Opinnäytetyö (AMK)

Tietojenkäsittelyn koulutusohjelma

Sähköisen liiketoiminnan järjestelmät

2015

Henri Rauhala

VARAOSASOVELLUKSEN SUUNNITTELU JA TOTEUTUS ASP.NET-YMPÄRISTÖSSÄ

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittely | Sähköisen liiketoiminnan järjestelmät

Toukokuu 2015 | 27

Päivi Nygren

Henri Rauhala

VARAOSASOVELLUKSEN SUUNNITTELU JA TOTEUTUS ASP.NET-YMPÄRISTÖSSÄ

Tässä opinnäytetyössä kerrotaan varaosasovelluksen suunnittelusta ja toteutuksesta ASP.NET-ympäristössä Saides Engineering Oy:n asiakkaalle. Kyseisellä asiakkaalla oli ennen projektin aloitusta aiempi versio web-pohjaisesta varaosasovelluksesta, jota haluttiin nyt kehittää kauppapaikan suuntaan. Opinnäytetyössä käydään läpi uuden sovelluksen suunnittelua, ominaisuuksia sekä sovelluksen ohjelmointiratkaisuja.

Projektin kehitystiimi koostui itsestäni sekä Saides Engineering Oy:n toimitusjohtajasta, joka asiantuntijana vastasi sovelluksen ominaisuuksista sekä käyttöliittymästä. Varaosasovelluksen toteuttamiseen käytettiin ASP.NET WebForms-arkkitehtuuria sekä VB.NET-ohjelmointikieltä. Varaosasovellus kehitettiin Visual Studiolla ja tietojen varastoinnissa hyödynnettiin SQL Serveriä.

Projektin lopputuloksena syntynyt sovellus on hyväksytty ja otettu käyttöön asiakkaalla.

ASIASANAT:

ASP.NET, sovelluskehitys, varaosa

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Business Information Technology | e-Business Systems

May 2015 | 27

Päivi Nygren

Henri Rauhala

DESIGNING AND IMPLEMENTING SPARE PARTS APPLICATION IN ASP.NET ENVIRONMENT

This thesis describes the design and implementation process of a spare parts application which was developed for a customer of Saides Engineering Oy. The previous version of this application was in use before this project started but the customer wished to develop the application further into a more store-like application. The thesis goes through the design process, features and programming solutions of the new spare parts application.

The project development team consisted of the author and the CEO of Saides Engineering Oy, who, as a professional in this field, was in charge of application's features and user interface. The spare parts application was created by using ASP.NET WebForms architecture and VB.NET programming language. The spare parts application was developed by using Visual Studio and SQL Server for data storage.

The application designed as the final result of this project has been approved by the customer and deployed to the production environment.

KEYWORDS:

ASP.NET, application development, spare part

SISÄLTÖ

KUVAT

KUVIOT

1 JOHDANTO

Saides Engineering Oy päätti lähteä jatkokehittämään eräälle asiakkaalle tehtyä varaosasovellusta. Aiempi varaosasovellus toimi asiakkaan toimittamien laitteiden rakenteiden hakusovelluksena, josta asiakkaan asiakas, eli loppukäyttäjä, sai haettua ja katsottua oman laitteensa rakenteen. Tätä ideaa lähdettiin kehittämään siten, että loppukäyttäjä pystyisi sovelluksen avulla kertomaan laitteen valmistajalle, eli Saides Engineering Oy:n asiakkaalle, kiinnostuksensa jonkin laitteen varaosasta tekemällä siitä tarjouspyynnön.

Uuden varaosasovelluksen kehitystiimi koostui itsestäni sekä Saides Engineering Oy:n toimitusjohtajasta, joka jo pitkään alalla työskennelleenä ja kyseisen alan (laitesuunnittelu ja –valmistus) tuntevana teki suurimman osan sovelluksen käyttöön liittyvistä päätöksistä, kuten ominaisuuksista sekä käyttöliittymästä.

Uudessa varaosasovelluksessa käytetään samaa tekniikkaa kuin aiemmassakin versiossa eli ASP.NET WebForms-arkkitehtuuria sekä VB.NET-ohjelmointikieltä, joskin uusi versio sovelluksesta rakennettiin täysin alusta alkaen uudelleen. Sovelluksen kehityksessä käytettiin Visual Studiota sekä sen mahdollistamia ominaisuuksia, kuten testausta helpottavia työkaluja. Aiempi versio sovelluksesta käytti käyttäjätietojen säilömiseen Active Directory-käyttäjätietokantaa sekä tietojen hakuun SQL Serveriä. Uuden sovellusversion tarkoituksena oli käyttää vain SQL Serveriä niin tietojen hakuun kuin talletukseenkin.

Tässä opinnäytetyössä kuvataan uuden varaosasovelluksen kehitysprosessia, ominaisuuksia sekä tehtyjä ohjelmointiratkaisuja. Alussa projekti esitellään ja kerrotaan sovelluksen keskeisistä asioista, kuten laitteen moduulipohjaisuudesta. Esittelyn jälkeen käydään läpi sovelluksen kehitysversiot sekä testaaminen. Lopussa käydään läpi projektiin liittyviä ja vaikuttaneita asioita sekä pohditaan, miten asiat olisi voitu tehdä toisin.

2 PROJEKTIN ESITTELY

Saides Engineering Oy:n asiakkaalle toteutetun varaosasovelluksen tarkoituksena oli helpottaa moduulipohjaisten laitteiden varaosamyyntiä web-pohjaisen käyttöliittymän avulla. Web-sovellus pohjautuu tietokantaan, jota käytetään laitteiden sekä varaosien tietojen säilömiseen. Web-sovelluksesta loppukäyttäjä löytää sarjanumeron perusteella itselleen räätälöidyn laitteen osarakenteen sekä laitteeseen liittyvät tiedot ja dokumentit. Näiden tietojen avulla loppukäyttäjä pystyy löytämään juuri omaan laitteeseensa sopivan varaosan. Prosessi helpottaa loppukäyttäjän sekä laitteen valmistajan välistä vuoropuhelua varaosien osalta, kun loppukäyttäjä tietää tarvitsemansa varaosan numeron.

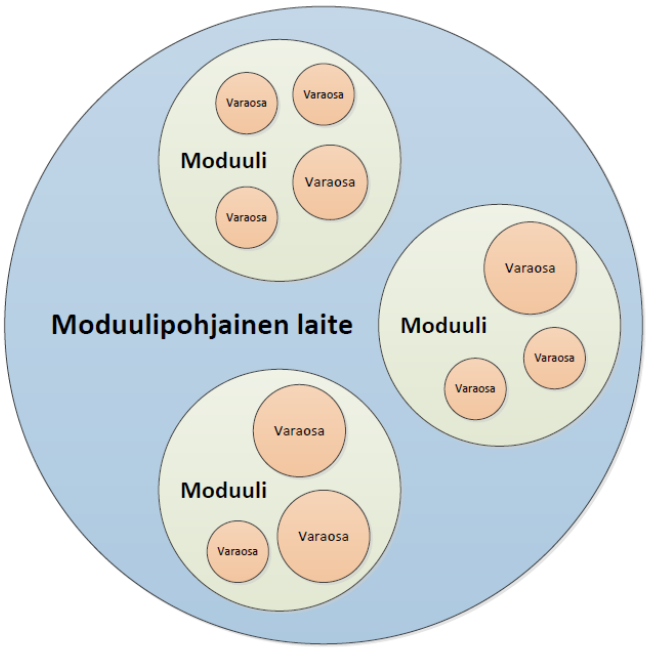
Sovelluksen käyttöliittymän sekä toimintojen suunnittelusta vastasi Saides Engineering Oy:n toimitusjohtaja, joka ohjasi ideansa minulle, ja minä parhaani mukaan toteutin ideat sovelluksessa. Omat vastuuni koostuivat ohjelmoinnin lisäksi erilaisten ohjelmointiratkaisujen etsinnästä ja vertailusta, sovelluksen testauksesta, sovelluksen ylläpidosta sekä tietokantaan liittyvistä asioista.

