SAVONIA
ammattikorkeakoulu

OPINNÄYTETYÖ - AMMATTIKORKEAKOULUTUTKINTO
TEKNIIKAN JA LIIKENTEEN ALA

KEVYTYRITTÄJÄPALVELUN SUUNNITTELU JA TOTEUTUS

TEKIJÄ Samu Miettinen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Tutkinto-ohjelma Tietotekniikan tutkinto-ohjelma	
Työn tekijä(t) Samu Miettinen	
Työn nimi Kevytyrittäjäpalvelun suunnittelu ja toteutus	
Päiväys 29.4.2022	Sivumäärä/Liitteet 21
Toimeksiantaja/Yhteistyökumppani(t) DuuniOnline Oy	

Tiivistelmä

Tämän projektin tavoitteena oli uudistaa DuuniOnline Oy:n kevytyrittäjäpalvelu helppokäyttöisemmäksi, nopeammaksi ja mobiiliystävällisemmäksi, sillä edellisessä versiossa näillä osa-alueilla oli puutteita.

Projektissa suunniteltiin ja toteutettiin nykyaikaisempi sekä parempi versio alkuperäisestä toteutuksesta, joka on ulkoasullisesti ja käytettävyydeltään enemmän sitä, mitä DuuniOnline visioi ohjelmiston alun perin olevan. Kokonaisuus on toteutettu Laravel 7 -kehitysympäristöön hyödyntäen kolmannen osapuolen taloushallinto rajapintaa, jossa hoidetaan laskujen hallinta, palkanmaksut, käyttäjien verotietojen ylläpito ja kirjanpito yritykselle. Toisena rajapintana toimii kolmannen osapuolen palvelu vahvaan tunnistautumiseen, jolla voidaan varmentaa käyttäjät ja vähentää väärinkäyttöä.

Lopputulokseksi toteutui sovellus kevytyrittäjän tarpeisiin, jolla luodaan ja hallitaan käyttäjiä, asiakkaita, laskuja, palkanmaksu pyynnöt sekä matka- ja kululaskuja. Kolmannen osapuolen taloushallinto-ohjelmisto hoitaa laskujen lähetyksen asiakkaille, maksujen vastaanoton, palkanmaksun, kirjanpidon ja muut jälkitoimet

Avainsanat
Kevytyrittäjä, laskutus, rajapinta, kirjanpito, Laravel

SAVONIA UNIVERSITY OF APPLIED SCIENCES THESIS
Abstract

Field of Study Technology, Communication and Transport	
Degree Programme Degree Programme in Information Technology	
Author(s) Samu Miettinen	
Title of Thesis Design and Implementation of Light Entrepreneurship Service	
Date 29 March 2022	Pages/Appendices 21
Client Organisation /Partners DuuniOnline Oy	

Abstract

The aim of this thesis was to make DuuniOnline Oy's light entrepreneurship service easier to use, faster and more mobile-friendly, as the previous version had shortcomings in these areas. The aim was to design and implement a more modern and improved version of the original implementation. The usability and visuality of this new version was to match the client organisation's visions of the software more closely.

The whole project was implemented in the Laravel 7 development environment, utilizing a third-party financial management interface, which handles invoice management, payroll, user tax information maintenance and accounting for the company. Another interface was a third-party service for strong authentication. This interface acts as an interface to authenticate users and reduce misuse.

As a result, the application for the needs of light entrepreneurs was created. This application creates and manages users, customers, invoices, salary payment requests, and travel and expense invoices. The third-party financial management software handles the sending of invoices to customers, receipt of payments, payment of salaries, accounting and other follow-up.

Keywords
light entrepreneurship, invoicing, interface, accounting, Laravel

SISÄLTÖ

KUVALUETTELO

LYHENTEET JA KÄSITTEET

Rajapinta – Eri ohjelmistojen välinen keskustelureitti, josta voi pyytää tietoja ja viedä tietoja.

Kevytyrittäjä – Laskutuspalvelun kautta toimiva ammatinharjoittaja, jolla ei ole yritystä tai työsuhdetta.

SPA (Single-Page Application) – Nettisivusto, jossa kaikki sisältö tapahtuu yhdellä sivulla, jota muutetaan dynaamisesti käyttäjän toimintojen mukaan.

URI (Uniform Resource Identifier) – Merkkijono, jolla kerrotaan tietyn tiedon paikka. URL on yhdenlainen URI-tieto.

HTML (Hypertext Markup Language) – Avoimesti standardoitu merkintäkieli, jolla nettisivut on kirjoitettu.

MVC (Model-View-Controller) – Ohjelmistoarkkitehtuuri, jonka tarkoitus on erottaa sovellus malleihin, näkymiin ja kontrollereihin.

Ohjelmistokehys – Muodostaa rungon sovellukselle, tarjoten erilaisia valmiita apuvälineitä.

Käyttöliittymä – Sovelluksen osa, jonka välityksellä käyttäjä käyttää sovellusta.

Palvelin – Tietokone, joka suorittaa palvelinohjelmistoa.

PHP (PHP: Hypertext Preprocessor) – Ohjelmointikieli, joka käytetään erityisesti web-palvelinympäristöissä

Git – Hajautettu versionhallintajärjestelmä.

BitBucket – Sivusto, josta voidaan hallita lähdekoodia ja versionhallintaa (Git).

Migraatio – Laravelin tietokannan versionhallinta.

REST (REpresentational State Transfer) – Arkkitehtuurityyli ohjelmointirajapintojen toteuttamaiseen, joka toimii HTTP- tai HTTPS-protokollan yli.

VSCode (Visual Studio Code) – Avoimen lähdekoodin tekstieditori.

1 JOHDANTO

Kevytyrittäjien määrä on räjähtänyt lähivuosina nousuun ja kysyntää on koko ajan enemmän, jolloin kevytyrittäjille tarkoitettujen laskutuspalveluiden tarjonta kasvaa, monipuolistuu, jonka takia myös näiden välinen kilpailu kovenee. Monet olemassa olevista laskutuspalveluista ovat suppeita, sekavia tai sisältävät erilaisia piilokuluja, jotka tulevat monesti kevytyrittäjälle yllätyksenä.

