Jarkko Lind

# Vieraskirjajärjestelmän digitalisointi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Automaatiotekniikka

Insinöörityö

21.12.2015

Metropolia

| | |
|---|---|
| Tekijä(t)<br>Otsikko<br><br>Sivumäärä<br>Aika | Jarkko Lind<br>Vieraskirjajärjestelmän digitalisointi<br><br>28 sivua + 18 liitettä<br>21.12.2015 |
| Tutkinto | Insinööri (AMK) |
| Koulutusohjelma | Automaatiotekniikka |
| Suuntautumisvaihtoehto | |
| Ohjaaja(t) | Kimmo Kokko, ohjelmistoryhmän ryhmäpäällikkö<br>Timo Tuominen, lehtori |

Opinnäytetyön tarkoituksena oli kehittää sähköinen vieraskirjasovellus korvaamaan Millog Oy:n tietokonelaboratorion alkuperäistä paperista vieraskirjajärjestelmää. Uuden järjestelmän lopulliset vaatimukset selvisivät työn edetessä, johtuen työn ketterästä toteutustyylistä.

Työn kehitysympäristönä oli Millog Oy:n Riihimäen toimipisteen ohjelmistoryhmän toimisto ja tietokonelaboratorio. Sovellustiedostot sijoitettiin tietokonelaboratorioon pystytetylle palvelimelle, josta on lähiverkkoyhteys vieraskirjasovellusta pyörittävään päätelaitteeseen.

Vieraskirjasovelluksen ohjelmointikielen valinta kohdistui verkko-ohjelmointikieliin PHP, JavaScript ja HTML, johtuen palvelimen Linux-alustasta. Näiden kielien avulla vaatimusten kattaminen oli helppoa ja yksinkertaista.

| | |
|---|---|
| Avainsanat | Verkkosovellus, JavaScript, Vieraskirja |

| | |
|---|---|
| Author(s)<br>Title<br><br>Number of Pages<br>Date | Jarkko Lind<br>Digitalization of Guest Book System<br><br>28 pages + 18 appendices<br>21 December 2015 |
| Degree | Bachelor of Engineering |
| Degree Programme | Automation Engineering (AMK) |
| Specialisation option | Manufacturing Automation |
| Instructor(s) | Kimmo Kokko, Software Group Leader<br>Timo Tuominen, Senior Lecturer |

The purpose of the Bachelor's thesis was to develop a digital version of a guestbook system to replace the original paper-version guestbook -system.
Development style of the thesis was agile, so the final requirements of the new system were clarified in the development process.

The development environment and the destination of the guestbook system was the office and the laboratory of Millog Oy in Riihimäki. Files for the guestbook application were placed in the server located in the software group laboratory. The application is running in the terminal device which is in the same network as the server.

The used programming languages were web programming languages and they were selected over other programming languages solely because they were platform independent. Used languages were PHP, JavaScript and HTML. With these web programming languages, the requirements of the application were met easily.

| | |
|---|---|
| Keywords | web application, JavaScript, guestbook |

## Sisällys

## Lyhenteet

| | |
|---|---|
| AJAX | Asynchronous JavaScript and XML. Asiakaspuolen ohjelmoinnissa käytettyjen ohjelmointitekniikkojen kokoelma. |
| CSRF | Cross-site Request Forgery. Eräänlainen verkkohyökkäystekniikka. |
| DOM | Document Object model. HTML-sivun puurakenne. |
| HTTP | HyperText Transfer Protocol. Verkkoliikenteessä käytetty tiedonsiirtoprotokolla. |
| HTTPS | HyperText Transfer Protocol Secured. HTTP-protokolla TLS tai SSL-salauksella. |
| MitM | Man in the middle attack. Eräs verkkohyökkäystekniikka. |
| Regex | Regular expression. Säännöllinen lauseke. Yksinkertainen merkkijonokieli, jota käytetään lukuisissa ohjelmointikielissä merkkijonon rakennevertauksen referenssirakenteena. |
| SSL | Secure Sockets Layer. Vanhentunut salaustekniikka HTTP-tiedonsiirtoprotokollaa käytettäessä. |
| SVG | Scalar vector graphics. XML-pohjainen tiedosto joka kuvailee graafista sisältöä vektoreiden avulla. |
| TLS | Transfer Layer Security. Uusin HTTPS-tiedonsiirtoprotokollan salaustekniikka. |
| URL | Unified Resource Locator. Merkkijono, jota käytetään verkkosivun sijaintitietona. |
| WKHTMLTOPDF | Usealle eri alustalle kehitetty komentorivityökalu HTMLl-tiedoston muuntamiseksi PDF-muotoon. |

| | |
|---|---|
| XML | Extensible Markup Language. Laajennettava merkintäkieli. Tiedonsiirtoon tarkoitettu tiedostotyyppi. |
| XSS | Cross-site scripting. Eräs verkkohyökkäystekniikka. |
| BASE64 | Tietovirtojen lähetyksessä käytetty enkoodaustekniikka. |

# 1 Johdanto

Talouskasvun paikallaan polkeminen, jatkuvat säästötavoitteet sekä työttömyys nostaa keskusteluun säästämiseen ja tehokkuuteen liittyviä aiheita. Niin julkiset kuin yksityisetkin yritykset pyrkivät tehostamaan palveluita ja vähentämään kustannuksia kaikilla mahdollisilla tavoilla.

Lähes kaikki prosessit on mahdollista toteuttaa sähköisinä, jolloin tietoa on helpompi manipuloida, varastoida ja hakea. Tiedonhaku on moninkertaisesti tehokkaampaa ja nopeampaa sähköisistä arkistoista kuin perinteisistä paperisista arkistoista. Tämän lisäksi sähköisten arkistojen varastointitilojen ylläpitokustannukset ovat pienemmät ja turvajärjestelyt ovat helpompia järjestää.

Arkistonnin sähköistäminen on tehtävä oikein, suunnitellusti ja kokonaisuutta tukevaksi, jotta siitä saatavan hyödyn maksimointi olisi mahdollista.

Pienten tietomäärien lyhytaikainen varastoiminen voidaan toteuttaa pienillä kustannuksilla myös perinteisellä arkistointimenetelmällä. Suuremman tietomäärän ollessa varastoituna sähköinen arkistointi perinteiseen verrattuna tuottaa huomattavia säästöjä.

Tulevaisuutta ajatellen on järkevää suunnitella, kannattaako perinteiset arkistot muuntaa sähköisiksi ennen tietomäärien mahdollista kiihtyvää kasvua. Arkiston analysoinnilla ja tietomääriä ennakoimalla voidaan todeta, onko sähköiseen arkistointiin siirtyminen järkevää.

Arkistojen sähköistämisen hyötyjen selvittämisessä on lähdettävä liikkeelle säilytettävien arkistoiden vaatimuksista. Pitääkö tietoja saada ulos jatkuvasti? Onko tiedonhaku aikakriittistä eli tarvitaanko tietoja nopeasti reaaliajassa? Kuinka suuria tietovirtoja arkistosta tarvitaan ja kuinka kauan arkistoja säilytetään?

Perinteisestä arkistosta siirtyminen suoraan vain sähköiseen arkistointimenetelmään ei ole aina välttämätöntä, saati kustannustehokasta. Joissain tapauksissa siirtymävaiheessa on tehokkainta käyttää järjestelmiä rinnakkain. Tämänkaltainen ratkaisu on optimaalinen tilanteissa, jossa perinteisen arkiston tietojen säilytyksen

aikajänne olisi suhteellisen lyhyt. Tällä tavalla kahden järjestelmän rinnakkaisuuksien epätehokkuus olisi minimoitu eikä vanhoja arkistoja tarvitse muuntaa sähköisiksi.

## 1.1 Sähköiset palvelut

Sähköisiä palveluita voidaan kutsua myös sähköiseksi asioinniksi. Sähköinen asiointi on jonkun asian hoitamista tai tuotteen hankintaa tietoverkossa olevan palvelun avulla [1]. Sähköisiä palveluita ovat esimerkiksi verkkopankkipalvelut ja kaikenlaiset sähköiset ajanvarauspalvelut. Myös kellokorttijärjestelmät voidaan lukea sähköisiksi järjestelmiksi, sillä ne ovat lähes kaikkialla nykyään sähköisiä. Erona esimerkiksi verkkopankkipalveluiden ja kellokorttijärjestelmien välillä on ainoastaan palvelun paikallisuus.

Palveluiden sähköistämistä on käytetty kustannustehokkuuden ja ylläpidon parantamiseen jo toistakymmentä vuotta [2]. Julkisten palveluiden sähköistäminen on ollut jo jonkin aikaa paineen alla. Tähän on monia syitä, joista päällimmäisinä ovat kuntaliitokset, säästötarpeet sekä harvaanasuttujen alueiden ongelmat.

### 1.1.1 Sähköisten palveluiden hyödyt palveluntarjoajalle

Sähköiset palvelut voivat olla kolmannen osapuolen tarjoamia palveluita tai alkuperäisen palveluntarjoajan oma palvelu. Molemmissa tilanteissa palveluntarjoaja hyötyy joko suoraan tai epäsuorasti sähköisestä palvelusta. Hyötyminen voi liittyä kysynnän ja tarjonnan parempaan kohtaamiseen tai esimerkiksi palvelun tehostamisesta saatuihin säästöihin.

Säästöjä syntyy ylimääräisen päivystyshenkilöstön vähentämisestä, yksinkertaisten toimintojen niputtamisesta ja tiedonsiirron tehostamisesta.

Sähköistäminen mahdollistaa kustannustehokkaan palvelun päivittämisen, helpon versionhallinnan sekä esimerkiksi varmuuskopioinnin. Samankaltaisia palveluita on myös mahdollista monistaa useaan käyttökohteeseen vaivattomasti. Lisäksi

kaikenlaisissa tarkkuutta vaativissa tai muulla tavalla kriittisissä palveluissa voidaan inhimillisen virheen poistaminen laskea suureksi eduksi, jonka ansiosta prosessia voidaan tehostaa huomattavasti.

### 1.1.2 Sähköisten palveluiden hyödyt asiakkaalle

Sähköiset palvelut ovat tehokkaita alueilla, jotka ovat harvasti asuttuja, välimatkat pitkiä, volyymi pientä ja paikallisen palvelun henkilöstö vähäistä. Näillä alueilla sähköinen palvelu kohtaa asiakkaan tehokkaammin, jolloin asiakas hyötyy ajallisesti sekä rahallisesti sähköisistä palveluista.

Esimerkiksi verkkopankkisovellusta käyttävä asiakas hyötyy itse palvelun maksuttomuudesta kuin myös säästöstä matkakustannuksissa sekä palvelun nopeudesta. Verkkopankkia voi myös käyttää vuorokauden ympäri, mikä on hyödyllinen ominaisuus vuorotyötä tekeville.

## 1.2 Millog Oy

Millog Oy on puolustusvoimien sekä Rajavartiolaitoksen strategisena kumppanina toimiva, kunnossapitoon, elinjakson hallinta – ja materiaalipalveluihin keskittynyt suomalainen yritys. Millog on syntynyt puolustusvoimien alaisten palveluiden yksityistämisestä joka tapahtui vuonna 2009. Millog työllistää noin 1000 henkilöä 22 paikkakunnalla ja liikevaihto on noin 150 miljoonaa euroa.

## 1.3 Työn tavoite

Työn tavoitteena oli suunnitella vieraskirjasovellus Millog Oy:n Riihimäen toimipisteen suojattuun tilaan. Suojatulla tilalla tarkoitetaan tässä työssä suojausluokan 2 tilaa, joka on rajattu vain sinne oikeutettujen henkilöiden käyttöön.

Vieraskirjasovelluksella tarkoitetaan tässä työssä sovellusta, joka vastaa esimerkiksi vierailijoiden nimen, vierailuajankohdan ja vierailun keston arkistoinnista. Vastaavassa käytössä on tähän asti ollut paperinen lomake, jonka sähköistäminen on ollut suunnitteilla jo jonkin aikaa.

Pääsyy järjestelmän sähköistämisessä ei ole ainoastaan rahallinen säästö, sillä vaikka paperiset arkistot vaativat runsaasti säilytystilaa, tulee arkistoitavaa tällä hetkellä suhteellisen vähän. Toisaalta arkistoituja tietoja on säilytettävä pitkiä aikoja, jolloin pienikin volyymi kasvattaa vuositasolla suuret arkistot. Sähköisen järjestelmän tavoitteena on tässä projektissa tehostaa tiedonhakua ja parantaa tiedonvälityksen tietoturva-asioita.

Vieraskirjasovelluksen kehitystapa otti ketterästä ohjelmistokehityksestä vaikutteita, sillä sovelluksen perusvaatimukset olivat selvillä ja vaatimuksia tarkennettiin ja lisättiin työn edetessä. Alustavasti vaatimuksena oli yksinkertaisen toiminnallisuuden omaava sovellus, joka voisi korvata paperisen kirjautumisjärjestelmän. Sovelluksen asteittainen kehittäminen pienensi riskiä luoda turhia ja käyttämättömiä ominaisuuksia sovellukseen.

