SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

PALVELUTARPEENARVIOINTI-VÄLINE

Opinnäytetyö

TEKIJÄ/T: Joonas Rissanen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Joonas Rissanen	
Työn nimi Palvelutarpeenarviointiväline	
Päiväys 19.2.2014	Sivumäärä/Liitteet 36
Ohjaaja(t) lehtori Keijo Kuosmanen, lehtori Sami Lahti, lehtori Jussi Koistinen	
Toimeksiantaja/Yhteistyökumppani(t) Kuopion kaupunki	

Tiivistelmä

Opinnäytetyön aiheena oli toteuttaa prototyyppi arviointivälineestä, joka mittaa Kuopion sosiaalitoimen asiakkaiden palveluntarvetta sekä tallentaa heidän arviointiinsa liittyviä tietoja tietokantaan. Tietoihin liittyivät henkilötiedot sekä asiakkaan arviointiin liittyvät taustatiedot.

Työn tuloksena ja tavoitteena oli keskeisten toimintojen määrittely ja toteutus verkkosovellukseksi PHP:lla käyttäen MySQL-tietokantaa tietojen tallennukseen. Sovelluksen käyttöliittymässä käytettiin moderneja ohjelmointikirjastoja sekä frameworkeja kuten Bootstrapia ja jQueryä. Projektin kehitys suoritettiin iteroiden ketterällä menetelmällä suuruutensa takia. Työ suoritettiin parityönä, josta syntyi kaksi opinnäytetyötä.

Tämä opinnäytetyö keskittyy ohjelmiston käytettävyyteen ja käyttökokemuksen sujuvuuteen.

Avainsanat
Framework, MVC, PHP, CodeIgniter, sosiaalitoimi, Kuopion kaupunki

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Joonas Rissanen			
Title of Thesis Service Need Evaluation Tool			
Date	19 February 2014	Pages/Appendices	36
Supervisor(s) Mr Keijo Kuosmanen, Lecturer, Mr Sami Lahti, Lecturer, Mr Jussi Koistinen, Lecturer			
Client Organisation /Partners City of Kuopio			

Abstract

The subject of this thesis was to implement a prototype of an evaluation tool which measures the need for social services for customers in the Kuopio Municipality. The prototype also stores this information for later use and refining along with some personal information and background information on the customer.

The objective and final outcome of the thesis was the definition and implementation of key functions and components as a PHP web application which uses MySQL to store information. Frameworks such as jQuery and Bootstrap were used to develop the software user interface. Due to the large size of the project, iteration using agile methods was employed. Two people worked on this project resulting in two theses.

This thesis concentrates on the usability and fluid user experience of the prototype.

Keywords
Framework, MVC, PHP, CodeIgniter, social services, city of Kuopio

SISÄLTÖ

TERMIT JA LYHENTEET

Iterointi	Menetelmä, jossa työvaiheita toistetaan kunnes haluttu lopputulos saavutetaan. Jokainen toisto on ns. iteraatio
PHP	Palvelinympäristöihin suunnattu ohjelmointikieli
MySQL	Suosittu relaatiotietokanta, jota usein käytetään PHP:n jatkeena. MySQL löytyy esimerkiksi Linuxille suunnatusta LAMP-paketista (Linux-Apache-MySQL-PHP)
Javascript	Yleisin skriptikieli verkkokäytössä, jota voidaan käyttää esimerkiksi verkkosivujen animoinnissa tai verkkosivujen muuntamisessa kokonaisiksi ohjelmiksi
Bootstrap	Verkkosivujen tekemistä helpottava työkalu
Jquery	Suosittu Javascript-kirjasto, joka lisää paljon funktionaalisuutta javascript-ohjelmointiin
JqueryUI	Jquerylle kehitetty jatkekirjasto, joka lisää yleisiä käyttöliittymäkomponentteja, kuten hälytysikkunat ja kalenteri
Netbeans	Kevyt ohjelmointiympäristö Java, PHP, HTML5 ym. ohjelmointikielille
Asynkroninen kommunikaatio	Ajasta riippumatonta viestintää palvelimen ja päätteen välillä. Asynkronisessa kommunikoinnissa pääte voi suorittaa nk. asynkronisen kutsun palvelimelle sovelluksen ajon aikana ilman, että se vaikuttaa muuhun ohjelmiston toimintaan (tällä voidaan välttää sivujen uudelleenlataus)

1 JOHDANTO

1.1 Taustat

Kuopion kaupungin sosiaalitoimi käyttää asiakkaiden arviointiin omaa arviointijärjestelmäänsä, jossa arvioija kysyy asiakkaalta kysymyksiä ja kirjoittaa vastaukset numeroarvoina kaavakkeeseen, joka laskee kyseiselle asiakkaalle palveluluokan arvojen perusteella. Lomakkeeseen myös merkitään päätökset palvelunannon suhteen. Tätä lomaketta käytetään lopullisen päätöksen tekoon asiakkaan palvelunannossa.

Kuopion kaupungin sosiaalitoimi muodostuu monista yksiköistä eri elämän tilanteissa oleville ihmisille. Näihin kuuluu mm. lastensuojelu, vammaispalvelu ja päihdeongelmapalvelu. Jokaisella yksiköllä on omat kysymysmallinsa, vastausten numeroarvojen vaihteluvälit ja niin edelleen.

Opinnäytetyön prototyypissä ei otettu kantaa eri yksiköiden eroavuuksiin järjestelmän kannalta.

1.2 Aikaisempi arviointiväline

Arviointiin käytettiin Kuopion kaupungilla aikaisemmin Excel-taulukkoa, johon pystyi syöttämään otsikoita, kysymyksiä ja vastauksia, jotka taulukko laski valmiiksi arvoiksi ja palveluluokiksi asiakkaille. Ongelmaksi taulukon käytössä muodostui skaalautuvuuden puute, tietojen arkistointi, muutosten seuranta ja käytännössä kaikki mahdolliset käytettävyyteen liittyvät seikat.

Aikaisemman arviointivälineen ongelmat synnyttivät tarpeen yhtenäiselle järjestelmälle, joka suorittaa samat toiminnot kuin Excel-taulukko, ja hoitaa myös asiakkaan tietojenkäsittelyn, historiatietojen käsittelyn, helpottaa yleistä käytettävyyttä ja palvelutarpeen arvioinnin sujuvuutta. Järjestelmä haluttiin intra- tai ekstranet-tyyppiseksi keskitetyksi työkaluksi verkon välityksellä.

1.3 Vaatimukset

Sovelluksen vaatimuksiin kuului helppo käytettävyys, yhtenäinen käyttöliittymä, tietojen moniulotteinen tallennus historiatietoineen, monipuoliset hallintatyökalut sekä monet muut moderneiksi mielletyt toiminnot helppokäyttöisyyteen liittyen. Projektin tavoitteina oli myös lisätä ymmärrystä järkevistä tavoista varastoida tietoa raportointia ja tutkimusta varten.

2 TEKNOLOGIA

Ohjelmiston toteutuksessa käytetään tyypillisiä verkkokehityssovelluksia ja -kieliä kuten PHP:tä, MySQL:ää, Javascriptiä, CSS:ää ja Apachea sekä Unix- että Windows-ympäristöissä. Ohjelmiston pohjaksi päätettiin valita MVC-arkkitehtuuria hyödyntävä dokumentoitu framework helpottamaan yleistä kehitystä ja yhtenäisen pohjan luontia. Frameworkin päälle vielä valittiin useita valmiita ohjelmistokirjastoja nopeuttamaan ja helpottamaan kehitystä.

Ohjelmointiympäristöksi valittiin Netbeans keveyden, helppokäyttöisyyden ja muokattavuuden vuoksi. Netbeans ei myöskään keveydessään ollut liian kevyt toiminnoiltaan, kuten ohjelmointimuistiot, mutta ei toisaalta niin raskas kuin esimerkiksi täydet ohjelmointiympäristöt, kuten Eclipse.

Kaikkea tätä varten myös otettiin käyttöön versiohallinta projektin hallinnointia varten. Versiohallinta toimi myös eräänlaisena lokikantana, jossa jokainen tallennus versiohallintaan kommentoitiin lyhyesti kehitysaikajanan havainnollistamisen helpottamiseksi.

2.1 MVC

MVC on suunnittelumalli, joka muodostuu kolmesta perusosasta. Yleensä MVC-malli toimii loogisena jatkeena vahvasti olio-ohjelmointiin nojaavassa kehityksessä. (Pitt, 2012) MVC on lyhenne sanoista Model-View-Controller tai malli-näkymä-käsittelijä. MVC-arkkitehtuurin tarkoituksena on jakaa ohjelmiston eri osa-alueet malleihin, näkymiin ja käsittelijöihin, mikä helpottaa muun muassa lähdekoodin lukua, ohjelmiston hahmottamista, modulaarista toteutusta ja yksikkötestausta.

Malliin kuuluu ohjelmiston ydinlogiikka, säännöt, data sekä funktiot. *Näkymä* voi olla mikä tahansa datan luettava muoto, mikä näytetään käyttäjälle. *Käsittelijä* toimii välikätenä näkymän ja mallin välillä muuttaen näkymästä saadun käyttäjän antaman tiedon dataksi mallille sekä muuttaen mallista saadun datan luettavaksi tiedoksi näkymälle ja käyttäjälle.

MVC-arkkitehtuuriin luonti painottaa koodin uudelleenkäytettävyyttä ja modulaarisuutta. Työkalut ja funktiot jaetaan pieniin osiin kategorisoiden ja luokitellen, jolloin pienistä logiikkaosista voidaan luoda suurempia kokonaisuuksia ilman koodin turhaa toistoa.

2.2 Framework

PHP:lle ja Javascriptille on vuosien varrella kehitetty useita frameworkeja, niin pieniä verkkosovelluksia kuin suuria yritysjärjestelmiä varten. Projektia varten frameworkien tärkeimmiksi ominaisuuksiksi muodostuivat dokumentaatio, helppokäyttöisyys, keveys ja tuttuus. Näihin vaatimuksiin perustuen, CodeIgniter valittiin projektille PHP-frameworkiksi, Bootstrap yleiseksi CSS-kirjastoksi tyylityksille sekä Jquery helpottamaan skriptausta.

Myös Symfony oli yksi mahdollisista valinnoista frameworkiksi sen laajuuden perusteella. Tulevaisuuden jatkokehityksen kannalta Symfony myös olisi ollut varmempi ja turvallisempi vaihtoehto, sillä CodeIgniter on jo käytännössä vanhentunut, ja tuki sille on lopetettu. Tärkeämpää oli kumminkin nopea käyttöönotto, jonka Symfonyn valinta olisi tehnyt käytännössä mahdottomaksi Symfonyn tuntemattomuudesta ja laajuudesta johtuen.

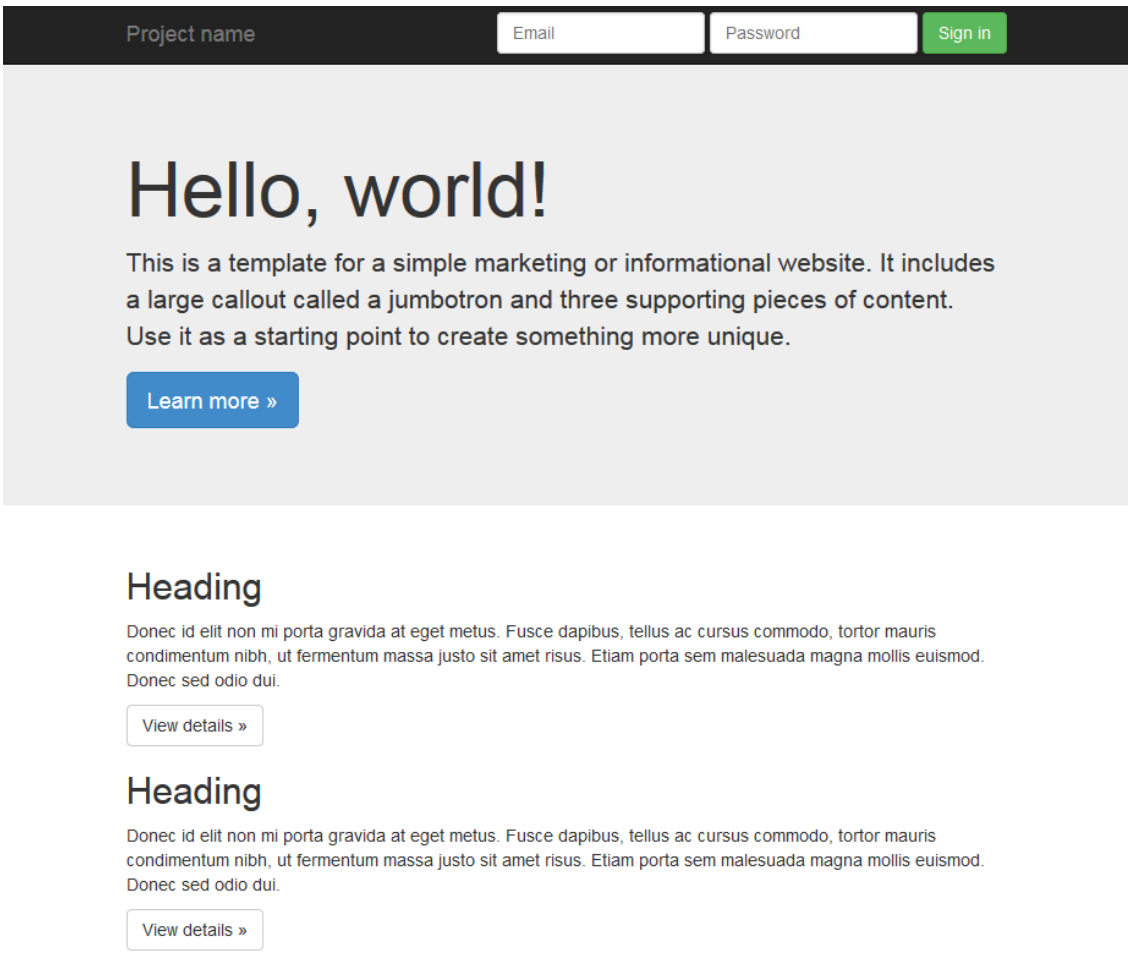
2.3 CodeIgniter

CodeIgniter on sivustonsa kuvauksen mukaisesti ”pienijälkinen ja tehokas framework PHP-ohjelmoijille, jotka tarvitsevat yksinkertaista ja eleganttia työkalua täysitoimisten verkkosovellusten toteutukseen.” (EllisLab, 2014) CodeIgniterista oli jo muodostunut aikaisempaa kokemusta, joten valinta oli helppo. CodeIgniterin, ja käytännössä kaikkien frameworkien, vahvuuksiin kuuluu valmiiden kirjastojen olemassaolo, ja näistä valittiin TankAuth hoitamaan käyttäjien todennus.