2.1 Moduulipohjainen laite

Moduulipohjainen laite, joka tunnetaan myös nimellä modulaarinen laite, koostuu eri moduuleista eli osakokonaisuuksista. Moduuli koostuu osista, joiden määrä ei ole vakio vaan se voi vaihdella muutamasta osasta useisiin tuhansiin. Moduulin rakennetta selventää kuvio 1. Moduulit pohjautuvat usein johonkin yleiseen standardiin, mikä mahdollistaa moduulin korvaamisen samankaltaisella moduulilla. Moduulipohjaiset laitteet myös suunnitellaan usein siten, että moduulien vaihto laitteeseen on mahdollista.

Moduulipohjaisia laitteita on monenlaisia ja niitä on kaikkialla, vaikka kyseistä sanaa ei juuri koskaan käytetä arkikielessä näistä laitteita. Moduulipohjaisia laitteita ovat muun muassa auto, moottoripyörä, metsänosturi, matkapuhelin ja polkupyörä. Polkupyörä koostuu muun muassa pyöristä, ohjaustangosta, penkistä ja tavaratelineestä. Näistä kaikista voidaan käyttää nimitystä moduuli, kos-

ka ne ovat omia osakokonaisuuksiaan, mutta yhdessä muiden moduulien kanssa ne muodostavat toimivan laitteen eli polkupyörän. Kaikki pyörän moduulit ovat vaihdettavissa samanlaisiin tai toisistaan hieman eriäviin moduuleihin.



Kuvio 1. Moduulipohjaisen laitteen rakenne.

Moduulipohjaisuudesta on asiakkaalle valtava hyöty, kun asiakas voi koota juuri halutunlaisen laitteen eri moduuleista. Tämä tarkoittaa sitä, että asiakkaalle tehty laite voi olla ainutlaatuinen, mistä johtuen laitteen valmistajalla täytyy olla tarkka rakennelista jokaisesta toimitetusta laitteesta. Ilman kyseistä rakennelistaa varaosien löytäminen tiettyyn laitteeseen on vaikeaa.

Oli kyseessä moduulipohjainen laite tai ei, laitteen rikkoontuessa mikään ei estä laitteen omistajaa käyttämästä rikkoutuneen osan tilalla halvempaa piraattiosaa. Kyseinen piraattiosa luultavasti on laadultaan huonompi kuin alkuperäinen osa laitteen valmistajalta, mutta se kuitenkin houkuttaa laitteen omistajaa hintansa puolesta. Tämä on yksi ongelma, johon tehdyn varaosasovelluksen oli tarkoitus puuttua. Ideana oli tehdä varaosan tilaamisesta niin helppoa ja varmaa, että laitteen omistajan ei kannata lähteä etsimään halvempaa piraattiosaa, kun muu-

tamalla napinpainalluksella saa laitteen valmistajalta osan, joka on juuri ko. laitteeseen sopiva.

2.2 Lähtötilanne

Projektin lähtökohtana oli aiempi versio varaosasovelluksesta, joka sisälsi Active Directory -käyttäjätietokantaan pohjautuvan kirjautumisen, laitehaun, osalistauksen sekä hakuun liittyvän kielivalinnan. Eräs tämän version ongelmista oli juuri uuden asiakkaan lisääminen, joka piti tehdä aina sovelluksen ylläpitäjän toimesta.

Aiempi varaosasovellus oli ollut käytössä asiakkaalla noin vuoden ajan, kun sovellusta lähdettiin uudistamaan. Aiempi versio sovelluksesta oli tehty Visual Basic -ohjelmointikielellä ASP.NET-ympäristöön WebForms-mallia hyödyntäen. Järjestelmä käytti tietojen hakuun sekä säilömiseen Microsoft SQL Serverin 2008 R2 –versiota. Projektissa käytettävää alustaa tai tekniikkaa ei haluttu muuttaa hyvien kokemusten perusteella.

2.3 Tavoite

Sovelluksen tarkoitus oli helpottaa varaosien myyntiä ja näin kasvattaa laitteen valmistajalta tilattavien varaosien osuutta koko varaosamyynnistä. Ideana oli tuoda varaosamyyntiin lisää varmuutta, mikä kannustaisi laitteen omistajia tilaamaan varaosat suoraan laitteen valmistajalta. Sovelluksen tuli tarjota mahdollisimman paljon tietoa laitteista, jotta kaikissa vikatilanteissa voitaisiin ensin turvautua tarjottuihin dokumentteihin ja laitteen rakennetietoihin. Tämä myös lisäisi varaosasovelluksen käyttöä ja näin muistuttaisi sovelluksen käyttäjiä varaosien tilaamisen mahdollisuudesta.

Varaosasovellusta oli tarkoitus lähteä kehittämään vanhan version pohjalta palveluksi, jota voitaisiin tarjota useille asiakkaille. Tämä vaatisi sovelluksen muuttamista siten, ettei valmiin sovelluksen rakennetta tarvitsisi muuttaa jokaisen uuden asiakkaan kohdalla.

2.4 Vaatimusmäärittely

Vaatimusmäärittelyssä otettiin huomioon aiemmasta varaosasovellusversiosta poisjääneet ominaisuudet, joita kyseiseen versioon ei ehditty toteuttaa. Projektin alussa oli jo idea siitä, mitä muitakin ominaisuuksia mahdollisesti voitaisiin toteuttaa uuteen sovellusversioon, mutta mitään ei kuitenkaan lyöty vielä lukkoon projektin alkaessa. Sovelluskehityksessä ei lopulta pidetty mitään kiirettä, mikä johti siihen, että suurin osa vaatimuksista varaosasovellukselle syntyivät iteratiivisesti sovelluksen kehittyessä.

Varaosasovelluksen uuteen versioon haluttiin sisällyttää uudenlainen kirjautuminen, joka mahdollistaisi myös loppukäyttäjän omasta toimesta tapahtuvan rekisteröitymisen. Rekisteröityminen olisi kaikille avoin, mutta jokainen rekisteröityminen pitäisi sovelluksen ylläpitäjän toimesta hyväksyä ennen loppukäyttäjän päästämistä käyttämään sovellusta. Uudenlainen rekisteröityminen mahdollistaisi tarkempien käyttäjätietojen keräämisen, mikä parantaa laitteen valmistajan asiakastuntemusta ja samalla kehittää sovelluksen turvallisuutta, kun tiedetään, kuka sovellusta käyttää.

Suurin muutoskohde oli tarve varaosasovelluksen muuttamisesta tietynlaiseksi kauppapaikaksi, jossa loppukäyttäjä voisi kerätä laitteen varaosia koriin ja tehdä niistä tarjouspyynnön laitteen valmistajalle. Varsinaista tilausta sovelluksella ei ollut tarkoitus tehdä, koska sovelluksen loppukäyttäjät tulisivat suurimmaksi osaksi olemaan laitteen valmistajan jälleenmyyjiä tai muita yhteistyökumppaneita, jolloin tilauksen lopullinen hinta ja maksutavat määräytyisivät loppukäyttäjäkohtaisesti.

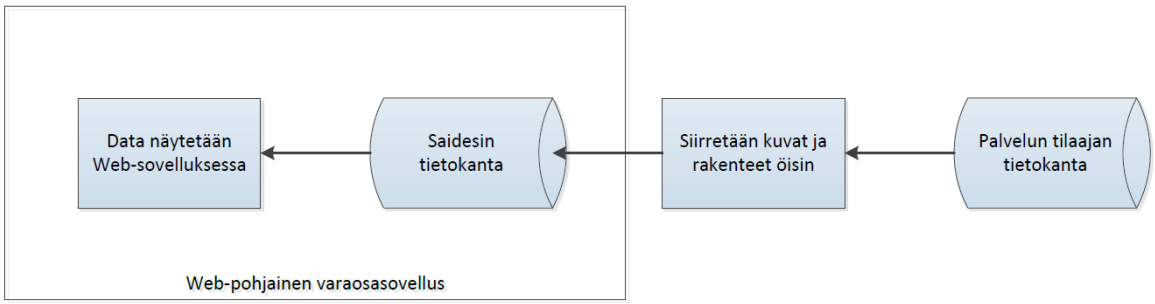
Uuteen sovellusversioon haluttiin myös mahdollisuus rajoittaa loppukäyttäjien tiedonsaantioikeuksia. Käytännössä tämä estäisi loppukäyttäjiä näkemästä laitteita tai osarakenteita loppukäyttäjälle määriteltyjen oikeuksien mukaisesti. Oikeus voitaisiin antaa joko yhteen laitteeseen tai useita laitteita sisältävään ryhmään. Loppukäyttäjä kykenisi näkemään vain esimerkiksi oman laitteensa tiedot ja näin muiden laitteisiin liittyvät tiedot jäävät näkemättä vaikka toisen laitteen laitenumero olisikin tiedossa.

