LAB University of Applied Sciences

Tiedostohallinnan käytettävyyden parantaminen toiminnanohjausjärjestelmässä

LAB-ammattikorkeakoulu
Tradenomi (AMK), Tietojenkäsittely
2022
Juho Alenius

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Alenius, Juho	Opinnäytetyö, AMK	2022
	Sivumäärä	
	26	

Työn nimi

Tiedostohallinnan käytettävyyden parantaminen toiminnanohjausjärjestelmässä

Tutkinto ja koulutusala

Tradenomi (AMK), Tietojenkäsittely

Toimeksiantajan nimi, titteli ja organisaatio

Pekka Savela, Hallituksen puheenjohtaja, InSure Group Oy

Tiivistelmä

Opinnäytetyön tavoitteena oli parantaa toimeksiantajan toiminnanohjausjärjestelmän tiedostojenhallintaa. Toiminnanohjausjärjestelmässä oli ongelmia käytettävyydessä liittyen tiedostojen lataamiseen järjestelmään ja tiedostojen arkistointiin.

Teoriaosuudessa tutustutaan käytettävyyteen, järjestelmässä käytettyyn suunnittelumalliin ja teknologioihin. Tietoperustan jälkeen käydään läpi toimeksiantajan toiminnanohjausjärjestelmän toimintaa ja järjestelmän tiedostohallintaan liittyviä ongelmia.

Toiminnallisessa osuudessa määritellään suunnitelma ongelmien korjaamiseksi. Suunnitelmassa käytiin läpi järjestelmän ongelmakohdat ja määritettiin ratkaisut kyseisiin ongelmiin. Työn toteutuksessa käydään läpi tehdyt muutokset järjestelmään.

Työn tuloksena järjestelmän tiedostojenhallintaan saatiin parannettua. Ongelmat ratkaistiin suunnitellulla tavalla. Lopuksi esiteltiin vielä mahdolliset jatkokehitys ideat uusia ominaisuuksia koskien.

Asiasanat

Toiminnanohjausjärjestelmä, tiedostojenhallinta, käytettävyys, Laravel

Abstract

Author(s)	Type of Publication	Published
Alenius, Juho	Thesis, UAS	2022
	Number of Pages	
	26	

Title of Publication

Improving usability of ERP system's file management

Degree and field of study

Bachelor of Business Administration, Information Technology

Name, title and organisation of the client

Pekka Savela, Chair of the board, InSure Group Oy.

Abstract

The goal of thesis was to improve commissioner's ERP system's file management. The ERP system had usability problems concerning systems file uploads and file archiving.

Usability, system's design pattern and technologies are introduced in theoretical part of the thesis. After knowledge base, commissioner's ERP system and problems with the system are introduced.

Functional part of the thesis starts with defining a plan to fix the problems in the ERP system. Plan consists of problems with the system and fix to these problems. Execution of the work consists of made changes in the system.

The goal of improving system's file management was succeeded. Problems were fixed as planned. Possible further development ideas were introduced at the end of the thesis.

Keywords

ERP system, file management, usability, Laravel

Sisällys

1 Johdanto

1.1 Tausta

Toiminnanohjausjärjestelmät tai ERP-järjestelmät ovat yrityksien kokonaisvaltaisia ohjelmistoja, joilla ohjataan ja hallitaan yrityksen toimintaa. Toiminnanohjausjärjestelmät koostuvat eri osioista, joilla hallitaan yrityksen eri osa-alueita. Eri osa-alueita ovat esimerkiksi tilaustenhallinta, varastonhallinta ja kirjanpito. Toiminnanohjausjärjestelmän osiot käyttävät kaikki samaa tietokantaa, jolloin sama ajankohtainen tieto on käytössä kaikissa järjestelmän osissa. (Logistiikan Maailma.)

Markkinoilla on paljon erilaisia toiminnanohjausjärjestelmiä, joista yritykset voivat valita haluamansa. Toiminnanohjausjärjestelmää hankkiessa ei kannata vertailla järjestelmiä suosion mukaan, vaan tarkastella mitä ominaisuuksia järjestelmässä tulisi olla, jotta se parhaiten palvelisi yrityksen tarpeita. (Logistiikan Maailma.)

Opinnäytetyön toimeksiantajana toimii InSure Group Oy. InSure Group Oy on vuonna 2007 perustettu vakuutusalan yritys, joka tarjoaa erilaisia tekniikkaturvia eri ajoneuvoille ja kodinkoneille. Tekniikkaturvat kattavat ajoneuvojen ja kodinkoneiden eri komponentteja yllättävissä ja ennalta arvaamattomissa rikkoutumisissa.

Toimeksiantajalla on käytössä oma yrityksen sisällä rakennettu toiminnanohjausjärjestelmä. Toimeksiantaja päätti rakentaa oman toiminnanohjausjärjestelmän, koska markkinoilta ei löytynyt toimeksiantajan tarpeisiin sopivaa valmista ratkaisua.

1.2 Työn tavoite ja rajaus

Tässä toiminallisessa opinnäytetyössä on tarkoitus parantaa toimeksiantajan toiminnanohjausjärjestelmän tiedostojenhallintaa. Toimeksiantajan järjestelmässä on ongelmia käytettävyydessä liittyen tiedostojen lataamiseen ja arkistointiin.

Opinnäytetyö rajattiin koskemaan vain järjestelmän ajoneuvotilauksien tiedostojenhallintaa. Järjestelmässä hallitaan myös erilaisia tiedostoja liittyen kodinkonetilauksiin ja vahinkojenkäsittelyyn, mutta nämä muutokset järjestelmään tulevat koskemaan vain ajoneuvopuolta.

1.3 Opinnäytetyön rakenne

Opinnäytetyössä käydään ensin läpi teoriaa liittyen käytettävyyteen, järjestelmän suunnittelumalliin ja järjestelmässä käytettyihin teknologioihin. Sen jälkeen tutustutaan tarkemmin toimeksiantajan käyttämään toiminnanohjausjärjestelmän toimintaan ja sen tiedostonhallintaan liittyviin ongelmiin.

Toiminallisessa osuudessa käydään ensin läpi työn suunnittelu, missä määritetään miten järjestelmää tullaan parantamaan. Suunnittelun jälkeen käydään läpi itse työn toteutus, eli mitä muutoksia järjestelmään tehtiin.

2 Käytettävyys

2.1 Käytettävyyden määrittäminen

Käytettävyys on osa käyttäjäkokemusta, jossa määritetään kuinka helppoa jonkin tuotteen tai palvelun käyttäminen on. Jos tuote on sekava ja käyttäjä joutuu nähdä paljon vaivaa tehtävän suorittamiseen, on käytettävyys heikkoa. Hyvä käytettävyys parantaa tehokkuutta ja alentaa mahdollisuuksia virheiden tekoon.

Jakob Nielsen (2012) määrittää käytettävyyden 5 laadulliseen osaan:

- opittavuus
- tehokkuus
- muistettavuus
- virheettömyys
- miellyttävyys.

Opittavuudella mitataan, kuinka helposti käyttäjä suorittaa tehtävän käyttäessään tuotetta ensimmäisen kerran. Mitä selkeämpi ohjaus tuotteessa on tehtävän suorittamiseen, sitä nopeampaa se on oppia.

Tehokkuudella mitataan, kuinka kauan tuotteeseen tutustuneella käyttäjällä kuluu aikaa tehtävän suorittamiseen.

Muistettavuudella mitataan, kuinka nopeasti käyttäjä saa saman tehokkuuden takaisin palatessaan käyttämään tuotetta pidemmän ajanjakson jälkeen.

Virheettömyydellä mitataan, kuinka paljon virheitä käyttäjä tekee tuotetta käyttäessään, millaisia virheitä käyttäjä tekee, huomaako käyttäjä virheet ja miten käyttäjä saa kumottua virheet.

Miellyttävyydellä mitataan, yleisesti kuinka miellyttävää tuotetta on käyttää.

