

Vikatketöintiratkaisun suunnittelu ja toteutus

Tino Ikonen

Opinnäytetyö
12/2014

Ohjelmistotekniikka
Tekniikan ja liikenteen ala





Tekijä(t) Ikonen, Tino	Julkaisun laji Opinnäytetyö	Päivämäärä 12.12.2014
	Sivumäärä 32	Julkaisun kieli Suomi
		Verkojulkaisulupa myönnetty: x
Työn nimi Vikatiketöintiratkaisun suunnittelu ja toteutus		
Koulutusohjelma Ohjelmistotekniikan koulutusohjelma		
Työn ohjaaja(t) Matti Mieskolainen		
Toimeksiantaja(t) Versine Oy Kari Aho		
Tiivistelmä <p>Opinnäytetyön aiheena oli Versine Oy:ltä saatu toimeksianto suunnitella ja toteuttaa vikatiketien hallinnan mahdollistava työkalu selaimessa toimivaan sovellukseen. Toteutuksessa tuli ottaa huomioon mobiililaitteet ja käyttää moderneja mobiili- ja web-työkaluja. Lisäksi toteutus tuli integroida Versine Oy:n juuri kehitteillä olevaan projektiin.</p> <p>Opinnäytetyössä vertailtiin vikatiketöintityökaluja ja niiden pohjalta kerättiin kaikki oleelliset ominaisuudet vikatiketien hallintaan. Toteutus pohjautui toimeksiantajan toiveesta tutkittuihin vikatiketöintityökaluihin, mutta siinä on myös tehty omia linjauksia. Toteutuksen esittelyssä on perusteltu erikseen käyttöliittymän ja tietokannan puolella tehdyt päätökset. Opinnäytetyössä pohdittiin myös web-sovellusten testaamista ja sen vaikutusta web-sovellusten kehittämiseen.</p> <p>Toteutus on tehty JavaScriptillä toimimaan osana web-sovellusta, joka Versine Oy:llä oli kehitteillä. Toteutus on pyritty pitämään mahdollisimman yksinkertaisena ja siistinä. Toteutukselle jäi monia mahdollisuuksia jatkokehitykseen. Toteutus oli yksi monesta työkalusta yhden osion alla. Jatkossa on suunnitelmia laajentaa se yhdestä työkalusta omaksi osiokseen ja lisätä dokumentointiin liittyviä mahdollisuuksia.</p>		
Avainsanat (asiasanat) JavaScript, HTML, selain		
Muut tiedot		



Author(s) Ikonen, Tino	Type of publication Bachelor's thesis	Date 12.12.2014
		Language of publication: Finnish
	Number of pages 32	Permission for web publication: x
Title of publication Design and implementation of an issue ticket solution		
Degree programme Software Engineering		
Tutor(s) Mieskolainen, Matti		
Assigned by Aho, Kari Versine Ltd.		
Abstract <p>The aim of this thesis was to design and implement an issue ticket managing tool in a web-application. The thesis was assigned by Versine Ltd. The solution was to take into account mobile devices and use modern mobile and web development tools.</p> <p>Multiple issue ticket management systems were inspected and compared in order to find all the relevant pieces of information for managing issue tickets. According to the company's wishes, the solution was heavily based on a certain bug tracking system. The solution was not about simply copying a given issue ticket management system, original thought process was involved. Design decisions regarding both the front and back end were elaborated on in the part describing the solution. The testing of web applications and their effects on developing web applications were also studied.</p> <p>The solution was implemented in JavaScript as a part of a web application the company was working on at the time. Simplicity and clean design were in a key position in the implementation. A list of features regarding the future development of the solution is briefly reviewed. As an example, the solution was a single tool under a segment. One of the aforementioned future development possibilities is expanding the tool into a segment of its own with increased documentation functionality.</p>		
Keywords/tags (subjects) JavaScript, HTML, browser		
Miscellaneous		

SISÄLTÖ

KÄSITTEET.....	3
1. TYÖN LÄHTÖKOHDAT.....	5
1.1. Tavoite.....	5
1.2. Toimeksiantaja.....	5
2. Tekniikat ja työkalut.....	7
2.1. Git.....	7
2.2. Backbone.....	9
2.3. Bootstrap.....	9
2.4. Bitbucket.....	10
2.5 Trac.....	11
3. Toteutus.....	12
3.1. Johdanto.....	12
3.2. Toimeksianto.....	13
3.3. Vikatiketöinti.....	14
3.4. Yleisesti rakenteesta.....	16
3.5. Front end.....	18
3.6. Back end.....	26
4. Testaus.....	28
4.1 Yleisesti testaamisesta.....	28
5. Lopputulos.....	29
6. Yhteenveto.....	29
6.1. Kuinka projekti eteni.....	29
6.2 Jatkokehitys.....	30
7. POHDINTA.....	31
LÄHTEET.....	32

KUVIOT

Kuvio 1. Esimerkki vikatiketin mahdollisesta elinkaaresta.....	17
Kuvio 2. Vikatikettien listausnäkyä.....	19
Kuvio 3. Komponentti avattuna, josta voi valita pylvääseen uuden tiedon.....	21
Kuvio 4. Uuden vikatiketin luontinäkyä.....	22
Kuvio 5. Uusi vikatiketti täytetyillä tiedoilla.....	22
Kuvio 6. Vikatiketin vastuuhenkilöiden lisäys.....	23
Kuvio 7. Avatun vikatiketin näkyä.....	24
Kuvio 8. Vikatiketin muokkausnäkyä.....	25

TAULUKOT

Taulukko 1. Selventävä taulukko vikatikettien kentistä.....	27
---	----

KÄSITTEET

API – Application Programming Interface. Ohjelmointirajapinta, joka määrittää mitä pyyntöjä sovellukset voivat palvelimelle tehdä.

Back end – Sovelluksen logiikan jaottelu ohjelmistotekniikassa, joka sisältää kaiken palvelimen puolen jaottelussa.

Branch – Kehityshaara. Gitissä käytetty nimitys kopiolle alkuperäisestä versiohallinnasta.

Bugi – Bug. Ohjelmistovirhe.

CSS – Cascading Style Sheets. Käytetään yhdessä HTML:n tiedostojen kanssa, HTML sisältää kaikki verkkosivun elementit yksinkertaisessa muodossa ja CSS sisältää tyyllitetyt HTML tiedoston elementeille.

DOM – Document Object Model. Rajapinta, joka sisältää HTML-dokumentin elementit ja mahdollistaa niiden muokkaamisen.

Front end – Sovelluksen logiikan jaottelu ohjelmistotekniikassa. Sisältää kaikki käyttäjän suoraan käsittelemät komponentit.

HTML – Hypertext Markup Language. Hyperlinkkejä kuvaava kieli, jolla tehdään verkkosivuja.

Integrointi – Ominaisuuden saumatonta järjestelmään liittämistä.

JavaScript – Selaimissa usein käytetty scriptauskieli, jota käytetään usein HTML:n ja CSS:n kanssa dynaamisten verkkosivustojen rakentamiseen asynkronisen kommunikation ansiosta.

Kovakoodattu – Datan sijoittaminen staattisesti suoraan koodiin sen sijaan, että se dynaamisesti määritettäisiin tai erikseen asetettaisiin johonkin valikkoon.

MVC – Model-View-Controller. Ohjelmistoarkkitehtuuriperiaate, jonka tarkoitus on erotella käyttöliittymä datan prosessoinnista.

MySQL – Suosittu web-palveluissa käytetty relaatiotietokanta.

PHP – Hypertext Preprocessor. Ohjelmointikieli, jota käytetään dynaamisten verkkosivujen laatimisessa.

Templaatti – Backbonessa käytetty HTML:ää sisältävä tiedosto, joka sisältää helposti uudelleenkäytettävän osion.

Wiki – Jonkin yhteisön ylläpitämä tietoa sisältävä verkkosivusto.

1. TYÖN LÄHTÖKOHDAT

1.1. Tavoite

Opinnäytetyössä tavoitteena oli luoda toimeksiantajan palveluun toiminnallisuus vikatikettien hallintaan. Ratkaisu on tiiviisti ilmaistuna selaimessa toimiva toiminnanohjausta yksinkertaistava työkalu. Toimeksiantaja toivoi ratkaisua vikatikettien hallintaan modernien mobiili- ja web-työkalujen avulla. Vikatikettejä tuli voida luoda ja muokata, niistä tuli saada notifikaatioita ja tikettien dokumentaatioon tuli kiinnittää huomiota. Tarkempi kuvaus toimeksiannosta käydään läpi toteutuksen kappaleessa.

