Opinnäytetyö (AMK)

Tietotekniikan ko.

Hyvinvointiteknologia

2011

Niko Mäkelä

TYÖNSEURANTAJÄRJESTELMÄ

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

Turun ammattikorkeakoulu

Tietotekniikan Ko. | Hyvinvointiteknologia

Kevät 2011 | 58

Ohjaajat: KTT Reetta Raitoharju

Niko Mäkelä

TYÖNSEURANTAJÄRJESTELMÄ

Rutiininomaisten toimintojen automatisointi ja tietojen linkitys tietojärjestelmien avulla parantaa yrityksen tehokkuutta niin toiminnallisesti kuin taloudellisestikin. Projektissa suunniteltiin ja toteutettiin digitaalinen WWW-selainpohjainen tietojärjestelmä forssalaiselle metalliteollisuuden alihankintaan ja teollisuuden kunnossapitoon erikoistuneelle yritykselle. Tietojärjestelmän päällimmäisenä tarkoituksena on helpottaa ja nopeuttaa töiden seurantaan ja tuntityölomakkeiden täyttöön liittyviä päivittäisiä toimia sekä arkistoida töihin liittyvä tieto digitaalisesti paremmin saataville. Työnseurantajärjestelmän toimintoihin sisältyvät mm. töiden, asiakkaiden ja työntekijöiden hallinta, tarjoustietokanta, tuotteet, työvaiheet, huoltokirjanpito, matka-, ylityö- ja aikapankkituntienkirjanpito, päivärahat, työajanlyhennykset, sairaslomat, työtapaturmien kirjaus, henkilökohtaiset muistiinpanot, tulostustoiminnot sekä kommentointi. Työnseurantajärjestelmä toimii myös ilmoitustauluna ja tiedotuskanavana yrityksen sisäisessä tiedotuksessa. Yritys ei ole aikaisemmin hyödyntänyt tietotekniikka työtuntien kirjanpidossa eikä järjestelmää siksi tarvinnut integroida muiden tietojärjestelmien kanssa yhteensopivaksi. Projektissa on huomioitu järjestelmän kehitysvaiheen kaikki osa-alueet: määrittely, suunnittelu, toteutus, testaus, raportointi ja julkaisu. Ohjelman laadinnassa on esitelty toteutetut ja hyväksikäytetyt ohjelmat ja toiminnot sekä niiden toteuttamiseen käytetyt menetelmät.

Järjestelmän selainpohjaisuus ja keveys mahdollistavat järjestelmän käyttämisen kaikilla modernin WWW-selaimen omaavien elektronisten laitteiden avulla. Projektissa on käytetty ketteriä ohjelmointimenetelmiä, Scrumia ja KanBania, jotta projektin etenemisestä syntyisi selkeämpi kuva ja vuorovaikutus toimeksiantajan kanssa säilyisi mahdollisimman aktiivisena. Ohjelmiston kehityksessä on käytetty Git-versionhallintaa kirjaamaan työn etenemistä, tehostamaan ohjelmiston etäkehitystä ja varmistamaan ohjelmiston lähdekoodin tallentamisen.

Työnseurantajärjestelmän käyttöönoton uskotaan säästävän aikaa useita tunteja kuukaudessa yrityksen laskutusta tehtäessä. Järjestelmän uskotaan myös parantavan yrityksen raportoinnin tasoa niin sisällöltään, saatavuudeltaan kuin säilyvyydeltäänkin. Tiedon ollessa digitaalisessa muodossa myös töiden tilastointi ja palautteen anto ovat huomattavasti helpompaa.

ASIASANAT:

työnseurantajärjestelmä, tietojärjestelmä, selainpohjaisuus, CMS

BACHELOR´S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Health Informatics

Spring 2011| 58

Instructor: Reetta Raitoharju, D.Sc. (Econ.& Bus.Admin.)

Niko Mäkelä

WORK MANAGEMENT SYSTEM

Automating rationalized working methods and linking data via information system have been proved to enhance efficiency, both functionally and financially. This project was planned and carried out on a digital information system working on a WWW browser, to be used by a metal industry company in Forssa, Finland. The main purpose of the information is to hasten and to ease the work monitoring as well as aspects regarding the filling of working hour forms. Besides this, documentation of work related information is now digitally easier to obtain. The work management system includes functions such as management tools to maintain work, customers and employees, database for work offer related issues, products, work tasks, bookkeeping maintenance work, travel and overtime bookkeeping, per diem allowance, reduction in working hours, sick leaves, occupational accidents, personal notes, printing and commenting features. The Work Management System also works as a news board and guidance channel inside the company. This report covers all the development stages of the system: definition, design, implementation, testing, documentation and publishing. It commits which applications have been used and how they have been implemented.

This project was assigned by a metal industry company in Forssa, Finland, which employs 11 professionals. The company has not utilised any information technology regarding bookkeeping working hours, which made it easy to deploy.

The system's WWW base and lightness makes it possible to use the system anywhere with a modern WWW browser. Agile software development methods like Scrum and KanBan were used to better brief the progress. A common revision control system called Git was used in the software development to book changes in the work process and to back up the source code.

The deployment of the Work Management System is believed to save a huge number of valuable hours every month especially in issues regarding invoicing. When the information is in a digital form, it makes it easier to give and receive work related feedback and to analyse the work process. The project has given much insight into programming and cooperative software development with a client as well as problems regarding the data security.

KEYWORDS:

work management system, information system, browser based, CMS

SISÄLTÖ

SANASTO

API	Application Programming Interface, ohjelmointirajapinta eli määritelmä, jonka mukaan eri ohjelmat voivat vaihtaa tietoja keskenään.
CSS	lyhenne sanoista Cascading Style Sheets eli porrastetut tyyliarkit. Erityisesti WWW-dokumenteille kehitetty tyyliohjeidenlaji.
DOM-malli	lyhenne sanoista Document Object Model, esitysmalli jonka avulla rakenteellisen asiakirjan rakennemalli esitetään ja tulkitaan (2Kmediat 2010)
HTML	lyhenne sanoista Hypertext Markup Language, suomennettuna hypertekstin merkintäkieli
HTTP	sovellustason protokolla hajautetuille hypermediajärjestelmille, WWW-selainten ja -palvelinten käyttämä tiedonsiirtoprotokolla
JSON	lyhenne sanoista JavaScript Object Notation, yksinkertainen tiedonsiirtomuoto
Ohjelmistokehys	sovelluskehys, ohjelmistotuote joka muodostaa rungon sen päälle rakennettavalle tietokoneohjelmalle
POST	yksi HTTP-protokollan WWW:ssä käyttämistä pyyntötavoista, jolla pyydetään palvelinsovellusta vastaanottamaan tietoa.
SQL	lyhenne sanoista Structured Query Language,kyselykieli jolla relaatiotietokantaan voi tehdä erilaisia hakuja, muutoksia ja lisäyksiä
skripti	komentosarja, tulkattava lyhyt ohjelma
WWW	lyhenne sanoista World Wide Web, internetverkossa toimiva hajautettu hypertekstijärjestelmä.

1 JOHDANTO

Yritykset käyttävät yhä suuremmissa määrin tietotekniikkaa hyödykseen päivittäisissä rutiineissaan. Työpäiviin liittyviä toistuvia toimintoja pyritään nopeuttamaan ja automatisoimaan, jotta varsinaiseen työn tekemiseen jäisi enemmän aikaa ja työvoima saataisiin maksimoitua. Ajan säästämisen lisäksi tietotekniikka auttaa parantamaan dokumentoinnin laatua, tiedon jakelua sekä arkistointia.