Opinnäytetyö on toteutettu DuuniOnline Oy:lle, jonka markkinointinimi on PalkkaOnline. Yrityksen ideana on toteuttaa mahdollisimman yksinkertainen, läpinäkyvä, edullinen ja piilokuluton laskutuspalvelu kevytyrittäjille. Tämä projekti oli aloitettu jo vuonna 2018, jolloin DuuniOnline osti ulkoisen ohjelmistoyrityksen toteuttamaan visiotaan laskutuspalvelusta, kunnes lopulta todettiin oman ohjelmistokehittäjän palkkaamisen olevan parempi vaihtoehto kyseiseen projektiin.

Kehitystyö alkoi siitä, että tutkittiin alkuperäinen järjestelmä läpi edellisten kehittäjien kanssa ja selvitettiin ohjelman puutteet sekä toimivat ratkaisut, joiden pohjalta aloitettiin kehittämään uutta järjestelmää. Tämän jälkeen aloitettiin käyttöliittymän toteutuksesta ja testauksesta, josta siirryttiin taustajärjestelmän kehitykseen ja liitettiin se käyttöliittymään. Lopuksi julkaistiin ohjelma testiryhmälle, johon opinnäytetyön aika päättyi, mutta jatkoimme silti lopulliseen julkaisuun ja uusien ominaisuuksien aikataulutukseen sekä kehitykseen tämän opinnäytetyön ulkopuolella.

Allekirjoittanut toimii DuuniOnlinessa teknologiajohtajana (CTO). DuuniOnlinen esimiehenä toimii CEO Petteri Räsänen. Kehitystyö osaltani alkoi tammikuussa 2021 ja opinnäytetyön osuus loppui elokuussa 2021.

1.1 Toimeksianto

Kuopiolainen DuuniOnline Oy, jonka markkinointinimi on PalkkaOnline. Yritys on perustettu vuonna 2019, jonka hallitukseen kuuluu neljä henkilöä ja työntekijöinä on kaksi henkilöä. Tämän opinnäytetyön alkamiseen asti yritys oli ostanut ulkoista ohjelmistokehityspalvelua, jonka jälkeen yritykseen palkattiin oma ohjelmistokehittäjä. Yrityksen visio on toteuttaa laskutuspalvelu vaihtoehto kevytyrittäjille, joka on mahdollisimman helppo, läpinäkyvä, edullinen ja piilokuluton. (PalkkaOnline)

2 KÄYTETYT TEKNIIKAT JA MENETELMÄT

2.1 Visual Studio Code

Visual Studio Code (eli VS Code) on nykyaikainen avoimen lähdekoodin tekstieditori, joka on kevyt, mutta erittäin tehokas minkä tahansa koodin muokkaukseen. Ohjelman suurin hyöty on koodin korostuksen ja värjäyksen lisäksi IntelliSense -ominaisuus, joka ennakoi haluamiasi sanoja ja voidaan hyödyntää sen valmiita koodipätkiä. Lisäksi ohjelma sisältää tuen ulkoisiin lisäosiin ja sieltä löytää paljon koodausta nopeuttavia ja helpottavia lisäosia. (Visual Studio Code)

2.2 Figma

Figma on vektorigrafiikka- ja prototyyppityökalu. Se on tarkoitettu käyttöliittymien, käyttökokemuksien sekä prototyyppien luomiseen. (Wikipedia)

2.3 BitBucket

Bitbucket on sivusto, jolla voidaan ylläpitää projektin lähdekoodia ja sen versioita. Bitbucketin taustalla toimii Git, joka hoitaa tietojen ylläpidon taustalla. Git toimii komentorivillä, mutta Visual Studio Code:n lisäosien ansiota sekin on tehty käyttäjälle todella helpoksi. Bitbucket sivuston kautta tarvittaessa voisi jakaa projektia muilla jäsenille, tehdä testejä ja seurata Git:n toimintaa kätevämmin. (BitBucket)

2.4 Laravel

Laravel on tyylikkäällä syntaksilla varustettu web-ohjelmointikehys. Laravel toimii PHP ohjelmointikielen ohjelmistokehyksenä. Laravel tarjoaa kehittäjälle valmiin paketin, jonka päälle on helppo lähteä kehittämään sovellusta. Lisäksi Laravel sisältää erinomaisen dokumentaation, jonka takia se soveltuu erinomaisesti aloitteleville koodareilla. (Otwell)

2.5 REST API

REST API on sovellusohjelmointirajapinta, joka noudattaa REST-arkkitehtuuria. REST on lyhenne sanoista REpresentational State Transfer. Sillä mahdollistetaan laitteiden ja sovellusten kommunikointi keskenään. REST-rajapinnan avulla voidaan siis lähettää tietoa laitteesta/sovelluksesta toiseen ja vastaanottaa tietoa. (Red Hat 2020)

2.6 MVC-arkkitehtuuri

Model-View-Controller eli MVC, on yleisesti käytetty arkkitehtuurimalli, jonka idea on jakaa esitys- ja logiikkakerros toisistaan. Ohjelmiston rakenne jaetaan kolmeen osaan eli malliin (Model), Näkymään (View) ja kontrolleriin (Controller). (Hurja)

Malli on ohjelmointikielellä kuvattu tieto, joka löytyy tietokannasta. Käyttäjätieto esimerkiksi voisi koostua etunimestä, sukunimestä ja syntymäajasta. Näkymä sisältää tiedon käyttöliittymästä. Yleensä malli ja näkymä ovat sidottuina toisiinsa eli, jos tieto muuttuu mallissa, niin tieto muuttuu myös näkymässä. Kontrolleri toimii tässä tiedon välittäjänä eli se ohjaa tulevan pyynnön sivustolla oikeaan malliin, joka taas välittää tiedon oikeaan näkymään. (Hurja)