2.2 Käytettävyyden tärkeys

Hyvä käytettävyys ei yleensä käy käyttäjien mielessä tuotetta käyttäessä. Kun kaikki toimii niin kuin pitää ja ilman suurempia ongelmia, on käytettävyyden toteutuksessa onnistuttu. Ongelmat tuotteen käytettävyydessä sen sijaan käyttäjä huomaa. Kun tehtävän suorittaminen on vaikeaa tai epäselvää turhauttaa se käyttäjää. (Niemelä.)

Jos käyttäjä on asiakas, nostaa se todennäköisyyttä tuotteen vaihtamista kilpailijan tarjoamaan tuotteeseen. Työpaikalla taas ongelmat käytettävyydessä syövät aikaa itse

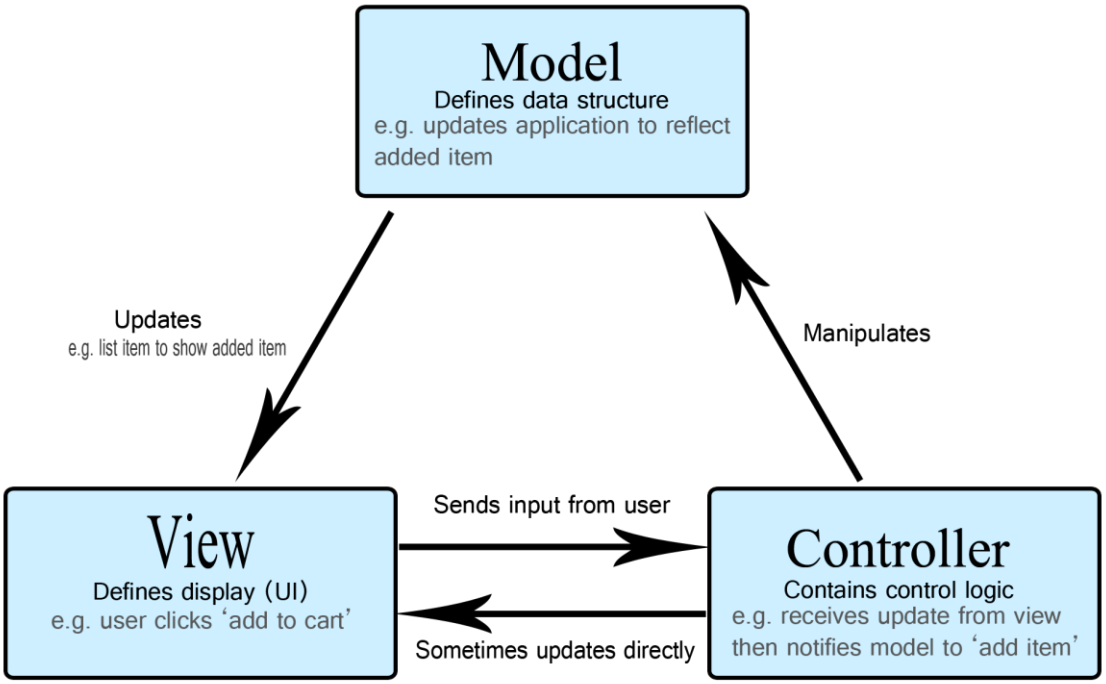
työnteolta. Työntekijän tehokkuus laskee, kun aikaa kuluu tuotteen toiminnan tutkimiseen ja virheiden korjaamiseen töiden tekemisen sijasta. (Nielsen 2012.)

Vaikka käytettävyys olisi kunnossa täytyy silti varmistaa, että tuotteella voi suorittaa sille määritetyn tehtävän. Hyöty on toinen tärkeä osa käyttäjäkokemusta, jossa varmistetaan, että tuotteella voi tehdä sen mihin se on tarkoitettu. Vaikka tuote olisi todella helppokäyttöinen poistuu sen hyödyllisyys, kun sillä ei voi tehdä siltä odotettua tehtävää. Tuotteesta tulee silloin hyödyllinen, kun hyöty ja käytettävyys ovat molemmat kunnossa. (Nielsen 2012.)

3 MVC-suunnittelumalli ja käytetyt teknologiat

3.1 MVC-suunnittelumalli

MVC-suunnittelumallissa sovelluksen käyttöliittymä, logiikka ja tiedonvarastointi pyritään eriyttämään toisistaan eri kerroksiin. MVC-suunnittelumallissa nämä eri osa-alueet on jaoteltu malliin (model), näkymään (view) ja kontrolleriin (controller), josta suunnittelumalli saa myös nimensä. Nämä kerrokset vaihtavat dataa ja käskyjä keskenään, mutta eivät suoraan vaikuta toisiensa toimintaan (Kuvio 1.). Tämän avulla tiettyä osaa sovelluksesta voidaan muokata, ilman että se vaikuttaa koko sovelluksen toimintaan.



Kuvio 1. MVC kerroksien toiminta (MDN Web Docs 2022)

MVC auttaa myös sovellusten tiedostojen organisoinnissa juuri tämän erittelyn ansiosta. Organisoinnista tulee varsinkin tärkeää, kun sovellus kasvaa kokoa ja tiedostojen määrä lisääntyy. Organisoinnin tärkeys myös lisääntyy, kun sovelluksella on monta kehittäjää.

Malli vastaa järjestelmässä kaikesta dataan liittyvästä, esimerkiksi tiedon tallentamisesta, hakemisesta, ja poistamisesta. Malli ei ole riippuvainen näkymästä tai kontrollerista, se vain suorittaa sille määrätyt tehtävät. Riippumattomuus auttaa kehityksessä, koska malli voidaan rakentaa ja testata ennen kuin käyttöliittymä on luotu. (Abeysinghe 2009, 48–49.)

Näkymä vastaa käyttöliittymästä. Kaikki mitä käyttäjä sovelluksesta näkee, on näkymän kautta. Perinteisessä web-sovelluksessa näkymä on yleensä HTML sivu, johon mallista haettu data asetetaan.

Kontrolleri vastaa logiikasta mallin ja näkymän välillä. Kontrolleri käsittelee näkymästä tulleet pyynnöt, lähettää käskyt mallille ja valmistelee mallilta saadun datan näkymää varten.

3.2 Laravel

Laravel on ilmainen avoimen lähdekoodin PHP kielen ohjelmistokehys, jota käytetään web-kehityksessä. Laravelin on luonut Taylor Otwell ja ensimmäinen beta versio Laravelista julkaistiin kesäkuussa 2011. Otwell loi Laravelin korvaamaan toisen PHP kielen ohjelmistokehyksen CodeIgniterin. Otwellin mielestä CodeIgniteristä ei löytynyt tarvittavia ominaisuuksia, kuten sisäänrakennettua autentikointia. Laravel on suunniteltu käyttämään MVC-suunnittelumallia. (Koffer 2022.)

Laravel sisältää paljon ominaisuuksia, jotka nopeuttavat ja helpottavat kehittäjiä sovelluksien kehityksessä. Vaikka Laravel onkin enimmäkseen suunniteltu käytettäväksi palvelinpuolen ohjelmoinnissa, tarjoaa se myös ominaisuuksia käyttöliittymän rakentamisessa. (Koffer 2022.)

3.2.1 Eloquent

Eloquent on Laravelin käyttämä ORM (Object-relational mapper). Eloquent toimii ”Active Record” tyylin mukaan, jossa jokainen malli vastaa taulua tietokannassa. Mallista voidaan hakea, lisätä, muokata ja poistaa dataa kyseistä taulusta. (Laravel b.)

Mallista voidaan asettaa myös relaatiot muihin malleihin. Relaatiolla tarkoitetaan kahta tai useampaa tietokannan taulua, jotka ovat yhteyksissä toisiinsa. Relaatioita on useita erilaisia, ja Eloquent tukee useita yleisimpiä relaatioita kuten yhden suhde yhteen, yhden suhde moneen ja monen suhde moneen. Relaatio pitää kuitenkin ilmoittaa myös tietokannassa tauluja tehdessä. Kun relaatio merkitään malliin, niin suhteissa olevia malleja voidaan helposti hakea, ilman kyselyjen kirjoittamista. (Laravel c.)