Opinnäytetyön toimeksiantona oli suunnitella ja toteuttaa digitaalinen työnseurantajärjestelmä pienelle forssalaiselle metallialan yritykselle. Tässä raportissa kuvataan järjestelmän ja toimintaympäristön eri tekijöitä sekä toteutettujen toimintojen toteuttamiseen liittyviä ongelmia ja etuja. Järjestelmän ensisijainen päämäärä on helpottaa ja nopeuttaa tehtyjen työtuntien dokumentointia sekä arkistoida töihin liittyvät eri raportit digitaaliseen muotoon, jolloin ne saadaan myös paremmin saataville. Järjestelmä on suunniteltu toimimaan myös yrityksen ilmoitustauluna ja tiedotuskanavana liittyen esim. loma-aikojen julkaisemiseen, yms. Työnseurantajärjestelmä on WWW -pohjainen eli se toimii WWW-selainta käyttäen. Tästä syystä se on myös erittäin kevyt ja tehokas kokonaisuus palvelimelle suoritettavaksi että myös helppo sekä tuttavallinen käyttää. Opinnäytetyön tuote, tietojärjestelmä, esiintyy tässä raportissa nimellä Työnseurantajärjestelmä.

Emma Karpin artikkelin mukaan yhä useamman yrityksen liiketoimintakriittiset tietojärjestelmät ovat vanhentuneita eikä niitä uudisteta siinä tahdissa kuin olisi tarpeen. Saman artikkelin mukaan yli 34 % ohjelmistotalo EfiCoden teettämään kyselyyn osallistuneista 358 asiantuntijasta käytti yrityksessään yli viisi vuotta vanhoja järjestelmiä. 26 % kyselyyn vastanneista sanoo järjestelmien olevan vanhentuneita. Artikkelin mukaan tilanteeseen on tulossa muutos, sillä 37 % vastanneista suunnittelee tietojärjestelmäuudistuksia vuoden sisällä ja 28 % seuraavan 3 vuoden aikana. Työnseurantajärjestelmän kehityksessä on käytetty tämän päivän tuoreimpia työkaluja eikä kehitettävyydestä ole tingitty.

Järjestelmää ohjelmoitaessa on pyritty luomaan mahdollisimman pitkäikäinen ja helposti päivitettävä tietojärjestelmä. [1]

Projektinhallinnassa on pyritty käyttää ketteriä projektinhallintamenetelmiä Scrumia ja KanBania. Niitä käytettäessä projektin tilanne on jatkuvasti selvillä ja projektin etenemiseen vaikuttavat tekijät on helposti nähtävissä.

2 MÄÄRITELMÄ JA TOIMEKSIANTO

2.1 Lähtötilanne

Opinnäytetyön toimeksiantaja on forssalainen metalliteollisuuden alihankintaan ja teollisuuden kunnossapitoon erikoistunut yritys. Heidän päätuotteenaan ovat mittojen mukaan valmistetut kaiteet, tasot, raput, lasiseinien rungot, teräsovet sekä erilaiset teräksiset runkorakenteet. Yrityksellä on alalta yli 15 vuoden kokemus ja yrityksen palveluksessa työskentelee 11 metallialan ammattilaista. Toistaiseksi yritys on kirjannut tehdyt työtunnit henkilökohtaisiin paperisiin tuntilomakkeisiin, jotka on kerätty valmiiksi täytettyinä yhteen, kerran viikossa. Täytetyt tuntityölomakkeet tarkistetaan työviikoittain työnjohtajalla, minkä jälkeen ne toimitetaan palkanlaskijalle. Yrityksellä ei ole erillistä palvelinlaitteisto sisäisen yritysverkon ylläpitämiseen, vaan verkko koostuu pelkästään työasemista. Palvelinlaitteiston hankintaan yrityksen sisäverkkoon ei myöskään ole kiinnostusta, joten järjestelmä tulisi ottaa käyttöön ulkoisesta verkosta. Työntekijöiden atk-osaaminen on vain kohtuullisella tasolla, joten yritys kaipaa erittäin helppokäyttöistä ja selkeää käyttöliittymää.

2.2 Tarvekartoitus

Uuden käyttöönotettavan tietojärjestelmän lähtövaatimuksena oli edistää työtuntien, matka- ja päivärahakorvauksien sekä töiden etenemisen kirjausta ja seurantaa. Yrityksen nykyisin käytössä olevat paperiset lomakkeet keräävät kaiken tarpeellisen tiedon, mutta digitaalisessa muodossa niiden täyttöä saataisiin automatisoitua ja niiden tarjoama tieto olisi helposti jäsenneltävissä sekä lajiteltavissa. Kun tieto on selkeästi ryhmiteltyä ja jäsenneltyä se tekee tiedosta huomattavasti helpommin löydettävää jolloin myös tiedon saatavuus paranee.

Eniten apua kaivattiin tehdyistä työtunneista merkittyjen työtuntien lajitteluun laskutuksen ja palkanmaksun välillä. Laskutettaessa asiakasta tehdyistä työtunneista tulisi tuntien olla ryhmiteltynä työn mukaan, kun taas maksettaessa palkkaa työntekijöille, tulisi niiden olla ryhmiteltynä työntekijän mukaan. Paperilomakkeelle tunteja merkittäessä tieto voi olla ryhmiteltynä vain 1 tavalla.

Koska tehdyt työtunnit kirjataan vain henkilökohtaisiin tuntilomakkeisiin, tulisi tietystä työstä asiakasta laskutettaessa kerätä työhön liittyvät tunnit kymmenistä sekaisista ja joskus tiedoiltaan puutteellisista lomakkeista. Koska myös paperisen lomakkeen täyttöä on pyritty yksinkertaistamaan, työstä merkitään lomakkeelle vain työnnumero, jolloin myös numeroa vastaava työ tulisi kaivaa tietoineen arkistoista.

Yritys käyttää ylityötuntien kertymistä rajoittaakseen ns. aikapankkijärjestelmää, jonka ideana on kerätä ylityötunnit pankkiin, josta ne voitaisiin käyttää kirjattua aikaa vastaavana vapaana. Aikapankkituntien merkkaamiseen ei kuitenkaan ole vielä käytössä minkäänlaista erillistä lomaketta, eikä olemassa olevista lomakkeista löydy niille omaa kirjauskenttää, mikä tekee aikapankkijärjestelmästä vaikean käyttää, etenkin työnantajan puolesta. Aikapankkituntien tulisi olla merkittävissä muiden tuntien tapaan tuntilomaketta täytettäessä niin, että aikapankin ajantasainen saldo olisi nähtävissä yhdellä silmäyksellä.

Yrityksen tuotanto-osastolla ei ole omia tietokonetyöasemia käytössään, vaan he jakavat taukotiloissa käytössä olevan työaseman. Tilanteeseen on todennäköisesti kuitenkin tulossa muutos järjestelmän käyttöönoton yhteydessä. Yrityksen sisäinen verkko on myös melko vähäisessä käytössä, eikä siihen ole kytketty lainkaan palvelintietokonetta. Yrityksellä ei siis ole erillistä palvelinlaitteistoa sisäisen yritysverkon ylläpitämiseen, vaan verkko koostuu pelkästään työasemista ja niiden keskinäisistä tiedostojenjaoista. Palvelinlaitteiston hankintaan yrityksen sisäverkkoon ei myöskään ole kiinnostusta, joten järjestelmä tulisi ottaa käyttöön ulkoisesta verkosta. Jos järjestelmä olisi käytössä ainoastaan yrityksen sisäisessä verkossa, tietoturvan

määrä lisääntyisi huomattavasti. Järjestelmään pääsy internetin yli on kuitenkin mahdollista vain, jos järjestelmä on yhteydessä ulkoiseen verkkoon.

