LAPIN AMK
Lapland University of Applied Sciences

WEB-APPLIKAATIO KOIRIEN AIHEUTTAMISTA VAHINGOISTA POROILLE

Hienonen Mirka

Opinnäytetyö

Tieto- ja viestintätekniikka
Insinööri (AMK)

2024

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tieto- ja viestintätekniikka
insinööri (AMK)

Tekijä	Mirka Hienonen	**Vuosi**	2024
Ohjaaja	Aku Kesti		
Toimeksiantaja	Paliskuntain yhdistys		
Työn nimi	Web-applikaatio koirien aiheuttamista vahingoista poroille		
Sivumäärä	45		

Tässä opinnäytetyössä kehitetään Paliskuntain yhdistykselle web-sovellus, jonka avulla käyttäjät voivat raportoida koirien poroille aiheuttamia vahinkoja lomakepohjan kautta. Opinnäytetyön päätavoitteena on esittää sovelluksen kehitysprosessi ja sen keskeiset ominaisuudet.

Kehitysprosessi käsitellään vaiheittain, alkaen käyttöliittymän (UI) ja käyttäjäkokemuksen (UX) suunnittelusta. Tämän jälkeen keskitytään frontendin toteutukseen Reactilla ja backendin toteutukseen PHP:llä, tuodaan esiin teknologioiden valintaperusteet suhteessa käytettävän tietokantatyypin vaatimuksiin ja rajoitteisiin. Opinnäytetyössä käsitellään lisäksi sovelluksen testaamista palvelinympäristössä.

Tuloksena syntyi operatiiviseen käyttöön suunnattu fullstack-sovellus, joka sisältää julkisen lomakeosion PC- ja mobiilikäyttöön sekä PC:lle suunnatun adminosuuden, jonka kautta tallennettua dataa voidaan hallita ja tarkastella. Valmis sovellus parantaa porotalouden digitaalista arkea helpottamalla tietojen keruuta, hallintaa ja tilastointia.

Avainsanat: backend, frontend, PHP, React, UI, UX

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

Study Programme in Information and Communication Technology
Bachelor of Science

Author Mirka Hienonen **Year** 2024
Supervisor Aku Kesti
Commissioned by Reindeer Herding Association
Title Web Application for Reindeer Damages Caused by Dogs
Number of pages 45

The aim of this thesis study was to develop a web application for the Reindeer Herding Association, enabling the users to report reindeer damages caused by dogs. The primary objective of the thesis is to present the development process of the application and its key features.

The development process was outlined step-by-step, beginning with the design of the user interface (UI) and user experience (UX). The focus then shifted to the implementation of the frontend using React and the backend using PHP, with an emphasis on the selection criteria for these technologies in relation to the requirements and limitations of the predefined database. The study also covered the testing of the application in a server environment.

This thesis presents the development process of the application and its key features. The final product is a comprehensive full-stack application designed for operational use. It includes a public form section accessible on both PC and mobile devices, as well as an admin section limited to PC users for efficient management and viewing of stored data. This application enhances digital management in the reindeer industry by facilitating seamless data collection, streamlined management processes, and in-depth statistical analysis.

Keywords: backend, frontend, PHP, React, UI, UX

SISÄLLYS

ALKUSANAT

Haluan kiittää Paliskuntain yhdistystä siitä, että sain mahdollisuuden toteuttaa tämän upean projektin. Erityiskiitos kuuluu Aarrelle ja Maarenille heidän sujuvasta yhteistyöstään ja arvokkaiden materiaalien toimittamisesta.

Samalla haluan osoittaa kiitokseni FrostBitin Web-tiimille heidän tarjoamastaan tuesta, hyvistä neuvoista ja jatkuvasta kannustuksesta. Suuret kiitokset myös ohjaajilleni Aku Kestille, Tuija Haapasalmelle ja Ritva Lampelalle, joiden ansiosta opinnäytetyö eteni ripeästi ja määrätietoisesti.

1 JOHDANTO

Opinnäytetyöni tarkoitus on suunnitella ja toteuttaa Paliskuntain yhdistykselle käyttäjäystävällinen web-sovellus, johon poronomistajat voivat kirjata koirien poroille aiheuttamat vahingot. Kirjatut vahingot tallennetaan tietokantaan tilastointia sekä myöhempää visuaalista tarkastelua varten. Sovelluksen kehitystyön taustalla oli tarve parantaa olemassa olevia työkaluja, jotka eivät sellaisenaan tukeneet kaikkia loppukäyttäjien tarpeita.

Uuden sovelluksen keskeisiä vaatimuksia olivat vahinkopaikan merkitseminen kartalle, lomakkeen toimivuuden varmistaminen myös mobiililaitteilla sekä mahdollisuus tarkastella ja vertailla vahinkodataa useammalta vuodelta horisontaalisten pylväsdiagrammien avulla. Näiden toiminnallisuuksien lisäksi sovelluksen tuli olla mahdollisimman automatisoitu, jotta prosessit, kuten lomakkeiden lähettäminen, muokkaaminen, poistaminen ja datan lataaminen ylläpitäjän koneelle, sujuisivat helposti ja tehokkaasti.

Erityisen tärkeää on, että sovellus mahdollistaa vuosien aikana kerätyn vahinkotiedon visuaalisen esittämisen karttapohjalla, mikä parantaa vahinkojen analysointia ja auttaa käyttäjiä havainnollistamaan tietoja. Näiden vaatimusten pohjalta kehitetty sovellus tarjoaa poronomistajille selkeän ja tehokkaan työkalun vahinkojen kirjaamiseen, hallintaan ja tilastointiin parantaen samalla tietojen käytettävyyttä ja analysointimahdollisuuksia.

Opinnäytetyössä kehityksen esittely keskittyy UI- ja UX suunnitteluun sekä käyttöliittymän eri toiminnallisuuksiin ja eri toteutusratkaisuihin. Toimeksiantajan pyynnöstä ja tietoturvasyistä tietokannan rakennetta, backend-koodia tai oikeaa dataa ei esitellä.

2 UI- JA UX-SUUNNITTELU

Sovelluksen toteutus käynnistyi käyttöliittymän suunnittelusta, joka toteutettiin Adobe XD:llä asiakkaan toiveita mukaillen. Adobe XD on vuonna 2016 julkaistu vektoripohjainen suunnittelutyökalu, joka keskittyy erityisesti käyttäjäkokemuksen suunnitteluun (Hyttinen 2023).

Suunnitelman tuli kattaa visuaalinen demonstraatio sovelluksen keskeisistä sivuista, toiminnallisuuksista ja navigaatiosta siten, että suunnitelma mahdollisti toteutuksen aloittamisen, mutta tilaa mahdollisille muutoksille silti jäisi. Apuna suunnittelussa käytettiin erilaisista aihetta sivuavista artikkeleista kerättyä tietoa sekä henkilökohtaista tutkimusta erilaisten työsovellusten ulkonäöstä ja toiminnallisuuksista, samoin kuin saavutettavuutta mittaavia online-työkaluja. Lähestymistapa toteutukseen oli asiakaslähtöinen, intuitiivinen ja kokeileva parhaan mahdollisen lopputuloksen saavuttamiseksi.

2.1 UI:n ja UX:n erot ja yhtäläisyydet

Käyttäjäkokemuksen (UX) ja käyttöliittymän (UI) suunnittelu ovat kaksi keskeistä osa-aluetta ohjelmistokehityksessä. Vaikka nämä kaksi kenttää usein nivoutuvat yhteen, niillä on selkeitä eroja, jotka vaikuttavat tuotteen kehitysprosessiin (Lamprecht 2023).

UX-suunnittelu keskittyy käyttäjän kokonaisvaltaiseen kokemukseen tuotteen kanssa ja kattaa kaikki käyttäjän ja tuotteen väliset vuorovaikutukset. UX-suunnittelijan päätehtävänä on ymmärtää, miten käyttökokemus vaikuttaa käyttäjän tunteisiin ja kuinka vaivattomasti käyttäjä voi suorittaa haluamansa tehtävät. Tämä vaatii käyttäjien käyttäytymisen tarkkailua, tehtävien analysointia ja kokemuksen parantamista, jotta se olisi mahdollisimman helppoa, tehokasta, merkityksellistä ja miellyttävää. (Lamprecht 2023.)

UX-suunnittelu ei keskity visuaalisiin elementteihin, vaan siihen, miten käyttäjä navigoi ja kokee tuotteen käytön. Tavoitteena on luoda mahdollisimman positiivinen käyttäjäkokemus ottaen huomioon kaikki vuorovaikutuksen osa-alueet.

Tämä voi sisältää esimerkiksi kilpailija-analyysiä, asiakasohjelmatutkimusta, rautalankamallien ja prototyyppien luomista sekä jatkuvaa testaamista ja iterointia. (Lamprecht 2023.)

UI-suunnittelu puolestaan keskittyy tuotteen visuaaliseen ilmeeseen ja interaktiivisiin elementteihin. Se käsittää kaikki visuaalisesti havaittavat osat, kuten ikonit, painikkeet, typografian, väriteemat ja responsiivisen suunnittelun. UI-suunnittelijan tehtävänä on varmistaa, että käyttöliittymä on intuitiivinen ja käyttäjäystävällinen. Tämä tarkoittaa, että käyttöliittymän tulee ohjata käyttäjää selkeästi ja tehokkaasti tuotteen käytössä minimoiden käyttäjän ajattelun tarpeen. (Lamprecht 2023.) Käyttöliittymä tulee suunnitella siis siten, että myös sovellusta ensi kertaa käyttävä henkilö saa asiansa hoidettua sujuvasti ja luonnollisesti.

UI-suunnittelu on tiukasti digitaalista ja keskittyy siihen, miten visuaalinen ja interaktiivinen suunnittelu vaikuttaa käyttäjän kokemukseen. UI-suunnittelija huolehtii siitä, että brändin visuaaliset elementit siirtyvät sujuvasti tuotteen käyttöliittymään sekä siitä, että design on johdonmukainen ja esteettisesti miellyttävä. (Lamprecht 2023.)

Voidaan siis todeta, että vaikka UX- ja UI-suunnittelu ovat läheisesti yhteydessä toisiinsa, niiden painopisteet ovat erilaiset. UX-suunnittelu keskittyy siihen, miten käyttäjä kokee ja käyttää tuotetta, kun taas UI-suunnittelu painottaa tuotteen visuaalista ulkoasua ja sen vuorovaikutteisia osia. UX-suunnittelija tarkastelee, miten kokemus voidaan optimoida käyttäjän näkökulmasta, kun taas UI-suunnittelija varmistaa, että visuaaliset ja interaktiiviset elementit tukevat tätä kokemusta. Molemmat roolit ovat tärkeitä, mutta niiden lähestymistavat ja tavoitteet ovat erilaiset täydentäen toisiaan käyttäjäystävällisen ja toimivan tuotteen luomisessa. (Lamprecht 2023.)

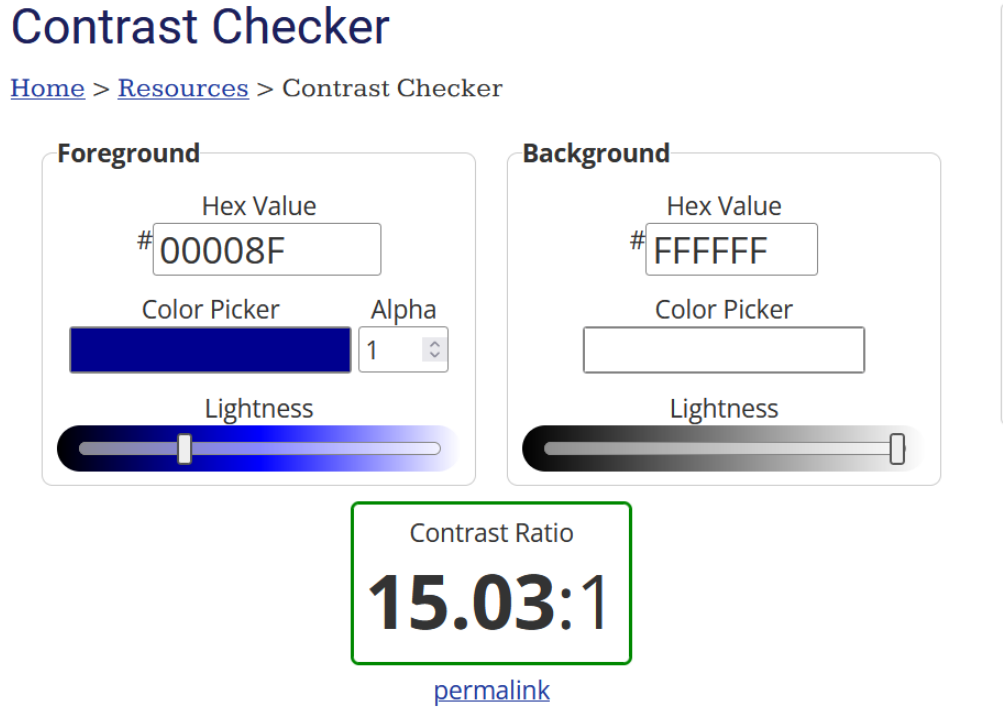
2.2 Saavutettavuus suunnittelun keskiössä

UI- ja UX-suunnittelussa kiinnitettiin erityistä huomiota saavutettavuuteen, koska kyseessä on työsovellus. Saavutettavuudella tarkoitetaan erilaisten käyttäjäryh-

mien huomioimista sovelluksen suunnittelussa ja toteutuksessa siten, että toteutus auttaa käyttäjiä vuorovaikuttamaan digitaalisten tuotteiden kanssa heille parhaiten sopivilla tavoilla (Koch 2024).

Suunnittelussa saavutettavuus huomioitiin pitämällä fonttikoko riittävän suurena sekä huolehtimalla riittävistä väleistä ja värikontrastieroista luettavuuden takaamiseksi. Tällä tavoin teksti ja elementit, kuten nappulat ja työkaluikonit erottuvat riittävästi sivulta, mikä mahdollistaa saavutettavuuden myös heikkonäköisille ja värisokeille käyttäjille. Lisäksi suunnittelussa on huomioitu navigaatio elementtien sijoittelu siten, että nappulat ovat tarpeeksi kaukana toisistaan vahinkoklikkausten minimoimiseksi, koska osa käyttäjistä saattaa navigoida sovelluksessa tasohiirtä käyttäen tai mobiililaitteella, jolloin näytön koko on luonnollisesti pienempi. (Koch 2024.)