3 JÄRJESTELMÄN TOTEUTUS

Projekti aloitettiin vanhan sovelluksen kartoituksella. Vanhaa sovellusta ei ollut tarkoitus käyttää uuden pohjana, mutta mahdollisimman paljon vanhasta versiosta haluttiin kuitenkin hyödyntää. Lopputuloksena vanhasta sovellusversiosta jäi jäljelle muutama SQL-kysely sekä joitakin ohjelmointifunktioita. Koko muu sovellus hylättiin ja uutta versiota lähdettiin rakentamaan puhtaalta pöydältä.

Alussa ei ollut täyttä varmuutta millainen uusitusta sovelluksesta tulisi, minkä vuoksi tarkkoja suunnitelmia ei tehty. Tämä mahdollisti nopeiden suunnanvaihdosten tekemisen projektin edetessä, mutta samalla vaikeutti kokonaisuuden hahmottamista, kun suunnitelmat saattoivat vaihtua yön aikana. Sovellusta kehitettiin rauhallisesti ja tulevista ominaisuuksista keskusteltiin jatkuvasti, mikä helpotti kokonaisuuden hallintaa. Sovellus kehittyi nopeasti, kun keskityttiin vain pieniin ominaisuuskokonaisuuksiin kerrallaan, mutta samalla useita ominaisuuksia jouduttiin muokkaamaan tai tekemään uudelleen useita kertoja.

Ideana oli jo kehitysvaiheessa kiinnittää huomiota sovelluksen toimintojen soveltuvuuteen eri tilanteisiin eri asiakkailla. Lopputuloksen haluttiin mukautuvan hyvin erilaisiin tarpeisiin. Tämä vaati useiden asetusten tallentamisen sovelluksen ulkopuolelle. Asetuksia muuttamalla voitaisiin sivustosta tehdä asiakkaan mieltymysten mukainen ja sovellus osaisi mukautua näihin muutoksiin automaattisesti. Esimerkiksi tietojen siirto sivustolle pitäisi tapahtua ongelmitta useissa eri tapauksissa. Tietojen siirto -logiikkaa selventää kuvio 2.

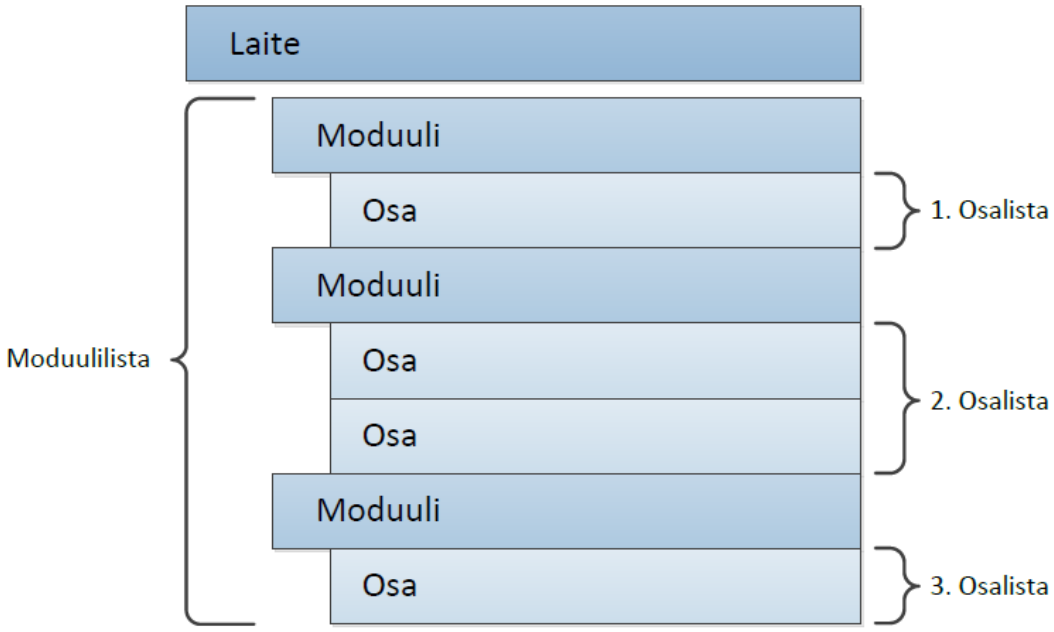


Kuvio 2. Sovelluksen tiedonkulku.

3.1 Ensimmäinen kehitysversio

Ensimmäiseen kehitysversioon oli tarkoitus saada vähintään kaikki alkuperäisen sovelluksen sisältämät ominaisuudet lukuun ottamatta kirjautumista ja kielivalintaa. Tämä käytännössä käsitti vain laitehaun sekä laitteen rakenteen listauksen.

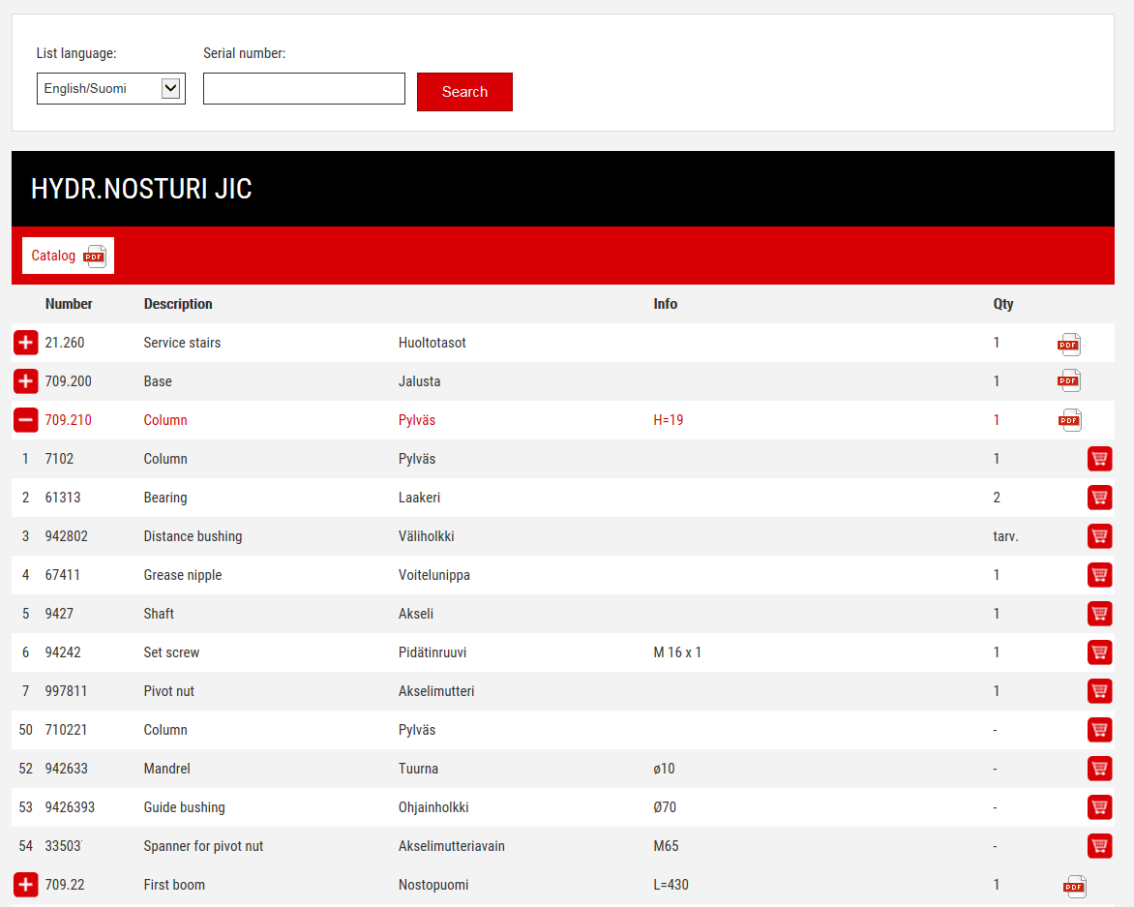
Laitteiden rakenne on kaksitasoinen ja käsittää laitteen moduulien sekä moduulien osien rakenteen. Laiterakenteen ensimmäinen taso eli laitteen moduulit listautuvat laitehakuun syötetyn laitenumeron perusteella. Toinen taso eli osat listataan moduulien alle moduulinumeron perusteella. Tasojen rakennetta selventää kuvio 3.



