålningsdoser, men nu ville man lägga till visualiseringen av luftens radionukliders koncentration.

6.1 Kraven

Kraven för visualisering av luftens radionuklidsnivåer är följande:

- Ett eget kartlager som innehåller samtliga provtagningsorter: Helsinki, Vantaa, Kotka, Imatra, Kuopio, Kajaani, Rovaniemi, Sodankylä och Ivalo.
- När man klickar på en provtagningsort ska data för de senaste 30 dagarna hämtas och visas.
- Resultaten ska presenteras i ett popup-fönster i form av nuklid-specifika tabeller.
- Tabellens rubriker ska innehålla nuklidens namn, till exempel Be-7 eller Cs-137.
- Kolumnerna i tabellen ska inkludera: Insamlingsperiod, genomsnittlig nuklidkoncentration (μBq/m³), och mätosäkerhet (%).
- Rubriker för artificiella nukliders tabeller ska vara i fetstil.
- Popup-fönstret ska ha en datepicker-funktion som möjliggör val av annan tidsperiod.

Implementeringen av visualiseringen av luftens radionuklidsnivåer kräver ändringar i både frontend- och backend-delarna av applikationen. På frontendsidan behövs nya komponenter, inklusive en knapp för att växla mellan olika lägen och ett nytt kartlager för att visa mätstationer som övervakar luftens radionukliders koncentration. På backendsidan krävs det att hämta data från en ny API-adress, konvertera XML till GeoJSON på samma sätt som för strålningsdoserna, samt att implementera en smart struktur för att spara data i filsystemet.

6.2 Implementering

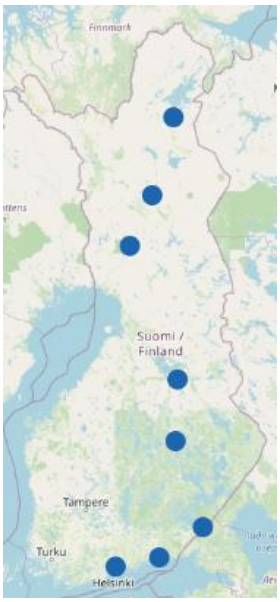
6.2.1 Frontend

Nya komponenter skapades:

1. **ButtonChangeMode.vue:**
   - En knapp för att växla mellan lägena för strålningsdos och luftens radionuklider.
   - Ändrar "mode" för hela applikationen till antingen "`dose_rates`" eller "`air_radionuclides`".

Figur 5. Knapp för att byta läge

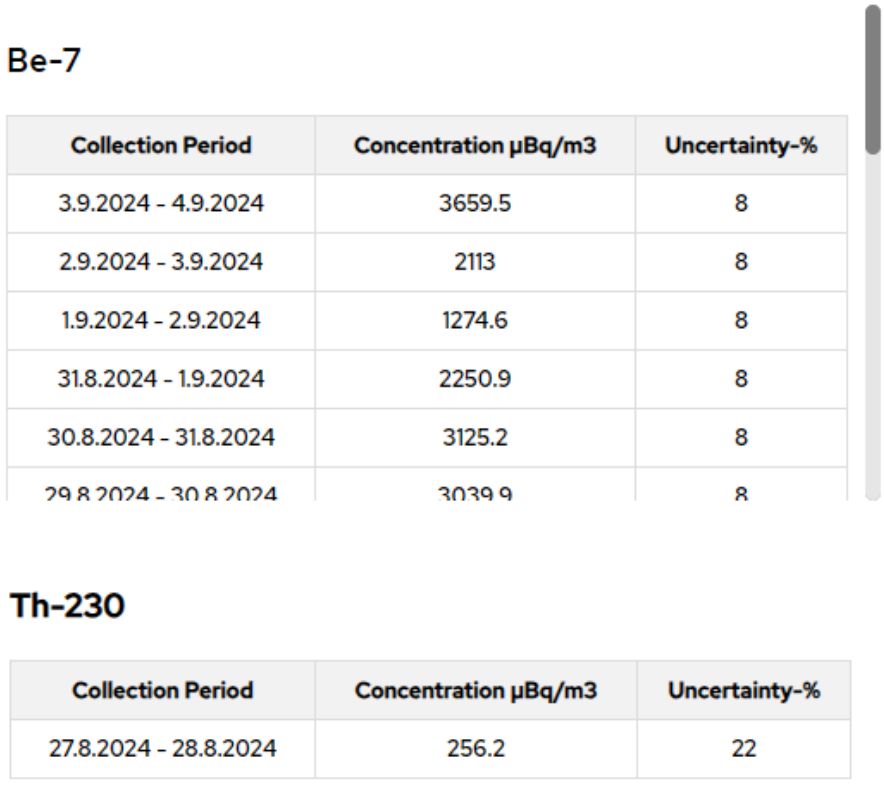
2. RadionuclideLayer.vue:
   - Ett kartlager för att visualisera mätstationerna på kartan.
   - Hämtar alla tillgängliga platser och deras koordinater från "`data/air_radionuclides/time_series/sites.json`".