2.7 Laravel Forge

Palvelimen hallintaan tehty ylläpitosivusto, joka helpottaa palvelimien hallintaa ja seurantaa. Forgella voidaan suoraan julkaista sovellus tietyillä kolmannen osapuolen palvelintarjoamissivustoilla, jollainen tässä projektissa otettiin käyttöön sen helppouden takia. Palvelu asentaa palvelinohjelmistot käyttövalmiiksi ja valmiiksi säädettynä. Palvelusta saadaan myös sertifikaatit, varmuuskopiot, tietokantapalvelin, käyttöoikeudet ja ajoitettujen tehtävien luonti. Palvelu siis nopeuttaa kehitystyötä huomattavasti, kun ei tarvitse säädellä palvelimia ja miettiä tietoturva-aukkoja. Sivusto on myös täysin Laravelin kehittäjien ylläpitämä, joten se on luotettava. (Laravel)

2.8 Single-page application

Single-page application eli SPA on nettisivu, jossa kaikki sisältö tapahtuu yhdellä sivulla, jota muutetaan dynaamisesti käyttäjän toimintojen mukaan. Normaalisti koko sivu päivittyy aina käyttäjän toiminnoista, mutta SPA muuttaa vain tarvittavia kohtia sivustolta. Sivuston kaikki tarpeellinen tieto on haettu jo silloin kun saavut sivustolle, joka myös nopeuttaa sivuston toimintaa, kun sen ei tarvitse aina hakea tietoja uudestaan. (Wikipedia)

2.9 Migraatio

Laravelissa käytetään migraatioita, jotka ovat kuin tietokannan versionhallinta. Kun migraatio ajetaan se käy järjestyksessä taulukon ja luo niiden viittaamien tiedostojen pohjalta tietokannan. Migraatioilla hoidetaan tietokannan kaikki muutokset eli lisäys, poisto ja muutokset tapahtuu kätevästi luomalla uusi migraatiotiedosto komennolla:

php artisan make:migration [migraation nimi]

Migraatioita voidaan myös kumota komennolla:

php artisan migrate:rollback

Migraatiot myös mahdollistavat tiimiläisten ottavan tietokannat ja niiden muutokset nopeasti käyttöön ilman ongelmia. (Otwell)

2.10 Laravel Blade

Laravelin mukana toimitetaan Blade, joka on yksinkertainen ja tehokas ”mallimoottori”. Blade tiedostot käyttävät tiedostopäätettä blade.php ja niitä käytetään käyttöliittymän logiikan luonnissa. (Otwell)

3 TOTEUTUS

3.1 Aloitus

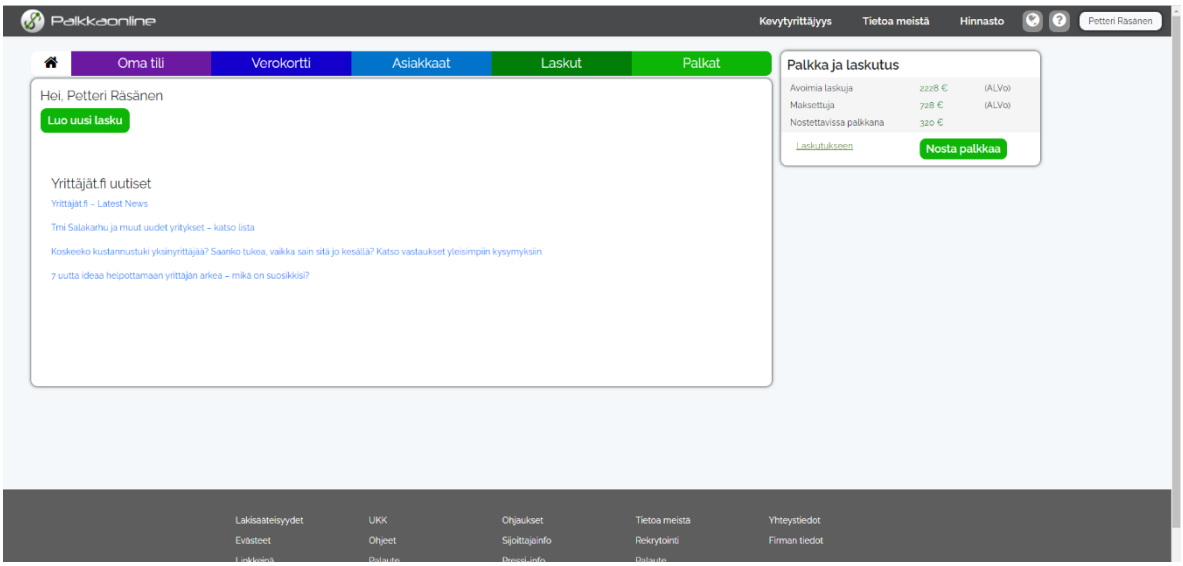
Ennen koodaamisen aloitusta valmisteltiin kehitysympäristö asentamalla kaikki tarpeelliset ohjelmat, lisäosat ja hankittiin tunnukset BitBuckettiin ja Figmaan. Heti aluksi tuli ongelmia siitä, kun projekti oli kehitetty Laravel 6 versioon ja koneellani oli väärä PHP versio ja totesimme että käytetään tilaisuus hyväksi ja päivitetään sovelluksen Laravel ja PHP uudempaan. Tietenkin myös silloinen palvelimen PHP versio täytyi päivittää.

Päivittelyiden jälkeen päästiin sitten tarkemmin katsomaan projektin tilannetta ja tutustumaan toteutettuun koodiin ja Laravel ympäristöön.