2.4 Bootstrap

Bootstrap on alun perin Twitter Oy:n kehittämä tyyli- sekä skriptikirjasto yrityksen Twitter-sivustolle. Alkuperäiseltä nimeltään se on Twitter Bootstrap ja se sisältää kymmeniä valmiita kirjastoja, komponentteja ja tyylimäärityksiä sivuasetelmien, lomakkeiden, navigaatiopalkkien ja monien muiden elementtien helppoon toteutukseen verkkosivuilla.

Bootstrap mahdollisti projektissa nopean iteroinnin helpottamalla komponenttien lisäystä, poistoa ja muokkausta. Bootstrapin verkkosivuilla monet esimerkit ja sivupohjat muuttivat käyttöönoton keston päivistä tunneiksi.



KUVA 1. Bootstrap esimerkkisivu

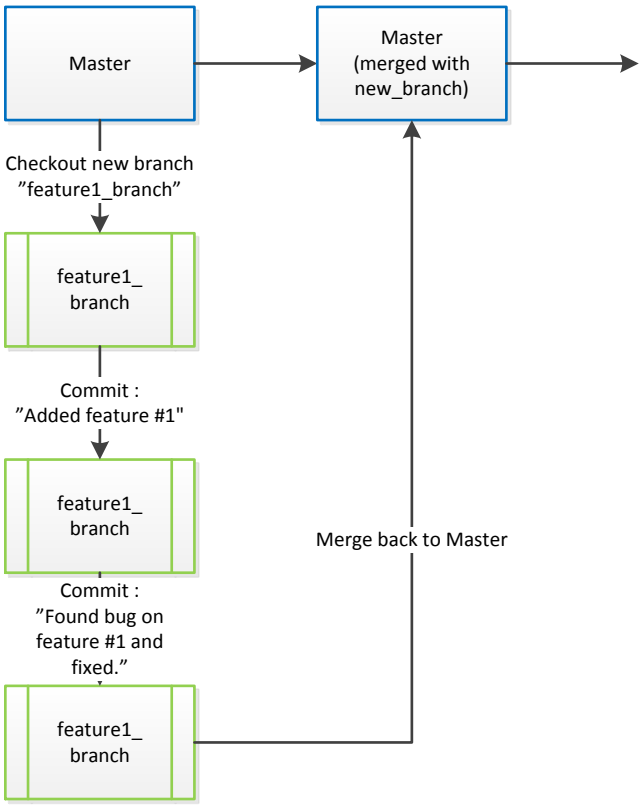
2.5 Versiohallinta

Ohjelmistokehityksessä tulee vastaan aina tiedon tallennus ja hallinta; etenkin kysymys, miten monta kehittäjää voi muokata ja kehittää samaa ohjelmistoa yhtä aikaa. Versiohallinta muodostuu tässä vaiheessa projektille elintärkeäksi.

Nopeaa iterointia varten Git tuntui erityisen toimivalta ratkaisulta. Gitin ominaisuuksiin kuului helppokäyttöisyys pienen opettelun jälkeen ja nopea käytettävyys, joka ei vienyt aikaa ja keskittymistä itse kehitystyöltä. Muutosten kirjaaminen tuli tapahtua lyhyesti ja ytimekkäästi. Tätä tarkoitusta varten Git vaikutti parhaimmalta ratkaisulta.

Gitin pääasiallinen käyttöliittymä on tekstipohjainen konsolisovellus. Myös graafisia käyttöliittymiä on kehitetty, kuten TortoiseGit, joka mukailee suosittua TortoiseSVN-sovelluslisää SVN-versiohallinnalle.

Gitin työnkulku on lopulta yksinkertainen (ks. KUVA 2). Käyttäjä aloittaa oman versiohaaran (checkout), jota työstetään, ja johon lisätään toimintoja (commit). Jokaiseen lisäykseen kirjataan oma lyhyt kommentti, missä kerrotaan, mitä tehtiin. Pienten lisäysten tekeminen lyhyillä kommenteilla katsotaan hyväksi työtavaksi. Näin voidaan katsoa myöhemmin sovelluksen kehityskulkua.



KUVA 2. Git - Yksinkertainen työnkulku

3 TYÖPROSESSIT

Työn suunnittelu ja toteutus olivat tyyliltään ketterää ohjelmistokehitysmallia. Työtä iteroitiin ja määrityksiä muuteltiin tarpeen mukaan. Käytännössä mitään ei suunniteltu tai määritelty tarkasti dokumentoiden vaan suurin osa kehityksestä suoritettiin vapaamuotoisesti iteroiden. Suunnittelu ja määrittely hoidettiin palavereissa asiakkaan kanssa ideoita keräten ja työstäen. Palavereiden jälkeen palavereista kirjoitettiin muistiot, joiden pohjalta projektia työstettiin nopeasti esiteltäväksi prototyypiksi.

3.1 Ketterä ohjelmistokehitysmalli

Ketterä kehitysmalli syntyi vastakaikuna ns. vesiputousmallin tarkoille ja joustamattomille ohjelmistomäärittelyille ja sopimuksille. Kehitysmallin juuret ovat jo 50-luvun lopulta. Ketterä ohjelmistokehitys kulminoitui vuonna 2001 niin kutsuttuun ketterän ohjelmistokehityksen julistukseen. Ketterän ohjelmistokehityksen julistus on lyhyesti:

”Löydämme parempia tapoja tehdä ohjelmistokehitystä, kun teemme sitä itse ja autamme muita siinä. Kokemuksemme perusteella arvostamme:

Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja
Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota
Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja
Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa

Jälkimmäisilläkin asioilla on arvoa, mutta arvostamme ensiksi mainittuja enemmän.”
(Beck;ym., 2001)

3.2 Määrittely ja suunnittelu

Ennen työn aloitusta, alustavan määrittelyn suoritti Petri Kejonen ja Riitta Holmström. Aloituspalavereissa projektia jatkomääriteltiin ja kuvailtiin tarkemmin käsitekartoilla ja prosessikaavioilla. Määrittely saattoi muuttua projektin kuluessa riippuen siitä, mikä koettiin tarpeelliseksi ja mikä tarpeettomaksi eri aikavälein. Palavereissa avustivat opinnäytetyön ohjaajat.

3.3 Toteutus

Toteutuksessa otettiin heti alkuun käyttöön Git-versiohallinta, jolla tallennettiin kaikki muutokset, lisäykset ja poistot koodista GitHub-palveluun. Git integroituu automaattisesti Netbeansiin, mikä teki projektin toteutuksesta sujuvamman.

3.4 Testaus

Jokaisen uuden toiminnon testaus yleensä suoritettiin toteutuksen aikana pikaisesti sekä pinnallisesti. Ajankäyttö painottui huomattavasti enemmän kehitykseen kuin toimintavarmuuteen ja rajatapauksien etsimiseen. Yksikkötestaus rajapinnan toteutusta kokeiltiin kehityksen aikana, mutta se todettiin liian aikaa vieväksi ja mahdollisesti turhaksi, koska mitkään funktiot ja toiminnot eivät olleet täysin varmasti lopullisia.

Projekti todettiin valmiiksi asiakastestaukseen 16.4., jolloin työ luovutettiin asiakkaalle testattavaksi. Testauksessa asiakas voi kokeilla ohjelman toimintoja, löytää mahdollisia virheitä ja ilmoittaa näistä myöhempää virheidenkorjausta varten.

3.5 Dokumentointi

Työn dokumentointi tapahtui suurimmaksi osaksi määrittelyn ja suunnittelun aikana ja lopulta ennen luovutusta käyttöohjeissa ja toimintolistauksissa. Määrittelyn ja suunnittelun aikana dokumentointi suoritettiin muistiinpanoihin. Projektia työstäessä muuttuvat määrittelyt kirjattiin palaverimuistioihin. Tämä raportti on osa dokumentaatiota.

Projektin versiohallinta on myös olennainen osa ohjelmiston dokumentaatiota. Jokainen sovellukseen tehty muutos kommentoitiin versiohallintaan tallennettaessa ja versiohallinta huomioi jokaisen tallennuksen ajan ja päivämäärän. Täten versiohallinnan historiaa katsomalla saa myös kuvan sovelluksen kehityksestä.

4 ARVIOINTIVÄLINE

4.1 Rakenne

Arviointivälinesovelluksen rakenne perustuu MVC-mallin malli/käsittelijä/näkymä -jakaumaan ja jakautuu aliluokkiin. Alla oleva taulukko (TAULUKKO 1. Ohjelmiston rakenne) kuvaa ohjelmiston rakennetta.

TAULUKKO 1. Ohjelmiston rakenne

Tyyppi	Aliluokka	Selitys
Mallit	*Battery*	Kysymyspatteriston malliluokka
	Card	Asiakaskortin malliluokka
	Customer	Asiakkaan malliluokka
	Message	Viestien ja viestinnän malliluokka
	Tank_auth	Kolmannen osapuolen autentikointiluokka, joka vastaa käyttäjän salasanan salauksesta ja perustietojen tallennuksesta
	User	Käyttäjän malliluokka
Käsittelijät	*Admin*	Pääkäyttäjäkäsittelijä. Pääkäsittelijä pääkäyttäjänäkymässä.
	Ajax	Ajax-käsittelijäluokka. Tätä luokkaa käytettiin melkein jokaisessa dynaamisessa sovelluksen osassa, kuten asiakaskorttien listauksessa tai taustatietojen haussa.
	Asiakas	Asiakaskäsittelijä. Pääkäsittelijä asiakasnäkymässä.
	Auth	Autentikointikäsittelijä. Hoitaa sisäänkirjauksen sekä käyttäjien käsittelyn.
	Patteri	Kysymyspatteriston käsittelijäluokka.
	Viesti	Viestintämoduulin käsittelijäluokka.

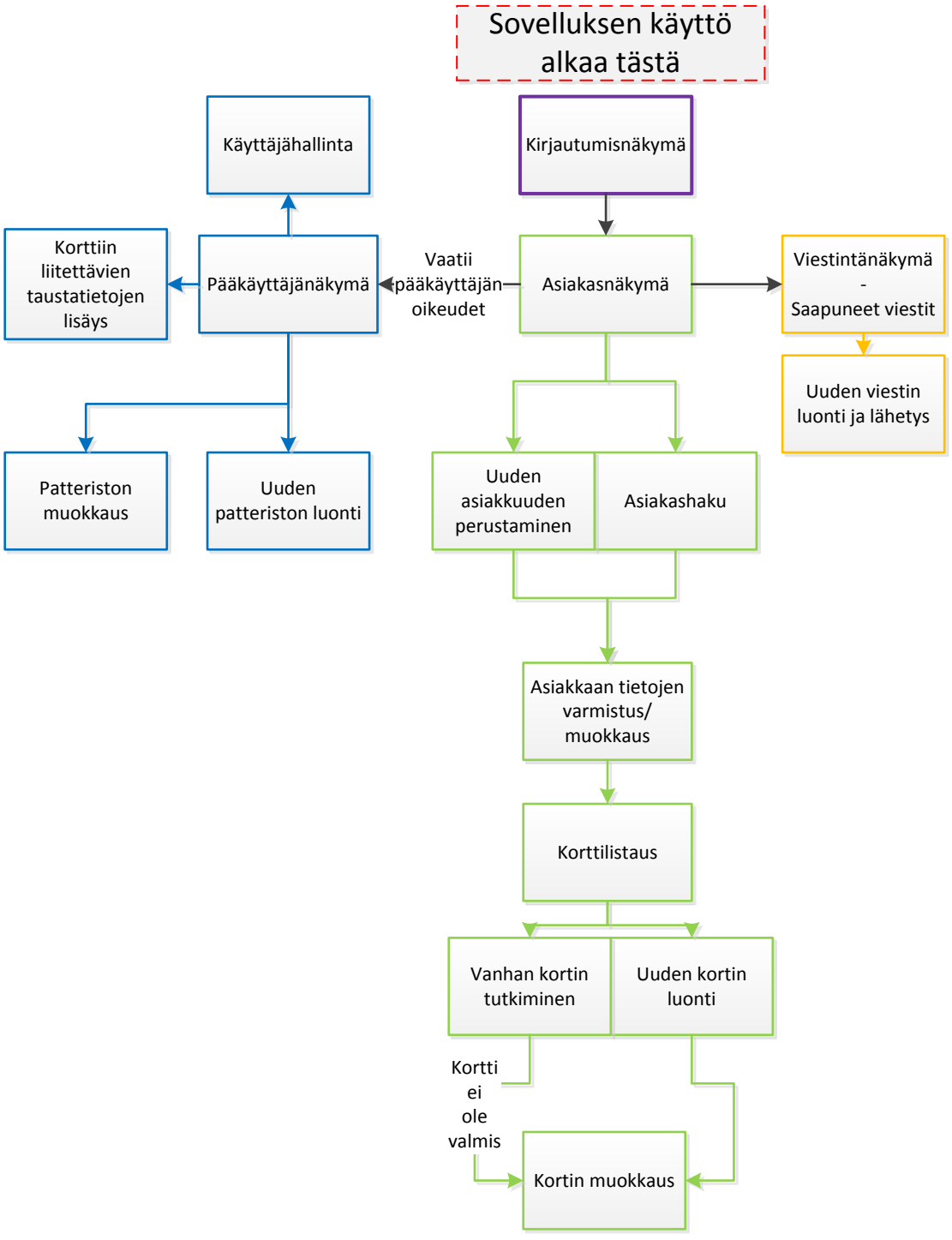
Tyyppi	Aliluokka	Selitys
Näkymät	*Admin* (**Virhe. Viitteen lähdettä ei löytynyt.**)	Pääkäyttäjänäkymä. Koostui kolmesta alinäkymästä: kysymyspatteristot, käyttäjät ja taustatiedot
	Ajax	Ajax-käsittelijä palauttaa aina Ajax-näkymän, kun halutaan jotain tietoa asynkronisesti sivun latauksen jälkeen ilman koko sivun uudelleenlatausta.
	Asiakas (**Virhe. Viitteen lähdettä ei löytynyt.**)	Asiakasnäkymässä tapahtuu kaikki asiakkaisiin liittyvät toiminnot, kuten haku, uuden asiakkuuden perustaminen, asiakkaan tietojen muokkaus ym.
	Auth	Sisäänkirjausnäkymä.
	Battery (**Virhe. Viitteen lähdettä ei löytynyt.**)	Kysymyspatteristonäkymä, jossa voi muokata tai tutkia kysymyspatteristoon syötettyjä arvoja riippuen siitä, onko asiakaskortti valmis vai keskeneräinen.
	Message (**Virhe. Viitteen lähdettä ei löytynyt.**)	Viestintänäkymä.
	Modals	Ilmoitusikkunanäkymä, jota käytettiin aina, kun haluttiin käyttää yksilöllistä sivunsisäistä ikkunaa (nk. modaalia). Käytetty esimerkiksi käyttäjähallinnassa uuden käyttäjän luonnissa (ks. **Virhe. Viitteen lähdettä ei löytynyt.**)
	Scripts	Jokaiseen näkymään lisätty yksilöllinen skriptinäkymä, jossa on kaikki sivun toimintoihin liittyvät skriptit, joita ei käytetä muissa näkymissä.
	Templates	Sisältää sovelluksen ylä- ja alatunnisteen, jotka ladataan aina muiden näkymien lisäksi.