Figur 6. Kartlagret med mätstationerna

RadionuclideTables.vue:

- Skapar tabellerna som visas i popup-fönstret.
- Anropar air-radionuclides.js för att hämta `available_radionuclides.json` för den valda mätstationen.
- Anropar air-radionuclides.js för att hämta tidsserierna för tillgängliga radionuklider baserat på det valda datumet.
- Sorterar tabellerna i följande ordning: "Be-7", "Cs-137", "Pb-210", "Na-22", övriga.
- Sorterar resultaten efter det senaste datumet.
- Markerar rubrikerna i tabellerna med fetstil om de inte är "Be-7", "Cs-137", "Pb-210", "Na-22".



Be-7

Collection Period	Concentration µBq/m3	Uncertainty-%
3.9.2024 - 4.9.2024	3659.5	8
2.9.2024 - 3.9.2024	2113	8
1.9.2024 - 2.9.2024	1274.6	8
31.8.2024 - 1.9.2024	2250.9	8
30.8.2024 - 31.8.2024	3125.2	8
29.8.2024 - 30.8.2024	3039.9	8

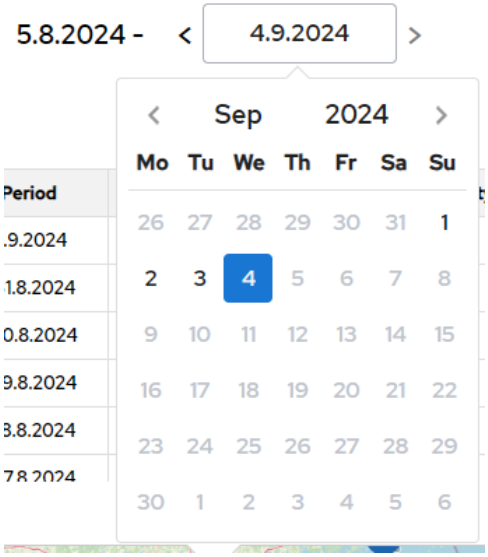
Th-230

Collection Period	Concentration µBq/m3	Uncertainty-%
27.8.2024 - 28.8.2024	256.2	22

Figur 7. Radionuklidernas tabeller

DatePicker.vue:

- En datumväljare i popup-fönstret.
- Begränsad till en tidsram på 30 dagar.



Figur 8. Datumväljare

6.2.2 Backend

Implementeringen gjordes med den äldre WFS API eftersom Finska Meteorologiska Institutets nya EDR API för luftens radionuklider ännu inte fungerar. Ändringar har gjorts i backend-filerna, där `dose_rates.py` har bytts ut mot `process_data.py`, som nu även bearbetar data om luftens radionuklider. En ny API-adress har lagts till i `fmi_utils.py`. I `get_data.py` har ett nytt argument införts för att specificera datatypen: `dose_rates`, `air_radionuclides`, eller `both`.

`process_data.py` skriver nu ner data om luftens radionuklider till katalogen `data/air_radionuclides/datasets`, där filerna namnges med datumet för den första datapunkten i datasetet. `time_series.py` genererar tidsserier för varje radionuklid i datasetet. Den skapar även en `sites.json`-fil som innehåller alla mätstationers koordinater, namn och ID.

För varje mätstations skapas kataloger baserade på ID, samt en `available_radionuclides.json`-fil i varje mätstations katalog, som listar tillgängliga radionuklider. Inom

varje mätstationskatalog skapas mappar för respektive radionuklid, där det även skapas en `available_filenames.json`-fil och tidsseriefiler.

Tidsseriefilerna namnges efter mätperiodens start- och slutdatum, till exempel `2024-06-17TO2024-06-24.json`. Filen `available_filenames.json` innehåller en lista över alla filnamn i mappen.

En API tjänst `air-radionuclides.js` skapas. Filens funktion är som följande:

- Hämtar `available_filenames.json` från filsystemet.
- Hämtar `available_radionuclides.json` från filsystemet.
- Kontrollerar om datumet som skickats från RadionuclideTables.vue finns i radionuklidens katalog i `available_filenames.json`.
- Returnerar tidsserierna för radionukliderna som ligger inom den angivna tidsramen.

Frontend-delen av applikationen kommunicerar med backend för att ta reda på vilka radionuklider som finns tillgängliga på en specifik station. Därefter frågar frontend backend för varje radionuklid om vilka filnamn som finns tillgängliga. `air-radionuclides.js` kontrollerar sedan om det efterfrågade datumet ligger mellan start- och slutdatumet i filnamnet.

Exempel på filstrukturen:

```
data/
└── air_radionuclides/
    ├── datasets/
    │   └── 2024-06-17T120000.json
    └── time_series/
        ├── 123429/
        │   └── Be-7/
        │       ├── 2024-06-17TO2024-06-24.json
        │       ├── 2024-06-24TO2024-07-02.json
        │       └── available_filenames.json
        ├── Cs-137/
        └── available_radionuclides.json
    └── sites.json
```

7 Implementering av kartbytande

En ny funktion som beställaren efterfrågade var möjligheten att byta ut standardbakgrundskartan från OpenStreetMap eller lägga till ytterligare en karttjänst, så att användaren kan välja mellan OpenStreetMap och en annan karta.

7.1 Kraven

Kraven för funktionen var följande:

- En separat fil.
- Filen ska innehålla exempel på formaten WMTS, WMS och VectorTiles.
- Möjlighet att göra ändringarna innan projektet byggs.
- Möjlighet att aktivera eller avaktivera funktionen före projektet byggs.