2.3 Tutkimustyö

Töiden ja työtuntien seurantaan liittyviä järjestelmiä on markkinoilla monia. Ne ovat kuitenkin usein monimutkaisia ja käyttöliittymältään vanhentuneita mikä tekee niistä myös vaikeita käyttää ja vaikeita kehittää. Mikäli ne eivät ole WWW-pohjaisia, ovat ne usein erittäin kalliita ottaa käyttöön ja niiden kehittäminen on kallista sekä usein myös kiellettyä ja siten mahdotonta. Tällöin ohjelmistoon asetetut rajoitukset rajoittavat yrityksen kirjanpitotoimintaa. Kilpailevat järjestelmät on myös usein suunniteltu vain kirjaamaan työntekijän tehtyjä työtunteja, ei edistämään työnsuorituksen seurantaa. Tämän opinnäytetyöprojektin aikana on kiinnitetty huomiota myös työn, ei pelkästään työntekijöiden seurantaan. Lähes kaikki vastaavat järjestelmät tarjoavat työntekijöiden ja tuotteiden tunnistusta viivakoodinlukijalla. Jotkin ohjelmistopaketit tarjoavat jopa kosketusnäyttöjä. Jotkin markkinoilla olevista tutkituista ohjelmistoista edellyttävät työkonekohtaista ohjelmiston asennusta, mikä vaikeuttaa järjestelmän käyttöönottoa ja ylläpitoa. Kalleimmat markkinoilla olevat ohjelmistot tarjoavat myös integraatioratkaisuja mm. erilaisten suosittujen kellokortti- ja palkanlasku-, murto-, palo- tai videoseurantajärjestelmien kanssa yhdistämiseen, mutta pienillä, etenkin alle 15 hengen yrityksillä ei kyseisille toiminnoille usein löydy tarvetta, jolloin kalliista integraatioista ei myöskään kannata maksaa. [2][3][4][5][6][7]

Järjestelmään liittyvää tutkimustyötä on projektin eteen tehty melko vähän. Tutkimustyön sisältö on ollut lähinnä tarjouspyyntöjen tekemistä muista vastaavista järjestelmistä. Tutkimusta on kohdistettu myös järjestelmää tulevaisuudessa pyörittävään palvelimeen ja virtuaalipalvelimien palvelujentarjoajiin.

2.4 Käyttöympäristö

Töiden suorituspaikka on usein jossain muualla kuin yrityksen toimisto- tai hallitiloissa. Yrittäjät joutuvat usein tekemään töitä pitkien päivien vuoksi myös kotonaan ja huolto- sekä asennustyöt tulee tehdä itse paikalla asiakkaan tiloissa. Työnseurantajärjestelmä toimii WWW:ssä, joten järjestelmän käyttö ei ole paikkasidonnaista, vaan töitä voi hallita ja tarkastella silloin, kun itselle parhaiten sopii. WWW-pohjaisuuden vuoksi myöskään mitään erityistä ohjelmistoa ei tarvitse asentaa työasemiin, vaan järjestelmää voidaan käyttää millä tahansa modernilla WWW-selaimella, jossa on kytkettynä JavaScript- ja evästeominaisuudet. WWW-pohjaisuus tekee kehityksestä ja ylläpidosta helpon, sillä päivitys järjestelmän palvelimessa vaikuttaa välittömästi kaikkiin työasemiin. Esim. uuden järjestelmän version julkaiseminen on nopea ja helppo toimenpide, johon ei tarvita suuria työresursseja, vaan yhden työntekijän 5 minuutin työpanoksen.

3 KEHITYSYMPÄRISTÖ

3.1 Käytetyt ohjelmointikielet

3.1.1 Python

Python on monipuolinen, avoimeen lähdekoodiin perustuva tulkattava ohjelmointikieli, jonka Guido Van Rossum on julkaissut 1990-luvulla. Ensimmäinen versio Python-kielestä syntyi kuitenkin jo 1980-luvun loppupuolella. Sitä pidetään monipuolisena, tulkattavana ohjelmointikielenä sekä helppona oppia sen yksinkertaisen syntaksin ja korkean tason tietorakenteiden vuoksi. Siksi sitä suositellaan usein myös ensimmäiseksi ohjelmointikieleksi. Pythonille on luonteenomaista pyrkiä selkeään ja helposti luettavaan ohjelmakoodiin. The Python Software Foundation eli avoimen lähdekoodin, Python ohjelmointikielen voittoa tavoittelematon omistajajärjestö, mainostaa Pythonin olevan ohjelmointikieli jonka avulla voi työskennellä nopeammin ja integroida järjestelmät tehokkaammin. [8]

Python on alusta riippumaton eli se toimii kaikilla tunnetuilla käyttöjärjestelmillä, esimerkiksi Microsoft Windows, Linux/Unix, OS/2, Mac ja Amiga käyttöjärjestelmillä. Koska Python kuuluu yleisimpiin käytettyihin ohjelmointikieliin, siihen helposti vertailtavia kieliä ovat mm. TCL, Perl, Ruby, Scheme ja Java. Pythonin sanotaan olevan vaikuttanut mm. Ruby-kielen syntyyn. Molemmat kielet, Python ja Ruby, ovat oliopohjaisia ja niissä käytetään vahvaa ja dynaamista ns. ankka tyypitystä. [8] [9]

Projektin versionhallinnassa käytetty Github-sivusto ilmoittaa projektissa Pythonilla kirjoitetun lähdekoodin osuudeksi n. 60 % kaikesta projektiin kirjoitetusta lähdekoodista. Loput n. 40 % ovat tilastojen mukaan JavaScript-ohjelmointikieltä. Pythonia käytetään projektissa kaikkien palvelimen päässä suoritettavien taustatoimintojen suorittamiseen.

3.1.2 JavaScript

JavaScript on alun perin Netscape Communications Corporationin kehitämä pääasiassa WWW-ympäristössä käytettävä komentosarjakieli, jonka tärkein sovellus on lisätä Web-sivuille dynaamista sisältöä ja toiminnallisuutta. Se on yksinkertainen, suoraan html-koodin sekaan kirjoitettava komentosarjakieli, mikä mahdollistaa interaktiivisten toimintojen lisäämisen WWW-sivuille. [10][11]

Yleisimmin JavaScriptiä käytetään selainkomentosarjana nk. asiakaspuolen ohjelmoinnissa (Clientside programming), eli WWW-dokumenttiin liitetyt komentojonot suoritetaan ja tulkitaan asiakaskoneelle. [10][11]

Käyttö kohdistuu yleisimmin eri aputoimintojen kuten virheilmoitusten ja kirjoituskenttien hallintaan ja syötettyjen tietojen tarkastamiseen. Etenkin syötettyä tietoa tarkastettaessa on kuitenkin syytä muistaa, että jos komentosarja suoritetaan asiakkaan koneella, on kentät tarkistettava uudelleen virheelliseltä datalta vielä palvelinpäässä ennen kenttien varsinaista käsittelyä.

Lähestulkoon kaikki selaimet tukevat JavaScript-kielen perusversiota, mutta myöhemmissä versioissa esitellyt edistyneemmät piirteet kuten DOM-mallin mukainen hallinta ovat joillakin selaimilla vieläkin jopa mahdotonta. [10]

JavaScriptiä ja jQuery JavaScript-kirjasto ovat osana kaikkia Työnseurantajärjestelmän WWW-sivuja. JavaScriptin avulla muokataan järjestelmän sivujen ulkoasua mm. luomalla sivuille dynaamisia listoja, välilehtinäkymiä, yms.