3.2 Käyttöliittymä

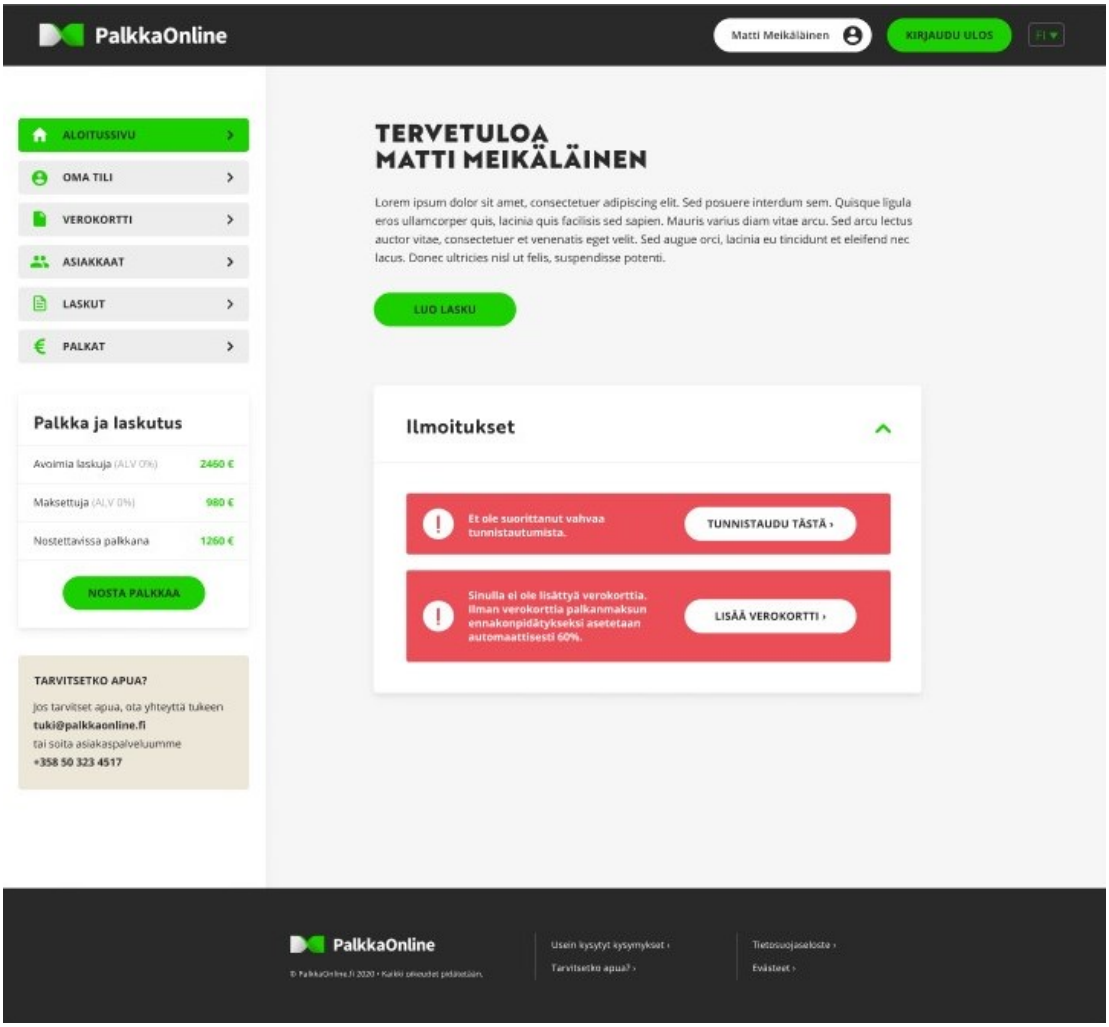
Käyttöliittymä toteutettiin ulkoisen suunnittelutoimiston toimittaman Figma suunnitelman pohjalta. Käyttöliittymässä oli tärkeää, että nappeja oli mahdollisimman vähän ja polut eri ominaisuuksiin olivat mahdollisimman lyhyitä. Mobiilikäytettävyys oli myös huomioitava, koska yli 50 % koko maailman verkkoliikenteestä tulee puhelimista. (perficient 2021; Statista 2022; Broadbandsearch)

Kun sivustoa aloitettiin kehittämään vuonna 2019 sillä ei ollut vielä tarkkaa värimaailmaa tai brändiä, joten sivustolla ei ollut kovin kummoinen ulkoasu vielä silloin (kuva 1).



Kuva 1. Vanhan sovelluksen ulkoasu. (PalkkaOnline 2020)

Suunnittelutoimiston tutkimuksen ja ulkoasu suunnitelman jälkeen uuden sivuston ilme on paljon nykyaikaisempi ja myös värejä on käytetty paremmin (kuva 2).



Kuva 2. Uuden sivuston Figma suunnitelma. (PalkkaOnline 2020)

Sivusto toteutettiin yksisivuiseksi (SPA), jolloin kaikki sivuston sisältö oli jatkuvasti saatavilla. Tämä aiheutti omat ongelmansa, mutta myös joltain osin se nopeutti kehitystä, kun ei tarvinnut siirtää tietoa kontrollerista toiselle. Sivusto toimii siis yhdellä pohjalla (ks. kuva 3), johon vaihdetaan sisältöä sivun mukaan. Pohjasta löytyy kaikki sivustolle tarpeelliset sivun toimintaan. Sivun rakenne koostuu yläpalkista, sivupalkista, sisällöstä ja alapalkista, jotka tuodaan views/layouts/app.blade.php. Yläpalkki on tuotu tiedostoon *@include('partials.header')* rivillä, sivupalkki *@include('partials.sidebar')* rivillä, vaihdettava sisältö tuodaan @yield('content') kohdassa ja alapalkki *@include('partials.footer')* kohdassa (kuva 3). Vaihdettavaa sisältöä muunnetaan kontrollereissa, josta kerrotaan myöhemmin.