Eloquentilla voidaan myös käyttää hyväksi Laravelin kysely rakentajaa (query builder). Kysely rakentajalla saadaan tietokannan kyselyistä helpommin luettavia, verrattuna perinteisiin tietokanta kyselyihin. Kysely rakentajassa on monia hyödyllisiä funktioita valmiina kehittäjälle, mutta sillä voidaan myös kirjoittaa normaaleja SQL-kyselyjä käyttämällä raakoja lausekkeita (raw expression). Laravelin kysely rakentaja suojaa järjestelmää SQL-

injektioilta, mutta raakoja lausekkeita käyttäessä suositellaan olemaan varovainen, koska se voi altistaa SQL-injektioille. (Laravel e.)

3.2.2 Blade

Blade on Laravelin käyttämä sivumoottori (templating engine). Sivumoottoreilla voidaan helposti sisällyttää dataa palvelimelta käyttöliittymään. Data asetetaan muuttujassa HTML koodin sekaan ja sivun renderöidessä muuttuja tulostaa arvon sivulle. Bladessa muuttujat asetetaan aaltosulkujen väliin. (Laravel a.)

```
<div class="col-sm-8 col-md-10">
    {{ $order->addressline }} <br/>
    {{ $order->zip }} {{ $order->city }}<br/>
    {{ $order->country }}
</div>
```

Kuva 1. Datan lisääminen sivulle.

Sovelluksessa on yleensä jotain osioita mitä halutaan näyttää joka sivulla. Tällaisia osioita voi olla esimerkiksi navigointivalikko ja alatunniste (footer). Sen sijaan että valikko ja tunniste kirjoitettaisiin joka sivulle, voidaan Bladella tehdä pohja (layout), jossa nämä osiot ovat. Muut sivut voidaan rakentaa tälle pohjalle, jolloin turhasta kirjoittamisesta päästään eroon. Pohjaan merkataan erikseen osiot johon toisen sivun sisältö lisätään. ”@yield” merkinnällä. Kun pohjaa halutaan käyttää, pitää se ilmoittaa ”@extends” merkinnällä ja lisätä oma haluttu sisältö ”@section” ja ”@endsection” väliin. (Laravel a.)

Bladella on myös mahdollista suorittaa esimerkiksi silmukoita ja ehtolausekkeita. Sivuille voidaan esimerkiksi tehdä osioita, jotka näkyvät vain sivuston ylläpitäjille ehtolausekkeella tai luoda listoja silmukoiden avulla. Myös perus PHP:n kirjoittaminen onnistuu Blade sivuille. (Laravel a.)

3.2.3 Migraatiot

Migraatioissa voidaan luoda, poistaa ja muokata tietokannan tauluja. Migraatiossa on kaksi funktiota: up ja down. Up funktio suoritetaan, kun migraatio ajetaan ja down funktio suoritetaan, jos migraatio halutaan peruuttaa. Up funktioon kirjoitetaan taulun kaavio, eli mitä sarakkeita tauluun halutaan ja mitä tietotyyppiä niihin tallennetaan (Kuva 2.). Sarakkeisiin voidaan myös lisätä määritteitä kuten pitääkö arvon olla uniikki ja saako arvo olla tyhjä. (Laravel d.)

```php
class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->increments('id');
            $table->string('name');
            $table->string('email')->unique();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::drop('users');
    }
}
```

Kuva 2. Esimerkki käyttäjä taulun luovasta migraatiosta.

Migraatiot voidaan luoda ja suorittaa Artisan komentorivin avulla. Artisan on Laravelin komentorivityökalu. Migraatio luodaan komennolla ”php artisan make:migration” jonka perään lisätään migraation nimi. Komento luo nimen mukaan tiedoston ja lisää sen perään aikaleiman, milloin migraatio on luotu, jonka avulla nähdään milloin mitäkin muutoksia tietokantaan on tehty. Esimerkiksi jos halutaan luoda käyttäjä taulu, olisi komento ”php artisan make:migration create_users_table”. Kun migraatio halutaan ajaa tietokantaan, Artisanilla suoritetaan komento ”php artisan migrate”. Tämä komento ajaa migraatiot tietokantaan. Migraatiot voidaan kumota Artisan komennolla ”php artisan migrate:rollback”. (Laravel d.)

3.3 Bootstrap

Bootstrap on ilmainen avoimen lähdekoodin ohjelmistokehys, joka sisältää runsaasti eri CSS tyylimäärityksiä ja JavaScript komponentteja. Käyttäjälle tarvitsee olla vain perustieto HTML ja CSS kielistä aloittaakseen Bootstrapin käytön. (W3Schools a.)

Bootstrap nopeuttaa ja helpottaa responsiivisien sivujen luomista. Responsiivisuudella tarkoitetaan sivuston automaattista skaalautumista eri kokoisille näytöille, jolloin sivusto näyttää hyvältä vierailtiin sivulla sitten puhelimella tai tietokoneella. (W3Schools a.)

4 Toimeksiantajan toiminnanohjausjärjestelmä

4.1 Järjestelmän toiminta

Toimeksiantajan toiminnanohjausjärjestelmässä tehdään, käsitellään ja arkistoidaan turvien tilaukset ja vahinkoilmoitukset. Järjestelmää käyttävät myös yhteistyökumppanit, jotka myyvät turvia. Yhteistyökumppaneille on luotu omia räätälöityjä turvia. Yhteistyökumppaneina on yksittäisiä yrityksiä ja isompia ketjuja, joiden alla toimii monia toimipisteitä.

Turvan ostajalle toimitetaan turvan ehdot, todistus turvasta ja tietosuojaseloste. Turvan ehdoissa käydään läpi

- turvan myöntäjä
- hallinnoijan yhteystiedot
- kyseisen turvan esittely
- saatavilla olevat vaihtoehdot turvasta
- miten toimia rikkoutumistilanteessa
- turvan kattavuussisältö
- yleiset rajoitukset
- yleiset ehdot
- oheispalvelut ja edut
- korvauskäsittelyn erityishuomiot
- miten toimia riitatilanteessa.

Turvan todistuksessa on kirjattu turvan tiedot, ajoneuvon tiedot ja asiakkaan tiedot. Kun uusi turva tehdään, lähettää järjestelmä asiakkaalle sähköpostitse turvan ehdot, todistuksen turvasta ja tietosuojaselosteen. Saman sähköpostin voi myös lähettää tilauksen sivulta, esimerkiksi jos asiakkaan sähköposti on merkattu väärin ja muokataan myöhemmin. Jos asiakkaalla ei ole sähköpostiosoitetta, ehdot ja todistuksen voi myös ladata tilauksen jälkeen tilaussivulta ja tulostaa asiakkaalle.

Järjestelmässä on turvien ehtotiedostoja ladattu kolmelle eri tasolle:

- Ensimmäinen taso on turvan taso.
- Toinen taso on yrityksen taso.
- Kolmas taso on järjestelmän yleinen taso.

Turvan tasolla oleviin ehtoihin on kirjattu kyseisen turvan ehdot ja kattavuussisältö. Yrityksen tasolla oleviin ehtoihin on yleensä sisällytetty yrityksen useamman turvan ehdot ja

kattavuussisältö. Järjestelmän tasolla oleva ehtotiedosto on osa vanhaa aikaa, kun yksi tiedosto sisälsi kaikkien myytävien turvien ehdot ja kattavuussisällöt.

Kun ehtoja ollaan lähettämässä asiakkaalle tai lataamassa tilaussivulta, järjestelmä tarkistaa järjestyksessä turva, yritys, järjestelmä onko ehtoja ladattu ja valitsee ensimmäisen vaihtoehdon, jossa tiedosto on ladattuna. Tarkistuksen ei pitäisi ikinä mennä järjestelmän tasolle asti, vaan ehtotiedoston pitäisi olla asetettuna joko turvalle tai yritykselle.

4.2 Ongelmat järjestelmässä

Turvien ehtoja muokataan noin vuosittain. Kun uusia ehtoja ladataan järjestelmään, tulevat ne vanhojen ehtotiedostojen tilalle ja vanhat tiedostot poistuvat.