3.1.3 HTML

HTML-merkintäkielen avulla kuvataan selaimelle WWW-sivun ulkoasu. Se tunnetaankin erityisesti kielenä, jossa WWW-sivut rakentuvat. Sillä voidaan myös merkitä tekstin rakenne eli esimerkiksi mikä osa tekstistä on otsikkoa ja mikä leipätekstiä. Merkintä tehdään tekstin sekaan kirjoitettavilla elementeillä ja elementeissä olevilla määritteillä. HTML on lyhenne sanoista Hypertext Markup Language. HTML:n historia on alkanut jo vuonna 1989. [12]

3.1.4 CSS

CSS-tyylejä käytetään WWW-sivun ulkoasun muotoilemiseen (esim. lihavoinnit, väritykset, jne.). CSS eli Cascading Style Sheets on erityisesti WWW-dokumenteille kehitetty tyyliohjeiden laji. CSS:ssä dokumentille voi määritellä useita tyyliohjeita, jotka yhdistetään tietyllä tavalla yhdeksi säännöstöksi. [13]

Jotta järjestelmän WWW-sivujen ulkoasua olisi helppoa muokata ja hallita ovat kaikki tyylit pyritty pitämään CSS-tiedostoissa. Joitakin poikkeuksia kuitenkin löytyy esim. silloin kun ulkoasun ei oleteta koskaan muuttuvan ja kun sisällön ulkoasu luodaan dynaamisesti eikä siihen liity kuin vain vähän tyylittelyä.

3.2 Palvelin ohjelmisto

3.2.1 Django

Django on korkeantason Python WWW-ohjelmistokehys joka on suunniteltu tekemään korkealaatuisesta WWW-kehityksestä tyylikästä ja nopeaa. Django on julkaistu avoimen lähdekoodin projektina vuonna 2005 ja sen kehitys on ollut nopeaa ja tehokasta heti sen julkaisupäivästä lähtien. Djangossa on panostettu nopeuden lisäksi paljon myös WWW-sivujen tietoturvaan ja luotettavuuteen. [14] [15]

Ohjelmistokehys on ohjelmoinnin apuväline, jonka tarkoituksena on nopeuttaa uusien ohjelmistotuotteiden valmistusta. Kehys tarjoaa valmiiksi rakennettuja tietokoneohjelman osia, joita ei tarvitse kirjoittaa uudelleen ohjelmistokehityksen aikana – tämä nopeuttaa kehitystyötä. Tavallisesti ohjelmistokehystä ei voi käyttää sellaisenaan suoritettavana ohjelmana, vaan varsinainen toimiva lopputuote saadaan aikaan rakentamalla uusi ohjelma kehyksen päälle. [16]

Django ohjelmistokehys tarjoaa monia ohjelmointikehitystä helpottavia kokonaisuuksia, kuten ilmaisen, mutta varsin kattavan dynaamisen tietokanta ohjelmointirajapinnan. Vaihtoehtoisesti ohjelmoija voi kuitenkin kirjoittaa myös SQL-lauseita, mikäli siihen on tarvetta. Apua saa myös tietokannan sisällön

hallintaan, sillä Django tarjoaa tuotantovalmiin hallintapaneelin sisällön muokkaamiseen järjestelmän valvojille. Djangosta löytyy myös tuki kielikäännöksille, välimuistin hyödyntämiseen ja paljon muuhun. Yhtenä isona kokonaisuutena mainittakoon vielä Djangon oma template-kieli eli esim. HTML-tiedostoissa käytettävä oma merkintäkieli jonka avulla saadaan tehostettua HTML-merkkauskielen tuottamaa sisältöä tarjoamalla mm. silmukkarakenteita ja ehtolauseita.

MVC-arkkitehtuuri

”MVC-arkkitehtuuri (sanoista model, view ja controller) on ohjelmistoarkkitehtuurityyli, jonka tarkoituksena on käyttöliittymän erottaminen sovellusaluetiedosta. MVC-arkkitehtuuria käytetään etenkin graafisten käyttöliittymien suunnittelussa ja ohjelmoinnissa.” [17]

MVC-arkkitehtuurissa ohjelma jaetaan kolmeen osaan: malliin, näkymään ja käsittelijään:

- Model (M) eli malli kuvaa järjestelmäntietomallin. Se voidaan erottaa sovelluksesta niin että muiden sovellus osien ei tarvitse välittää lainkaan sen teknisistä yksityiskohdista. Tietomallia käytetään hyödyksi tiedon tallentamisessa, ylläpidossa ja käsittelyssä. [17] [18]
- View (V) eli näkymä toteuttaa käyttöliittymän. Se käyttää hyödykseen HTML:n ja CSS:n kaltaisia merkkaus- ja tyylikieliä sekä JavaScript kirjastoja käyttöliittymän ulkoasun muodostamiseen ja tietojen esittämiseen käyttöliittymässä. [17] [18]
- Controller (C) eli käsittelijä tai ohjain ohjaa sovelluksen toimintaa sekä yhdistää malli- ja näkymätasot toisiinsa. Se vastaanottaa käyttäjältä tulevat käskyt sekä muuttaa mallia ja näkymää vastauksena niihin. [17] [18]

Django poikkeaa perinteisestä MVC-arkkitehtuurista varsinaisen käsittelyn osalta niin että käsittely tapahtuu itse Djangon toimesta ja kehittäjä voi keskittyä enemmän malleihin, näkymiin ja varsinaiseen esitystapaan. Djangosta

puhuttaessa viitataankin usein ns. MVT-arkkitehtuuriin jossa näkymät (View) sisältävät logiikan mallien käsittelyyn ja asettavat tiedon oikeisiin esityspohjiin (Template). MVC:stä poiketen näkymät eivät siis kerro miten tieto esitetään, vaan määräävät mitä tietoa esitetään ja esitysnäkymät määräävät miten tämä lopullinen näytettävä tieto esitetään, yleisesti HTML-merkkauskieltä käyttäen. [14] [15]

Toiminnallisuus

Perinteinen tietokantaa ja MVT-arkkitehtuuria käyttävä Django-ohjelma koostuu 4:stä eri tiedostosta (Kuva 1):

- Models.py sisältää ohjelmakoodin tietokantataulujen sisältöön liittyen.
- Urls.py sisältää tiedon URL:ien asettamisesta eli hakemistojen tai muun tiedon sekä näiden käyttöön tarvittavan yhteyskäytäntöjen asettamisesta. Käytännössä, urls.py-tiedostossa kerrotaan että kun siirryt WWW-selaimellasi tiettyyn URL:in, sinulle aukeaa tietty näkymä (View).
- Views.py sisältää toimintalogiikan. Views.py:ssä siis käytetään hyväksi models.py:n määrittelemää tietoa tietokannan rakenteesta, ja haetaan tietokannasta tarvittava tieto. Lopulta Djangolle kerrotaan mitä tietoa halutaan esitettäväksi tarjota, ja mikä esitysnäkymä määrää tiedon esitystavan.
- HTML-tiedostot, kertovat varsinaisen tiedonesitystavan HTML-merkkauskielellä määriteltynä. Djangon esityskieltä (Template language) hyödyntämällä voidaan kertoa miten views.py:n tarjoama tieto pitäisi esittää, eli miltä lopullinen WWW-sivu näyttää ulkoasultaan ja sisällöltään.

```
# models.py (the database tables)

from django.db import models

class Book(models.Model):
    name = models.CharField(max_length=50)
    pub_date = models.DateField()


# views.py (the business logic)

from django.shortcuts import render_to_response
from models import Book

def latest_books(request):
    book_list = Book.objects.order_by('-pub_date')[:10]
    return render_to_response('latest_books.html', {'book_list': book_list})


# urls.py (the URL configuration)

from django.conf.urls.defaults import *
import views

urlpatterns = patterns('',
    (r'^latest/$', views.latest_books),
)


# latest_books.html (the template)

<html><head><title>Books</title></head>
<body>
<h1>Books</h1>
<ul>
{% for book in book_list %}
<li>{{ book.name }}</li>
{% endfor %}
</ul>
</body></html>
```

Kuva 1. Esimerkki Djangossa käytettävien tiedostojen sisällön jaosta ja varsinaisesta sisällöstä.