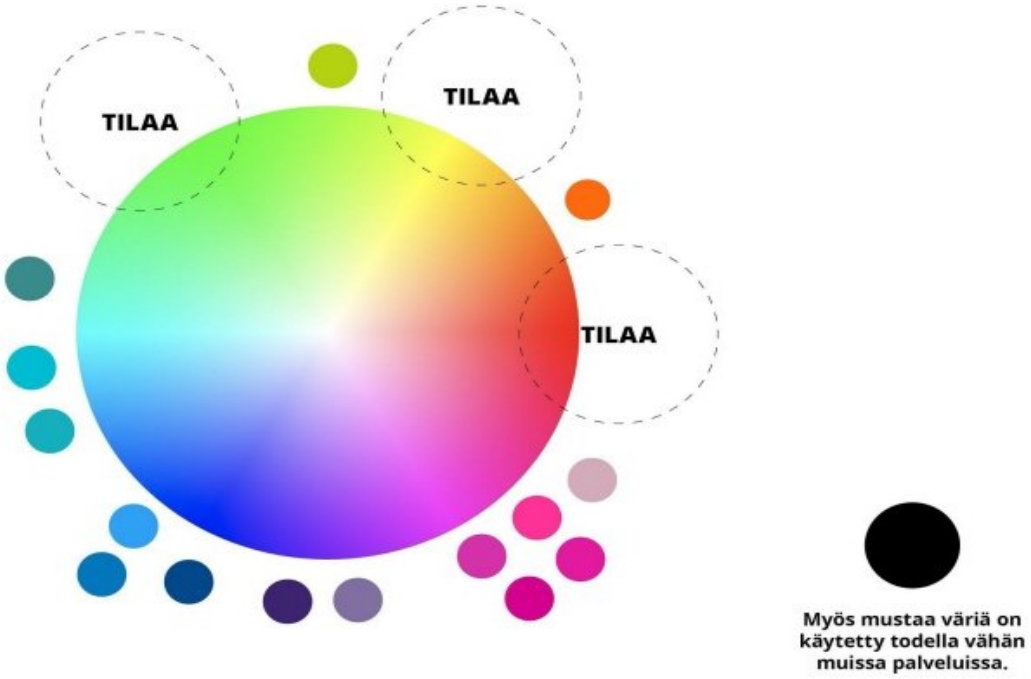
```
<body>
    @include('includes.googletagmanagerbody')
    <div id="app" class="container-fluid" style="overflow-x: hidden">
        <div class="row">
            @include('partials.header')
        </div>

        <div class="row row-sm-no-padding">
            <!-- Sidebar -->
            <div class="sidebar col-md-3">
                @include('partials.sidebar')
            </div>
            <!-- Content -->
            <div class="main col-12 col-md-9 pt-4">
                @yield('content')
            </div>
        </div>

        <div class="row">
            @include('partials.footer')
        </div>
    </div>
    <!-- Scripts -->
    @stack('scripts')

    {!! Toastr::message() !!}
</body>
```

Kuva 3. Yksisivuisen nettisivun pohjakoodi tiedostossa views/layouts/app.blade.php.

Yrityksen liikemerkin suunnitteli myös aiemmin mainittu suunnittelutoimisto, koska edellinen ei tuonut haluttua tunnelmaa. Uusi liikemerkki muodostuu kuviosta, jonka toinen puoli on käännetty ympäri paljastaen taustalla olevan vihreän värin. Liikemerkin tausta-ajatuksena on muutos, helppous, yhdistyminen ja rehellisyys. Vanhan ja uuden kuvan välillä on huomattava ero, kun edellinen on vaikeaselkoinen ja vanhanaikainen toisin kuin uusi, joka on valoisa, selkeä ja nykyaikainen (kuva 4).

Kuva 4. Liikemerkin muutos. (PalkkaOnline 2022)

Värimaailma sivustolle muodostui vertailemalla kilpailijoita. Väriympyrästä löytyi värejä joita kilpailijat eivät vielä tämän suunitelman aikana käyttäneet, joten pääväreiksemme muodostoi vihreä, harmaanmusta ja valkoinen (kuva 5). Lisäväreinä toimii punainen, keltainen ja beige.



Kuva 5. Kilpailijoiden värimaailma väriympyrällä (PalkkaOnline 2020)

3.3 MVC-komponentit

Käytetyissä menetelmissä selitettiinkin jo kyseisestä menetelmästä. Laravel siis käyttää MVC-arkkitehtuuria, joka tarkoittaa sitä, että sivuston rakenne on jaettu kolmeen osaan: malleihin, näkymiin ja kontrollereihin.

Tässä työssä aloitettiin luomalla näkymät, koska ajateltiin että niiden perusteella voidaan sitten päätellä kuinka kontrolleri ja näkymä muodostuu. Koska sivusto on SPA niin kontrollereita tehtiin jokaiselle osiolle yksi eli ylläpitäjät, käyttäjät ja vieraat. Ylläpitäjille mahdollistettiin ominaisuuksia muokata ja seurata kaikkia käyttäjiä, joten heidät erotettiin toisistaan. Vieraat taas ovat käyttäjiä, jotka eivät ole vielä kirjautuneet sisään, joten heidät pidettiin poissa käyttäjien sivuista.

3.3.1 Näkymä

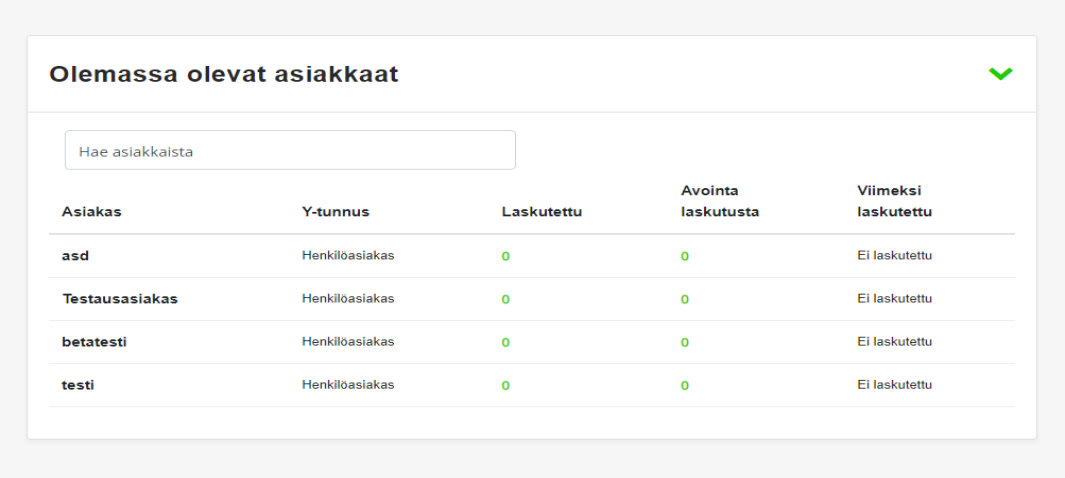
Näkymät muodostuivat Blade-tiedostoista, joilla käyttäjille saadaan näkymä. Näiden luonti ensimmäiseksi oli järkevä koska sillä pääsimme näkemään vaadittavat muutokset muihin osioihin ja saimme suoraan visuaalista näkymää tutkittavaksi, joten tiesimme että se olisi toimiva.