Kun suuremman ketjun ehtotiedostoja päivitetään, joudutaan yritystasolla jokaiselle toimipisteelle asettamaan uusi tiedosto. Koska kyseessä on saman ketjun toimipisteet, ovat ehtotiedostot kaikilla samat. Tällä hetkellä suurimalla ketjulla on noin 30 toimipistettä, johon jokaiseen on ladattu sama ehtotiedosto. Ehtotiedostojen päivittäminen järjestelmään on työläs projekti, jossa ylläpitäjän pitää olla tarkkaavainen, jotta jokaisen yrityksen ehdot päivitetään oikealla tiedostolla.

Jotkin turvat saattavat alkaa muutaman vuoden päästä turvan tekemisestä. Esimerkiksi jos ajoneuvossa on valmistajan takuu vielä voimassa turvan luonti vaiheessa, astuu turva voimaan vasta valmistajan takuun jälkeen. Jos asiakas esimerkiksi hävittää turvan ehdot ja pyytää lähettämään ehdot uudestaan ehtojen päivittämisen jälkeen, ei järjestelmän kautta ostohetken ehtoja voida toimittaa asiakkaalle. Vahinkoilmoituksien käsittelyssä aiheutuu myös ylimääräistä vaivaa pitää kirjaa milloin ja miten ehdot ovat muuttuneet eri turvissa.

5 Työn suunnittelu

5.1 Ehtotiedoston lataaminen ketjulle

Ehtotiedostojen päivittämisestä päätettiin tehdä tehokkaampaa luomalla ehtotiedostoille uusi taso. Uudeksi tasoksi tulisi ketjutaso, johon yritykset liitettäisiin. Kun ehtotiedostot saadaan ladattua vain yhteen sijaintiin, saadaan tiedostojen päivittämisen prosessi tehokkaammaksi ja mahdollisia riskejä väärän tiedoston lataamiselle pienennettyä.

Uusi taso tulisi korvaamaan yritystason. Ketjutasolle asetetaan ehtotiedostot ja tilauksen yhteydessä tarkistetaan tilauksen tekijästä mihin ketjuun kyseinen toimipiste kuuluu.

Ketjun malli ja relaatio yritykseen oli jo järjestelmään luotu, mutta sen kehitys oli jäänyt kesken. Ketjulle pitää luoda luettelosivu missä on listattu kaikki ketjut. Luettelosivulle pitää myös lisätä mahdollisuus luoda uusia ketjuja. Yksittäisen ketjunsivulle pitää luoda luettelo siihen kuuluvista yrityksistä ja lisätä mahdollisuus liittää yrityksiä ketjuun. Ketju pitää myös pysytä poistamaan ja yritys pitää myös pysytä poistamaan ketjusta.

5.2 Ehtotiedoston linkittäminen tilaukseen

Asiakkaalle tulee toimittaa ne ehdot mitkä on kaupantekohetkellä olleet voimassa. Sähköpostin lähetys tilauksen sivulta on käytettävyydeltään hyvässä kunnossa. Mutta jos ehtotiedosto on päivitetty, menettää se hyötynsä, koska siinä tapauksessa asiakkaalle lähetettäisiin väärä ehtotiedosto.

Väärän tiedoston lähettämisen ja tulostamisen välttämiseksi päätettiin linkittää ehtotiedosto tilaukseen. Ehtotiedoston linkitys auttaa tilanteissa, jossa turva alkaa reilusti myöhemmin turvan tilauspäivämäärästä. Näin saadaan helposti selville mitkä ehdot ovat olleet voimassa turvan tilaushetkellä.

Tilauksen muokkaustilanteissa pitää ehtotiedostojen myös muuttua muokkauksen mukaisesti. Jos esimerkiksi tilauksen turvaa muutetaan, pitää järjestelmän hakea ehtotiedostot uudestaan. Turvasta pitää voida myös manuaalisesti lisätä tai vaihtaa ehtotiedosto.

5.3 Ehtotiedostojen arkistointi

Jotta ehtotiedoston linkityksestä tilaukseen on apua eivät vanhat ehtotiedostot saa poistua järjestelmästä. Vanhat ehtotiedostot tulee siis arkistoida järjestelmään. Järjestelmästä tulee myös saada selville, milloin tiedostot ovat olleet käytössä järjestelmässä.

Molempien turvan ja ketjun sivuille pitää luoda mahdollisuus ladata uudet ehtotiedostot ja arkistoida vanhat ehtotiedostot. Ehtotiedostoille pitää asettaa arvot mistä päivämäärästä mihin päivämäärään ne ovat olleet käytössä. Uuden ehtotiedoston lisäyksessä määritetään alkamispäivämäärä. Jos turvalla tai ketjulla on vanha aktiivinen tiedosto, pitää sen loppua ennen kuin uusi astuu voimaan. Ketjulla tai turvalla voi olla aktiivisena kolme eri ehtotiedostoa, koska käännökset pitää myös ottaa huomioon. Käännöksetkin pitää arkistoida järjestelmään.

Molemmille ketjulle ja turvalle pitää luoda luettelo niiden ehtotiedostoista. Kun uutta ehtotiedostoa ladataan järjestelmään, pitää siihen määritellä alkamispäivämäärä, kieli ja asettaa tiedosto. Järjestelmässä tulee olla myös mahdollisuus poistaa ehtotiedosto järjestelmästä, jos esimerkiksi ladataan väärän turvan ehtotiedosto. Ehtotiedostot pitää myös pystyä lataamaan järjestelmästä.

6 Työn toteutus

6.1 Ehtotiedoston luominen

Itse tiedostonhallinnan muutos alkoi luomalla ehtotiedostolle malli. Ehtotiedosto tauluun asetettiin seuraavat sarakkeet:

- tunniste
- turvan tunniste
- ketjun tunniste
- tiedoston nimi
- tiedoston polku
- kieli
- voimassa alkaen
- voimassa asti
- luontipäivämäärä
- muokkauspäivämäärä
- poistopäivämäärä (Kuva 3.).

```php
class CreateTermsFilesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('terms_files', function (Blueprint $table) {
            $table->increments('id');
            $table->integer('insurance_id')->unsigned()->nullable();
            $table->integer('chain_id')->unsigned()->nullable();
            $table->string('file_name');
            $table->string('file_path');
            $table->string('language');
            $table->date('valid_from');
            $table->date('valid_to')->nullable();
            $table->timestamps();
            $table->softDeletes();

            $table->foreign('insurance_id')->references('id')->on('insurances')->onDelete('cascade')->onUpdate('no action');
            $table->foreign('chain_id')->references('id')->on('chains')->onDelete('cascade')->onUpdate('no action');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('terms_files');
    }
}
```

Kuva 3. Ehtotiedoston migraatio.

Turvan tunniste ja ketjun tunniste toimivat viiteavaimina turvan ja ketjun tauluihin. Relaatiot ehtotiedostoon on määritetty niin että ehtotiedosto kuuluu yhteen turvaan tai ketjuun ja molemmilla ketjulla ja turvalla voi olla monta ehtotiedostoa.

6.2 Tilauksen ja ehtotiedoston suhde

Ehtotiedoston ja tilauksen relaatio määriteltiin monen suhde moneen relaatioon, koska tilaukseen voi linkittää käännösten takia monta tiedostoa ja yksi tiedosto voi olla linkitettynä moneen tilaukseen. Kun kyseessä on monen suhde moneen relaatio, tarvitaan tietokannassa taulujen väliin uusi taulu, johon relaatiot merkataan. Tätä välissä olevaa taulua kutsutaan pivot-tauluksi. Laravel ohjeistaa nimeämään taulun molempien taulujen nimien mukaan aakkosjärjestyksessä alaviivalla eroteltuna. Migraatioon merkattiin taulun nimeksi ”order_terms_file”. Tauluun lisättiin kaksi saraketta, ”order_id” ja ”terms_file_id” ja molemmat asetettiin viiteavaimiksi viittaamaan omien taulujensa perusavaimia. Tauluun lisättiin myös aikaleimat, milloin rivi on luotu ja viimeksi muokattu (Kuva 4.).

```php
class CreateOrderTermsFileTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('order_terms_file', function (Blueprint $table) {
            $table->integer('order_id')->unsigned();
            $table->integer('terms_file_id')->unsigned();
            $table->timestamps();

            $table->foreign('order_id')->references('id')->on('orders')->onDelete('cascade');
            $table->foreign('terms_file_id')->references('id')->on('terms_files')->onDelete('cascade');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('order_terms_file');
    }
}
```

Kuva 4. Pivot taulun migraatio.