Kuvio 3. Laitteen rakennelistaus.

Laitteen rakennelistauksen tekemiseen käytettiin ListView- oliota (Microsoft Developer Network 2014a). Kyseinen olio on tarkoitettu listan näyttämiseen, mistä johtuen se sisältää jo valmiiksi suuren määrän eri ominaisuuksia, joiden tarkoituksena on helpottaa listan tekoa. Moduulilistaus on yksinkertainen lista, mutta moduulien osien rakenne vaatii niin sanotun sisäkkäisen listan (Dumond 2009), jossa ListView-olio pitää upottaa toisen ListView-olion sisään. Tässä tapauk-

sessa moduulit sisältävään ListViewiin upotettiin jokaista moduulia kohden uusi osalista, joka sisälsi kyseessä olevan moduulin rakenteen. Näin saatiin aikaan lista, josta näkyi laitteen koko rakenne. Rakennelistausta selventää kuva 1.



	Number	Description		Info	Qty
+	21.260	Service stairs	Huoltotasot		1
+	709.200	Base	Jalusta		1
−	709.210	Column	Pylväs	H=19	1
1	7102	Column	Pylväs		1
2	61313	Bearing	Laakeri		2
3	942802	Distance bushing	Väliholkki		tarv.
4	67411	Grease nipple	Voitelunippa		1
5	9427	Shaft	Akseli		1
6	94242	Set screw	Pidätinruuvi	M 16 x 1	1
7	997811	Pivot nut	Akselimutteri		1
50	710221	Column	Pylväs		-
52	942633	Mandrel	Tuurna	ø10	-
53	9426393	Guide bushing	Ohjainholkki	Ø70	-
54	33503	Spanner for pivot nut	Akselimutteriavain	M65	-
+	709.22	First boom	Nostopuomi	L=430	1

Kuva 1. Laitteen rakennelistaus.

Rakennelistausta rajoitettiin vain yhden moduulin rakenteen näyttämiseen kerrallaan. Näin saatiin listasta selkeämpi ja samalla säästetään palvelimen laskentatehoa sekä loppukäyttäjän internet-yhteyttä.

3.2 Toinen kehitysversio

Toiseen kehitysversioon oli tarkoitus lisätä ensimmäisestä kehitysversiosta poisjäänyt rekisteröityminen sekä kirjautuminen. Lisäksi haluttiin rakentaa ominaisuus, joka mahdollistaisi eri osien lisäämisen ostoskoriin ja tarjouspyynnön

tekemisen. Yritysten välisessä kaupassa hinnat usein neuvotellaan erikseen, mikä tarkoitti, ettei tarvetta maksujärjestelmälle ollut. Sovelluksen tuli vain ilmoittaa molemmille osapuolille tarjouspyynnön tekijä, sisältö sekä aika.

3.2.1 Sähköpostiviestit

Sovelluksen toisen kehitysversion ominaisuudet vaativat toimiakseen useiden sähköpostivarmistusten lähettämistä eri tilanteissa eri henkilöille. Sovellus lähettää sähköpostit käyttämällä .NET:in omaa MailMessage-luokkaa (Microsoft Developer Network 2014b). Luokasta muodostetaan olio, jolle syötetään sähköpostipalvelimen tiedot, sähköpostiviestin sisältö sekä vastaanottajat. Kaikki sovelluksessa käytetyt sähköpostiviestit ovat HTML-muotoisia. Viestit säilöttiin omiin HTML-tiedostoihinsa, joihin merkittiin paikkamerkkejä tietojen täyttöä varten. Viestin sisältö ladataan ensin tiedostoista sähköpostiviestin sisällöksi ja sen jälkeen viestistä täytetään paikkamerkit. Paikkamerkit ovat merkattu hakasulkeilla. Paikkamerkeistä on esimerkki kuvassa 2.

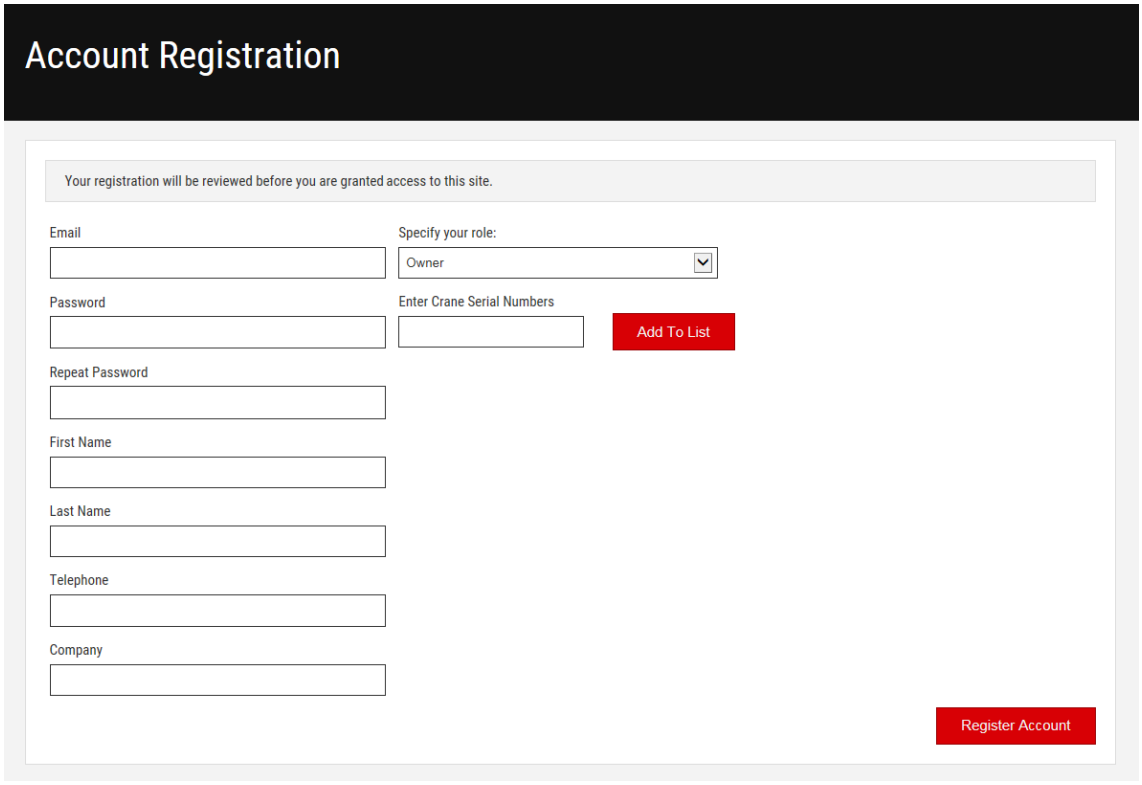
```
<h2>Email Validation</h2>
You have requested an account for the [sitename].
<br />
<br />
You will need to confirm your email before your account can be activated. You will
be notified about the activation via email.
<br />
<br />
[confirmlink]
```

Kuva 2. Paikkamerkit sähköpostiviestipohjassa.

Paikkamerkit mahdollistavat viestin sisällön muuttamisen dynaamisesti sovelluksen sisällä. Paikkamerkkien avulla saadaan sähköpostiviestin sisältöön muutettua muun muassa vastaanottajan tiedot. Esimerkiksi ”Hei [Loppukäyttäjä]” muuttuu muotoon ”Hei Essi”.

3.2.2 Rekisteröityminen

Rekisteröityminen tapahtuu sovelluksen web-pohjaisen käyttöliittymän kautta. Rekisteröitymistä varten rakennettiin oma web-sivu, johon loppukäyttäjä syöttää omat tietonsa ja lähettää rekisteröitymispyynnön. Rekisteröitymisessä tiedot ovat vapaasti syötettävissä, ainoa tarkistettava asia on syötetty sähköpostiosoite. Rekisteröityminen varmistetaan lähettämällä loppukäyttäjälle varmistusviesti. Rekisteröitymiseen käytettävä lomake näkyy kuvassa 3.



Kuva 3. Käyttäjätunnuksen rekisteröinti.