Kehityspalvelimen käyttö

Djangon mukana tulee myös oma kevyt ohjelmistokehitykseen tarkoitettu HTTP-palvelin. Kehitys käyttöön tarkoitettu palvelin ei ole kelvollinen tuotantopalvelimeksi tietoturva syistä, palvelimen toiminnan vakaudesta ja testauksen puutteesta johtuen. Djangon virallisessa dokumentaatiossa todetaan Djangon kehitysryhmän keskittyvän HTTP-palvelimen kehityksen sijaan ainoastaan ohjelmistokehyksen kehittämiseen palvelin puolen ollessa kokonaan heidän liiketoimintasuunnitelmansa ulkopuolella.

Kun kehityspalvelin käynnistetään ja aina kun ohjelmakoodiin tehdyt muokkaukset tallennetaan, kehityspalvelin lataa automaattisesti Python koodin uudelleen, tehden muutosten ohjelmoijan kehitystyön huomattavasti helpommaksi. Tehdyt muutokset tulevat siis heti voimaan ja ohjelmoijan nähtäville. Kehityspalvelimen käynnistyksen yhteydessä myös kaikkien asennettujen moduulien eheys tarkistetaan. Mikäli eheyden tarkistuksessa löydetään virheitä, kehityspalvelin tulostaa virheet nähtäville, mutta ei pysäytä palvelimen toimintaa. [19]

3.2.2 MySQL

MySQL on GNU lisensointiin perustuva SQL standardia noudattava suosittu SQL-tietokannanhallintajärjestelmä. MySQL:ä kehittää ruotsalainen yritys MySQL AB ja sen avoimuuden vuoksi se on asennettu ainakin yli 6 miljoonaan tietokoneeseen. GNU lisensointi mahdollistaa ohjelman asennuksen, käytön ja muokkauksen ilmaiseksi. MySQL:stä löytyy kuitenkin myös kaupallinen lisenssi. [20] [21] [22]

3.3 Versionhallinta ja etäkehitys

Versionhallintaa käytetään, jotta lähdekoodin muutoksille saataisiin luotua nk. palautepisteitä uusia toimintoja kehitettäessä ja testattaessa. Jos järjestelmää ei saadakaan toimimaan niin kuin pitäisi, se voidaan helposti palauttaa sen aikaisempaan tilaan. Versionhallinnan myötä lähdekoodi pysyy varmuuskopioituna ja sen kehitysaskeleet kirjattuina. Työnseurantajärjestelmän lähdekoodia on pidetty versionhallinnan alla koko sen kehityksen ajan.

Työnseurantajärjestelmän versionhallinta on toteutettu ilmaisella ja erittäin suositulla vapaan lähdekoodin ohjelmalla, Git-versionhallinnalla. Git on yksi nykyaikaisimmista, erityisesti avoimen ohjelmistokehityksen yhteisössä suosittu versiohallintajärjestelmä, joka on suunniteltu Linuxin kehittämisen pohjalta kerätyn kokemuksen avulla lähtökohtaisesti suuriin, hajautettuihin projekteihin. Git huomaa muutokset tiedostojen ja tiedostohakemistojen sisällössä ja kirjaa tiedostoon tehdyt uudet lisäykset plus-merkein ja merkkien poistot miinus-

merkein. Kun kehitettävän ohjelman lähdekoodiin tehdään muutoksia, käsketään GIT ohjelman tarkistaa muutokset ja lisätään muuttuneet tiedostot nk. muutosjonoon. Kun jonossa on muokattuja tiedostoja, voidaan ne merkitä ohjelman muutosversioksi. Paketin sisältö ja muutokset on myöhemmin tunnistettavissa versionumerosta ja viestistä joka on kirjoitettava versiointi toimenpiteen (commit) yhteydessä. Kirjatut muutosversiot kirjataan ensin vain työasemalle, mutta ne voidaan lähettää nk. työntökäskyllä (push) GIT-projektin versionhallintapalvelimelle. Uudet palvelimella olevat muutosversiot puolestaan voidaan ladata työasemalle vetokäskyllä (pull). Työnseurantajärjestelmän versioiden säilönä käytetään suosittua sosiaalista ohjelmistonkehitysalustaa GitHubia. [23]

4 TESTAUSYMPÄRISTÖ

Ohjelmistojen laatuun kiinnitetään yhä enenevissä määrin huomiota asiakkaiden ja käyttäjien vaatiessa ohjelmilta enemmän, ja ollessaan aikaisempaa laatutietoisempia. Viimeisen 10 vuoden aikana myös laatujärjestelmät, tapahtumakeskeiset käyttöympäristösovellukset sekä sovellusten tuottaminen yhä useampiin ympäristöihin ovat kasvattaneet testauksen ja sen automatisoinnin tarvetta. Testauksen tehtävänä on tarkistaa ohjelmiston toiminnallisuuden toimiminen ja siten varmistaa ja parantaa kehitettävän ohjelmiston laatua. Ohjelmaa ei kuitenkaan voida käydä läpi täydellisesti, vaan kaiken kattavan testauksen sijaan testauksessa tulisi keskittyä järkevissä määrin mahdollisimman kattavan testauksen toteutukseen.

Testausta voidaan suorittaa seuraavilla tasoilla:

- Moduulitestaus (yksikkötestaus, unit testing) käsittää pienimpien ohjelmayksiköiden, esimerkiksi funktioiden, ja niistä muodostuvien pienten kokoelmien eli moduulien, testauksen. Moduulitestauksen tavoitteena on löytää selvien virheiden lisäksi ristiriitoja yksiköiden ja moduulien määrittelyjen ja toiminnan välillä sekä korjata ne. [24] [25]
- Integraatiotestaus (Integration testing) paljastaa viat rajapinnoissa ja moduulien vuorovaikutuksessa [24] [25]
- Järjestelmätestauksen (System testing). päätavoitteena testata kokonaan integroitua järjestelmää ja tutkia täyttääkö valmis ohjelmisto sen määrittelyn asettamat vaatimukset. [24] [25]
- Järjestelmän integraatiotestaus vahvistaa, että järjestelmän integraatio ulkoisten järjestelmien kanssa vastaa asetettuja järjestelmävaatimuksia. [24] [25]
- Alfa-testaus (Acceptance tai Alpha testing) on simuloitu tai oikea toiminnallinen testaus potentiaalisen käyttäjän tai asiakaan toimesta,

jossa voidaan käyttää myös itsenäistä testiryhmää. Alfatestaus suoritetaan ennen beta-testausvaihetta. [24] [25]

- Beta-testauksessa ohjelmiston betaversiot julkaistaan rajatulle yleisöll, ohjelmistokehitysryhmän ulkopuolelle. Tätä vaihetta jatketaan, kunnes järjestelmän kehittäjä ja asiakas hyväksyvät tuotteen ominaisuudet. [24] [25]

Jos testattava kohde ei läpäise jotakin testivaihetta, tulee kaikki virheet ja ristiriidat paikantaa, korjata ja kohde testata uudelleen. Kyseinen toimenpide on nk. regressiotestausta, jossa kaikki virhettä edeltäneet testit suoritetaan uudelleen varmistaen että uudet muutokset ei ole synnyttäneet uusia virheitä alemmille tasoille. [24]

Työnseurantajärjestelmän kehityksessä ei ole käytetty erillistä testipalvelinta vaan testausalustana käytetään kehitys- ja tuotantopalvelimia. Järjestelmän testaaminen suoritetaan Pythonin yksikkötestaus ohjelmisto kehystä käyttäen, jonka avulla voidaan kirjoittaa automaattisesti yhdellä komennolla suoritettavat testitapaukset. Testitapaukset voidaan ajaa testitapauksittain 1 kerrallaan tai koko testikirjaston kaikki testitapaukset peräkkäin.