1.2. Toimeksiantaja

MDO Group Oy on uudella nimellä toimiva mobiili- ja palvelinteknologiayhtiö, joka jatkaa siihen yhdistettyjä Telonet Oy:n ja Versine Oy:n liiketoimintoja. MDO Group tarjoaa operaattoreille, laitevalmistajille ja toimittajille suunnattuja tietoliikenne-, sähkötekniikka- ja energiainfrastruktuurin rakentamiseen, seurantaan sekä ylläpitoon suunniteltuja ohjelmistoratkaisuja. MDO Group Oy:n päätoimipiste sijaitsee Jyväskylässä ja muut toimipisteet Suomessa viidellä eri paikkakunnalla. (Aho, K. 2014.)

MDO:n taustalla oleva tiimi on erittäin vahva sen henkilö- ja yritysomistajien koostuessa mm. ohjelmisto-osaamisen ja rahoitusalan huippuammattilaisista. Edellä mainittuja vahvistaa ja täydentää yrityksen sisäinen toimialaosaaminen mm. tietoliikenneinfrastruktuurin rakentamisen, ylläpidon, testauksen, suorituskyvyn analysoinnin sekä alaan liittyvien standardien ja laitteiden tutkimuksen ja kehityksen saralla. Näin ollen kilpailuvalttina ja ratkaisun edelläkävijöiden takana on toimialan laaja osaaminen ja osaajaverkosto koko tietoliikenneinfrastruktuurin elinkaaren hallinnan saralta. (Aho, K. 2014.)

Versine Oy, MDO Groupin pääomistaja, on keskittynyt tuottamaan kuluttajille ja yrityksille suunnattuja moderneja pilvipalvelu- ja mobiiliratkaisuja. Yrityksen tähän mennessä kehittämiä kuluttajasovelluksia on ladattu maailmanlaajuisesti yli 60 000 kappaletta ja kehitettyjä sovelluksia on nostettu esille mm. radio-ohjelmissa ja operaattoreiden maakohtaisissa sovellussuosituksissa. Yritysratkaisujen osalta Versine on kehittänyt ratkaisuja operaattori-, laitevalmistaja-, urakoitsija ja logistiikka-alan tarpeisiin. Yrityksen henkilö- ja yritysomistajat (pääomistaja Magister Solutions Oy) toimivat vahvasti mukana tietoliikenneteknologioiden ja laitekehityksen kärjessä tuoden MDO Oy:lle paitsi yrityksen omaa vahvaa mobiili- ja palvelinohjelmisto-osaamista niin myös korvaamatonta langattomien verkkojen ja laitteiden asiantuntija-, analysointi ja algoritmiosaamista. (Aho, K. 2014.)

Versinen omistajien referensseihin kuuluvat mm. Kuvaboksi-palvelun ja Kuvat.fi-palvelun integrointi, jossa olemassa oleva yli 200 000 käyttäjän ja 30 miljoonan tietueen järjestelmä integroitiin yhteen toisen olemassa olevan, täysin eri ohjelmointikieliä ja palvelinalustoja käyttävän järjestelmän kanssa. Versinen tiimistä löytyy myös vahva kokemus vahvasti skaalautuvien verkkopalveluratkaisuiden toimittajana. MDO Group Oy on osa Versine-konsernia, joten Versinellä on välitön intressi tehdä MDO Oy:stä seuraava kansainvälinen menestystarina. (Aho, K. 2014.)

Magister Solutions Oy, Versinen pääomistaja, tarjoaa asiakkailleen kansainvälisesti tunnustettuja palveluita aina uusien konseptien ja laitealgoritmien kehittämisestä avaruusteknologian huippuvaatimukset täyttäviin ohjelmistoratkaisuihin. Magister Solutions toimii Suomessa kolmella ja konsernin laajuisesti viidellä eri paikkakunnalla, joista päätoimipiste sijaitsee Jyväskylässä. Yksityisessä omistuksessa oleva yritys on perustettu vuonna 2005 alun perin tekemään tutkimus- ja kehitystyötä langattoman tiedonsiirron radorajapintojen, -resurssien, -simulointien ja verkkojen optimointien parissa ja se työllistää tällä hetkellä noin 30 ICT-alan johtavaa ammattilaista emoyhtiön osalta ja kaiken kaikkiaan konsernin resurssit kattavat noin 40 henkeä. (Aho, K. 2014.)

Magisterin monikansallinen ja korkeakoulutettu henkilöstö koostuu yli 60 % professoreita, tohtoreita ja tohtorikoulutettavia, jotka keskittyvät päivittäisessä työssään ratkaisemaan maailman johtavien mobiili-, ohjelmisto- ja avaruusteknologiatoimittajien ja operaattoreiden haasteita. Magister Solutions on toimittanut perustamisvuodestaan lähtien pitkillä toimitussopimuksilla tulevaisuuden langattomien verkkojen ja päätelaitteiden asiantuntija- ja tutkimuspalveluita Nokialle, Nokia Siemens Networkille (NSN), Renesas Mobile Corporationille sekä vuoden 2012 syksystä lähtien Euroopan avaruusjärjestö ESA:lle. Magisterin lisäksi konserniin kuuluvat tytäryhtiöt Versine ja MDO Group Oyt, joiden päävastuualueena on kehittää ja markkinoida moderneja mobiili- ja (pilvi)palvelinratkaisuja mm. operaattori-, laitevalmistaja- ja operaattori-käyttöön. (Aho, K. 2014.)

2. Tekniikat ja työkalut

Tässä luvussa esitellyt työkalut ja tekniikat olivat käytössä opinnäytetyöprojektin parissa työskennellessä. Alkuperäistä projektia kehitti useamman henkilön ryhmä, mistä johtuen Gitin käyttö versionhallintana oli suotavaa. Backbone on esitelty, koska projektissa oli tarkoituksena tuottaa selaimessa toimiva sovellusratkaisu. Bootstrap auttaa suuresti mobiililaitteiden huomioon ottamisessa. Bitbucket oli käytössä projektin hallinnassa ja toimeksiantaja toivoi Bitbuckettia käytettäväksi mallina ratkaisun suunnittelussa. Trac on otettu toisena vikatiketöintipalveluna vertailukohteeksi Bitbucketin kanssa, jotta ratkaisu olisi kattavampi. Nämä esitellään tämän osion alakappaleissa tarkemmin.

2.1. Git

Versiohallinta on systeemi, joka pitää kirjaa kaikista tiedostoihin tehdyistä muutoksista ja mahdollistaa niiden palauttamisen aiempaan tilaan. Ohjelmistoprojektissa versiohallinnan käyttö on erittäin suotavaa, erityisesti siinä tapauksessa, että projektissa

työskentelee useampi henkilö.

Versiohallinnan voi toteuttaa usealla tavalla:

- Paikallisessa versionhallintajärjestelmässä (Local Version Control System) tiedostojen eri tilat säilytetään nimensä mukaisesti samassa sijainnissa eikä erillisellä palvelimella. Yksinkertaisimmillaan tämä voi olla manuaalisesti tiedostojen kopiointia eri kansioihin, ja asteen hienostuneempi tapa olisi pitää kirjaa tiedostoihin kohdistuneista muutoksista omassa levyosiossa. Tämä tapa sopii yksin työskentelyyn, mutta on virhealtis. (Git - About Version Control 2014.)
- Keskitetyssä versionhallintajärjestelmässä (Centralized Version Control System) otetaan huomioon useamman henkilön työskentely samojen tiedostojen parissa. Tiedostoihin kohdistuneista muutoksista pidetään kirjaa palvelimella, minkä ansiosta on mahdollista nähdä muiden tekemiä muutoksia. Riskinä tässä menetelmässä on palvelimen tavoittamattomuus, eli kukaan ei saisi talletettua tai noudettua tehtyä työtä, mikäli palvelimeen kohdistuisi jotakin saata-vuutta hankaloittavaa. Mikäli palvelimeen kohdistuisi pysyvää vahinkoa kuten kovalevyn tuhoutuminen, menetettäisiin kaikki tieto tiedostojen muokkauksista. (Git - About Version Control 2014.)
- Hajautetussa versionhallintajärjestelmässä (Distributed Version Control System) kaikilla osapuolilla on kopio kaikista muutoksista, joita tiedostot ovat kokeneet. Tämä käytäntö ehkäisee mahdolliset datahävikit, koska palvelimen vaurioituessa riittää, että yhdellä osapuolella on tallessa koko versiohallinnan historia kyseisessä projektissa. Git on rakennettu edellämainitun periaatteen pohjalta. (Git - About Version Control 2014.)