Sähköpostiosoitteen tarkistuksen implementointi on usein vaikeaa sähköpostiosoitteisiin liittyvien useiden sääntöjen takia. (RFC 2822; Akins 2015) Päädyimme hyödyntämään tarkistuksessa MailMessage-luokkaa (Hume 2014), jota käytetään myös sähköpostiviestien lähetykseen. MailMessage-luokasta muodostetaan olio, jolle syötetään argumenttina loppukäyttäjän antama sähköpostiosoite. Mikäli MailMessage-olio ilmoittaa virheestä, on sähköpostiosoite olion silmissä virheellinen eikä siihen voida lähettää sähköpostiviestiä. Sähkö-

postiosoitteen oikeellisuudella ei tällä tavalla periaatteessa ole mitään merkitystä, sillä mikäli käytetty olio ei pysty lähettämään sähköpostiviestiä annettuun osoitteeseen, on sähköpostiosoite hyödytön. Sähköpostin tarkistukseen käytettävä funktio näkyy kuvassa 4.

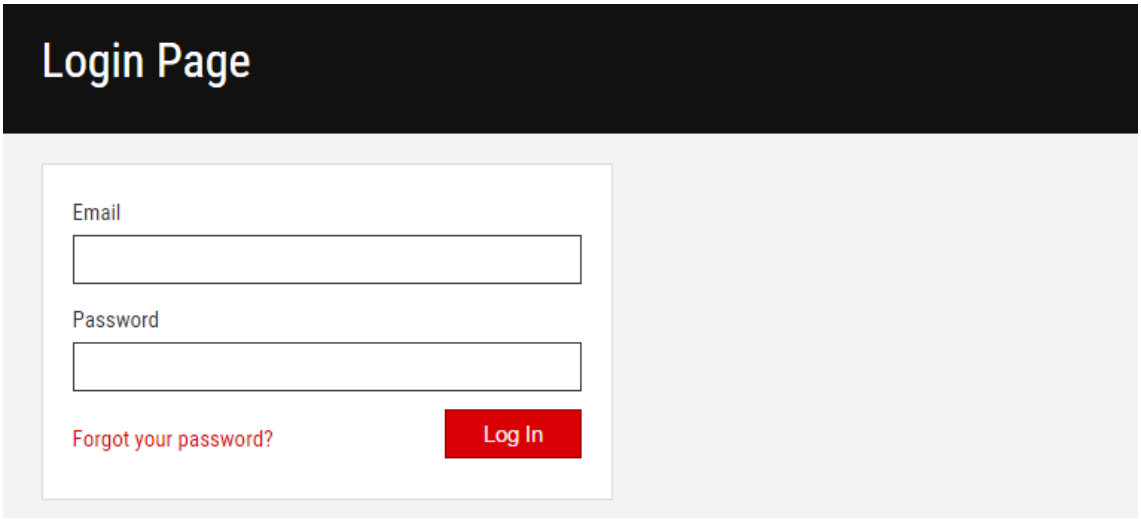
```
''' <summary>
''' Validates an email address.
''' </summary>
''' <param name="EmailAddress">Email address which will be validated.</param>
''' <returns>Returns true if email address was successfully validated.</returns>
Public Function IsValidEmail(ByVal EmailAddress As String) As Boolean

    Try
        'Checks for empty string
        If EmailAddress.Equals(String.Empty) = False Then

            'Creates a new instance of the MailAddress class using the specified Email address
            Dim Email As MailAddress = New MailAddress(EmailAddress)

            Return True
        Else
            Return False
        End If
    Catch ex As FormatException
        'Catches possible error
        Return False
    Catch ex As Exception
        Return False
    End Try

End Function
```

Kuva 4. Sähköpostin tarkistukseen käytettävä funktio.

Loppukäyttäjän sähköpostiosoitteen varmistuksen jälkeen varsinainen rekisteröitymispyyntö lähtee sähköpostitse laitteen valmistajalle, joka näkee viestissä rekisteröityjän tiedot ja päättää tapauskohtaisesti joko hyväksyä tai hylätä rekisteröinnin sähköpostissa olevien linkkien kautta.

3.2.3 Kirjautuminen

Rekisteröinnin hyväksynnän jälkeen loppukäyttäjä voi kirjautua sisään ja käyttää sovellusta. Kirjautuminen tapahtuu sovelluksen web-pohjaisen käyttöliittymän kautta. Kirjautumiseen tarvitaan rekisteröinnin yhteydessä annetut sähköpostiosoite ja salasana. Kirjautumiseen käytettävä lomake näkyy kuvassa 5.

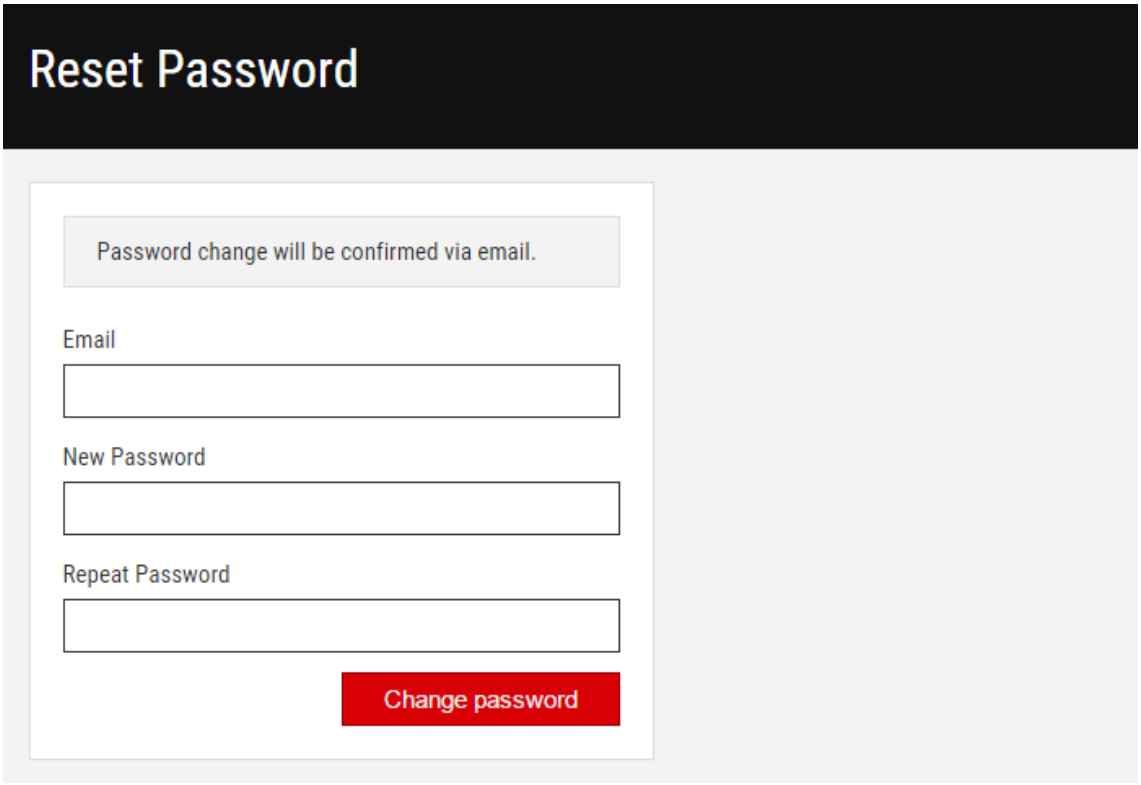


Kuva 5. Sovellukseen kirjautuminen.

Salasana on yksi loppukäyttäjän tiedoista, joka on suojattu, eikä sitä tiedä edes sivuston ylläpitäjät. Salasanan salauksessa on käytetty SHA-2 hajautusalgoritmia, joka on yksisuuntainen eli sen muodostamasta tiivisteestä ei voida laskea alkuperäistä salasanaa. Algoritmi muodostaa jokaisesta eri merkkijonosta ainutlaatuisen tiivisteen, mutta muodostettu tiiviste pysyy kuitenkin samana, mikäli myös syötetty merkkijono pysyy samana. Jokaiselle loppukäyttäjälle muodostetaan myös ainutlaatuinen kryptograafinen salt- eli suola-arvo, jota käytetään lisänä salasanan suojauksessa (Laakkonen 2013).