Näkymissä painotettiin käytettävyyttä ja informaation selkeää ja jäsenneltyä näyttämistä. Myös sivujen uudelleenlatausta yritettiin minimoida käyttämällä skriptausta ja palvelinkutsuja Ajaxilla. Ajax-kutsuja varten implementoitiin oma käsittelijä- ja näkymäpari, mikä helpotti dynaamisen tiedon tuontia käyttäjälle sovelluksen ajon aikana.

Ohjelma jakautui lopulta kolmeen päänäkymään: asiakas, viestintä ja pääkäyttäjä.

4.2 Sovelluksen prosessikuvaus

Sovelluksen käyttö aloitetaan aina kirjautumalla sisään tunnuksilla, jotka pääkäyttäjä on määrittänyt sovelluksen peruskäyttäjille. Kirjautumisen jälkeen asiakasnäkymä avautuu ensimmäisenä sisäänkirjautuneelle käyttäjälle. Alla oleva kuva (KUVA 3) havainnollistaa mahdollista käyttöskenaariota.



KUVA 3. Prosessikaavio

4.3 Sovelluksen sisäiset käsitteet

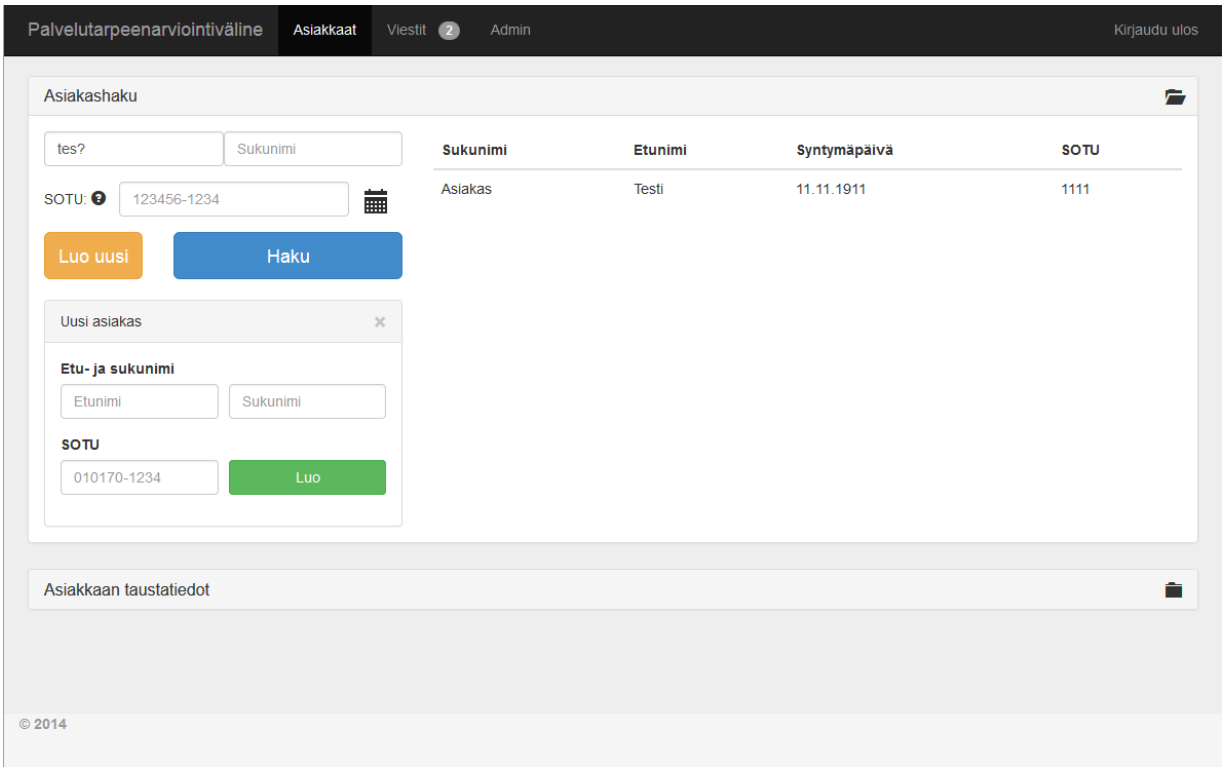
Arviointivälineen suunnittelu- ja määrittelyvaiheessa vakiintui käsitteitä, joita kaikkia ei aikaisemmin ollut määritelty.

ASIAKAS	Henkilö, jonka palveluntarvetta arvioidaan sovelluksen avulla. Asiakkaasta tallennetaan sovelluksen tietokantaan perustiedot: nimi, osoite, puhelinnumero, sähköpostiosoite, kaupunki ja postinumero.
ASIAKASKORTTI	Asiakaskortti on oma käsitteensä sovelluksessa eikä välttämättä vastaa sosiaalitoimen yksiköiden merkitystä asiakaskortista. Asiakaskortti sovelluksessa on asiakkaalle perustettava lomake, johon talletetaan kaikki asiakkaan palvelutarpeen arviointiin liittyvät asiat. Asiakaskortti perustuu kysymyspatteristoon, johon vastaamalla selvitetään asiakkaan palveluntarve. Arvioija haastattelee asiakasta ja merkitsee kortissa olevan kysymyspatteriston kysymyksiin oman arvionsa ja mahdollisesti asiakkaan oman arvion kysymyksiin annettujen vastausten perusteella.
KYSYMYSPATTERISTO	Lista kysymyksiä, joihin vastataan asiakasta haastattelemalla. Kysymyksiin voi kuulua asiakkaan sen hetkinen asuntotilanne, taloudellinen tilanne, jaksaminen ym. elämäntilannekysymykset. Kysymyspatteristo on sarakkeista ja riveistä muodostettu taulukko. Sarakkeisiin kuuluu palvelutyypit sekä arvioijan ja asiakkaan arviot palveluntarpeesta. Rivit ovat kysymyspatteriston kysymyksiä, joihin vastaamalla määritellään lopullinen palveluntarve. Kysymyspatteriston vastaukset (arviot) ovat kaikki numeroarvoja, joita käytetään palveluntarpeen laskemisessa.
TAUSTATIETO	Kysymyspatteristoon liitettävät yksilölliset tiedot, jotka eivät vaikuta palvelutarpeen laskemiseen sovelluksessa, mutta ovat silti merkittävää tietoa.
ARVIO	Arvioijan merkitsemä numeroarvo asiakaskorttiin, jota käytetään lopullisessa palveluntarpeen laskennassa. Jokainen arvo kerrotaan korjauskertoimella.
KORJAUSKERROIN	Arvioiden numeroarvot kerrotaan korjauskertoimella, jolla muodostetaan lopullinen numeroarvo, joka määrittelee palveluntarpeen.

4.4 Asiakasnäkymä

Asiakassivu (KUVA 4) on yksi suuri kokonaisuus, jossa tapahtuu kaikki asiakkaisiin liittyvät toiminnot asiakkaan hausta asiakkaan korttien tietojen katsomiseen. Asiakasnäkymässä voi hakea rekisterissä olevia asiakkaita tai luoda uuden asiakkaan rekisteriin.

Haun tai luonnin jälkeen asiakkaan tiedot päivitetään tai tarkistetaan oikeellisiksi, jolloin ohjelma siirtää käyttäjän kyseisen asiakkaan asiakaskorttilistaan. Asiakaskorttilistassa käyttäjä voi perustaa asiakkaalle uuden kortin, muokata keskeneräisiä kortteja tai katsoa valmiita kortteja, joita ei voi enää muokata.

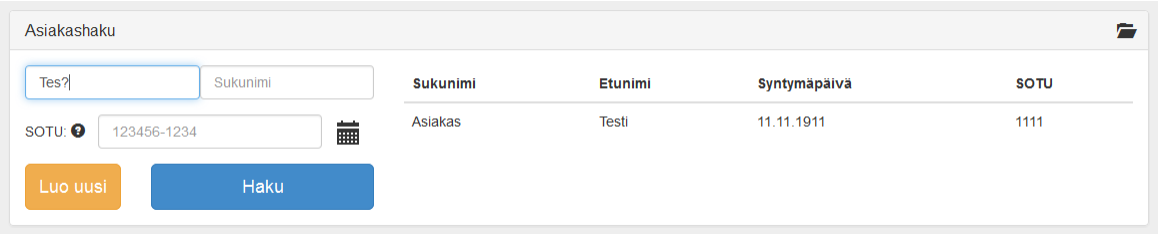


KUVA 4. Asiakasnäkymä

4.4.1 Asiakashaku

Palvelunarviointiprosessi aloitetaan asiakkaan hakemisella ja valitsemisella (KUVA 5). Jos asiakasta ei ole tietokannassa, uusi asiakas voidaan luoda antamalla nimi ja sosiaaliturvatunnus. Sosiaaliturvatunnus voidaan syöttää tietokantaan vain kerran.

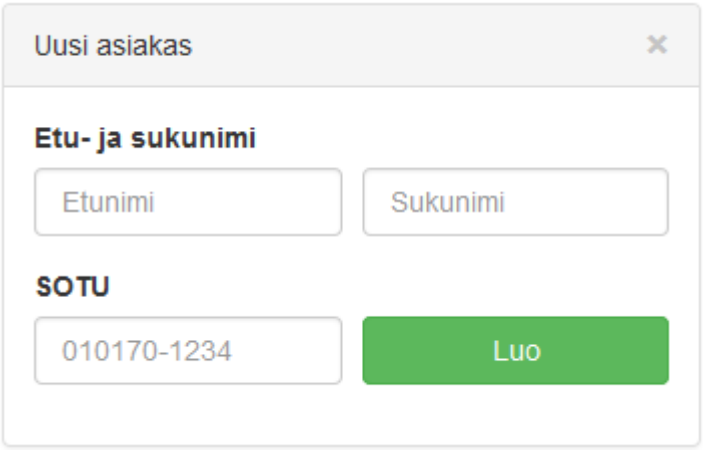
Asiakkaan hakuun voi käyttää neljää eri hakuehtoa: etunimeä, sukunimeä, syntymäpäivää tai sosiaaliturvatunnusta. Sosiaaliturvatunnus on absoluuttinen ja poissulkeva hakuehto. Muuten haun voi suorittaa millä tahansa etunimen, sukunimen tai syntymäpäivän yhdistelmällä. Nimen perusteella hakiessa kysymysmerkki (?) toimii korvausmerkkinä, joko nimen alulle tai lopulle.
Päivämäärän valintaa varten otettiin myös käyttöön JQueryUI:n päivämääränvalitsin (Datepicker). Sosiaaliturvatunnus ja syntymäpäiväkenttä toteutettiin yhdistelmäkenttänä, johon pystyy syöttämään sosiaaliturvatunnuksen tai syntymäpäivän.