Vaikka työskentelin vikatikettien kanssa yksin, oli Git silti hyödyllinen uusien ominaisuuksien toteuttaessa. Samassa projektissa toimi silti useampi kehittäjä. Uutta ominaisuutta, jonka toteutuksen saattoi vielä muuttaa, oli helppo erotella omaan branchiin. Brancheihin erottelu auttoi myös mahdollisia vikoja etsiessä. Toinen hyöty Gitin käyt-

töstä oli mahdollisuus helposti jatkaa projektin tekoa toisesta sijainnista, toisella työpisteellä.

2.2. Backbone

Työskennellessä web-sovelluksen parissa, jossa on käytetty paljon JavaScriptiä, on tärkeää olla sitomatta dataa dokumenttioliomalliin (Document Object Model). JavaScript-sovelluksissa voi helposti käydä niin, että datan pitäminen synkronoituna käyttöliittymän, JavaScript-logiikan ja tietokannan välillä menee monimutkaiseksi ylläpitää. Selkeämmin jäsenelty rakenne tällaisten rikkaampien sivustojen taustalla säästää paljon resursseja. (Backbone.js 2014.)

Backbonessa data on esitetty olioita muistuttavina malleina, joita voidaan luoda, validoida, tuhota ja tallettaa palvelimelle. Aina kun käyttöliittymästä lähtee mallin tietoa muuttava tapahtuma, kaikki mallin tietoja esittelevät näkymät päivittyvät. Backbonea käyttämällä ei tarvitse etsiä dataa sisältävää elementtiä tunnisteiden perusteella ja päivittää HTML:ää manuaalisesti, mallin päivittyessä näkymät päivittyvät automaattisesti. (Backbone.js 2014.)

Periaatteena Backbonessa on sisällyttää vähimmäismäärä datarakennetta ja käyttöliittymäosia, joiden käyttö on yleisesti hyödyllistä web-sovellusten rakentamisessa JavaScriptillä (Backbone.js 2014.). Alkuperäinen projekti oli toteutettu Backbone.js:llä, joten ei ollut syytä harkita vaihtoehtoja. Usein kookkaampaan MVC-arkkitehtuuria vaativaan JavaScriptillä toteutettuun web-sovellukseen valitaan usein kehitysalustaksi pohjalle joko Backbone.js, Ember.js tai AngularJS.

2.3. Bootstrap

Bootstrap oli projektissa käytössä, koska se on suunniteltu ottamaan ensisijaisesti

mobiililaitteet huomioon. Mikäli web-sovellus ei alun perin ole suunniteltu ottamaan mobiililaitteita huomioon, on niiden huomioon ottaminen myöhemmässä vaiheessa työlästä.

Bootstrap on nettisivujen ja web-sovellusten kehittämistä avustava kokoelma työkaluja, jota tuetaan kaikissa suosituimmissa selaimissa. Se sisältää HTML ja CSS templaatteja yleisille elementeille, kuten painikkeille, navigaatiolle sekä lomakkeille. Alun perin Bootstrapin tarkoitus oli yhdenmukaistaa Twitterin kehittämistä, koska ongelmana oli monen kehittäjän omat tavat toteuttaa uusia ominaisuuksia. (Bootstrap in A List Apart No. 342 2014.)

2.4. Bitbucket

Alkuperäisen tuotteen kehityksen tukemiseksi projektin hallintaan on käytetty Bitbucket-sivustoa. Yksi Bitbucketin käytöistä on ohjelmistobugien vikatiketöintityökalu, josta työnantaja toivoi otettavan mallia vikatiketöintiä projektiin integroitaessa. Bitbucketista löytyy muutakin toiminnallisuutta, kuten versionhallintaa ja dokumentaation ylläpitoa helpottava wiki. Toimeksiannon kannalta oleellista oli vikatikettien osuus.

Vikatikettien päänäkyvä sisältää tikettien listauksen ja filterit niiden seuraamisen helpottamiseksi. On mahdollista listata kaikki tiketit, vain auki olevat tiketit, omat tiketit tai juuri tarkastelun alla olevat tiketit. Vikatikettejä listataan vain 25 kappaletta kerrallaan, jotta näkyvä pysyy siistinä eikä kuormita selainta liikaa. Vain oleellisimmat tiedot on kerrottu listauksessa, jotta tiedon omaksuminen käyttäjälle olisi mahdollisimman nopeaa: tietoja ovat otsikko, tyyppi, tärkeys, tila, vastuuhenkilö, luontipäivämäärä ja edellisen muokkauksen päivämäärä. Päälistauksen toimintoina ovat tietenkin haku ja uuden vikatiketin luominen.

Uuden tiketin luontinäkyvä on yksinkertainen. Vikatiketille vaaditaan nimi, tyyppi ja tärkeys, mutta toivottavaa on antaa myös kuvaus sekä mahdollinen vastuuhenkilö tai

liitetiedosto. Vikatikettiin on hyvä lisätä liitteenä kuva, josta kyseisen ongelman havainnollistaminen on tarkkaa.

Avatun tiketin näkymässä näytetään vain oleellinen, jotta näkymä säilyy tiiviinä ja helpposti omaksuttavana. Oleellisimmat esitetyt tiedot ovat vikatiketin otsikko, kuvaus, luoja, luomispäivämäärä, lisätyt kommentit, tyyppi, tärkeys, tila, sekä vastuuhenkilö. Tärkeimmät toiminnot ovat kommentin lisääminen ja tiketille luomisessa annettujen tietojen muokkaaminen.

2.5 Trac

Bitbucket ei ollut ainoa ohjelmistobugien vikatiketöintiin käytettävä palvelu, johon toteutus on perustettu. Ohjelmistotuotannon kurssilla tutustuin Tracciin, jossa on kätevästi wiki ja vikatiketöinti samassa. Trac on ulkoasultaan huomattavasti vähemmän työstetympi kuin Bitbucket.

Vikatiketit listaava näkymä listaa oletuksena 100 tikettiä, mutta tämän muuttaminen on helppoa sillä lukumäärää säätelevä elementti on näkyvästi esillä ensikertalaisellekin. Tiketiltä listataan lähes samat olennaiset tiedot kuin Bitbucketissäkin. Listauksessa on tunniste, otsikko, tyyppi, tila, tiketin luoja ja milloin tiketti on luotu. Lisänä Tracissa on versio ja virstapylväs, eli omalla tavallaan tiketin tavoitteellinen aikaraja.

Listauksesta avatessa yksittäisen tiketin avautuu näkymä, jossa listataan tiketin tiedot, listataan tiketin historia, listataan tiketin kommentit, annetaan mahdollisuus lisätä kommentteja ja mahdollisuus muokata tiketin tietoja. Näkymä on ensikertaa tiketin avaajalle hyvin sekava. Elementtejä on paljon, teksti on pientä ja näkymä listaa kerralla liikaa kaikkia toimintoja.

Näkymä josta luodaan uutta vikatikettiä on huomattavasti selkeämpi, kaikki täytettävät tiedot erottaa ensivilkaisulla. Tiketin voi erikseen esikatsella, koska luontilomake ei juuri muistuta varsinaista tikettiä kun se avataan tarkasteltavaksi.

Huomioitavaa tiketin luonnissa on ettei vian havainnoijan tarvitsi olla erikseen käyttäjänä palvelussa, vaan tiketin voi tehdä myös nimettömänäkin.

3. Toteutus

3.1. Johdanto

Käyttöliittymällä on erittäin suuri vaikutus järjestelmän käytettävyyteen. Hyväkin järjestelmä, joka oikein käytettynä säästäisi paljon resursseja, on hyödytön niin kauan kuin kukaan ei osaa sitä käyttää. Järjestelmiä tuskin otetaan käyttöön ajatellen niiden tulevan hyvin pienen käyttäjäryhmän käyttöön, vaan ne usein otetaan hyvinkin laajan ryhmän käyttöön. Tällaisessa tapauksessa resursseja säästyy mittava määrä, kun jokainen käyttäjä saa tehtävänsä tehtyä käyttämättä turhaa aikaa käyttöohjeiden ymmärtämiseen.

Yksi nykyajan trendeistä on käyttöliittymän tiivistäminen. Turhat sanat käännetään ikoneiksi, ja kaikki palaset pyritään pakkaamaan hyvin tiiviiseen ja minimalistiseen kokonaisuuteen. Montaa tuntemaani kehittäjää, itse mukaan lukien, hidastaa eri sovellusten kausittaiset käyttöliittymien muutokset. Usein tietyt avaintoiminnot vaikuttavat täysin kadonneen tai ne vähintäänkin on piilotettu salaperäisten ikonien alle. Pyrin välttämään liikaa tiivistämistä, sillä tiesin miten montaa käyttäjää se tulisi haittaamaan.