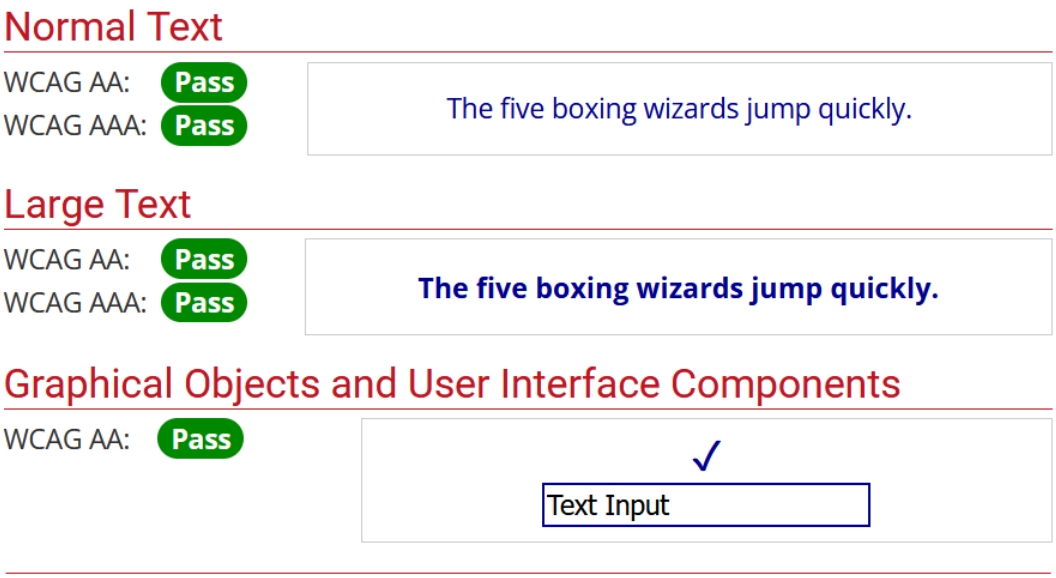
Apuna käytettiin lisäksi erilaisia online-työkaluja, kuten Web Aimin Contrast Checkeriä eli kontrastitarkistinta, johon syötettiin vertailtavien värien Hex-arvot. Kontrastimittari (kuvio 1) vertailee värien välistä kontrastia UI-komponenteissa huomioiden myös eri fonttivahvuudet ja ilmoittaa minkä saavutettavuusluokan kontrasti läpäisee.



Kuvio 1. Kontrastitarkistin (WebAIM 2024)

Kuvio 1 esittelee sovelluksen päävärien välistä suhdetta. Eri värien ja värisävyjen vertailu sujuu vaivattomasti liukusäätimellä ja läpi menneen värin koodi on helppo kopioida kontrastitarkistimesta suoraan sovelluksen koodiin.

Kuvio 2 on jatkoa kuviolle 1 ja esittelee loput kontrastitarkistimen tarjoamasta informaatiosta. Kuviosta nähdään, että vertaillut värit ovat läpäisseet AA-luokan jokaisessa skenaariossa ja yltäneet myös AAA-luokkaan. AA-luokan läpäisemisen katsottiin olevan riittävä sovelluksen toteutukselle, koska kaikki elementit eivät kykene saavuttamaan AAA-luokkaa.



Kuvio 2. Kontrastitarkistimen saavutettavuusluokat (WebAIM 2024)

Kuviossa 2 mainituilla luokilla tarkoitetaan saavutettavuusluokkia, joita esittelemällä Milbergs (2023) tarjoaa syventävän katsauksen Kochin (2024) huomioihin saavutettavuudesta. Lyhenne WCAG tulee sanoista Web Content Accessibility Guidelines, joka on kansainvälinen ohjeistus verkkosivujen saavutettavuudesta. Suomeksi ohjeistuksesta käytetään termiä verkkosisällön saavutettavuusohjeet. (Aluehallintavirasto 2024.)

Saavutettavuusluokat jaotellaan WCAG-vaatimustenmukaisuustasojen perusteella kolmeen eri tasoon. Taso A on vähimmäisvaatimustaso, joka kattaa perus saavutettavuusominaisuudet, kuten tekstivastineet, selkeän ja johdonmukaisen navigoinnin sekä näppäimistön käytettävyyden. (Milbergs 2023.)

Taso AA edustaa saavutettavuusvaatimuksien edistyneempää luokkaa, jonka tavoitteena on parantaa verkkosivujen käytettävyyttä laajemmalle yleisölle. Se rakentuu Taso A:n perusohjeiden varaan, mutta lisää vaatimuksia, jotka tekevät sivustosta entistäkin saavutettavamman. Näihin vaatimuksiin sisältyvät esimerkiksi tekstitykset videoille, mikä parantaa sisältöjen saatavuutta kuuroille tai heikkokuuloisille käyttäjille. Lisäksi Taso AA:n mukaiset sivustot tukevat visuaalisia ominaisuuksia, kuten tummaa tai värisokeustilaa, jotka helpottavat sisällön lukemista ja ymmärtämistä erilaisilla näköhaasteilla. Erityisen tärkeää on myös se, että kaikkia sivuston osia voidaan käyttää vain näppäimistöllä, mikä on keskeistä käyttäjille, joilla on rajoituksia hiiren käytössä. Näin Taso AA nostaa saavutettavuuden vaatimustason huomioimalla entistä monipuolisemmin erilaisten käyttäjien tarpeet. (Milbergs 2023.)

Taso AAA edustaa digitaalisen saavutettavuuden korkeinta tasoa, ja sen saavuttaminen tarkoittaa, että digitaalinen sisältö on erittäin saavutettavaa kaikille, mukaan lukien vammaiset henkilöt. Tämä taso perustuu aikaisempien tasojen A ja AA vaatimuksiin ja lisää tiukkoja ohjeistuksia, joiden avulla saavutettavuus voidaan viedä optimaaliselle tasolle. Taso AAA sisältää muun muassa viittomakielen tulkkauksen kaikille ennakkoon tallennetuilla videoilla ja äänitallenteilla sekä mahdollistaa tekstin skaalauksen jopa 200 prosenttia suuremmaksi ilman toiminnallisuuden heikkenemistä. (Milbergs 2023.)

Taso AAA on digitaalisen saavutettavuuden huippu ja sen saavuttaminen asettaa tiukkoja vaatimuksia. Tämä taso kattaa aikaisempien tasojen A ja AA vaatimukset ja se käsittelee saavutettavuuden haastavimpia osa-alueita, mikä tekee siitä tärkeän virstanpylvään saavutettavassa verkkosisällössä. (Milbergs 2023.)

3 FRONTEND

Frontendin eli käyttöliittymän toteutukseen valittiin ohjelmointikieleksi React yhdessä TypeScriptin kanssa. React on Facebook-kehittäjä Jodran Walken vuonna 2013 luoma ohjelmointikieli, joka tuli julkiseen levitykseen vuonna 2015 (Mohsin 2024). TypeScript puolestaan on vahvasti tyypitetty ohjelmointikieli, joka laajentaa JavaScriptiä lisäämällä tyyppien syntaksin. Tämä parantaa koodin turvallisuutta ja virheiden havaitsemista kehitysvaiheessa. TypeScript-koodi käännetään JavaScriptiksi, joten sitä voidaan käyttää missä tahansa JavaScript-ympäristössä, kuten selaimissa ja Node.js:ssä. (Microsoft 2024.)

React.js valittiin projektin ohjelmointikieleksi komponenttipohjaisen arkkitehtuurinsa takia, joka mahdollisti monimutkaisen käyttöliittymän jakamisen uudelleen käytettäviksi komponenteiksi. Kehys hyödyntää JavaScript-ominaisuuksia, kuten JSX:ää, joka integroi HTML:n JavaScriptiin ja tekee komponenttien koodaamisesta ja hallinnasta helpompaa (Mohsin 2024). Tässä yhteydessä uudelleen käytettävillä komponenteilla tarkoitetaan käyttöliittymän osia, kuten navigaatiopalkkia, näppäimiä tai vastaavia, sovelluksen toimivuuden kannalta tärkeitä toistuvia elementtejä. Tämä ei ainoastaan yksinkertaistanut kehitysprosessia, vaan paransi myös tehokkuutta, erityisesti sovelluksen koon huomioon ottaen (Mohsin 2024).

3.1 Käyttöliittymän osat ja tehtävät

Sovelluksen käyttöliittymä jakaantuu kahteen osaan: vahinkolomakkeeseen, jonka lähettäminen ei vaadi kirjautumista, sekä kirjautumistunnusten takana olevaan admin-sivuun, josta tietyt käyttäjät pääsevät tarkkailemaan ja lataamaan dataa. Käyttöliittymän jako palvelee johdannossa esiteltyjä vaatimuksia, jotka jakautuvat kuviossa 3 esitetyllä tavalla seuraaviin kategorioihin:

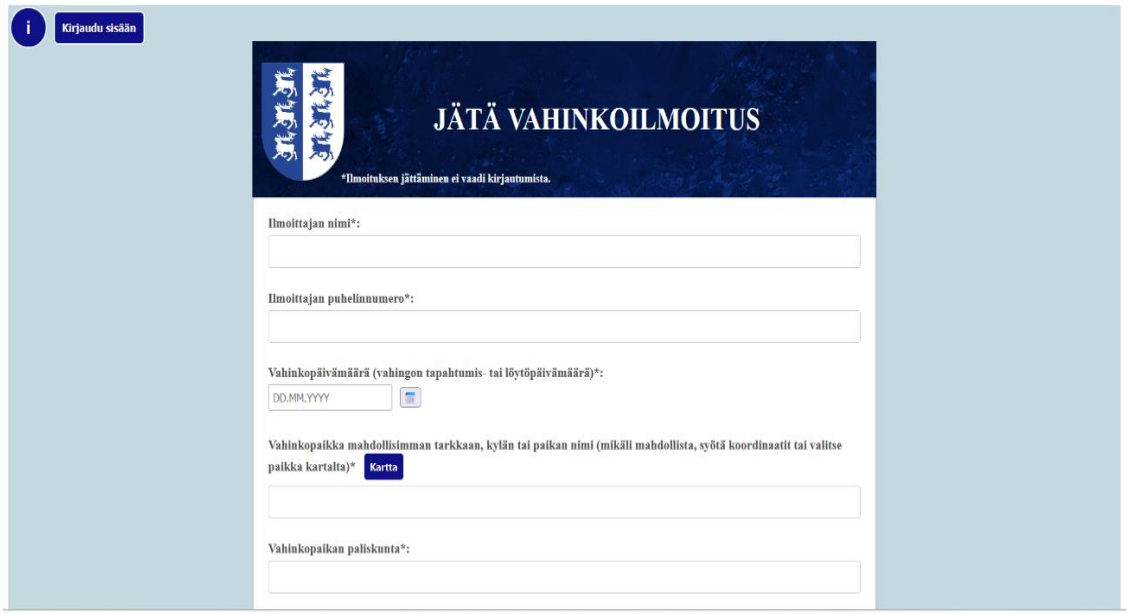
- **Käytettävyys:** Varmistaa helpon ja sujuvan käyttäjäkokemuksen, tekee sovelluksen käytöstä intuitiivista ja alentaa oppimiskynnystä.
- **Data-analyysi:** Mahdollistaa tehokkaan tiedon käsittelyn ja raportoinnin, auttaa käyttäjiä tekemään tietoon perustuvia päätöksiä ja seuraamaan sovelluksen suorituskykyä.

- **Automaatio:** Tehostaa prosesseja ja vähentää manuaalista työtä, parantaa työskentelyn tehokkuutta ja mahdollistaa käyttäjien keskittymisen tärkeämpiin tehtäviin.

Kuvio 3. Sovelluksen keskeiset vaatimukset kategorioittain

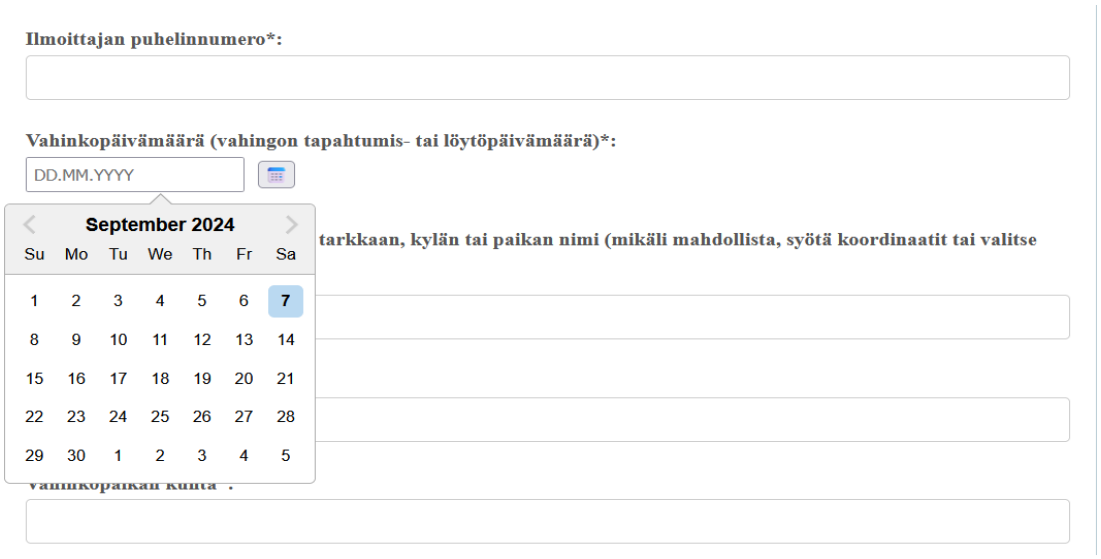
3.2 Vahinkolomake ja sen toiminnallisuudet

Käyttöliittymän värit valittiin Paliskuntain yhdistyksen logon mukaan sekä sävyt kuviossa 1 ja kuviossa 2 esiteltyjen kontrastimittareiden avulla. Eläväisyyttä sovellukseen tuovat JPG- ja PNG- formaatein toteutetut tekstuurit, jotka esiintyvät esimerkiksi lomakkeen ylätunnisteessa ja kirjautumisnapissa (kuvio 4).



Kuvio 4. Sovelluksen vahinkolomake

Lomake käsittää kaikkiaan 20 kysymystä, joista osa tulee käyttäjälle näkyviin aiempien vastausten perusteella. Kysymykset käsittivät sekä vapaasti kirjoitettavaa tekstiä, valintapainikkeita että päivämäärän (kuvio 5) ja koordinaattien asettamisen.