KUVA 5. Asiakashaku

4.4.2 Asiakkaan luonti

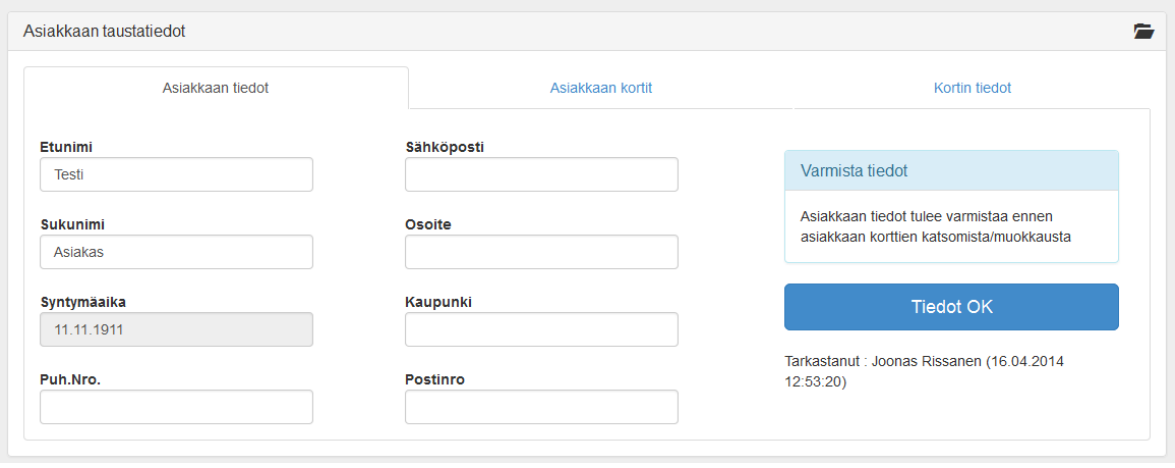
Uuden asiakkaan luonnissa (KUVA 6) täytyy syöttää asiakkaan nimi ja sosiaaliturvatunnus järjestelmään. Onnistuneen luonnin jälkeen siirrytään asiakkaan tietojen varmistukseen.



KUVA 6. Uusi asiakas -lomake

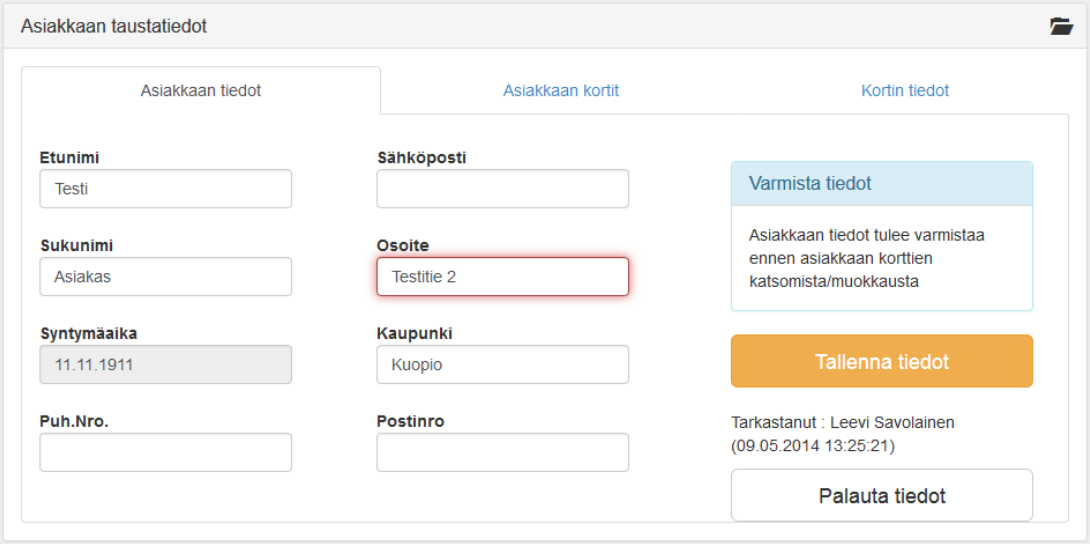
4.4.3 Asiakkaan tietojen varmistus

Asiakkaan viimeisimmät tiedot tallennetaan aina järjestelmään ennen asiakkaan korttien muokkauksen tai tutkimisen aloittamista. Jos tiedot eivät ole muuttuneet, tiedot vain varmistetaan oikeellisiksi (KUVA 7).



KUVA 7. Taustatietojen varmistuslomake

Jos asiakkaan tietoja muutetaan lomakkeessa, lomake korostaa muuttuneita tietoja punaisilla reunoilla ja muuttamalla varmistus-napin tallennus-napiksi (KUVA 8). Tiedot voidaan myös palauttaa ennen tallennusta, jos tietojen muuttamisessa tapahtui virhe.



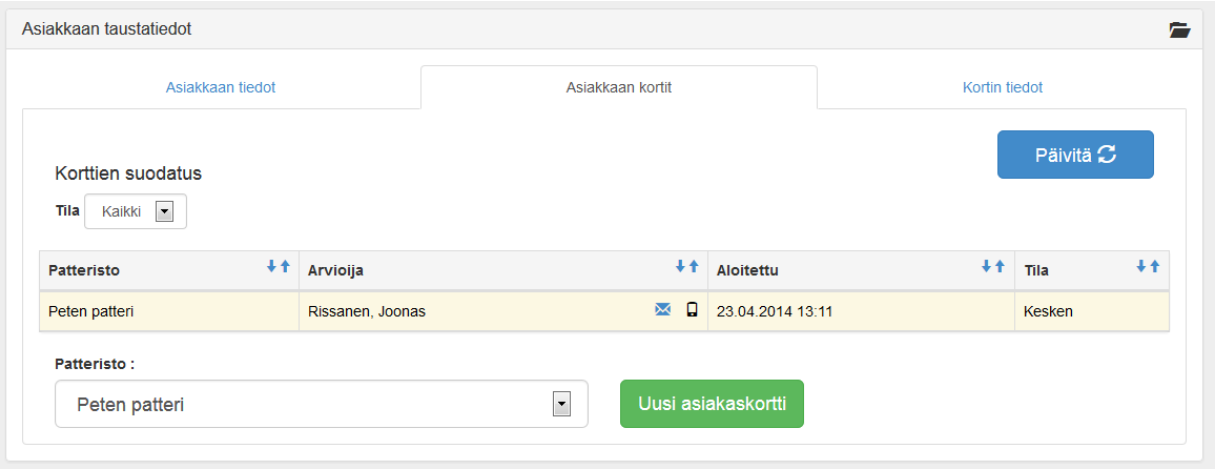
KUVA 8 Taustatietoja muutettu

4.4.4 Asiakaskorttien listaus

Asiakkaan kortit listataan tietojen tarkistuksen jälkeen (KUVA 9). Asiakaskortit ovat järjestelmän keskeisin ominaisuus. Tästä johtuen korttilistaukseen otettiin monta suodatus- ja lajittelutapaa oikean kortin löytämisen helpottamiseksi.

Korttilistauksen voi järjestää aakkosjärjestykseen patteriston tai kortin perustaneen arvioijan sukunimen mukaan, kortin perustusajan mukaan tai tilan mukaan. Käyttäjä voi myös suodattaa keskeneräiset tai valmiit kortit pois listasta.

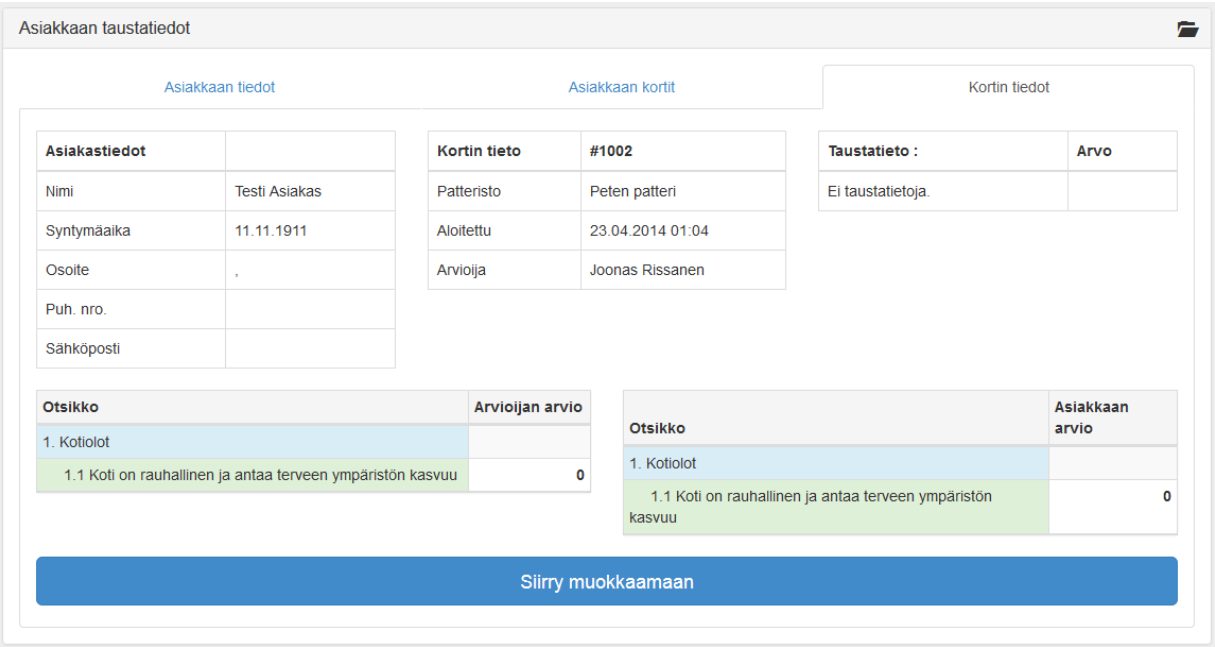
Listaus on prototyyppitasolla ja siitä puuttuu ominaisuudet esimerkiksi arvioijan tai patteriston nimen perusteella hakuun, mikä toimisi paremmin pitkään käytetyssä järjestelmässä, jossa yhdellä asiakkaalla voi olla monta kymmentä korttia. Järjestelmä ei automaattisesti lähetä uusinta listaa tietyin aikavälein prototyyppitasolla, minkä takia päivitä-nappi implementoitiin viimeisimmän korttilistauksen hakemiseen asynkronisesti.



KUVA 9. Asiakkaan kortit

4.4.5 Asiakaskortin tiedot

Asiakaskortin valinnan jälkeen, käyttäjälle näytetään asiakaskortin yhteenveto sekä asiakaskortissa olevat yksilölliset taustatiedot (KUVA 10). Yhteenvetoruudusta näkee kortin ennen sen muokkaamista.



KUVA 10. Kortin tiedot

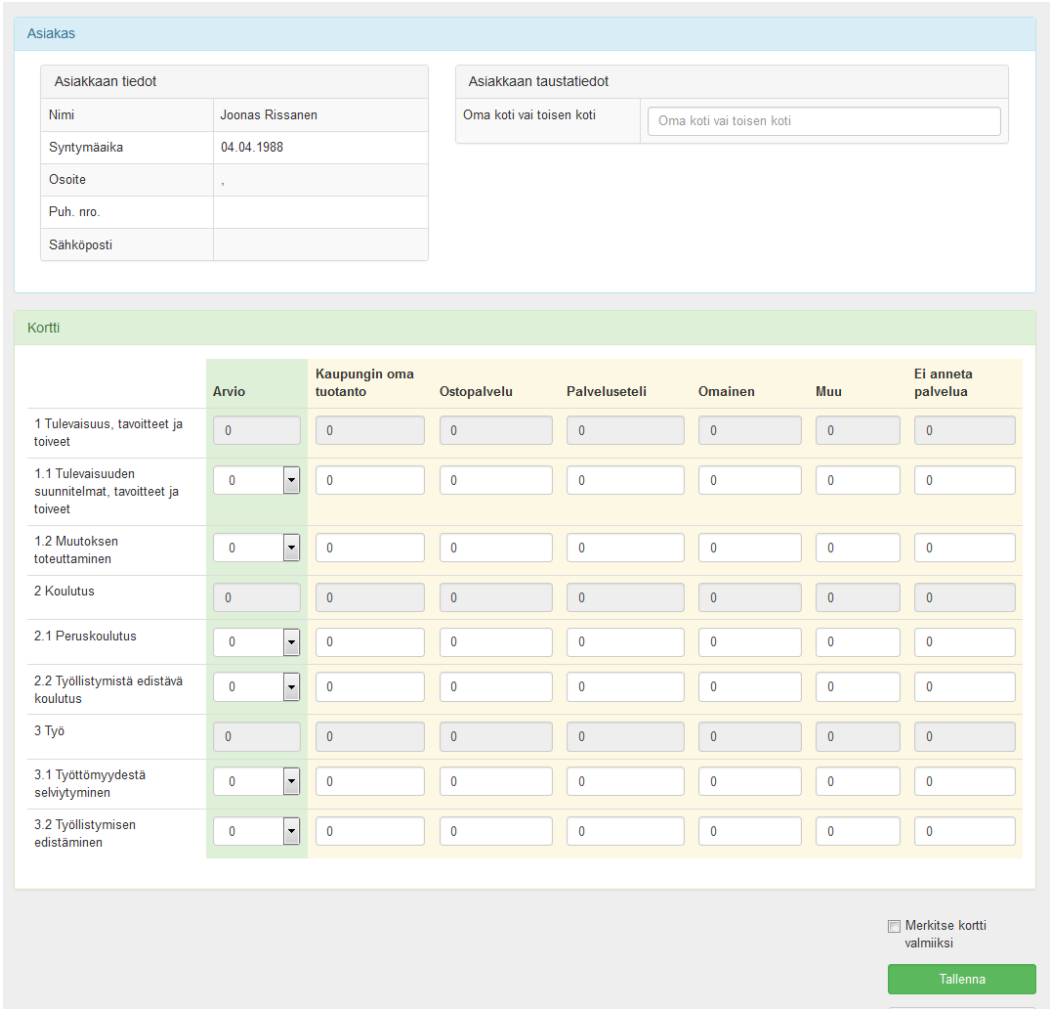
Asiakaskortin yhteenvedossa listataan kortin kysymykset, kysymysten alirivit ja näille arvioijan ja asiakkaan antamat arviot (KUVA 11). Yhteenveto sekä muut kortin tiedot palautetaan asynkronisella kutsulla näkymään.

