HAAGA-HELIA
University of Applied Sciences

Tietoturvallisen verkkopalvelun toteutus yhdistykselle

Juha Kukkonen

Opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

2014

HAAGA-HELIA
University of Applied Sciences

Tiivistelmä

Tietojenkäsittelyn koulutusohjelma

Tekijä Juha Kukkonen	**Aloitusvuosi** 2009
Raportin nimi Tietoturvallisen verkkopalvelun toteutus yhdistykselle	**Sivu- ja liitesivumäärä** 29 + 18
Opettajat tai ohjaajat Juhani Välimäki	

Toimeksiantajalla on tarve saada sähköinen palvelu, jonka avulla yhdistyksen jäsenet voivat ilmoittautua vuosikokouksiin ja muihin yhdistyksen tapahtumiin. Järjestelmä toimisi keskitettynä hallinta-alustana niin tapahtumille kuin jäsenillekin.

Tavoitteena oli toteuttaa tarpeiden mukainen sovellus tietoturvallisesta näkökulmasta. Tähän sisältyi tapahtumien hallinnan toteuttaminen sisältäen tapahtumien lisäämisen, päivittämisen, arkistoimisen, tapahtumiin ilmoittautuminen ja tapahtumien osallistujien listaamisen. Lisäksi tavoitteena oli toteuttaa jäsenhallinta sisältäen rekisteröinnin, omien tietojen muokkaamisen ja järjestelmävalvojan jäsentietojen muokkaamisen sekä jäsenten listauksen.

Sovellus toteutettiin käyttäen Java-ohjelmointikieltä. Tietoturvallisuuden määrittämisessä käytettiin OWASP 2013 Top 10 -listaa tietoturvauhkista sekä Valtiovarainministeriön 2013 laatimaa Sovelluskehityksen tietoturvaohjetta VAHTI 1/2013. Sovelluksen toteutuksessa käytettiin Spring-sovelluskehystä, Twitter Bootstrap CSS -kehystä, jQueryä sekä Font Awesome -ikonikirjastoa.

Tuloksena syntyi tavoitteiden mukainen tietoturvallinen sovellus. Projektia kohdanneita pieniä haasteista olivat aikataulun virhearviointi ja tekijän työllistyminen. Ne johtivat projektin uudelleen aikataulutukseen. Kaikesta huolimatta opinnäytetyöprojekti pystyttiin suorittamaan onnistuneesti sekä etuajassa uuteen aikatauluun nähden.

Asiasanat
Tietoturva, WWW-sivut, Palvelut

HAAGA-HELIA
University of Applied Sciences

Abstract

Business Information Technology

Author Juha Kukkonen	**Year of entry** 2009
Title of report Building secure web application for an association.	**Number of report pages and attachment pages** 29 + 18
Advisor Juhani Välimäki	

The client organization of this thesis project has a need to get an automated service for its members to register for the yearly meetings and other events arranged by the organization. The new system would also serve as a centralized platform for maintaining the information about the members and events.

The objective of this thesis project was to implement an application according to the needs of the client organization from the perspective of security. This included administrating events with functionalities, such as, adding events, updating events, listing event attendees, attending events and archiving events. In addition, the features of the application included member administration with functionalities, such as, registering members, updating the member info, and the administrator's view for updating the member info and listing the members.

The development of the application was done by using the Java programming language. The definition of the security aspects was done according to the OWASP 2013 Top 10 list of threats and the Ministry of Finance 2013 Security of software development VAHTI 1/2013. In the development process, Spring Framework, Twitter Bootstrap CSS Framework, jQuery and Font Awesome icon library were used.

The outcome of this thesis project was a secure application according to the client's needs. Due to some small challenges, such as the miscalculation of the schedule and the employment of author of thesis, the project had to be rescheduled. However, the project was finished successfully and ahead of the final timeframe.

Keywords
Security, Web-sites, Services

Sisällys

Raportissa käytetyt lyhenteet ja termit

API	Application Programming Interface määrittää sovelluskomponenttien toiminnon toistensa välillä.
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart on tietotekniikassa käytetty kuvavarmennustesti, jolla varmistetaan että käyttäjänä on ihminen.
CSRF	Cross-Site Request Forgery on uhka, jonka tarkoituksena on saada käyttäjä suorittamaan http-pyyntö, josta voisi olla haittaa käyttäjälle.
CWE/SANS	Common Weakness Enumeration on virallinen lista heikkouksista sovelluksissa. SANS-instituutti on suurin turvallisuuskoulutuksen ja sertifioitumisen tarjoaja.
Framework	Kehys tai runko, jonka avulla toteutetaan sovelluskehitystä.
Interceptor	Pysäyttäjä on Spring Framework -komponentti, joka suoritetaan ennen kuin kysely saapuu kontrollerille.
OWASP	Open Web Application Security Project on maailman laajuinen aatteellinen organisaatio, joka on keskittynyt parantamaan sovellusten tietoturvallisuutta.
Repository	Sijoituspaikka yleisesti koodille, jossa sitä ylläpidetään.
Scrum	Ketterä ohjausmalli.
Sprint	Scrumin työjakso, josta tässä raportissa käytetään suomenkielistä käännöstä iteraatio.
SQL	Structured Query Language, jota käytetään tietojen hakemiseen tietokannasta ja viemiseen tietokantaan.
SSH	Secure Shell on turvallinen tietoliikenneyhteys paikallisen ja etätietokoneen välille.
SSL/TSL	Secure Sockets Layers / Transport Layer Security on protokolla, joka on tarkoitettu turvalliseen ja salattuun tietoliikenteeseen Internetissä.

URL	Uniform Resource Locator on käytännössä web-osoite.
XSS	Cross-Site Scritping tarkoittaa irrallisen JavaScriptin suorittamisen sallimista verkkosivuilla

1 Johdanto

Toimeksiantaja on Helsingin Uudenkaupungin Kilta ry. Yhdistys on perustettu vuonna 1937 ja sen tarkoituksena on ylläpitää pääkaupunkiseudulla asuvien uusikaupunkilaisten kotiseuturakkautta, herättää yhdistyksen jäsenissä harrastusta ja kiinnostusta entiseen kotiseutuun. Yhdistyksen tavoitteena on kertoa jäsenistölleen Uudenkaupungin asioista ja ylläpitää vanhan kielen säilymistä.

Toimeksiantajalla on tarve sähköistää tapahtumien hallinta ja niihin ilmoittautuminen sekä sähköistää jäsenhallinta. Tarkoituksena on korvata nykyinen toimintamalli, mikä toimii lähinnä sähköpostin kautta ja on epäkäytännöllinen.

1.1 Opinnäytetyön tavoitteet

Opinnäytetytön tekijän tavoitteena on oppia lisää tietoturvallisen verkkosivun toteuttamisesta. Opinnäytetyöprojektin tarkoituksena on tehdä tietoturvalliset verkkosivut toimeksiantajalle opittujen asioiden mukaisesti.

Verkkosivujen kautta toimeksiantaja tavoittaa jäseniänsä ja markkinoi toimintaansa entistä paremmin. Verkkosivujen avulla yhdistys voi tavoittaa uusia jäseniä ja vilkastuttaa jäsentoimintaa.

1.2 Projektin tehtävä ja rajaus

Tietoturvallisten verkkosivujen toteuttamisen tarkoituksena on mahdollistaa jäsenhallinta sähköisesti. Uusi palvelu sisältää jäsentietojen lisäämisen, poistamisen ja muokkaamisen. Sovelluksella on tarkoitus sähköistää tapahtumien ilmoittaminen, niiden lisääminen, arkistoiminen ja muokkaaminen sekä tarjota jäsenille mahdollisuus ilmoittautua yhdistyksen vuosikokouksiin sekä muihin tapahtumiin. Verkkosivujen toteutukseen sisältyy kirjautumisen ja rekisteröinnin toteuttaminen.

Syntyviä tuloksia ovat tietoturvalliset verkkosivut toimeksiantajalle ja tämä opinnäytetyöraportti liitteineen.

Tässä opinnäytetyössä ei toteuteta Facebook-sivua, pikaviestitoimintaa jäsenille, ilmoitustaulua tai foorumia, eikä tiedostojen tai kuvien jakamispalvelua. Ne ovat jatkokehitysmahdollisuuksia tässä projektissa saavutetulle tulokselle.

2 Tietoturvallinen sovellus

Tietoturvallisten verkkosivujen edellytyksenä on tietojen säilyminen eheänä ja sivujen säilyminen luotettavina ja käytettävinä. Tietoturvallisuus on erittäin tärkeää. Monissa verkkopalveluissa eivät tietoturvallisuuden periaatteet toteudu kovin hyvin.

Tehtäessä tietoturvallisia verkkosivuja on otettava huomioon tietoturvallinen sovelluskehitys ja sen periaatteet. Lisäksi on otettava huomioon tietoturvallisen sovelluskehityksen hallinta.

Tietoturvallista sovelluskehitystä edesauttavat kehykset (framework), joita voidaan käyttää apuna ja perustana sovelluskehitykselle. Usein se on kannattavaa. Nykypäivänä sovelluskehityksessä käytetään paljon kehyksiä. Kehyksien tarkoitus on antaa kehittäjille perusasiat valmiina ja helpottaa niiden asioiden tekemistä, jotka tehdään jokaisessa sovelluksessa. Usein kehykset antavat turvallisuutta ja asiat, jotka normaalisti vaativat monta riviä koodia, voidaan toteuttaa vain muutamalla rivillä käyttäen kehyksiä apuna.

2.1 Tietoturvariskien välttäminen

Erittäin hyviä paikkoja lähteä tutustumaan tietoturvalliseen sovelluksen toteutukseen, on tutustua OWASP:iin. OWASP 10 sekä CWE/SANS top 25 tarjoavat paljon hyödyllistä tietoa tietoturvaongelmista ja, miten ne otetaan huomioon toteutuksessa. (Valtionvarainministeriö 2013, 42.)

Suurimmat riskit sovelluskehityksessä ovat injektiontiongelmat, rikkonainen autentikointi sekä sessiohallinta, XSS-haavoittuvuus sekä turvaton suora olioon viittaaminen. Muun muassa nämä ovat olleet kautta aikojen top 10-listalla ja siksi ne ovat riskejä, mitkä tulisi ottaa huomioon sovelluskehityksessä ja välttää niitä.

2.1.1 Haitallinen kysely

Injektoinnilla tarkoitetaan haitallista SQL-kyselyä. Se yritetään syöttää suoraan verkkosivulla olevan lomakkeen, url:in tai muun tavan kautta palvelimelle suoritettavaksi. Tämä on suuri riski, koska silloin voidaan esimerkiksi saada pääkäyttäjän oikeudet hyök-

kääjälle tai varastaa, muuttaa tai jopa poistaa tietoja sovelluksesta. Siitä aiheutuu erittäin suurta tuhoa sovellukselle, jolloin sovelluksen tietoja ei voida pitää enää ehjänä eikä sivustoa luotettavana. Se voi olla iso liiketoimintariski.

Injektointiriskin torjuminen pitää ottaa huomioon verkkosovelluksia tehdessä ja siihen on hyvä käyttää kehystä. Kehys tarjoaa parametrisoidun rajapinnan käytettäväksi SQL-kyselyjen kanssa. Jos rajapintaa ei ole käytettävissä, on syytä sulkea pois erikoismerkit käyttäen tiettyä syntaksia. Edellä mainittujen keinojen lisäksi on hyvä tarkistaa, validoida, syötekentät ja sulkea pois erikoismerkit jo ennen kuin pääsemme SQL-suoritustasolle. Mikäli sovelluksen halutaan käyttävän erikoismerkkejä, on syytä käyttää turvallista neutralisointimenettelytapaa erikoismerkkien käsittelyyn, jotta ne eivät aiheuta haavoittuvuuksia sovellukseen. (OWASP A1, 2013.)

2.1.2 Autentikointi

Hyvän autentikoinnin eli tunnistautumisen sekä tunnistamisen, ja sessiohallinnan tekeminen on usein haastavaa, ja siksi niihin suositellaan käytettäväksi kehystä. Autentikointiin sisältyy monia eri riskejä, ja haavoittuvuuksia on useita. Jos ei ole hyvää kehystä käytettävissä, on syytä tutustua OWASP:in ohjeisiin autentikoinnin ja sessiohallinan osalta, ja käyttää näitä ohjeita turvallisen autentikoinnin luomisessa.

Hyvään autentikointiin ja sessiohallintaan sisältyy oikeanlainen salasanojen suojaus, vahvojen tunnusten hallintatyökalujen luominen sekä käyttäminen, sessiotunnisteen (session id) oikeanlainen käyttäminen, session oikeanlainen hallinta sekä salauksen käyttäminen. Näiden asioiden oikeanlaisella toteuttamisella saadaan vahva autentikointi ja sessiohallinta, jota voidaan käyttää sovelluksissa. (OWASP A2, 2013.)

2.1.3 Haitallinen JavaScript

XSS-haavoittuvuudella tarkoitetaan käytännössä ulkopuolisen JavaScriptin suorittamisen sallimista verkkosivuilla. Tästä koituu usein ongelmia, sillä hyökkääjän on mahdollista muuttaa sivuston sisältöä ulkopuolelta käsin. Se voi johtaa jopa autentikaation varastamiseen.

XSS-haavoittuvuus huonon sessiohallinnan yhteydessä antaa hyökkääjälle mahdollisuuden ryöstää autentikointuneen käyttäjän sessio-id. Se antaa käytännössä hyökkääjälle mahdollisuuden toimia autentikoituneen käyttäjän nimissä kyseisessä verkkopalvelussa. Tämän haavoittuvuuden korjaamisessa täytyy ottaa huomioon syötteiden validointi ennen kuin ne hyväksytään palvelinpuolella. Se ei kuitenkaan yksin riitä estämään hyökkäystä.

Kun autentikoinnin yhteydessä käytettävää sessiokeksiä (cookie) muokataan vain http-keksiksi, ei JavaScriptillä päästä käsiksi sessiokeksiin. Tämä jo vähentää riskiä joutua sessiokaappauksen uhriksi. XSS-haavoittuvuuden estämiseksi on suositeltavaa estää HTML-tagien käyttö kaikissa kohdissa HTML-sivuilla, minne tietoa haetaan. (OWASP A3, 2013.)

2.1.4 Suora viittaus

Turvaton suora objektiin viittaaminen voi koitua turvallisuusriskiksi, jos se sallitaan ilman toimenpiteitä. Tästä esimerkkinä on URL:n loppuun syötettävä käyttäjänimi. Sen avulla haetaan yleensä käyttäjätiedot tietokannasta verkkosivulle. Tässä on kyseessä suora objektiviittaus, jossa käyttäjänimi URL-osoitteessa on sama kuin tietokannassa.

Suora objektin viittaus voidaan estää käyttämällä satunnaisia URL-listoja, joissa on satunnaisesti valittu teksti, joka vastaa objektin tunnistetta (object id) tietokannassa. Tällöin ei voida tietää kyseisen objektin id:tä, eikä tässä tapauksessa käyttäjänimeä.

Jos täytyy käyttää suoraa objektiviittausta, on syytä suorittaa pääsyoikeustarkistus käyttäjälle. Silloin joka kerta, kun viitataan suorasti objektiin, tarkistetaan käyttäjäoikeudet ennen kuin toiminto sallitaan verkkosivulla. Tämä tarkistus on hyvä suorittaa myös käyttäen URL-listoja antamaan lisäturvallisuutta. (OWASP A4, 2013.)