Salasanan tiivisteestä sekä suolasta laskettu yhteistiiviste on talletettu tietokantaan suola-arvon ohella. Jokaisen kirjautumisen yhteydessä syötetystä salasanasta sekä suola-arvosta lasketaan yhteistiiviste, jota verrataan tietokannasta löytyneeseen yhteistiivisteeseen. Mikäli laskettu sekä tietokannasta löytyneet yhteistiivisteet täsmäävät, on loppukäyttäjä syöttänyt salasanansa oikein ja hänet voidaan päästää käyttämään sovellusta.

Käyttäjä voi vaihtaa salasanansa web-käyttöliittymän avulla syöttämällä sähköpostiosoitteensa sekä uuden salasanan. Salasanan vaihdosta lähetetään loppukäyttäjälle varmistussähköposti, josta löytyy varmistuslinkki, jota painettuaan salasana vaihtuu sivustolla. Mikäli linkkiä ei paineta, ei salasana vaihdu. Salasana vaihdetaan lomakkeella, joka näkyy kuvassa 6.



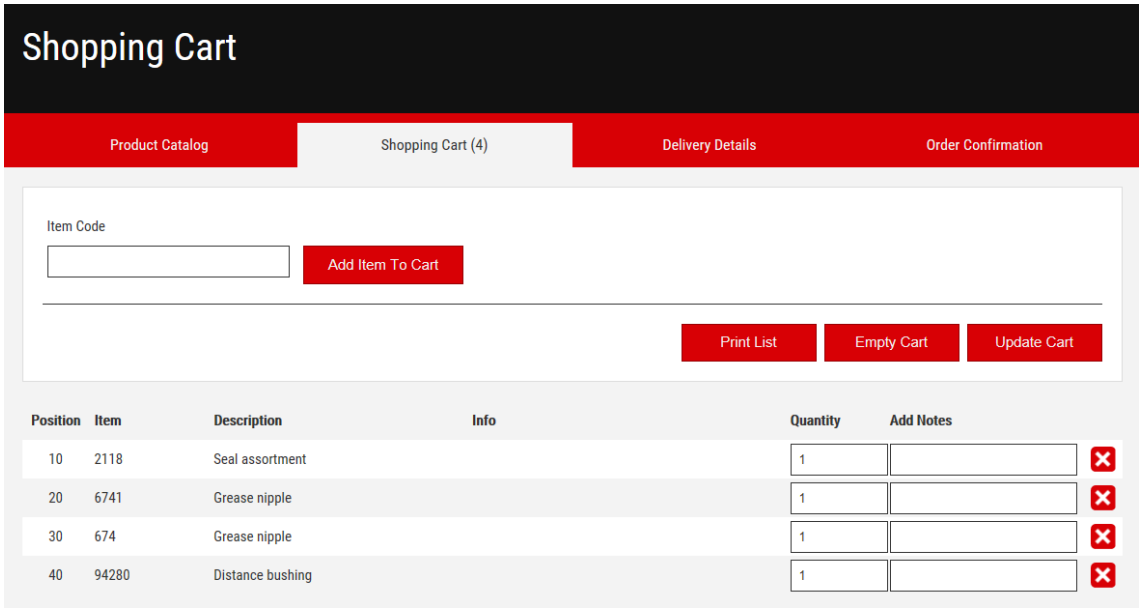
Kuva 6. Salasanan vaihto.

Yhden istunnon pituudeksi on määritelty 60 minuuttia. Mikäli loppukäyttäjä on käyttämättä sivustoa 60 minuuttia, kirjautuu hän automaattisesti ulos sovelluksesta. Loppukäyttäjä voi kuitenkin nollata tämän laskurin takaisin 60 minuuttiin käyttämällä jotakin sovelluksen toimintoa. Tämän toiminnon tarkoituksena on estää sovelluksen väärinkäyttö, mikäli loppukäyttäjä unohtaa kirjautua ulos sovelluksesta.

3.2.4 Ostoskori

Varaosa voidaan lisätä ostoskoriin laitteen rakennelistauksen kautta. Jokaiseen osalistan rivin oikeaan reunaan lisättiin painike, josta kyseisen osan pystyi lisäämään ostoskoriin. Kyseistä painiketta ei kuitenkaan lisätty moduuliriveille, koska moduuleita ei ollut tarkoitus myydä kokonaisina.

Kaikki ostoskoriin lisätyt osat tallennetaan suoraan tietokantaan. Tällöin loppukäyttäjä pystyy jatkamaan tarjouspyynnön tekemistä myöhemmin ilman, että osia tarvitsisi lisätä ostoskoriin uudelleen. Tämä mahdollistaa myös keskeneräisten tarjouspyyntöjen tutkimisen, mikä selventää loppukäyttäjien kiinnostuksen kohteita. Ostoskori näkyy kuvassa 7.



Kuva 7. Ostoskori.

Ostoskori toimii rivinumeroiden perusteella. Jokaiselle ostoskoriin lisätylle varaosalle luodaan ostoskoriin oma rivinsä, ja riville annetaan rivinumero. Tämä tarkoittaa, että samoja varaosia ei yhdistetä samalle riville vaan niitä saattaa olla usealla eri riveillä. Rivinumeroiden avulla sekä loppukäyttäjä että laitteen valmistaja tietävät, mistä osasta puhutaan, mikäli esimerkiksi toimituksessa ilmenee ongelmia.

Ostoskorin riveillä on mahdollisuus muuttaa kyseisen rivin osien määrää, lisätä kommentteja tai poistaa kyseinen rivi. Kommenttien avulla voidaan lisätä muun muassa väreihin tai muihin osan ominaisuuksiin liittyviä huomioita, ja näin erottaa esimerkiksi eri riveillä olevat tuotenimikkeet toisistaan.

3.2.5 Tarjouspyynnön tekeminen

Kun ostoskoriin on lisätty tarvittavat varaosat, voidaan ostoskorin sisällöstä lähettää laitteen valmistajalle tarjouspyyntö sille tarkoitetun painikkeen avulla. Tarjouspyyntö lähetetään sähköpostitse kummallekin osapuolelle, ja sähköpostiviestissä ilmenee kaikki tarjouspyyntöön liittyvät tiedot.

Laitteen valmistaja sekä loppukäyttäjä joutuvat vielä erikseen sopimaan toimituksesta sekä hinnoista tarjouspyynnön lähettämisen jälkeen, koska sovelluksella ei voi tehdä lopullista tilausta.

3.3 Käyttöönotettava versio

Toinen kehitysversio sovelluksesta sisälsi jo suurimman osan halutuista ominaisuuksista, mutta ennen käyttöönottoa haluttiin lisätä vielä muutama ominaisuus, jotka helpottavat käyttäjäkokemusta.

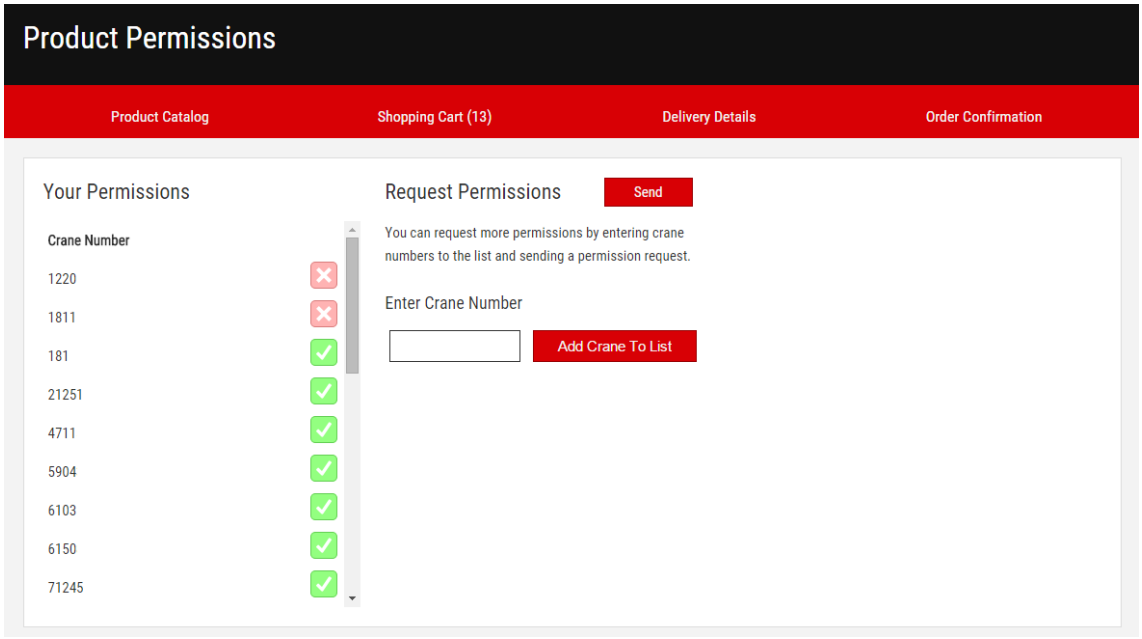
Käyttökokemusta helpottavia ominaisuuksia olivat muun muassa käyttäjän oma profiili, josta hän voi muuttaa omia ja yrityksensä tietoja. Tämän lisäksi tehtiin kohta toimitusosoitteelle, mikä helpottaa toimituskulujen laskemista. Toimitusosoitteita myös oli tarkoitus tallentaa muistiin seuraavia tarjouspyyntöjä varten. Sovelluksen toisesta kehitysversiosta puuttui myös yhteenveto ennen tarjouspyynnön lähettämistä. Yhteenveto auttaa loppukäyttäjää tarkistamaan kaikki tarjouspyyntöön liittyvät tiedot sekä lisäämään mahdollisia kommentteja ennen tarjouspyynnön lähettämistä.