Työssä tullaan kertomaan yleisesti työn edellyttämien tekniikoiden käyttötavasta sekä rakenteesta. Lopuksi käydään läpi mahdollisesti sovellukseen tulevat laajennukset sekä kehityskohteet.

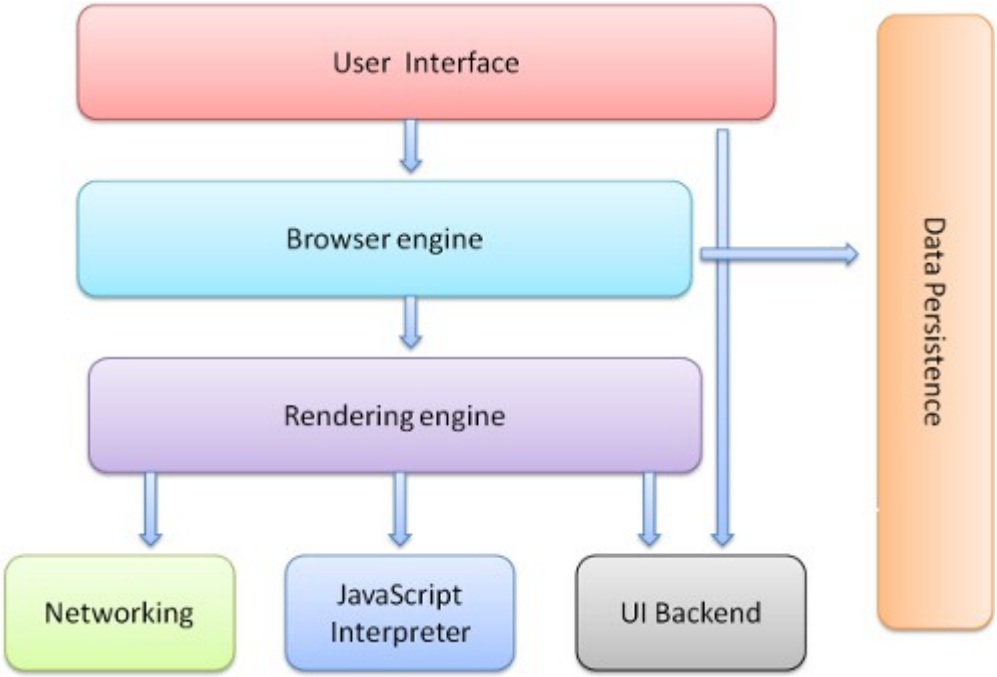
# 2 Vieraskirjasovelluksen tekniikat

## 2.1 Selain ja tiedonsiirtoprotokollat

Selain on tietokonesovellus, jossa on helppokäyttöinen graafinen käyttöliittymä verkkosivujen katseluun. Selaimella on mahdollisuus ladata tiedostoja ja siirtää tietoa oman koneen ja palvelimen välillä. Palvelimien kanssa kommunikoidaan pääosin HTTP-protokollaa käyttäen mikä mahdollistaa tietojen ja tiedostojen pyytämisen ja vastaanottamisen palvelimilta. [3.] Yleisimpiin nykyisin käytettäviin moderneihin selaimiin kuuluvat Google Chrome, Safari, Firefox, Internet Explorer sekä Opera.

Kuva 1 esittää selaimen rakennetta pääkomponenteittain toiminnan mukaan eroteltuna. Selaimen tärkein komponentti on *User Interface* eli käyttöliittymä. Käyttöliittymään kuuluvat osoiterivi, työkalurivit sekä kaikki muut osat paitsi selaimen pääikkuna. *Browser engine* on selaimen moottori, jonka tehtävänä on toimia rajapintana renderöintimoottorille. *Rendering engine* eli renderointimoottori on komponentti, joka näyttää palvelimelta pyydetyn sivun sisällön selaimen pääikkunassa. HTML- ja CSS-

muotoilut jäsennetään tämän moottorin toimesta. *Networking*- eli verkkotyöskentelykomponentti suorittaa erityyppisten pyyntöjen lähetyksen verkon yli. Esimerkiksi HTTP- ja FTP-tyyppiset pyynnöt suoritetaan tämän komponentin toimesta. *JavaScript Interpreter* eli JavaScript-tulkki on komponentti, jota käytetään suoritettaessa JavaScript-tyyppisiä rakenteita. Selain käyttää *UI Backend* -komponenttiä ikkunoiden sekä muiden pienoisohjelmien luomisessa. UI-kirjainlyhenne tulee sanoista *user interface* ja *Backend* kuvastaa käyttäjältä suoranaisesti piilossa olevaa osiota. *Data Persistence* eli tiedon säilytys on komponentti, joka huolehtii selaimessa säilytettävistä tiedoista, kuten evästeistä. [4.]




Kuva 1.

Tiedonsiirtoprotokollat

Yleisin käytössä oleva tiedonsiirtoprotokolla on HyperText Transfer Protocol eli HTTP, joka on sovellustason tiedonsiirtoprotokolla. Tämä tiedonsiirtoprotokolla on ollut perustana maailmanlaajuiselle verkolle eli internetille vuodesta 1990. [5.] Muita yleisiä protokollia ovat FTP eli file transfer protocol, HTTPS eli HyperText Transfer Protocol Secure ja RTSP eli Real-Time Streaming Protocol.

HTTPS-protokolla on http-protokollaa vastaava SSL- tai TLS-menetelmällä salattu yhteys. HTTPS-protokollan käyttö mahdollistaa tietoturvallisen tiedonsiirron asiakkaan ja palvelimen välille. Tiedonsiirron turvaaminen perustuu verkkosivun suunnitteluun, joka mahdollistaa vain tietyn avaimen sisältämän tiedon välittämisen. Tämän avaimen nuuskiminen ja täten myös replikointi olisi mahdollista ilman salattua yhteyttä.

Sovelluksessa on käytetty HTTP-protokollaa, sillä järjestelmällä ei ole yhteyttä ulkoiseen verkkoyhteyteen. Tästä johtuen lähetystietojen salaaminen ei ole olennaista. Lähiverkkoyhteys on myös erittäin rajattu, joten istuntoavaimien joutumista ulkopuolisten käsiin ei tarvitse pelätä. Sivustolla tiedonsiirto tapahtuu GET- ja POST-metodeilla, joiden parametrit ovat tilanteesta riippuen määritetty joko HTML-lomakkeessa tai JavaScript-funktiossa.

Sovelluksen tiedonsiirto olisi ollut mahdollista toteuttaa käyttäen ainoastaan GET-metodia, joka on hieman nopeampi toiminnaltaan kuin POST. Muutamalla tietyllä selaimella käytettynä POST-metodi ottaa palvelimeen yhteyttä useamman kerran jokaista kutsua kohden, mikä voi kuormittaa palvelinta. Tämänkaltaisen toiminnan takia POST-metodi on myös hitaampi toiminnaltaan kuin GET-metodi [6]. GET-metodissa parametrit enkoodataan ja sijoitetaan urliin, kun POST-metodissa arvot sijoitetaan lähetettävän sivun runkoon. Tämä aiheuttaa GET-metodilla lähetettyjen lomakkeiden parametrien näkymisen osoiterivillä. Henkilökohtaista tietoa lähetettäessä on siksi aina syytä käyttää POST-metodia. Parametrit on mahdollista piilottaa käyttämällä sivua IFRAME-elementin kautta.

Vieraskirjassa allekirjoitusparametri on BASE64-enkoodattua SVG-dataa ja yhden allekirjoituksen pituus on minimissään lähes 1 000 merkkiä pitkä. GET-metodissa on 2 048 merkin pituinen merkkirajoitus, mikä on Vieraskirjan allekirjoitustiedolle liian tiukka. Sovelluksessa täytyy tästä syystä ainakin osittain käyttää POST-metodia.

## 2.2 Apache-palvelinohjelmisto

Vuoden 2015 Helmikuussa 20 vuotta täyttänyt Apache on maailman suosituin web-palvelin. Apache http server on Apache Software Foundationin avoimen lähdekoodin projekti [7.] Projektin aloittivat ryhmä joka kutsui itseään Apache Groupiksi ja jonka tarkoitus oli jatkaa NCSA:n hylkäämää HTTPD-projektia.

Vieraskirjasovelluksessa käytetään versiota  tällä hetkellä versiota 2.4.10. Apachessa on integroitu tuki MySQL tietokannalle sekä komentosarjakielille kuten JavaScript, PHP, Python ja Ruby [7].

Apache asennettiin testipalvelimelle paketinhallinnan kautta suoraan verkosta. Lopulliselle palvelimelle sovellukset asennettiin paketinhallinnan kautta, mutta tietoturvasyistä verkkorepositoryjen sijasta käytettiin asennuslevykkeitä.

## 2.3 PHP

PHP eli Hypertext Preprosessor on palvelinpuolella suoritettava komentosarjakieli, joka polveutuu tuotteesta PHP/FI. Vuonna 1994 Rasmus Lerdorfin luoma versio oli yksinkertainen binääripaketti joka sisälsi C-kielellä ohjelmoidun CGI:n (Common Gateway Interface). Tämän ainoa tehtävä oli tarkkailla vierailijamääriä hänen kotisivullaan, jossa sijaitsi hänen työhakemuksensa. Tämän johdosta hän antoi paketille nimeksi Personal Home Page Tools. [8.]

Nykyaikaisen PHP-tulkin tehtäviin kuuluvat palvelimella sijaitsevien PHP-komentosarjatiedostojen suorittaminen ja tulkilta saadun ulostulon lähettäminen sivua kutsuneeseen selaimeen HTML-tyyppisenä.

PHP-tulkin suorittaman komentosarjatiedoston suoritusjärjestys voidaan jakaa kolmeen osaan. PHP-komentosarjatiedoston suorittaminen alkaa asiakkaan kutsuessa php-sivua, jolloin palvelimen php-tulkki prosessoi koodin. Prosessoinnin aikana tulkki tunnistaa mitkä osiot sivusta sisältävät PHP-kieltä. Tämän jälkeen tulkki suorittaa PHP-osiot sivusta, muuntaa tiedot suoritetut tiedot HTML-muotoiseksi, liittää tiedot alkuperäiseen sivuun ja lähettää sivun asiakkaan selaimeen. [9.]

Tämänkaltainen toiminnallisuus luo rajoituksia palvelinpuolen koodille, sillä palvelimen PHP-tulkki suorittaa koodin ainoastaan sivun latauksen yhteydessä ja siksi reaaliaikainen tiedonsiirto vain PHP:tä käyttäen ei ole mahdollista. Rajoitus parantaa tietoturvaa, sillä esimerkiksi JavaScriptillä ei ole mahdollisuutta manipuloida PHP-muuttujia tai funktioita. Tällä tavoin voidaan tiedonsiirto PHP:stä JavaScriptiin estettyä.

## 2.4 HTML

### 2.4.1 HTML-dokumentin rakenne

HTML on merkkauskieli jolla luodaan verkkodokumentteja. Dokumentit koostuvat tekstistä ja rakenteista, jotka määritellään erilaisin elementein. Ohjelmointiesimerkki 1 kuvaa HTML-sivun perinteistä rakennetta. Esimerkissä ensimmäisenä oleva <! DOCTYPE>-elementti ei kuulu HTML-elementteihin vaan on tyyppimäärittelyelementti. Tämän elementin tehtävänä on kertoa selaimelle, mitä HTML-versiota sivun luomisessa on käytetty sekä minkätyyppistä tietoa sivun odotetaan sisältävän. [10.]
Seuraava elementti on <html>-elementti, joka on dokumentin juurielementti. Tämä elementti löytyy jokaisesta HTML-dokumentista ja pitää sisällään kaikki muut HTML-elementit. Seuraava elementti on <head>-elementti, joka sisältää yleistä tietoa dokumentista mukaan lukien dokumentin otsikon sekä erilliset muotoilutiedostot. Kaikki loput tiedot tulevat dokumentin rungon eli <body>-elementin sisään.

```
<!DOCTYPE HTML PUBLIC ”-//W3C//DTD HTML 4.01//EN”
“http://www.w3.org/TR/html4/strict.dtd”>

<html>
          <head>
                    <title>esimerkkiotsikko</title>
          </head>
          <body>
                    <p>Hei Maailma!</p>
          </body>
</html>
```

Ohjelmointiesimerkki 1. Yksinkertaisen HTML-dokumentin rakenne.

Tyyppimäärittelyn käyttö on suositeltavaa, sillä selaimen ei tällöin tarvitse ”arvata” dokumentin tyyppiä ja dokumentti näkyy selaimessa suurella todennäköisyydellä

toivotulla tavalla [11]. HTML 4.01 -versio tarvitsee tyyppimäärityksen, koska dokumentin tyyppejä on useita erilaisia.