Testitapaukset käsittelevät erityisesti uusien työtuntien syöttämiseen liittyviä oikeellisuuden tarkastamiseen liittyviä toimenpiteitä. Ajamalla automaattiset testitapaukset, voidaan hetkessä varmistua järjestelmään tehtyjen muutosten toimivuudesta. Muut järjestelmän testitapaukset liittyvät mm. linkkien ja osoitteiden toiminnan tarkistukseen, eri listausten toimimisen testaamiseen ja tietojen tallentamisen testaamiseen. Testitapauksissa käsitellään niin virheellisiä kuin virheettömiäkin käyttötapauksia. Testitapaukset tulisi ajaa ennen jokaista ohjelmaan kohdistuvaa muutosta julkaistaessa. Kuvassa 2 on esimerkkinä lyhyt yksinkertainen esimerkki kirjoitetuista testitapauksista ja kuvassa 3 testiajon lopputuloksesta.

```
import unittest

class SimpleWidgetTestCase(unittest.TestCase):
    def setUp(self):
        self.widget = Widget('The widget')

class DefaultWidgetSizeTestCase(SimpleWidgetTestCase):
    def runTest(self):
        self.assertEqual(self.widget.size(), (50,50),
                         'incorrect default size')

class WidgetResizeTestCase(SimpleWidgetTestCase):
    def runTest(self):
        self.widget.resize(100,150)
        self.assertEqual(self.widget.size(), (100,150),
                         'wrong size after resize')
```