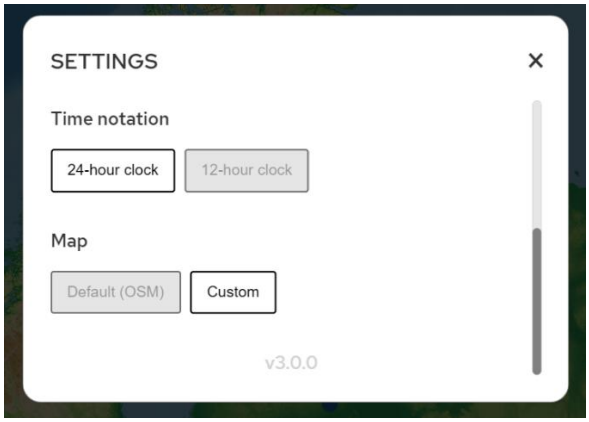
7.2 Implementering

En `mapSettings.js`-fil skapades i mappen Utils. Filen innehåller exempel på olika kartformat, såsom WMTS, WMS och VectorTiles, samt en möjlighet att aktivera eller avaktivera anpassade kartor med `true` eller `false`. Användaren behöver bara ta bort kommentarerna från de relevanta raderna för att använda en anpassad karta och sätta `customMapsEnabled` till `true`. Filens konstanter exporteras till `TheMap.vue`, där OpenLayers (ett bibliotek för att hantera interaktiva kartor i webbläsaren) läser in konstanterna för kartlagren. `TheMap.vue` kontrollerar om ett anpassat lager (`customLayer`) är definierat; om inte, används endast standardlagret (`defaultLayer`).

Konstanten `customMapsEnabled` aktiverar eller avaktiverar knappen för anpassade kartor i applikationens inställningspanel.

```
JS mapSettings.js ×
src > utils > JS mapSettings.js > ...
22  ////////////////////////////////////////////////////////////////////////
23  ///                          DEFAULT OPENSTREETMAP                   ///
24  ////////////////////////////////////////////////////////////////////////
25  // This section configures the default OpenStreetMap layer, which is
26  // a tile layer using the OpenStreetMap tile source.
27
28  const defaultLayer = new TileLayer({
29      source: new OSM({
30          url: "https://{a-c}.tile.openstreetmap.org/{z}/{x}/{y}.png"
31      }),
32      preload: 100,
33  })
34
35  ////////////////////////////////////////////////////////////////////////
36  ///                          CUSTOM MAP SETTINGS                     ///
37  ////////////////////////////////////////////////////////////////////////
38  // Enable custom maps by setting the flag below to true or false.
39  // After enabling, you can switch between the default and a custom
40  // map in the settings panel.
41
42  const customMapsEnabled = false // true
43
44  ////////////////////////////////////////////////////////////////////////
45  ///                    EXAMPLE CUSTOM WMTS VECTORTILE MAP            ///
46  ////////////////////////////////////////////////////////////////////////
47  // This section configures a custom WMTS vector tile map layer using
48  // data from the Finnish National Land Survey (Maanmittauslaitos).
49  // The vector tile format used is MVT, and the data is fetched using
50  // the provided API key.
51
52  /* customLayer = new VectorTileLayer({
53      source: new VectorTile({
54          format: new MVT(),
55          url: `https://avoin-karttakuva.maanmittauslaitos.fi/vectortiles/taust
56      }),
57      preload: 100,
58      declutter: true,
59  }) */
```

Figur 9. mapSettings.js



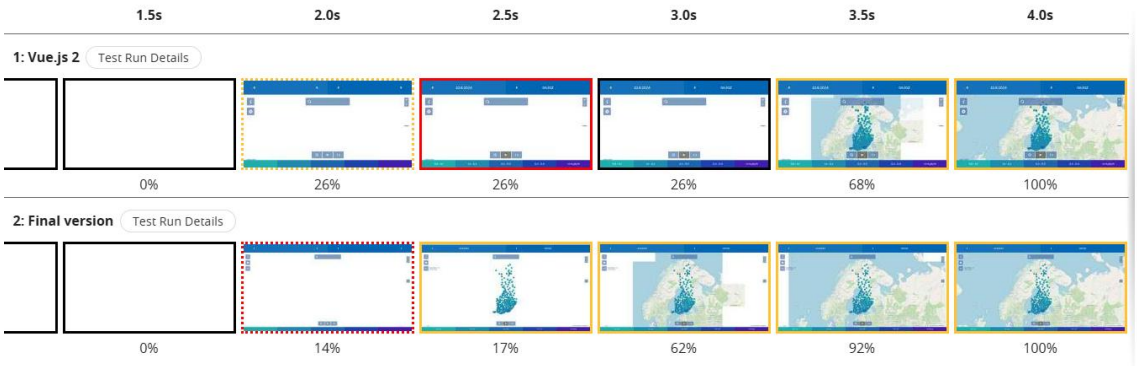
Figur 10. Inställningspanel med kartbytande

8 Resultat

8.1 Effekter på prestanda

WebPageTester kördes på både den gamla sidan, byggd med Vue.js 2, och den nya sidan *Final version* som använder Vue.js 3 med de senaste implementationerna. Två separata testomgångar genomfördes: en för *Vue.js 2* och en för *Final version*. En testomgång bestod av 9 tester, och medianen för omgångarna har beräknats. Testerna utfördes med en resolution på 1920x1080, en upp- och nedladdningshastighet på 100 Mbps, med en ping på 20 ms. Testerna genomfördes från Fredrikshamn, Finland, med Chrome som webbläsare.