2.1.5 Arkaluontoinen tieto

Arkaluontoisen materiaalin näyttäminen on haitallinen uhka yksityisyydelle. Se voi altistaa tunnusten varastamiseen ilman oikeanlaisia menetelmiä. Arkaluontoisena materiaa-

lina pidetään ja käsitellään salasanoja, luottokortin numeroita, terveystilastoja, henkilökohtaisia tietoja sekä sosiaaliturvatunnuksia.

Turvatakseen arkaluontoiset tiedot on hyvä ottaa käyttöön salausmenetelmä, kuten SSL, mikä on nykyisin ehdoton vaatimus lähes jokaisessa sovelluksessa. Lisäksi sivuja, joilla käsitellään arkaluontoista tietoa, ei pitäisi jäädä välimuistiin. Automaattinen täydennys pitäisi olla asetettuna pois päältä lomakekentistä, joilla käsitellään arkaluontoista tietoa. Mikäli et tarvitse tietoa kyseisellä hetkellä, hylkää se. Tietoa mitä ei ole, ei voida varastaa. (OWASP A6, 2013.)

2.1.6 Autorisointi

Puuttuva palvelutason autorisointi voi koitua ongelmaksi verkkosivuilla, missä vain HTML-sivulla piilotetaan luvattomat toiminnot. Käytännössä tämä antaa hyökkääjälle mahdollisuuden suorittaa luvattomia toimintoja sivulla syöttämällä vain URL-osoitteen osoiterivillä kyseiselle toiminnolle ja siirtymällä osoitteeseen. Tilanteen korjaamiseksi autorisointitarkistukset pitää lisätä vähintäänkin kontrolleriin tai palvelutasolle. Suositeltavaa on lisätä kyseiset tarkistukset kumpaankin paikkaan.

Turvallisuutta on hyvä lisätä sallimalla tietyt pyyntötavat tietyille toiminnoille kontrollerissa. Eli GET-pyyntöä tarvitseva toiminto sallitaan vain GET-pyynnöllä, kun taas POST-pyyntöä vaativat sallitaan vain POST-pyynnöllä. Silloin toimintoa ei voida suorittaa virheellisellä pyynnöllä. (OWASP A7, 2013.)

2.1.7 Piilotettu toiminto

CSFR-hyökkäys voi olla haitallinen, sillä hyökkääjä pyrkii saamaan uhrin suorittamaan itselleen haitallisen toiminnon huomaamatta. Uhri ei hevin havaitse toimintoa tapahtuneeksi. Piilotettu toiminto voi tapahtua vaikka kuvan tai linkin klikkauksen kautta tai suorittamalla piilotetun JavaScriptin mukainen toiminto.

Käytettämällä useita suojausmalleja samaan aikaan saadaan aikaiseksi parempi turvallisuus. Esimerkiksi, jos hyökkääjä pyrkii saamaan uhrin siirtämään tämän varoja hyök-

kääjän tilille tai muuttamaan salasanan hänen huomaamatta, hyökkäys voidaan estää useammalla tavalla.

Yhtenä suositeltuna tapana voidaan käyttää satunnaista tunnistetta, mikä on vähintään sessiokohtainen. Tämä laitetaan jokaiseen http-pyyntöön mukaan. Silloin hyökkääjä ei voi saada oikeaa satunnaista tunnistetta kyseiseen pyyntöön. Ei ole suositeltavaa laittaa tunnistetta URL-osoitteen perään. Silloin on riski, että hyökkääjä saa tunnisteen itselleen. Vaihtoehtoisesti tätä uhkaa vastaan voidaan suojautua vaatimalla käyttäjää autentikoitumaan uudelleen toiminnon suorittamisen yhteydessä tai mahdollisesti käyttämällä CAPTCHA -toimintoa. (OWASP A8, 2013.)

2.1.8 Uudelleenohjaus

Validoimattomat uudelleenohjaukset ovat riski, joita usein käytetään oikeuksien kalastelemiseen. Uudelleenohjauksia käytettäessä on syytä pitää huoli siitä, miten se tehdään turvallisesti.

Käyttäessä uudelleenohjauksia on suositeltavaa, että URL-osoitteeseen ei anneta parametrina osoitetta, minne palvelimen pitäisi uudelleenohjata käyttäjä, sillä käyttäjä voi käydä vaihtamassa tämän helposti. Jos kuitenkin on näytettävä käyttäjälle URL:in osana uudelleenohjausosoite, on syytä tehdä pääsytarkistus sallittuja osoitteita vasten. Parempi vaihtoehto olisi, jos ei näytettäisi oikeaa osoitetta käyttäjälle lainkaan, vaan sen sijaan näytettäisiin teksti, mikä vastaa osoitetta palvelimella, ja palvelin uudelleenohjaa tekstin perusteella. Tällöin käyttäjä ei saa tietää ohjattavaa osoitetta ainakaan etukäteen. (OWASP A10, 2013.)

2.2 Tietoturvallisuuden hallinta

Ottaen huomioon edellä kuvatut varsin yleiset ongelmat tietoturvallisuus ei ole vain ohjelmointia, vaan se on myös toimintatapoja, roolien jakoa ja vastuun ottamista oikealla tavalla. Hallinnollinen tietoturvallisuus pyrkii edesauttamaan toistettavuutta yksittäisten sovelluskehitysprosessien välillä. Tähän voi hyvin tutustua sovelluskehityksen tietoturvaohjeessa. (Valtiovarainministeriö 2013, 31.)

Organisaatiossa toimivien henkilöiden on kartoitettava eri vastuualueet ja roolit. Näitä rooleja ovat esimerkiksi: arkkitehdit, projektipäälliköt, sovelluskehittäjä, tekninen päällikkö, testauspäällikkö ja kehitettävän sovelluksen omistaja. Rooleihin on valittava henkilöt ja heille varahenkilöt. Kyseiset henkilöt on koulutettava vastuualueiden tehtäviin, jotta he osaavat toimia tehtävissään noudattaen tietoturvaa. Heidän on seurattava mahdollisia ongelmia ja niiden mahdollisia toteutumia.

Valtiovarainministeriön laatiman ohjeistuksen (2013, 32 – 33) mukaan korotetulla tietoturvatasolla roolijakojen lisäksi tulee määritellä strategia. Suunnitteluvaiheessa täytyy määritellä tavoitteet ja mittarit tavoitteiden toteutumiselle. Tavoitteiden pitää olla linjassa organisaation liiketoiminnan kanssa. Lisäksi on määritettävä tarkastuspisteet strategian toteutumiselle.

Organisaation hallintaan kuuluu politiikkojen määrittäminen. Politiikkoja ovat esimerkiksi tietoturvapolitiikka, lokipolitiikka tai käyttövaltuuspolitiikka. Korotetulla tasolla politiikkoja katselmoidaan vuosittain. (Valtionvarainministeriö 2013, 34.)

2.2.1 Riskien hallinta ja ohjaus

Riskinhallinta kuuluu tietoturvallisen organisaation toimintaan. Riskien hallintaa tulee suorittaa säännöllisesti. Riskien hallinta sisältää kunkin riskin analysointia, tietoturvaan liittyvien riskien käsittelemistä ja toimintatapojen määrittelemistä riskin välttämiseksi. Organisaation on tunnistettava riskit sen ydintoiminnalle ja kirjoitettava riskinhallintadokumentti tunnistetuista riskeistä. Siihen sisältyy riskin aihe, todennäköisyys, vaikutus ja hallintataso. Riskinhallintatoimenpiteiden onnistumista seurataan ja riskin profiilin merkittävän muuttumisen jälkeen on suoritettava riskin uudelleen määrittäminen. (Valtionvarainministeriö 2013, 35 – 36.)

Tietoturvallisen sovelluskehityksen edellytys on osaamisen kehittäminen sekä kouluttaminen. Organisaatioissa tulee olla tietoturvavastaavia, henkilöitä, jotka antavat tarvittavan koulutuksen niin uusille kuin useamman vuoden töitä tehneille työntekijöillekin. Koulutus tulisi suorittaa toistuvasti vähintään yhden kerran vuodessa. Tietoturvakoulutuksessa tulee käydä läpi tyypillisiä tietoturvaongelmia sovelluksille, joita organisaatio

toteuttaa ja käyttää. Organisaation olisi hyvä tehdä tietoturvaohjeistusdokumentti, mikä sisältää listan kaikista organisaation kannalta oleellisista tietoturvadokumenteista ja web-osoitteista, jotka tarjoavat tietoturvaohjeistusta työntekijöille. Korotetulla tasolla työntekijöille pitäisi olla mahdollista antaa roolikohtaista koulutusta ja tukea. Koulutustilaisuuksiin osallistumisia seurataan ja niistä kerätään raportteja organisaation yleisen tietoturvan kartoittamiseksi. (Valtitonvarainministeriö 2013, 36 – 38.)

2.2.2 Vaatimuksia tietoturvalle

Tekniseen tietoturvaan kuuluu työasemien perustietoturvallisuus, kuten päivitysten asentaminen, virustorjunta ja pääsynhallinta. Perustietoturvallisuuden lisäksi tekniseen tietoturvallisuuteen kuuluu tietoturvallisten ratkaisujen määrittäminen, ympäristöjen eristäminen, komponenttikirjastojen käyttäminen, versiohallinnan käyttäminen sekä varmuuskopiointi. Tekniset tietoturvaratkaisut riippuvat sovelluksen tietoturvavaatimuksista. Niihin luetaan yleensä autentikointi, autorisointi, virheidenkäsittely ja lokien keräys. Käytetyistä komponenteista tulee muodostaa paikallinen komponenttikirjasto, jota ainoastaan saadaan käyttää sovelluksia kehittäessä. Näin voidaan varmistaa sovelluksen toimivuus tietoturvalliseksi ja määritysten mukaiseksi.

Eriytettyihin ympäristöihin kuuluu hyvin usein kehitysympäristö sekä tuotantoympäristö. Testausympäristö voidaan myös eriyttää näistä kolmesta ympäristöstä. Varmuuskopioinnista on huolehdittava ja kehitykseen liittyvät asiat on oltava säännöllisen varmuuskopioinnin alaisuudessa. Näitä ovat muun muassa versiohallinta, lokit sekä sovelluskehitykseen käytetyt wikit.

Jos tuotantopalvelin tai kehityspalvelin tai testipalvelin sijaitsee ulkoverkossa, on yhteys siihen salattava vahvalla salauksella. Näitä salauksia ovat esimerkiksi SSL/TSL ja SSH. Korkeammilla tasoilla on kaikki tehtävät eriytettävä sekä kaikki komponentit määritettävä tietoturvallisuusasetuksen antamien suojaustasojen mukaan. (Valtionvarainministeriö 2013, 38 – 40.)

2.2.3 Tietoturvan suunnittelu sovelluskehitysprosessissa

Sovelluskehitystä voidaan toteuttaa eri sovelluskehitysmalleilla, joita ovat muun muassa vesiputousmalli eli vaihejakomalli ja V-malli. Vesiputousmallia käyttämällä on usein vaikea suorittaa yhtä vaihetta loppuun ilman palaamista aiempiin vaiheisiin. Tämä aiheuttaa helposti lisäkustannuksia, joka voi johtaa projektin jäädyttämiseen. V-malli on vastaava malli, mutta lähestyy sovelluskehitystä testauksen näkökulmasta.

Scrum on tyypillisin ketterä ohjausmalli ja sitä käytetään nykyisin paljolti. Ketterät menetelmät perustuvat iteratiiviseen toimintamalliin, jossa sovelluskehitysprosessia suoritetaan niin sanotusti kierroksissa, jossa jokaiseen kierrokseen sisältyy tarvittaessa kaikki kehitysprosessin vaiheet. Näitä kierroksia suoritetaan tiiviissä yhteistyössä sidosryhmien kanssa. (Valtionvarainministeriö 2013, 43 – 44.)

Tietoturvallisessa sovelluskehitysprosessissa on kaikki samat vaiheet riippumatta mitä sovelluskehitysmallia käytetään. Näitä vaiheita ovat esitutkimus, vaatimusmäärittely, suunnittelu, toteutus, testaus, käyttöönotto, ylläpito ja käytöstä poisto. Esitutkimusvaiheessa otetaan huomioon lait, asetukset, organisaation periaatteet ja ohjeet, sidosryhmien vaatimukset sekä muut vaatimukset. Näitä kaikkia apuna käyttäen tulee määritellä sovelluksen tietoturvataso. Määrityksiä käytetään koko sovelluskehitysprosessin ajan sen tietoturvamekanismien valinnassa ja toteutuksessa. Vaatimusmäärittelyssä toteutetaan tietoturvatason, tietoturvaratkaisujen, lainsäädäntöjen, riskianalyysien, tietoturvavaatimuksien, tietoturva-analyysien, uhkamallinnuksien ja arkkitehtuurilinjauksien määrittäminen. Suunnitteluvaiheessa toteutetaan määritetyt asiat käyttäen yleisesti tunnettuja standardeja. Tämä tekee järjestelmästä helposti integroitavan ja tietoturvallisemman.

Sovellus on suunniteltava helposti päivitettäväksi ja yksityiskohtariippumattomaksi käyttäen yleisesti tunnettuja turvallisia suunnittelumalleja. Toteutusvaiheessa toteutetaan suunnitellut asiat määritysten mukaisesti. Toteutuksessa on otettava huomioon virheidenkäsittely, lokit, suojaus, yhteinen aikalähde, käyttäjäntunnusten hallinta ja yleisimmät tietoturvaongelmat ohjelmoinnissa. Testauksessa suoritetaan ja varmistetaan sovelluksen toimivuus ja tietoturvallisuus määrityksien mukaisesti. Testauksessa ei saa käyttää samaa dataa kuin tuotantoympäristössä, vaan testidata on generoitava erikseen.

Viimeinen käyttöönottotesti tulee suorittaa tuotantoympäristön kaltaisessa ympäristössä. Käyttöönotossa ylläpitäjät vastaavat kaikesta konfiguroinnista, asennuksista ja dokumentin kirjoittamisesta. Asennuksen yhteydessä alkuperäisasetukset muutetaan tuotantoympäristön asetuksiksi. Valtiovarainministeriön ohjeen (2013, 60) mukaan kirjoitetaan niin sanottu kovennusdokumentti, jossa kuvataan asetusten vaikutukset tietoturvaan. Kovennusdokumentti sisältää seuraavat asiat:

- Oletusarvoiset salasanat ja käyttäjätunnukset
- Hallintaliittymien näkyvyys
- Vianselvitysominaisuuksien poistaminen
- Mahdollinen testidata
- Salauksen aktivointi, omien sertifikaattien poistaminen ja luotettujen sertifikaattien käyttöönotto
- Sertifikaattien elinkaaren seuranta
- Tietoturvallisuuteen vaikuttavat sovellusasetukset.

Ylläpitovaiheessa seurataan sovelluksen tietoturvallisuuden tasoa sekä vaaditun suorituskyvyn riittävyyttä. Päivitykset tulee asentaa ja aloittaa reilusti ennen tukiajan päättymistä. Tietokantaan tehtävien muutosten tulee jäädä tietokantalokiin, ja lupa on kysyttävä tuotteen omistajalta. Sovelluksen tyypillisimmät virhetilanteet tulee olla dokumentoituna. Varmuuskopioinnista on huolehdittava ja sovelluksen käyttämät tiedot vähintäänkin tulee varmuuskopioida. Käytöstä poiston aikana sovelluksen sisältämälle tiedolle tehdään arvon määritys, jonka perusteella tiedot joko poistetaan tai säilytetään. Tiedoista voidaan säilyttää vain osa tai tuhota osa. (Valtionvarainministeriö 2013, 45 – 49, 54 – 66.)