Ohjelmointiesimerkissä 2 käytetty tyyppimäärittely vastaa uusimman HTML5-version tyyppimäärittelyä, joka on hyvin yksinkertainen johtuen version yksinkertaistamispäätöksestä. Huomioitavaa on myös esimerkin 2 <head>-elementin poissaolo. HTML5-versiossa sen kirjoittaminen ei ole välttämätöntä, sillä tässä dokumenttityypissä <body>-elementin ulkopuolista osaa käsitellään oletuksena <head>-elementtinä. Tämä johtuu yksinkertaisesti siitä, että HTML-tiedoston rakenne sisältää vain nämä kaksi elementtiä, joten tällainen oletus vähentää kirjoitusmäärää.

```
<!DOCTYPE HTML>

<html>
                        <title>dokumentin otsikko</title>
            <body>
                        <p>tekstikappale</p>
            </body>
</html>
```

Ohjelmointiesimerkki 2. Yksinkertaisen HTML5-dokumentin rakenne.

Elementit määritellään avaus- ja sulkutageilla. Tagien väliin kirjoitetaan elementin visuaalinen osa. Elementin ominaisuudet määritellään avaustagiin ohjelmointiesimerkin 3 osoittamalla tavalla.

```
<body>
            <p id=”tunnus”>Hei Maailma!</p>
</body>
```

Ohjelmointiesimerkki 3. Elementin ominaisuuden määrittely.

HTML sisältää lähes kaikenlaisiin tarpeisiin sopivia ominaisuuksia, ja on suunnittelijan etu tuntea omaa käyttötarkoitustansa varten käytännöllisimmät ominaisuudet.

HTML-sivun toiminnallisuutta suunnitellessa on hyvä muistaa erilaisten HTML-elementtien oletustoiminnallisuudet. Elementit käyttäytyvät yksilöllisesti erilaisissa selaimissa, ja ulkoasun muotoilua voi yksinkertaistaa oikeiden elementtien valinnoilla.

### 2.4.2 HTML versiot

Uusin HTML-versio on HTML5, joka tukee video- ja audio-elementtejä, joiden avulla voidaan suoratoistaa palvelimen dataa selaimessa. Oleellista tämän projektin kannalta on sisääntuloelementeissä käytettävä ominaisuus ”required”, joka mahdollistaa rajoitusten luomisen sisääntuloihin ja jolla on rajapintayhteys käyttöliittymän visuaaliseen käyttäjäpalauteilmoitukseen.

Ohjelmointiesimerkki 4 havainnollistaa edellä mainitun ominaisuuden käyttötapaa

```
body>
            <form>
            <input value=”” name=”nimi” required=”required”/>
            </form>
</body>
```

Ohjelmointiesimerkki 4. Elementin ominaisuuden määrittely.

Hyödyllinen ominaisuus required-ominaisuuden parina on ”pattern”-ominaisuus, jolla voi rajoittaa sisääntulon laatua tarkemmin esimerkiksi säännöllistä lauseketta hyödyntämällä. Ohjelmointiesimerkki 5 kuvaa tilannetta, missä säännöllistä lauseketta käytetään lomake-sisääntulon rakenteen määrittämiseen pattern-ominaisuuden avulla.

```
<body>
            <form>
            <input value=”” name=”nimi” required=”required”
            pattern=”[(a-öA-Ö|a-öA-Ö)]”/>
            </form>
</body>
```

Ohjelmointiesimerkki 5. pattern-ominaisuuden määrittely.

## 2.5 CSS

CSS mahdollistaa HTML-sivuston sisällön erottamisen tyylien määrittelyistä. Ohjelmointiesimerkissä 6 esitetään CSS-muotoilun rakennetta. Edellä mainitussa esimerkissä muotoilulla määritetään kaikkien sivulla sijaitsevien <p>-elementtien tekstinväri punaiseksi.

```
***muotoilutiedosto.css


p{font-color:red;}
```

Ohjelmointiesimerkki 6. CSS-muotoilun rakenne.

Seuraavat kolme ohjelmointiesimerkkiä näyttää erilaiset tavat lisätä CSS-muotoilua HTML-sivulla. Ohjelmointiesimerkissä 7 esitetty tapa on liittää CSS-tiedosto HTML-sivulle. CSS-muotoilun voi sijoittaa kirjoittaa suoraan itse elementtiin esimerkin 8 kuvaamalla tavalla. Kolmas tapa on määritellä CSS-muotoilu suoraan HTML-elementissä ohjelmointiesimerkin 9 esitetyllä tavalla.

```
<html>
          <head>
                    <title>esimerkkiotsikko</title>
                    <script src=”muotoilutiedosto.css”></script>

          </head>
          <body>
                    <p>Hei Maailma!</p>
          </body>
</html>
```

Ohjelmointiesimerkki 7. CSS-muotoilun liittäminen tiedostona HTML-sivuun.

Ohjelmointiesimerkissä 8 käytetään tapaa kirjoittaa CSS-muotoilut tyyli-elementtiin.

```
<html>
          <head>
                    <title>esimerkkiotsikko</title>
                    <style>


                    p{ font-color:red;}


                    </style>


          </head>
```

```
            <body>
                        <p>Hei Maailma!</p>
            </body>
</html>
```

Ohjelmointiesimerkki 8. CSS-muotoilun kirjoitus <style>-elementtiin.

```
<html>
            <head>
                        <title>esimerkkiotsikko</title>
            </head>
            <body>
                        <p style=”font-color:red;”> Hei Maailma!</p>
            </body>
</html>
```

Ohjelmointiesimerkki 9. CSS-muotoilun määrittely kyseisen elementin sisällä.

CSS-muotoilua tukevat kaikki modernit selaimet, mutta selaimien käyttäytymisessä on eroja, jotka on otettava suunnitteluvaiheessa huomioon. Selainten käyttäytymiseroja selittävät erilaisuudet selainten CSS- sekä JavaScript-tulkeissa. Oletusmuotoilut selaimissa toimivat käytännössä kuin CSS-tiedostot, jotka ladattaisiin sivun lataamisen yhteydessä. Uusilla CSS-muotoiluilla voidaan ylikirjoittaa oletusmäärittelyjä, jos ulkoasu ei miellytä käyttäjää.

## 2.6 Tietokannat

Tietokanta on tehokas tapa varastoida kaikenlaista sähköistä tietoa tekstistä kuviin. Tiedon säilöminen kantoihin on suhteellisen yksinkertaista ja helppoa. Järkevästi muodostetuista kannoista tietojen siirto on nopeaa. Siksi tietokantoja käytetään useasti verkkosivujen yhteydessä tietojen varastoimista tai tietojen hakutarkoituksia varten.

### 2.6.1 Relaatiotietokantojen taustaa

Vuonna 1970 Tri E. F. Codd julkaisi artikkelin: ”A Relational Model Of Data For Large Shared Data Banks”, joka käsitteli teoreettista tiedonvarastoimismallia. Tämä artikkeli oli ensimmäinen hahmotelma tiedon varastoimisesta ja manipuloimisesta käyttäen tauluja. IBM käytti edellä mainittua artikkelia pohjatietona relaatiotietokannan suunnittelussa. Vuoteen 1982 mennessä IBM, Oracle sekä Relational Technology Inc. olivat julkaisseet omat kaupalliset relaatiotietokantansa. Kyselykielenä on toiminut toimi SQL, vuodesta 1986 lähtien. SQL on lyhenne sanoista Structured Query Language. SQL on IBM:n ja Oraclen kehittämä kieli. [12.]

### 2.6.2 MySQL-tietokanta

MySQL on maailman suosituin avoimen lähdekoodin relaatiotietokanta. Se perustuu SQL-kieleen, lukuun ottamatta muutamia poikkeuksia. Tällaisia poikkeuksia ovat muun muassa kyselyt ”SELECT INTO TABLE” ja ”UPDATE” jotka toimivat hieman eritavalla. Eroja on myös ulkoisissa avaimissa sekä CAST()-että REVOKE()-funktioissa. [13].

Vieraskirjasovelluksessa MySQL-asennettiin palvelimelle paketinhallinnan kautta. Uusimmat mysql-server- ja mysql-client paketit olivat asennushetkellä versionumeroltaan 5.1.

## 2.7 JavaScript

### 2.7.1 Historiaa

JavaScript on komentosarjakieli, jonka on kehittänyt Brendan Eich vuonna 1995 työskennellessään Netscapelle. Eich palkattiin kehittämään jotakin uutta komentosarjakieltä, jolla saisi verkkosivujen toimintoja automatisoitua tai verkkosivuja dynaamisemmaksi. JavaScriptin toinen tavoite oli parantaa Java-tuen käytettävyyttä niiden verkkosuunnittelijoiden ja ohjelmoijien keskuudessa, jotka eivät olleet perehtynyt Java-kieleen. [14;15.]

JavaScriptin rooli selaimen apukielenä on muuttunut alkuperäisestä versiostaan. Nykyään JavaScriptiä on mahdollista käyttää selaimen ulkopuolisten sovellusten luomiseen esimerkiksi Node.JS-ohjelmistokehystä hyödyntämällä.

Sun Microsystemsin kehittämään ECMAscriptiin pohjautuva JavaScript on saanut vaikutteita Java-kielestä, ja sen syntaksi perustuu löyhästi C-kieleen. ECMAscriptiin perustuu lisäksi myös ActionScript, joka on JavaScriptin jälkeen tunnetuin ECMAscript-sovellus.

Uusimmilla selainversioilla on tuki ECMAscript6:een asti. Vanhempien selainten kanssa on käytettävä vanhempia standardeja, mutta suositeltavaa olisi päivittää selainversio uudempaan, sillä vanhemmissa versioissa voi olla tietoturvan riskeeraavia aukkoja.

### 2.7.2 Käyttötarkoitus

JavaScriptillä on pääsy HTML-puuhun, jonka selain on rakentanut lähdekoodista sivun lataamisen yhteydessä. Tämän puun elementtejä voidaan manipuloida reaaliaikaisesti JavaScriptillä, jolloin sivun ulkoasu ja sisältö saadaan reagoimaan ohjelmistokehittäjän suunnittelemalla tavalla. Verkkosivun ulkoasua voi muokata myös CSS-muotoilua muuttamalla JavaScript-koodista käsin reagoimaan käyttäjän liikkeisiin.

JavaScript toimii hyvin verkkosovelluksissa, sillä sen avulla saa verkkosivuista interaktiivisia ilman jatkuvaa tiedonsiirtoa serverin ja asiakkaan välillä.

JavaScriptiä voi hyödyntää myös tiedonsiirrossa AJAX-metodien avulla. Jos sivu on suunniteltu tukemaan täysin AJAX-tyyppistä tiedonsiirtoa, on mahdollista saada sivu toimimaan saumattomasti ilman lataamisesta aiheutuvaa odotusta. AJAX on lyhenne sanoista asynchronous JavaScript and XML, eli asynkroninen JavaScript ja XML. Asynkronisuus tarkoittaa sitä, että tiedonsiirrossa ei ole vuoroja lähettäjällä ja vastaanottajille ja tietoa voidaan siirtää näiden kahden välillä molempiin suuntiin samanaikaisesti. Tämä on JavaScriptin tärkein ja hyödyllisin ominaisuus verkkosovellusten kannalta, sillä AJAX-metodeilla voidaan hakea tietoa palvelimelta samanaikaisesti sivun sisällönsuorittamisen kanssa. AJAX-metodeja voidaan käyttää kahdella tavalla, jotka esitellään seuraavissa ohjelmointiesimerkeissä.

Ohjelmointiesimerkissä 10 esitellään XMLHttpRequest-olion luonti ja POST-metodin käyttö AJAXia hyödyntämällä. Esimerkissä vastaanotettavan tiedon oletetaan olevan JSON-muotoista.

```
xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function ajaxkirjautuneet() {
          //jos serveriltä saadaan hyväksytty vastaus
if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
var data = JSON.parse(xmlhttp.responseText);
                    if(data.responseText != "null"){
// tämä rivi suoritetaan, jos vastaus ei ole ”null” eli
tyhjä arvo
}
                    }

xmlhttp.open("post","/funktiot.php?fname="+str,true);


xmlhttp.send();

}
```

Ohjelmointiesimerkki 10. AJAX POST -metodi XMLHttpRequest-oliota hyödyntämällä.

Ohjelmointiesimerkissä 11 luodaan POST-pyyntö jQuery-kirjaston AJAX-funktiota käyttämällä.

```
//alustetaan julkinen muuttuja ”request”


 var request;
       if (request){

        request.abort();
        }
       //sijoitetaan JavaScript muuttujan osoittama olio
jQuery-muuttujaan
    var $form = $("#"+formiid);
    // valitaan kaikki sisääntulot

        var $inputs = $form.find("input, select, button,
textarea");
```

```
// muunnetaan lomaketiedot merkkijonoksi tiedonsiirtoa
varten
var serializedData = $form.serialize();

     // sisääntulot poistetaan käytöstä siihen asti kunnes
saadaan vastaus pyynnölle

    $inputs.prop("disabled", true);

    // lähetetään pyyntö sivulle funktiot.php
   request = $.ajax({
        url: "/funktiot.php",
        type: "post",
        data: serializedData,
        dataType: 'json',
    });

    request.done(function (responseText, textStatus, jqXHR){
        // tämä rivi suoritetaan pyynnön onnistuessa
}
//Funktion parametri responseText on olio joka sisältää
pyynnön vastauksen.
```

Ohjelmointiesimerkki 11. jQuery-kirjastoa hyödyntämällä luotu AJAX-post metodi.