I den första delen av testet jämförs sidans laddningshastighet i olika bilder, där man ser hur stor andel av sidan som har laddats vid olika tidpunkter. Bilden visar tydligt att i de tester som genomfördes med *Final version*, laddas datapunkterna och bakgrundskartan snabbare än i testerna med *Vue.js 2*.

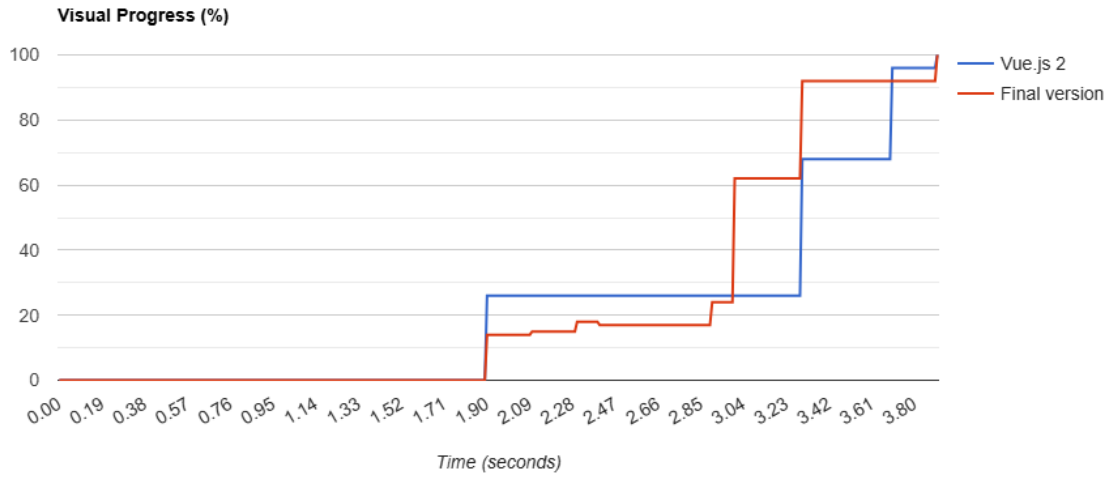


Figur 13. Tidslinje över sidans laddningsprocess

Nästa graf för sidans laddningstid visar en tydlig förbättring i sidans laddningsprestanda med den slutliga versionen, jämfört med den tidigare versionen byggd på Vue.js 2. Den röda linjen, som representerar den slutliga versionen, visar att sidan börjar ladda snabbare och mer konsekvent, medan Vue.js 2 (blå linje) visar större variationer i laddningstiden och flera avbrott. Den slutliga versionen har en jämnare och snabbare ökning av den visuella progressen.

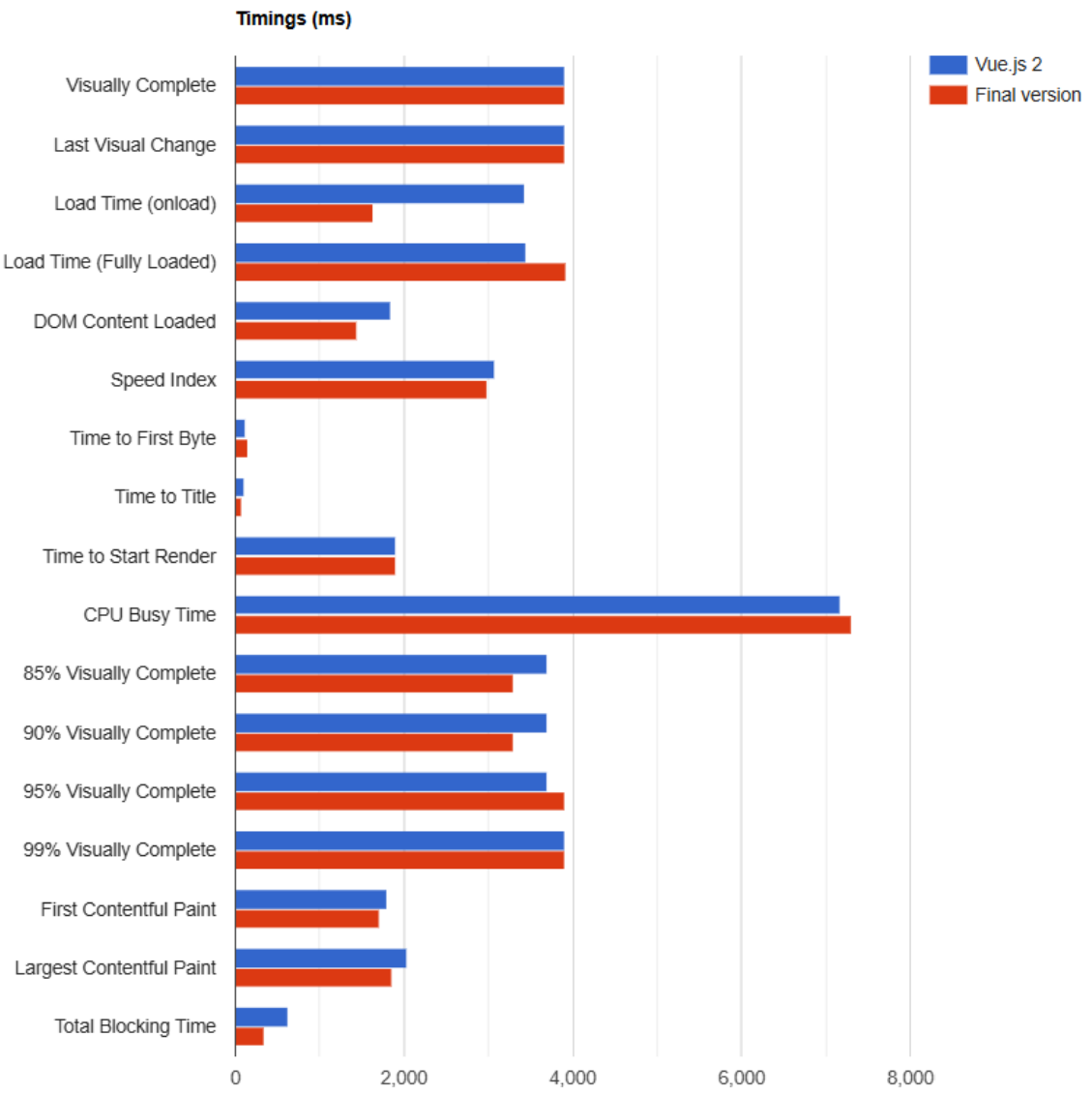
Det är tydligt att i början av laddningsprocessen når den slutliga versionen en högre procentuell visuell progress jämfört till Vue.js 2. Det innebär att användaren får snabbare en visuell sida, där exempelvis kartbilderna laddas och visas tidigare.