Ehtotiedoston ja tilauksen malleihin molempiin asetettiin ”belongsToMany” relaatio toisiinsa (kuva 5.). Kun relaatio on ilmoitettu mallissa, saadaan koodissa helposti selville tilaukseen linkitetyt ehdot ja onko ehtotiedostolla jo tilauksia järjestelmässä.

```php
class TermsFile extends Model
{
    use softDeletes;
    protected $dates = ['valid_from', 'valid_to'];
    protected $fillable = ['file_name', 'file_path', 'language', 'valid_from', 'valid_to'];

    public function chain()
    {
        return $this->belongsTo(Chain::class);
    }

    public function insurance()
    {
        return $this->belongsTo(Insurance::class);
    }

    public function orders()
    {
        return $this->belongsToMany(Order::class);
    }
}
```

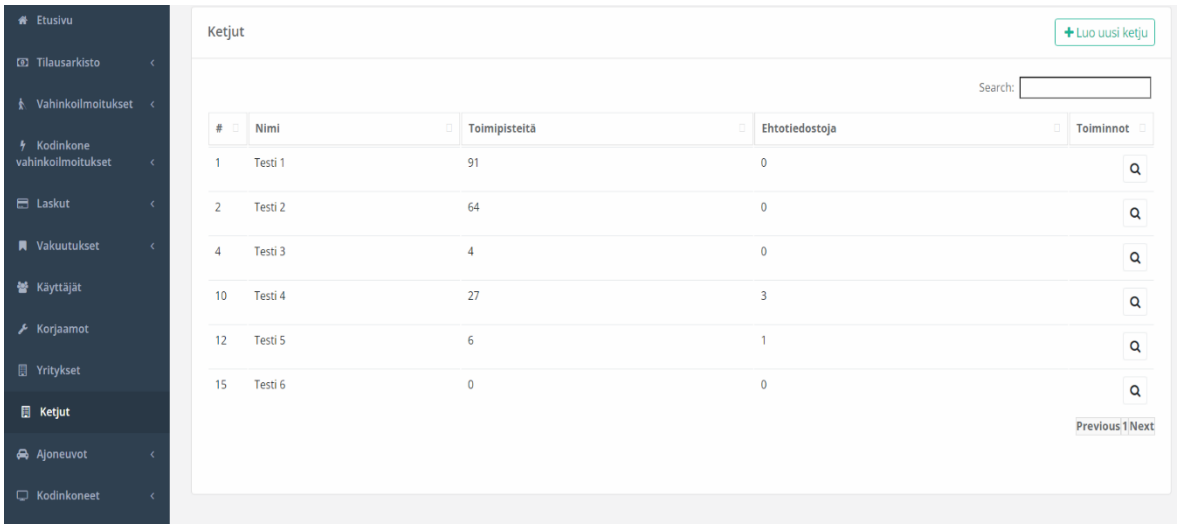
Kuva 5. Ehtotiedoston relaatiot.

6.3 Ketjun sivujen luonti

Ketjulle luotiin kontrolleri, johon tehtiin funktiot uuden ketjun luomiselle, ketjun poistamiselle, yrityksen lisäämiselle ketjuun ja yrityksen poistamiseen ketjusta. Ketjun ja yrityksen malleissa relaatiot oli määritelty seuraavasti: ketjulla voi olla monta yritystä ja yritys voi kuulua yhteen ketjuun.

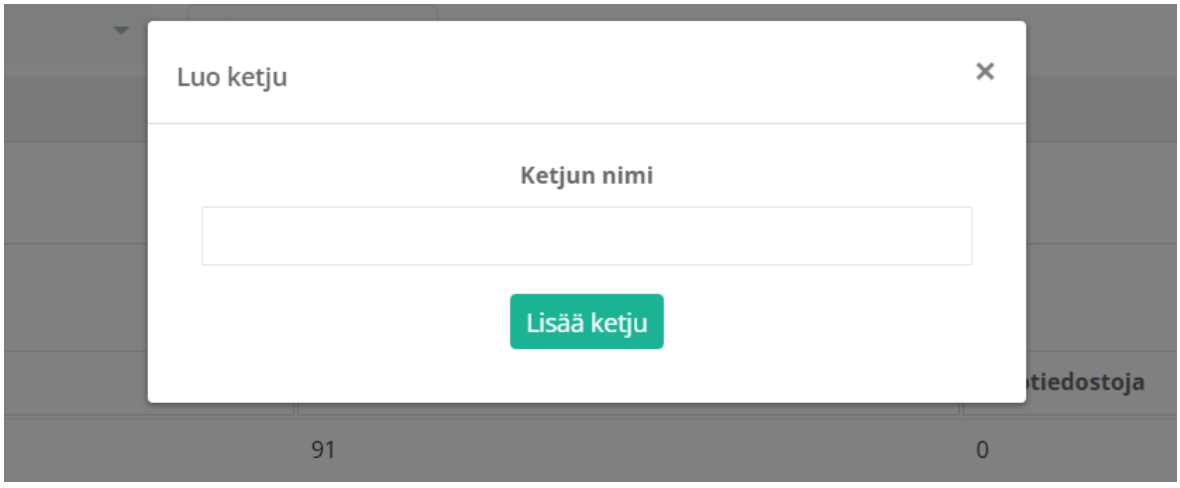
Ketjulle tehtiin luettelo sivu, jossa listataan kaikki ketjut (Kuva 6.). Ketjut sijoitetaan tauluun, jossa ilmoitetaan

- ketjun tunniste
- nimi
- toimipisteiden määrä
- ladattujen ehtotiedostojen määrä
- ketjun sivulle vievä painike.



Kuva 6. Ketju luettelo

Taulun yläpuolelle lisättiin painike ketjun luomiselle, joka avaa modaali ikkunan, jossa on lomake uuden ketjun luontia varten. Modaali ikkuna avautuu pääsivun päälle ja estää pääsivun käytön, kunnes uusi ketju lisätään tai ikkuna suljetaan. Uutta ketjua varten tarvitaan vain ketjun nimi, joten lomakkeessa on vain tekstikenttä nimelle ja painike lomakkeen lähettämiseen (Kuva 7.).