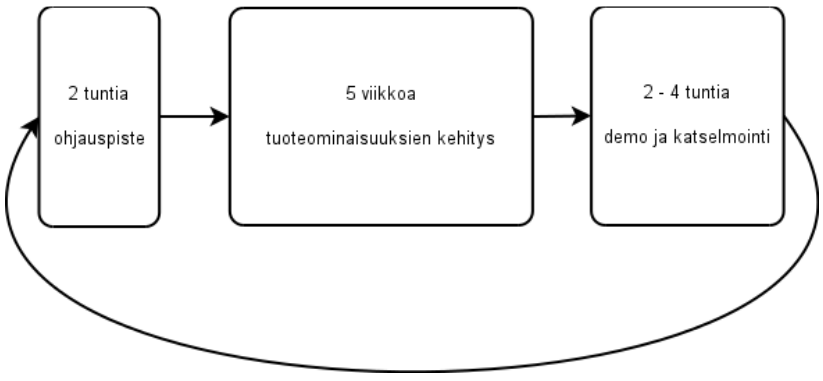
3 Opinnäytetyöprojektin toteutussuunnitelma

Toimeksiannon mukainen sovelluksen kehittäminen aloitettiin kartoittamalla, pohtimalla, keräämällä ja testaamalla mahdollisia tekniikoita sekä toimintatapoja. Selvitin, miten sovellus kuuluisi toteuttaa, jotta se vastaa tilaajan toiveita, tarvittavia ominaisuuksia, on helppokäyttöinen ja helposti lähestyttävä sekä tietoturvallinen.

3.1 Sovelluskehityksen vaiheet

Sovelluskehitys aloitettiin suunnittelemalla ulkoasu ja toteuttamalla mock-up -luonnoksia rautalankamallikuviksi sovelluksen ulkoasusta, ilmeestä ja elementtien sijainnista. Sitten aloin suunnitella alustavaa versiota tietokannasta, jonka jälkeen toteutin tietokannan ensimmäisen version. Tietokantaa muutettiin sovelluskehityksen edetessä ja tarpeen niin vaatiessa. Kun tietokanta oli saatu kuntoon, laadin ensimmäisen version verkkosivuista. Se oli lähinnä tulevan sovelluksen ulkoasuratkaisu. Ulkoasun toteutuksen jälkeen kehitin sovellusta käyttäen Scrumin mukaista ohjaustapaa: suunnittelin sprinttien eli iteraatioiden sisällön ja neuvottelin niistä tilaajan edustajan kanssa.

Scrum oli minulle tuttu jo opinnoistani ennen opinnäytetyötä. Ohjauskäytäntö on yleinen sovelluskehitysyrityksissä ja tarjoaa paremman läpinäkyvyyden toteuttamiseen kuin vesiputousmalli. Tässä projektissa toteutin iteraatioissa aina sovitun palasen kerrallaan. Sovelluskehitysprojektin iteraatiot päätettiin viisiviikkoisiksi. Kuvassa 1 on esitetty työskentelyni rytmiä.



Kuva 1. Työskentely iteraatioissa

Näin varmistettiin työrauha iteraatioissa ja selvästi rajattujen tuloksien aikaan saaminen. Sovelluksen kehitystyön aikana laadin sovellukseen liittyviä kuvauksia ja kaavioita sekä kirjoitin tätä opinnäytetyödokumenttia. Lopulta viimeistelin dokumentit ja kaaviot sovelluksen jatkokehittämistä varten. Viimeisenä vaiheena on sovelluksen luovutus tilaajalle ja projektin päättäminen.

Sovelluksen käyttöönottoon on laadittu kovennusohje. Se on kuvattu liitteessä 3.

3.2 Sovelluksen ratkaisut

Sovellus toteutettiin tietoturvallisia menetelmiä käyttäen. Ohjeena toimii OWASP 2013 Top 10 sekä sovelluskehityksen tietoturvaohje (VAHTI). Ne antavat sovellukselle perustietoturvaedellytykset ja -tason. Molemmat on otettu sovelluskehitykseen mukaan oppimiskokemuksena opinnäytetyöaiheen mukaisesti.

Toteutuksessa käytettiin versiohallintaa ohjelmakoodin säilyttämiseksi ja varmuuskopioinniksi. Versiohallintaan otettiin käyttöön Git. Se on nopea, monipuolinen, hyvin toimiva, ja yhä useammat sovelluskehittäjät käyttävät Git-versiohallintaa. Git:lle ominaista on sen kaksi sijoituspaikkaa (repository): paikallinen ja etäsijoituspaikka. Git:n sanotaan olevan hajautettu versiohallinta, kun esimerkiksi SVN on keskitetty versionhallintajärjestelmä. Git eroaa muista versiohallintajärjestelmistä juuri tästä syystä. Sovelluksen toteuttajana minulla on vähemmän kokemusta Git-versionhallintajärjestelmästä kuin SVN:stä.

Sovellus kehitettiin käyttäen Java-ohjelmointikieltä Eclipse-sovelluskehitysohjelmalla. Sovelluksen toteuttajana minulla oli eniten kokemusta Java-ohjelmoinnista ja web-sovelluksen kehittämisestä käyttäen Java-ohjelmointikieltä. Ohjelmoinnissa käytin myös Spring-kehystä.

Spring-kehys edesauttaa sovelluksen ohjelmointia ja yksinkertaistaa sovelluksessa tarvittavien komponenttien kirjoittamista huomattavasti. Lisäksi Spring on hyvin joustava, ja minulla oli siitä entuudestaan kokemusta. Spring-kehykseen on saatavilla lisäkirjastoja, joita on mahdollista käyttää. Sovelluksessa käytettiin jQuery-javascriptkirjastoa. Se an-

taa mahdollisuuden toteuttaa hieman käyttäjäystävällisempiä verkkosivuja, ja minulla oli tarvittavaa osaamista.

Sovellus käyttää MySQL-tietokantaa. Se on paljon sovelluksissa käytetty ilmainen tietokanta, ja minulle tuttu. Sovelluksen alustana toimii Tomcat7-web-palvelin Java-sovelluksille. Se on myös paljon käytetty. Tämän palvelimen eteen laitetaan Apache2-web-palvelin, jotta verkkosivut olisivat tavoitettavissa normaalisti selaimen kautta. Apache2-palvelinta käytetään Tomcat7-palvelimen edessä yleisesti monissa verkkosivuissa, sillä Tomcat itsessään sisältää portteja auki moneen suuntaan ja on näin ollen tarkoitettu toimimaan taustalla. Lisäksi Apache2-webpalvelin on helpommin konfiguroitavissa kuin Tomcat7-web-palvelin. Tällöin salausasetukset asetetaan Apache2 -webpalvelimelle.

Verkkosivujen toteutuksessa käytin Twitter Bootstrap CSS -kehystä. Tämä antaa sivulle perusasiat ja -tyylit valmiiksi ja tuen monille selaimille. Käytin myös Font Awesome -ikonikirjastoa, koska se tarjoaa paljon hyviä ikoneja rikastuttamaan sivuston ulkoasua. Gimp-kuvankäsittelyohjelmaa käytettiin sovelluksessa tarvittavien taustojen tekemiseen. Gimp on ilmainen ja open source -työkalu, jolloin ei tarvitse murehtia lisensseistä. Se on otettu mukaan oppimiskokemukseksi.

4 Sovelluskehitysprosessin hallinta

Sovelluskehitystyötä ohjattiin Scrum menetelmällä iteraatiossa. Tämä oli luonteva toimintatapa tähän sovelluskehitysprojektiin, ja tarvittavat asiat saatiin toteutettua.

Tämän projektin alkuvaiheessa iteraation (sprint) kesto oli määritelty kolmeviikkoiseksi. Kahden kuukauden kuluttua se ei ollut enää realistinen tavoitteen saavuttamiseksi. Niinpä projektin aikataulutu jouduttiin tekemään uudelleen. Uuden aikataulun iteraatiot muutettiin viisiviikkoisiksi. Uusi aikataulu oli sopiva ja pitävä, ja sovellus valmistui lopulta hiukan etuajassa.

4.1 Vaatimusmäärittely ja ulkoasun suunnittelu

Vaatimusmäärittelyksi katsottiin riittävän mock-up -rautalankamallikuvat, jotka määrittelivät ulkoasun perusnäkymän. Näiden tekemiseen käytettiin netissä olevaa MockFlow -sovellusta, jota voidaan käyttää ilmaiseksi. Sovelluksen käyttäminen vaatii kuitenkin rekisteröitymisen, joka on ilmaista.

Liitteessä 1 esitetyt kuvaukset ovat hyvin varhaisen vaiheen versioita. Ne on tehty sovelluksen perusulkoasun määrittämiseksi jo ennen kuin sovelluksen toteutus aloitettiin. Näin ollen kuvaukset eivät ole enää täysin toteutetun sovelluksen ulkoasun mukaiset, sillä ulkoasu on muuttunut sovelluskehitysprosessin edetessä.

4.2 Tietokannan suunnittelu ja toteutus

Tietokantana on MySQL-relaatiotietokanta. Relaatiokaavio määrittää tietotyypit ja kannan rakenteen. Tietokanta on normalisoitu viidenteen normaalimuotoon.

Tietokannan suunnittelussa on käytetty Dia-ohjelmaa, mikä on ilmainen kaavioiden piirtämisohjelma. Aluksi hahmottelin hieman tauluja ja tein alustavan luokkakaavion. Tämän jälkeen hahmottelin ja suunnittelin relaatiokaavion, toteutin kannan luontilauseet ja testidatan. Viimeisenä vaiheena oli lauseiden suorittaminen. Tietokannan luontilauseet ja relaatiokaavio on esitetty liitteessä 2.

4.3 Sovelluksen toteutus ja tietoturva

Sovellus sisältää kolme isompaa kokonaisuutta. Ne ovat tapahtumahallinta, jäsenhallinta, ja turvallisuus. Toteutuksesta jätettiin kolme toiminnallista ominaisuutta pois, sillä ne eivät mahtuneet opinnäytteenä toteuttavaan laajuuteen tai projektin laajuus olisi kasvanut liian suureksi. Rajatut ominaisuudet olisivat vaatineet huomattavasti enemmän aikaa toteutustyölle ottaen huomioon opinnäytetyöprojektin tarkoituksen ja valitun tietoturvanäkökulman.

Sovelluskehityksen tarkoitus tässä opinnäytetyössä oli toteuttaa tietoturvallisesta näkökulmasta verkkosivut yhdistykselle. Sovellukselle on määritetty perustietoturvataso sovelluskehityksen tietoturvaohjeen (VAHTI) pohjalta viitaten OWASP 2013 Top 10 -listaan. Uhkalistalta toteutettiin asiat, jotka nousivat esille kyseessä olevan tietoturvaohjeen sisällöstä.

Sovelluksen antamat globaalit virheilmoitukset käyttäjälle ovat:

- 400 Bad request
- 401 Unauthorized
- 404 Not found
- 405 Parameter not found
- 503 Service unavailable (Service is temporarily unavailable).

Sovellus ei paljasta käyttäjälle mitä tapahtuu niin sanotun ”pellin alla”, vaan antaa käyttäjälle ymmärrettävämpiä virheilmoituksia virheiden sattuessa. Se on yksi tietoturvallisuuden perusedellytys, jotta sovellus olisi vaikeammin murrettavissa.

Sovelluskehitysprojektin aikana on opittu paljon uutta aisaa tietoturvallisesta sovelluskehityksestä ja sen vaatimuksista. Sovellus on toteutettu tietoturvalliseksi opittujen asioiden mukaisesti.

4.4 Turvallisuuden näkyminen sovelluksen toteuttamisessa

Kaikista osa-alueista tämän osa-alueen toteuttaminen vaati enimmän osan työajasta. Toteuttavaan kokonaisuuteen kuului sovittujen toiminnallisuuksien kartoittaminen, selvittäminen ja haluttujen ominaisuuksien tietoturvallinen toteuttaminen. Ratkaisussa vähintään 70 prosenttia on uutta ja tietoturvaan tähtäävää asiaa, mikä piti ottaa huomioon toteutuksessa.

4.4.1 Haitallisen kyselyn ehkäiseminen

Injektointiriskin ehkäisy on toteutettu muutamassa eri paikassa ohjelmakoodissa. Ne paikat ovat olio-luokat sekä dao-luokat. Päivämääriä on validoitu räätälöidyssä validaattorissa sekä olio-luokissa. Ehkäisymenetelminä, joita käytettiin sovelluksessa, ovat parametrisoitu rajapinta sekä syötteiden validointi. Seuraavassa on otteita ohjelmakoodista. Otteet kuvaavat tiettyä tietoturvariskiin varautumista ja sen välttämistä.

Kuvassa 2 näkyy, kuinka SQL-kyselyyn upotettuun kysymysmerkkiin liitetään metodin mukana tuleva id.

```
GET_ROLES_FOR_USER = "SELECT j.j_etunimi, j.j_sukunimi, r.r_nimi "
        + "FROM jasen j "
        + "JOIN jasen_rooli jr ON jr.j_id = j.j_id "
        + "JOIN rooli r ON jr.r_id = r.r_id WHERE j.j_id = ?"

@Override
public List<Rooli> getRoles(int id) {
        log.info("Getting roles by id " + id);
        return jdbcTemplate.query(GET_ROLES_FOR_USER,
                new Object[]{id}, new RooliRowMapper());
}
```

Kuva 2. Spring-kehyksen parametrisoitu perusrajapinta

Kuvassa 3 näkyy, kuinka lomakkeiden kentät on validoitu käyttäen Spring-kehyksen tarjoamaa validointimenetelmää annotaatioilla.

```
public class JasenImpl implements Jasen {

        private int jId;
        @NotBlank
        @Pattern(regexp="[a-zöää\\s-]*",
                flags=Pattern.Flag.CASE_INSENSITIVE)
        private String jEtunimi;
        @NotBlank
        @Pattern(regexp="[a-zöää\\s-]*",
                flags=Pattern.Flag.CASE_INSENSITIVE)
        private String jSukunimi;
        @NotBlank
        @Pattern(regexp="((\\d){2}\\.(\\d){2}\\.(\\d){4})?",
                flags=Pattern.Flag.CASE_INSENSITIVE)
        private String jSyntymaaika;
```

Kuva 3. Syötteiden validointi annotaatioina

Kuvassa 4 näytetään esimerkki räätälöidystä tapahtuman päivämäärän validoinnista.

```
if (tapahtuma.getAloituskello().trim().length() == 0 ||
        !this.matches(tapahtuma.getAloituskello(),
                Pattern.compile("([0-9]{2}:[0-9]{2}){1}"))) {
        FieldError ak = new FieldError("uusi-t", "aloituskello",
                tapahtuma.getAloituskello(), false,
                new String[]{"Pattern.uusi-t.aloituskello"}, null,
                        "must match ([0-9]{2}:[0-9]{2}){1}");
        result.addError(ak);
}

private boolean matches (String text, Pattern pattern) {
        Matcher matcher = pattern.matcher(text);
        return matcher.matches();
}
```

Kuva 4. Räätälöity päivämäärän validointi

4.4.2 Autorisoinnin ja autentikoinnin toteuttaminen

Spring Security -lisäkirjastolla määritettiin perustietoturva-asetuksia. Muut osa-alueet toteutettiin räätälöidysti. Aluksi oli tarkoituksena toteuttaa koko autorisointi käyttäen Spring Security -lisäkirjastoa. Tämä ei ollut muutamien päivien kokeilujen ja implementointiyrityksien jälkeen toteutettavissa. Jotta autorisointi olisi saatu toteutettua käyttäen Spring Secrurity -lisäkirjastoa, olisi pitänyt muokata Spring Security -lisäkirjaston koko autorisointiketjua. Tämän tekeminen olisi vaatinut huomattavasti enemmän aikaa testaukselle, tutkimiselle ja implementoinnille. Se olisi johtanut huomattavaan toteuttavan kokonaisuuden rajaamiseen.