### 2.7.3 Yleisimmät käyttökohteet

JavaScript soveltuu verkkosivun toiminnallisuuden sulavoittamiseen, visuaalisen sisällön manipuloimiseen, lomakkeiden käsittelyyn ja käyttäjän aktiivisuuden seuraamiseen [16]. Nykyään monen jopa sivuston toiminnallisuus perustuu täysin JavaScriptiin. Tästä voi seurata ongelmia, sillä jos selaimen JavaScriptin suoritus on estetty, sivut eivät toimi lainkaan.

Useammalle eri alustalle suunniteltavat verkkosovellukset ovat erinomainen kohde JavaScriptin käyttöön, sillä JavaScript suoritetaan selaimessa ja nykyiset modernit selaimet ovat hyvin standardoituja. Tällä tavalla sama sovellus on mahdollista saada toimimaan monella erilaisella alustalla ilman radikaaleja muutoksia koodissa. Verkkosivuja suunnitellessa on huomioitava, että jos käyttäjällä ei ole JavaScript käytössä selaimessa, ei verkkosivun JavaScriptiä suoriteta.

Älypuhelimien yleistyessä JavaScriptin suosio on kasvanut, johtuen käyttötarkoituksesta älypuhelinsovellusten suunnittelussa sekä mobiiliverkkosivujen ominaisuuksien luomisessa [16].

### 2.7.4 Tietoturva

Verkkosivujen kehittämisessä on syytä käyttää palvelinpuolen komentosarjakielen ominaisuuksia tietoturvan parantamiseksi. JavaScriptin hyödyntäminen tietoturva-asioissa ei ole suositeltavaa, sillä verkkosivujen lähdekoodin on lähes aina helposti nähtävissä. Esimerkiksi selaimen tai minkä tahansa serverille pyynnön lähettävän sovelluksen avulla JavaScriptillä toteutettu tiedon salaamiseen käytetty menetelmä on helppoa paljastaa. Kaikki JavaScriptillä toteutetut turvatoimet ovat siis kaikkien osaavien käyttäjien näkyvissä. Lomakkeiden sisääntulojen viimeinen tarkistus onkin syytä suorittaa huolellisesti erityisesti serveripuolen verkko-ohjelmointikielellä, sillä tietoturvallisesti suunnitellun palvelimen komentosarjatiedostoja ei ole mahdollista muokata palvelimen ulkopuolelta.

Yleisimpiin verkon tietoturvaongelmiin kuuluvat Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF) sekä Man In The Middle (MITM) -hyökkäykset. XSS-hyökkäyksessä hyökkääjä manipuloi serveriltä pyydetyn verkkosivun lähdekoodia, jonka jälkeen lähettää sen pahaa aavistamattoman käyttäjän selaimeen [16;17].

Kahdella yleisesti tunnetulla menetelmällä voidaan rajoittaa JavaScriptin aiheuttamia turvallisuusongelmia. Menetelmät ovat nimeltään sandboxing sekä same origin policy (saman lähteen politiikka). Sandboxingilla tarkoitetaan toiminnallisten osuuksien eristämistä toisistaan ja sallitaan niille pääsy vain niiden tarvitsemiin resursseihin. Same origin policy estää AJAX-pyynnöt verkkosivuston ulkopuolelta.

Same Origin Policy -suojaus on kuitenkin mahdollista kiertää esimerkiksi iframe-elementtien hyväksikäytöllä.

Siirrettävää tietoa voi salata salausavaimella ja käyttämällä istuntoja, jolloin tiedon hankkiminen ja hyökkäykset ovat huomattavasti vaikeampia toteuttaa.

### 2.7.5 Rakenne

JavaScript-koodi sijoitetaan HTML-tiedostossa script-elementtien (<script></script>) sisään tai linkitetään tiedostona HTML-sivulle (<script src=”javascript.js”></script>). Liitteenä oleva koodi voi sisältää esimerkiksi kokonaisen JavaScript-kirjaston.

Yleisimpiin JavaScript-kirjastoihin ja kehikkoihin kuuluvat esimerkiksi Bootstrap, Node.js, jQuery, HTML5 Boilerplate sekä Backbone.js [18].

### 2.7.6 Kirjastot

jQuery on JavaScript-kielellä kirjoitettu kirjasto, joka on luotu helpottamaan ohjelmointityötä toistuvien operaatioiden luomisessa. Kirjaston vahvuudet piilevät menetelmien rakenteessa, joka on kirjoitettu tavalla, joka mahdollistaa samanlaisen JavaScript-rakenteen suorittamisen vanhemmissakin selaimissa. jQueryn kaltaisten lisäkirjastojen ja kehysten käyttö voi myös parhaassa tapauksessa lyhentää ja yksinkertaistaa koodia sekä helpottaa koodin kirjoittamista.

### 2.7.7 Vieraskirjasovelluksen kirjastot

Vieraskirja-sovelluksessa toimiva allekirjoitusominaisuus on luotu käyttäen jSignature-kirjastoa, joka on Willow Systemsin luoma jQuery-kirjastoon perustuva lisäosa. Kirjaston käyttöpäätöksen jälkeen jQueryn käyttötarkoitus laajentui muutamaan funktioon.

Tässä projektissa kirjastojen käyttö ei olennaisesti hidasta sovellusta, sillä sivu ladataan vain kerran, jonka jälkeen sivun sisältöä päivitetään ilman koko sivun uudelleenlatausta. Selain osaa myös tallettaa joitakin lähdetiedostoja välimuistiin, jolloin tiedostoja ei tarvitse ladata täysin uudestaan jokaisella sivun päivityskerralla.

## 2.8 Muistivuodot

Muistivuodoilla tarkoitetaan tilanteita, missä muuttuja tai joku muu osa lähdekoodista jää varaamaan selaimen muistia. Esimerkkinä tilanne jossa HTML-elementin viite on sisällytetty johonkin aktiiviseen muuttujaan, jolloin selaimen roskienkeräin eli *Garbage Collector* ei voi tuhota muuttujaa ja muuttuja jää varaamaan selaimen muistia. Viitteiden jääminen muuttujiin voi johtua esimerkiksi siitä, että muuttujan näkyvyysalue on määritelty liian väljäksi, jolloin muuttuja on aktiivinen läpi suoritettavan koodin ja muuttujassa on määritetty suora viite HTML-elementtiin.

Sivuston toiminta kärsii muistivuodoista, sillä muistivuotojen aiheuttama selaimen muistinvaraus hidastaa järjestelmän toimintaa käynnistämällä muistinvapautusprosesseja. Tämä voi heikentää huomattavasti päätelaitteen suorituskykyä ja aiheuttaa negatiivisen käyttökokemuksen sivustolla.

Muistivuodon ongelmat ilmentyvät toiminnallisilla verkkosivuilla, joissa käyttäjät viipyvät ja suorittavat toimenpiteitä. Tämä johtuu siitä, että kun verkkosivun käyttäjä käyttää toimintoa, joka aiheuttaa muistivuodon, muistia varataan selaimen käyttöön lisää kerta kerran jälkeen. Selaimen varaama muisti vapautuu vasta käyttäjän siirtyessä toiselle sivustolle. Tästä syystä käyttöliittymiksi suunnitellut verkkosivut sekä muut toiminnalliset verkkosivut ovat haavoittuvimpia muistivuodoille verrattuna staattista tietoa sisältäviin verkkosivuihin. Tämä johtuu toiminnallisten sivujen reaktiivisuuden kääntöpuolesta. Käyttäjää varten sivun on reagoitava sivulla mahdollisesti tehtyihin toimenpiteisiin, jolloin sivun sisältöä on muutettava. Käyttäjä myös viipyy toiminnallisilla sivuilla useasti kauemmin ja lataa sivun useamman kerran, jolloin mahdolliset muistivuodot alkavat vaikuttamaan kerrannaisvaikutusten takia.

Reaktiivisten ja toiminnallisten sivujen JavaScript-koodi on hyvin monimutkaista, mikä hankaloittaa muistivuotojen paikantamista.

## 2.9 JSON

JSON muodostuu sanoista JavaScript Object Notation ja on tiedonsiirtoon kehitetty, kevyt tekstimuotoinen tiedostoformaatti. JSON on riippumaton JavaScriptistä, vaikka rakenne perustuukin nimensä mukaisesti JavaScriptiin. Oikeellisen JSON-tyyppisen tiedon luominen on helppoa esimerkiksi PHP:n json_encode()-funktion avulla. Paras tapa luoda JSON-rakenne on käyttää siirrettävään taulukkomuuttujaan edellämainittua enkoodausfunktiota. JSON-rakenteesta on helppoa palauttaa arvo, sillä se perustuu arvo-pari tyyppisiin ja lista-tyyppisiin rakenteisiin. Nämä rakenteet ovat samanlaisia kuin yleisimpien ohjelmointikielien olio- ja taulukkorakenteet. [19.] JSON-rakenteet esitellään ohjelmointiesimerkissä 12.

```
arvopari
{nimi:arvo}
```

```
taulu/lista
[arvo,arvo,arvo]

yhdistelmä
{nimi:[arvo,arvo,arvo]}
```

Ohjelmointiesimerkki 12. JSON-rakenteiden esittelytavat.

# 3 Vieraskirjasovellus

Vieraskirjasovellus on selainpohjainen sovellus vierailijoiden tietojen kirjaukseen. Lähtökohtana sovelluksen kehittämiselle oli sen tarpeellisuus ja vanhan järjestelmän parantaminen. Sähköisen vieraskirjasovelluksen tarpeellisuus perustuu vanhan paperisen lomakejärjestelmän ominaisuuksien parantamiseen ja ongelmakohtien poistamiseen.

## 3.1 Vanhan kirjausmenetelmän ongelmat

Paperisessa lomakejärjestelmässä on useita ongelmakohtia, jotka vaikuttavat negatiivisesti järjestelmän tehokkuuteen. Sekä tietojen arkistointi paperiarkistoon että tiedonhaku paperiarkistosta on epätehokasta ja työlästä. Paperisten arkistojen säilytys aiheuttaa hukkatilaa ja tuo lisäkustannuksia, sillä paperin säilytystilojen on paperin säilymisen kannalta oltava hallitut. Tietosuojan säilyminen yksittäisten henkilöiden tietojenraportoinnin kohdalla on mahdotonta, jos useamman henkilön tiedot ovat samassa paperissa. Jos taas henkilöiden kirjaustiedot ovat erillisillä papereilla, voivat paperikustannukset volyymistä riippuen kasvaa suuriksi.

Paperilomakkeiden tarkistamista hankaloittaa esimerkiksi lomakkeentäyttäjän suttuinen tai huono käsiala. Lomakkeen täyttäminen käsin on keskimäärin hitaampaa kuin hyvin tuotetun sähköisen sovelluksen käyttö.

Edellä mainittujen ongelmien poistaminen vaatii paperisen järjestelmän muuntamisen sähköiseen järjestelmään.

# 4 Suunnittelu

## 4.1 Vieraskirjasovelluksen suunnittelu

Vieraskirjasovelluksen suunnittelun alkuvaiheessa määriteltiin ensimmäiseksi sovellukselle asetetut vaatimukset ja käytössä olevat resurssit. Työn ohjaaja teki ehdotuksia lisävaatimuksille ja täydensi aiempia vaatimuksia projektin edetessä.

### 4.1.1 Vaatimukset

Sovelluksen alustavat vaatimukset toiminnallisuudelle periytyivät vanhan paperisen kirjausjärjestelmän vaatimuksista. Vaatimuksia tuli lisää työn edetessä työn ohjaajilta sekä kollegoilta.

Sovelluksen toimintaympäristön oli määrä olla mahdollisimman tietoturvallinen. Sovelluksen tuli toimia kosketusnäyttöpäätelaitteella. Kirjautumistiedot oli varastoitava tietokantaan. Ylläpitokäyttäjän profiili, jolla oli mahdollista tulostaa kirjautumistiedoista raportti PDF- tai CSV-muotoisena. Käyttöliittymän tuli olla selkeä ja helppokäyttöinen. Lomakkeen tekstikenttien täytyi vastata paperisen lomakkeen kenttiä.

Sovelluksen rajatun käyttöympäristön huomioon ottaen oli tietoturvan kannalta paras ratkaisu rajata verkkoyhteys sovelluksen käyttöympäristön alueelle ja ilman ulkoista verkkoyhteyttä.

Palvelimen rakenteessa päädyttiin LAMP-kokoelmaan, joka helpotti myös ohjelmointikielen valitsemista työn suunnittelussa. Lyhenne LAMP muodostuu sanoista Linux, Apache, MySQL ja PHP [20.] Nämä ohjelmat olivat asennettuna eräällä palvelimella, jota suunniteltiin käytettäväksi sovelluksen testaamiseen.

Projektissa käytettiin tietokantana MySQL-relaatiotietokantaa, joka oli asennettuna valmiina testipalvelimelle. Tietokantayhteys luotiin PHP-komentosarjakielellä.