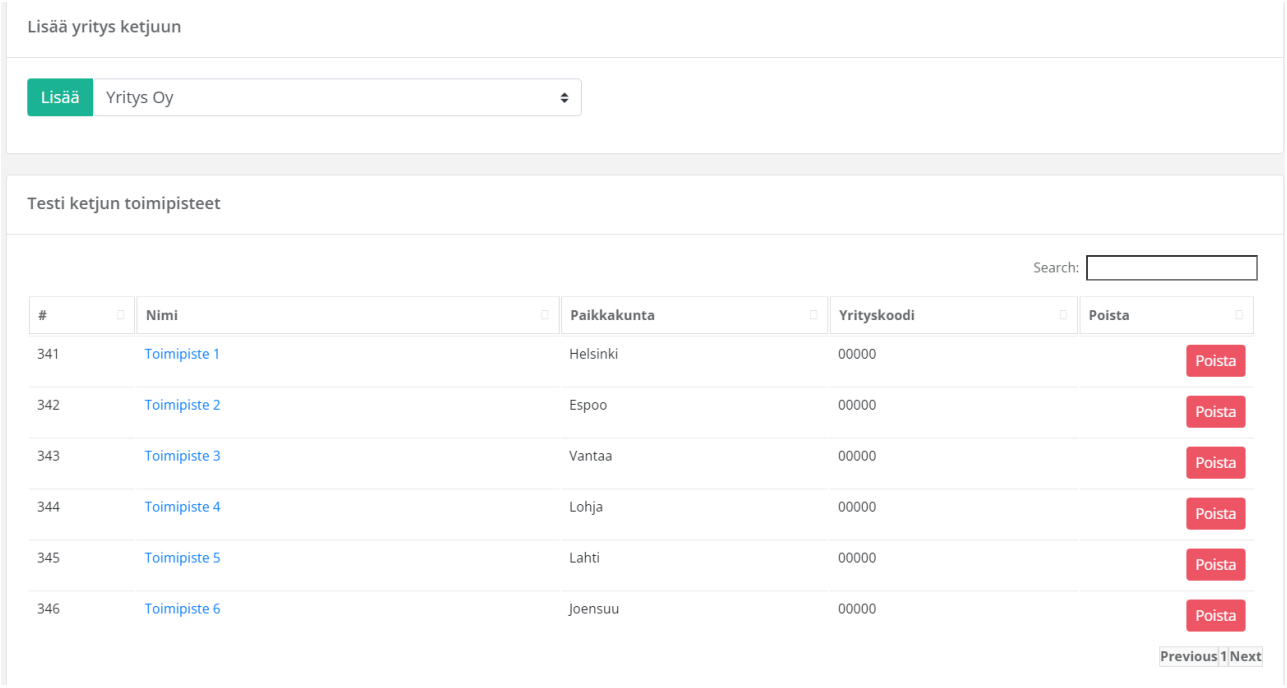


Kuva 7. Modal ikkuna uuden ketjun luontia varten.

Yksittäisen ketjun sivulla voidaan lisätä ja poistaa yrityksiä ketjusta. Yritykset ovat samanlaisessa taulussa kuin ketjun luettelo sivulla. Tauluun asetettiin seuraavat sarakkeet:

- yrityksen tunniste
- nimi
- paikkakunta
- yrityskoodi
- painike toimipisteen poistoa varten.

Uusi yritys voidaan lisätä ketjuun pudotusvalikosta, johon on listattu kaikki yritykset mitkä eivät ole linkitetty mihinkään ketjuun (Kuva 8.).



Kuva 8. Ketjusivun luettelo toimipisteistä ja toimipisteen lisäyslomake.

6.4 Ehtotiedoston lataaminen ja poistaminen

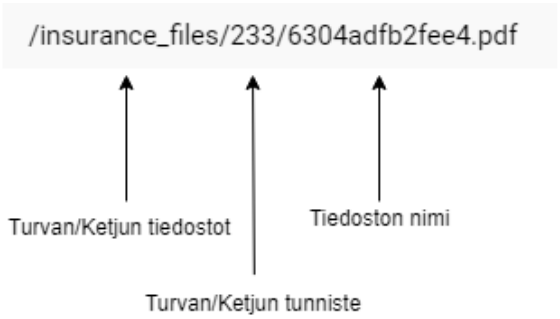
Ketjun ja turvan sivuille luotiin lomake, johon määritetään ehtotiedoston kieli, alkamispäivämäärä ja ladattava ehtotiedosto. Uuden ehtotiedoston pyyntö lähetetään joko ketjun tai turvan kontrolleriin, riippuen siitä kumpaan ehtotiedostoa ollaan lisäämässä. Kontrolleri vastaanottaa ja validoi pyynnön. Validoinnin tuloksen mukaan kontrolleri joko luo uuden ehtotiedoston tai lopettaa prosessin ja ilmoittaa käyttäjälle mikä aiheutti virheen validoinnissa.

Uuden ehtotiedoston validoinnissa, järjestelmä ensin tarkastaa ladatun ehtotiedoston alkamispäivämäärän. Alkamispäivämäärän pitää olla joko kuluva päivä tai tulevaisuudessa. Järjestelmä myös tarkistaa viimeisimmän ladatun ehtotiedoston, ja vertailee uuden ja aikaisemman ehtotiedoston päivämääriä. Jos molemmissa tiedostoissa on sama aloituspäivämäärä tai vanhat ehdot eivät ole vielä aktiivisia, niin uuden ehtotiedoston asettaminen ei onnistu. Näin varmistetaan, ettei kahdet saman kieliset ehdot voi olla aktiivisia samaan aikaan turvassa tai ketjussa.

Jos validointi menee läpi, asetetaan tiedoston nimi PHP:n uniqid funktiolla. Koska tallennetavat tiedostot ovat yleensä nimetty tekniikkaturvaehdoiksi, tiedostolle annetaan tallentaessa uniikki nimi, jotta saman nimisiä tiedostoja ei vahingossa tallenneta hakemistoon. Uniqid funktio luo tekstijonon kellon mukaan mikrosekuntien perusteella. Vaikka uniqid ei

palautakkaan 100 % todennäköisyydellä aina uniikkia arvoa, on todennäköisyys siihen, että sama arvo tulisi kahdesti sen verran pieni, että funktiota voi käyttää luotettavasti tähän käyttötarkoitukseen. (W3Schools b.)

Ketjun ja turvan tiedostot on eritelty omiin kansioihin, ja näiden sisällä luodaan jokaiselle turvalle ja ketjulle tunnisteella nimetty kansio, johon sen ehtotiedostot tallennetaan (Kuvio 2.). Tiedostopolku tallennetaan myös tietokantaan tiedostopolku sarakkeeseen. Jos samalla kielellä oli aikaisemmin asetettu ehtotiedosto, asettaa järjestelmä sen päättymispäivämääräksi uuden ehtotiedoston alkamispäivämäärää edeltävän päivän.



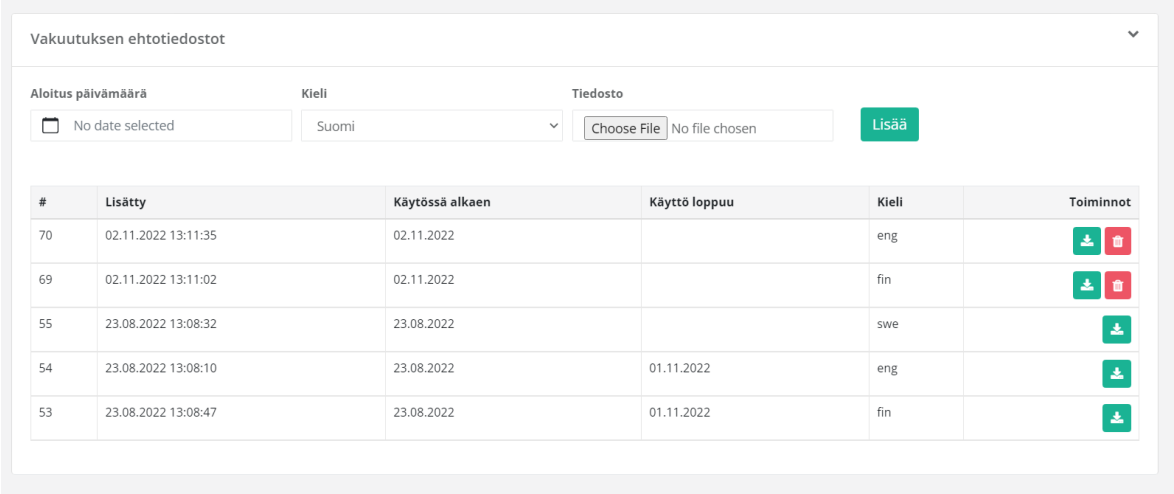
Kuvio 2. Esimerkki tiedostopolusta

Jos vahingossa järjestelmään esimerkiksi ladataan väärä tiedosto tai tiedostolle asetetaan väärä kieli, voidaan ehtotiedosto poistaa järjestelmästä. Tiedoston poistoon on myös asetettu tietyt säännöt. Tiedoston voi poistaa, jos ehtotiedosto ei ole linkitettynä tilauksiin ja alkamispäivämäärä ei ole kuluvaa päivää aikaisempi. Jos nämä ehdot toteutuvat, näkyy poistopainike ehtotiedostojen taulussa. Samat ehdot tarkistetaan kuitenkin vielä kontrollerissa siltä varalta, jos esimerkiksi ehtotiedosto on linkitetty tilauksiin, sillä välillä kun sivu on ollut auki. Tiedoston poiston jälkeen viimeksi samalla kielellä olevan aktiivisen ehtotiedoston loppumispäivämäärän arvo muutetaan tyhjäksi, jolloin siitä tulee taas aktiivinen ehtotiedosto.