Autentikoinnin turvalliseen toteutukseen sisältyy web.xml-tiedoston konfigurointi, turvallinen autentikointi sekä turvallinen autorisointi. Salasanat suojataan SHA-256

-salausalgoritmilla ja suolataan 32-bittisellä numeroaakkosyhdistelmällä. Sivustolla käytetään SSL-salausta liikenteen turvaamiseksi.

Kuvassa 5 näytetään esimerkki web.xml tiedoston konfiguroinnista. Siinä asetetaan sessiolle vaatimuksia, jotka ovat http-only ja secure. Http-only:n ollessa voimassa sessiokeksiin ei voida päästä käsiksi JavaScriptillä. Secure tarkoittaa sitä, että sessiokeksi lähetetään vain SSL-yhteyden kautta.

```
<session-config>
    <cookie-config>
        <http-only>true</http-only>
        <secure>true</secure>
    </cookie-config>
</session-config>
```

Kuva 5. Web.xml tiedosto-ote

Kuvassa 6 näytetään esimerkki sovelluksen autentikointikoodista. Sovelluksessa, käyttäjän kirjautuessa sisään, edellinen istunto suljetaan ja uusi käynnistetään. Näin toimitaan käyttäjän muuttaessa omia tietoja sivustolla.

```
private void authorize (HttpServletRequest request,
        Jasen jasen, List<Rooli> roles) {
        AuthorizedJasen authorized = new AuthorizedJasenImpl(
                jasen.getJId(), jasen.getJEtunimi() +
                        " " + jasen.getJSukunimi(), roles);
        request.getSession().invalidate();
        request.getSession().setAttribute("context", authorized);
}
```

Kuva 6. Ote autentikointimetodista

Turvalliseen autorisointiin sisältyvät metodien määritykset ja pääsytarkistukset palvelutasolla ja kontrollerissa. Alla oleva kuva 7 sisältää esimerkin annotaatiosta ensimmäisellä rivillä. Tällä voidaan määrittää pyyntötapa, jolla metodi on tavoitettavissa. Toisella rivillä on pääsytarkistusesimerkki sovelluksen kontrollerissa. Viimeisenä on pääsytarkistusesimerkki palvelutasolla.

```
@RequestMapping(value="/logout", method=RequestMethod.POST)

AuthorizationLevel level = getKiltaSecurityContext()
        .hasAnyRole(new String[]{ROLE_ADMIN, ROLE_MEMBER});
if (level.equals(AuthorizationLevel.AUTHORIZED)) {
        ...
}


if (getKiltaSecurityContext()
        .hasRoles(new String[]{ROLE_ADMIN, ROLE_MEMBER})) {
         ...
}
```

Kuva 7. Kolme erilaista autorisointia

4.4.3 Haitallisen JavaScriptin ehkäiseminen

Haitallisen JavaScriptin ehkäisemiseksi voidaan määrittää sessiokeksi vain http-only -keksiksi. Se on tehty kuvassa 5. Tietoa haettaessa web-sivulle voidaan käyttää jstl:ää. Seuraavassa (kuva 8) on esimerkki tiedon sijoittamisesta verkkosivulle. Kuvassa olevalla tagilla on xml:n ehkäisyominaisuus automaattisesti. Se estää HTML-tagit paikoissa, missä niistä voisi koitua ongelmia.

```
<c:out value="${event.getNimi()}"/>
```

Kuva 8. Jstl-esimerkki

4.4.4 Turvallinen objektiin viittaaminen

Objektiin viitattaessa on syytä tehdä pääsyoikeustarkistukset. Ne voidaan tarkistaa kuvan 7 esimerkkien mukaisesti. URL-listoja käytetään turvalliseen objektiin viittaamisessa. Näihin listoihin määritetään URL, mikä näytetään web-osoitteessa ja id, mikä vastaan objektia tietokannassa. Sovellukseen on toteutettu räätälöity URL-lista.

Kuvassa 9 ensimmäisellä rivillä haetaan nimenmukainen URL verkkosivulla. Rivillä kaksi haentaan URL osoitteen mukainen id palvelinlogiikan puolella.

```
<c:out value="${urls.getUrl(event.getNimi())}" />

Integer id = UrlMapper.getIdByUrl(name);
```

Kuva 9. URL-listan käyttöesimerkkejä

4.4.5 Arkaluontoisen tiedon turvaaminen

Arkaluontoisen tiedon turvaamiseksi on ehdottoman tärkeää käyttää salausta, kuten SSL. Sivut, joilla arkaluontoista tietoa käsitellään, eivät saa jäädä välimuistiin, eikä tieto saa jäädä lomakekenttien muistiin.

Kuvassa 10 on esitetty määrityksiä, mitkä tulisi asettaa JSP-sivun ylälaitaan välimuistiin jäämisen estämiseksi.

```
<%response.setHeader("Cache-Control", "no-cache, no-store, must-revalidate");
response.setHeader("Pragma", "no-cache");
response.setDateHeader("Expires", 0);%>
```

Kuva 10. Sivun jäämisen välimuistiin estäminen

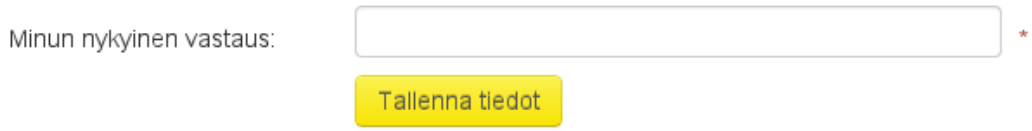
Kuvassa 11 määrittämällä autocomplete off-asentoon, pystytään estämään lomakkeessa käsiteltävien tietojen jääminen lomakekenttien muistiin.

```
<form:input autocomplete="off" path="syntymaaika" class="input-100" type="text" />
```

Kuva 11. Tietojen jääminen lomake kenttiin estäminen

4.4.6 Piilotetun toiminnon ehkäiseminen

Piilotetun toiminnon ehkäiseminen on toteutettu sovelluksessa pyytämällä käyttäjiä autentikoitumaan uudelleen, mikäli he suorittavat tietojen muokkausta. Tästä on esimerkki kuvassa 12.



Kuva 12. Esimerkki uudelleen autentikoitumisesta

4.4.7 Turvalliset uudelleenohjaukset

Sovelluksessa ei ole paljon uudelleenohjauksia, mikä on jo itsessään ehkäisevää toimintaa. Sovellus käyttää uudelleenohjausta käyttäjän kirjautuessa sisään. Uudelleenohjaus-

osoite tallennetaan keksiin, mikä on voimassa kaksi (2) minuuttia tai kunnes kirjautuminen on suoritettu onnistuneesti. Tästä on esimerkki kuvassa 13.

```
Cookie rCookie = new Cookie("r", request.getRequestURI());
rCookie.setHttpOnly(true);
rCookie.setPath(request.getContextPath());
rCookie.setMaxAge(120);
response.addCookie(rCookie);
```

Kuva 13. Uudelleenohjauskeksi

4.5 Tapahtumien hallinnan ja tapahtumaan ilmoittautumisen toteuttaminen

Tapahtumien hallinta on yksi kolmesta toteutettavan sovelluksen toiminnallisesta kokonaisuudesta. Tapahtumienhallinnan toteutus eteni suunnitelmien mukaisesti. Kaikki siihen vaaditut toiminnallisuudet on toteutettu. Tapahtumien hallinnan toteuttaminen oli käytännössä aikaisemman osaamisen hyödyntämistä ja uudelleen käyttöä.

Tapahtumiin ilmoittautuminen vaati suurimman osan projektiin varatusta työajasta. Ilmoittautumisessa jouduttiin selvittämään, kokeilemaan ja miettimään mahdollisia ratkaisuja toteuttamiseksi, koska kyseinen toiminnallisuus toi mukanaan asioita, joista ei ollut aikaisempaa kokemusta. Ilmoittautumiseen sisältyi myös useita tietoturvaan liittyviä kysymyksiä ja tilanteita. Tietokantaa jouduttiin muuttamaan hieman pariinkin kertaan, jotta osallistujien listaus voitiin toteuttaa oikein eli halutun kaltaisena.

4.6 Jäsenhallinnan toteuttaminen

Jäsenhallinnan toteutus oli kolmas ja viimeisin toteutettava kokonaisuus sovelluksessa. Toiminnallisuus sisältää rekisteröitymisen, omien tietojen muokkaamisen ja järjestelmävalvojan muokkaustilan sekä jäsenlistauksen.

Toteutus oli todella nopeaa ja helppoa, sillä siinä seurattiin jo sovelluskehityksen aikana tullutta rutiinia ja tehtyjä valintoja ja ratkaisuja. Siksi tämän toiminnallisuuden toteuttamiseen kului vähiten aikaa.

5 Projektin ja oppimisen arviointia

Projektissa sovellukselle asetettiin perustietoturvataso. Sovelluksen tietoturvallisuus on määritetty Valtionvarainministeriön ohjeen mukaisesti viitaten OWASP 2013 Top 10 -listaan. Opinnäytetyön aikana opittiin laajemmin ja sovellettiin käytäntöön tämän raportin luvussa 2 kuvatut osa-alueet, jotka liittyivät tietoturvaan.

Sovelluskehitysprojektin osalta saavutin tavoitteet vähintään 90-prosenttisesti. Täydellinen suoritus jäi uupumaan tietoturvallisuuden ollessa erittäin suuri osuus koko sovelluksen kehittämisessä. Siksi joitakin toiminnallisuuksia ja toiminnallisuuskokonaisuuksia päätettiin projektin aloitusvaiheessa rajata opinnäytteen ulkopuolelle. Väistämättä pitkin sovelluskehitysprojektia on täytynyt toteuttaa turvallisuuskorjauksia sovellukseen. Ne parantavat sovelluksen ja sen käytön tietoturvallisuutta.

Tietoturvallisuus ei ole pelkästään hyvä autorisointimenetelmä, vaan se näkyy ihan perusasioissa ja niiden toteutuksen huolellisuudessa. Tästä esimerkkinä ovat oikeanlainen virheiden käsittely ja syötteiden validointi. Sovelluksen koon kasvaessa virheidenkäsittely laajenee huomattavasti, sillä niin virhetilanteisiin varautuminen kuin virheiden, joita täytyy käsitellä oikeaoppisesti, määrä kasvaa samassa suhteessa.

Tietoturvallisuuden osalta projekti oli mielenkiintoinen. Projektin aikana olen oppinut paljon ja uusia asioita, jotka täytyy ottaa huomioon, sekä oppinut, miten ne otetaan huomioon sovelluskehityksessä.

Kirjallisuusselvityksessä lähteinäni olivat VAHTI ja OWASP 2013 Top 10 -uhkalista. VAHTI Sovelluskehityksen tietoturvaohje antaa selkeän kuvan siitä, miten jokaisen julkisen sektorin sovelluksen pitäisi saavuttaa sovellustietoturvan perustaso. VAHTI Sovelluskehityksen tietoturvaohjeen leimat ja ISBN-numerot täsmäävät verkkosivuilla annettuun ISBN-numeroon. Tämä todentaa lähteen aitouden. OWASP Top 10 on Open Web Application Security Project -tahon julkaisema ja ylläpitämä keskeisten tietoturvaa lisäävien ominaisuuksien lista ja ohjeistus uhkien eliminoimiseksi. Vapaan, sovellustietoturvaprojektin taustalla on aatteellinen maailmanlaajuinen organisaatio, nimeltään IRS.

OWASP-projektiin saa osallistua kuka tahansa, ja sillä on useita yhteistyökumppaneita ympäri maailmaa.

Tässä opinnäytetyöprojektissa sovelluksen kehittämiseen valitut menetelmät sekä työkalut olivat hyviä, sillä niitä käytetään tämän päiväisissä sovelluksissa ja niiden kehitystyössä. Menetelmien osalta Scrumia ei voitu toteuttaa täydellisesti, sillä sen käyttäminen vaatisi kehitystiimin. Sovellus kehitettiin toki Scrumin mukaisesti iteraatioissa, mutta jättäen Scrumin hallinnallisen osuuden suurimmaksi osaksi pois.

Projektinhallinnan näkökulmasta tämä projekti on ollut pelkkää oppimista. Projektin alkuvaiheessa tekijällä ei ollut varsinaisesti käsitystä tämmöisen projektin vetämisestä. Kuitenkin jo ensimmäisten kahden iteraation jälkeen alkoi kuva muodostua. Nyt tekijällä on jo kohtalainen käsitys tällaisen projektin läpi vetämisestä. Projektin hallintaan on kuulunut aloitus-, ohjaus- ja päättökokousten lisäksi demotilaisuuksia tilaajan edustajille eli yhdistyksen johtokunnan jäsenille. Projektissa oli vaatimusmäärittelyä, suunnittelua ja backlogien laatimista Scrumin mukaisesti, unohtamatta koodausta ja testausta ja kaiken toistamista iteraatio iteraatiolta.

6 Yhteenveto, johtopäätöksiä ja jatkokehitysehdotuksia

Opinnäytetyöprojekti oli sopivan haastava, kaiken kaikkiaan mielenkiintoinen ja erittäin hyvä oppimiskokemus. Siinä tuli vastaan niin uutta asiaa kuin vahvistui vanhaakin. Projektia työstäessäni oli aluksi vaikea pysyä aikataulussa, mutta aikataulun ja iteraatioiden keston muuttamisen jälkeen pysyin aikataulussa jopa suunniteltua paremmin.

Sovellus toteutettiin Spring-sovelluskehyksen avulla jo opittuja asioita hyödyntäen. Tietoturvallinen näkökulma oli haaste. Sen oivaltaminen tiesi tutkimista, lukemista, valintaa ja ratkaisujen etsimistä sovelluksen kehittämiseen ja itse sovellukseen. Tietoturvaosuuden toteuttaminen tuotti eniten töitä sovelluksen koko kehityksen aikana. Sovellukseen tehtiin muutoksia, kuten on ymmärrettävää, koko sen kehityskaaren aikana. Täysin uutta oli toteuttaa listamaisten rakenteiden päivittäminen: tiedon lisääminen ja poistaminen ja vieminen tietokantaan. Tämäkin oli pienen tiedonhaun takana, ja loppujen lopuksi erittäin helposti toteuttavissa. Muuten sovelluksen kehittäminen oli jo aikaisemman osaamisen uudelleen käyttämistä ja opitun parantamista.

Kehitystyössä käytettiin pitkälti tekijälle entuudestaan tuttuja tekniikoita ja teknologioita. Näitä olivat MySQL, Java, Tomcat, Acpache, ja jQuery. Vähemmän tuttuja teknologioita olivat Twitter Bootstrap CSS -kehys ja Gimp-kuvankäsittelyohjelma. Kehityksessä oli mukana myös Font Awesome -ikonikirjasto ja Git-versiohallitajärjestelmä. Tekijän osaaminen on karttunut merkittävästi näissä kaikissa teknologioissa, joita on käytetty tämän projektin sovelluskehityksessä tai sen apuna.