Yksinkertaisin ratkaisu oli tehdä sovellus verkko-ohjelmointikielillä, koska verkko-ohjelmointikielillä tiedonsiirtotekniikat ovat helppoja toteuttaa, dokumentointi on kattavaa ja sovellus on siirrettävissä erilaisten alustojen välillä lähes vapaasti.

### 4.1.2 Käyttöliittymä ja ominaisuudet

Käyttöliittymän rakenne ja toiminnallisuudet luotiin alustavasti HTML-rakenteilla ja muotoiltiin CSS:llä. Tiedonsiirtoon käytetään HTTP:n GET- ja POST-metodeita, joilla tietoa haetaan kahdelta erilliseltä sivulta. Sivut sijaitsevat samassa tiedostossa käyttöliittymän kanssa ja ovat PHP-komentosarjatiedostoja, jotka hakevat MySQL-tietokannasta tietoja ja tulostavat ne JSON-muotoisena sivulle. Allekirjoitus toteutetaan JavaScript-kirjastolla nimeltä-JSignature, joka on toiminnallisuuksiltaan sopiva käyttötarkoitukseen. Huono puoli JSignaturessa on vaatimus JQuery-kirjastolle, josta ei ollut muuta hyötyä käyttöliittymässä, eikä sitä alustavasti tarvittu muiden ominaisuuksien luontiin. Kun JQuery-kirjaston välttämättömyys ilmeni, koodissa pyrittiin käyttämään mahdollisimman paljon kyseistä kirjastoa hyödyn maksimoimiseksi.

Käyttöliittymän sisällönmuokkaus lopullisessa versiossa tapahtuu AJAX-metodeilla, joka poistaa JQueryn huonot puolet. Kirjastoja tarvitaan vain sivun latauksen yhteydessä, jonka jälkeen kirjastoja ei tarvitse ladata uudestaan ennen sivun päivitystä. Lomaketietojen lähetys käyttöliittymästä palvelimelle tapahtuu myös AJAX-metodeilla, jolloin sivusta saadaan ilman turhia sivun päivityksiä, sulavasti - ja saumattomasti toimiva.

Tulostusominaisuus toteutettiin sovellukseen WKHTMLTOPDF-nimisellä komentorivityökalulla. Tulostusfunktio kutsuu PHP-komentosarjatiedostoa, jossa kutsutaan PHP:n exec()-funktiolla edellä mainittua komentorivityökalua, joka muuttaa kohteena olevan HTML-sivun PDF-muotoon määriteltyjen CSS-muotoilujen perusteella. Tämän jälkeen PDF-tiedosto lähetetään palvelimelta käyttäjän selaimeen.

Tietoturvaominaisuuksista ei tarvinnut huolehtia, koska sovellusta käytetään suljetussa langallisessa lähiverkossa. Tämä tila on täysin eristetty kaikilta ulkoisilta verkkoyhteyksiltä mukaan lukien myös langattomat verkkoyhteydet. Käyttöliittymää käytetään koneessa Google Chrome -selaimella, johon on asennettu Kiosk-sovellus. Jos verkkosivun käynnistää Kiosk-sovelluksen kautta, käyttäjällä ei ole käytössä työkalu- eikä osoiteriviä, joten selaimen sulkeminen on mahdotonta ilman näppäimistöä. Sovelluksen oli tarkoitus tästä syystä toimia alustavasti ilman näppäimistöä, jotta käyttäjällä ei olisi mahdollisuuksia sulkea selainta. Näppäimistö otettiin myöhemmin käyttöön, sillä virtuaalisen näppäimistön käyttö oli hidasta ja kömpelöä.

Tulevassa versiossa päätelaitteena sovellukselle käytetään tabletti-tietokonetta jolloin näppäimistöä ei tarvita.

### 4.1.3 Käyttäjän kirjautumisen toimintakuvaus

Aluksi käyttäjä täyttää kaikki tekstikentät (Liite 1) ja valitsee korttinsa numeron vetovalikosta (Liite 2) tai vaihtoehtoisesti valitsee nimensä sekä muut tiedot pikakirjausvalikosta (Liitteet 7 ja 8) ja painaa ”Kirjaudu”-painikkeesta. Jos kaikki tekstikentät täytetään oikein, ilmestyy allekirjoituskenttä ruudulle (Liite 3). Muussa tapauksessa käyttäjäpalaute ilmestyy virheellisesti täytetyn kentän alapuolelle (Liite 9). Käyttäjä kirjoittaa allekirjoituksen ruudulle avautuneeseen allekirjoituskenttään ja painaa Ok-painiketta. Ruudulle ilmestyy teksti ”sisäänkirjautuminen onnistui!” (Liite 4). Käyttäjän nimi, kirjautumisaika ja uloskirjautumispainike ilmestyvät käyttöliittymän ”kirjautuneet henkilöt” -osioon (Liite 5).

Uloskirjautuessa käyttäjä painaa ”Kirjaudu ulos” -painiketta, jonka jälkeen ruudulle ilmestyy teksti ”uloskirjautuminen onnistui!” ja käyttäjän tiedot poistuvat käyttöliittymästä (Liite 6).

## 4.2 Työssä ilmenneet ongelmat

Työssä ilmeni muutamia ongelmia, joiden ratkaiseminen vei paljon aikaa. Eniten ongelmia oli WKHTMLTOPDF-tulostustyökaluun liittyen.

WKHTMLTOPDF-työkalua oli vaikea saada toimimaan palvelimella käyttöoikeusongelmien vuoksi, koska näiden ongelmien perimmäistä syytä oli hyvin vaikea jäljittää. Tämä johtui siitä, että työkalua kutsuttiin palvelimen kautta, jolloin työkalun käyttäjä oli palvelin eikä selaimen käyttäjä. Tiedosto, joka WKHTMLTOPDF-työkalulla luotiin, oli palvelimen omistama, eikä työkalun virheilmoituksia ollut helppo saada tämän vuoksi näkymään muille kuin sovellusta käyttävälle profiilille.

Ongelma korjaantui muutettaessa tulostuskohdekansion alitiedostojen oletusoikeuksia sekä muuttamalla palvelintiedostojen käyttöoikeudet palvelimen alaisiksi.

Tietokantoihin liittyvä ongelma tuli ilmi sähkökatkoksen aikana. Tietokannan erään taulun sisäinen laskuri nollaantui tietokantaprosessin pysäyttämisen yhteydessä. Tämän korjaaminen onnistui pienen PHP-funktion avulla, joka palautti taulun sisäisen laskurin ennen tauluun kirjoittamista. Kustannustehokkaampi korjaustapa olisi vaatinut shell-skriptin sisällyttämisen palvelimen käynnistyskansioon, joka suoritettaessa tekee samat toimenpiteet kuin edellä mainittu PHP-funktio.

### 4.3 Jatkokehitys

Sovelluksen jatkokehitystä varten on suunnitteilla sovelluksen toimintaympäristön laajennus, jolloin sen tarkoitus on kattaa Riihimäen varuskunta-alue kokonaisuudessaan.

Lopullisen sovellusversion toiminta käynnistyy, kun vierailijaksi pyrkivä henkilö täyttää verkossa lomakkeen, jonka käsittely tapahtuu sähköisesti. Hyväksytyn turvallisuusselvityksen jälkeen vierailijan isännäksi valikoitunut henkilö hyväksyy vierailevan henkilön. Isännän hyväksyttyä vierailevan henkilön saa tämä sähköpostiinsa koodin. Tämän koodin hän syöttää kirjautumissovellukseen saapuessaan vierailulle. Koodin syötettyään hän allekirjoittaa vaitiolovelvollisuuslomakkeen, jonka jälkeen järjestelmä lähettää vierailijan isännälle tekstiviestin vieraan saapumisesta. Järjestelmä mahdollisesti myös tulostaa värikoodatulla tunnisteella varustetun vierailijakortin, josta käy ilmi kulkuoikeuden laajuus.

## 5 Yhteenveto ja päätelmät

Projektin tuloksena syntyi toimiva prototyyppi kirjautumissovelluksesta sekä kirjaututumishistorian hallintatyökaluista. Prototyyppi on aktiivisessa käytössä ja antaa hyvät perusteet jatkokehitykselle.

Jatkokehitysmahdollisuuksia on järjestelmässä paljon, joista suurin osa liittyy vieralijakäynnin kokonaisprosessiin ja sen suoraviivaistamiseen. Esimerkiksi vierailupyyntölomakkeiden täyttö voisi tapahtua  Millogin virallisen sivuston alaisuudessa, josta tiedot ohjattaisiin asianmukaisille henkilöille.

Verkkosovelluksen suunnittelussa on otettava huomioon tulevaisuuden vaatimukset, luomalla mahdollisimman monipuolinen ja laajennuskelpoinen sovelluspohja. Jos sovelluksen kautta manipuloidun tiedon volyymi kasvaa huomattavasti tulevaisuudessa, on sovelluksen oltava mahdollisimman tehokkaasti skaalautuva.

Tietokantakyselyiden on oltava mahdollisimman tehokkaita, sillä jo muutaman kymmenesosasekunnin ylimääräiset viiveet voivat vaikuttaa negatiivisesti aikakriittisiin toimintoihin, jotka tarvitsevat tietokantatietoja. Kyselyitä voidaan nopeuttaa indeksoimalla tietokannan tauluja sekä ottamalla käyttöön tietokantojen välimuistiasetuksia.

## Lähteet


1 CIM/NetSiteStory. 2008.Sähköinen asiontipalvelu. Verkkodokumentti. Kansallisarkisto <www.narc.fi/asiointikaavio/> Sivu päivitetty 08.02.2005.Luettu 25.10.2015.

2 Jeskanen-Sundström, Heli. 2002. Julkinen palvelu sähköistyy vauhdilla. Verkkodokumentti. Tilastokeskus. <www.stat.fi/tup/tietoaika/tilaajat/ta_11_02_paak.html> Luettu 25.10.2015.

3 Beal,Vangie. Web Browser. Verkkodokumentti. Webopedia. <www.webopedia.com/TERM/B/browser.html> Luettu 26.10.2015.

4 Garsiel, Tali: How browsers work: Behind the scenes of modern web browsers. Verkkodokumentti. <www.taligarsiel.com/Projects/howbrowserswork1.htm> Luettu 2.10.2015.

5 HTTP Tutorial. Verkkodokumentti. <www.tutorialspoint.com/http/> Luettu 26.10.2015.

6 Deering, Sam 2011: jQuery AJAX Differences Between GET vs POST. Verkkodokumentti. <www.sitepoint/key-diffeerences-post/> Luettu 2.10.2015.

7 Luostarinen, Joni 2013: Asiakashallinta-sovellus Elysium Solutions Oy:lle. Opinnäytetyö. Savonia-ammattikorkeakoulu.

8 History of PHP. Verkkodokumentti. PHP. <php.net/manual/en/history.php.php> Luettu 10.11.2015.

9 Kujala, Pauli 2000: Dynaamisten WWW-sivujen tuottaminen PHP-kielen avulla. LuK-tutkielma. Jyväskylän yliopisto.Tietotekniikan laitos. <www.mit.jyu.fi/opetus/opinnayte/LuK/PHP/> Luettu 10.11.2015.

10 HTML <!DOCTYPE> Declaration. Verkkodokumentti. W3Schools. <w3schools.com/tags/tag_doctype.asp> Luettu 20.10.2015.

11 Thereaux, Olivier. 2002.Tips for Webmasters. Verkkodokumentti. <www.w3.org/QA/Tips/Doctype> Päivitetty 20.9.2013. Luettu 15.10.2015.

12 Relational Database Basics. Verkkodokumentti. Webucator. <www.webucator.com/tutorial/learn-sql/relational-database-basics.cfm> Luettu 2.10.2015.

13 MYSQL differences from standard SQL. Verkkodokumentti. <dev.mysql.com/doc/refman/5.7/en/differences-from-ansi.html> Luettu 16.10.2015.

14 A Short History of JavaScript: 2012. Verkkodokumentti. W3C.

<https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript>
Luettu 26.10.2015.

15 Champeon, Steve. 2001. JavaScript: How Did We Get Here? Verkkodokumentti. O'Reilly Network:Web DevCenter. <http://archive.oreilly.com/pub/a/javascript/2001/04/06/js_history.html> Luettu 30.10.2015.

16 Fergal, Glynn. JavaScript Security. Verkkodokumentti. Veracode. <www.veracode.com/security/javascript-security> Luettu 17.10.2015.