Resultatet är förväntat, eftersom den slutliga versionen har optimerats för bättre prestanda och effektivare resursanvändning. Även om skillnaden i laddningstid är 0,5 sekunder, vilket kanske inte har en märkbar effekt för alla användare, bidrar den till en jämnare och mer konsekvent upplevelse. Den snabbare visuella progressen innebär att sidan upplevs som mer responsiv, särskilt vid upprepade sidladdningar.



Figur 14. Graf på visuella laddningsprogressen

Följande graf visar en detaljerad jämförelse av laddningstiderna och av olika prestandamått mellan de två versionerna. Färgerna på staplarna – blå för Vue.js 2, samt röd för den slutliga versionen.



Figur 15. Graf med olika prestandamått

Grafen visar att de två versionerna når ”Visually Complete” och ”Last Visual Change” samtidigt. Måtten för ”Load Time” (både ”onload” och ”Fully Loaded”) indikerar att den slutliga versionen har snabbare initiala laddningstider, även om den kan ta något längre tid för att ladda alla element. Detta tyder på att den slutliga versionen prioriterar att ladda det kritiska innehållet, som är synligt och användbart för användaren, medan bakgrunds-processer och mindre viktiga element laddas senare.

Mätvärdet ”DOM Content Loaded” och ”Speed Index” indikerar också på förbättringar i den slutliga versionen. En snabbare tid för ”Dom Content Loaded” innebär att sidans

struktur och initiala skript laddas snabbare, medan en lägre ”Speed Index” betyder att sidan uppnår ett användbart tillstånd snabbare.

Mätningar som ”First Contentful Paint” och ”Largerst Contentful Paint” är båda snabbare i den slutliga versionen, vilket innebär att viktiga visuella element visas tidigare. Genom att snabbt visa meningsfullt innehåll förbättras användarupplevelsen.

Slutligen visar mätningen av ”Total Blocking Time” att den slutliga versionen har kortare blockeringstid, vilket gör sidan mer responsiv och snabbare interaktiv efter den initiala laddningen.

8.2 Effekter på säkerhet

En säkerhetsscan genomfördes med hjälp av Snyk, ett verktyg som identifierar och rapporterar sårbarheter i beroenden och kodbaser för att förebygga potentiella säkerhetsrisker. I den tidigare versionen av projektet upptäcktes totalt 8 säkerhetshot: 4 klassificerade som höga, 3 som medelhöga, 1 som låg.

Säkerhetshoten som hittades var följande:

1. **Plotly.js Prototype Pollution:**
    - Den allvarligaste sårbarheten (CVE-2023-46308) upptäcktes i `plotly.js@2.17.1`. Sårbarheten tillåter ”prototype pollution”, vilket innebär att en angripare kan manipulera applikationens prototyper och potentiellt ändra funktioner eller orsaka säkerhetsproblem som påverkar hela applikationen.

2. **Word-wrap – Regular expression denial of service (ReDos):**
    - En annan kritisk sårbarhet (CVE-2023-26115) hittades också i `plotly.js@2.17.1`. Denna sårbarhet gör det möjligt för en angripare att överbelasta systemet genom att utnyttja ineffektiva reguljära uttryck.

3. **Decode-uri-component Denial of service (DoS):**
    - Ett allvarligt problem (CVE-2022-38900) upptäcktes i `vue-resource@1.5.3`, där en felaktig hantering av URI-komponenter kan orsaka att tjänsten blir överbelastad.