Käyttöliittymän tuli olla huoliteltu, järkevä ja intuitiivinen. Uuden, mahdollisesti vähän tietokoneita käyttäneen käyttäjän tulisi osata käyttää vikatikettejä oikein ensimmäisellä kerralla. Suunnittelussa on pyritty äärimmäiseen itsestäänselvyyteen niissä puitteissa, että vikatikettien käyttöliittymä edelleen muistuttaa muun järjestelmän käyttöliittymän tyyliä. Mitä vähemmän käyttäjälle ovat modernit web-sovellukset tuttuja, sitä tärkeämpi on käyttöliittymän ohjaavuus.

Toimeksiannossa korostettiin mobiililaitteiden huomioon ottaminen tärkeänä seikkana. Mikäli alkuperäisessä tuotteessa ei Bootstrappia olisi ollut käytössä, olisi se tässä vaiheessa otettu projektiin. Alkuperäisessä projektissa oli jossain määrin ongelmana käyttöliittymän epäjohdonmukaisuus. Bootstrap kehitettiin juuri vastaavanlaisten ongelmien korjaamiseksi ja se on kehitetty mobiililaitteet huomioiden, joten se sopi erittäin hyvin työkaluksi projektiin.

3.2. Toimeksianto

Versine Oy oli kehittämässä asiakkaalle dokumentaation ja tiedonhallintaa yksinkertaistavaa ratkaisua, jonka oli tarkoitus vähentää useamman erillisen työkalun tarvetta. Kaikki oleellinen oli koottu yhteen web-sovellukseen, johon tehtävä oli lisätä mahdollisuus pitää kirjaa erilaisista vioista vikatikettien muodossa. Tikettejä oli tarkoitus kytä kätevästi mobiilista luomaan, kun mahdollisuutta tietokoneelle pääsyyn ei ole.

Sen lisäksi tiketin luonnin piti olla mahdollista pääsovelluksesta. Tiketeillä oli tarkoitus pitää kirjaa siitä missä vika sijaitsee, vian tyypistä, kenelle vika on nimitetty ja vian elinkaari. Käytännössä toimeksianto muistutti vianhallintajärjestelmää (Issue tracking system), tai ehkä sopivampi vertaus on tässä tapauksessa ohjelmistovikojen hallintaan käytettävä järjestelmä (Bug tracking system).

Tehtävänantoon kuului myös antaa ilmoituksia virhetikettien tilojen muutoksista. Ottaen huomioon, että järjestelmää tulee käyttämään suuri määrä käyttäjiä, on syytä olettaa aikataulujen mutkistavan asiaa. Lisäksi aina ei välttämättä ole mahdollisuutta päästä kirjautumaan järjestelmään. Pelkkä järjestelmän sisäinen ilmoitus juuri saapuneesta tiketistä ei siis riitä. On syytä tarjota mahdollisuus lähettää myös sähköposti.

Toimeksianto ei ollut vain jäljentää olemassa olevaa vikatikettien hallintatyökalua, vaan tarkoitus oli integroida se järjestelmään. Toisin kuin vikatikettijärjestelmissä,

joissa ongelmien fyysisellä sijainnilla ei ole erillistä visuaalista esitystä, tässä tapauksessa se on mahdollista varsinkin, kun järjestelmässä muutenkin käytetään sijainteja kartalla. Tämä ominaisuus sitoisi työkalun huomattavasti tiiviimmin järjestelmään, mutta se liittyy hyvin etäisesti vikatikettien dokumentointiin. Kartan ja vikatikettien yhdistäminen on siis korkeintaan jatkokehitysmahdollisuus.

Olisi epäkäytännöllistä pitää kirjaa vioista, jos siihen käytettäisiin vain sähköpostitiliä. Dokumentaation puute muodostuisi myöhemmin ongelmaksi, ja vikojen määrän kasvassa suureksi ja jakautuessa useaan eri sijaintiin olisi ongelmien hallinta erittäin työlästä ja aikaa tuhlaavaa. Sen sijaan, että joku ylläpitäisi kirjaa vioista erillisessä taulukossa sähköpostien perusteella ja myöhemmin kadottaisi tiedoston, kun on tarvetta, jolloin tilanne olisi sama tai mahdollisesti pahempi kuin edellisessä menetelmässä. Erillinen työkalu muutenkin olisi epäkäytännöllistä käyttää puhumattakaan mahdollisista muutoksista, joiden päivittäminen irralliseen työkaluun olisi resurssien hukkaa.

Toimeksiantoon kuului myös vikatikettien dokumentointi. Dokumentoinnin osalta oli selvää pitää kirjaa tiketin tiloista. Milloin ticketti on tehty, kenelle se on määritetty, milloin se on korjattu tai mahdollisesti avattu uudelleen. Muuta mahdollisesti hyödyllistä dokumentaatiota on kartalla tietyn säteen sisällä kaikki löytyneet viat tai kaikki tietyn vastuuhenkilön korjaamat ticketit. Mahdollinen kuvaaja, jossa on esitetty tehdyt vikaticetit ajan funktiona, voisi olla hyödyllinen.

Järkevää on integroida vikojen hallinta ja dokumentointi. Alkuperäisen projektin pohjalla oli idea yhdistää useampi työkalu yhdeksi järjestelmäksi ja kyetä hoitamaan kaikki tarpeellinen yhden sovelluksen alla. Näin käytännöllisyys ja ylläpito ovat molemmat korkealla.

3.3. Vikatiketöinti

Järjestelmä on useamman komponentin muodostama kokonaisuus, jonka tarkoituksena on saavuttaa jokin päämäärä. Integroitu tarkoittaa yhteen liitettyä. Laadukkaas-

sa järjestelmässä ominaisuuksien integroinnit ovat saumattomia, hankaloittaen niiden tunnistamista erillisiksi ominaisuuksiksi. Päämäärä oli tämä määritelmä mielessä pitäen kehittää työnantajalle toteutus.

Ominaisuutta ei voi kutsua integroiduksi vain, koska se on liitetty osaksi kiinni järjestelmään. Integroidun ominaisuuden on tarkoitus liittyä järjestelmään saumattomasti. (Integrated management system 2014.)

Ohjelmistovikojenhallintajärjestelmä on vikatiketin luontia yhdestä ongelmasta ja ongelman ratketessa tiketti suljetaan. Mikäli ongelma muuttuu tai löydetään uusi vika tehdään siitä erillinen tiketti. Vianhallintajärjestelmä puolestaan on vikatiketin luontia yhden asiakkaan ongelmasta. Tiketti merkitään valmistuneeksi vasta, kun kaikki siihen liittyvät ongelmat on saatu käsiteltyä. Toimeksiantajan kuvaus toteutettavasta työkäystä oli hyvin suurelta osin kuvaus ohjelmistovikojen hallintaan käytettävän järjestelmän ominaisuuksista, joten päätin tutkia niitä tarkemmin.

Vianhallintajärjestelmille yleisiä piirteitä ovat :

- Usein käytössä organisaation tukikeskuksessa, jonne kirjataan asiakkaitten tai työntekijöitten kohtaamat viat
- vianhallintajärjestelmät usein sisältävät myös tietoa asiakkaista sekä ratkaisuja yleisiin ongelmiin

Ohjelmistovikojenhallintajärjestelmälle yleisiä piirteitä ovat:

- integroitu yhteen muiden hallintatyökalujen kanssa, ohjelmistovirheet ovat vain yksi osa vianhallintajärjestelmää
- käyttö on merkki hyvästä softatuottajasta
- tietokantaan löytöaika, vakavuus, kuvaus, kuka löysi, kuka korjaamaan
- ohjelmistovirheelle annetaan tila, jota seurataan tietokannassa
- järjestelmänhallitsijalla kyky hallita oikeuksia ohjelmistovirheen tilan perusteella, vaihtaa ohjelmistovirheen tilaa, poistaa tiketin järjestelmästä
- jotkin järjestelmät sähköpostittavat ohjelmistovirheeseen liittyvistä muutoksista