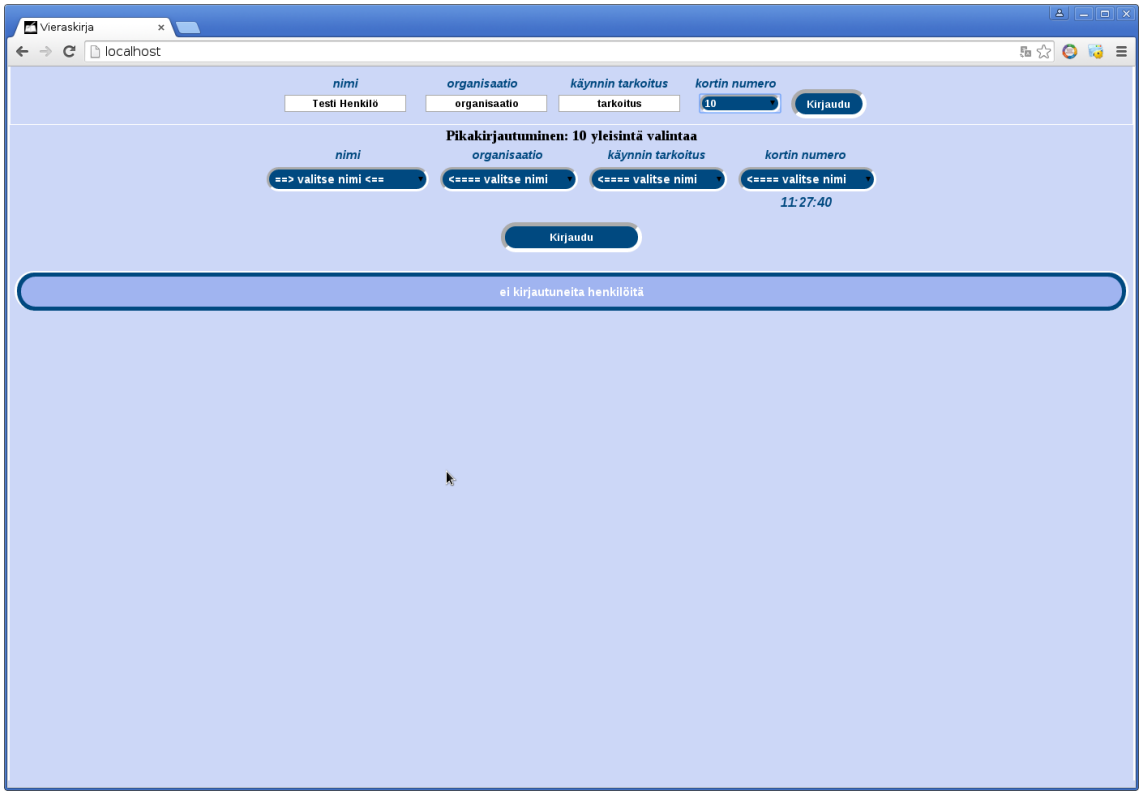
17 Cross-site Scripting (XSS). 2015. Verkkodokumentti. OWASP. <www.owasp.prg/index.php/Cross-site_Scripting_(XSS)> Luettu 24.11.2015.

18 Mysore, Aniruddha 2015: 10 Best JavaScript Frameworks and Libraries of 2015. Verkkodokumentti. Beebom. <www.beebom.com/2015/04/best-javascript-frameworks-and-libraries> Luettu 20.11.2015.

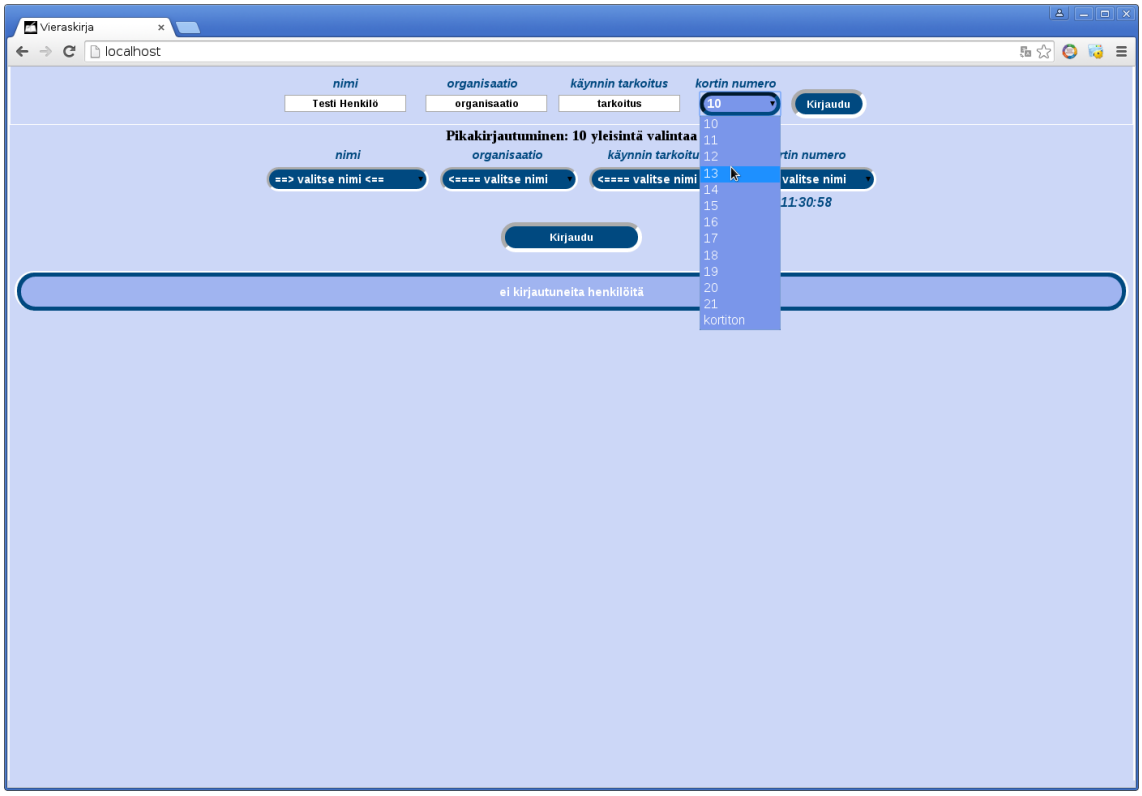
19 Introducing JSON. Verkkodokumentti. <www.json.org> Luettu 30.10.2015.

20 LAMP. Verkkodokumentti. Webopedia. <www.webopedia.com/TERM/L/LAMP.html> Luettu 2.10.2015.

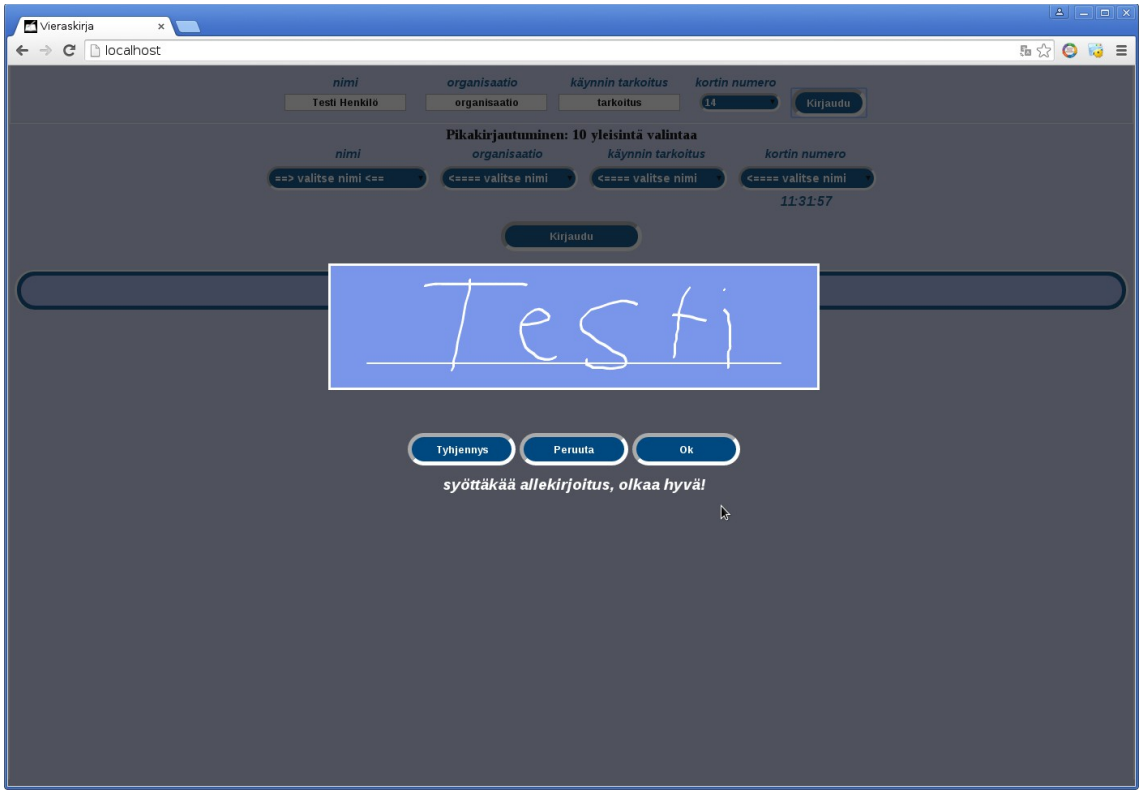
## Käyttöliittymä ilman kirjautuneita henkilöitä