Otsikko	Arvioijan arvio
1. Tulevaisuus, tavoitteet ja toiveet	
1.2 Muutoksen toteuttaminen	**5**
1.1 Tulevaisuuden suunnitelmat, tavoitteet ja toiveet	**2**
2. Koulutus	
2.1 Peruskoulutus	**2**
2.2 Työllistymistä edistävä koulutus	**1**
3. Työ	
3.2 Työllistymisen edistäminen	**5**
3.1 Työttömyydestä selviytyminen	**4**

KUVA 11. Kortin yhteenveto

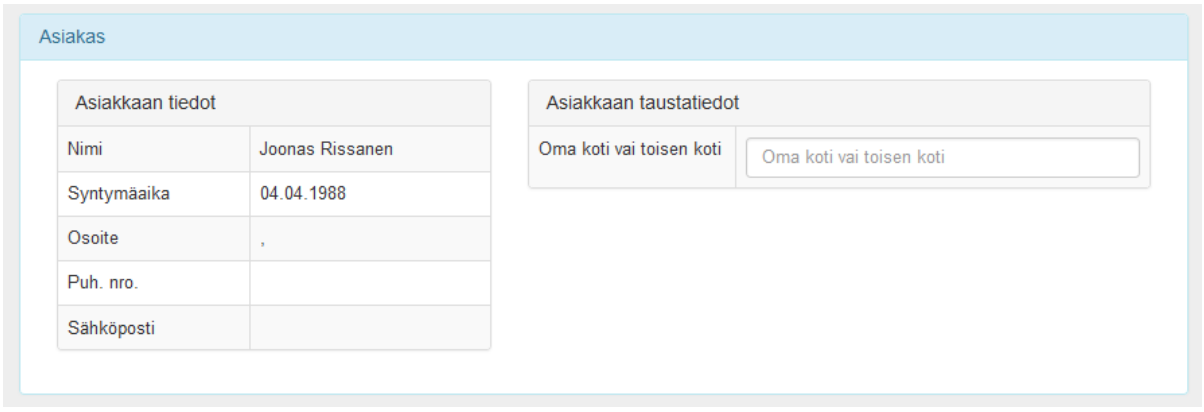
4.4.6 Asiakaskortti

Asiakaskortti näytetään kysymyspatteristonäkymässä (KUVA 12), jossa voi katsella ja muokata asiakaskortin kysymyspatteriston ja taustatietojen arvoja. Muokkaus on mahdollista vain, jos kortti on keskeneräinen.

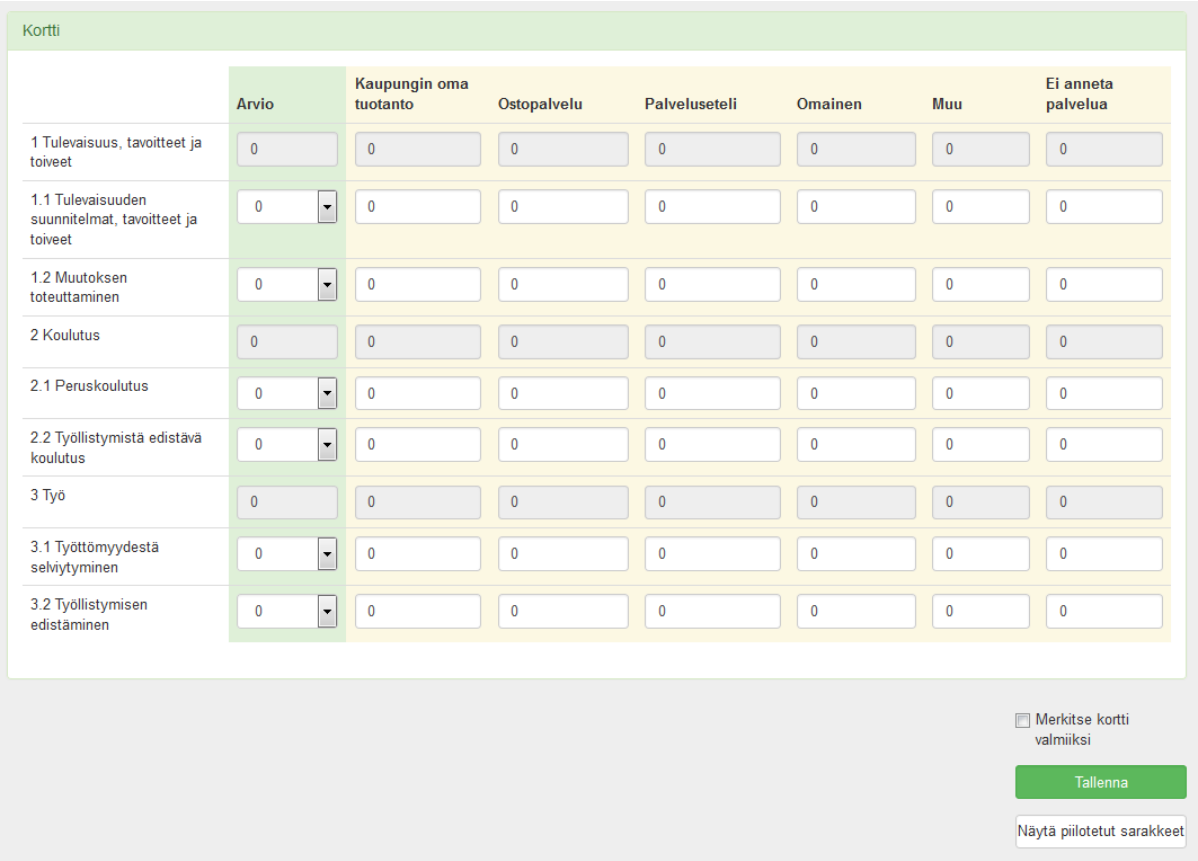


KUVA 12. Asiakaskorttinäkymä

Asiakaskorttinäkymä on jaettu kahteen sisäiseen paneeliin: asiakas (KUVA 13) ja kortti (KUVA 14). Asiakaspaneelissa näytetään asiakkaan oleelliset tiedot, jotta tiedetään, että ollaan oikean asiakkaan kortissa sekä patteristokohtaiset taustatiedot, joihin voidaan tallentaa muita asiakkaan tietoja.

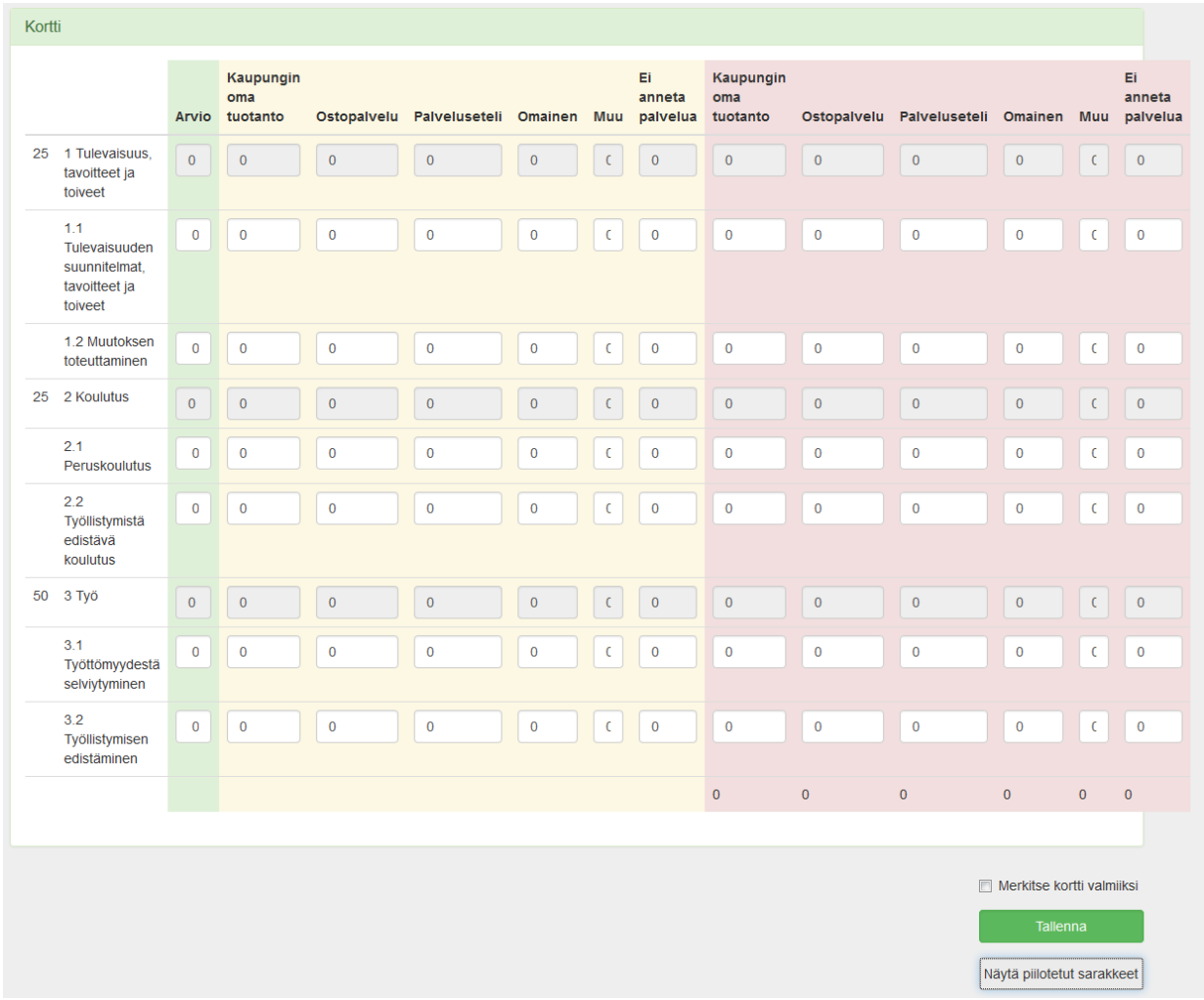


KUVA 13. Asiakkaan tiedot asiakaskortissa



KUVA 14. Asiakaskortti valmiina tietojen syöttöön

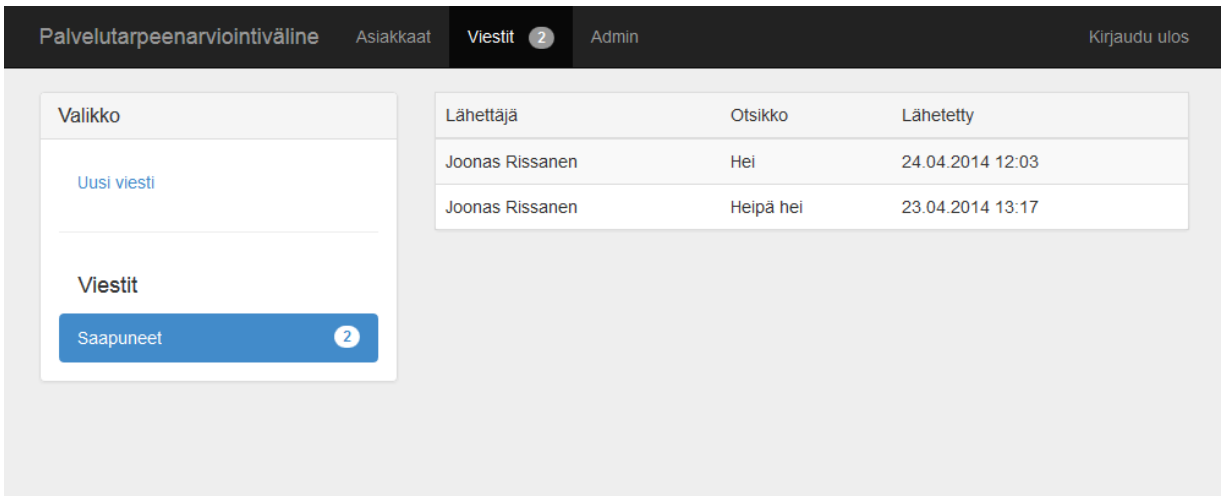
Korttipaneelissa näytetään asiakaskortin kysymyspatteriston lukuarvotaulukko, johon syötetään vastausarvot, jotka taulukko laskee automaattisesti. Käyttäjä voi myös näyttää laskennassa käytetyt piilosarakkeet tarpeen vaatiessa (KUVA 15).



KUVA 15. Asiakaskortin piilotetut laskentasarakkeet

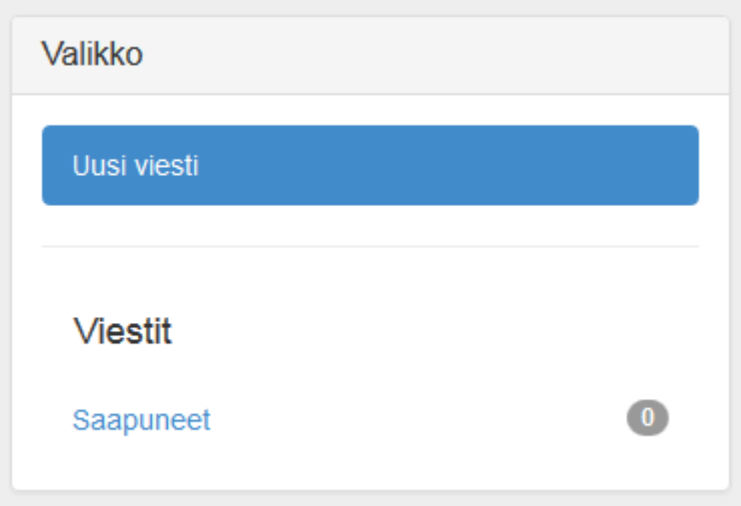
4.5 Viestinäkymä

Viestinäkymässä (KUVA 16) on arviointivälineen sisäisen viestintäjärjestelmän prototyyppi, jolla voi lähettää ja vastaanottaa viestejä järjestelmän käyttäjien kesken. Viestinäkymässä navigoidaan vasemmalla olevan valikon avulla (KUVA 17). Viestit ovat prototyyppitasolla yksinkertaisia sisältäen itse viestin, otsikon, lähettäjän, vastaanottajat ja lähetysajan. Vastaanottaja voi myös merkitä viestin luetuksi tai lukemattomaksi.