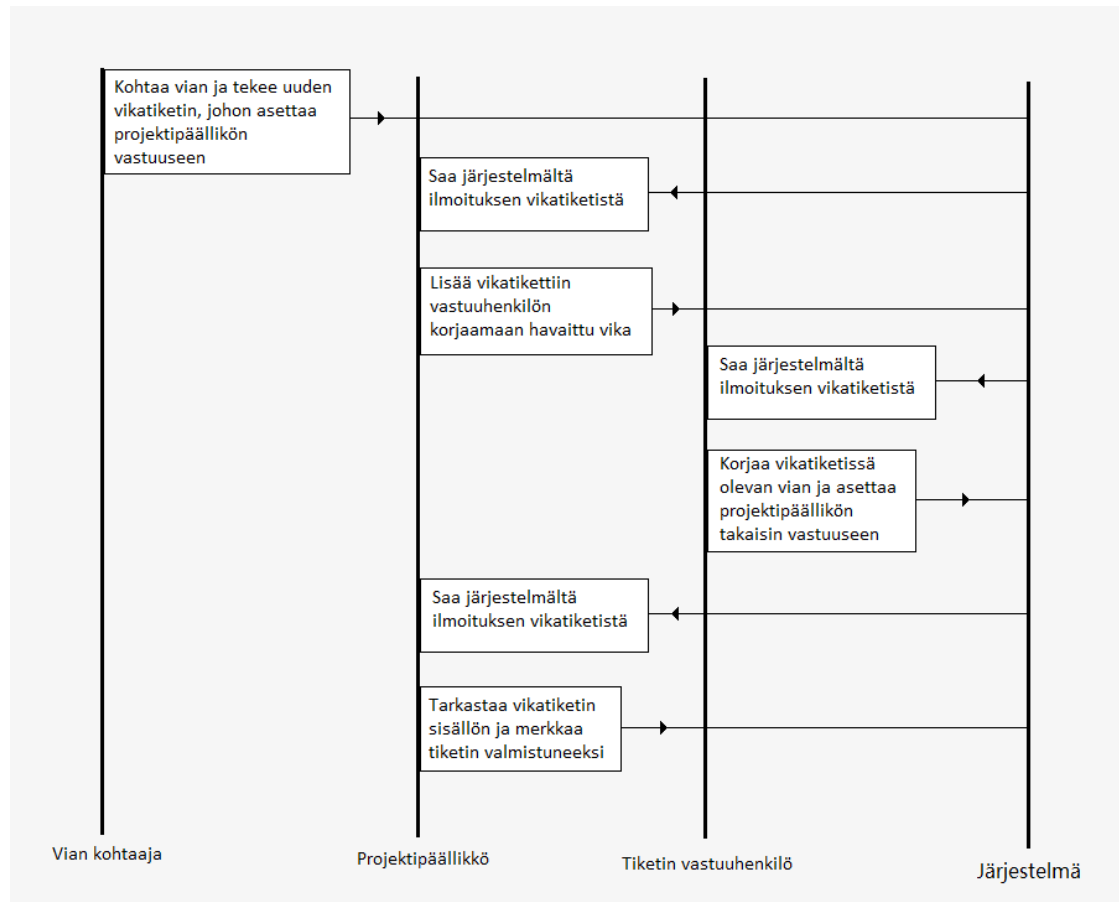
3.4. Yleisesti rakenteesta

Toimeksiantajan projekti on toteutettu front endissä JavaScriptillä ja back endissä PHP:llä. Lisäksi tietokannaksi on valittu MySQL. Havainnollistan tässä kappaleessa mistä vikatiketti syntyy ja mihin se päättyy. Työkalu jättää vastuun vikatketin läpivienmistä asiakkaalle, joka tuotetta käyttää. Ei siis ole olemassa vain yhtä tapaa käyttää niitä, käyttö riippuu asiakkaana olevasta yhtiöstä ja sen omista käytänteistä.

On siis mahdollista, että itse vian löytäjä myös suorittaa korjaavat toimenpiteet ja merkkää lopuksi vikatketin suoritetuksi ennen kuin mahdollisesti raportoi asiasta eteenpäin. Joissain tapauksissa vian löytäjä ohjaa vikatketin jollekin projektipäällikölle, joka puolestaan ohjaa tiketin oikealle vastuuhenkilölle. Vastuuhenkilö suorittaa korjaavat toimenpiteet, ohjaa vikatketin takaisin projektipäällikölle ja projektipäällikö tarkastaa tiketin sisällön ennen sen sulkemista.

Toteutukselle oleellista on joustavuus, jotta mahdollisten tulevien asiakkaiden haluama työnkulku vikatikettien suhteen olisi toteutettavissa mahdollisimman helposti. Kuviossa 1 on esitetty mahdollinen kulku vikatketille luonnista valmistumiseen.

Kuvio 1. Esimerkki vikatietin mahdollisesta elinkaaresta



Firma ottaa käyttöön kehitetyn järjestelmän ja yksi työntekijöistä kohtaa fyysisen vian työhönsä liittyen. Vian kohtaaja on etukäteen määritetty järjestelmään käyttäjäksi, joten hän kirjautuu sisään ja avaa vikatietien näkymän. Vian kohtaaja luo uuden vikatietin paikanpäällä mobiililaitteestaan tai luo sen myöhemmin pääsovelluksesta.

Vian kohdanneen työntekijän tiketti osoitetaan projektipäällikölle, joka ohjaa sen eteenpäin työntekijälle, joka määritetään tikettiin vastuuhenkilöksi. Vastuuhenkilöksi nimetty työntekijä suorittaa fyysiset korjaavat toimenpiteet ja ohjaa tiketin takaisin projektipäällikölle. Projektipäällikkö tarkastaa tiketin sisällön ja suorittaa toimenpiteet, joilla vastaavia ongelmia mahdollisesti jatkossa ehkäistään. Vikatietinratkaisu ei tarjoa tällaista toiminnallisuutta.

3.5. Front end

Vikatiketit listaava näkymä sisältää oleelliset toiminnot yleisesti vikatikettien kanssa. Klikkaamalla ”Uusi tiketti”-painiketta avautuu uuden tiketin luontinäkymä. Tikettejä voi suodattaa haulla, jolloin listaus esittää vain tiketit, joiden tiedoissa jokin tieto täsmää hakuun. Klikkaamalla ylimmältä riviltä painikkeista järjestellään kyseisen pylvään tiedot aakkosjärjestykseen. Klikkaamalla uudelleen samasta painikkeesta käännetään järjestys toisinpäin.

Järjestystä muuttavan painikkeen vieressä sijaitsevaa nuolella varustettua painiketta klikkaamalla avautuu valikko kaikista tiedoista, joita tiketeiltä on mahdollista tarkastella. Valikosta valittaessa yksi tieto, kyseinen pylväs vaihtuu näyttämään valitun pylvään tiedot.

Tuote oli vielä kehitteillä, joten moni taulukko oli tehty eri tavalla. Niissä oli kuitenkin kaksi yhteistä ongelmaa: taulukot vaativat responsiivisuuden korjaamisen mobiililaitteita varten ja saman taulukon käytön kaikkialla, jotta yhdenmukaisuus säilyisi ja ylläpitoon ei menisi resursseja hukkaan. Tekijästä riippuen taulukot saattoivat olla hyvin erinäköisiä ja toteutettu eri tavoilla.

Kuviossa 2 käytetty taulukko on kehitetty ratkaisemaan taulukoiden ongelmat responsiivisuuden sekä yhdenmukaisuuden suhteen. Taulukossa on hyödynnetty Bootstrapin ominaisuuksia selvittää päätelaitteen näytön koko. Taulukon pylväiden määrä määrittyy juuri näytön koon perusteella, jotta käyttöliittymä pysyy siistinä. Pienemmillä laitteella kuten puhelimella pylväitä voi olla esimerkiksi kolme ja loput tiedot on mahdollista nähdä pylvään tietoja vaihtamalla nuolella varustetusta painikkeesta.

Kuvio 2. Vikatikettien listausnäkyminen

Vikatiketit:

Uusi tiketti

Tunniste	Otsikko	Tila	Tärkeys	Tiketin työntekijät
1	Lamppu hajonnut	Ei aloitettu	Normaali tärkeys	Versine Oy, Testimies
2	Lentävä alligaattori	Meneillään	Suuri tärkey	Testi käyttäjä, tester, Versine Oy

Kyseisen komponentin käyttö on säästänyt resursseja taulukoiden ylläpidon osalta, kun se on otettu käyttöön muuallakin projektissa. Kehittäjien aikaa säästyy huomattavasti, kun taulukoita ei tarvitse erikseen muokata jokaiseen näkymään. Klikkaamalla yksittäistä riviä taulukosta avataan vikatiketin tiedot esittävä näkymä.

```

for (i = borderWidthMobile; i < this.originalList.length + 1; i += 1)
{
    this.$(".column" + i).addClass("hidden-phone");
}
for (i = borderWidthTablet; i < this.originalList.length + 1; i += 1)
{
    this.$(".column" + i).addClass("hidden-tablet");
}
for (i = borderWidthDesktop; i < this.originalList.length + 1; i += 1)
{
    this.$(".column" + i).addClass("hidden-desktop");
}

```

Yllä olevassa koodiesimerkissä näkyy miten komponentin pylväiden määrää säädel-
lään vaihdellen päätelaitteesta. Pylväiden määrään säätelevät muuttujat
ovat määritetty erikseen JOSSAIN kovakoodaamisen sijaan. For-silmukoissa käydään
läpi komponentin pylväät ja niitä piilotetaan riippuen siitä, kuinka monta pylväitä on
yhteensä ja mikä päätelaite on.

```

if (currentCol !== undefined && targetCol !== undefined && !this.switched)
{
    this.switched = true;
    this.swapButtonFilters(currentCol, targetCol);
    this.currentList[parseInt(currentCol, 10)] = parseInt(targetCol, 10);
}

```

Yllä olevassa koodiesimerkissä käsitellään pylväiden tiedon vaihtumista. Pylväs merkkataan vaihdetuksi ja järjestystä muuttavien painikkeiden teksti vaihdetaan. Listasta, joka sisältää kaikki näytettävät tiedot otetaan sen pylvään tiedot, joka on valittu nähtäväksi.