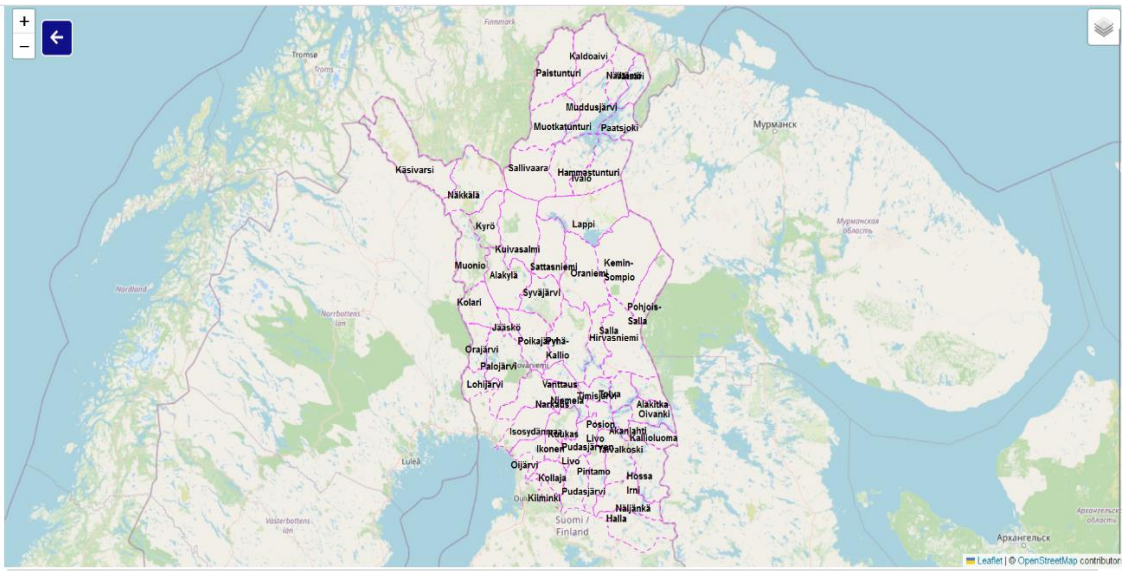
Kuvio 5. Datepicker kalenteri

Kehityksessä harkittiin kalenterin lukitsemista meneillään olevaan kuukauteen, mutta lopulta päätettiin sallia myös menneiden ja tulevien kuukausien valitsemi-

nen. Tämä päätös tehtiin, koska ilmoitusta ei aina voida jättää heti vahingon tapahduttua. Esimerkiksi, jos vahinko sattuu kuukauden viimeisenä päivänä, ja ilmoitus jätetään vasta seuraavana päivänä, tilastointi vääristyisi. Lisäksi admintilassa voidaan vertailla ilmoituksen jättöpäivää ja vahingon tapahtumapäivää, jolloin esimerkiksi tulevaisuuteen kirjattuja vahinkoja, jotka ovat virheellisiä, voidaan poistaa tietokannasta.

Vahinkopaikan valitseminen kartalta oli eräs sovelluksen toivotuista ominaisuuksista. Kartta toteutettiin Leafletillä, joka on johtava avoimen lähdekoodin JavaScript-kirjasto ja sopii hyvin mobiiliystävällisiin interaktiivisiin karttoihin. Sen koko on vain noin 42 kilotavua, ja se sisältää kaikki kartoitustoiminnot, joita useimmat kehittäjät tarvitsevat (Agafonkin 2024).

Jotta oikean vahinkopaikan valitseminen sujuisi mahdollisimman helposti, karttaan (kuvio 6) lisättiin useita eri topo-kerroksia, kuten open-street-map, vaalea topografinen kartta, satelliittikartta ja Maanmittauslaitoksen sivuilta saatu maastokartta. Lisäksi karttaan luotiin GeoJSON-muodossa paliskuntien rajat ja nimet. Käyttäjä voi tarkastella karttaa myös valitsemaltaan zoom-tasolta.



Kuvio 6. Lomakkeen kautta avautuva karttasivu

GeoJSON on formaatti, jota käytetään erilaisten maantieteellisten tietorakenteiden koodaamiseen. Se on JSON-objekti, joka sisältää tietoa tietyistä geometristä muodoista, kuten Point, LineString, Polygon, MultiPoint, MultiLineString ja MultiPolygon. Geometriset objektit, joilla on lisäominaisuuksia, ovat Feature-objekteja,

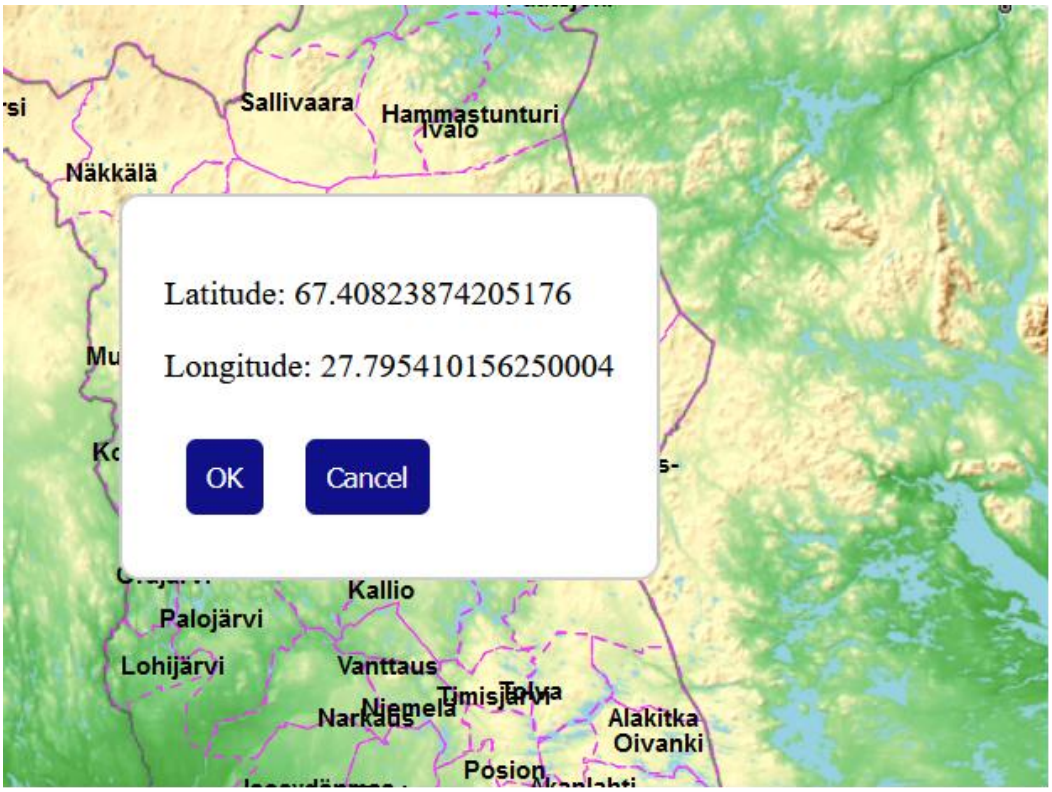
ja ominaisuusjoukot sisältyvät FeatureCollection-objekteihin. GeoJSON-muotoilu seuraa tiettyjä standardeja, joiden rikkominen voi heikentää tiedoston toimivuutta (Lodha 2020).

GeoJSON mahdollistaa myös metatietojen, kuten tunnistenumeroiden, lisäämisen ominaisuuksiin. Esimerkiksi tyypillinen GeoJSON-tiedosto alkaa FeatureCollection-objektilla, jossa on listattuna ominaisuudet. Oikein muotoiltu GeoJSON-tiedosto voi sisältää lisämetatietoa, kuten koordinaattijärjestelmän (CRS) tiedot. (Lodha 2020.)

GeoJson-tiedosto luotiin Paliskuntain yhdistykseltä saadun shape-tiedoston pohjalta konvertoimalla se QGIS-työpöytäsovelluksella oikeaan formaattiin. QGIS (Quantum Geographic Information System) on ilmainen avoimen lähdekoodin ohjelmisto, joka mahdollistaa maantieteellisen tiedon luomisen, muokkaamisen, analysoinnin ja julkaisemisen. (Palino & Sparks 2021.)

GeoJsoniksi konvertoitu tiedosto sisälsi FeatureCollection-objektin, josta renderöitäväksi valittiin viivojen muodostamat paliskuntienrajat (*multipolygon*) sekä paliskuntien nimet (*feature*). Lukuisista yrityksistä huolimatta GeoJson ei kuitenkaan konvertoitunut suoraan sovelluksen luettavaan muotoon, joten se piti jakaa kokeilupohjaisesti useaan pienempään FeatureCollection-objectin sisältävään tiedostoon, jotta virhe saatiin eliminoitua ja data pystyttiin renderöimään kartalle.

Vahinkopaikan koordinaatit lisätään lomakkeeseen karttapohjaa klikkaamalla. Klikkauksen jälkeen käyttäjälle avautuu ikkuna (kuvio 7), josta hän voi valita OK, jolloin koordinaatit välitetään suoraan lomakkeeseen, tai Cancel, jolloin uuden karttamerkin asettaminen karttapohjaan on mahdollista. Nämä toiminnallisuudet mahdollistavat tarkan tiedonvälityksen ja eliminoivat vahinkoklikkauksista aiheutuvan datan vääristymisen.



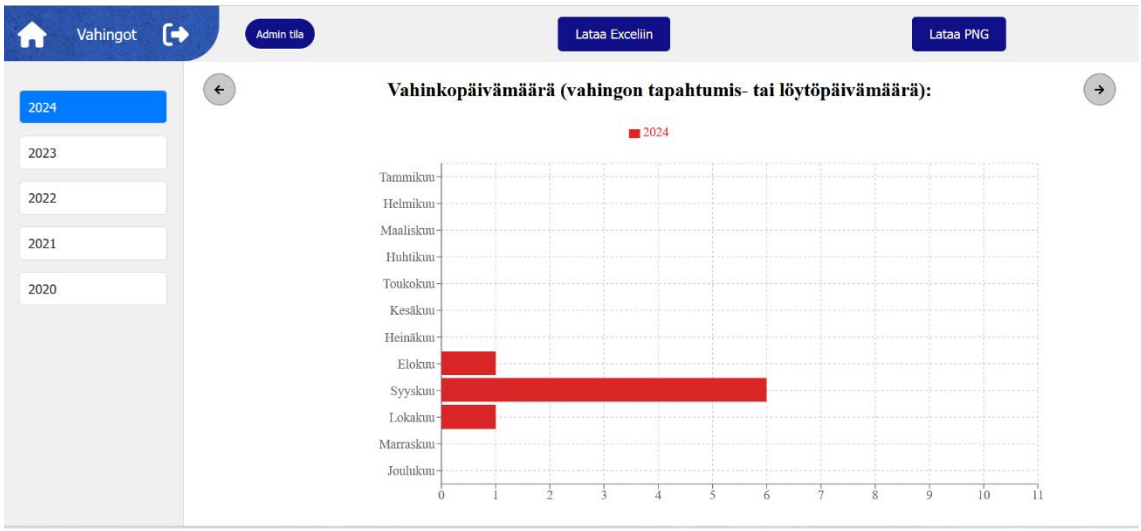
Kuvio 7. Karttapohjan koordinaatti-ikkuna

3.3 Admin-tila ja datanhallinta

Eräänä keskeisenä toiveena oli sovellukseen luotava erillinen admin- eli ylläpitäjätila, jossa tietyt käyttäjät pääsevät kirjautumaan sisään sekä tutkimaan ja käsittelemään kerättyä dataa. Koska aiempina vuosina tallennettu data on kerätty Exceliin ja muutettu siellä graafeihin yleistä esittämistä varten, sovellukseen luotiin toiminnallisuus, joka esittää tietokantaan tallennetun dataan suoraan graafeissa, joista admin voi ladata png-kuvankaappauksia suoraan koneelleen napin painalluksella.

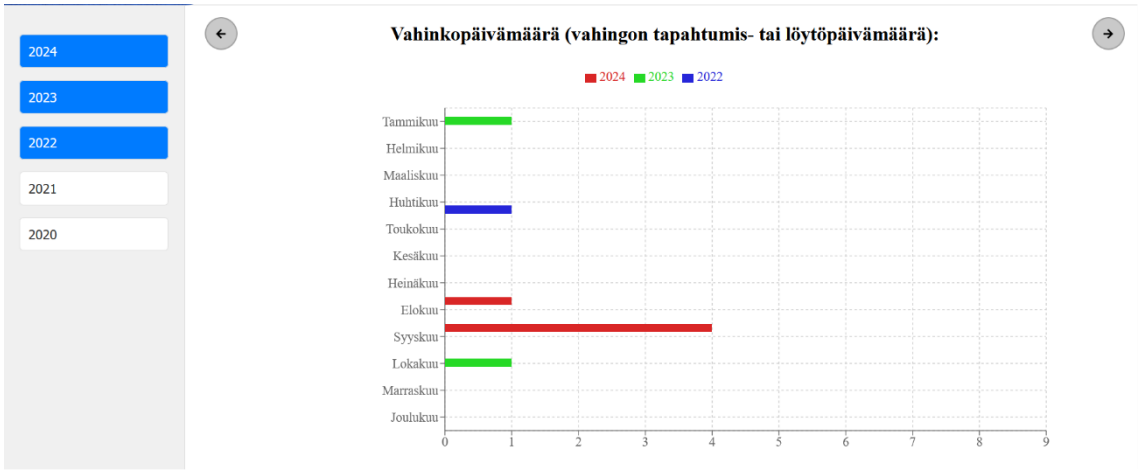
3.3.1 Admin-tilan datakaruselli

Graafeihin voidaan valita dataa vuosi kerrallaan tai useamman vuoden jaksoissa. Graafit muodostavat datakarusellin (kuvio 8), joka esittää yhteenvedon lomakkeen kysymyksiin valituilta vuosilta.



Kuvio 8. Datakarusellin ensimmäinen kaavio vahinkokuukausista

Jotta graafien luettavuus säilyy hyvänä, on käyttäjälle näkyvän graafin maksimiarvo aina viisi numeroa edellä todellisen datan määrää (kuvio 9). Kun ilmoitusten määrä kasvaa, numeroiden väli graafin x-akselilla pienenee. Tämä mahdollistaa graafin säilymisen kuvankaappaukseen tarkoitetulla alueella.