KUVA 16. Viestinäkymä

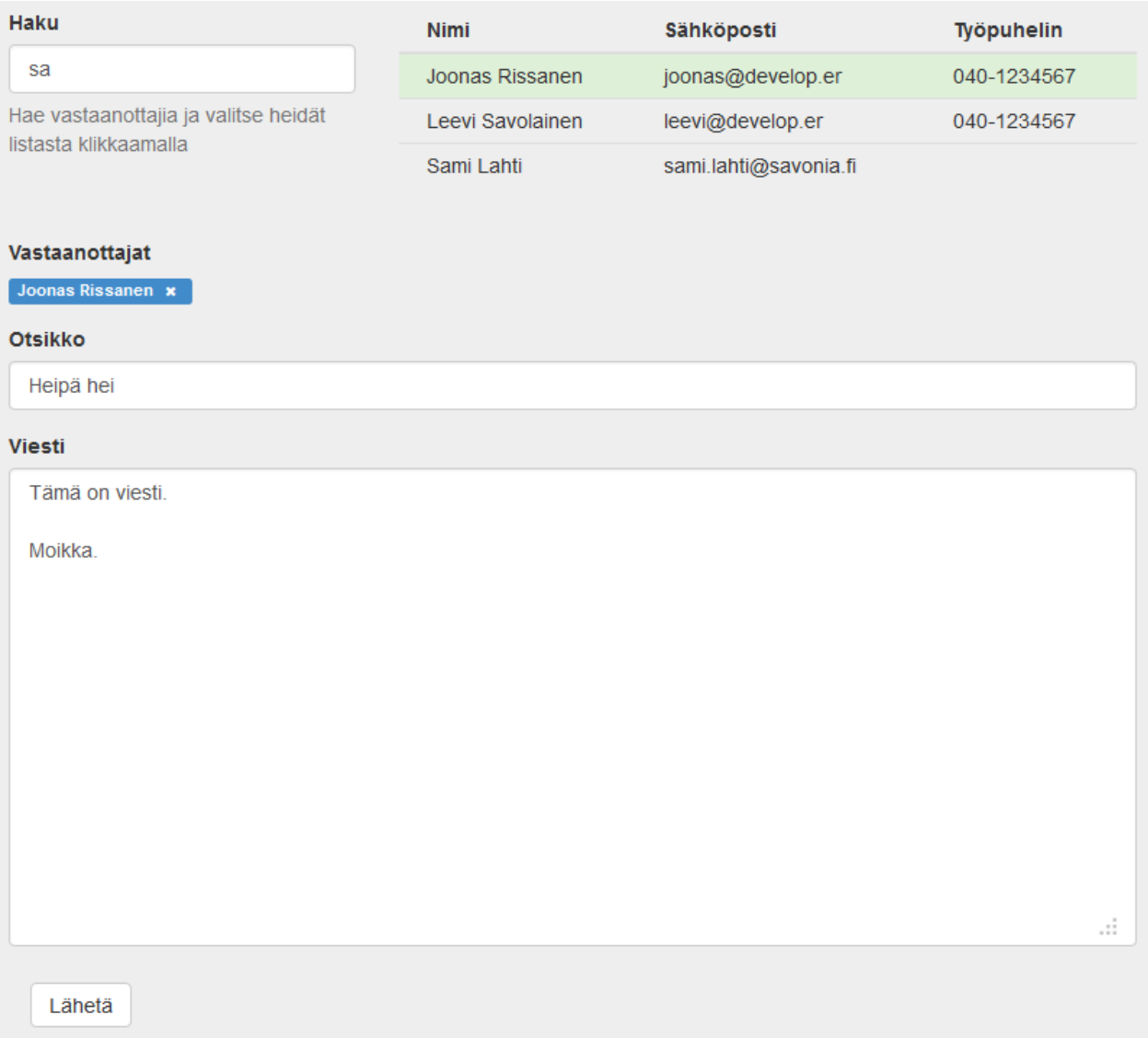
Viestinäkymä koostuu valikosta, josta voi valita uuden viestin kirjoittamisen tai katsoa saapuneita viestejä. Prototyyppiin oli tehty myös lähetetyt- ja roskakorinäkymät, mutta ne eivät olleet täysin valmiita projektin määräajan loppuun mennessä eivätkä vaikuttaneet itse prototyyppisovelluksen esiteltävyyteen, joten ne poistettiin näkyvistä.



KUVA 17. Viestintävalikko

4.5.1 Uusi viesti

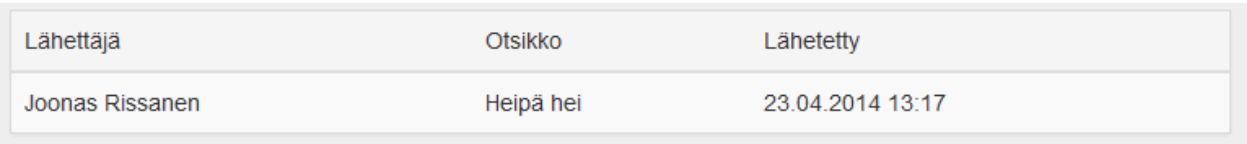
Uuden viestin lähettämisessä haetaan järjestelmästä nimen perusteella käyttäjät, joille viesti halutaan lähettää. Hakuikkuna näyttää dynaamisesti tekstikenttään syötetyn merkkijonon perusteella kaikki löytyvät nimet etunimen ja sukunimen perusteella (KUVA 18).



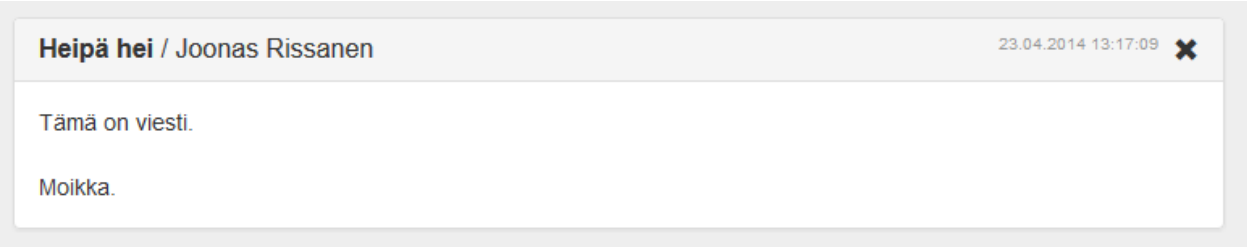
KUVA 18. Uuden viestin luonti

4.5.2 Saapuneet

Saapuneet viestit listataan oletuksena ensimmäiseksi käyttäjän siirtyessä viestinäkymään (KUVA 19). Viestiä klikkaamalla voi katsoa viestin sisällön (KUVA 20). Prototyyppiin oli implementoitu paljon funktionaalisuutta, kuten viesteihin vastaus, viestin poisto ja luetuksi merkkaaminen. Kaikkea ei saatu valmiiksi ja esittelykuntoon, joten keskeneräiset toiminnot jätettiin pois.



KUVA 19. Saapuneet viestit

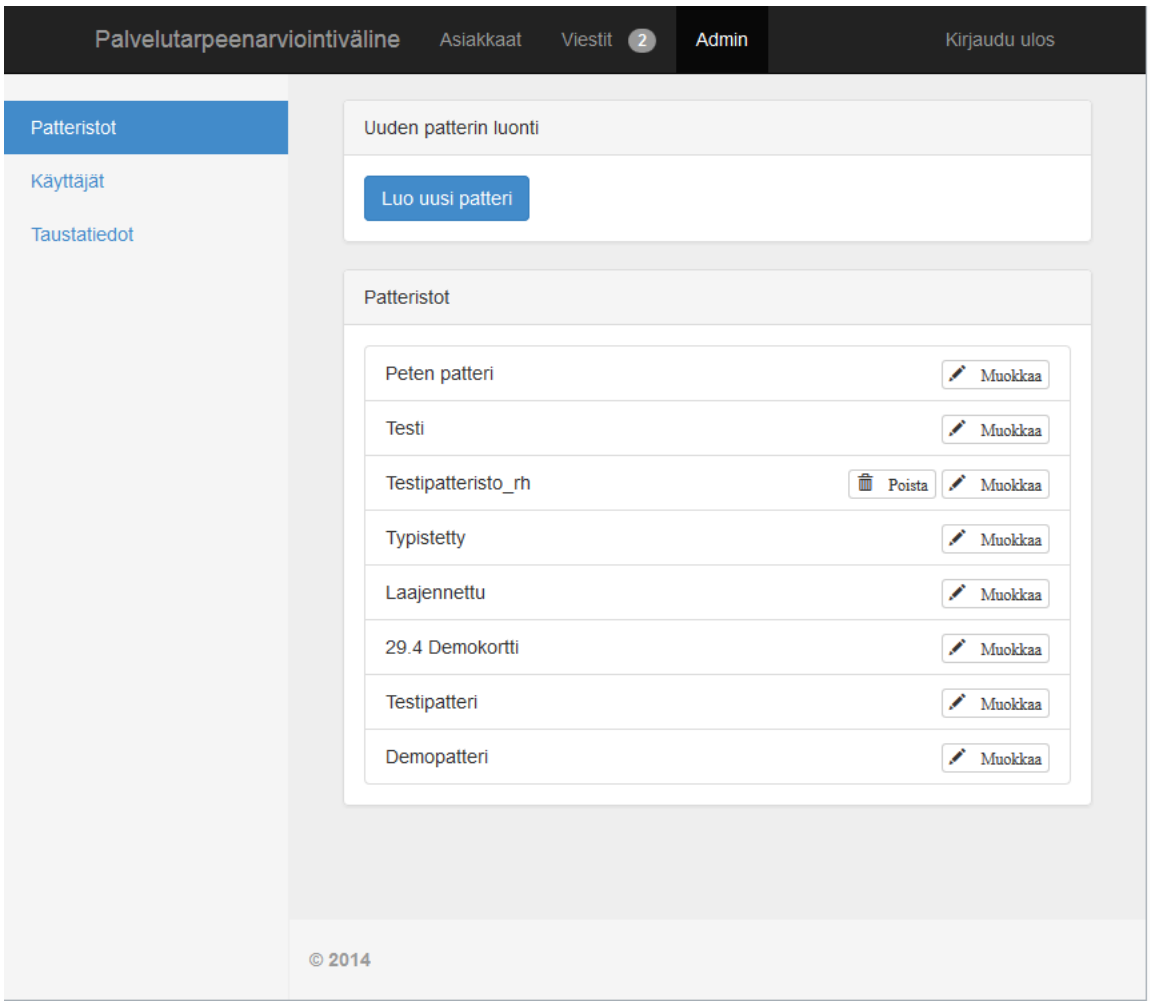


KUVA 20. Saapunut viesti

4.6 Pääkäyttäjänäkymä

Pääkäyttäjänäkymä (KUVA 21) on järjestelmän hallintapaneeli, josta käsin voi luoda uusia pattereita, käyttäjiä ja pattereiden taustatietoja. Pääkäyttäjänäkymään pääsee käsiksi vain käyttäjät, joilla on pääkäyttäjän oikeudet.

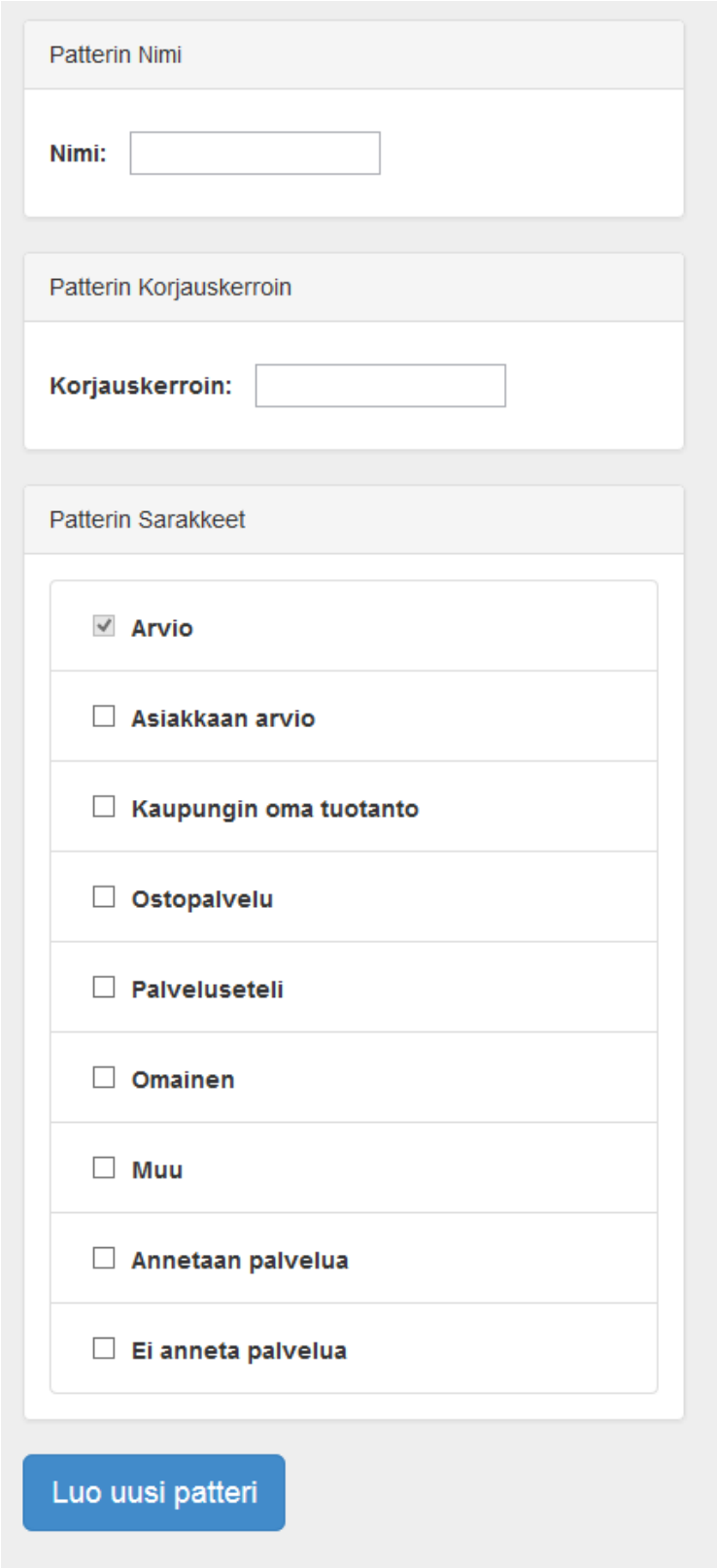
Näkymä koostuu vasemmalla olevasta navigaatiolistasta sekä itse näkymästä, johon on navigoitu.