Alla olevassa koodiesimerkissä valmistellaan templaatille annettava data. Kaikki komponentille annettujen objektien tiedot käydään läpi ja ne järjestellään sen mukaan mitä tietoja käyttäjä on komponentista valinnut saada nähtäväksi. Vaihdot otetaan talteen kokoelmaan, jota myöhemmin käytetään templaatissa.

```

_(this.originalFilters).each(function (attribute) {
    originalData.push(item[attribute]);
});

```

Alla olevassa koodiesimerkissä on osa kyseisen komponentin templaatista. Silmukassa käydään läpi kaikki tarkasteltaviksi määritellyt tiedot, niiden perusteella luodaan filttereinä toimivat painikkeet ja lopuksi kuviossa 3 avattuun listaan lisätään tarkasteltaviksi määritellyt tiedot. Tämä operaatio suoritetaan tietenkin vain näkyviksi merkatuille pylväille.

```

{{#eachProperty templateButtons}}
  <th class="{{value.class}}">
    <div class="btn-group">
      <button class="btn sortButton"
id="{{value.id}}">{{value.name}}</button>
      <button class="btn btn-small dropdown-toggle" data-toggle="drop-
down"><span class="caret"></span></button>
      <ul class="dropdown-menu">
        {{#eachProperty value.menu}}
          {{#ifCond property "!=" "0"}}
            <li class="listOption"><a href="#">{{value}}</a></li>
          {{/ifCond}}
        {{/eachProperty}}
      </ul>
    </div>
  </th>
{{/eachProperty}}

```

Kuvio 3. Komponentti avattuna, josta voi valita pylvääseen uuden tiedon

Vikatiketit:

Uusi tiketti

Tunniste	Otsikko	Tila	Tärkeys	Tiketin työntekijät
1	Lamppu hajonnut	Ei aloitettu	Normaali tärkeys	
2	Lentävä alligaattori	Meneillään	Suuri tärkeä	

- Tunniste
- Otsikko
- Tila
- Tärkeys
- Tiketin työntekijät
- Viimeksi muokattu
- Luotu

Vikatikettien luontinäkyssä, joka on havainnollistettu kuviossa 4, mahdollisia toimintoja ovat tiketin tallentaminen ja takaisin listaukseen poistuminen. Näkyssä täytetään vikatiketin otsikko, kuvaus, tila, tärkeys sekä nimitetään tiketti jollekin vastuhenkilölle ja lopuksi talletetaan.

Kuvio 4. Uuden vikatietin luontinäköm

Uusi tiketti:

Otsikko	<input type="text"/>	<input type="button" value="Tallenna & Poistu"/> <input type="button" value="Takaisin"/>
Kuvaus	<input type="text"/>	
Tila	Ei aloitettu ▾	
Työntekijä	Ei nimettyä työntekijää <input type="button" value="Nimitä henkilölle"/>	
Tärkeys	Normaali tärkeys ▾	

Vikatietin luontinäköm, jossa tiedot täytetty, on esitetty kuviossa 5. Vastuuhenkilöitä on mahdollista lisätä useampia, mutta vastuuhenkilöä ei tietenkään ole pakko lisätä tässä vaiheessa, mikäli tiedossa on vain itse vika ilman varmuutta, kenelle se tuli allokoida.

Kuvio 5. Uusi vikatietti täytetyillä tiedoilla

Uusi tiketti:

Otsikko	otsikko	<input type="button" value="Tallenna & Poistu"/> <input type="button" value="Takaisin"/>
Kuvaus	kuvausta	
Tila	Ei aloitettu ▾	
Työntekijä	Versine Oy, normiuser <input type="button" value="Nimitä henkilölle"/>	
Tärkeys	Pieni tärkeys ▾	

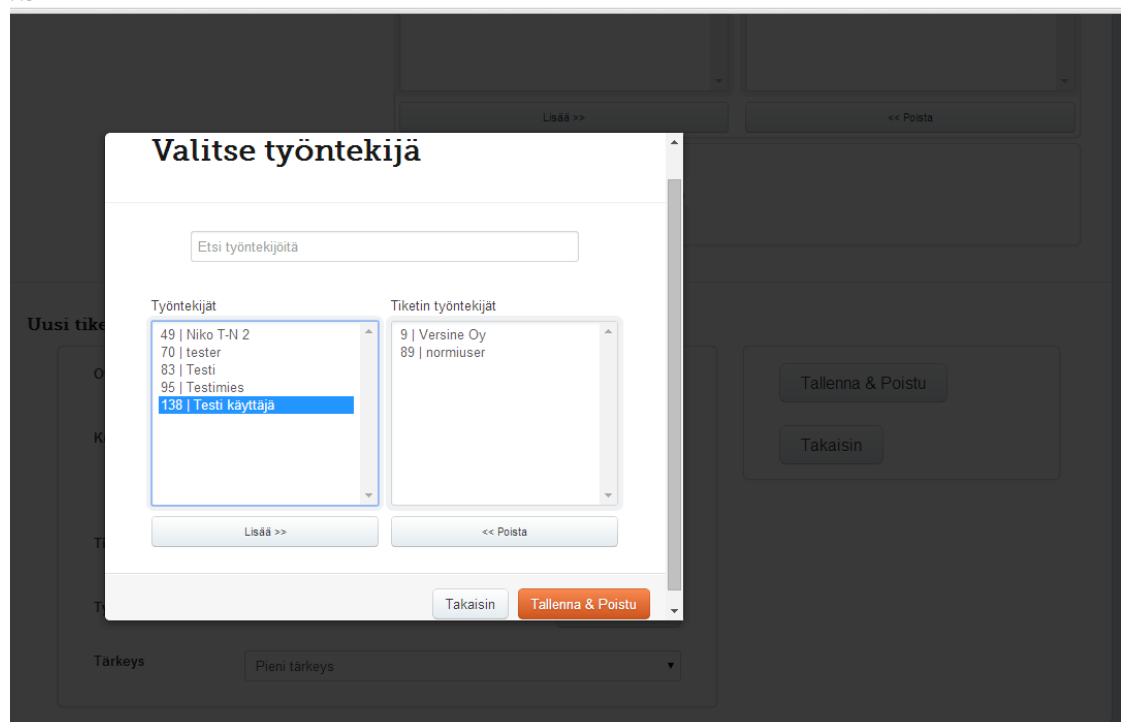
Vikatietin vastuuhenkilöiden valitsemisnäköm on havainnollistettu kuviossa 6. Haku on oleellinen, sillä työntekijöitä voisi olla satoja ja manuaalinen listan selaaminen tuhansi aikaa sekä aiheuttaisi mahdollisia virheitä. Vasemmanpuoleisessa kontrollissa on

esitetty kaikki vastuuhenkilöt, jotka on suodatettu. Näistä valitaan halutut henkilöt ja painetaan kontrollin alla sijaitsevaa painiketta, jolloin valitut henkilöt siirtyvät oikeanpuoleiseen kontrolliin.

Oikeanpuoleisessa kontrollissa sijaitsevat vastuuhenkilöt ovat valittuja, ja klikkaamalla ”Tallenna & Poistu”-painiketta palataan edelliseen näkymään, jossa on listattu valitut vastuuhenkilöt. Vastuuhenkilöiden valitsemiseen käytetty modaali ei ole niin aukottoman ilmiselvä kuin olisi suotavaa, mutta se on vähintään yhdenmukainen muun järjestelmän kanssa.

Kuvio 6. Vikatiketin vastuuhenkilöiden lisäys

140



Kun vikatikettien päänäkymän listauksesta on klikattu yksittäistä tikkettä, avataan tike-
tin tiedot listaava näkymä. Kuviossa 7 on havainnollistettu, miltä näkymä näyttää. Toi-

mintoja näkymässä on tiketin muokkaaminen, joka tapahtuu klikkaamalla ”Muokkaa tikettiä”-painiketta.