Asiakkaiden listaaminen näkymään tapahtui taulukkoelementin avulla, johon luotiin halutut otsikot thead-elementtiin. Otsikoita otettiin viisi kappaletta, joille pystytään näyttämään oleellisimmat tiedot käyttäjästä nopeasti. Taulukon sisältö lisättiin forelse-silmukassa, joka on sama kuin foreach-silmukka, mutta sisältää lisäksi vaihtoehdon sille, että tietueita ei löydy eli else-haaran (kuva 6).

```
<div class="table-responsive">
    <table class="table" id="clientsTable">
        <thead>
            <tr>
                <th scope="col">Asiakas</th>
                <th scope="col">Y-tunnus</th>
                <th scope="col">Laskutettu</th>
                <th scope="col">Avointa <br> laskutusta</th>
                <th scope="col">Viimeksi <br> laskutettu</th>
            </tr>
        </thead>
        <tbody id="customers">
            @forelse ($customers as $customer)
            <tr>
                <td>{{ $customer->name }}</td>
                <td>{{ $customer->business_id }}</td>
                <td>{{ $customer->invoices->sum('total_alv0') }}</td>
                <td>{{ $customer->openinvoices }}</td>
                <td>{{ $customer->lastinvoiceDate }}</td>
            </tr>
            @empty
                <p>Ei asiakkaita</p>
            @endforelse
        </tbody>
    </table>
</div>
```

Kuva 6. Asiakas näkymän sisältö.

Tavoitteena oli siis näyttää vain oleellisimmat tiedot asiakkaasta ja tarkentavat tiedot asiakkaasta piilotettiin painalluksen taakse, joka tapahtui painamalla asiakkaan riviä (kuva 7).



Kuva 7. Asiakkaat sivun näkymä.

3.3.2 Malli

Malli-komponentin tarkoitus oli kuten aiemmin oli jo mainittu kuvata koodilla tietokannan taulua. Yksinkertaisimmillaan se on vain taulun tietojen kirjaus. Tiedot, joita haluttiin pystyä muokkaamaan,

lisättiin taulukkoon nimeltä fillable (ks. kuva 8), jonka malli tiedosto tunnistaa automaattisesti tiedoiksi, jotka halutaan mahdollistaa muokattavaksi.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Customer extends Model
{
    protected $fillable = [
        'name',
        'email',
        'company_id',
        'street_address',
        'postnumber',
        'postal_district',
        'country',
    ];
}
```

Kuva 8. Asiakas malli tiedoston sisältöä.

3.3.3 Kontrolleri

Voimme kontrollerissa asiakkaiden haussa varmistaa, että käyttäjä pääsee näkemään vain omat asiakkaansa aikaisemmin luodussa listassa eikä vahingossakaan saa tietoja muista asiakkaista. Lisäksi rajoitamme, että kirjautumaton käyttäjä ei pääse näkymään vaan vaadimme kirjautuneen käyttäjän kyseisen kontrollerin käyttöön. Kuvassa 9 on yksinkertaistettu versio asiakkaiden palautuksesta, jossa palautetaan valmiiksi HTML koodiksi muokatun sisällön, jotta voin korvata edellisen sisällön tällä uudella sisällöllä.

```
public function __construct()
{
    $this->middleware('auth');
}

public function showCustomers(){
    $customers = Auth::user()->customers()->paginate(20);

    $content = view('customers')->with(['customers' => $customers])->render();
    return Response::json(array(
        'success'   => true,
        'html'      => $content,
    ))
}
```

Kuva 9. Asiakkaiden haku kontrollerissa.

3.4 Käyttäjän tunnistaminen

Sovellukseen vaadittiin vahva tunnistautuminen väärinkäytön estämiseksi ja verotietojen tarkistukseen. Vahva tunnistautuminen tapahtuu kolmannen osapuolen tarjoaman tunnistus palvelun kautta, jonne lähetämme julkisella avaimella salatun pyynnön ja vastauksena saamme salatun vastauksen, jonka voimme purkaa omalla salausavaimella sovelluksessa. Vastaus sisältää käyttäjän henkilötiedot, jota voimme sitten hyödyntää käyttäjän tunnistukseen.

Ohjelmistossa käsitellään paljon henkilötietoja tunnistamiseen, laskun luontiin ja palkan maksuun, jotka nykyisen tietosuoja asetuksen mukaan täytyy tiedottaa sivuston käyttäjille ja luoda tietosuojaseloste. Henkilötietoja ovat tiedot, joista henkilö voidaan tunnistaa esimerkiksi nimen, henkilötunnuksen tai puhelinnumeron perusteella. (Tietosuojavaltuutetun toimisto)

Jotta käyttäjämme henkilöllisyys ja muut tiedot eivät ole ulkopuolisten saatavilla ne täytyy tallentaa tietokantaan turvallisesti ja rajoittaa sinne pääsy vain välttämättömille henkilöille. Lisäksi henkilötunnuksen väärinkäytön estämiseksi se on salattu tietokantaan salausalgoritmilla niin että sitä ei voi hyödyntää enää myöhemmin uudelleen.

3.5 Kolmannen osapuolen taloushallinto rajapinta

Jotta sovelluksen ei tarvitsisi olla täysi kirjanpito-ohjelmisto hyödynnämme kolmannen osapuolen ohjelmistoa ja sen laajoja ominaisuuksia ja vuosien kehitystä rajapinnan avulla. Käytännössä kaikki sovelluksesta tuleva tieto viedään rajapinnan yli jossain vaiheessa. Kuten vahvassa tunnistautumisessa niin tässäkin meidän täytyy todentaa itsemme, että olemme oikeutettuja kyseisen rajapinnan käyttöön julkisilla ja yksityisillä salausavaimilla.

Kun käyttäjä luo tunnukset siitä tieto tallennetaan tietokantaan odottamaan tunnistautumista ja loppujen tietojen täyttämistä. Käyttäjän täytettyä loput tiedot lähetämme rajapintaan tiedon uudesta palkansaajasta (ks. kuva 10). Tallennettava tieto viedään aina XML-muodossa, joka on merkintäkielen standardi, jonka ansiota sitä on helppo käsitellä missä vain ohjelmistossa.