KUVA 21. Pääkäyttäjänäkymä ja patteristojen hallinta

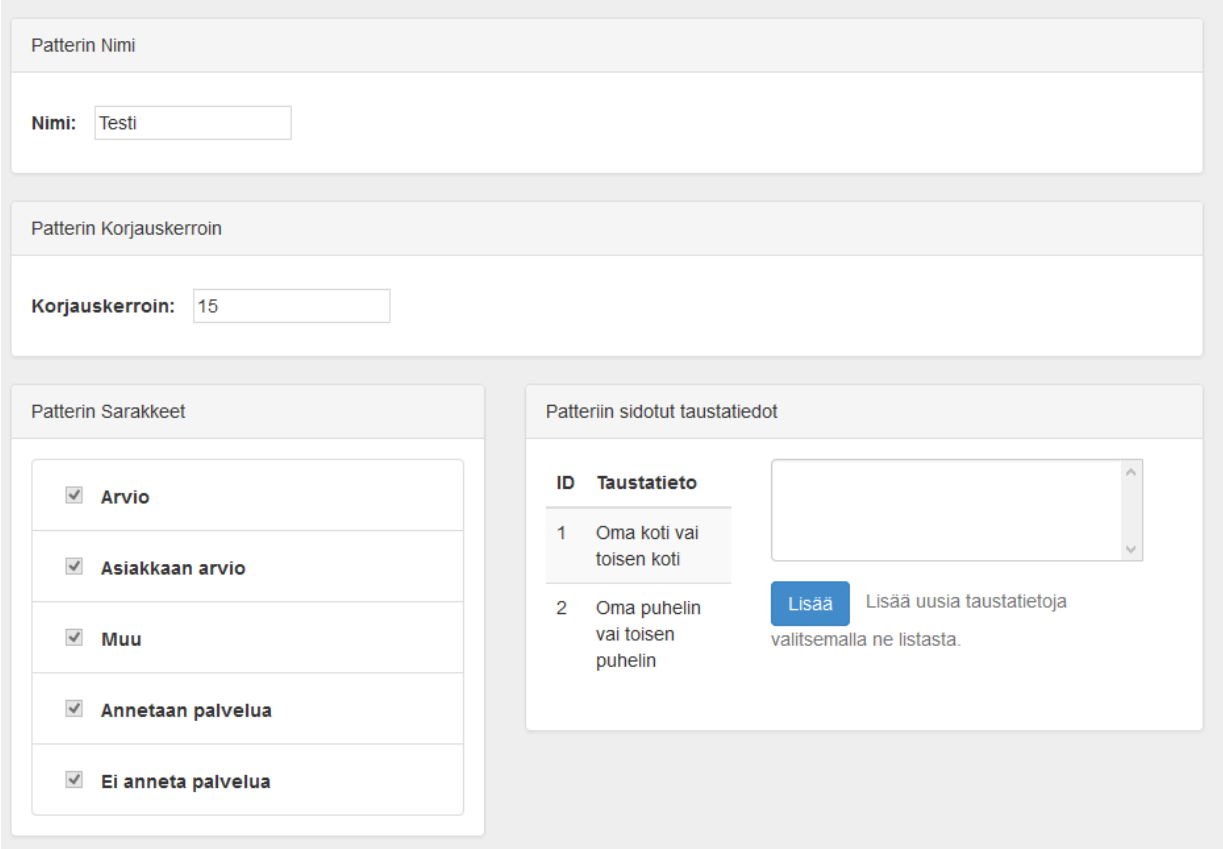
4.6.1 Patteriston luonti ja muokkaus

Uuden patteriston luonti aloitetaan nimeämisellä ja korjauskertoimen syöttämisellä. Patteristolle määritellään sarakkeet merkitsemällä tarvittavat sarakkeet listasta (KUVA 22).

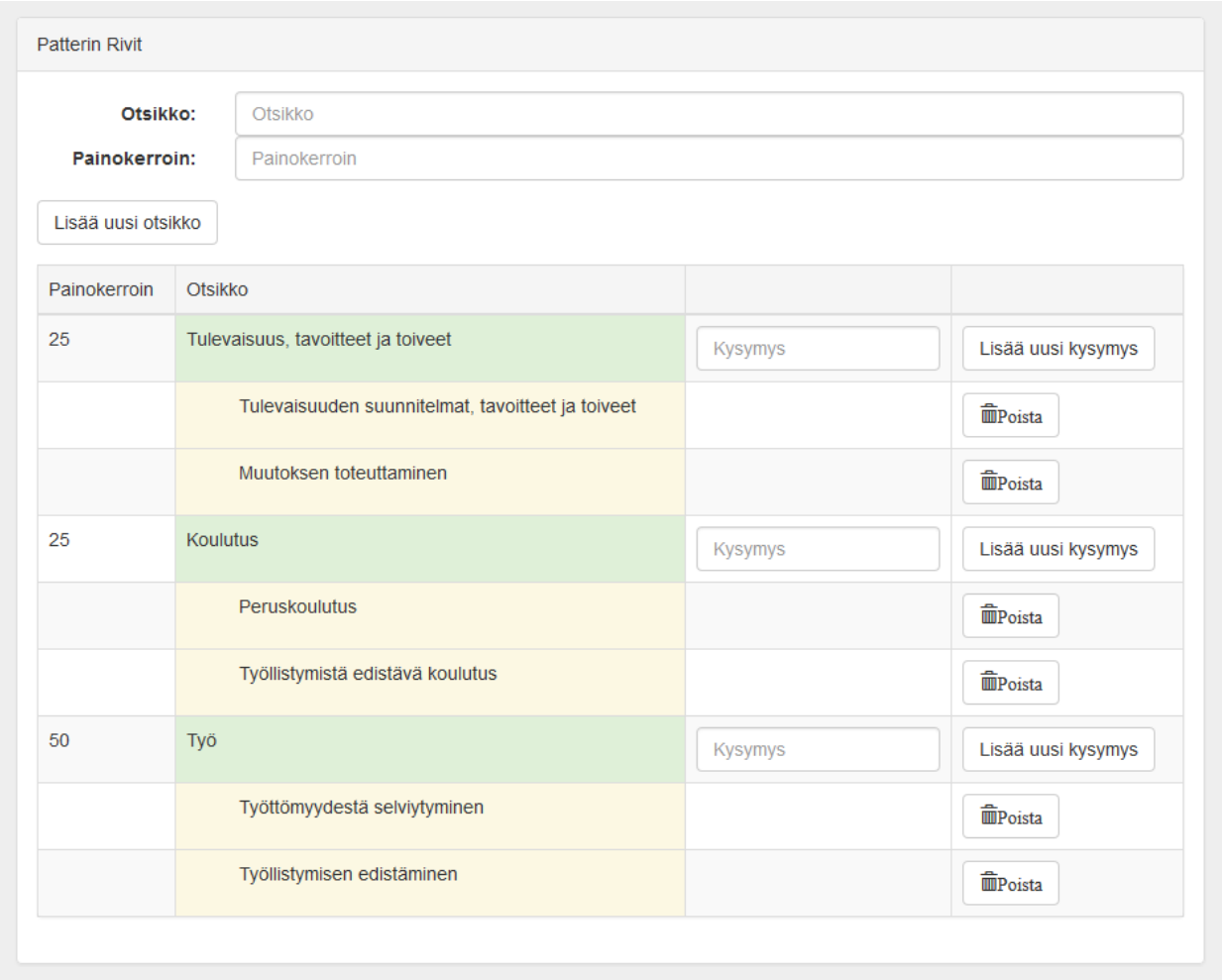


KUVA 22. Uuden patteriston luonti

Patteriston luonnin jälkeen patteristo siirtyy patteristohallinnassa olevaan listaukseen, josta käsin päästään muokkaamaan patteristoa. Muokkauksessa voi sitoa tarvittavia taustatietoja patteristoon (KUVA 23) sekä lisätä kysymyksiä itse patteristoon (KUVA 24). Kysymykset jaetaan kategorioihin otsikon alle. Jokaisella kategorialla on oma painokertoimensa.



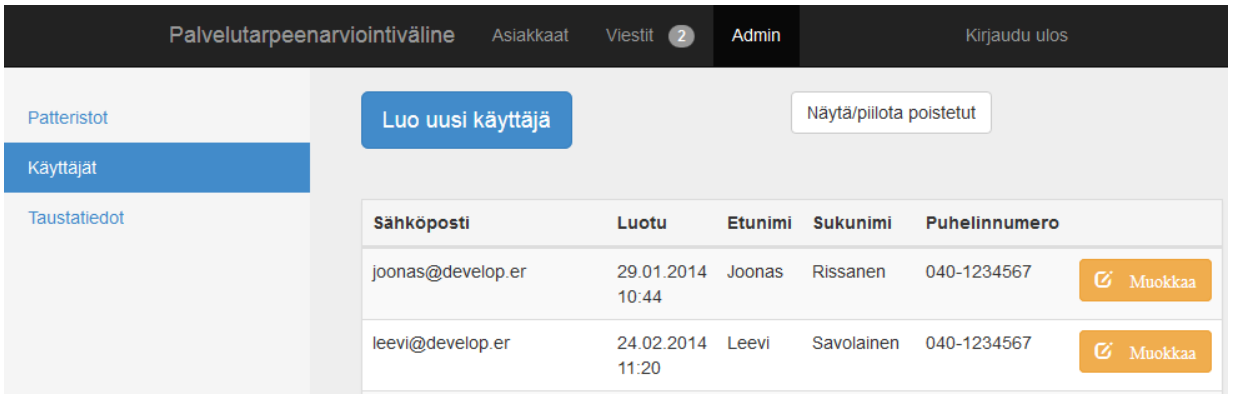
KUVA 23. Patteriston muokkaus



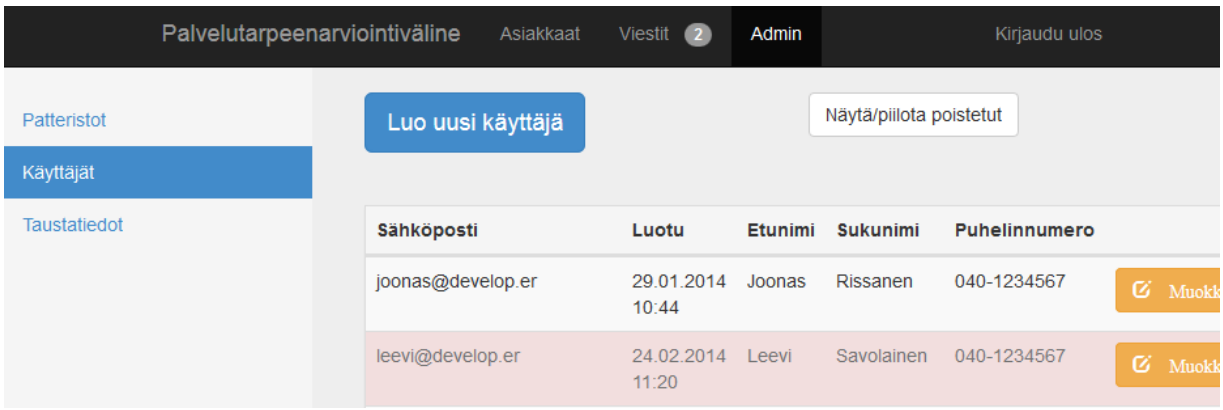
KUVA 24. Patteriston kysymysten muokkaus