3.3.1 Käyttäjäprofiili

Käyttäjäprofiili eriytettiin tarjouspyynnön tekoprosessista selkeyden vuoksi. Käyttäjäprofiili koostuu kahdesta osasta, jotka ovat loppukäyttäjän tietojenhallinta sekä loppukäyttäjän tiedonsaantioikeuksien hallinta.

Tietojenhallinta on yksinkertainen ja se käsittää loppukäyttäjän omien sekä yrityksen tietojen muuntamisen ja tallentamisen. Sähköpostiosoitetta loppukäyttäjä ei voi muuttaa, koska sitä käytetään loppukäyttäjän tunnistamisessa.

Tiedonsaantioikeuksien hallinnan kautta loppukäyttäjä näkee laitteet, joiden rakenteisiin hänellä on tiedonsaantioikeus. Loppukäyttäjä voi myös pyytää lisää laitteiden tiedonsaantioikeuksia lähettämällä hallinnan kautta sähköpostipyynnön. Loppukäyttäjä lisää haluamansa laitenumerot listaan, joka lisäyksen yhteydessä tarkistaa, löytyykö laite ja pääseekö loppukäyttäjä jo näkemään kyseisen laitteen. Tiedonsaantioikeuksien hallintaan käytettävä lomake näkyy kuvassa 8.

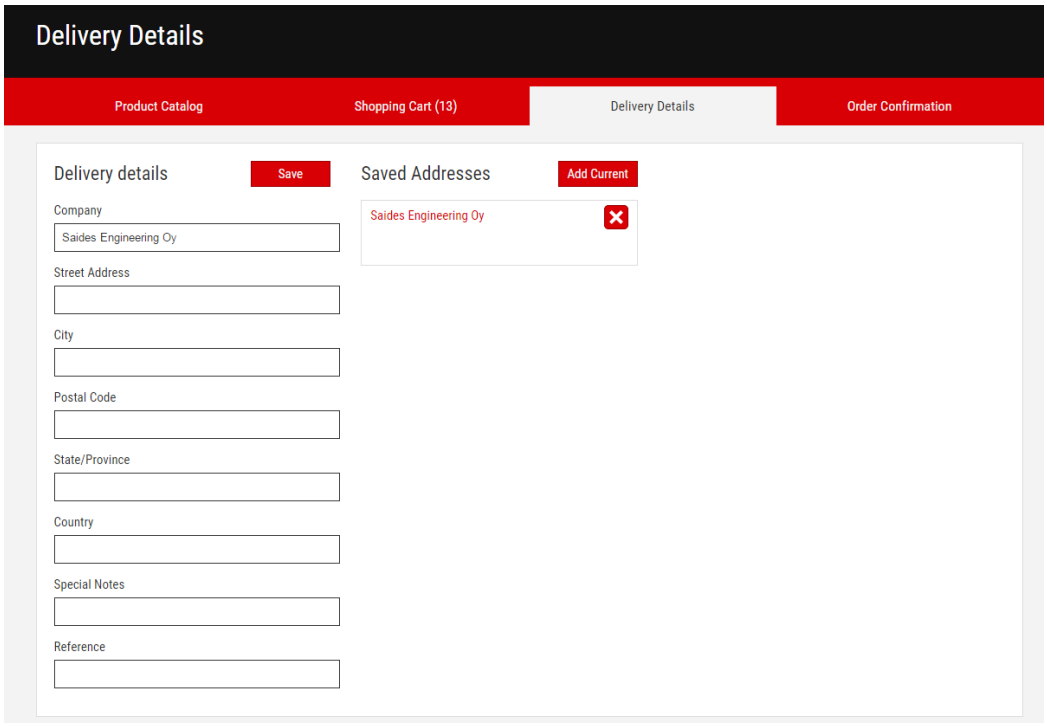


Kuva 8. Tiedonsaantioikeuksien hallinta.

Kun loppukäyttäjä lähettää pyynnön, lähetetään hyväksymis- sekä hylkäämislinkit sisältävä sähköpostiviesti laitteen valmistajalle, jonka hyväksynnän jälkeen loppukäyttäjä näkee uudet laitenumerot tiedonsaantioikeuslistassa.

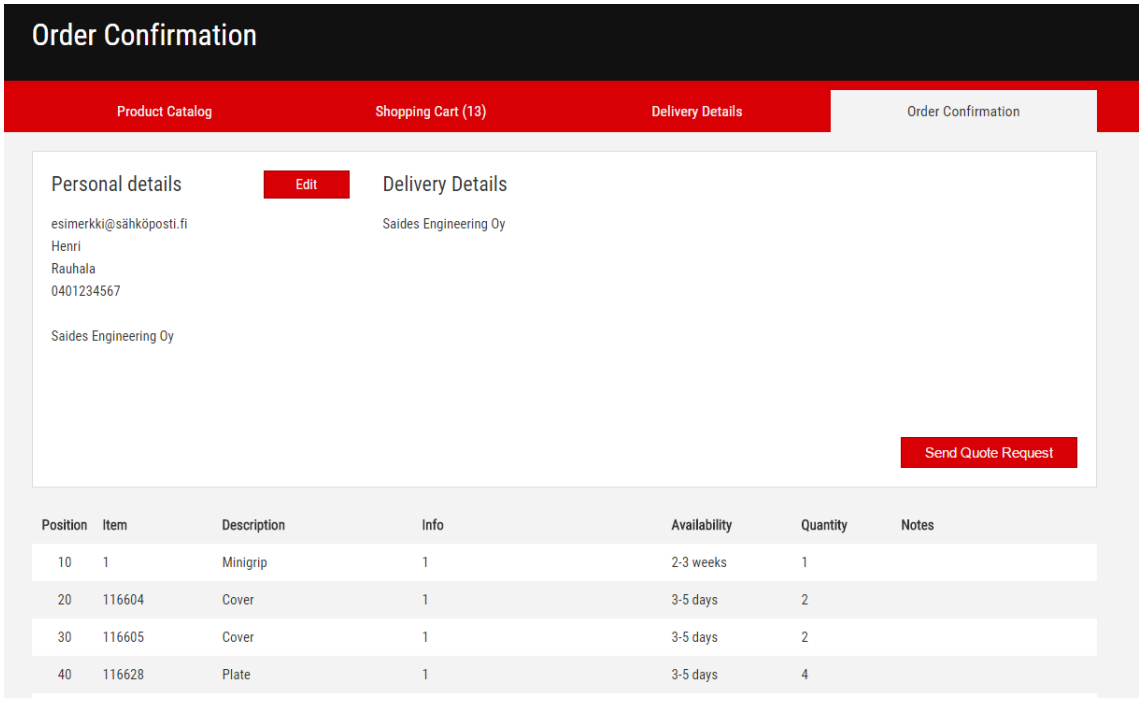
3.3.2 Tarjouspyynnön tiedot

Tarjouspyynnön tekoprosessiin lisättiin osa, joka mahdollistaa toimitusosoitteen syöttämisen toimituskulujen laskun helpottamiseksi. Toimitusosoitteen täytöstä ei tehty pakollista, koska loppukäyttäjä ei välttämättä tiedä tarkkaan, mihin tilatut varaosat toimitetaan. Toimitusosoitteita voidaan tallentaa useampi niille tarkoitettuun listaan, josta loppukäyttäjä voi valita haluamansa käytettäväksi tarjouspyynnössä. Tarjouspyyntöön on myös mahdollista tätä kautta lisätä kommentti tai viite. Lomake toimitusosoitteiden hallintaan näkyy kuvassa 9.