```
<root>
    <employee>
        <employeebaseinformation>
            <employeeidentifier>'. $finnishId .'</employeeidentifier>
            <firstname>'. $firstname .'</firstname>
            <lastname>'. $lastname .'</lastname>
            <phonenumber>'. $phone .'</phonenumber>
            <email>'. $email .'</email>
        </employeebaseinformation>
```

Kuva 10. Palkansaajan vienti rajapintaan.

Rajapintaan tuotava tieto, vaatii muutamasta rivistä kymmeniin, ellei jopa sataan riviin tietoa. Vaikka kolmannen osapuolen ohjelmisto on tehnyt erittäin laajan ja hyvän dokumentaation, niin silti joissain vaiheissa joutui päättelemään mitä tieto tarkoittaa, mutta suurin osa on selkeästi dokumentoitua tai helposti pääteltävissä kentän nimestä.

Kolmannen osapuolen taloushallinto-ohjelmistoon viedään esimerkiksi palkansaajat, asiakkaat ja laskut. Voimme myös rajapinnasta hakea tiedon siitä onko esimerkiksi lasku maksettu. Tarvittaessa voidaan myös vertailla sovelluksen ja rajapinnan lukuja toisiinsa, jotta tiedetään että kaikki summat täsmäävät.

3.6 Tietokanta

Tietokanta luodaan migraatioilla, joista aiemmin oli mainittu luvussa 2.9. Migraatiot olivat siis tietokannan versionhallinta, jotka ovat tehokas keino kehittää tietokantoja. Migraatiot luodaan siten että ne sisältävät kaksi metodia: up ja down. Up -metodiin lisätään koodi, joka tehdään, kun migraatio suoritetaan ja down -metodiin lisätään koodi, jolla voidaan kumota up -metodi eli esimerkiksi jos tauluun lisätään kenttä up -metodissa, niin down -metodissa kenttä poistetaan (kuva 11).

Laravelissa on valmiiksi taulut kirjautumiselle, mutta niissä ei ollut vahvaa tunnistautumista vaativaa kenttää. Tiedon voi kätevästi lisätä luomalla uusi migraatio, johon kerrotaan mitä taulukkoa käsitellään ja kertomalla siihen millainen uusi tieto on kyseessä. Lisäsimme käyttäjät tauluun henkilötunnusta kuvaavan tiedon ja mahdollistettiin nullable -metodilla sen tyhjänä oleminen, jolloin sitä ei vaadita vielä rekisteröinti vaiheessa (kuva 11).

```
public function up()
{
    Schema::table('users', function (Blueprint $table) {
        $table->string('ssn')->nullable(true);
    });
}

public function down()
{
    Schema::table('users', function (Blueprint $table) {
        $table->dropColumn('ssn');
    });
```

Kuva 11. Henkilötunnuksen lisäyksen migraatio tiedosto.

3.7 Reitit

Laravel-reitit luodaan routes/web.php tiedostoon, jolla mahdollistetaan eri kontrollereiden ja näkymien näyttämisen. Reitit käyttävät yksinkertaisimmillaan URI:n ja palauttaa jonkin arvon, mutta yleisemmin reiteissä annetaan URI ja kontrolleri ja kyseisen kontrollerin metodi, jota kyseisellä reitillä halutaan suorittaa. Yleensä jokaiselle sivulle tulee oma kontrolleri, joka hoitaa kyseisen sivun toiminnot, mutta kyseisessä projektissa käytettiin vain muutamaa kontrolleria, joka myöhemmässä vaiheessa osoittautui ongelmaksi.

Reitti luodaan alkuun kertomalla reitin tyyppi eli get tai post, jonka jälkeen lisätään URI-osoite ja lopuksi lisätään kontrollerin nimi ja sen metodi, jota käytetään. Joissain tilanteissa myös voidaan lisätä name -metodilla nimi helpottaakseen reitin käyttöä (ks. kuva 12). Kuvassa 12 on vain osa sovelluksen lopullisista reiteistä.