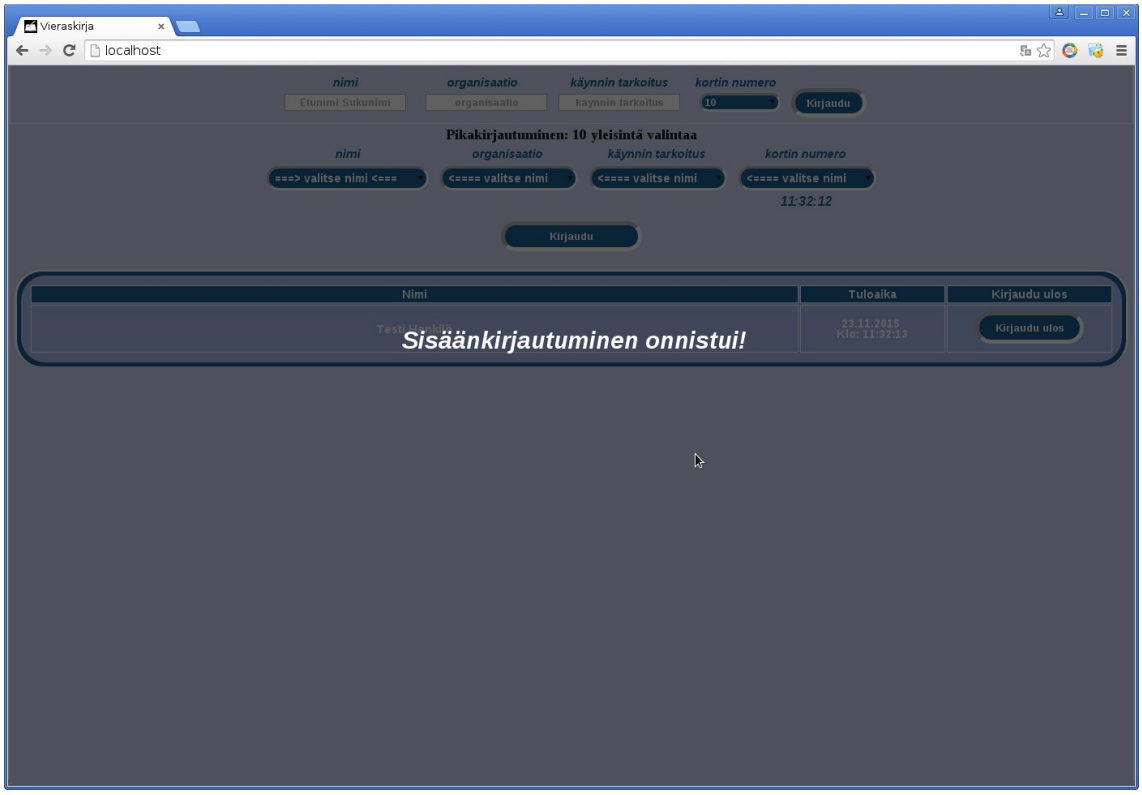
Kortinnumeron valitseminen



Allekirjoituskentän täyttö

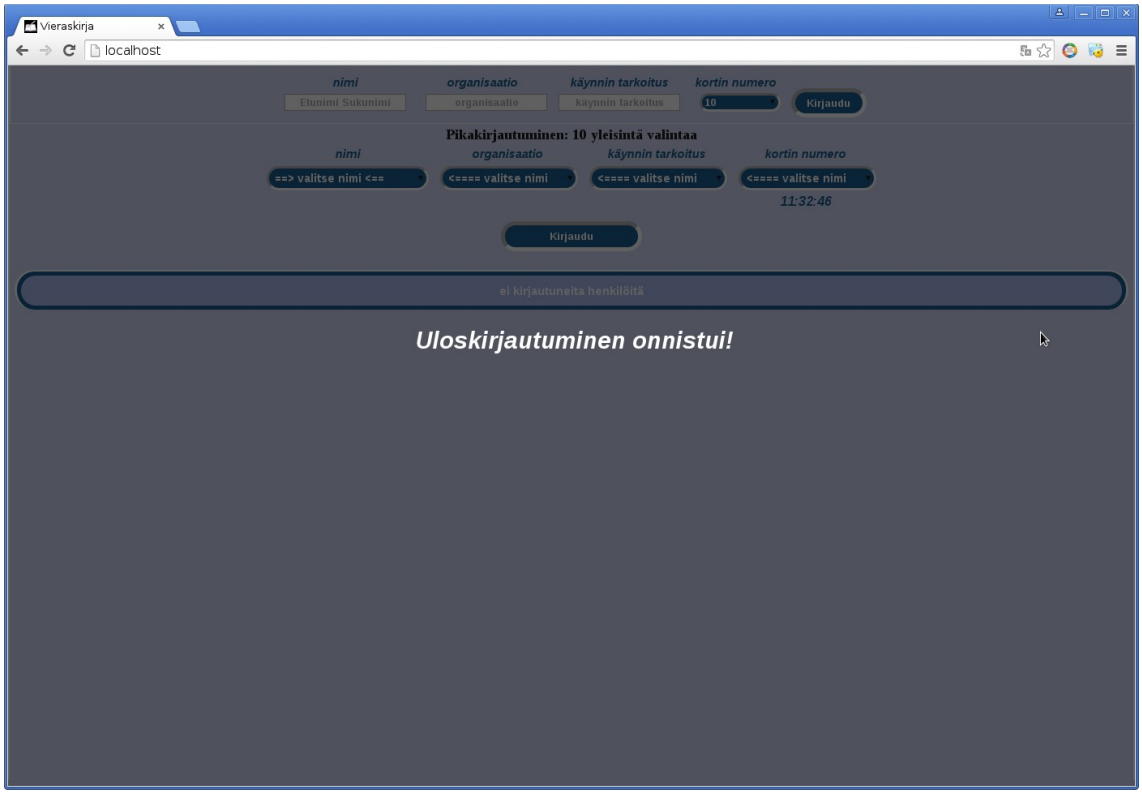


## Onnistunut sisäänkirjautuminen



## Käyttöliittymässä kirjautunut henkilö

## Onnistunut uloskirjautuminen

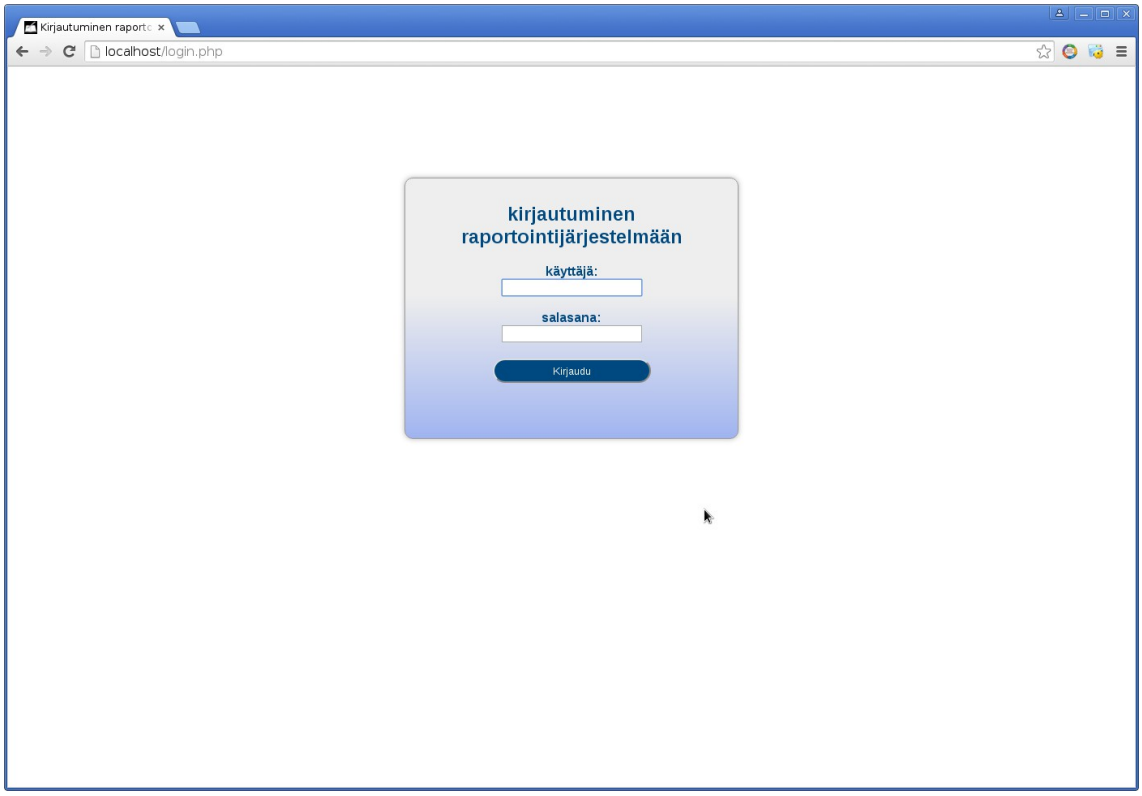


## Nimen valitseminen pikakirjautumisvalikosta

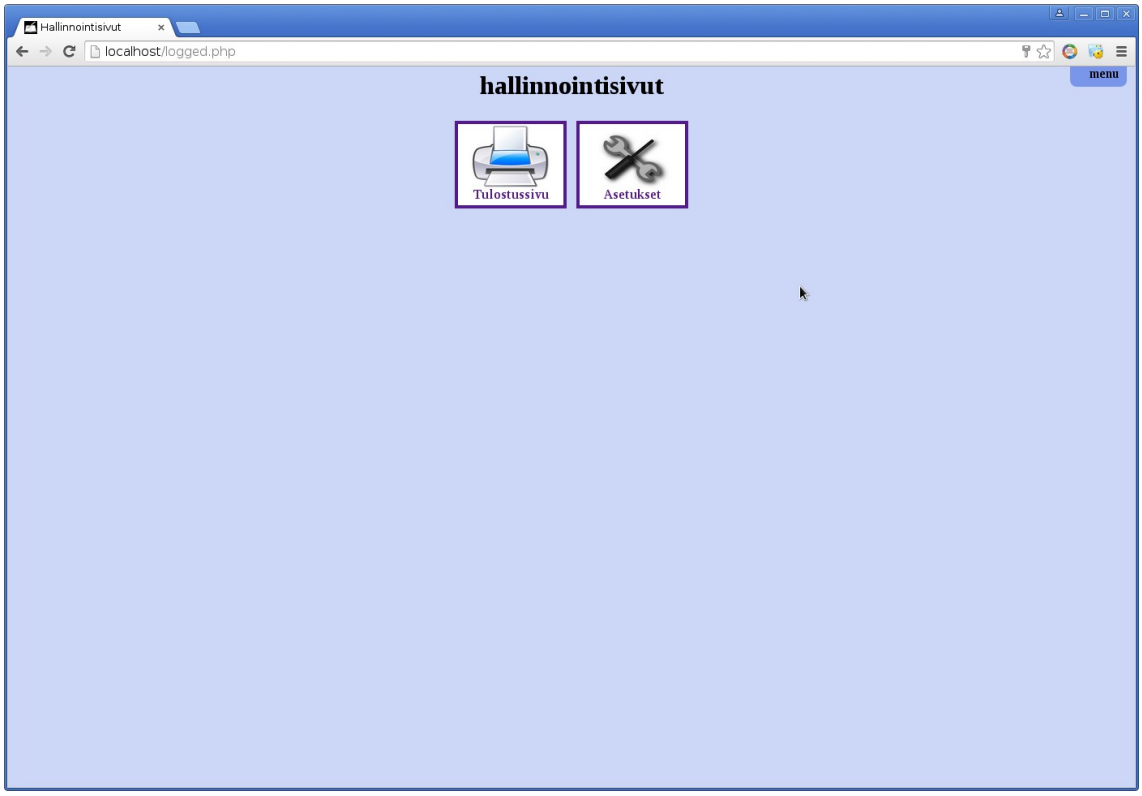
## Pikakirjautumislomakkeen automatiosoitu täyttö