Merkittävin kokemus tässä opinnäytetyöprojektissa oli se, miten nopeasti kaikki saatiin loppujen lopuksi toteutettua. Oli mahdollista alittaa aikataulu reilusti, vaikka kaikki tuntui aluksi mahdottomalta ja työ isolta, vaikka se nyt tuntuu vähäiseltä. Käännekohta oli joulu ja vuodenvaihde. Silloin pääsin tekemään opinnäytetyötäni keskittyen pääsääntöisesti vain siihen.

Toisena erittäin merkittävänä asiana voidaan pitää projektinhallintataitoja, jotka olivat lähes olemattomat ennen opinnäytetyön aloittamista. En ollut aikaisemmin ottanut projektipäällikön tehtäviä, vaan olin valinnut ohjelmistoarkkitehdin tehtäviä projekteissa.

Projektinhallintataidot ovatkin vahvistuneet valtavasti tämän opinnäytetyön aikana. Nyt minulla on käsitys projektin vetämisestä ja onnistuvan projektin vaatimuksista. Edelleen on varmasti paljon asioita, joista ei ole kokemusta tai edes tietoa, ja ne kaipaavat lisää kokemuksia: opettelemista, tutkimista ja käyttöön ottamista niin projektinhallinnassa kuin sovelluksen kehittämisessäkin. Kuitenkin tämän kokemuksen myötä minulla on rohkeutta tarttua seuraavaan projektiin.

6.1 Johtopäätöksiä

Tietoturva on mahdollista toteuttaa verkkosovellukseen, kunhan se otetaan alusta asti huomioon. Tietoturvavaatimukset tulee määritellä siinä missä muutkin toiminnalliset vaatimukset ja suoriutumisvaatimukset. Tietoturvallisuuden toteuttaminen ei ole sen monimutkaisempaa kuin muidenkaan asioiden. Se ei vaadi ihmeitä, vaan hyvien käytäntöjen seuraamista.

Tietoturva on laatua ja käytettävyyttä. Se on käyttäjän huomioon ottamista niin uhkien kuin niiltä suojautumisen kannalta. Tilaajan sivuille toteutettiin perustietoturvatasoon vaatimusten mukaiset tietoturvaa parantavat tekijät.

Koska sovellus sisältää turvallisen toteutuksen OWASP Top 10 -listan uhkista, joilla on merkitystä sovelluksen kannalta. On sovellus vaatimusten mukaisesti tietoturvallinen. Toteutettu sovellus on tietoturvallisempi kuin osa julkisista verkkosovelluksista.

6.2 Jatkokehitysehdotuksia

Sovelluksen toiminnallisuuksia rajattiin tämän projektin alussa. Rajatut ominaisuudet on syytä toteuttaa, samoin tuotantoon siirtotoimet.

Tietoturvaa voi edelleen lisätä. Sovellukseen voitaisiin implementoida nykyisen autentikointimenetelmän lisäksi varmistusautentikointi. Tämä voisi olla kertakäyttöinen pinkoodiautentikointi samaan tapaan kuin pankeissa. Tämä on vielä tänäkin päivänä murtamaton ja toisi äärimmäistä turvallisuutta sovellukselle. Käyttäjämäärän kasvaessa voi käyttäjien arvaaminen tulla helpommaksi, mikä voi vaarantaa käyttäjätilin. Nykyisen ratkaisun tapa antaa oikeuden tietojen perusteella näkyviin kysymyksen, mikä helpottaa

vastauksen arvaamista. Se voi johtaa oikean käyttäjän paljastumiseen, mutta ei vielä vastauksen autentikointia varten. Oikein käytettynä tämä autentikointi on turvallinen, mutta käyttäminen on käyttäjäkohtaista. Tietysti antamalla ohjeita oikeanlaisen kysymys-vastaus -parin luomiseen voidaan asiaan vaikuttaa.

Toisena jatkokehitysideana ovat yhdistyksen jäsenten tarinat kuvineen. Se jätettiin tietoisesti pois tästä sovelluksen versiosta. Syynä oli tietoturvallisuuden tärkeys sovelluksessa. Lisäksi ajatusta ei ole vielä mietitty tarkemmin. Tarkoituksena on kerätä tarinoita vuosien varrella tapahtuneista asioista ja tarjota mahdollisuus niiden selailuun, jopa kommentointiin, oikeuksien salliessa.

Kolmas jatkokehitysehdotus liittyy jäsenhallinnan laajentamiseen. Siinä seurataan jäsenmaksukertymää, jäsenyyttä ja muistutetaan jäsenmaksusta.

Sovelluksen tietoturvaa voitaisiin parantaa myös estämällä robottirekisteröityminen. Tämä voitaisiin toteuttaa lähettämällä rekisteröityneelle käyttäjälle tunnuksen aktivoiva sähköpostiviesti, jossa on linkki. Linkkiä klikkaamalla tunnus aktivoidaan. Tai sovellukseen voisi toteuttaa kuvavarmennus. On mahdollista, että sovellukseen laitetaan piilokenttä, minkä avulla voidaan myös estää robottien rekisteröinti.

Pienenä lisänä tapahtuman markkinoinnin yhteyteen voisi liittää karttanäkymän. Tämä voisi selkeyttää tapahtuman sijaintia, ilman että käyttäjän tarvitsee mennä erilliseen karttasovellukseen tarkistamaan sijainti.

Lähteet

VATHI 2013

Valtiovarainministeriö 2013 Sovelluskehityksen tietoturvaohje, VAHTI 1/2013 Luettavissa:
http://www.vm.fi/vm/fi/04_julkaisut_ja_asiakirjat/01_julkaisut/05_valtionhallinnon_tietoturvallisuus/20130207Sovell/VAHTI_1_Sovelluskehityksen_tietoturvaohje_NETTI.pdf Luettu: 26.12.2013

OWASP A1

Owasp Foundation 2013 Top 10 A1 Injection Luettavissa:
https://www.owasp.org/index.php/Top_10_2013-A1-Injection Luettu: 30.12.2013

OWASP A2

Owasp Foundation 2014 Top 10 A2 Broken Authentication and Session Management Luettavissa: https://www.owasp.org/index.php/Top_10_2013-A2-Broken_Authentication_and_Session_Management Luettu: 30.12.2013

OWASP A3

Owasp Foundation 2013 Top 10 2013 A3 Cross-Site Scripting (XSS) Luettavissa:
https://www.owasp.org/index.php/Top_10_2013-A3-Cross-Site_Scripting_(XSS) Luettu: 30.12.2013

OWASP A6

Owasp Foundation 2013 Top 10 A6 Sensitive Data Exposure Luettavissa:
https://www.owasp.org/index.php/Top_10_2013-A6-Sensitive_Data_Exposure Luettu: 30.12.2013

OWASP A7

Owasp Foundation 2013 Top 10 2013 A7 Missing Function Level Access Control Luettavissa: https://www.owasp.org/index.php/Top_10_2013-A7-Missing_Function_Level_Access_Control Luettu: 30.12.2013

OWASP A8

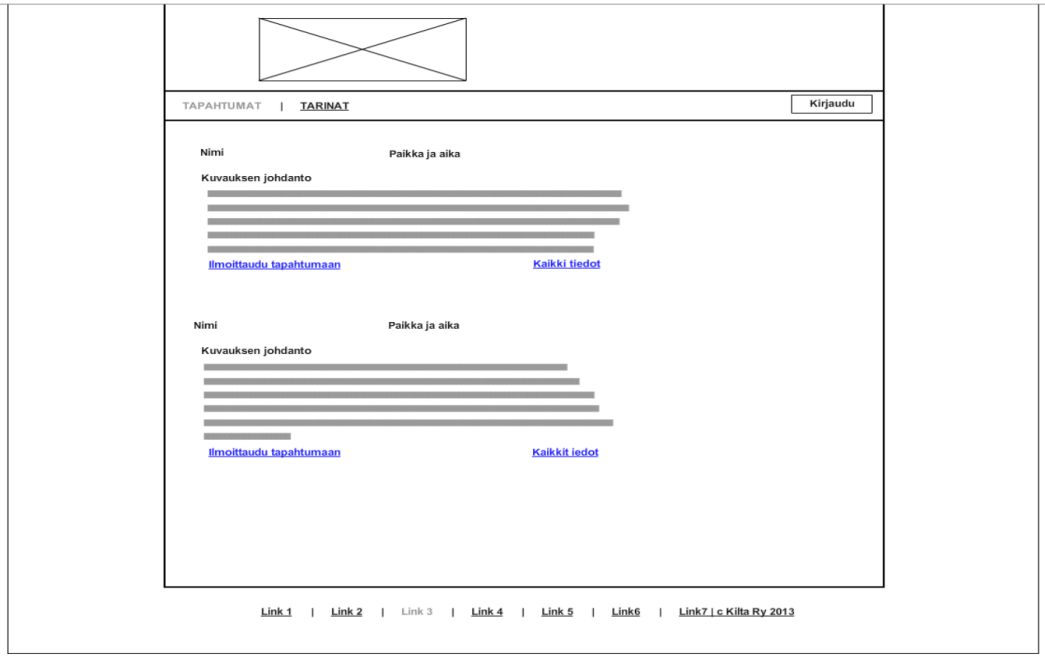
Owasp Foundation A8 Cross-Site Request Forgery (CSRF) Luettaviss: https://www.owasp.org/index.php/Top_10_2013-A8-Cross-Site_Request_Forgery_(CSRF) Luettu: 30.12.2013

OWASP A10

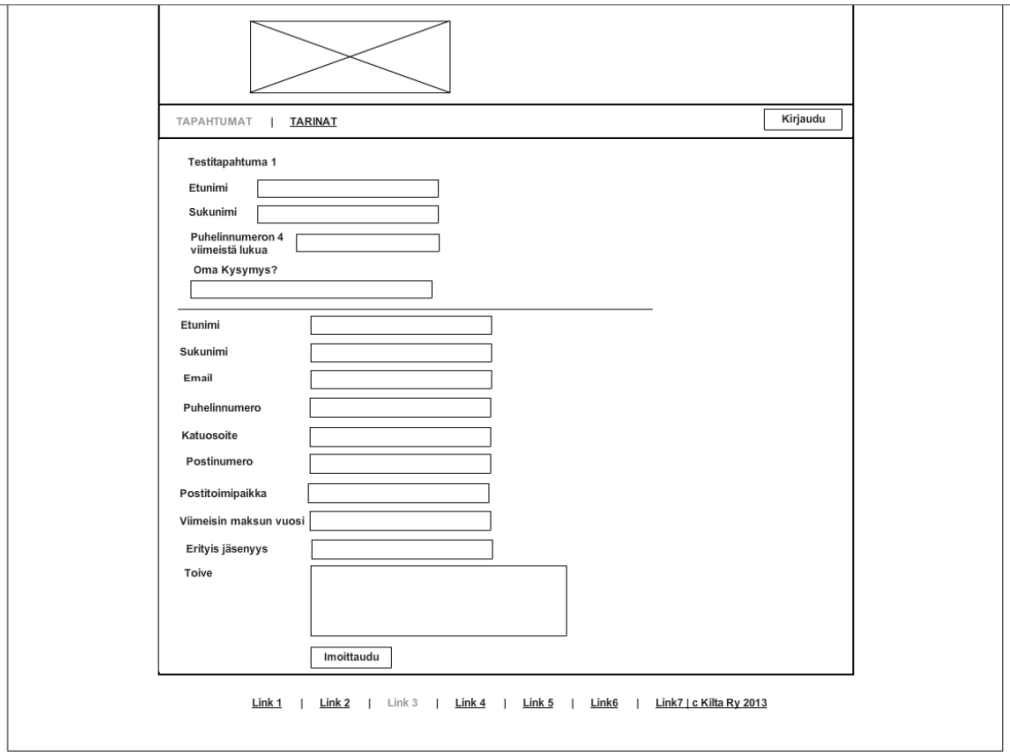
Owasp Foundation A10 Unvalidated Redirects and Forwards Luettavissa: https://www.owasp.org/index.php/Top_10_2013-A10-Unvalidated_Redirects_and_Forwards Luettu: 30.12.2013