Kaikki turvalle tai ketjulle asetut ehtotiedostot ovat näkyvissä niiden omilta sivuilta. Sivulle asetettiin taulu, johon asetettiin sarakkeet:

- tiedoston tunniste
- lisäyspäivämäärä
- aloituspäivämäärä
- loppumispäivämäärä
- kieli
- toiminnot.

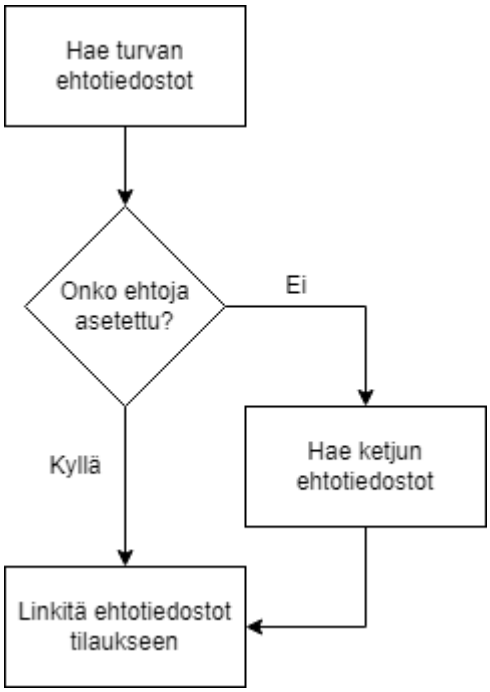
Toiminnot sarakkeella on latauspainike, josta kyseisen ehtotiedoston voi ladata ja mahdollinen painike ehtotiedoston poistoa varten (Kuva 9.).



Kuva 9. Uuden ehtotiedoston lomake ja ehtotiedostojen taulu turvansivulla.

6.5 Ehtotiedoston linkittäminen tilaukseen

Kun uusi tilaus tehdään järjestelmään, tarkistetaan valittu turva ja tilauksen luonut yritys. Järjestelmä koittaa ensin hakea turvan ehtotiedostot ja linkittää ne tilaukseen. Jos turvan tasolla ei ole asetettu turvia, hakee järjestelmä seuraavaksi yrityksen kautta ketjun tason ehtotiedostot ja linkittää ne tilaukseen (Kuvio 3.).



Kuvio 3. Ehtotiedostojen hakeminen ja linkittäminen.

Järjestelmä hakee kaikki joko turvan tai ketjun ehtotiedostot ja vertailee kuluvaa päivää ehtotiedostoon asetettuihin päivämääriin. Jos kuluva päivä on suurempi tai yhtä suuri kuin ehtotiedoston aloituspäivämäärä ja ehtotiedoston loppumispäivämäärää ei ole asetettu tai loppumispäivämäärä on suurempi kuin kuluva päivä, linkitetään ehtotiedostot tilaukseen. Huomioitavaa on myös se, että järjestelmä linkittää kaikki aktiiviset ehtotiedostot tilaukseen, eli myös mahdolliset ruotsin- ja englanninkieliset käännökset linkitetään tilaukseen, vaikka niitä ei erikseen tilauslomakkeella ilmoitettaisi.

6.6 Ehtojen lähettäminen tai tulostaminen asiakkaalle

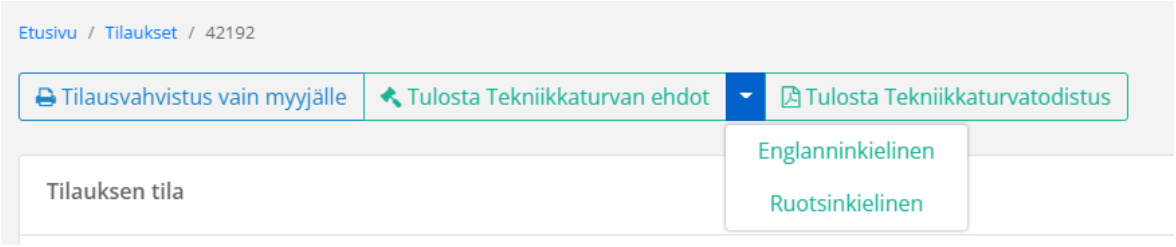
Asiakkaalle lähetetään ehtotiedosto, turvatodistus ja tietosuojaseloste aina tilauksen teon yhteydessä automaattisesti järjestelmästä. Järjestelmästä on myös mahdollista lähettää materiaalit uudestaan tilauksen sivulta. Jos asiakkaalla ei ole esimerkiksi sähköpostiosoitetta, voidaan turvan ehdot ja turvan todistus myös lataa ja tulostaa asiakkaalle turvan sivulta.

Kun tilaus tehdään järjestelmään, käydään ennen sähköpostin lähetystä läpi, että tietyt ehdot täyttyvät. Jos ehtotiedostoja ei ole linkitetty, niin sähköpostin lähetys keskeytetään ja käyttäjälle ilmoitetaan puuttuvista ehtotiedostosta ja kehotetaan ottamaan yhteyttä ylläpitoon. Toiseksi tarkistetaan tilaukseen merkitty asiakkaan sähköpostiosoite. Jos sähköpostiosoitetta ei ole lisätty, niin sähköpostin lähetys keskeytetään ja käyttäjää pyydetään tulostamaan materiaalit asiakkaalle. Jos ehtotiedostot on linkitetty ja asiakkaan sähköposti merkattu, siirtyy järjestelmä sähköpostin lähetykseen.

Tilaussähköpostissa lähetetään asiakkaalle aina suomenkieliset ehdot ja todistus. Järjestelmä tarkistaa tilauksesta halutaanko materiaalit myös käännettynä. Jos käännökset halutaan ruotsin ja englannin kielellä, joudutaan lähettämään asiakkaalle kaksi sähköpostia, koska liitteiden koko ylittää kokorajoituksen. Jos käännökset halutaan vain ruotsiksi tai englanniksi, niin riittää yksi sähköposti. Järjestelmä rakennettiin kaikkien käännöksien kohdalla lähettämään englanninkieliset materiaalit erillisessä sähköpostissa. Liitetiedostojen nimi muutetaan Tekniikkaturvaehdoiksi ja käännöksissä nimen eteen lisätään käännöksen kieli. Tilaussivulta sähköpostin lähettämisen yhteydessä voidaan sähköpostin tekstiä muuttaa, mutta muuten sähköpostin lähetyslogiikka toimii samalla tavalla kuin tilauksen teon yhteydessä.

Tilaussivulla olevaa ehtojen tulostamista varten olevan painikkeen logiikkaa muutettiin myös toimimaan uudella ehtotiedoston logiikalla. Isoimmasta painikkeesta päästään aina tulostamaan tilaukseen linkitetty suomenkielinen ehtotiedosto. Painikkeen yhteyteen lisättiin pudotusvalikko, josta voi ladata tilaukseen linkitetyt käännökset (Kuva 10.). Jos

käännöksiä ei ole linkitetty tilaukseen, ei pudotusvalikko näy käyttäjälle. Huomattavaa on myös, vaikka tilaukseen ei ole erikseen ilmoitettu halutaanko käännöksiä, ovat kaikki aktiiviset käännökset ehtotiedostosta silti tulostettavissa tilauksen sivulta. Kun tiedosto ladataan, muutetaan tiedoston nimi vielä samalla tavalla kuin sähköpostinlähetyksessä.



Kuva 10. Painike ehtojen tulostamiselle.