**Virheellisen käyttäjäsyötteen laukaisema käyttäjäpalauteilmoitus**

## Kirjautuminen hallinnointisivulle

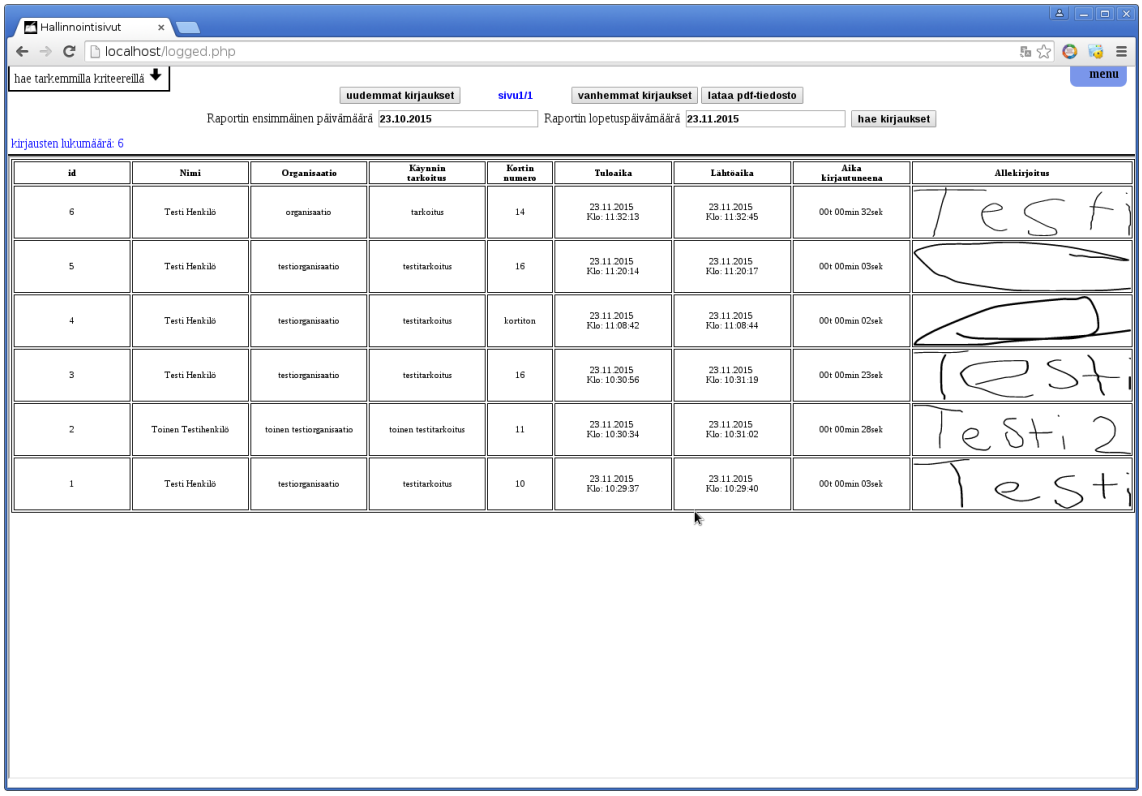


**Hallinnointisivujen päävalikko**



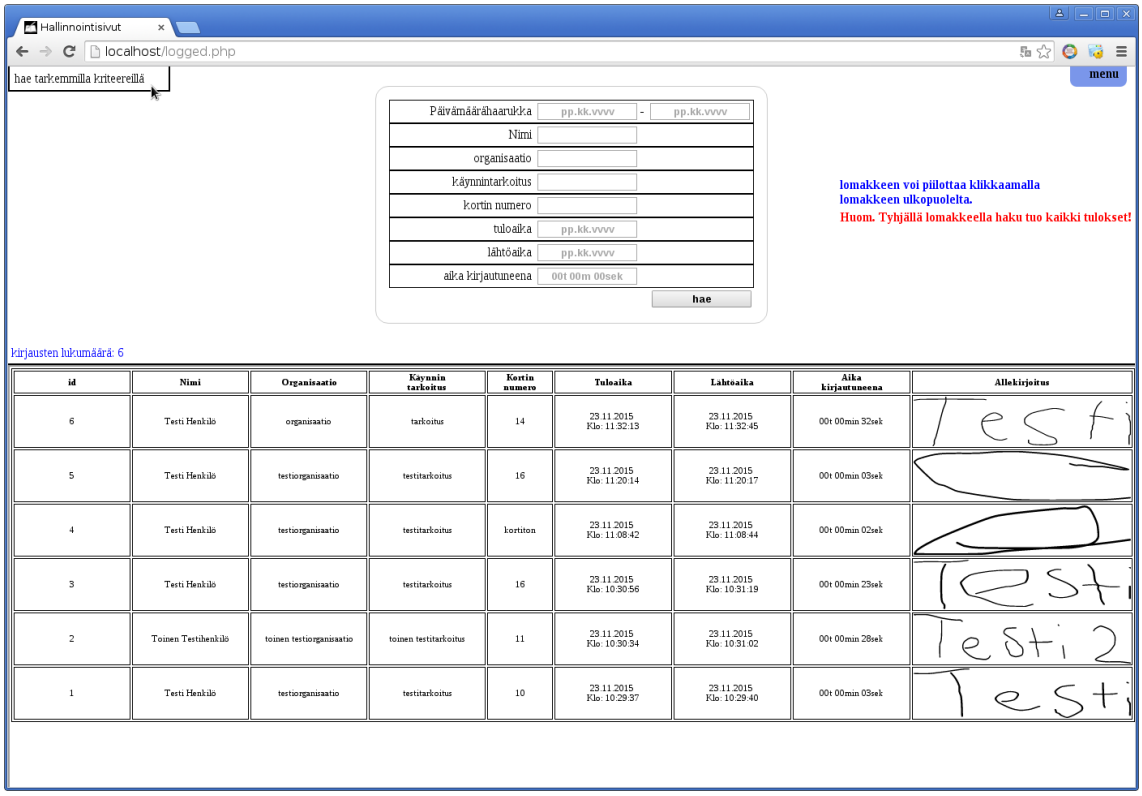
## Hallinnointisivujen navigointipalkki

## Kirjautuneiden henkilöiden lokitiedot



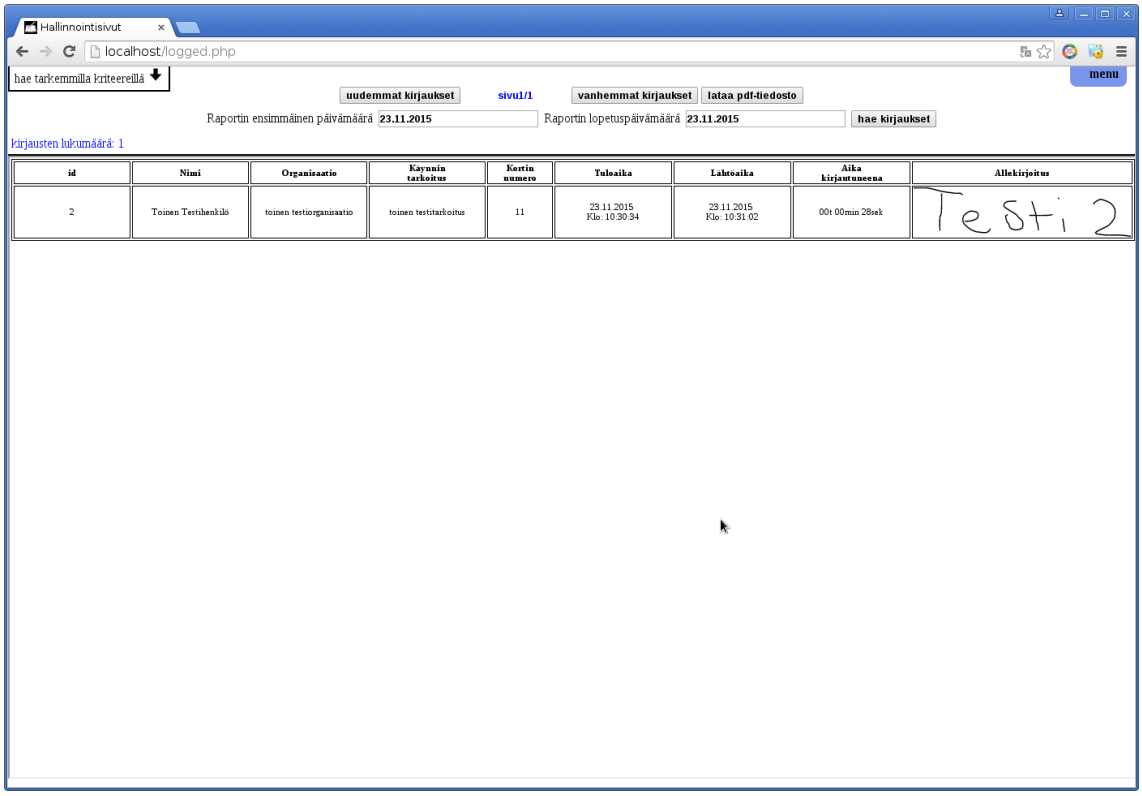

| id | Nimi | Organisaatio | Käynnin tarkoitus | Kortin numero | Tuloaika | Lähtöaika | Aika kirjautuneena | Allekirjoitus |
|---|---|---|---|---|---|---|---|---|
| 6 | Testi Henkilö | organisaatio | tarkoitus | 14 | 23.11.2015 Klo: 11:32:13 | 23.11.2015 Klo: 11:32:45 | 00t 00min 32sek | |
| 5 | Testi Henkilö | testiorganisaatio | testitarkoitus | 16 | 23.11.2015 Klo: 11:20:14 | 23.11.2015 Klo: 11:20:17 | 00t 00min 03sek | |
| 4 | Testi Henkilö | testiorganisaatio | testitarkoitus | kortiton | 23.11.2015 Klo: 11:08:42 | 23.11.2015 Klo: 11:08:44 | 00t 00min 02sek | |
| 3 | Testi Henkilö | testiorganisaatio | testitarkoitus | 16 | 23.11.2015 Klo: 10:30:56 | 23.11.2015 Klo: 10:31:19 | 00t 00min 23sek | |
| 2 | Toinen Testihenkilö | toinen testiorganisaatio | toinen testitarkoitus | 11 | 23.11.2015 Klo: 10:30:34 | 23.11.2015 Klo: 10:31:02 | 00t 00min 28sek | |
| 1 | Testi Henkilö | testiorganisaatio | testitarkoitus | 10 | 23.11.2015 Klo: 10:29:37 | 23.11.2015 Klo: 10:29:40 | 00t 00min 03sek | |

## Henkilölokin rajattuun hakuun tarkoitettu lomake

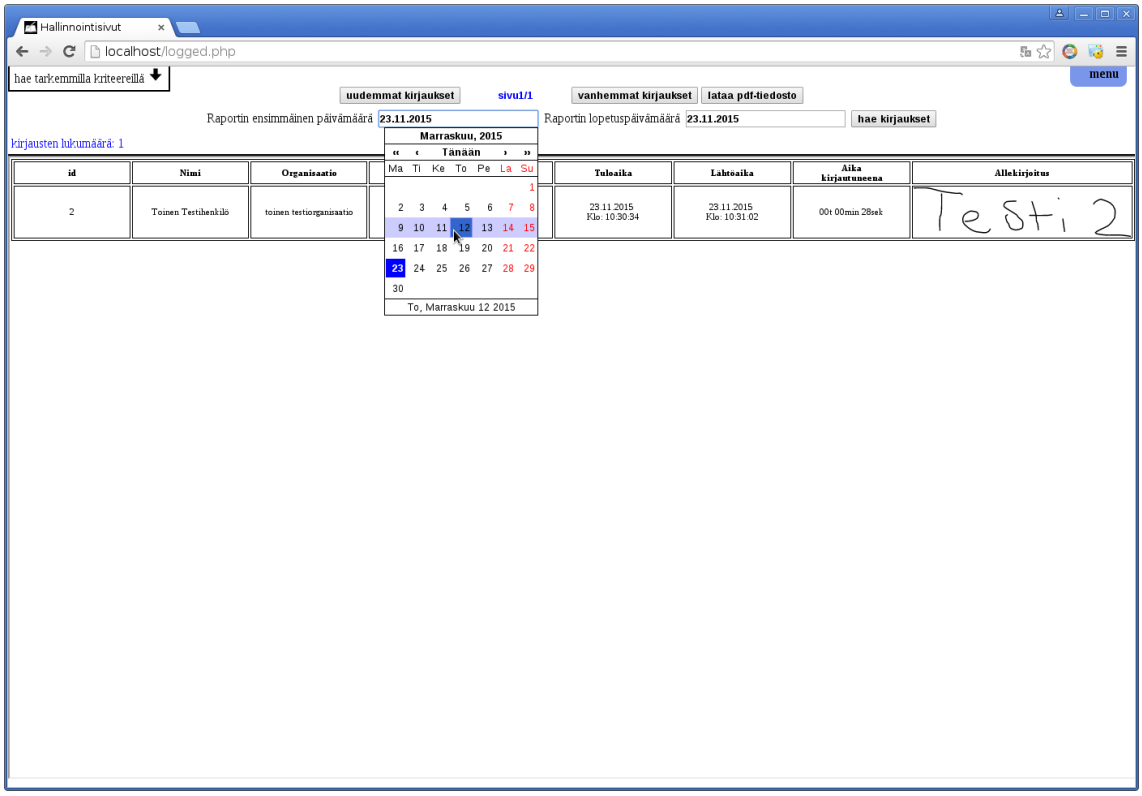



| id | Nimi | Organisaatio | Käynnin tarkoitus | Kortin numero | Tuloaika | Lähtöaika | Aika kirjautuneena | Allekirjoitus |
|---|---|---|---|---|---|---|---|---|
| 6 | Testi Henkilö | organisaatio | tarkoitus | 14 | 23.11.2015 Klo: 11:32:13 | 23.11.2015 Klo: 11:32:45 | 00t 00min 32sek |  |
| 5 | Testi Henkilö | testiorganisaatio | testitarkoitus | 16 | 23.11.2015 Klo: 11:20:14 | 23.11.2015 Klo: 11:20:17 | 00t 00min 03sek |  |
| 4 | Testi Henkilö | testiorganisaatio | testitarkoitus | kortiton | 23.11.2015 Klo: 11:08:42 | 23.11.2015 Klo: 11:08:44 | 00t 00min 02sek |  |
| 3 | Testi Henkilö | testiorganisaatio | testitarkoitus | 16 | 23.11.2015 Klo: 10:30:56 | 23.11.2015 Klo: 10:31:19 | 00t 00min 23sek |  |
| 2 | Toinen Testihenkilö | toinen testiorganisaatio | toinen testitarkoitus | 11 | 23.11.2015 Klo: 10:30:34 | 23.11.2015 Klo: 10:31:02 | 00t 00min 28sek |  |
| 1 | Testi Henkilö | testiorganisaatio | testitarkoitus | 10 | 23.11.2015 Klo: 10:29:37 | 23.11.2015 Klo: 10:29:40 | 00t 00min 03sek |  |

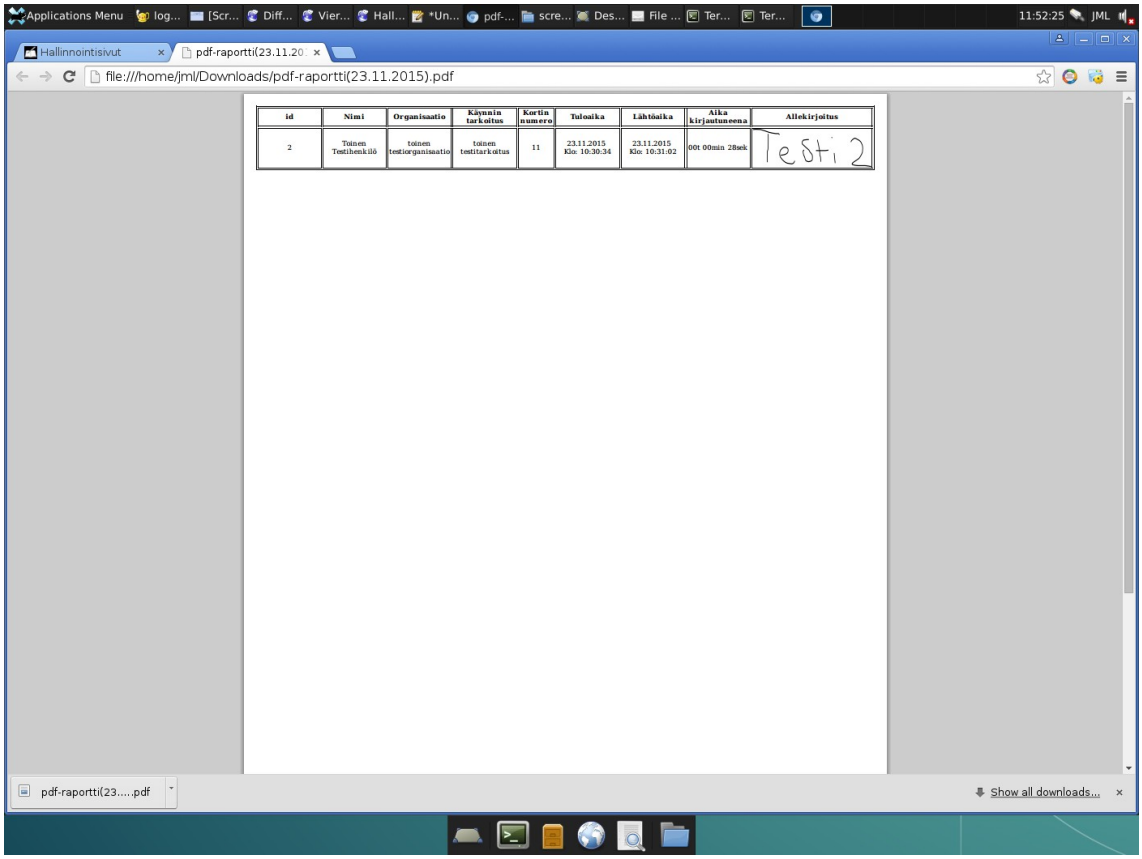
## Esimerkki rajatun haun tuloksesta



## Henkilölokin aikahaarukan valitsemislomake



## PDF-tulosteen ulkoasu



**Henkilökorttien numeroiden - ja lukumäärien manipulointisivu**