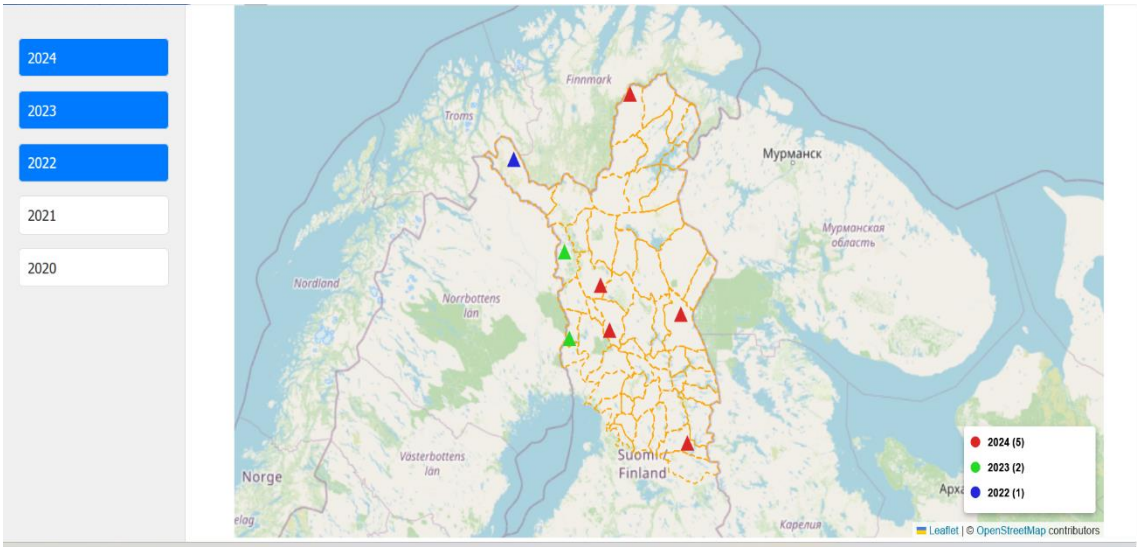


Kuvio 9. Esimerkki useiden vuosien tarkastelusta

Jotta vuosien välisiä eroja on helppo seurata, admin voi valita niitä haluamansa määrän ja haluamassaan järjestyksessä historiakomponentista. Valitut vuodet näkyvät graafissa eri väreillä, mikä tekee datan tulkitsemisesta helpompaa. Kuvion 9 graafista voidaan esimerkiksi havaita joka vuoden aktiivisin vahinkokuukausi, mutta myös se, että vahinkokuukausissa on eroa vuosittain. Jos eri vuosina olisi kirjattu vahinkoja samalle kuukaudelle, eriväriset palkit olisivat kiinni toi-

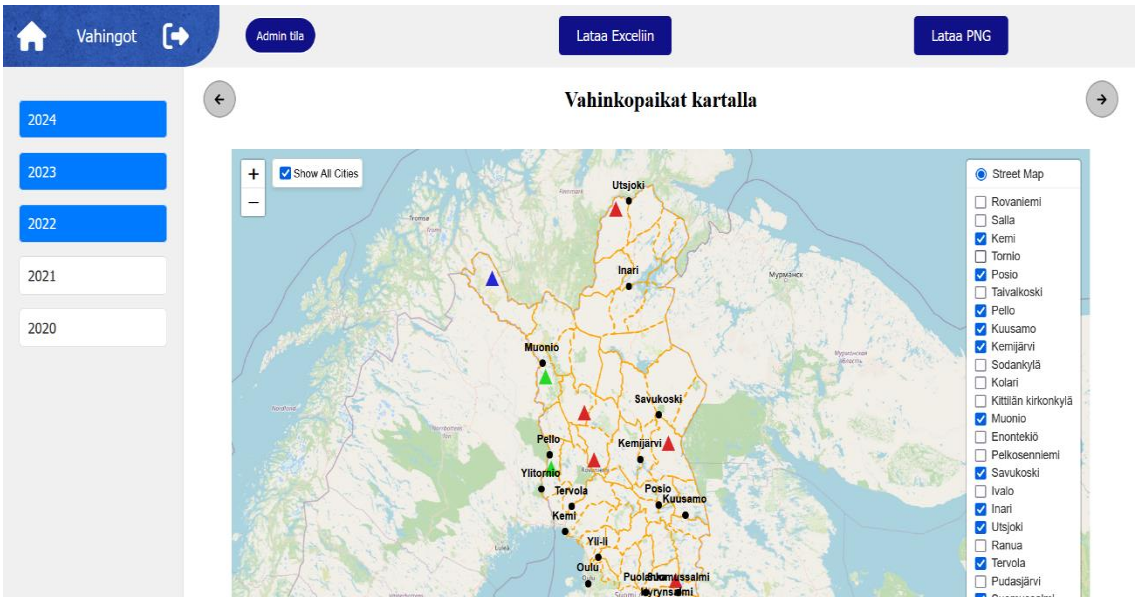
sissaan kuukauden kohdalla. Vuodet päivittyvät historiakomponenttiin automaattisesti ajankulun mukaan vuoteen 2099 asti siten, että viimeisin vuosi on aina ylimpänä.

Koska vahinkopaikkoja haluttiin tarkastella myös karttapohjaan merkittynä, kartta lisättiin admin-tilan datakaruselliin. Tämän avulla admin voi tarkastella dataa samoin kuin graafeista. Kartalla näkyvistä vahinkopaikoista, eli karttaesityksestä (kuvio 10), voidaan ottaa myös kuvankaappauksia.



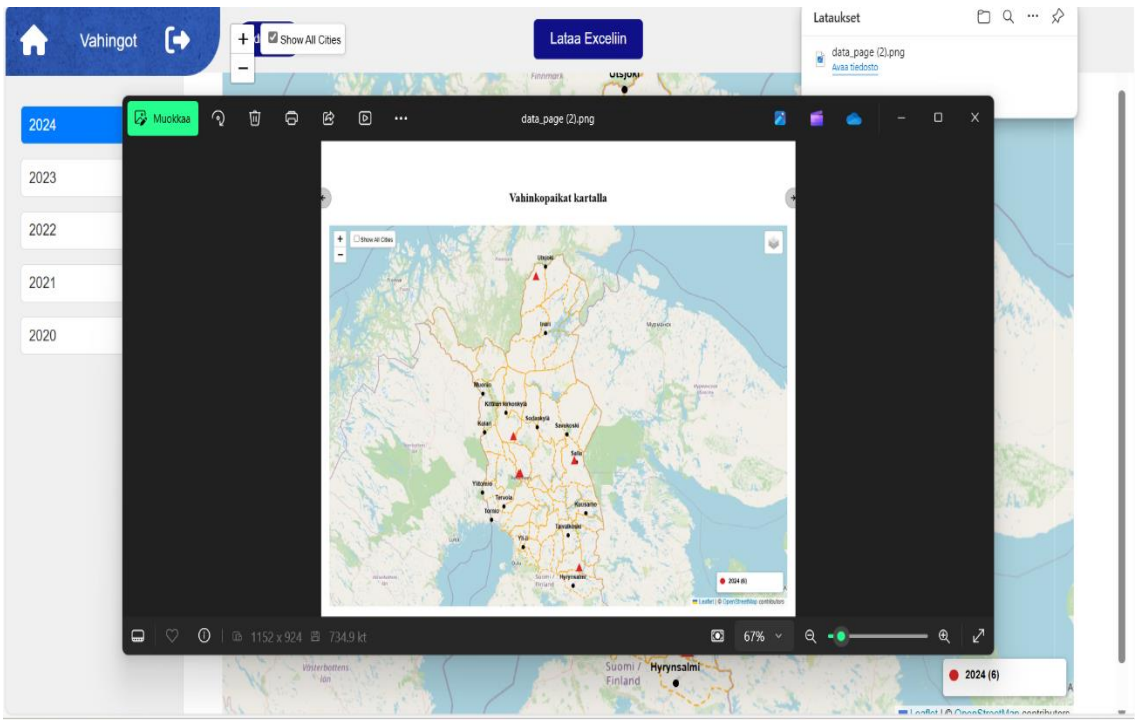
Kuvio 10. Admin-tilan karttapohja

Myös admin-tilan kartta on täysin zoomattava, ja se väri-koodaa eri vuosien vahinkopaikat ja vahinkojen määrän graafien tapaan. Kartan voi lisäksi laittaa näyttämään kaupunkeja, jotka voidaan valita renderöitäväksi joko kaikki kerrallaan tai yksitellen (kuvio 11).



Kuvio 11. Admin-tilan karttapohjan kontrollivalikko ja ominaisuudet

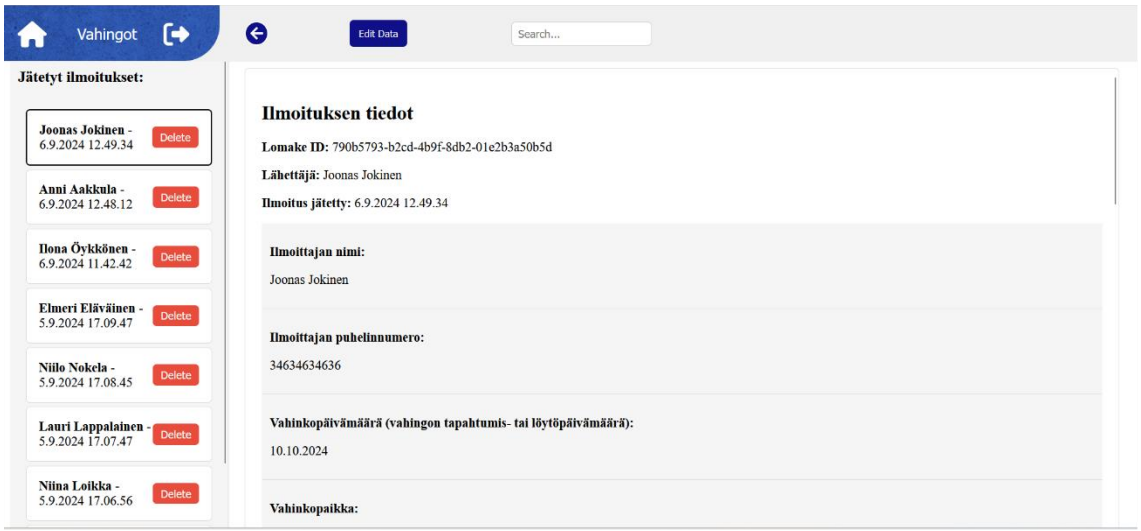
Kuvankaappauksen sisältö perustuu siihen, mitä sovelluksessa on renderöity kullakin hetkellä. Admin voi siten itse päättää, haluaako hän sisällyttää kuvankaappaukseen kaupunkien nimiä vai pelkät karttamerkit, sekä miltä zoom-tasolta kaappaus otetaan (kuvio 12).



Kuvio 12. Esimerkki kartasta otetusta kuvankaappauksesta

3.3.2 Admin-tilan hallinnointinäkymä

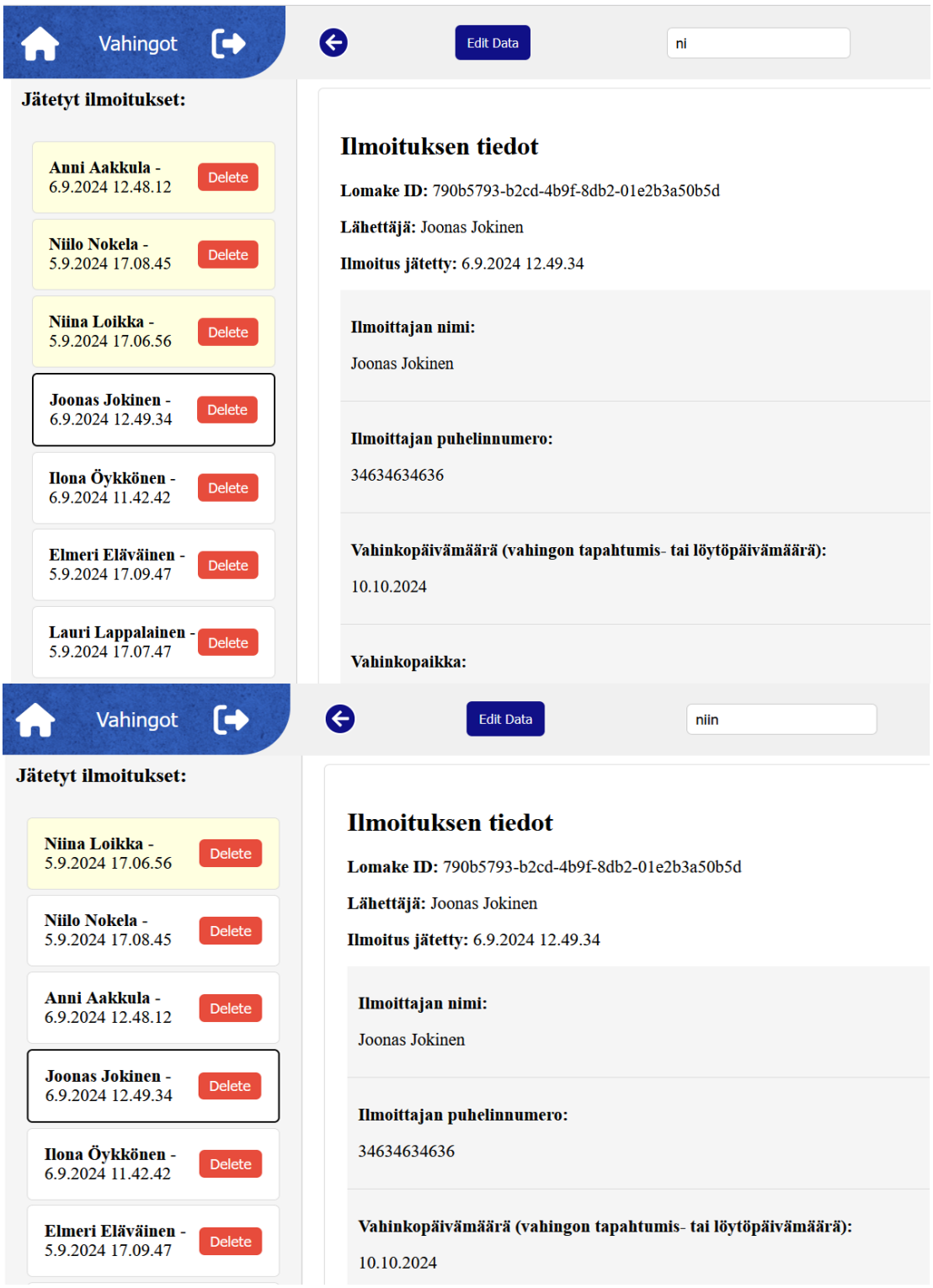
Kolmas sovellukseen toivottu ominaisuus oli jätettyjen ilmoitusten hallinnoiminen. Hallinnoimista varten sovellukseen lisättiin oma sivu, jonne admin pystyi navigoimaan suoraan datakarusellista. Hallintanäkymä (kuvio 13) on automatisoitu siten, että viimeisin ilmoitus näkyy listassa aina ylimpänä.



Kuvio 13. Admin-tilan datanhallinnointinäkymä

Ilmoitusta klikkaamalla adminille avautuvat lomakkeen tiedot, josta käyvät ilmi lomakkeen uniikki ID, lähettäjä ja ilmoituksen jättöpäivämäärä sekä lomakkeeseen annetut vastaukset. Jätetyt ilmoitukset voidaan poistaa delete-napista, jolloin ne poistuvat sekä tietokannasta että jätetyt ilmoitukset -näkymästä.

Jotta datan hallinnointi olisi mahdollisimman vaivatonta, näkymään on lisätty hakukenttä, johon admin voi kirjoittaa lähettäjän nimen tai päivämäärän (kuvio 14) ja rajata näin hakutulosta. Rajaus tarkentuu riippuen siitä, miten tarkkaan tieto hakukenttään kirjoitetaan. Jos kenttään kirjoitetaan esimerkiksi pelkkä n-kirjain, jätetyt ilmoitukset listan kärkeen nousevat korostettuna ne ilmoitukset, joiden lähettäjän etu- tai sukunimessä on n-kirjain. Jos haku puolestaan tehdään koko nimellä, korostettuna näkyvät vain ne ilmoitukset, joiden lähettäjä vastaa hakua.