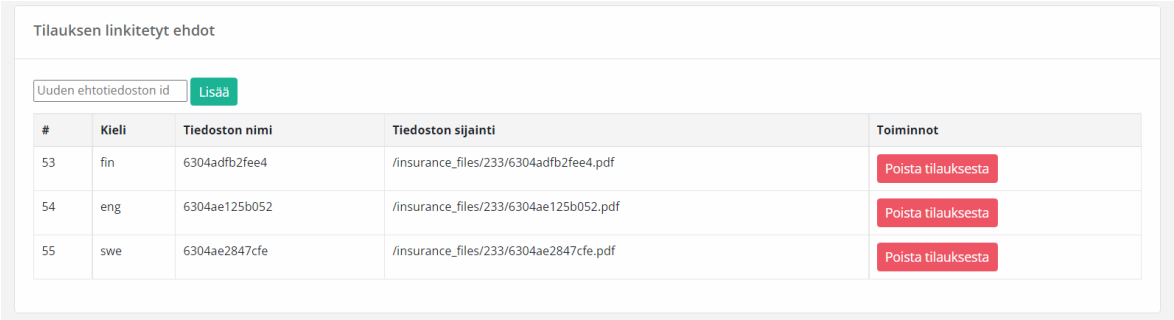
6.7 Tilauksen muokkaaminen

Tilauksista on mahdollista muokata melkeinpä kaikkia tietoja, mukaan lukien vaihtaa turvaa. Kun tilausta on muokattu, järjestelmä tarkistaa tapahtuiko ehtotiedostoihin liittyviä muutoksia. Jos turvaa tai haluttuja käännöksiä muutetaan, poistetaan linkitetyt ehtotiedostot tilauksesta ja haetaan ehtotiedostot uudestaan samaan tapaan kuin tilauksen tekohetkellä (Kuvio 3).

Tilauksen muokkaussivulle lisättiin taulu, josta näkee kaikki tilaukseen linkitetyt ehtotiedostot. Tauluun asetettiin seuraavat sarakkeet:

- tunniste
- kieli
- tiedoston nimi
- tiedoston sijainti
- toiminnot.

Toiminnot sarakkeesta on mahdollista poistaa ehtotiedoston linkitys tilauksesta. Tilauksen muokkaussivulle lisättiin myös mahdollisuus linkittää haluttu ehtotiedosto tilaukseen tunnisteen avulla (Kuva 11.). Tilauksella voi kuitenkin olla vain yksi kunkin kielen ehtotiedostoa linkitettynä, eli jos haluaa vaihtaa esimerkiksi suomenkielisen ehtotiedoston, pitää ensin vanhan suomenkielisen ehtotiedoston linkitys poistaa tilauksesta.



Kuva 11. Tilaukseen linkitetyt ehtotiedostot taulu.

7 Yhteenveto ja pohdinta

Opinnäytetyön tavoitteena oli parantaa toimeksiantajan toiminnanohjausjärjestelmän tiedostojen hallintaa. Opinnäytetyö rajattiin koskemaan vain ajoneuvopuolen ehtotiedostoja, ja niitä koskevia ongelmia. Ongelmat koskivat käytettävyydeltään ja hyödyltään tiedostojen päivittämistä järjestelmään. Opinnäytetyössä käytiin teoriaosassa läpi käytettävyyden määrittämistä ja tärkeyttä, MVC-suunnittelumallia, Laravelia ja sen ominaisuuksia, Bootstrappia ja toimeksiantajan toiminnanohjausjärjestelmän toimintaa. Opinnäytetyön toiminnallisessa osuudessa käytiin läpi suunnitelma, miten järjestelmää koskevia ongelmia voidaan parantaa. Lopuksi käytiin läpi työn toteutus.

Yhtenä ongelmana järjestelmässä oli samojen ehtotiedostojen lataaminen järjestelmään ketjujen toimipisteille. Ongelma ratkaistiin liittämällä toimipisteet ketjuun ja lataamalla ehtotiedosto suoraan ketjulle. Toisena ongelmana oli ehtotiedostojen poistuminen järjestelmästä. Ongelma ratkaistiin arkistoimalla vanhat ehtotiedostot järjestelmään ja linkittämällä ehtotiedosto tilauksiin. Mielestäni opinnäytetyön tavoitteeseen päästiin. Toimeksiantajan toiminnanohjausjärjestelmän tiedostonhallinnan käytettävyyteen liittyviin ongelmiin löydettiin ratkaisut.

Vaikka tiedostonhallintaa saatiinkin parannettua ei sitä voida vielä toimeksiantajalla ottaa käyttöön. Uusi tiedostonhallinta pitää saada vielä toimimaan vanhan järjestelmän päällä, koska vanhat tilaukset eivät olet linkitetty ehtotiedostoihin. Työssä ei myöskään asetettu mitään auktorisointeja ketkä uusia ominaisuuksia voivat käyttää, eli kuka tahansa järjestelmän käyttäjä voisi lisätä ja poistaa ehtotiedostoja miltä tahansa ketjulta.

Ketjutason jatkokehitystä tullaan myös todennäköisesti jatkamaan. Esimerkiksi sen sijaan että uudet järjestelmän käyttäjät lisättäisiin jokaiseen ketjun toimipisteeseen käyttäjäksi, voisi ketjua jatkokehittää niin että käyttäjät voisi liittää suoraan ketjuun. Tällä saataisiin myös uusien käyttäjien lisäämisen liittyvää tehokkuutta nostettua.

Ehtotiedostojen arkistointia ja linkittämistä voi myös jatkokehittää. Ehtotiedostojen arkistoinnin ja linkittämisen voisi liittää osaksi myös kodinkonepuolen tilauksien tiedostojenhallintaa.

Lähteet

Abeysinghe, S. 2009. PHP team development easy and effective team work using MVC, agile development, source control, testing, bug tracking, and more. Birmingham: Packt Publishing.

Koffer, P. 2022. A brief guide through Laravel. Blogi. Viitattu 25.11.2022. Saatavissa https://mdevelopers.com/blog/a-brief-guide-through-laravel

Laravel a. Blade Templates. Viitattu 28.10.2022. Saatavissa https://laravel.com/docs/8.x/blade

Laravel b. Eloquent: Getting Started. Viitattu 27.10.2022. Saatavissa https://laravel.com/docs/8.x/eloquent

Laravel c. Eloquent: Relationships. Viitattu 27.10.2022. Saatavissa https://laravel.com/docs/8.x/eloquent-relationships

Laravel d. Database: Migrations. Viitattu 28.10.2022. Saatavissa https://laravel.com/docs/8.x/migrations

Laravel e. Database: Query Builder. Viitattu 27.10.2022. Saatavissa https://laravel.com/docs/8.x/queries

Logistiikan Maailma. Toiminnanohjausjärjestelmä. Viitattu 20.11.2022. Saatavissa https://www.logistiikanmaailma.fi/logistiikka/ohjausjarjestelmat/toiminnanohjausjarjestelma/

MDN Web Docs 2022. Model-view-controller light blue. Viitattu 26.10.2022. Saatavissa https://developer.mozilla.org/en-US/docs/Glossary/MVC

Nielsen, J. 2012. Usability 101: Introduction to Usability. Nielsen Norman Group. Viitattu 24.11.2022. Saatavissa https://www.nngroup.com/articles/usability-101-introduction-to-usability/

Niemelä, A. Miten ja miksi käytettävyyttä tutkitaan? Johdanto käytettävyyden ja käyttäjäkokemuksen tutkimiseen. Fraktio. Blogi. Viitattu 24.11.2022. Saatavissa https://www.fraktio.fi/blogi/miten-ja-miksi-kaytettavyytta-tutkitaan-johdanto-kaytettavyyden-ja-kayttajakokemuksen-tutkimiseen

W3schools a. Bootstrap 4 Get Started. Viitattu 18.11.2020. Saatavissa https://www.w3schools.com/bootstrap4/bootstrap_get_started.asp

W3schools b. PHP uniqid() Function. Viitattu 27.10.2022. Saatavissa https://www.w3schools.com/php/func_misc_uniqid.asp