Kuvio 7. Avatun vikatiketin näkymä

Tiketin tiedot:

Tunniste	1
Tiketin lähettäjä	Versine Oy
Otsikko	Lamppu hajonnut
Kuvaus	Rakennuksen länsisiiven kolmannen kerroksen huoneesta 321 on pimentynyt lamppu.
Tila	Ei aloitettu
Työntekijä	Versine Oy, Testimies
Tärkeys	Normaali tärkeys
Viimeksi muokattu	30.4.2014 14:25:56
Luotu	27.4.2014 08:47:02

Tiketin historia

13.5.2014 09:15:27 muokkaaja: Versine Oy muokattu: title
15.5.2014 10:08:01 muokkaaja: Testimies muokattu: description
15.5.2014 17:24:59 muokkaaja: Testimies

Tiketin tiedoista kaikki oleellinen näytetään tässä näkymässä käyttäjälle. Tiketin tunniste, tiketin luoja, otsikko, kuvaus, tila, vastuuhenkilöiksi nimetyt, tärkeys, tiketin tietojen muokkaamisen viimeisin ajankohta sekä tiketin luomisen ajankohta. Lisäksi sivussa on tiketin muokkaushistoria, eli aina kun vikatiketin kenttää muutetaan, otetaan talteen, kuka muutti, mitä muutti ja milloin muutos tapahtui. Tiketille lisättiin historian siitä syystä, että mahdolliset tahattomat sekä tahalliset virheet olisi mahdollista selvittää perusteellisesti.

Avatun vikatiketin näkymässä klikattaessa ”Muokkaa tikettiä”-painiketta avautuu tiketin muokkaamisnäkyvä. Kuviossa 8 on havainnollistettu, miltä kyseinen näkymä näyttää. Näkymä on sama kuin mitä se oli tiketin tietoja tarkastellessa, mutta kentät joihin on mahdollista vaikuttaa on muutettu kontrolleiksi, joissa on mahdollista muokata kyseisiä tietoja. Toimintoina näkymässä ovat tiketin tallentaminen ja vikatikettien päälis-taukseen poistuminen.

Kuvio 8. Vikatiketin muokkausnäkyvä

Tiketin tiedot:

Tunniste	1
Tiketin lähettäjä	Versine Oy
Otsikko	Lamppu hajonnut
Kuvaus	Rakennuksen länsisiiven kolmannen kerroksen huoneesta 321 on pimentynyt lamppu.
Tila	Ei aloitettu
Työntekijä	Ei nimettyä työntekijää Nimitä henkilölle
Tärkeys	Pieni tärkeys
Viimeksi muokattu	30.4.2014 14:25:56
Luotu	27.4.2014 08:47:02

Tallenna & Poistu
Takaisin

Tiketin historia

13.5.2014 09:15:27
muokkaaja: Versine Oy
muokattu: title

15.5.2014 10:08:01
muokkaaja: Testimies
muokattu: description

15.5.2014 17:24:59
muokkaaja: Testimies
muokattu: description

17.5.2014 09:56:11
muokkaaja: Versine Oy
muokattu: status

Tiketteihin liittyvät näkymät on suunniteltu mahdollisimman yhdenmukaisiksi ja intuitiivisiksi. Monissa palveluissa toiminnot on koottu samaan tilaan kaikissa näkymissä. Oli erittäin todennäköistä, että useimmat keskittäisivät huomionsa ensin tiedon omaksumiseen vasemmalta oikealle ja oikeassa laidassa huomattaisiin toiminnot. Kaikki toiminnot olisivat pääsääntöisesti koottu oikealle, jotta käyttäjän ei tarvitsisi erikseen niitä etsiä. Näkymissä on siis suuressa määrin ottaa huomioon vähemmän nykYTEKNIIKAN kanssa tekemisissä olevat yksilöt.

3.6. Back end

Tietokannaksi projektissa on valittu MySQL. Luonnollisesti tietokantaan tuleva data suodatetaan mahdollisten injektioiden varalta, tätä ei voi missään nimessä jättää pelkän käyttöliittymän vastuulle.

Vikatikettien taulussa tunniste on luonnollisesti pääavaimena. Molemmat Bitbucket ja Trac listaavat tikettien päänäkyvässä tiketille tunnisteeseen. Vaikka tunnisteeseen ja luomispäivämäärän perusteella tikettien järjestäminen aikaansaa täsmälleen saman lopputuloksen, näytetään tunniste käyttäjälle siitä huolimatta. Tunniste on nopein ja varmin tapa viitata tiettyyn tickettiin käyttäjälle, joka ei kyseisestä ticketistä tiedä mitään.

Tiketeillä on myös kentät siihen kuka tiketin on luonut ja kenet siihen on nimitetty vastuuhenkilöksi. Myöhemmin toimeksiantaja lisäsi, että järjestelmässä olevia käyttäjäryhmiäkin tulisi voida asettaa vastuuseen vikaticetistä, joten tauluun lisättiin tieto mahdolliselle käyttäjäryhmälle vastuuseen vikaticetistä.

Vikatikettien taulussa on tietenkin kentät erikseen otsikolle ja vikaticetin viestille. En kyennyt kuvittelemaan tilannetta, jossa olisi kätevää pitää nämä tiedot samassa kentässä. Vikatikettien listauksessa näytetään yleensä vain otsikko, tiketin syvämpi sisältö on näytetty vasta ticketti erikseen avatessa. On siis huomattavasti loogisempaa pitää nämä tiedot erillään, varsinkin kun vikaticettejä tulee joku muu kehittäjä käyttämään.

Vikatiketteihin liittyviä hyödyllisiä ajankohtia ovat milloin ticketti on luotu ja milloin tickettiä on viimeksi muokattu. Myöhemmin kantaan lisättiin yleisesti ohjelmistovikojen hallintajärjestelmistä siinä, että tiketille annettiin mahdollisuus lisätä aikaraja.

Tiketeillä on myös kenttä mahdollisille kommenteille ja tilojen muutoksille. Kun tiketin

tilaa muutetaan otetaan talteen muuttuneen tiedon kenttä, kuka sitä muutti sekä milloin muokkaus tapahtui. Tiketin historian osilta jää vielä varaa jatkokehitykselle, sillä se on varsin tärkeä ominaisuus. Mikäli joku harhaanjohtavasti tai tahattomasti muokkaa jotakin tiketin kenttää, on syytä saada selville mikä alkuperäinen arvo tiketissä oli ennen muokkausta.

Taulukko 1. Selventävä taulukko vikatikettien kentistä

Kenttä	Kentän tyyppi
Tunniste	Numero
Otsikko	Tekstiä
Sisältö	Tekstiä
Luontiajankohta	Aikaleima
Viimeisimmän muokkauksen ajankohta	Aikaleima
Aikaraja	Aikaleima
Tiketin luoja tunniste	Numero
Tiketin vastuhenkilöt	Kokoelma numeroita
Tiketin vastuuryhmät	Kokoelma numeroita

Vikatiketin tila ja tärkeys päätettiin lisätä kantaan joustavuuden takia. Mikäli asiakas myöhemmin haluaa lisätä tai muuttaa tikettien mahdollisia tiloja, se on helpoin tehdä silloin, kun tilat ja tärkeydet löytyvät tietokannasta, ja ne haetaan sieltä. Tilojen ja tärkeyksien ylläpitäminen olisi työläämpää, mikäli ne olisi kovakoodattuna sovelluksessa.

Pohdin myös oliko tarvetta pitää kirjaa tiketin jokaisesta iteraatiosta. Eli kun tikettiä muokataan, onko tarve saada tietää jokaisen tiketin muokkauksen jälkeen mitä vikatiketin tiedot annetussa iteraatiossa olivat. Tämä ominaisuus vaikutti liioittelulta, etenkin kun tarkoitus oli aikaansaada yksinkertainen työkalu. Jatkokehityksen mielessä pitäen ominaisuus on jotain, mitä joku asiakas voisi mahdollisesti toivoa vikatikettien kykenevän tekemään.

4. Testaus

4.1 Yleisesti testaamisesta

Jo ennen ratkaisua kehittäessäni työskentelin toimeksiantajan projektin parissa. Opin testaamisen tärkeyden ja ennen kaikkia miten helposti useamman kehittäjän toimies-sa samojen ominaisuuksien parissa voi tulla ennalta näkemättömiä bugeja.