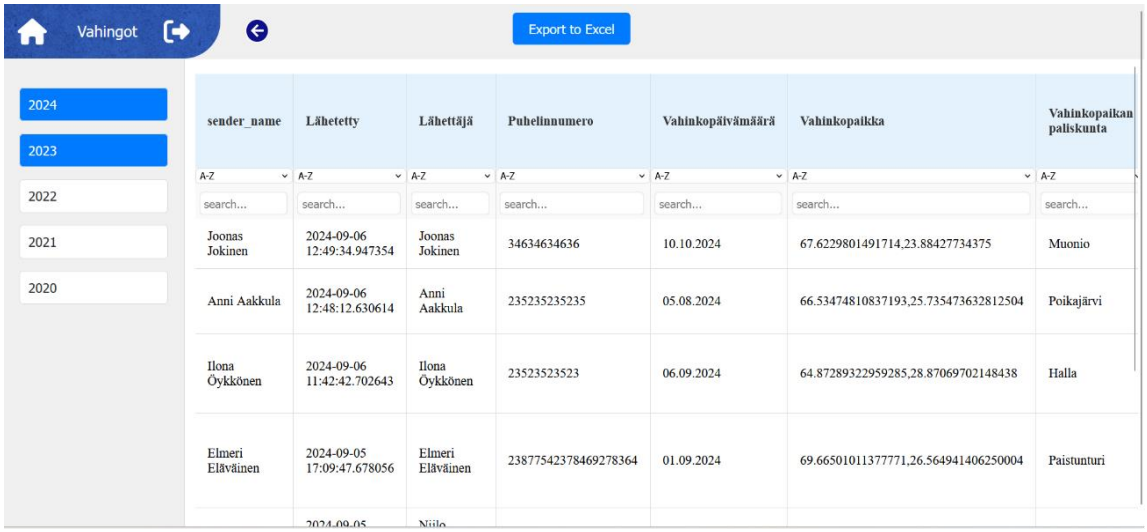


Kuvio 14. Hakukenttätoiminnallisuus admin-tilassa

3.3.3 Admin-tilan lajittelusivu

Koska sovelluksessa esitettävä data halutaan ladata myös Exceliin, lisättiin latamista varten oma sivu ja latausnappi (kuvio 15). Tarkoituksena oli esittää data

jo valmiiksi mahdollisimman samanlaisessa muodossa kuin se esitetään Excelissä. Eräänä toiveena oli, että admin pystyy lajittelemaan lähetettyjä ilmoituksia aakkosjärjestyksessä tai käänteisessä järjestyksessä. Lisäksi jokaisessa sarakkeessa on hakukenttä, joka toimii samalla periaatteella kuin yksittäisten ilmoitusten tarkasteluun liitetty haku kuviossa 14.



Kuvio 15. Näkymä datan lajittelusivusta

Datataulukkoa voidaan liikuttaa sivusuunnassa, jolloin kaikkia kolumneja ja niiden dataa voidaan tarkastella. Admin voi itse päättää, haluaako ladata datan aakkostettuna vai käänteisessä järjestyksessä, sillä Exceliin latautuu juuri sillä hetkellä lajiteltu näkymä. Jos meneillään on haku esimerkiksi yksittäisen henkilön datasta, ja näkyvillä on vain yksi rivi, latautuu data Exceliin viimeisimmän lajittelun mukaan, ruutunäkymästä huolimatta.

4 BACKEND

Taustajärjestelmä (*backend*) on palvelimella toimiva koodi, joka käsittelee asiakasohjelmien lähettämät pyynnöt ja sisältää logiikan, jolla oikea tieto palautetaan asiakasohjelmalle. Asiakasohjelmat (*clients*) ovat järjestelmiä, jotka tekevät pyyntöjä taustajärjestelmälle. Useimmiten nämä ohjelmat ovat selaimia, jotka pyytävät HTML- ja JavaScript-koodia sivustojen näyttämistä varten. Asiakasohjelmia voi kuitenkin olla monenlaisia, kuten mobiilisovelluksia, toisella palvelimella pyöriviä ohjelmia tai älylaitteita, jotka ovat yhteydessä verkkoon. (Codeacademy Team 2024.)

Taustajärjestelmä koostuu kolmesta keskeisestä osasta: palvelimesta, sovelluksesta ja tietokannasta. Palvelin on tietokone, joka vastaanottaa pyynnöt. Sovellus toimii palvelimella ja kuuntelee pyyntöjä, hakee tarvittavat tiedot tietokannasta ja luo vastauksen, joka lähetetään asiakasohjelmalle. Tietokanta puolestaan järjestää ja tallentaa tiedot pysyvästi sovelluksen käyttöön. (Codeacademy Team 2024.)

4.1 PHP

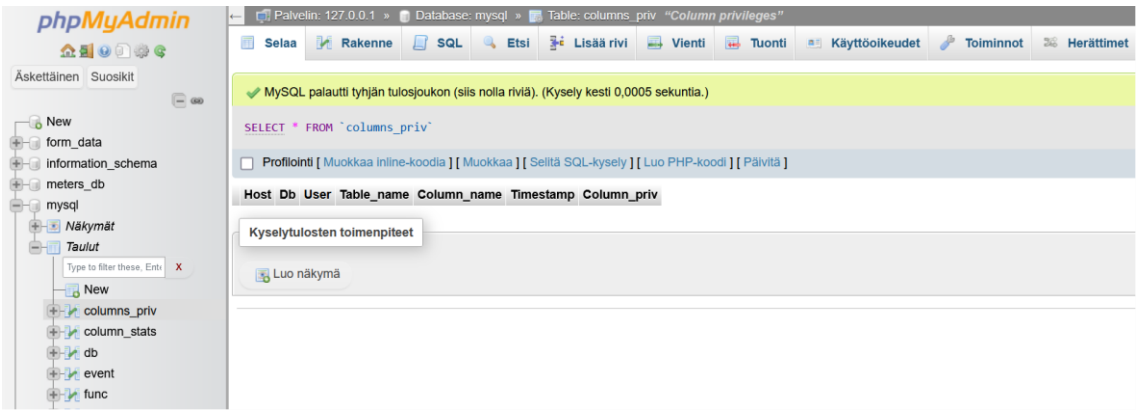
Sovelluksen backend-koodin toteutukseen valittiin ohjelmointikieleksi PHP (Hypertext Preprocessor), joka on ilmainen, monipuolinen ja laajasti käytetty avoimen lähdekoodin ohjelmointikieli. PHP soveltuu erinomaisesti taustajärjestelmien kehittämiseen. (W3Schools 2024.)

Ohjelmointikielen valintaan vaikutti ensisijaisesti se, että sen tuli sopia yhteen Paliskuntain yhdistyksen jo olemassa olevan SQL-pohjaisen tietokannan kanssa. Valintaan vaikuttivat lisäksi kielen helppous, nopeus sekä sen erityinen soveltuminen tarvittavien toiminnallisuuksien toteuttamiseen. PHP:n kyky käsitellä tiedostoja palvelimella sekä kerätä ja hallita lomaketietoja oli keskeisessä roolissa. Lisäksi PHP mahdollistaa käyttäjäoikeuksien hallinnan sekä tietokannan tietojen lisäämisen, poistamisen ja muokkaamisen. Tietoturva on tärkeä osa PHP-toimintoja, sillä se voi salata tietoja ja suojata näin arkaluontoista informaatiota. Näiden

ominaisuuksiensa ansiosta PHP on keskeinen työkalu modernien verkkosovellusten kehittämisessä, ja se on siten oikea valinta opinnäytetyön taustajärjestelmän ohjelmointikieleksi. (Astari 2024; W3Schools 2024.)

Jotta koodi olisi helposti hallittavissa kehityksen aikana, se jaettiin useaan eri tiedostoon, joista jokainen hoitaa omaa tehtäväänsä. Koodin tehtäviin kuuluvat muun muassa tietokannan luominen phpMyAdminiin, yhteyden luominen koodin ja tietokannan välille, lomakkeen lähettäminen tietokantaan, lähetettyjen lomakkeiden hakeminen jättöpäivämäärään perusteella, vastausten hakeminen lomakkeen ID:n perusteella sekä jätettyjen lomakkeiden poistaminen tietokannasta.

PhpMyAdmin on ilmainen, PHP:llä kirjoitettu ohjelmointityökalu, joka hallinnoi MySQL-tietokantoja verkkoselaimen kautta. Se tarjoaa laajan valikoiman toimintoja, kuten tietokantojen, taulukoiden ja käyttäjien hallinnan ja mahdollistaa myös SQL-lauseiden suorittamisen suoraan. Käyttöliittymä (kuvio 16) on suunniteltu helpottamaan tavallisten tietokannan hallintatehtävien suorittamista, samalla kun käyttäjillä on mahdollisuus syventyä yksityiskohtaisempiin SQL-toimintoihin. (phpMyAdmin 2024.)



Kuvio 16. Esimerkki phpMyAdminin käyttöliittymästä

Visuaalinen käyttöliittymä mahdollisti nopean toimivuuden tarkastamisen ja tietojen muokkaamisen kehityksen aikana. Tietojen ja erityisesti jättöpäivämäärän ja vahinkopäivämäärän muokkaus oli välttämätöntä, jotta historiakomponentin ja kuviossa 9 esiteltyjen graafien toimintaa voitiin testata ennen sovelluksen siirtämistä varsinaiselle testipalvelimelle.

4.2 REST API

RESTful API on tehokas tapa yhdistää frontend ja backend hyödyntämällä HTTP-protokollaa, joka mahdollistaa tiedonsiirron käyttäjän ja palvelimen välillä. API (Application Programming Interface) määrittelee säännöt ja protokollat ohjelmistojen väliselle kommunikaatiolle. HTTP (Hypertext Transfer Protocol) toimii protokollana tiedon siirrossa asiakasohjelmien, kuten selaimen, ja palvelimen välillä. REST (Representational State Transfer) on arkkitehtuurinen tyyli, joka hyödyntää HTTP-metodeja tilattomien ja skaalautuvien sovellusten kehittämiseen. (Bigelow & Gillis 2024.)

Endpointit ovat palvelimen osoitteita, joiden kautta voidaan suorittaa toimintoja, kuten tiedon hakua tai tietojen päivittämistä. Jokainen API-pyyntö sisältää URI-osoitteen, otsikkotiedot (*header*), ja tarvittaessa body-osan, joka sisältää pyynnön tiedot. Vasteen kautta palvelin palauttaa pyydetyt tiedot, jotka voivat olla esimerkiksi JSON- tai XML-muodossa. Eri HTTP-metodit määrittävät, minkälaisen toiminnon API-pyyntö suorittaa. (Bigelow & Gillis 2024.)

REST API:n haasteita ovat muun muassa endpointien johdonmukaisuuden ylläpitäminen, sillä endpoint-polkujen tulisi noudattaa yhteisiä web-standardeja. Toinen keskeinen haaste on API-versiointi, jossa on varmistettava, että sisäisesti tai muiden sovellusten kanssa käytetyt URL-osoitteet eivät vanhene. Lisäksi haasteita tuottavat pitkät vasteajat ja liiallinen tiedon määrä, mikä voi kasvattaa kuormaa ja viivästyttää vasteita. (Bigelow & Gillis 2024.)

4.2.1 HTTP-menetelmät ja -protokolla

HTTP-menetelmät ovat keskeisiä työkaluja verkkotoimintojen suorittamisessa. Yleisimmät menetelmät ovat seuraavat:

- **GET**, joka pyytää resurssin kokonaisuudessaan
- **HEAD**, joka hakee vain resurssin metatiedot ilman sisältöä
- **POST**, joka lisää uusia tietoja olemassa oleville verkkosivuille

- **PUT**, joka muokkaa olemassa olevaa resurssia tai luo uuden URI
- **DELETE**, joka poistaa määritellyt resurssit
- **TRACE**, joka näyttää käyttäjille kaikki muutokset tai lisäykset, joita on tehty resurssille
- **OPTIONS**, joka kertoo, mitkä HTTP-menetelmät ovat käytettävissä tietyllä URL-osoitteella
- **CONNECT**, joka muuntaa pyyntöyhteyden läpinäkyväksi TCP/IP-tunneliksi
- **PATCH**, joka mahdollistaa resurssin osittaisen muokkaamisen. (ExtraHop Networks, Inc. 2024.)

Näiden menetelmien avulla käyttäjät voivat vuorovaikuttaa web-resurssien kanssa tehokkaasti. Ne tarjoavat monipuoliset mahdollisuudet resurssien hakemiseen, muokkaamiseen ja poistamiseen verkkoympäristössä. (ExtraHop Networks, Inc. 2024.)

Menetelmät eroavat HTTP-protokollasta siten, että protokolla määrittelee säännöt ja rakenteet tietojen siirtämiselle, kun taas menetelmät ovat käytännön välineitä, joiden avulla erityiset toiminnot toteutetaan. Protokolla luo perustan, kun taas menetelmät tarjoavat erilaisia tapoja vuorovaikuttaa verkkosisältöjen kanssa. (ExtraHop Networks, Inc. 2024.)

4.2.2 HTTP-menetelmät toiminnallisessa osuudessa

Opinnäytetyössä käytettiin erilaisia HTTP-menetelmiä tietojen käsittelyyn. GET-metodia hyödynnettiin tiedon hakemiseen palvelimelta. Tämä menetelmä on optimoitu tietojen lukemiseen ilman, että se vaikuttaa palvelimen tilaan. Esimerkiksi käyttäjän lomakkeen vastauksia haettaessa GET-kutsut mahdollistavat selkeän ja yksinkertaisen tavan hakea tietoja, jotka eivät vaadi palvelimella tehtäviä muutoksia (McKenzie 2023).

POST-metodia puolestaan käytettiin uusien tietojen lähettämiseen ja olemassa olevien tietojen päivittämiseen. POST-kutsu on erityisen hyödyllinen, kun halutaan luoda uusi resurssi tai päivittää monimutkaisempaa tietoa (McKenzie 2023). POSTia käytettiin esimerkiksi lomakkeen tietojen tallentamiseen ja muokkaamiseen. Tämä lähestymistapa tarjoaa joustavuutta, sillä se mahdollistaa suuren tietomäärän lähettämisen palvelimelle.

Vaikka RESTful-arkkitehtuurin mukaan PUT- ja DELETE-menetelmiä olisi voitu käyttää, rajoitettiin menetelmät vain GET- ja POST-menetelmiin. Tämä valinta tehtiin yksinkertaisuuden vuoksi sekä palvelinympäristön rajoitusten vuoksi. Monet palvelinratkaisut voivat olla herkempiä tietyille HTTP-menetelmille, joten yksinkertaisempi lähestymistapa vähentää mahdollisten virheiden riskiä ja helpottaa kehitysprosessia.

Yhteenvetona voidaan todeta, että vaikka GET- ja POST-menetelmät eivät kata kaikkia mahdollisia toiminnallisuuksia, ne tarjoavat silti riittävän joustavuuden sovelluksen tarpeisiin. Tämä valinta yksinkertaisti tietojen käsittelyä ja vähensi virheiden riskiä, mikä puolestaan paransi kehitysprosessin tehokkuutta samalla, kun API:n käyttö pysyi yksinkertaisena ja suoraviivaisena.