Kuva 2. Yksinkertainen UnitTest-esimerkki, jossa testataan objektin koon muuttamista.

```
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```

Kuva 3. Testin tulos ilmoitetaan OK- tai virhesanomalla.

5 TUOTANTOYMPÄRISTÖ

Tuotantoympäristössä käytetään Djangon oman kehityspalvelimen sijaan Apache HTTP-palvelinta, johon on asennettu Python- ja WSGI-moduulit. Siirrettäessä järjestelmää tuotantopalvelimelle on tietoturvasyistä syytä vaihtaa tietokantayhteyden luomiseen käytettävät käyttäjänimet ja salasanat. Myös järjestelmän vikailmoitus tila eli debug asetus on kytkettävä pois päältä, sillä järjestelmän suorittamia kriittisiä virheitä ei haluta näyttää järjestelmän käyttäjille. Tarkat vikailmoitukset järjestelmän kriittisistä virheistä muodostavat suuren tietoturvariskin, sillä Djangon automaattisesti luoma virheraportti on sisällöltään erittäin kattava ja saattaa sisältää myös tietoa järjestelmää koskevista tietoturva-aukoista. Virheraporttien sijaan tuotantopalvelimelle tulee luoda yleiset virheiden ilmoitukseen tarkoitetut WWW-sivut. Yleisimmät HTTP-palvelimen suorittamat virhekoodit ovat ”HTTP 404”, joka ilmoittaa siitä että palvelimeen on saatu yhteys mutta pyydettyä sisältöä ei löydetty ja ”HTTP 500”, joka ilmoittaa järjestelmän sisäisestä virheestä.

6 JÄRJESTELMÄN SUUNNITTELUVAIHE

6.1 Päämäärä ja motivointi

Järjestelmän ensisijaisia vaatimuksia on ollut helpottaa työhön käytettyjen työtuntien seurantaa. Ennen järjestelmän käyttöönottoa toimeksiantajayritys on suorittanut kaiken kirjanpidon käsin paperille, jolloin yhteen työhön liittyvä tieto tehdyistä työtunneista on pahimmillaan merkittynä kymmeniin eri paperilomakkeeseen, tieto on paikoittain vaikeasti luettavaa ja mikä pahinta, se on myös helposti vahingossa hukattavaa.

Työnseurantajärjestelmää suunniteltaessa on lähdetty siitä, että järjestelmää olisi mahdollisimman helppoa ja nopeaa käyttää ja että sen käyttö olisi heti ensimmäisellä käytöllä opittavissa. Monimutkaisimpienkin toimenpiteiden tulisi olla helposti löydettävissä ja käytettävissä. Koska työnseurantajärjestelmän ensimmäinen versio on kehitetty räätälöitynä työnä metalliteollisuuden alan yritykselle, on varsin selvää että järjestelmän käyttäjäkuntaan kuuluvat eivät ole tietokoneen tehokäyttäjiä. Tästä syystä käyttöliittymän helppokäyttöisyyteen on panostettu ja esimerkiksi kirjoittamisen määrä on pyritty pitämään mahdollisimman vähäisenä, jotta järjestelmän käyttäminen olisi mielekästä myös niille joilla on vaikea löytää näppäimistöstä oikeita kirjaimia. Päämääränä on myös ollut lyöttää järjestelmään yhteen mahdollisimman suuri toteutettavissa oleva määrä erilaisia työelämän arkipäivän toimintoja.

Järjestelmän kehittämisessä suurin motivaationlähde on ollut halu oppia uutta ja saada aikaan viimeistelty järjestelmä josta on oikeasti hyötyä kehittäjälle itselle kuin sen käyttäjillekin. Koko projekti on alusta saakka ollut erittäin opettavainen, sillä se on antanut todenmukaisen työelämälähtöisen kuvan ohjelmistoprojektin kokonaisuudesta. Uuden ohjelmointikielen ja ohjelmointitavan oppiminen on kannustavaa, kun projektin on saanut paloiteltua mahdollisimman hyvin itselleen sopiviin toteutuskokonaisuuksiin.

6.2 Projektin osapuolet ja projektin hyödyt

Työnseurantajärjestelmä-projektin osapuolet ovat olleet selvillä heti projektin ensi askeleista lähtien. Projektin päällimmäisenä tarkoituksena on työn toimeksiantajan, Fetek Oy:n puolesta ollut auttaa järjestelmän suunnittelussa, toimintojen testaamisessa ja ideoimisessa, tietojen syöttämisen tarkistussääntöjen määrittelyssä ja sitä kautta saada itselleen ideoitua heidän arkipäivän toimintoja mahdollisimman hyvin palveleva järjestelmä.

Projektin ohjelmoijan näkökulmasta projektin tarkoitus on antaa hyvä oikea työelämälähtöinen aihe uuden ohjelmointikielen sekä ohjelmointitavan oppimiseen, josta saa edes hieman taloudellista etua ja josta jää hyvät referenssit sekä toimiva järjestelmä omaan käyttöön kehitettäväksi.

6.3 Suunnittelu käytännössä

6.3.1 Suunnittelussa eteneminen

Järjestelmän suunnittelu lähti liikkeelle työn toimeksiantajan käyttämien lomakkeiden ja käytäntöjen tutkimisella sekä samankaltaisten järjestelmien analysoinnilla. Koska toimeksiantaja yrityksellä ei ole olemassa käytössä olevia tietojärjestelmiä, suunnittelu tuli aloittaa täysin puhtaalta pöydältä.

Järjestelmän projektinhallinta suunniteltiin toteutettavaksi usein ketterässä ohjelmoinnissa käytettävällä Scrum-projektinhallintamenetelmällä. Scrumissa ideana on yhden Scrum-pyrähdyksen aikana suorittaa esimääritettyjä tehtäviä niin, että lopputuloksena on toimiva ja käytettävä kokonaisuus. Järjestelmän kaikkiin kehitysvaiheen uusien toimintojen ohjelmointiprosesseihin sisältyy siis toiminnon suunnittelu, toteutus, testaus, raportointi ja julkaisu. Järjestelmän kaikkea toiminnallisuutta ei siis tarkoitettu suunniteltavaksi kerralla, vaan suunnittelutyötä tehtäisiin lisää aina uutta toimintoa tehdessä tai vanhaa parantaessa. Opinnäytetyöprojektin lyhyeksi rajaaman ajan vuoksi testaus- ja raportointityö on Scrum-pyrähdyksissä jäänyt usein lyhyiksi ja pinnallisiksi, joten testitapauksia ei siis jokaisen uuden toiminnon toteutuksen yhteydessä keritty

kirjoittamaan. Testitapaukset on kuitenkin otettu tarkempaan käsittelyyn kehityksen myöhemmässä vaiheessa, erityisesti testaukseen keskityttäessä. [26] [27]

Järjestelmän kehittäminen aloitettiin kevyellä harjoitusta antavavalla Django-ohjelmalla. Ensimmäiseen Scrum pyrähdykseen sisältyi uuden tyhjän Django-projektin ja -ohjelman luominen. Varsinainen ohjelmoiminen aloitettiin tietomallien luomisella. Päätoimintojen sijaan, ensimmäiset järjestelmään tehdyt tietomallit ja ohjelmakokonaisuudet olivat ilmoitustaulu ja henkilökohtaiset muistiinpanot. Kyseiset ohjelmat toimivat ns. harjoitteluohjelmina, joiden avulla voitiin kuitenkin laajentaa Työnseurantajärjestelmän toimintoja. Tietomalli työntekijöistä oli myös luotava ensimmäisten mallien joukossa, sillä se liittyy olennaisesti lähes kaikkiin järjestelmän muihin tietomalleihin, niin myös ilmoitustaulun ja muistiinpanojen malleihin. Ilman työntekijää muistiinpanosta ei tule henkilökohtainen ja ilmoituksella on oltava aina kirjoittaja eli merkinnän luoja ja julkaisija.

6.3.2 Ilmoitustaulu ja muistiinpano ohjelmat

Ilmoitustaulu ja muistiinpano -ohjelmien ensimmäisten koeversioiden valmistuttua, tietomallien suunnittelua ja luontia jatkettiin työ-mallin ja siihen liittyvien tietomallien kuten asiakkaan luonnilla.

Ilmoitustaulu nimettiin News-ohjelmaksi ja muistiinpano-ohjelma Notes-ohjelmaksi. Molemmat ohjelmat jätettiin siis varsinaisesta tuntijärjestelmä ohjelmasta ulkopuolelle jolloin ne voidaan ottaa myös pois käytöstä, ilman että tunti- ja työjärjestelmän toiminta häiriintyisi. Muistiinpano-ohjelman toiminnot eivät säilö paljoa tietoa, joten sen suunnittelu oli melko yksinkertaista. Kuten kaikille muillekin ohjelmille ja sisällön muokkaus toiminnoille, myös muistiinpanoille ja ilmoitustaulun ilmoituksille tuli luoda näkymät objektin muokkaukselle, objektin näyttämiselle sekä monen objektin listaukselle. Sekä ilmoitukset että muistiinpanot säilövät tiedon merkinnän luojasta, merkinnän ajankohdasta sekä merkinnän sisällöstä. Lisäksi ilmoituksille tuli suunnitella toiminnot ilmoituksen näkyvyyden rajaamiseen. Ilmoituksesta voidaan kirjoittaa

vain luonnos joka ei asetu näkyville koskaan tai se voidaan rajata näkyväksi vain tietyn ajanjakson ajaksi.

6.3.3 Huoltis-ohjelma

Järjestelmän varsinainen ydintoiminta on luotu Huoltis-nimiseen ohjelmaan joka periytyy järjestelmän ensimmäisestä nimestä ja ohjelmiston kokonaisuuteen sisältyvästä toiminnallisuudesta, huoltotietojärjestelmästä.

Huoltis-ohjelman toteuttaminen aloitettiin työskentelemällä työ- ja asiakas-tietomallin kanssa, luomalla HTML-pohja sivujen ulkoasun perustaksi, keksimällä tapa esittää paljon tietoa sisältävät listat järkevästi (toteutettu haitarinäkymänä) ja miettimällä tapoja sivustolla navigoimiseen. Töiden tilausrivit eli ns. työvaiheet on keksitty työ-tietomalliin vasta järjestelmän kehityksen myöhemmässä vaiheessa ja on luotu työ-malliin jälkikäteen South Django-ohjelman migraatiotyökalujen avulla.

Työmallin valmistuttua tuli vuoroon koko järjestelmän tärkein tietomalli eli malli tuntirivistä. Tuntirivin suunnittelun pohjana on käytetty suoraan toimeksiantajan paperista tuntilomaketta. Alkuperäinen paperinen tuntilomake on todettu hyväksi ja kaikin puolin toimivaksi, sillä se kattaa kaiken siltä tuntien kirjaamiselta vaadittavan tiedon eikä siinä ole turhaan merkittäviä tietokenttiä. Myös tottumusten vuoksi on hyvä käyttää vanhan kaltaista tuntilomaketta digitaalisen lomakkeen luonnissa, jotta uusi tuntilomake ei herättäisi hämmennystä erilaisuudellaan vanhaan täysin toimivaan nähden.