4.6.2 Käyttäjähallinta

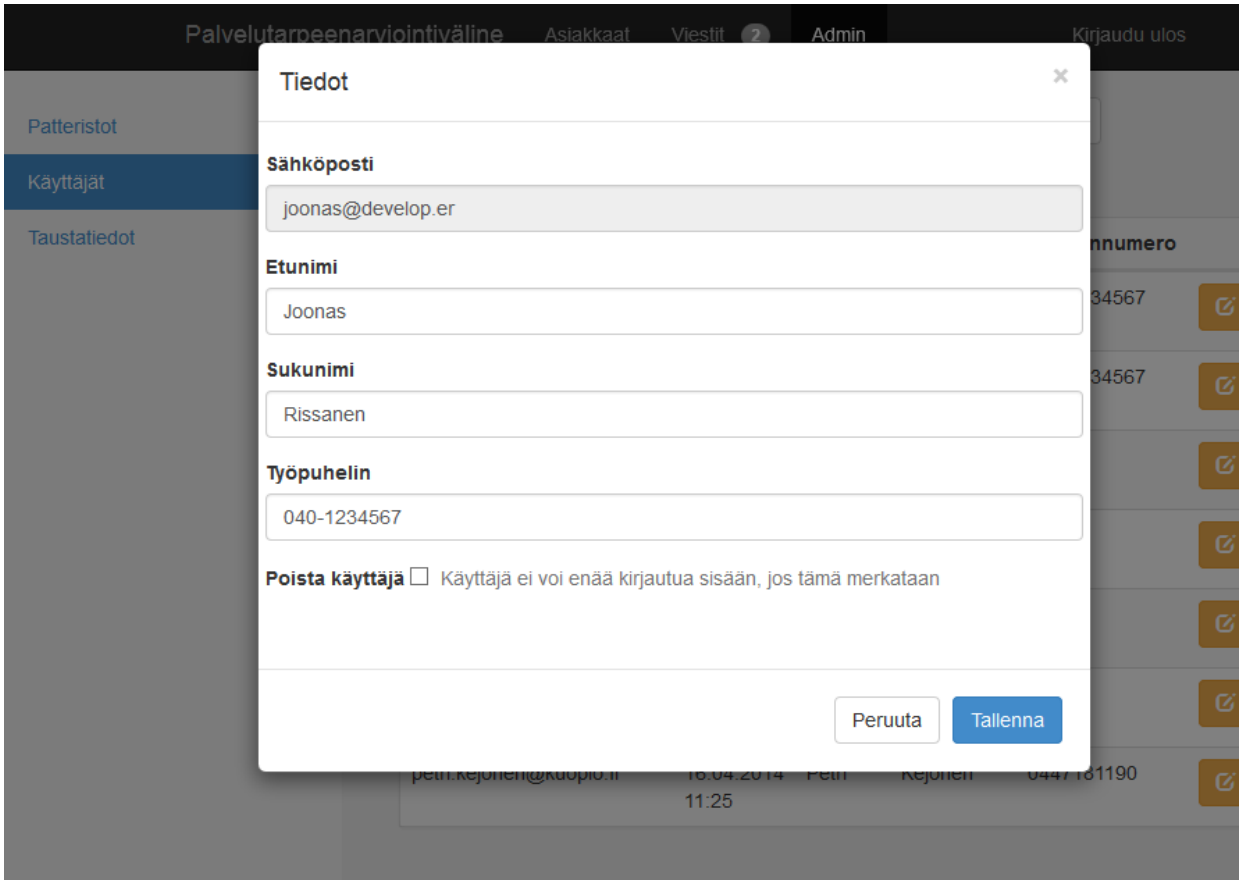
Käyttäjähallinta (KUVA 25) on prototyypissä minimalistinen ja lähinnä demonstrointiin riittävä. Pääkäyttäjä voi lisätä uusia käyttäjiä (KUVA 28), muokata käyttäjien tietoja (KUVA 27) tai poistaa käyttäjän järjestelmästä (KUVA 27 ja KUVA 26).



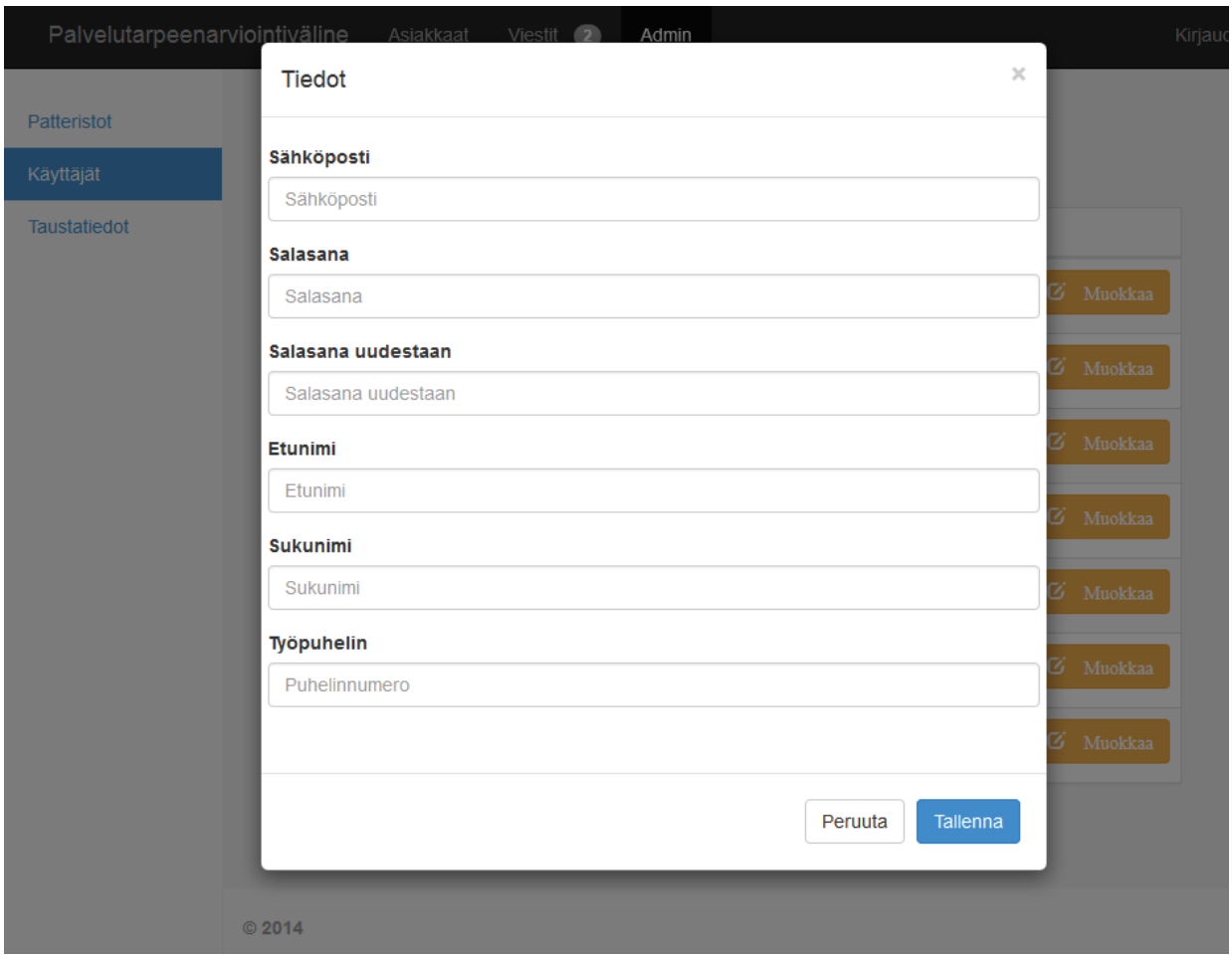
KUVA 25. Käyttäjähallinta



KUVA 26. Poistettu käyttäjä näkyy punaisella



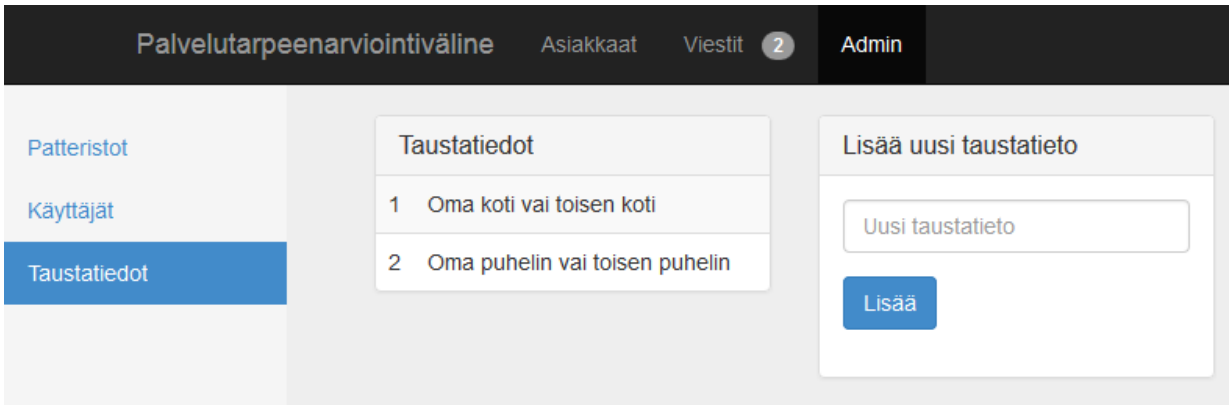
KUVA 27. Käyttäjän muokkaus



KUVA 28. Uuden käyttäjän luonti

4.6.3 Taustatietojen hallinta

Taustatietojen käsittelyssä järjestelmään voidaan lisätä taustatietoja (KUVA 29), jotka voidaan sitoa patteristoihin patteristohallinnassa. Ristiriitatilanteiden välttämiseksi taustatietojen poisto, muokkaaminen tai vanhentuneeksi merkitseminen jätettiin pois prototyypistä.



KUVA 29. Taustatietojen listaus ja lisäys

5 KÄYTTÖLIITTYMÄ JA KÄYTETTÄVYYS

5.1 Ajax

Ajax muodostuu tärkeäksi osaksi sujuvaa verkkokokemusta, jos kyseessä on palvelimen ja selaimen välistä kommunikointia muulloinkin kuin verkkosivun latausvaiheessa (asynkroninen kommunikaatio). Ajax mahdollistaa reaaliaikaisten käyttöliittymäkomponenttien käyttöönoton toimimalla palvelimen ja päätteen välisenä rajapintana.

Näin esimerkiksi vältetään sivujen uudelleen lataaminen takaisin samaan näkymään. Pääte voi suorittaa pyynnön ajax-rajapinnalla palvelimelle ohjelman käytön aikana ja palvelin voi palauttaa tietoa päätteelle JSON-muotoisena, jonka voi kääntää esimerkiksi javascriptillä tarvittavaan muotoon. Palvelin voi myös palauttaa pelkkää tekstiäkin, kuten valmista HTML:ää, jota ei tarvitse muokata päätteelle sopivaksi.

5.2 Sivujen uudelleen latauksen välttäminen välilehtien avulla

Sivujen uudelleen lataaminen kerta toisensa jälkeen on kaistankäytöltään hyvin tehotonta datan uudelleenlataamista, vaikeasti ylläpidettävää lomakepainotteisissa ohjelmissa sekä käyttökokemusta hidastavaa ja vaikeuttavaa. Arviointivälineessä on useita lomakkeita, joiden sujuvassa käyttämisessä koko sivun uudelleen lataaminen olisi äärimmäisen tehotonta ajan- sekä verkkokaistankäyttöä.

Välilehti-malli mahdollistaa nopean näkymän vaihtamisen ilman datan katoamista tai datan lähettämistä uudelleen POST- tai GET-metodina.

5.3 Väritys, asettelu ja (väärät) valinnat

Bootstrap mahdollistaa helpon ja nopean käyttöliittymäelementtien värityksen CSS-luokilla ”alert”, ”warning”, ”default” ja ”success”, jotka kääntyvät punaiseksi, keltaiseksi, siniseksi ja vihreäksi. Kuten luokkien nimistä voidaan päätellä, eri värit soveltuvat eri tarpeisiin hyvinkin loogisesti. Punaista esimerkiksi voidaan käyttää varoituksena siitä, että käyttäjä on tekemässä mahdollisesti peruuttamattomia muutoksia.

Värityksen lisäksi on tärkeää asettaa elementit järjestykseen, jota käyttäjä seuraa intuitiivisesti. Esimerkiksi länsimaissa lukusuunta on vasemmalta oikealle käytännössä joka paikassa ja lomakkeet täytetään ylhäältä alas. Hyvällä asettelulla ja värien oikeaoppisella käytöllä voidaan täten välttää sekavuus käytettävyydessä.

On myös tärkeää poistaa käyttäjältä turhat tai väärät valinnat käytöstä. Oletuksena käyttöliittymää suunnitellessa käyttäjä ei ole täydellinen ja tulee tekemään virheitä, jos hänelle annetaan siihen mahdollisuus. Näillä metodeilla on tarkoitus minimoida käyttäjien virheet.

6 VIIMEINEN POHDINTA

Prototyypin keskeiset ominaisuudet saatiin valmiiksi ja lopputulos osoitti projektin mahdollisuudet. Tavoitteet saada toimiva demonstraatio saavutettiin. Projektin parissa työskentely jatkossa ei kuminkaan ollut mahdollista tulevaisuudessa, mikä oli pettymys. Kuopion kaupunki valitsi Istekin toteuttajaksi lopulliselle työlle.

Projekti saatiin alkuun hyvin nopeasti aikaisemman kokemuksen ja CodeIgniter-osaamisen ansiosta. Parityönä myös projektin suorittaminen helpottui huomattavasti. Pahimmat riskit projektia yksin tehdessä ovat aina motivaation hankkiminen ja ongelmien ratkaisu varsinkin, jos samalla joutuu opettelemaan paljon uusia asioita. Projekti opetti ryhmätyön tärkeyden.

Loppujen lopuksi projektissa ei ilmennyt suuria ongelmia. Mitään korjaamatonta tai ylitsepääsemätöntä ongelmaa ei tullut vastaan ja kaikki pienet esteet olivat sopivan haastavia ja hyviä oppimismahdollisuuksia. Projekti päinvastoin herätti paljon kiinnostusta myös vapaa-ajalla. Projekti onnistui mielestäni siis hyvin enkä olisi merkittävästi tehnyt mitään toisin.

LÄHTEET JA TUOTETUT AINEISTOT

Beck, Kent;ym. 2001. Ketterän ohjelmistokehityksen julistus. *Ketterän ohjelmistokehityksen julistus.* [Online] 2001. [Viitattu: 23. 10 2014.] http://agilemanifesto.org/iso/fi/.

EllisLab. 2014. CodeIgniter / EllisLab. *EllisLab.* [Online] EllisLab, 17. 4 2014. [Viitattu: 17. 4 2014.] http://ellislab.com/codeigniter.

Pitt, Chris. 2012. *Pro PHP MVC.* s.l. : Apress, 2012. 9781430241645.