5 SOVELLUKSEN TESTAAMINEN

Sovellus haluttiin Paliskuntain yhdistyksen edustajille testattavaksi ennen sen siirtämistä heidän palvelimilleen. Testaamista varten koodit siivottiin varoituksista ja virheviesteistä sekä ylimääräisistä kommenteista. Siivottavat kohdat löytyivät nopeasti komennolla *'npm run build'*. Konsolikomennon tarkoitus on luoda build-hakemisto, joka sisältää sovelluksen tuotantoversion (Facebook, Inc. 2024).

Reactin kehitystila ja tuotantotila on suunniteltu eri vaiheisiin sovelluskehitysprosessissa, ja niillä on merkittäviä eroja toiminnallisuudessa. Kehitystila on tarkoitettu sovelluksen aktiiviseen kehittämiseen ja virheenkorjaukseen. Se tarjoaa kehittäjille reaaliaikaista palautetta, kuten yksityiskohtaisia virheilmoituksia ja varoituksia konsolissa, mikä helpottaa ongelmien löytämistä ja korjaamista. Kehitystilassa käytetään myös "Fast Refresh" -ominaisuutta, joka näyttää koodimuutokset välittömästi selaimessa ilman manuaalista päivitystä. Suorituskykyoptimointeja ei kuitenkaan ole käytössä, mikä voi tehdä sovelluksesta kehityksen aikana hitaamman ja sen pakettikoosta suuremman. (Facebook, Inc. 2024; Murukesan 2023.)

Tuotantotila on tarkoitettu sovelluksen valmiiseen käyttöön loppukäyttäjille. Siinä sovellus optimoidaan suorituskykyä ja tehokkuutta ajatellen. Kehityksessä käytetty ylimääräinen koodi, kuten virheilmoitukset ja varoitukset poistetaan, jolloin JavaScript-paketit pienenevät ja latausajat nopeutuvat. Tuotantotilassa koodi myös minifioidaan ja pakataan, mikä edelleen pienentää sovelluksen kokoa. Lisäksi tuotantotila parantaa turvallisuutta piilottamalla konsolin virheviestit, mikä estää arkaluontoisen tiedon vuotamisen käyttäjille. Sovellus voi myös hyödyntää välimuistia tehokkaammin, mikä parantaa latausnopeuksia erityisesti palaaville käyttäjille. (Facebook, Inc. 2024; Murukesan 2023.)

5.1 Render-pilvipalvelu

Koodien testipalvelimeksi valittiin Render sen tuttuuden, helppouden ja ilmaisen 30 päivää kestävän jaksollisen palvelun tarjoamisen takia. Render on pilvipalvelun tarjoaja, joka auttaa ohjelmistotiimejä julkaiseman tuotteita nopeasti ja missä tahansa mittakaavassa (Render 2024).

Sovelluksen testipalvelimelle saamiseksi sen frontend- ja backend-koodeista tehtiin tuotantopaketti (*build*). Frontend otettiin käyttöön (*deployment*) staattisena websivuna siten, että se on yhteydessä GitHubiin. Näin ollen muutokset koodissa menevät testaukseen reaaliajassa. Staattinen sivusto koostuu sivuista, jotka sisältävät muuttumatonta HTML-, Javascript- tai CSS-koodia. Sisältö pysyy samana kaikille käyttäjille, eikä se tarjoa juurikaan vuorovaikutusmahdollisuuksia. Koska sisältö ei ole käyttäjäkohtaisesti räätälöityä, se voidaan esirenderöidä, mikä parantaa suorituskykyä ja tekee sivustosta kevyen ja nopean yksinkertaisissa käyttötapauksissa. (Li 2024.)

Myös backend-koodi oli yhteydessä palvelun tarjoajaan GitHubin kautta ja se laitettiin testiserverille webservicenä eli palvelurajapintana. Palvelurajapinta on ennalta määritelty menetelmä, jonka avulla eri sovellukset voivat kommunikoida keskenään. Se kuvaa tavan, jolla yksi sovellus voi pyytää tietoa tai palveluita toiselta sovellukselta. (TEPA-Termipankki 2024.)

Palvelurajapinta ei itsessään ole toiminnallinen, vaan se määrittää, kuinka viestinnän tulisi tapahtua. Termiä "palvelurajapinta" käytetään usein virheellisesti kuvaamaan rajapintapalvelua, vaikka palvelurajapinta tarkoittaa menetelmän määrittelyä ja rajapintapalvelu sen toteutusta. Alun perin API (ohjelmointirajapinta) viittasi ohjelmistokomponenttien rajapintoihin, mutta nykyään sitä käytetään myös verkkopalveluiden rajapinnoista, minkä vuoksi termit API ja palvelurajapinta ovat lähes synonyymeja. (TEPA-Termipankki 2024.)

Koska Render tuki vain PostgreSQL-tyyppisiä tietokantoja, piti backend-koodiin tehdä muutoksia ennen docker buildin tekemistä. PostgreSQL-tietokanta eroaa MySQL-tietokannasta siten, että PostgreSQL on monipuolisempi ja laajennettavampi soveltuen erityisesti monimutkaisiin ja suurivolyymisiin tietokantaoperaatioihin. Se tukee mukautettuja tietotyyppejä ja indeksejä sekä mahdollistaa laajemman joustavuuden tietokantarakenteissa. MySQL taas on tunnettu nopeudestaan ja vakaudestaan, ja se on suosittu valinta yksinkertaisempiin verkkosovelluksiin ja sisällönhallintajärjestelmiin, joissa ei tarvita yhtä monipuolisia toiminnallisuuksia kuin PostgreSQL:ssä. (Smallcombe 2023.)

5.2 Docker

Tarvittavien muutosten jälkeen backend-koodi oli valmis Docker Buildille eli rakennusprosessille. Kyseessä on prosessi, jossa luodaan Docker-kuvia Dockerfile-tiedostossa annettujen ohjeiden perusteella. Dockerfile on tekstimuotoinen tiedosto, joka määrittelee kuvan konfiguraation. Rakennusprosessi kokoaa yhteen kaikki tarvittavat komponentit, kuten koodin, riippuvuudet ja ympäristöasetukset, luoden niistä siirrettävän Docker-kuvan. (GeeksforGeeks 2024.)

Docker Buildin vaiheet ovat selkeitä ja tehokkaita ja mahdollistavat sovellusten käyttöönoton eri ympäristöissä. Ensimmäinen vaihe on luoda Dockerfile (kuvio 17), jossa määritellään esimerkiksi FROM-komennolla pohjakuva, joka tässä tapauksessa on php:8.0-apache. COPY-komennolla tiedostot kopioidaan isäntäkoneelta Docker-kuvaan, kuten esimerkissä COPY . /var/www/html/, joka siirtää sovelluksen tiedostot konttiin. RUN-komennolla asennetaan tarvittavat ohjelmistot ja laajennukset rakennuksen aikana; esimerkissä käytetään RUN apt-get update && apt-get install -y libpq-dev && docker-php-ext-install pgsql pdo_pgsql, joka asentaa PostgreSQL-laajennukset PHP:lle. (GeeksforGeeks 2024.)

```
FROM php:8.0-apache

RUN apt-get update && apt-get install -y libpq-dev \
    && docker-php-ext-install pgsql pdo_pgsql

COPY . /var/www/html/

WORKDIR /var/www/html

EXPOSE 80
```

Kuvio 17. Esimerkki Dockerfilen sisällöstä

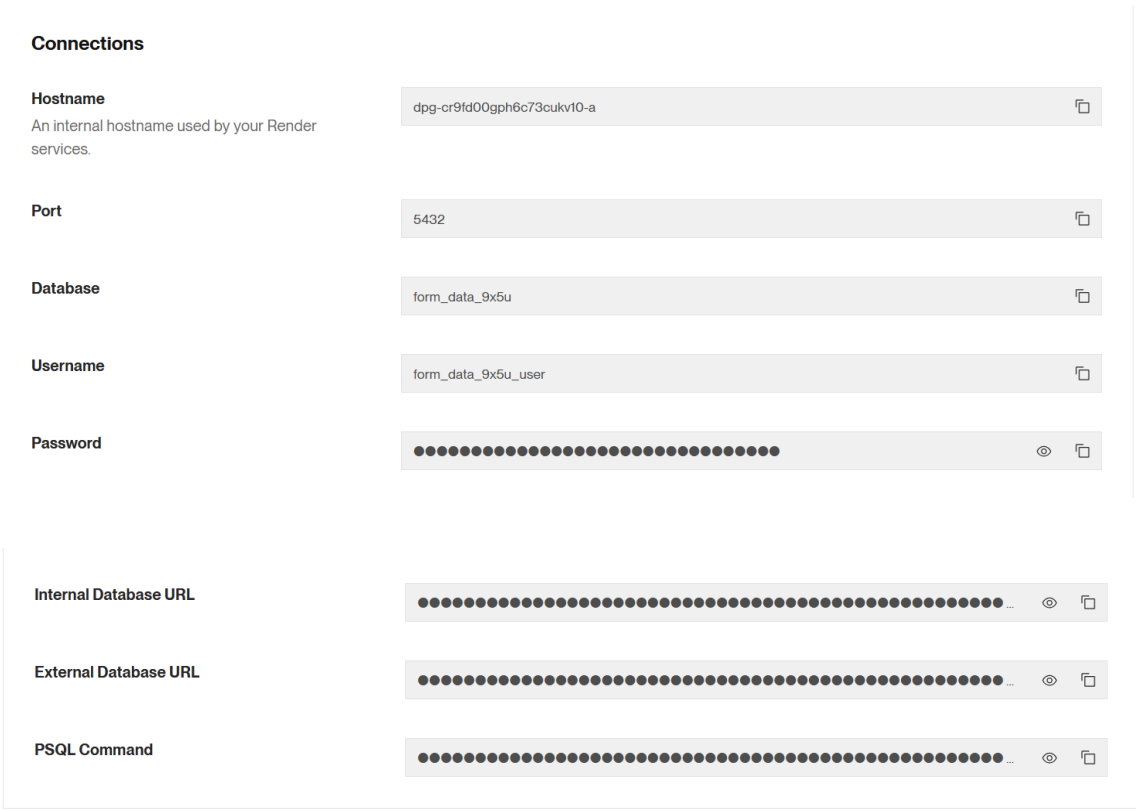
Kun Dockerfile on valmis, kuva rakennetaan komennolla *docker build -t kuvan_nimi .*, jossa -t mahdollistaa kuvan nimeämisen ja piste osoittaa Dockerfile-

tiedoston sijainnin. Tämä komento suorittaa kaikki Dockerfile-ohjeet järjestyksessä ja kokoaa tarvittavat komponentit toimivaksi Docker-kuvaksi. (GeeksforGeeks 2024.)

Rakennettu Docker-kuva voidaan suorittaa komennolla *docker run*, joka käynnistää uuden säilön kuvasta. Tämä prosessi takaa, että sovellus toimii samalla tavalla missä tahansa ympäristössä varmistaen kehityksen, testauksen ja tuotannon yhtenäisyyden. Docker Build automatisoi ja yksinkertaistaa sovelluksen käyttöönoton tehden siitä joustavan ja helposti hallittavan. (GeeksforGeeks 2024.)

5.3 Asiakassovelluksen ja tietokannan kommunikointi

Tietokanta luotiin Renderiin napin painalluksella valitsemalla PostgreSQL, jolloin Render loi automaattisesti yhteysparametrit (kuvio 18). Yhteysparametrit ovat työkaluja tai ohjelmistoja, jotka mahdollistavat asiakasohjelmasovelluksen kommunikoimisen tietokantapalvelimen kanssa. Näitä parametreja tarvitaan, kun asiakasohjelmasovellus haluaa päästä käsiksi tietokantoihin, jotka voivat sijaita paikallisesti, yrityksen palvelimilla tai pilvessä. (McQuillan 2023.)



Kuvio 18. Testipalvelimen tietokannan yhteysparametrit

Yhteyksien konfigurointi on kriittistä, sillä se määrittelee, miten asiakassovellus käyttää tietokantaa ja vaikuttaa sovelluksen suorituskykyyn ja turvallisuuteen. Konfigurointi tarkoittaa prosessia, jossa määritellään tarvittavat tiedot ja asetukset, kuten palvelimen osoite, porttinumero, tietokannan nimi, käyttäjätunnus ja salasana. Nämä tiedot kootaan connection stringiksi eli yhteysmerkkijonoksi, joka mahdollistaa yhteyden avaamisen tietokantapalvelimeen. Tämän jälkeen käytetään erityistä funktiota, joka hyödyntää connection stringiä ja avaa tietokantayhteyden, jolloin backend voi suorittaa kyselyitä ja operaatioita, kuten tietojen hakemista, tallentamista ja päivittämistä. (McQuillan 2023.)

Oikeat käyttöoikeudet ovat myös tärkeitä, sillä ne voivat rajoittaa pääsyä erityisiin tietoihin tai koko tietokantaan. Tietokantayhteys mahdollistaa datan liikkumisen eri solmujen välillä järjestelmässä, mikä vaikuttaa sovelluksen nopeuteen ja luotettavuuteen. Tämän vuoksi on olennaista ymmärtää, miten yhteysparametrit konfiguroidaan tehokkaasti ja miten tietokantayhteys toimii. (McQuillan 2023.)

Tietokannan yhteysparametrit liitettiin backend-koodiin, mikä mahdollisti viestinnän backendin ja tietokannan välillä. Tämän ansiosta backend voi hoitaa tehtäviä, kuten datan tallentamista, lähettämistä sekä tietojen muokkaamista ja käsittelyä. Samalla backend-koodin palvelimen osoite on liitetty frontend-koodiin, jolloin frontend voi kommunikoida backendin kanssa. Tämä vuorovaikutus tapahtui HTTP-pyyntöjen kautta. Kyseessä on luvussa 4.2 esitelty RESTful API -kommunikaatio.

Frontend yhdistettiin backendin API:iin Axios-kirjaston avulla, joka teki HTTP-pyyntöjen lähettämisestä vaivatonta. Axios on lupauspohjainen HTTP-asiakasohjelma, joka toimii sekä node.js:ä että selaimessa. Palvelinpuolella se käyttää Node.js HTTP-moduulia, kun taas selaimessa se hyödyntää XMLHttpRequestejä (Axios 2024.)

Axiosin avulla lähetettiin pyyntöjä sopivilla metodeilla, kuten GET tai POST, ja tarvittavat tiedot, kuten lomakkeen sisältö tai käyttäjätunnus, liitettiin pyyntöihin. Tämä takasi, että backend pystyi käsittelemään pyynnöt oikein. Backendiltä saadut vastaukset käsiteltiin frontendissä, jolloin käyttöliittymä päivitettiin vastaanotettujen tietojen perusteella.