Web-sovellusten testaamisessa on otettava huomioon yhteensopivuus eri selainten kanssa. Uusimmat ominaisuudet eivät välttämättä ole tuettuja kaikissa selaimissa ja mikäli ne ovat tuettuja, voi niissä silti olla eroja. Tämä lisää testaamisen tarvetta huomattavasti. Lisäksi ongelmana on ihmisvirheet eli kehittäjä itse testatessaan sovellusta unohtaa testata jonkin ominaisuuden. Toinen ongelma on kun kehittäjä testaa kaikki suoraan uuteen ominaisuuteen liittyvät toiminnot, mutta ei käy läpi järjestelmän muuta toiminnallisuutta.

Testien automatisointi on edellä mainittujen ongelmien äärimmäisen tärkeää. Kehittäjät itse eivät joudu testaamaan jokaisen muutoksen jälkeen, joten tilaa virheiden tapahtumiseen on vähemmän. Lisäksi ihmisen aiheuttamia virheitä testien suorittamisessa ei tule. Testien automatisointi on ideana loistava. Pyritään siirtämään mahdollisimman paljon mekaanista toistoa vaativat toimenpiteet koneen suoritettavaksi ja näin säästetään aikaa sekä muita resursseja.

5. Lopputulos

Toimeksianto oli siis suunnitella ja toteuttaa vikatikettien hallinnan mahdollistava osio toimeksiantajan kehityksessä olevaan järjestelmään. Lopullinen tuotos vastasi suurimmaksi osaksi toimeksiantoa. Toteutuksessa vikatiketin luonti, tarkastelu ja muokkaaminen vastasivat sitä, mitä olin suunnitellut. Näkymät kaikille noista olivat selkeitä ja käyttöliittymässä oli otettu huomioon mobiililaitteet.

Vikatikettien lähettämät notifiikaatiot toimivat suunnitellusti. Järjestelmän sisäinen notifiikaatio, sekä sähköpostitse aina kun tiketin tila vaihtuu tai vastuuhenkilöitä on muutettu. Vikatikettien dokumentointi jäi vähemmälle painolle. Tiketeistä ei ollut erikseen mitään muodostettavia koosteita, joista olisi nähnyt haluttua dataa tiketien listausta lukuun ottamatta. Mikäli haluttiin saada selville esimerkiksi vastuuhenkilön N.N. kaikki keskeneräiset tiketit, kirjoitettiin tiketit listaavan näkymän hakuun N.N. ja klikattiin tiketit listaavassa näkymässä tila-pylvästä. Näin listauksesta on nopeasti selvitettävissä halutut tiketit.

6. Yhteenveto

6.1. Kuinka projekti eteni

Tarkoituksena projektissa oli saada aikaan toimiva vikatiketöintiratkaisu, jossa otettiin huomioon modernit mobiilityökalut. Suunnitteluvaihe eteni varsin mallikkaasti. Työkentely lähti eri näkymien hahmottelulla ja suunnittelulla. Tutustuin tässä vaiheessa pääasiassa Bitbuckettiin ja koitin ymmärtää kaiken oleellisen vikatiketteihin liittyen. Suunnittelin käyttöliittymää muutaman iteraation ja kysyin muiden projektissa työkennelleiden kehittäjien mielipidettä.

Tässä vaiheessa aloitin toteuttamaan käyttöliittämästä prototyyppiä ja alustavasti suunnittelemaan vikatikettien toteutusta tietokannan puolella. Taulujen suunnittelu ei ollut ongelma, sillä olin käynyt tietokantojen suunnitteluun ja hallintaan liittyen kurseja, joilla kyseiset asiat opetettiin varsin perusteellisesti. Näiden osalta oli ennestään valmiudet, joitten hyödyntäminen oli mekaaninen suoritus.

Seuraavassa vaiheessa suunnittelin vikatiketteihin liittyviä API-kutsuja. Näitä en ollut aiemmin itse tehnyt ja sain neuvoa niiden hiomiseen kollegoilta. Aluksi niiden suunnittelu oli hankalaa kokemattomuuden takia, mutta ensimmäisen jälkeen ymmärsin miten prosessi etenee ja tämäkin haaste muuttui suoraviivaiseksi. Ainut aiempi läheinen kokemus API-kutsuihin oli Python-kielellä toteutetun projektin parissa, johon muokkasin hieman olemassa olevia kutsuja.

Lopuksi kehitin käyttöliittymän prototyypistä kunnollisen version, jossa oli hyödynnetty juuri Bootstrappiä. Sain palautetta eri vaiheissa, etenkin käyttöliittymän osilta. Käyttöliittymän parantelun kannalta palautteen saaminen oli äärimmäisen arvokasta. Suurimmaksi osaksi pysyin alkuperäisessä suunnitelmassa, mutta muokkasin sitä pienissä määrin palautteen perusteella.

6.2 Jatkokehitys

Toimeksiannolle jäi vielä monia mahdollisuuksia jatkokehityksen saralle. Dokumentaatioon liittyvä toiminnallisuus oli mielestäni hyvin tärkeää. Myöhemmin toimeksiantajalta tuli tarkentavia toiveita, että vikatiketillä tulisi olla mahdollisuus lisätä kuvia ja liitteitä. Eli mahdollisesta viasta otetaan tikettiä tehdessä kuva, joka selventää vikatiketin vastuuhenkilölle myöhemmin varsinaisen vian.

Liitteet ovat yleishyödyllisiä ja niiden käyttö on mitä luultavimmin samaa kuin kuvalla. Liitteenä tosin on mahdollista laittaa esimerkiksi videokuvaa, mikäli se on aiheellista ja selventää ongelmaa tavalla, jota ei muulla tavalla voi selostaa.

Yksi vikatiketöintiratkaisun keskeisiä kehityskohteita on virheiden korjaamisen vaikutus. Pelkkä tiketöinti viittaa mekaaniseen tietojen kirjaamiseen, mutta vikatiketöinti puolestaan viittaa vikoihin puuttumiseen. Vikatikettijärjestelmän tulisi kyetä vaikuttamaan virheiden esiintymisen vähintään ilmoittamalla milloin on ylitetty raja tietynlaisten virheiden osalta. Järjestelmään suunniteltiin mahdollisena jatkokehityskohteenä tiketeille ennaltamääritettyjä kategorioita, joiden yleisyyttä voisi valvoa. Kun ennalta määritetty raja ylittyy tietyllä tiketin kategorialle, järjestelmä antaisi siitä erikseen tiedon.

7. POHDINTA

Opinnäytetyön aihe oli yksinkertainen, mutta haastava. En ollut työskennellyt JavaScriptillä vielä kovin kauaa ennen opinnäytetyötä aloittaessani. Järjestelmä, johon toimeksianto toteutettiin, oli melko kookas ja aluksi kului paljon aikaa oppia tuntemaan sen rakenne. Onneksi olin työskennellyt järjestelmän parissa usean kuukauden ennen opinnäytetyön aloittamista, joten tiesin etukäteen millainen järjestelmä oli rakenteeltaan.

Itse ratkaisun suunnittelu ja toteuttaminen olivat melko helppoja. Kursseilla oli opetettu kaikki oleellisimmat asiat, joita opinnäytetyön tekemiseen tarvittiin. Loput asiat kuten käyttöliittymän suunnittelu ja järjestelmän rakenteen mukainen työkalun rakenteen suunnittelu olivat hieman haastavampia, mutta eivät silti lannistavia.

Koin opinnäytetyön dokumentoinnin hyvin vaikeaksi. Vaikka kirjoitustyylini on usein melko virallista sävyllään, en silti kyennyt kirjoittamaan opinnäytetyötä ilman ongelmia. Ajoittain vaivaannuin siitä, onko aihe varmasti riittävän laaja. Vasta kun päätin olla miettimättä onko materiaalini riittävän ammattimaisen kuuloista, alkoi opinnäytetyön dokumentaatio syntymään. Yllätyksekseni opinnäytetyön ohjaajalla ei ollut negatiivista sanottavaa kirjallisuuden sävystä, joten olin pyrkinyt liian hienostuneeseen lopputulokseen.

LÄHTEET

Aho, K. 2014. Viesti 19.5.2014 Versine Oy:n hallituksen puheenjohtajan kuvaus yritysrakenteesta.

Backbone.js. 2014. Viitattu 16.11.2014.

<http://backbonejs.org/> Introduction

Bootstrap in A List Apart No. 342. 2014. Viitattu 26.11.2014.

<http://markdotto.com/2012/01/17/bootstrap-in-a-list-apart-342/>

Git - About Version Control. 2014. Viitattu 16.11.2014.

<http://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

Integrated management systems. 2014. Viitattu 17.11.2014.

<http://www.thecqi.org/Knowledge-Hub/Resources/Factsheets/Integrated-management-systems/>