4. **es5-ext ReDos:**
    - Ytterligare en ReDos-sårbarhet (CVE-2024-27088) identifierades i `plotly.js@2.17.1`, vilket utgör en risk för liknande attacker som utnyttjar ineffektiva reguljära uttryck.

5. **PostCSS Improper Input Validation:**
    - En sårbarhet i `vue@2.7.14` (CVE-2023-44270) relaterad till bristfällig validering av data, kan leda till oväntat beteende eller säkerhetsbrister.

6. **D3-color ReDoS:**
    - En till ReDoS sårbarhet hittades i `plotly.js@2.17.1`, specifikt i d3-color, där ineffektiva reguljära uttryck kan utnyttjas för att överbelasta applikationen.

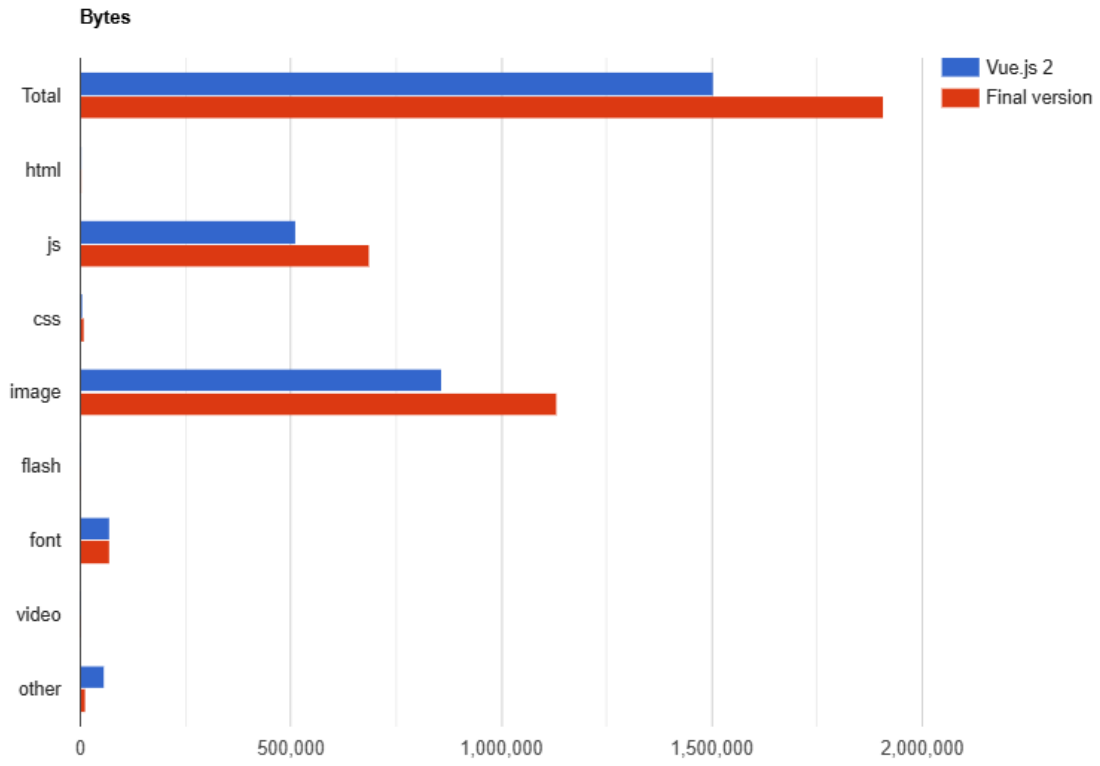
7. **Http-cache-semantics ReDoS**
    - Ett problem i `vue-resource@1.5.3` (CVE-2022-25881), där en ReDoS-sårbarhet påverkar cachehanteringen och kan användas för att överbelasta applikationen.

8. **Got Open Redirect**
    - I `vue-resource@1.5.3` (CVE-2022-33987) identifierades en Open Redirect-sårbarhet som potentiellt kan användas av angripare för att omdirigera användare till skadliga webbsidor.

I den nya versionen av applikationen fanns det inga kända säkerhetshot.

8.3 Effekter på sidans storlek

Om man jämför storlekarna mellan Vue.js 2 och den slutgiltiga versionen ser man att den slutgiltiga versionen har ungefär 405 000 bytes mer i totalt innehåll. Denna ökade storlek består av cirka 150 000 bytes JavaScript och omkring 274 000 bytes mer i bildfiler. De större bildfilerna beror främst på att bakgrundskartan i den nya versionen laddas in tidigare (preload) med ett värde av 100. Detta har implementerats för att förbättra användarupplevelsen genom att se till att bakgrundskartan laddas snabbare och är redo när användaren interagerar med kartan.



Figur 16. Graf över laddade resursernas storlekar

8.4 Resultat av bytet till EDR API

Bytet till det nya EDR API kunde inte totalt fullföljas eftersom Finska Meteorologiska Institutets EDR API för luftens radionuklider ännu inte var färdigt eller fullt fungerande vid tidpunkten för bytet. EDR API fungerar endast för strålningsdosdata, vilket var målet, men inte för data om luftens radionuklider.