```
$router->group(['middleware' => 'auth'] , function() {
    Auth::routes();
    Route::get('/', 'HomeController@index')->name('home');
    Route::get('/home', 'HomeController@index')->name('home');
    Route::post('/getClientsList', 'HomeController@getClientsList');
    Route::post('/addTravel', 'HomeController@addTravelToDatabase');
    Route::post('/addExpense', 'HomeController@addExpenseToDatabase');
    Route::post('/sendInvoice', 'HomeController@sendInvoice');
    Route::post('/createCustomer', 'HomeController@createCustomer');
    Route::post('/sendPayments', 'HomeController@sendPayments');
    Route::post('/saveCustomer', 'HomeController@saveCustomer');
    Route::post('/getClientInfo', 'HomeController@getClientInfo');
    Route::get('/api/securelogin', 'SecureLoginController@SecureLogin');
    Route::get('/api/securelogout', 'SecureLoginController@SecureLogout');
    Route::post('/uploadTaxFile','HomeController@uploadTaxCard');
    Route::post('/changeemail', 'ProfileController@changeEmail');
    Route::post('/changepassword', 'ProfileController@changePassword');
    Route::get('/downloadReceipt/{file}', 'HomeController@downloadFile');
    Route::post('/GetUserFeesReport', 'HomeController@GetUserFeesReport');
    Route::post('/GetUserBalanceReport', 'HomeController@GetUserBalanceReport');
    Route::get('/downloadPDF','HomeController@downloadPDF');
    Route::post('/getInvoiceInfo', 'HomeController@getInvoiceInfo');
});
```

Kuva 12. Sovelluksen reitityksiä routes/web.php tiedostossa.

4 POHDINTA

Opinnäytetyön tavoite oli toteuttaa nykyaikaisempi ja toimivampi ratkaisu vanhan sovelluksen tilalle. Onnistuimme toteuttamaan uuden brändin mukaisen sovelluksen, joka on helppokäyttöinen kilpailijoihin verrattuna. Asettamamme tavoitteet projektin alussa toteutettiin aikataulussa. Vaikka projektin ohjelmistokehys Laravel ja sen lisäosat blade, migraatiot ja muut olivat tekijälle täysin uusia asioita niin aikaisemman koodausosaamisen ja kokemuksen ansiota niistä pääsi nopeasti perille.

Jatkuva yhteydenpito sovelluksen tilaajan kanssa nopeutti myös kehitystä, kun huomattiin nopeasti toimimattomat ratkaisut ja muutokset saatiin tehtyä heti, eikä katselmoinnin jälkeen.

Opinnäytetyön ansiota tekijällä on nyt paljon itsevarmempi olo kehittää isompiakin kokonaisuuksia ja antoi valmiudet full-stack kehittäjäksi. Työn aikana oppi laajasti erilaisia ohjelmistokehitykseen liittyviä asioita, joita tavallisissa harrastus projekteissa ei välttämättä ole tarvinnut huomioida. Esimerkiksi kommentoinnin ja dokumenttien tärkeys, rajapintojen ja lisäosien hyödyntäminen tehokkaasti.

Yritykselle kehitystyö oli tärkeä, jotta päästiin markkinoimaan sovellusta ja saatiin liiketoiminta pyörimään kunnolla.

5 JATKOKEHITYS

Jo opinnäytetyön kehityksen alussa tiedostimme, että tärkeintä on saada sovellus toimintaan asiakkailleen eikä tarkoitus ollut yrittää opinnäytetyön lyhyessä ajassa valmistaa koko sovellusta valmiiksi. Opinnäytetyöajan loputtua olemmekin jo kehittänyt järjestelmää hurjasti eteenpäin ja lisännyt ominaisuuksia esimerkiksi raportointiin, matka- ja kulukorvauksien laskutukseen, palkkojen pikamaksuihin. Lisäksi ylläpitäjien sivustoa on kehitetty käytännöllisemmäksi ja sovellus ajaa esimerkiksi joka kuukausi raportit, josta selviää kuluneen kuukauden kehitys.

Tässä vaiheessa huomasimmekin, että pidemmän päälle tämän pohjan päälle koodaaminen ei ole toimiva ja tehokas ratkaisu, koska edellisen järjestelmän pohjakoodi oli alusta asti vanhentunut versio, joka sisälsi paljon lisäosia, jotka eivät enää olleet käytössä ja niiden päivitys on vaikeaa tai mahdotonta. Myöskään kaikki ratkaisut, jota sovellukseen kehitettiin eivät olleet parhaimpia mahdollisia. Tulevaisuudessa tämä olisi näkynyt rikkoutuneina ominaisuuksina, hidastuneena kehityksenä ja pahimmillaan virheilmoituksia tai sivuston kaatumisen aiheuttavia ongelmia. Aloitimmekin vuoden vaihteessa kokonaan uudelta pohjalta kehittämään tätä sovellusta paljon suuremmin kuin alkuperäisen koskaan tarkoitus oli olla. Uudessa järjestelmässä onkin Laravel 8 ja käyttöliittymiä kehitetään Livewire:llä ja tietokantaa hoitaa Laravelin oma Eloquent ORM. Uuteen järjestelmään myös toteutetaan myös PWA, joka on natiivin mobiilisovelluksen ja nettisivun välimuoto, jossa saamme molemmista parhaimmat puolet käyttöön. Tulevaisuudessa saatamme luoda oman rajapintaa hyödyntävän täysin itsenäisen mobiilisovelluksen, mutta sitä harkitaan vasta kun järjestelmä on mielestämme ”valmis”.

LÄHTEET

PalkkaOnline. Haettu 12.3.2022 osoitteesta https://palkkaonline.fi/

Visual Studio Code. Haettu 12.3.2022 osoitteesta https://code.visualstudio.com/docs

Wikipedia. Haettu 12.3.2022 osoitteesta https://en.wikipedia.org/wiki/Figma_(software)

Wikipedia. Haettu 14.3.2022 osoitteesta https://en.wikipedia.org/wiki/Single-page_application

Red Hat. What is a REST API? Haettu 12.3.2022 osoitteesta https://www.redhat.com/en/topics/api/what-is-a-rest-api

Laravel. Haettu 14.3.2022 osoitteesta https://forge.laravel.com/

Otwell, T. Laravel. Haettu 12.3.2022 osoitteesta https://laravel.com

Otwell, T. Laravel Haettu 13.3.2022 osoitteesta https://laravel.com/docs/7.x/blade

Otwell, T. Laravel Haettu 14.3.2022 osoitteesta https://laravel.com/docs/7.x/migrations

BitBucket. Haettu 12.3.2022 osoitteesta https://bitbucket.org/product/

Trajectory. Haettu 13.3.2022 osoitteesta https://www.trajectorywebdesign.com/blog/web-design-color-psychology/

Hurja. Haettu 14.3.2022 osoitteesta https://www.hurja.fi/blogi/mvc-for-dummies-malli-nakyma-ja-ohjain-arkkitehtuuri-web-sovelluksissa/

Broadbandsearch. Haettu 13.3.2022 osoitteesta https://www.broadbandsearch.net/blog/mobile-desktop-internet-usage-statistics

Statista 2022. Haettu 13.3.2022 osoitteesta https://www.statista.com/statistics/277125/share-of-website-traffic-coming-from-mobile-devices/

Perficient 2021. Haettu 13.3.2022 osoitteesta https://www.perficient.com/insights/research-hub/mobile-vs-desktop-usage

Tietosuojavaltuutetun toimisto. Haettu 19.4.2022 osoitteesta https://tietosuoja.fi/tietosuojalaki