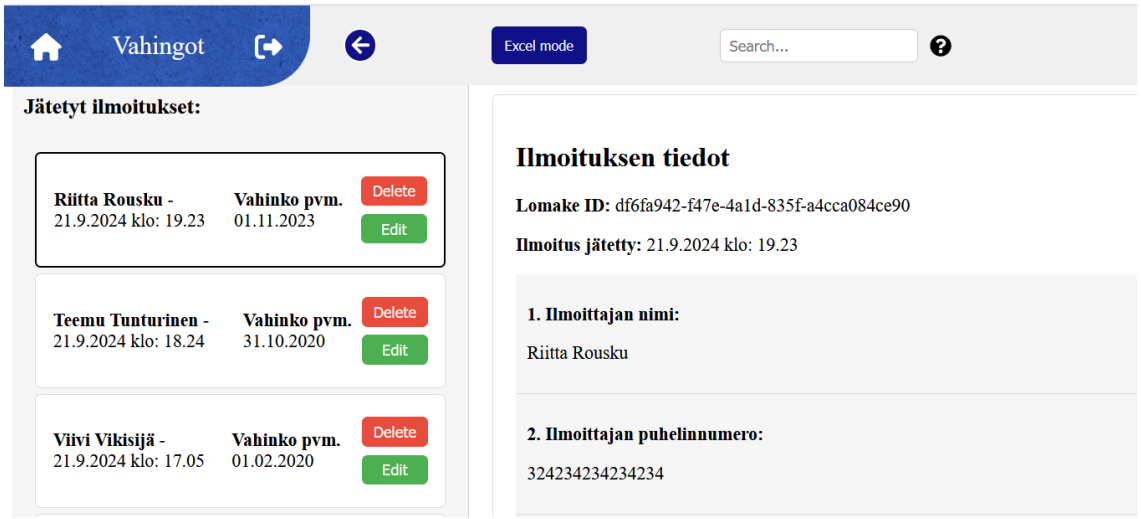
Mikäli haussa tai muussa toiminnassa ilmeni virheitä, käyttäjälle näytettiin tarvittavat ilmoitukset. Tämä prosessi varmisti sujuvan tiedonkulun frontendin ja backendin välillä, mikä takasi sovelluksen tehokkaan ja käyttäjäystävällisen toiminnan.

6 TESTAUKSEN TULOKSET JA NIISTÄ AIHEUTUVAT TOIMENPITEET

Testauksesta saatiin hyvää palautetta, mutta kuten aina sovelluskehityksessä, myös päivitysehdotuksia ja muutostarpeita ilmeni. Arvokas palaute käyttäjiltä on tärkeää kehityksen kannalta, sillä se auttaa suuntaamaan kehitystyötä ja parantamaan sovelluksen käyttökokemusta. Ehdotukset ja havainnot antoivat näkemyksiä siitä, missä voidaan tehdä parannuksia ja lisätä sovelluksen arvoa käyttäjille.

6.1 Admin-tilan uudistukset

Uutena ominaisuutena sovellukseen toivottiin datan muokkausominaisuutta, joka toteutettiin admin-sivulle jo olemassa olevan datan hallinnan yhteyteen (kuvio 19). Lisäksi dataa haluttiin suodattaa vahinkopäivämäärään perusteella lähetyspäivämäärän sijaan, joten ominaisuus lisättiin hakumahdollisuudeksi admin-tilan hakupalkkiin. Hakupalkin viereen lisättiin vihjenappula, jota aktivoi vihjeikkunan hiiren kursorin ollessa sen päällä. Vihjeikkunasta käyttäjä pystyy näkemään, mitä hakusanoja kenttään kannattaa syöttää.



Kuvio 19. Admin-tilan päivitetty käyttöliittymä

Graafien yhteyteen haluttiin pikavalintanäppäimet (kuvio 20), jotka numeroitiin esitettävän lomakkeen kysymyksen mukaan. Koska lähettäjän nimeä tai puhelinnumeroa ei haluttu esittää graafeissa, ei niille annettu pikanäppäimiä, koska niitä esittäviä graafejakaan ei ole olemassa. Lisäksi vahinkopaikkoja esittävää karttaa

päivitettiin toimeksiantajan pyynnöstä siten, että sille lisättiin harmaansävyisiä topografiavaihtoehtoja.

Kuvio 20. Datakarusellin päivitetty käyttöliittymä

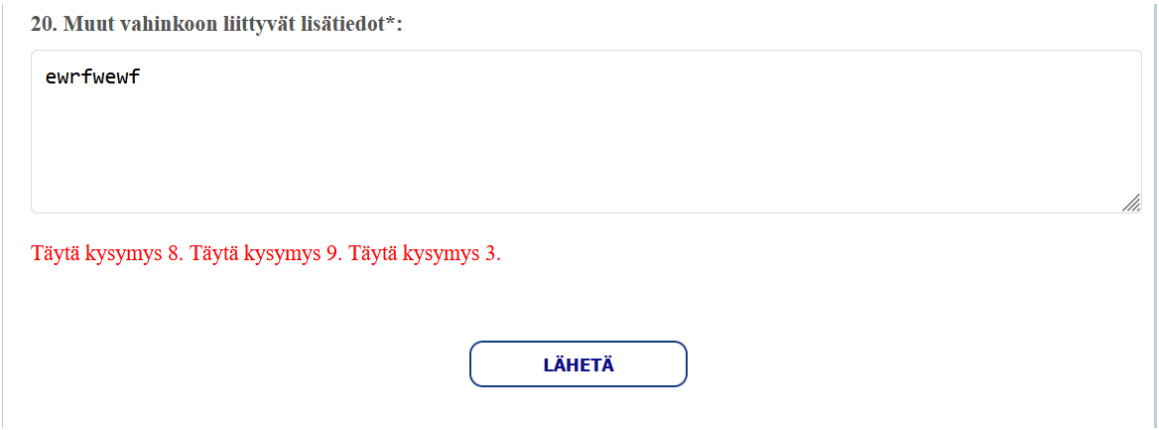
Erisävyiset topografiavaihtoedot mahdollistavat karttamerkkien paremman näkyvyyden karttapohjasta myös ruutukaappauksen yhteydessä, varsinkin kun karttamerkkejä pienenettiin hieman ja kolmiot käännettiin seisomaan kärjellään, jotta ne eivät zoomatessa sekoittuisi karttapohjaan merkittyjen huippujen kanssa. Nämä muutokset varmistavat, että eriväriset karttamerkit eivät renderöidy toistensa päälle ja näkyvät hyvin karttapohjasta myös silloin, kun niitä on paljon lähekkäin ja useita vuosia katsellaan kerralla.

Kuviossa 11 esiteltyjen karttaan liittyvien valikoiden paikka vaihdettiin vasempaan reunaan. Tämä muutos varmistaa, että lista valittavista kaupungeista ei renderöidy avattuna kartan oikeassa alalaidassa olevan vuosittaisten vahinkojen määrästä ilmoittavan legend-komponentin alle.

Käyttöliittymää selkeytettiin myös hieman siirtämällä Excel-latausnappi pelkästään aakkostettavan datan yhteyteen. Joitakin nappeja nimettiin uudestaan kuvaamaan paremmin toiminnallisuutta tai navigaatiota. Kuten kuviosta 20 voidaan havaita, ’Lataa PNG’-nappi on saanut tekstin ’Ota ruutukaappaus’, joka ohjaa kuvaformaatteihin perehtymätöntä käyttäjää selkeämmin.

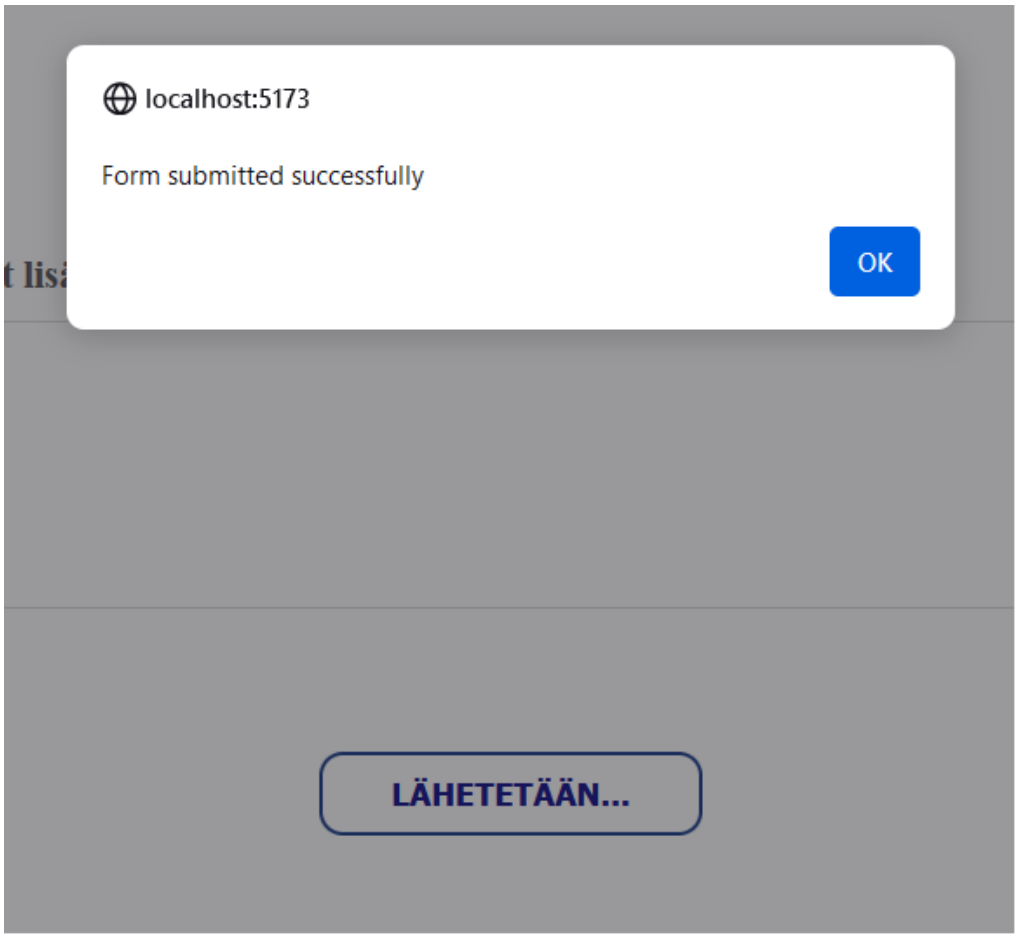
6.2 Vahinkolomakkeen uudet toiminnot

Vahinkolomakkeen suhteen haluttiin varmistaa, että kaikki näkyvät kentät olisivat varmasti täytettyjä, ennen kuin lomakkeen pystyy lähettämään. Lisäksi lomakkeen alalaitaan lisättiin käyttäjää ohjaava huomautus (kuvio 21), jos jokin kentistä on jäänyt täyttämättä.



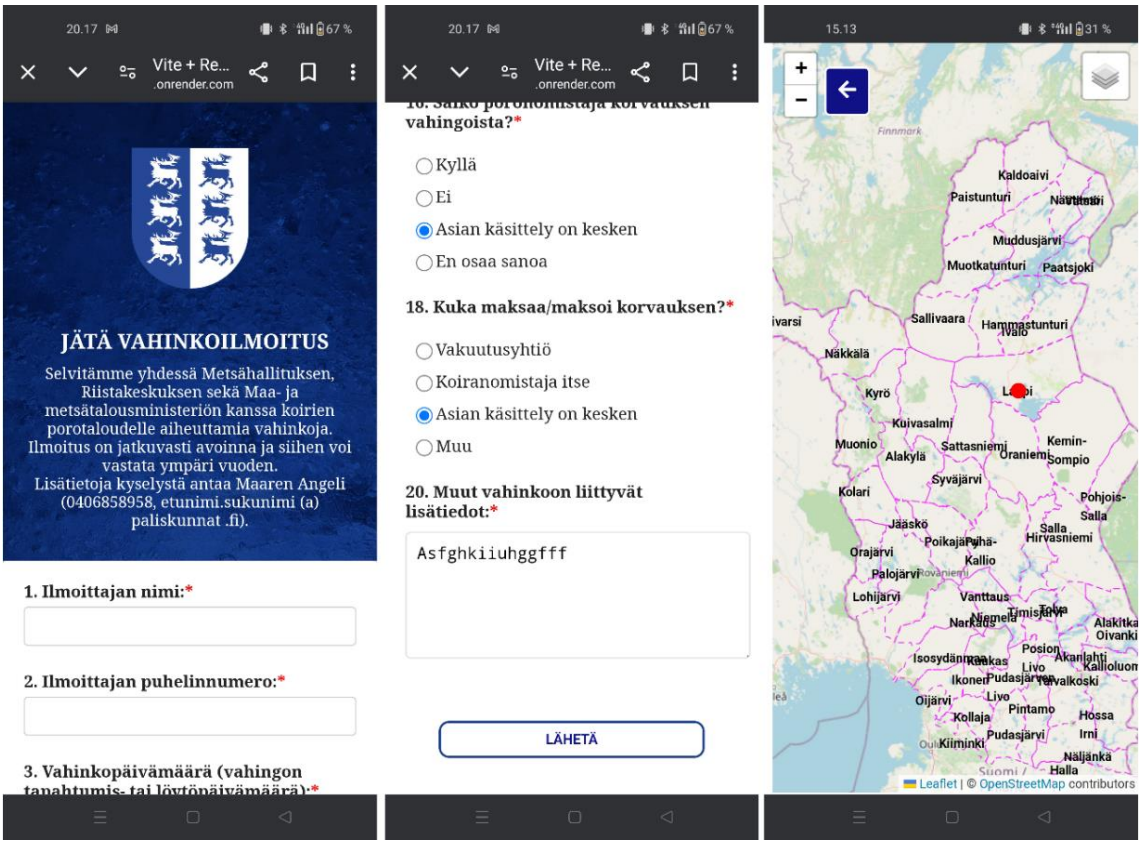
Kuvio 21. Vahinkolomakkeen huomautus vastaamattomista kysymyksistä

Jotta useiden ilmoitusten jättämiseltä vältyttäisiin tilanteissa, joissa käyttäjä painaa lähetä-nappia useita kertoja peräkkäin, lomakkeeseen lisättiin ilmoitus siitä, että sitä lähetetään (kuvio 22). Toiminnallisuus myös lukitsee lähetä-napin siten, ettei uutta ilmoitusta lähetetä, vaikka sitä klikataan useita kertoja lähettämisen aikana. Käyttäjä saa lisäksi ilmoituksen lomakkeen onnistuneesta lähetyksestä. Painamalla OK-nappia lomake tyhjenee ja lähetä-nappi palaa alkuperäiseen tilaansa.