Liitteet

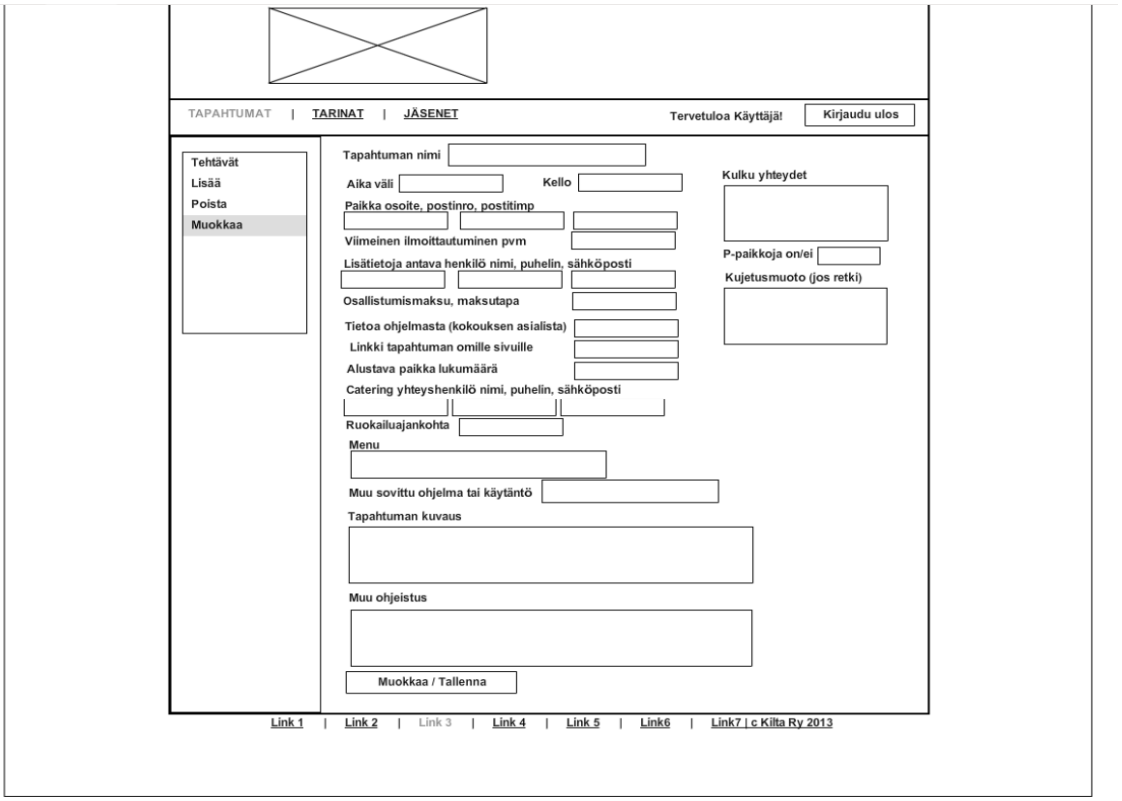
Liite 1 Mock-up -kuvat



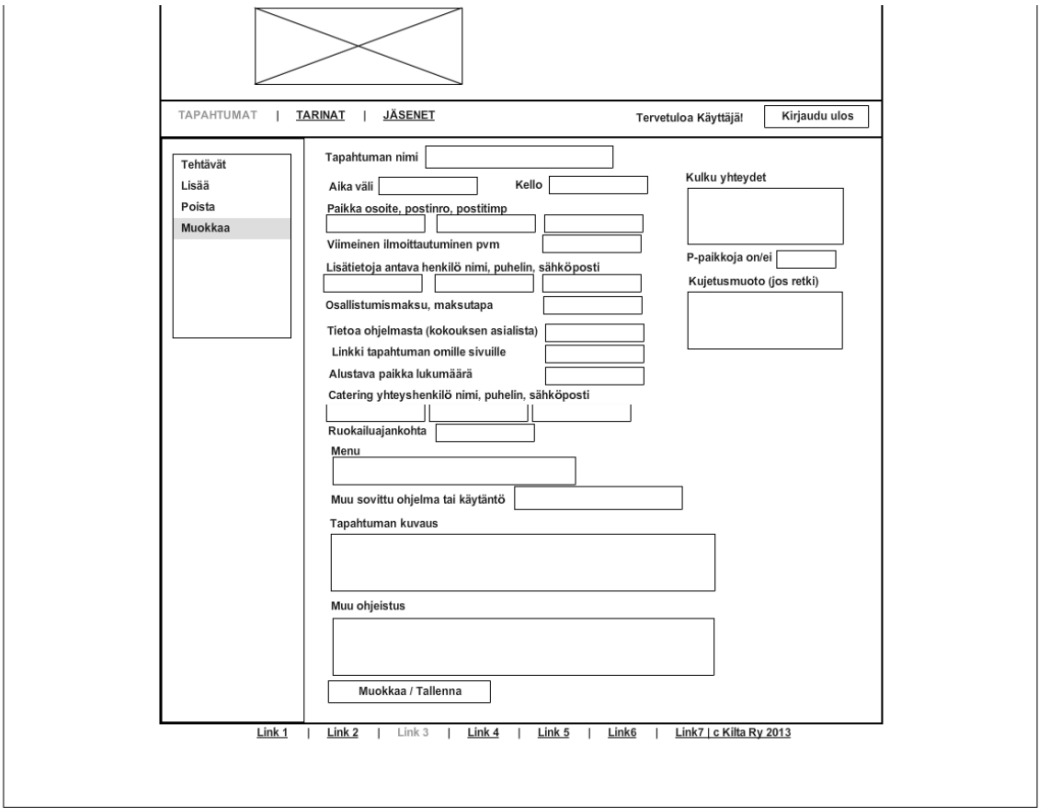
Kuva 1.1 Tapahtumat etusivu



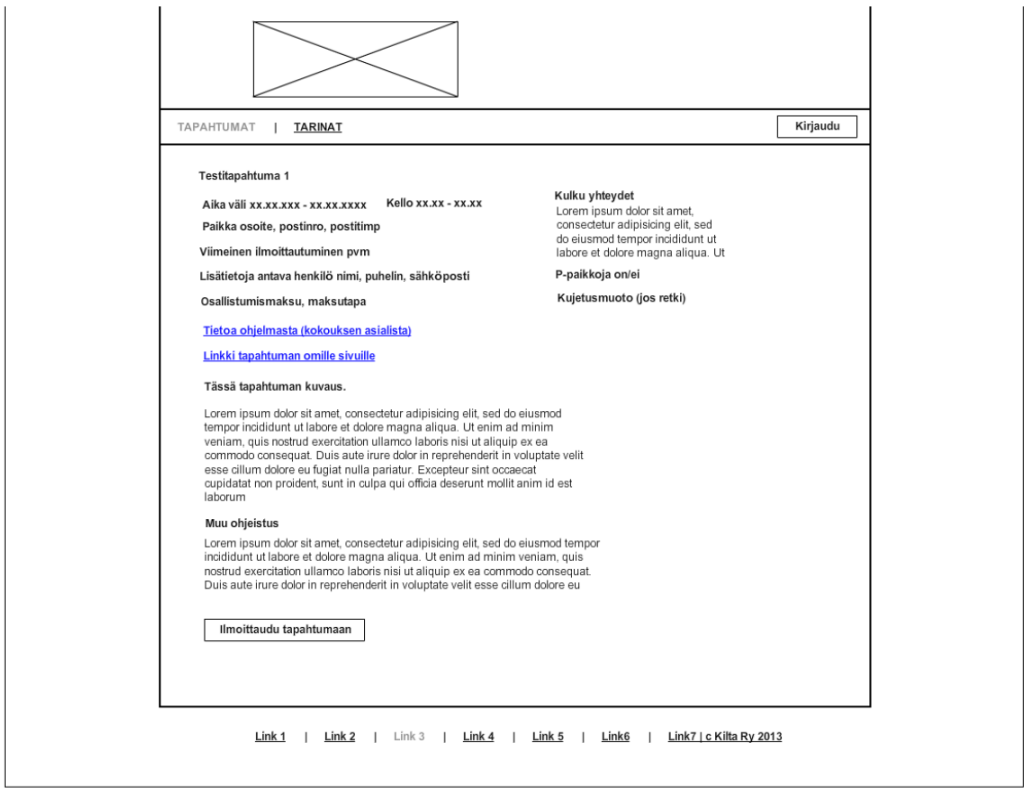
Kuva 1.2 Tapahtumaan ilmoittautuminen



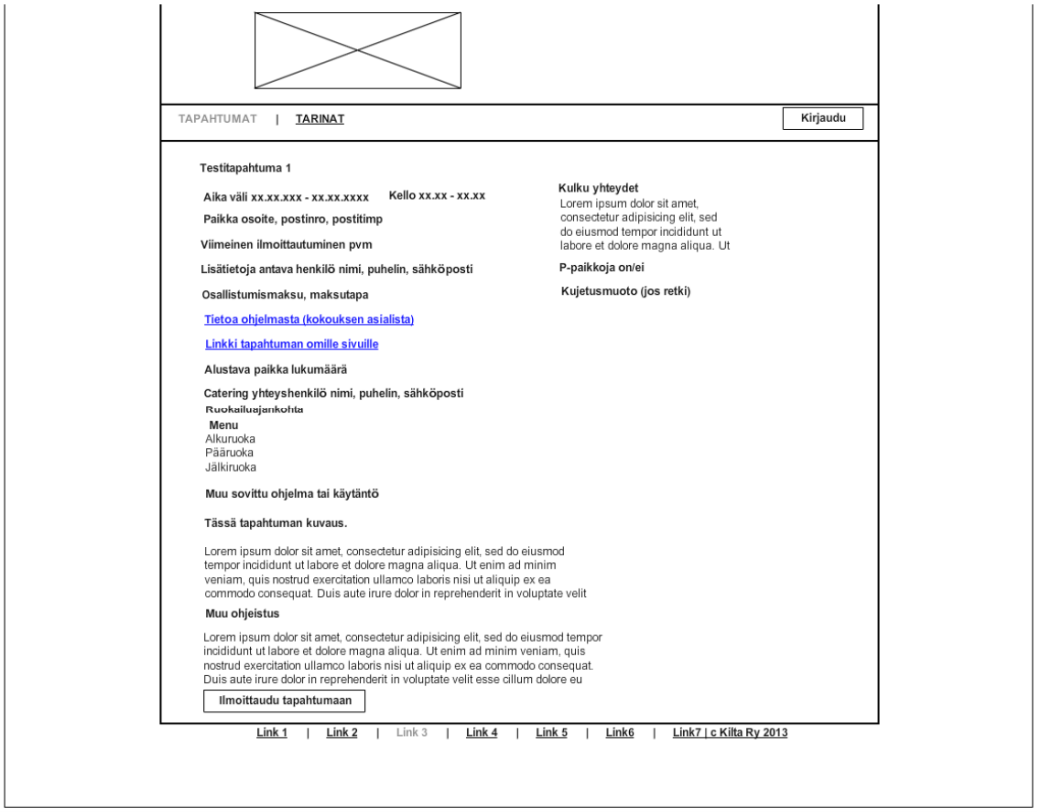
Kuva 1.3 Tapahtuman manuaalisesti arkistoiminen



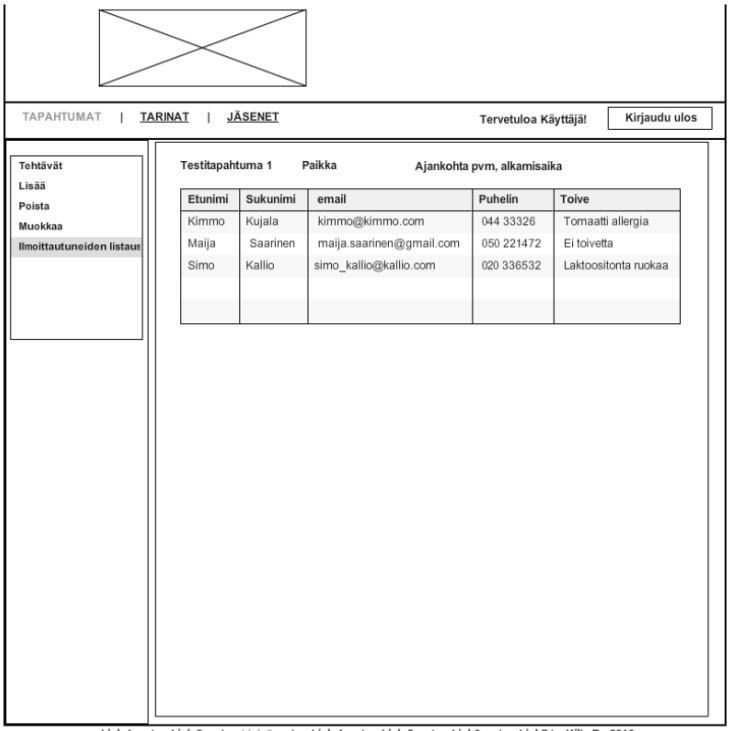
Kuva 1.4 Tapahtuman lisääminen



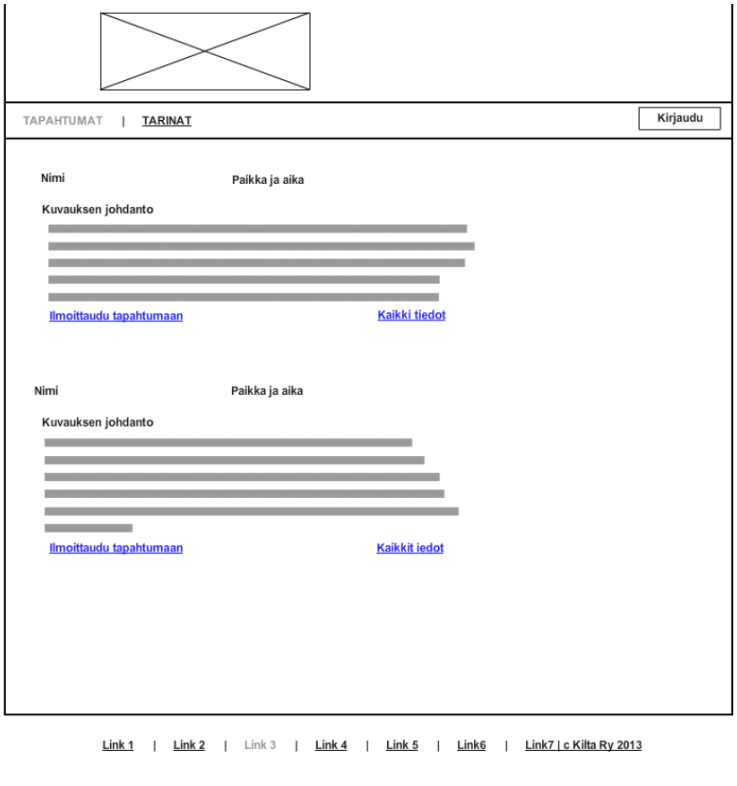
Kuva 1.5 Tapahtuman lisätiedot



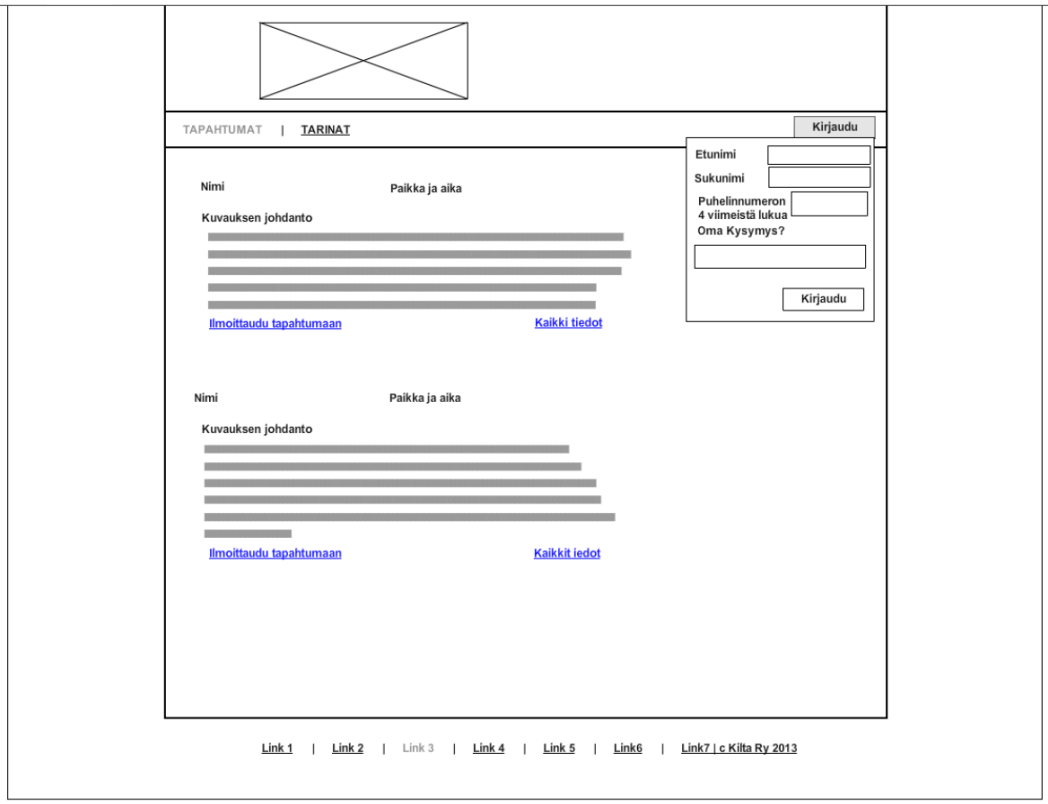
Kuva 1.6 Tapahtuman lisätiedot järjestelmänvalvojaoikeuksilla