Kuva 9. Toimitusosoitteiden hallinta.

Tarjouspyynnön tekoprosessin lopussa näytetään vielä yhteenveto tarjouspyynnön tiedoista. Tietoja voidaan vielä muuttaa ennen tarjouspyynnön lähetystä. Yhteenvetolomake näkyy kuvassa 10.



Kuva 10. Yhteenveto tarjouspyynnöstä.

Kun loppukäyttäjä kokee tietojen olevan oikein, voi hän lähettää tarjouspyynnön laitteen valmistajalle. Tarjouspyynnön lähetyksen jälkeen ohjataan loppukäyttäjä näkymään, jossa hän näkee laitteen myyjän yhteystiedot sekä vielä tarjouspyynnön tiedot. Tämän jälkeen loppukäyttäjä voi palata selaamaan rakennelistausta.

3.4 Testaus ja käyttöönotto

Suurin osa testauksesta toteutettiin sovelluksen kehityksen yhteydessä iteratiivisesti. Aina kehitysversion valmistuessa yritettiin kuitenkin tuottaa mahdollisia virheitä imitoimalla loppukäyttäjää. Sovelluksen kehittäjä kykenee tähän paremmin, koska tietää, miten sovelluksen pitäisi toimia. Laitteen valmistaja on kuitenkin tässä tilanteessa parempi sanomaan, onko sivustolla näytetyt tiedot oikein. Löydetyt virheet olivat yleensä varsin pieniä ja vähäpätöisiä, johtuen sovelluksen rakenteesta sekä yksinkertaisuudesta.

Virheiden jäljityksessä auttoi suuresti käytetty virheloki-järjestelmä, johon virheiden tiedot tallentuivat. Virheloki hyödyntää StackTrace-luokkaa (Microsoft Developer Network 2014c), joka sisältää listan metodeista, jotka muodostavat reitin virheen alkuperään. Kyseinen metodi-lista tietoineen tallennetaan tietokantaan aina virheen tapahtuessa. Virhelokista näkee suoraan, missä sovelluksen osassa virhe ilmaantui ja mitä reittiä pitkin virhe syntyi. Näiden tietojen avulla virheet on usein helppo jäljittää ja korjata. Kaikkia virheitä ei kuitenkaan virhelokista saatavien tietojenkaan avulla voida selvittää, miksi onkin hyvä tallentaa lokiin myös tietoja loppukäyttäjästä, joka on saanut virheen aikaiseksi. Tällöin avautuu mahdollisuus kysyä tarkempia tietoja virhetilanteesta suoraan kyseiseltä loppukäyttäjältä.

Käyttöönotto oli hyvin suoraviivainen tämän sovelluksen kohdalla. Vaikka sovelluksesta olikin olemassa aiempi versio, oli siirrettävien tietojen määrä vähäinen, koska aiempi versio sovelluksesta ei säilönyt mitään tietoja tietokantaan. Käyttöönotossa tuli siis ainoastaan siirtää käyttäjien tiedot Active Directorysta uuden sovelluksen käyttämään tietokantaan ja kehottaa käyttäjiä tämän jälkeen vaihtamaan salasanansa. Kaikki muut tiedot saatiin suoraan siirrettyä testiympäristöstä käyttöympäristöön.

4 PROJEKTIN YHTEENVETO JA ARVIOINTIA

Projektin tarkoituksena oli kehittää aiempaa varaosasovellusta siten, että laitteen valmistaja pystyisi itse hallinnoimaan sivustolla tapahtuvia asioita. Samalla pyrittiin vähentämään ylläpidettäviä asioita, kuten asiakkaiden lisäämistä.

Sovelluksen kehitys tapahtui ensin vain Saides Engineering Oy:n sisällä. Uuden sovelluksen ominaisuuksista ei ollut alussa tarkkaa tietoa ja sovellusta kehitettiinkin alussa hyvin rauhallisella aikataululla. Sovellus esiteltiin asiakkaalle vasta sen valmistuttua muutaman kuukauden kehityksen jälkeen. Käyttöönoton tapahduttua myös asiakas osallistui vahvasti sovelluksen kehitystyöhön.

Jälkikäteen on helppo todeta, mitä projektin aikana olisi kannattanut tehdä toisin. Sovellukselta puuttui alussa suunta eikä tarkkoja suunnitelmia tehty, koska ei tiedetty, mitä tarkalleen halutaan. Sovellus paisuikin alkuperäisesti suunnitelmasta todella paljon, mikä osaltaan hidasti kehitystyötä, kun sovelluksen eri osiin jouduttiin lisäämään ominaisuuksia jälkikäteen. Projektin alussa olisikin ollut hyvä kartoittaa sovellukselta halutut ominaisuudet paremmin.

Sovelluksen alustana käytettiin aiemmasta sovellusversiosta tuttua WebForms-arkkitehtuuria. Kyseinen malli auttoi sovelluksen kehitystyön alullepanossa, mutta sovelluksen paisuttua se on muuttunut vaikeaksi ylläpitää. Mikäli sovelluksen koko olisi tiedetty jo alussa, olisi kenties ollut järkevämpää käyttää projektissa MVC-arkkitehtuuria, joka muun muassa antaa täyden kontrollin HTML:n osalta ja helpottaa testausta (Hambrick 2013).

Sovelluksen ulkonäköasioista vastasi Saidesin toimitusjohtaja, mutta itse olisin kaivannut, erityisesti projektin alussa, myös web-suunnittelijan mielipidettä ja mahdollisesti jotakin valmista sivustopohjaa. CSS:n opetteluun ja sen kanssa taisteluun kului lopulta todella paljon aikaa ja se söi resursseja sovelluksen varsinaisilta ominaisuuksilta.

Sovelluksesta saatiin lopulta asiakkaan mieltymysten mukainen ja suurin osa ominaisuuksista saatiin toteutettua. Jatkoa ajatellen kenties tärkein ominaisuus

olisi mahdollisesti saada sovellus skaalautumaan eri näyttökokoihin, jotta sovellusta voisi käyttää myös pienemmillä näytöillä luontevasti.

LÄHTEET

Akins, T. 2015. Email Validation Rules. Viitattu 6.4.2015 http://rumkin.com/software/email/rules.php.

Dumond, L. 2009. Nested ListViews and More - Working with Databound Controls Inside the ListView. Viitattu: 14.2.2014 http://leedumond.com/blog/nested-listviews-and-more-working-with-databound-controls-inside-the-listview.

Hambrick, P. 2013. .NET Web Forms vs. MVC: Which is Better? Viitattu 14.3.2015. http://www.seguetech.com/blog/2013/12/05/dotnet-web-forms-vs-mvc-which-better.

Hume, D. 2014. Validating email addresses in .NET – Handy Tip. Viitattu 14.3.2015. http://deanhume.com/home/blogpost/validating-email-addresses-in--net---handy-tip/5103.

Laakkonen, C. 2013. Salasanat talteen turvallisesti. Viitattu 14.3.2015. https://www.sofokus.com/blogi/salasanat-talteen-turvallisesti/.

Microsoft Developer Network. 2014a. ListView Class. Viitattu 6.3.2014. https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.listview(v=vs.100).aspx.

Microsoft Developer Network. 2014b. MailMessage Class. Viitattu 3.10.2014. https://msdn.microsoft.com/en-us/library/system.net.mail.mailmessage(v=vs.110).aspx

Microsoft Developer Network. 2014c. StackTrace Class. Viitattu 7.6.2014. https://msdn.microsoft.com/en-us/library/System.Diagnostics.StackTrace(v=vs.110).aspx

RFC 2822. Internet Message Format. Viitattu 14.3.2015 http://tools.ietf.org/html/rfc2822.