Kuvio 22. Ilmoitus onnistuneesta lähetyksestä

Jotta lomakkeen täyttäminen olisi houkuttelevampaa jo vahinkotilanteen sattuessa, lomake muunnettiin myös mobiililla ja erityisesti puhelimella toimivaksi. Mobiililla lomaketta katsoessa (kuvio 23) tarpeettomat komponentit, kuten kirjautumisnappula, poistuvat näkyvistä, tehden siitä miellyttävämmän käyttää, vaikka muutoin toiminnallisuudet eivät eroa PC-versiosta mitenkään. Mobiilikäyttäjiä ajatellen valintanäppäinten ympärille lisättiin myös lisää tilaa, jotta valintojen raksiminen sormella olisi helpompaa myös suuremman käden koon omaaville henkilöille.



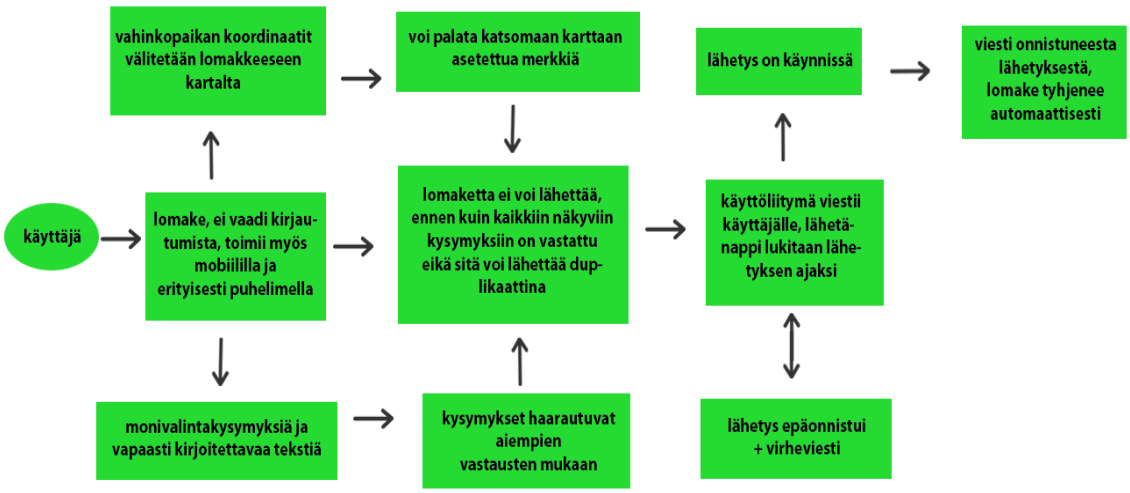
Kuvio 23. Lomakkeen ulkoasu ja keskeiset toiminnot puhelimella katsottuna

Mobiili-toiminnallisuuden lisäksi lomakkeen yhteyteen lisättiin mahdollisuus palata katsomaan karttaan asetettua merkkiä, mitä alkuperäisessä versiossa ei ollut. Tämä parantaa käytettävyyttä, sillä käyttäjä voi palata varmistamaan paliskunnan tai vahinkokunnan nimen ilman, että epävarmuutta jo asetetun karttamerkin sijainnista pääsee syntymään.

6.3 Sovelluksen lopullinen rakenne ja toiminnallisuudet

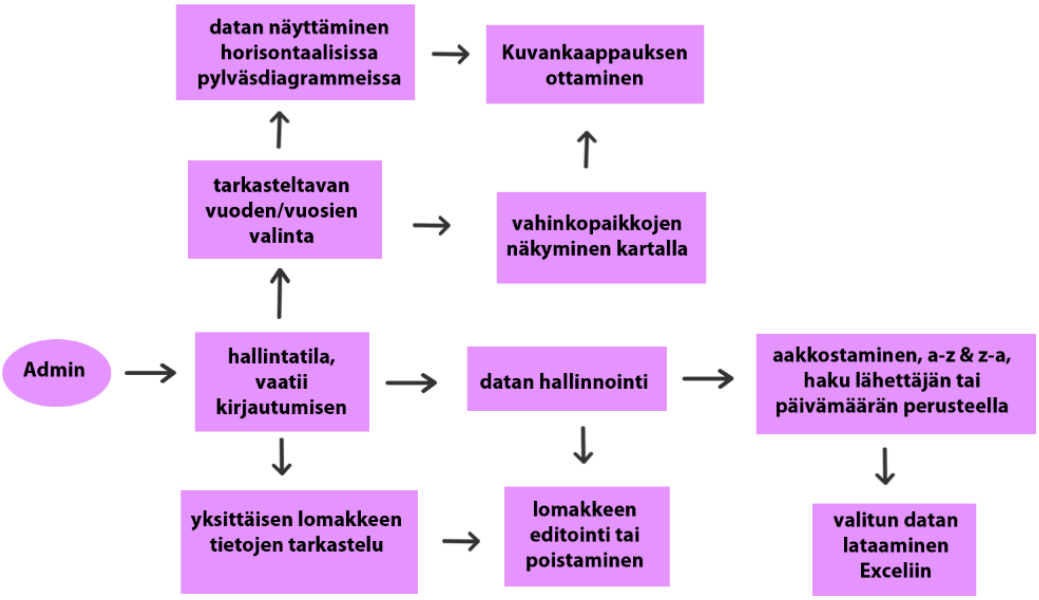
Sovelluksen lopullinen versio noudattaa alkuperäistä jakoa admin-osuuteen ja vahinkolomakkeeseen, mutta tarjoaa käyttäjäystävällisemmän ja parannetun käyttäjäkokemuksen, joka on syntynyt testauksen eri vaiheissa saadun palautteen perusteella. Sekä admin-osio että lomakeosio on päivitetty vastaamaan käyttäjien tarpeita ja parantamaan toiminnallisuutta.

Lomakkeen päivitetyt toiminnallisuudet (kuvio 24) tarjoavat käyttäjille sujuvampia ja tehokkaampia tapoja syöttää tietoja, jolloin prosessi on nopeampi ja intuitiivisempi. Parannuksia ovat muun muassa selkeämpi käyttöliittymä, reaaliaikaiset virheilmoitukset ja ohjeet, jotka auttavat käyttäjiä täyttämään lomakkeen oikein.



Kuvio 24. Lomakkeen viimeistellyt toiminnallisuudet

Adminille suunnattu kokonaisuus (kuvio 25) nopeuttaa tietojen käsittelyä ja hallintaa sekä sisältää näitä nopeuttavia työkaluja, kuten kuvankaappausnapin ja erilaisia karttapohjia, jotka toteutettiin ylimääräisenä toimeksiantona.



Kuvio 25. Admin-puolen lopulliset toiminnallisuudet

Yhdessä nämä toiminnallisuudet muodostavat yhtenäisen ja helppokäyttöisen sovelluksen. Sovellus on räätälöity tarkasti vastaamaan toimeksiantajan tarpeita ja vaatimuksia. Sen pääasiallisena tarkoituksena on helpottaa porotalouden digitaalista arkea. Tämän saavutuksen myötä käyttäjät voivat nauttia sujuvammasta ja tehokkaammasta työskentelystä.

7 POHDINTA

Opinnäytetyön tarkoitus oli suunnitella ja toteuttaa fullstack-menetelmällä web-sovellus, joka on helppo ja miellyttävä käyttää sekä edistää parhaimmillaan porotalouden digitaalisen arjen sujuvuutta. Toteutuksen suunnittelussa ja sen aikana kiinnitettiin huomiota saavutettavuuteen, esteettisyyteen ja kehitysprosessin aikana muuttuviin tarpeisiin sovelluksen sisältöön ja rakenteeseen liittyen.

Suunnittelu ja toteutus olivat pääosin tarvepohjaista ja hoidettiin yhteistyössä Paliskuntain yhdistyksen toimihenkilöiden kanssa. Tarpeisiin kuuluivat vahinkopaikan merkitseminen karttapohjaan, datan esittäminen graafeissa, sen lataaminen Exceliin sekä datan hallinta ja suodattaminen sovelluksen sisällä.

Sovelluksen suunnittelu ja toteutus oli hyvin mielenkiintoinen, mutta haastava prosessi, joka antoi mahdollisuuden uusien ohjelmistojen (QGIS) ja koodikielten (PHP & GeoJson) opetteluun sekä kehitti ongelmanratkaisutaitojani ja insinöörimäistä ajattelua. Tärkeintä sovelluksen onnistuneessa toteutuksessa oli mielestäni työjärjestyksen hahmottaminen ja joustavuus, sillä kehityksen edetessä oli vastattava myös muutostarpeisiin, kuten uusien ominaisuuksien lisäämiseen.

Voinkin todeta, että vaikka suunnitelmallisuus on eräs sovelluskehityksen tärkeimmistä kulmakivistä, on joustavuus ja nopea päätöksenteko vielä tärkeämpää. Joustavuus mahdollistaa tuotteen hiomisen asiakasta miellyttävään lopputulokseen, kun taas nopea päätöksenteko helpottaa aikataulussa pysymistä. On kuitenkin hyvä huomata, että nopea päätöksenteko vaatii hyvää hahmotuskykyä sovelluksen rakenteeseen, toiminnallisuuksiin ja koodin sisäisiin yhteyksiin liittyen. Onnistuin edellä mainitussa mielestäni erinomaisesti, varsinkin kun ottaa huomioon sovelluksen erilaiset toiminnot ja laajuuden.

Kokonaisuudessaan opinnäytetyön toteutus on ollut opettavainen ja laaja prosessi, joka on auttanut minua kehittymään ohjelmoijana ja tukenut kasvamistani ICT-asiantuntijana. Erityisesti koen opinnäytetyön toiminnallisen osuuden kehittäneen insinöörimäistä ajatteluani, samalla kun olen voinut syventää osaamistani myös UI- ja UX-suunnitteluun liittyen.

Opinnäytetyön tuloksena syntyi suunnitelman mukainen, alkuperäistä laajempi työsovellus, joka on räätälöity toimeksiantajan tarpeisiin. Sovelluksen rakenne on kuitenkin muunneltavissa myös muihin vastaaviin projekteihin, kuten petojen aiheuttamien vahinkojen tai porokolareiden tapahtumapaikkojen kirjaamiseen. Jatkokehitysmahdollisuutena voisi olla kaikkien näkymien laajentaminen mobiililaitteille sekä aiempien vuosien datan lataaminen sovellukseen esimerkiksi importtoiminnallisuuden avulla.

LÄHTEET

Agafonkin, V. 2024. Leaftet an open-source JavaScript library for mobile-friendly interactive maps. Viitattu 7.9.2024 https://leafletjs.com/.

Aluehallintavirasto 2024. WSAG 2.1: lain vaatimukset. Viitattu 25.9.2024 https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/wcag-2-1/.

Astari, S. 2024. What is PHP: Understanding the Scripting Language. Hostinger Tutorials 1.3.2024. Viitattu 23.9.2024 https://www.hostinger.com/tutorials/what-is-php/.

Axios 2024. Getting started. Viitattu 23.9.2024 https://axios-http.com/docs/intro.

Bigelow, J. & Gillis, A. 2024. RESTful API. TechTarget toukokuu 2024. Viitattu 5.10.2024 https://www.techtarget.com/searchapparchitecture/definition/RESTful-API.

Codecademy Team 2024. Back-End Web Architecture. Codecademy. Viitattu 22.9.2024 https://www.codecademy.com/article/back-end-architecture.

ExtraHop Networks, Inc. 2024. Hypertext Transfer (HTTP) Protocol. Viitattu 13.9.2024 https://www.extrahop.com/resources/protocols/http.

Facebook, Inc. 2024. Deployment. Viitattu 8.9.2024 https://create-react-app.dev/docs/deployment/.

GeeksforGeeks 2024. What is Docker Build? GeeksforGeeks 7.5.2024. Viitattu 12.9.2024 https://www.geeksforgeeks.org/docker-build/.

Hyttinen, M. 2023. Adobe XD vai Figma? Mitä eroa ohjelmistoilla ja mihin ne parhaiten soveltuvat? Corellia 9.3.2023. Viitattu 12.9.2024 https://corellia.fi/adobe-xd-vai-figma-mita-eroa-ohjelmistoilla-ja-mihin-ne-parhaiten-soveltuvat/.

Koch, R. 2024. Accessibility in UI Design: Best Practices for Inclusive Interfaces. Resonio 26.1.2024. Viitattu 4.7.2024 https://www.resonio.com/blog/accessibility-in-ui-design/.

Lamprecht, E. 2023. The Difference Between UX and UI Design: A Beginner's Guide. CareerFoundry 2.6.2023. Viitattu 22.9.2024 https://careerfoundry.com/en/blog/ux-design/the-difference-between-ux-and-ui-design-a-laymans-guide/.

Li, J. 2024. What is a Static Website. Hygraph 9.8.2024. Viitattu 11.9.2024 https://hygraph.com/blog/what-is-a-static-website.

Lodha, K. 2020. Everything about GeoJson. Medium 22.11.2020. Viitattu 5.10.2024 https://medium.com/random-gis-talks/everything-about-geojson-67ddc6eda11e.

McKenzie, C. 2023. HTTP request methods explained. The Server Side, TechTarget 29.8.2023. Viitattu 24.9.2024 https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/HTTP-methods.

McQuillan, R. 2023. What are Database Connections? BudiBase 5.9.2023. Viitattu 12.9.2024 https://budibase.com/blog/data/database-connections/.

Microsoft 2024. What is Typescript? Viitattu 5.10.2024 https://www.typescriptlang.org/.

Milbergs, K. 2023. Accessibility Levels: WCAG Compliance A vs. AA vs. AAA. Accessibly 16.11.2023. Viitattu 4.9.2024 https://accessiblyapp.com/blog/accessibility-levels-wcag-a-aa-aaa/.

Mohsin, H. 2024. An Introduction To React Programming Language. Invozone 25.6.2024. Viitattu 7.9.2024 https://invozone.com/blog/an-introduction-to-react-programming-language/.

Murukesan, R. 2023. Understanding React Production vs. Development Modes. Medium 6.9.2023. Viitattu 8.9.2024 https://rameshmurukesan.medium.com/understanding-react-production-vs-development-modes-9dff83cf33d5.

PHPMyAdmin 2024. Bringing MySQL to the web. Viitattu 8.9.2024 https://www.phpmyadmin.net/.

Palino, G. & Sparks, E. 2021. QGIS: An introduction to open-source geographic information system (Publication 3269, MASGP-18-038). Mississippi State University, Coastal and Marine Extension Program 22.10.2021. Viitattu 8.9.2024 http://extension.msstate.edu/publications/qgis-introduction-open-source-geographic-information-system.

Render 2024. We build accessible and reliable cloud infrastructure. Viitattu 8.9.2024 https://render.com/about.

Smallcombe, M. 2023. PostgreSQL vs MySQL: The Critical Differences. Integrate.io 20.9.2023. Viitattu 11.9.2024 https://www.integrate.io/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/.

TEPA-Termipankki 2024. Palvelurajapinta. Viitattu 11.9.2024 https://termipankki.fi/tepa/fi/haku/palvelurajapinta.

W3Schools 2024. PHP Introduction. Viitattu 4.9.2024 https://www.w3schools.com/php/php_intro.asp.

WebAIM 2024. Contrast Checker. Viitattu 5.10.2024 https://webaim.org/resources/contrastchecker/.