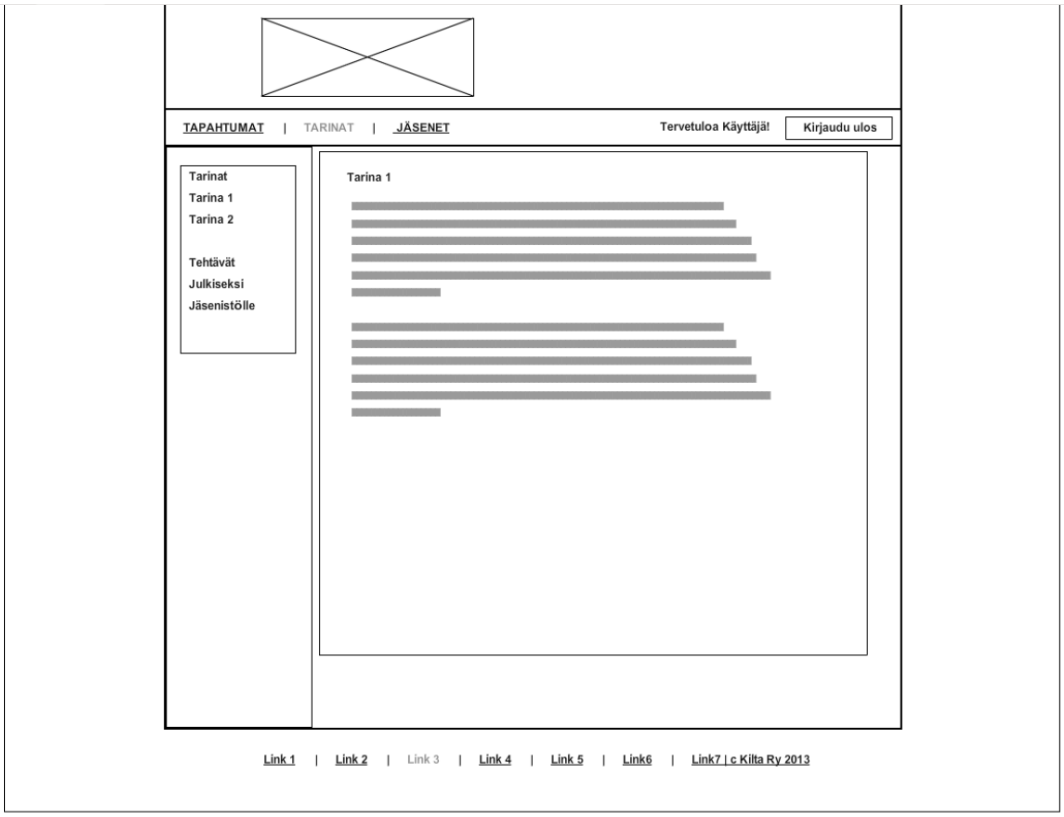
Kuva 1.7 Tapahtumaan ilmoittautuneiden listaus



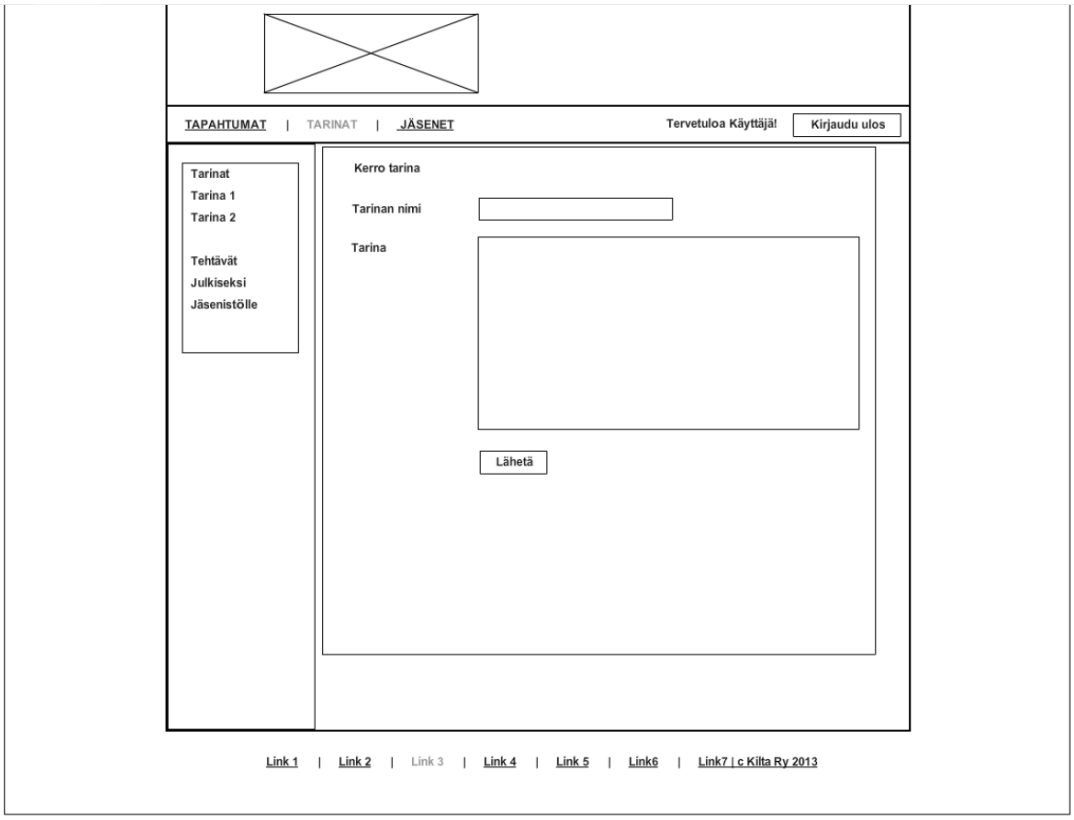
Kuva 2.1 Sovelluksen etusivu



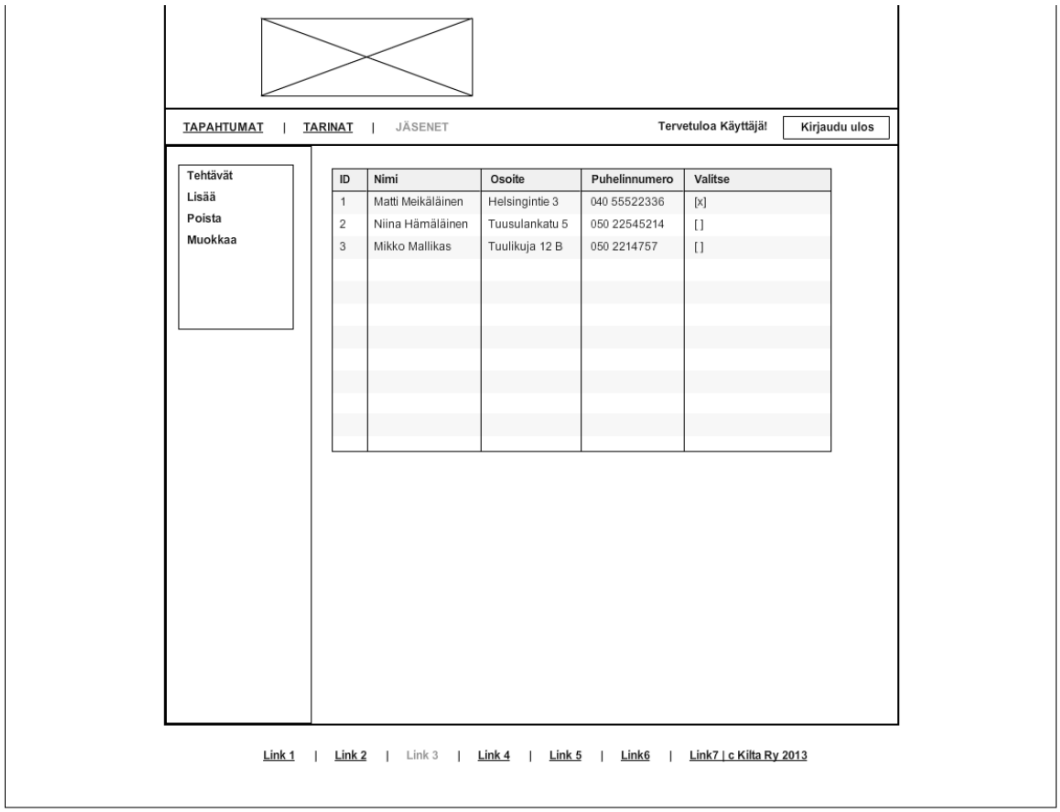
Kuva 2.2 Sovelluksen etusivu pikakirjautuminen auki