Eftersom det nya EDR API inte kunde implementeras fullt ut, var det inte möjligt att utvärdera dess effekt på exempelvis prestanda och resursanvändning. Dock har grunden lagts för en framtida övergång när EDR API är fullständigt operativt, och en struktur har skapats för att underlätta den fortsatta implementeringen.

9 Slutsatser och framtida utvecklingsmöjligheter

Migrationen från Vue.js 2 till Vue.js 3 var lyckad, och den nya versionen av applikationen har förbättrad prestanda och säkerhet jämfört med den tidigare versionen. För att genomföra migrationen framgångsrikt och förbättra applikationens prestanda och säkerhet användes en stegvis uppdatering av kodbasen, där äldre Vue.js 2-komponenter ersattes med Vue.js 3-kompatibla motsvarigheter. Migrationen genomfördes med stöd av Vue Migration Build och Vue CLI för att identifiera och uppdatera äldre syntax och metoder. De största utmaningarna under migreringen var förändringar i Vue.js 3:s reaktivitetsmodell och livscykelhantering, vilket krävde omskrivning av vissa delar av koden. Problem som uppstod vid refaktoreringen av komponenter hanterades genom att uppdatera kodstrukturen och använda Vue Composition API för bättre kodhantering och underhållbarhet.

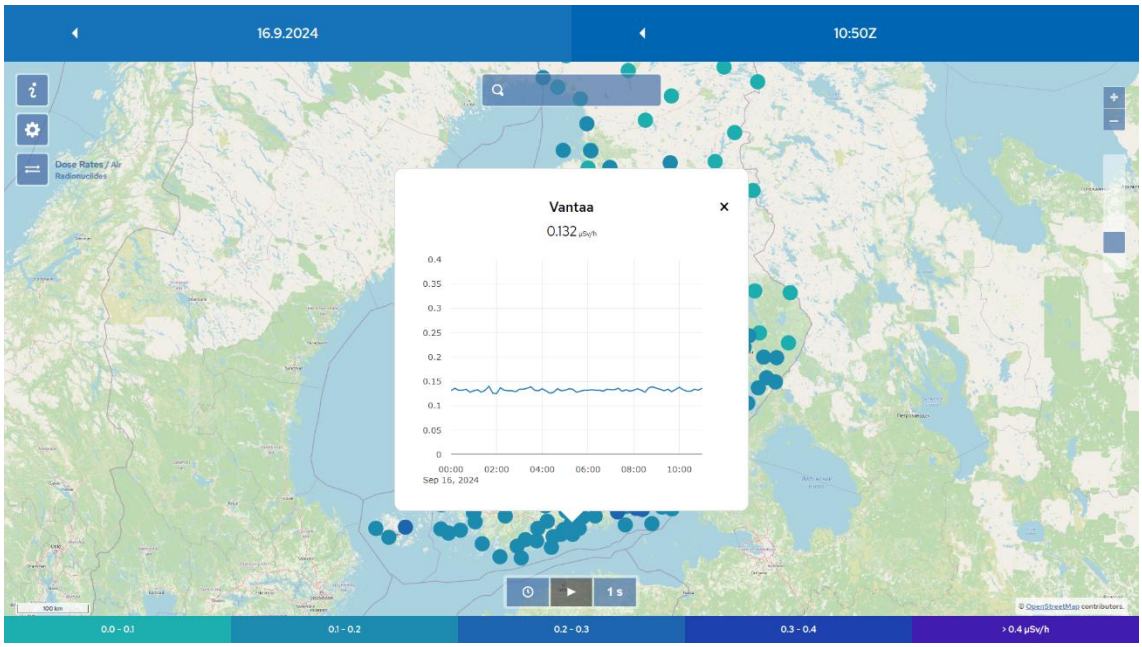
Integrationen av visualiseringen av luftens radionuklider samt möjligheten att byta bakgrundskarta genomfördes utan problem. Planen var att hämta strålningsdosdata via EDR API och luftens radionuklider via WFS API, vilket lyckades. EDR API har dock flera fördelar jämfört med WFS API, såsom effektivare datahämtning, bättre stöd för tidsserier och högre precision vid dataförfrågningar. För framtiden är det en möjlighet att fortsätta bytet från WFS API till EDR API, när EDR API är fullt fungerande. En ny mappstruktur skapades för `edr_scripts` och `wfs_scripts`, där `edr_scripts` innehåller ett fungerande skript för att hämta strålningsdosdata via EDR API. Däremot är stödet för att ladda luftens radionukliddata från EDR API ännu inte implementerat.

För att verifiera förbättrad prestanda och säkerhet genomfördes prestandamätningar baserade på laddningstider, filstorlek och renderingsprestanda. Säkerhetsgranskning utfördes genom analysverktyg som Snyk, men en framtida förbättring vore att inkludera fler verktyg såsom ESLint Security Plugins eller SonarQube för mer omfattande analys. Resultaten visade att den uppdaterade applikationen hade lägre initial laddningstid och

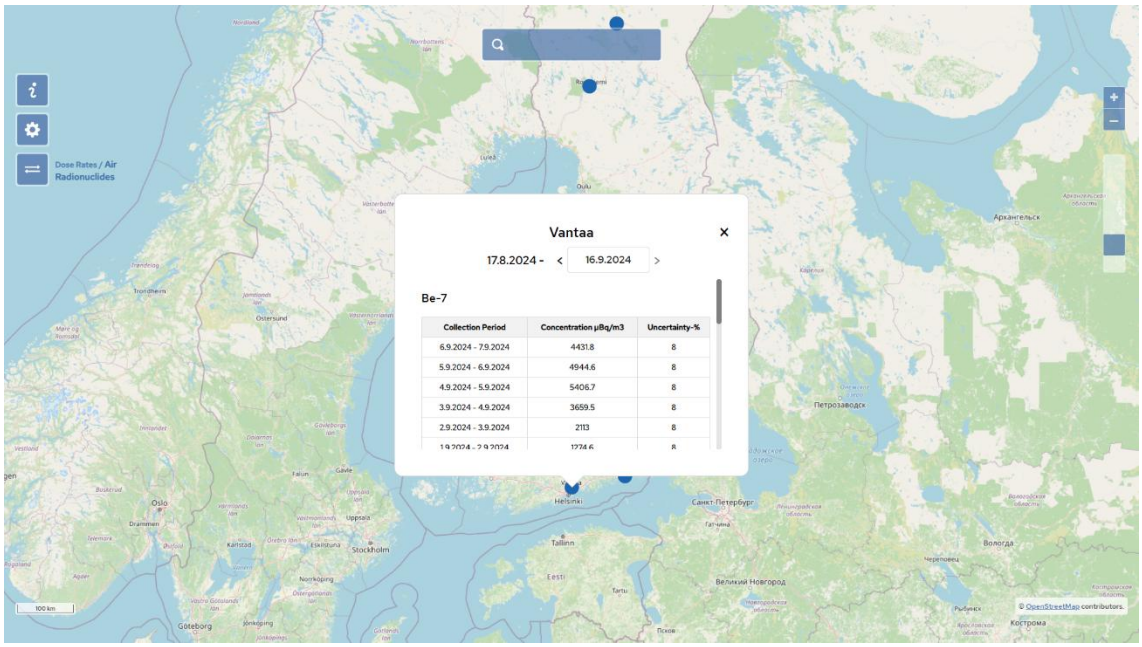
förbättrad renderingseffektivitet, samtidigt som säkerhetsrisker relaterade till äldre beroenden eliminerades.

Grunden är lagd för att implementera dataladdning från EDR API för luftens radionuklider, eftersom WFS-skriptet kan användas som bas och endast kräver små justeringar i koden för att stödja EDR. Ett annat utvecklingsområde är att optimera applikationen genom att skapa en version som hämtar data direkt från EDR API till frontend. Detta skulle minska både lagringskraven och reducera applikationens komplexitet.

En rekommendation är att kontinuerligt hålla applikationen uppdaterad med de senaste versionerna av beroenden och ramverk. Detta minskar risken för allvarliga säkerhetsproblem och undviker situationer där en uppdatering av hela kodbasen är nödvändig. Regelbundna uppdateringar gör det också enklare att hantera säkerhetshot i tid och säkerställer att applikationen är stabil och presterar optimalt.



Figur 17. Färdiga applikationen, strålningsdoser



Figur 18. Färdiga applikationen, luftens radionuklider

Källor

OGC. (u.å.-a). OGC API – Environmental Data Retrieval. OGC.

https://www.ogc.org/standard/ogcapi-edr/ Hämtad 29.7.2024

OGC. (u.å.-b). Web Feature Service. OGC.

https://www.ogc.org/standard/wfs/ Hämtad 29.7.2024

Monterail. (2023, March 3). *Vue 2 to Vue 3 Migration [Video]:*

Youtube.

https://www.youtube.com/watch?v=TKM2vwTF3QQ

Tinker. (2023, December 27). *Differences between VueJS 2 and VueJS 3*. Devblog from SHIFT ASIA.

https://blog.shiftasia.com/difference-between-vuejs-2-and-3-when-to-use-them/

Hämtad 24.7.2024

Versionlog. (u.å.). *Vue.js*. Versionlog.
https://versionlog.com/vuejs/ Hämtad 24.7.2024

Vue.js. (2015, October 26). *Vue.js 1.0.0 Released.* Vue.js.

https://v2.vuejs.org/2015/10/26/1.0.0-release/ Hämtad 24.7.2024

Vue.js. (2016, April 27). *Announcing Vue.js 2.0*. Vue.js.
https://v2.vuejs.org/2016/04/27/announcing-2.0/ Hämtad 24.7.2024

Vue.js. (u.å.-a). *Breaking Changes***.** Vue.js.

https://v3-migration.vuejs.org/breaking-changes/ Hämtad 29.7.2024

Vue.js. (u.å.-b). *Frequently Asked Questions*. Vue.js.

https://vuejs.org/about/faq Hämtad 24.7.2024

Vue.js. (u.å.-c). *Introduction*. Vue.js.

https://vuejs.org/guide/introduction.html Hämtad 24.7.2024

Vue.js. (u.å.-d). *Migration Build*. Vue.js.
https://v3-migration.vuejs.org/migration-build.html Hämtad 6.8.2024

Wikipedia. (u.å.). *Vue.js.* Wikipedia.
https://en.wikipedia.org/wiki/Vue.js Hämtad 24.7.2024