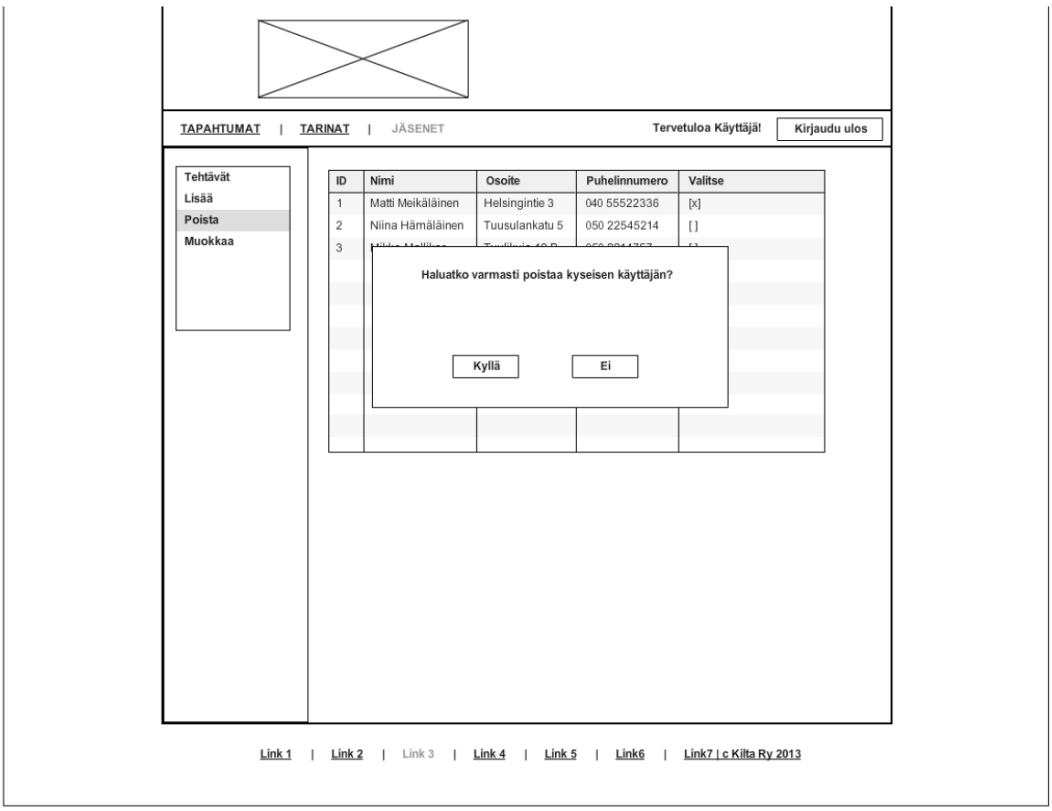
Kuva 3.1 Tarinat etusivu



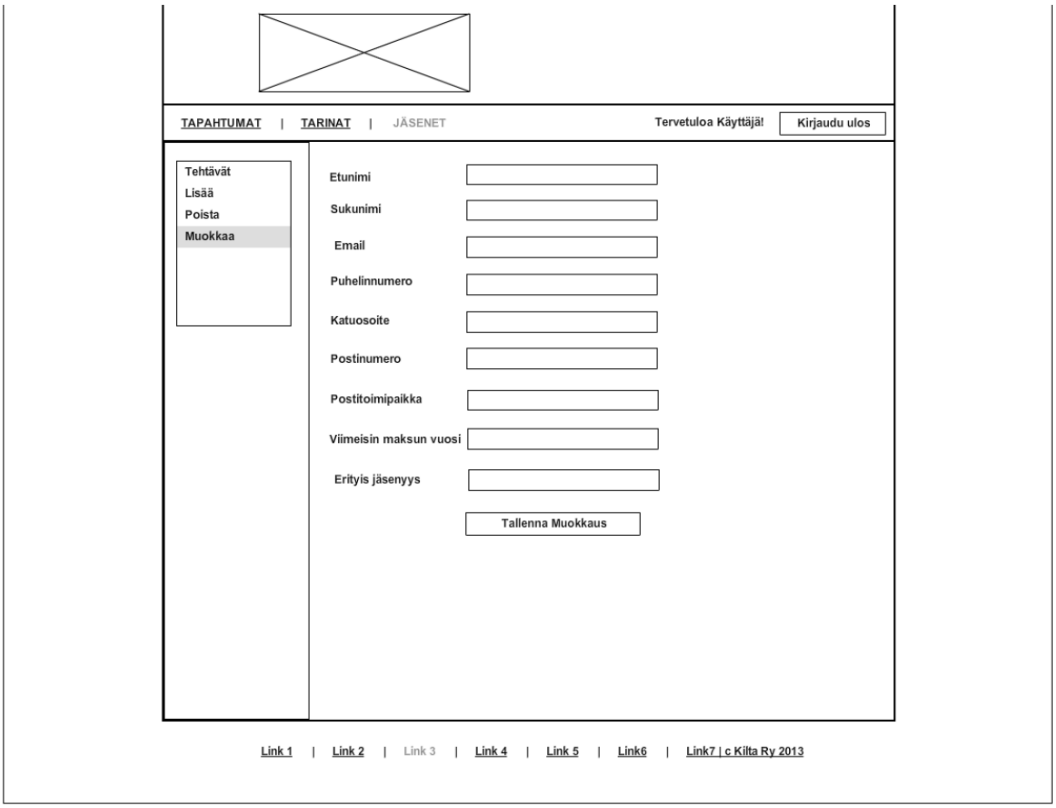
Kuva 3.2 Tarinoiden lisääminen



Kuva 4.1 Jäsenien listaus

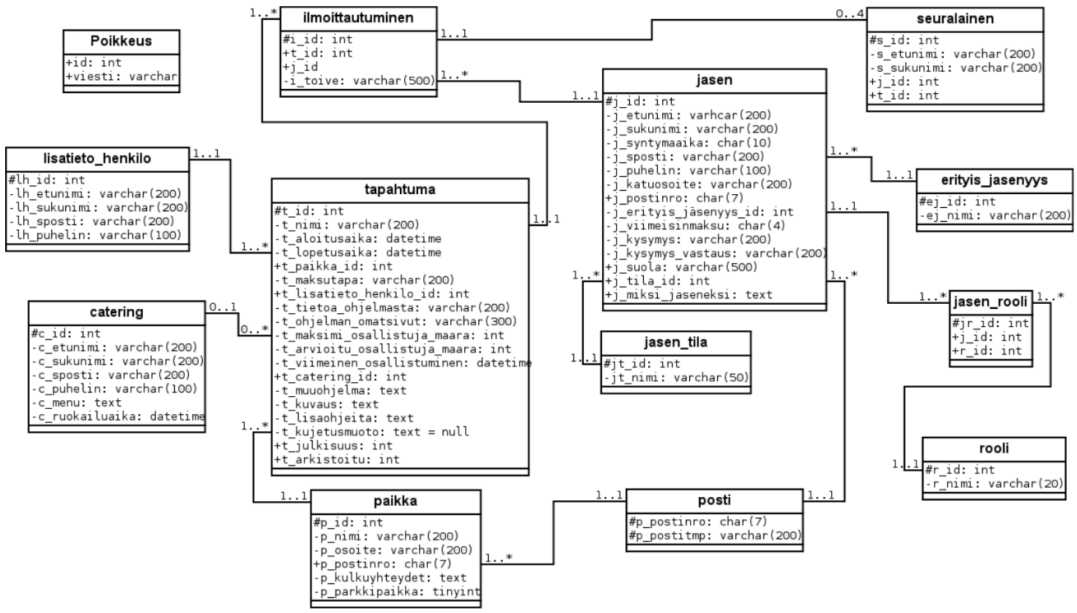


Kuva 4.2 Jäsenien deaktivoiminen



Kuva 4.3 Jäsentietojen muuttaminen sekä lisääminen

Liite 2 Tietokannan suunnittelu ja toteutus



Kuva 5.1 Tietokannan relaatiokaavio

Tietokannan luontilauseet

```
CREATE DATABASE kilta_db2 DEFAULT CHARACTER SET utf8;
USE kilta_db2;

CREATE TABLE jasen_tila (
jt_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
jt_nimi VARCHAR(200)
)ENGINE InnoDb;

CREATE TABLE erityis_jasenyys (
ej_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
ej_nimi VARCHAR(200)
)ENGINE InnoDb;

CREATE TABLE rooli (
r_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
r_nimi VARCHAR(20)
```

```
)ENGINE InnoDb;

CREATE TABLE posti (
p_postinro CHAR(7),
p_postitoimipaikka VARCHAR(200),
PRIMARY KEY (p_postinro, p_postitoimipaikka)
)ENGINE InnoDb;

CREATE TABLE  jasen (
j_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
j_etunimi VARCHAR(200) NOT NULL,
j_sukunimi VARCHAR(200) NOT NULL,
j_syntymaaika DATETIME NOT NULL,
j_sposti VARCHAR(200) NOT NULL,
j_puhelin VARCHAR(200) NOT NULL,
j_katuosoite VARCHAR(200) NOT NULL,
j_postinro CHAR(7) NOT NULL,
j_erityis_jasenyys_id INT,
j_viimeisinmaksu CHAR(4),
j_kysymys VARCHAR(200) NOT NULL,
j_kysymys_vastaus VARCHAR(200) NOT NULL,
j_suola VARCHAR(500),
j_tila_id INT,
j_miksi_jaseneksi TEXT,
INDEX(j_etunimi, j_sukunimi, j_syntymaaika),
FOREIGN KEY (j_tila_id) REFERENCES jasen_tila (jt_id),
FOREIGN KEY (j_erityis_jasenyys_id) REFERENCES erityis_jasenyys (ej_id),
FOREIGN KEY (j_postinro) REFERENCES posti (p_postinro)
)ENGINE InnoDb;

CREATE TABLE jasen_rooli (
jr_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
```

```
j_id INT,
r_id INT,
FOREIGN KEY (j_id) REFERENCES jasen (j_id),
FOREIGN KEY (r_id) REFERENCES rooli (r_id)
)ENGINE InnoDb;

CREATE TABLE paikka (
p_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
p_nimi VARCHAR(200),
p_osoite VARCHAR(200),
p_postinro CHAR(7),
p_kulkuyhteydet TEXT,
p_parkkipaikka TINYINT,
FOREIGN KEY (p_postinro) REFERENCES posti (p_postinro)
)ENGINE InnoDb;

CREATE TABLE lisatieto_henkilo (
lh_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
lh_etunimi VARCHAR(200),
lh_sukunimi VARCHAR(200),
lh_sposti VARCHAR(200),
lh_puhelin VARCHAR(200)
)ENGINE InnoDb;

CREATE TABLE catering (
c_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
c_etunimi VARCHAR(200),
c_sukunimi VARCHAR(200),
c_sposti VARCHAR(200),
c_puhelin VARCHAR(200),
c_menu TEXT,
c_ruokailuaika DATETIME
```

```
)ENGINE InnoDb;

CREATE TABLE tapahtuma (
t_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
t_nimi VARCHAR(200),
t_aloitusaika DATETIME,
t_lopetusaika DATETIME,
t_paikka_id INT,
t_maksutapa VARCHAR(200),
t_lisatieto_henkilo_id INT,
t_tietoa_ohjelmasta VARCHAR(200),
t_ohjelman_omatsivut VARCHAR(200),
t_maksimi_osallistujat INT,
t_arvioitu_osallistuja_maara INT,
t_viimeinen_osallistuminen DATETIME,
t_catering_id INT,
t_muuohjelma TEXT,
t_kuvaus TEXT,
t_lisaohjeita TEXT,
t_kuljetusmuoto TEXT,
t_julkisuus TINYINT,
t_arkistoitu TINYINT,
INDEX(t_julkisuus),
INDEX(t_lopetusaika, t_arkistoitu),
INDEX(t_lopetusaika, t_julkisuus, t_arkistoitu),
INDEX(t_id, t_julkisuus),
FOREIGN KEY (t_paikka_id) REFERENCES paikka (p_id),
FOREIGN KEY (t_lisatieto_henkilo_id) REFERENCES lisatieto_henkilo (lh_id),
FOREIGN KEY (t_catering_id) REFERENCES catering (c_id)
)ENGINE InnoDb;

CREATE TABLE ilmoittautuminen (
```

```
i_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
t_id INT,
j_id INT,
i_toive VARCHAR(500),
INDEX(t_id, j_id),
FOREIGN KEY (t_id) REFERENCES tapahtuma (t_id),
FOREIGN KEY (j_id) REFERENCES jasen (j_id)
)ENGINE InnoDb;

CREATE TABLE seuralainen (
s_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
s_etunimi VARCHAR(200) NOT NULL,
s_sukunimi VARCHAR(200) NOT NULL,
j_id INT,
t_id INT,
INDEX(j_id, t_id),
FOREIGN KEY (j_id) REFERENCES ilmoittautuminen (j_id),
FOREIGN KEY (t_id) REFERENCES ilmoittautuminen (t_id)
)ENGINE InnoDb;

CREATE TABLE poikkeus (
id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
viesti VARCHAR(200)
)ENGINE InnoDb;
```

Tietokannan testidata

```
INSERT INTO jasen_tila (jt_nimi) VALUES ("vahvistettu");
INSERT INTO jasen_tila (jt_nimi) VALUES ("vahvistamaton");

INSERT INTO erityis_jasenyys (ej_nimi) VALUES ("Ei määritetty");
INSERT INTO erityis_jasenyys (ej_nimi) VALUES ("Kokki");
INSERT INTO erityis_jasenyys (ej_nimi) VALUES ("Siivjooja");
```

```
INSERT INTO erityis_jasenyys (ej_nimi) VALUES ("Puheenjohtaja");
INSERT INTO erityis_jasenyys (ej_nimi) VALUES ("Pormestari");
INSERT INTO erityis_jasenyys (ej_nimi) VALUES ("Tilahoitaja");
INSERT INTO erityis_jasenyys (ej_nimi) VALUES ("Kirjanpitäjä");
INSERT INTO erityis_jasenyys (ej_nimi) VALUES ("Autonpesijä");
INSERT INTO erityis_jasenyys (ej_nimi) VALUES ("Kunniajäsen");

INSERT INTO rooli (r_nimi) VALUES ("ROLE_ADMIN");
INSERT INTO rooli (r_nimi) VALUES ("ROLE_MEMBER");

INSERT INTO posti (p_postinro, p_postitoimipaikka) VALUES ("00200", "Helsin-
ki");
INSERT INTO posti (p_postinro, p_postitoimipaikka) VALUES ("03750", "Vantaa");

INSERT INTO jasen (j_etunimi, j_sukunimi, j_syntymaaika, j_sposti, j_puhelin,
j_katuosoite, j_postinro,
j_erityis_jasenyys_id, j_viimeisinmaksu, j_kysymys, j_kysymys_vastaus, j_suola,
j_tila_id) VALUES ("Matti",
"Siikala", "1970-05-20", "matti.siikala@google.com", "0458855874", "hiidenkuja 3b",
"00200", 1,
"2013", "Mikä on Maijan koiran ni-
mi?","b9e6f0e1a5bb74a1e9a540854ac6dd860dd816d978d7283cfad2edbc55e922bf",
"ll", 1);

INSERT INTO jasen (j_etunimi, j_sukunimi, j_syntymaaika, j_sposti, j_puhelin,
j_katuosoite, j_postinro,
j_erityis_jasenyys_id, j_viimeisinmaksu, j_kysymys, j_kysymys_vastaus, j_suola,
j_tila_id, j_miksi_jaseneksi) VALUES ("Minna",
"Kaikkonen", "1980-06-17", "minna.kaikkonen@foo.com", "0503321221", "Helsingin-
tie 12", "00200", 1,
```

"2012", "Moi
vaan","8765afd77950a3c0cb7194ff7cec6107fedb88d495ac29bba4f71c138b4aea94",
"kk", 2, "Olen innostunut uusikaupunkilainen");

INSERT INTO jasen_rooli (j_id, r_id) VALUES (1 ,1);
INSERT INTO jasen_rooli (j_id, r_id) VALUES (2 ,2);

INSERT INTO lisatieto_henkilo (lh_etunimi, lh_sukunimi, lh_sposti, lh_puhelin)
VALUES ("Kaisa",
"Kallio", "kaisa.kallio@msn.com", "0206332789");

INSERT INTO catering (c_etunimi, c_sukunimi, c_sposti, c_puhelin, c_menu,
c_ruokailuaika) VALUES ("Natalia",
"Kraz", "natalia.kraz@catering.com", "0408887875", "Alkuruoka: sinappinen kalkkuna file\n\nPääruoka: Hummeri hunajakastikkeessa\n\nJälkiruoka: Kinuskinen omenapii-rakka",
"2013-11-20 16:00:00");

INSERT INTO paikka (p_nimi, p_osoite, p_postinro, p_kulkuyhteydet,
p_parkkipaikka) VALUES ("Seurasaari",
"seurasaarenkatu 15", "03750", "Bussi lähtee Helsingin rautatieasemalta.", 1);

INSERT INTO tapahtuma (t_nimi, t_aloitusaika, t_lopetusaika, t_paikka_id,
t_maksutapa, t_lisatieto_henkilo_id,
t_tietoa_ohjelmasta, t_ohjelman_omatsivut, t_maksimi_osallistujat,
t_arvioitu_osallistuja_maara,
t_viimeinen_osallistuminen, t_catering_id, t_muuohjelma, t_kuvaus, t_lisaohjeita,
t_kuljetusmuoto,
t_julkisuus, t_arkistoitu)
VALUES ("Retki seurasaareen", "2013-11-20 14:00:00", "2013-11-20 19:00:00", 1,
"Paikanpäällä käteinen",

1, "Linkki ohjelma tieto isku pdf:ään", "Linkki tapahtuman omille sivuille", 30, 20, "2013-11-15 18:00:00",
1, "Saunomista luvassa kierroksen jälkeen. ", "Retki seurasaaren kauniiseen ympäristöön sekä grillausta paikanpäällä",
"Laita lämmintä päälle", "Bussi", 1, 0);

INSERT INTO ilmoittautuminen (t_id, j_id, i_toive) VALUES (1, 1, "Haluan istua Matin vieressä");

INSERT INTO poikkeus (viesti) VALUES ("Haettua kohdetta ei löytynyt, ota yhteys ylläpitoon");

Tietokannan alustuslauseet

INSERT INTO jasen_tila (jt_nimi) VALUES ("vahvistettu");
INSERT INTO jasen_tila (jt_nimi) VALUES ("vahvistamaton");

INSERT INTO erityis_jasenyys (ej_nimi) VALUES ("Ei määritetty");
INSERT INTO erityis_jasenyys (ej_nimi) VALUES ("Olderman");
INSERT INTO erityis_jasenyys (ej_nimi) VALUES ("Kriivar");
INSERT INTO erityis_jasenyys (ej_nimi) VALUES ("Räntmestar");
INSERT INTO erityis_jasenyys (ej_nimi) VALUES ("Kiltmestar");
INSERT INTO erityis_jasenyys (ej_nimi) VALUES ("Vakkamestar");
INSERT INTO erityis_jasenyys (ej_nimi) VALUES ("Emänt");
INSERT INTO erityis_jasenyys (ej_nimi) VALUES ("Huvitoimikunnan jäsen");
INSERT INTO erityis_jasenyys (ej_nimi) VALUES ("Kunniajäsen");
INSERT INTO rooli (r_nimi) VALUES ("ROLE_ADMIN");
INSERT INTO rooli (r_nimi) VALUES ("ROLE_MEMBER");

INSERT INTO posti (p_postinro, p_postitoimipaikka) VALUES ("00100", "Helsinki");

```
INSERT INTO jasen (j_etunimi, j_sukunimi, j_syntymaaika, j_sposti, j_puhelin,
j_katuosoite, j_postinro,
j_erityis_jasenyys_id, j_viimeisinmaksu, j_kysymys, j_kysymys_vastaus, j_suola,
j_tila_id) VALUES ("Kilta",
"Ylläpito", "1970-05-20", "kilta.yllapito@kilta.fi", "040000000", "helsingintie 12",
"00100", 1,
"2014", "Mikä on kil-
ta?","3d889766e5554f11fd9fc6762759aa85d7bbcc5d1f399783d4e02ef79ffbd16f",
"HyuanClCe1qDuLS1m9A78ROvnbxbBUG", 1);

INSERT INTO jasen_rooli (j_id, r_id) VALUES (1 ,1);
```

Liite 3 Asennus- ja kovennusdokumentti

Tekniset vaatimukset

Taulukko 1. Tekniset vaatimukset sovellukselle ja version joiden alla sovellus on toteutettu.

Asia	Versio
Java	1.7.0_45
Tomcat	7.0.35
Apache	2.2.22
Git	1.8.1.4
MySQL	5.5.33
Twitter Bootstrap	2.3.2
Font Awesome	4.0.3
Spring Framework	3.2.4.RELEASE
Spring Security	3.1.4.RELEASE

Sovelluksen asentaminen Linux käyttöjärjestelmään

- Asenna Java, Tomcat, Apache, MySQL palvelut käyttöjärjestelmään.
- Asenna Apache2 palvelimelle lisäosat ModJk ja SSL, mikäli eivät ole asennettuna.
- Muokkaa hosts-tiedostoa hakemistossa etc/hosts. Lisää sinne rivi, jossa on ulkoverkon IP osoite ja URL osoite. Mikäli kyseessä ei ole ulkoverkkoon näkyvä sovellus, niin lisää rivi jossa jossa on koneen IP osoite ja URL osoite. Jos sovellusta pyöritetään vain localhostin alla, ei hosts tiedostoon tarvitse lisätä riviäkään.
- Muokkaa Tomcatin server.xml tiedostoa lisäämällä sinne uusi host, mikäli kyseessä on tuotantoversio.
- Kopioi kilta_development.war ja halutessasi kilta_production.war Tomcat palvelimelle webapps kansioon. Tuotantoversio vaatii sertifikaation toimiakseen.
- Konfiguroi Apache2 Tomcat palvelimen eteen ja luo virhtualhost tiedosto(t). Konfiguroi MokJk Apahce2 palvelimelle ja muokkaa Tomcatin server.xml tiedostoa ja poista kommentit AJP protokollaa koskevan rivin ympäriltä, jos ei ole valmiiksi tehty.

- Konfiguroi SSL Apache2 palvelimelle ja asenna sertifikaatit Apache2 palvelimelle.
- Suorita tietokannan luontilauseet. Katso liite 2.
- Tarvittaessa voit asettaa testidatan lauseet tietokantaan. Katso liite 2.
- Mikäli et suorittanut testidatalauseita voit suorittaa tietokannan alustuslauseet. Katso liite 2. (Käyttäjä: Kilta Ylläpito, syntymäaika: 20.05.1970, vastaus kysymykseen: Admin1234!) Muuta heti.

Asennuspaketin sisältö

- Mukana tulee kilta_development.war ja kilta_production.war, voit laittaa kummatkin palvelimelle tai vain toisen niistä.
- Mukana tulee malli sertifikaatti ja avain
- Mukana tulee malli server.xml
- Mukana tulee malli hosts-tiedosto
- Mukana tulee malli virtualhost-tiedosto

Huomioitavaa

Asentaessa tuotantoversiota julkiseen verkkoon, on syytä vaihtaa sertifikaatit oikeisiin.