Ensimmäinen versio tuntienkirjaustoiminnosta uudessa Työnseurantajärjestelmässä kattoi ainoastaan normaalien tuntien syöttämisen, mutta tarve vaati mahdollisuuden myös matkatuntien, ylityötuntien, aikapankkituntien sekä työlisien merkitsemiseen. Tarve kyseisille ominaisuuksille tulee erityisesti tehtäessä töiden laskutusta, sillä esimerkiksi matkatunnit laskutetaan eri hinnalla kuin normaalit työtunnit. Ylityötunneista puolestaan maksetaan enemmän työntekijälle ja niiden määrää on seurattava myös siksi, ettei niitä tehdä liikaa ja että tiedettäisiin milloin ylityöstä

maksettavan korvauksen määrää tulee korottaa (50 % ja 100 % ylityötunnit). Matkatunneille luotiin matka-tietomalli ja aikapankkitunneille aikapankki-tietomalli. Matkamerkintöihin liittyy aina myös ajoneuvo, joka voi olla joko asiakkaan, yrityksen omassa tai työntekijän henkilökohtaisessa omistuksessa oleva ajoneuvo. Riippuen ajoneuvosta lomakkeessa kerättävän tiedon määrä ja laatu vaihtelee. Tiedot normaalien työtuntien ja normaalista laskutuksesta poikkeavien erikoistuntien määrästä tallennetaan myös tuntirivit tietokantatauluun tietokannan denormalisoinnin vuoksi, eli jotta voidaan vähentää tietokantahakujen määrää. Tietokannan denormalisointi vaikeuttaa tietokannan hallittavuutta, mutta parantaa hakujen tehokkuutta.

Suurimman osan ajasta tuntienkirjaustoiminnon toteutuksessa on vienyt lomakkeeseen syötetyn tiedon tarkastamiseen eli validointiin kirjoitetut funktiot. Koska lomakkeeseen syötettävän tiedon määrä vaihtelee 5 kentästä reiluun 20:een, ovat myös lomakkeeseen syötettyjen tietojen tarkistusfunktiot monimutkaisia, sillä niiden tulisi kattaa kaikki lomakkeen eri variaatiot ja ilmoittaa lomakkeen syötössä tapahtuneista virheellisistä merkinnöistä.

Tuntienkirjaustoimintojen jälkeen järjestelmän kehityksessä on keskitytty huoltotoimintojen ja materiaalitunnisteiden toiminnallisuuden luomiseen. Materiaalitunnisteita käytetään löytämään työssä käytössä olevaa materiaalia käyttävät muut tulevat työt, jotta materiaaleja ei kuljetettaisi ja vaihdettaisi töiden välillä turhaan.

Järjestelmän huoltomerkintä -osa-alue osoittautui vaikeaksi suunniteltavaksi, mutta lopputuloksena päädyttiin toteutukseen jossa luodaan tietomallit laitteista, laitteen osista ja huoltomerkinnöistä. Laite-tietomallia käytetään niputtamaan samaan laitteeseen liittyvät osat yhteen, mutta varsinaiset huoltomerkinnät tehdään osille.

Päätoimintojen ensimmäisten versioiden toteutuksen jälkeen kehityksessä on keskitytty suurimmaksi osaksi järjestelmän ohjelmointivirheiden löytämiseen, käyttöliittymän hiomiseen ja käyttölogiikan parantamiseen. Muita ohjelmoituja toimintoja ovat esimerkiksi palautelomake, karttaominaisuudet tarjoustietokanta,

tulostustoiminnot, hallintasivuston muokatut toiminnot ja paljon pieniä toiminnallisuuksia mm. edellä mainittuihin toimintoihin liittyen.

6.4 Projektinhallinta

Ohjelmistoprojektin alkuvaiheessa projektin hallintaan käytettiin ketterässä ohjelmistokehityksessä yleisesti käytettyä SCRUM-projektinhallintamenetelmää. Välimatkoista, ajan puutteesta ja kulkemisen hankaluudesta johtuen projektinhallinta vaihtui kuitenkin enemmän kanban-menetelmän tyyliseksi.

Ketterässä kehityksessä perusajatuksena on jakaa ohjelmistokehitys lyhyisiin jaksoihin, jotka tyypillisesti kestävät 1 - 4 viikkoa. Kehitysjaksoon sisältyy projektin suunnittelu, ohjelmointi, testaus ja dokumentointi. Kehitysjakson tavoitteena on julkaisukelpoinen ohjelmisto, jonka pohjalta voidaan arvioida uudelleen projektin tärkeys vaatimukset ja aloittaa uusi kehitysjakso.

6.4.1 Scrum

Scrum tarjoaa sovelluskehitykseen mallin, jonka mukaan projektia ohjataan. Scrum ei ota kantaa matalan tason insinöörikäytäntöihin, vaan keskittyy ennen muuta projektin vaiheistamiseen ja jatkuvaan kontrolliin projektin etenemisestä. Scrum on ketteristä kehitysmalleista suosituin. [26] [27] [28]

Yleisesti käytetyssä projektin vesiputousmallissa on yleensä ainakin määrittelijä, suunnittelija, ohjelmoija, testaaja ja projektipäällikkö, joissa projektipäällikköä lukuun ottamatta kussakin roolissa voi olla useampia henkilöitä. Scrum-projektissa esiintyy vain 3 eri roolia: tuotteen omistaja, Scrum-mestari ja tiimi. Tuotteen omistaja vastaa tuotteen ominaisuuksista eli omistaa tuotteen. Scrum-mestarin rooli on huolehtia siitä, että tiimi voi tehdä työtään optimaalisella tavalla. Tiimiin puolestaan kuuluvat kaikki henkilöt, jotka projektia ovat tekemässä. Tiimin sisällä ei tule nimittää mitään tehtäviä tai rooleja vaan siihen vain kasataan tarvittavan osaamisen omaavat henkilöt. Työnseurantajärjestelmäprojektissa rooleja kuitenkin oli vain 2, joten sitä ei alun alkaenkaan pystytty johdattamaan oikein Scrum-mestarin ollessa myös ohjelmointiryhmä. [26][28]

Scrumissa kaikki ihmiset jaetaan kahteen ryhmään: sikoihin ja kanoihin. Sikoja ovat kaikki, joilla on jokin rooli projektissa (tuotteen omistaja, scrum-mestari tai tiimiläinen) suhde projektiin, ja kanoja ovat muut, jotka ovat kiinnostuineita projektista. Nämä voivat olla esimerkiksi ylempää johtoa tai toisen Scrum-tiimin jäseniä. Nimien taustalla on Scrumiin liittyvä vitsi kanan ja sian ravintolan perustamisesta. [26] [27]

6.4.2 KanBan

Japaninkielinen sana kanban on suomeksi näkyvä taulu. Kanban-menetelmä sisältää vain kolme sääntöä, jotka voi osin ajatella Scrum-laajennuksena. Kanbanin kolme sääntöä ovat työkulun näkyvöittäminen eli töiden pilkkominen sopivan kokoisiin tehtäviin sekä niiden näkyvä kirjaaminen, työn etenemistä kuvaavan taulun määrittäminen eli tehtävävaiheiden määrittäminen ja tehtävien läpimenoaikojen kirjaaminen. [29]

6.5 Käyttöliittymän kieli ja käännökset

Työnseurantajärjestelmä on alun alkaen suunniteltu ja ohjelmoitu englannin kielelle ja lopulta päätoimintojen valmistuttua, sisältöä on käännetty vähitellen myös suomen kielelle. Käyttöliittymän monikielisyys tuo ohjelmointiin hieman lisää rutiininomaisia toimenpiteitä: Kaikki käännettäväksi haluttava teksti tulee merkitä käännöskomennoin, niin HTML-esitysasuja tehdessä kuin taustakoodissa Pythonilla. Pienetkin muutokset tekstin sisältöön rikkovat käännöksen, jolloin teksti on käännettävä uusiksi, mikäli se halutaan saada täysin vastaamaan tekstiin tehtyjä muutoksia.

Käännösten luomiseen on olemassa myös muutamia Django ja Python apuohjelmia joista Työnseurantajärjestelmässä on käytetty Rosetta nimistä 3. osapuolen ohjelmaa. Käyttöliittymän kielen voi vaihtaa ns. lennossa järjestelmän sivuja tarkasteltaessa, jolloin tieto käytettävästä kielestä asetetaan evästeisiin josta Django puolestaan osaa tiedon lukea ja siten valita näytölle tulostettavan kielen.

6.6 Jatkokehitys

Opinnäytetyöprojektin puitteisiin ei sisältynyt lainkaan järjestelmän varsinaista käyttöönottoa. Myös järjestelmän testaus on jäänyt erittäin vähäiseksi. Kehityssuunnitelmia etenkin testaukseen ja ohjelmointivirheiden korjaamiseen liittyen löytyy siis pitkälle tulevaisuuteen. Myös muutamia opinnäytetyön aikana suunniteltuja toimintoja jäi kokonaan toteuttamatta ajanpuutteen vuoksi. Järjestelmän kehitys siis jatkuu opinnäytetyöprojektin ja järjestelmän käyttöönoton jälkeenkin. Tavoitteena on kehittää järjestelmästä kokonaisvaltainen yrityksen kaikkea toimintaa tukeva tietojärjestelmä.

Järjestelmän jatkokehitystä on pidetty tärkeässä roolissa koko järjestelmän suunnittelun ajan. On tärkeää että ohjelmiston lähdekoodi pysyy loogisena ja selkeänä koko järjestelmän laajuudelta. Koska sisältö ja toiminnot ovat selkeitä, tulevaisuuden kehitystyötä voidaan tehdä pienillä muutoksilla, eikä aikaa ja vaivaa kulu jo toteutettujen toimintojen tarkasteluun ennen uuden suunnittelua.

7 JÄRJESTELMÄN TOTEUTUSVAIHE

Työ-tietomalli on yksi järjestelmän keskeisimmistä asioista tehtyjen työtuntien ohella. Kirjattavista tunneista ei tule työtunteja ennen kuin ne liittyvät työhön ja työntekoon. Jokainen järjestelmään kirjattava työtunti liittyy siis johonkin työhön. Jokaisella työllä puolestaan tulee olla asiakas.

Työn tuotoksella on aina oltava joku joka työn tuloksesta hyötyy eli joku jolle työtä tehdään. Yrityksen sisäisissä töissä voidaan asiakkaaksi merkitä yritys itse, muulloin työn asiakas on jokin toinen yritys, yhteisö tai muu toimija.

Kaikki merkittävät tuntirivit liittyvät siis aina johonkin työhön ja työn kautta johonkin asiakkaaseen. Näin ollen on järkevintä rajata päätoiminnot tukemaan järjestelmän perusideaa eli tuntien seurantaa työntekijän, työn ja asiakkaan näkökulmasta.

Järjestelmä on tarkoitettu yrityksen kaikkien työntekijöiden käytettäväksi eli sillä on tällöin luonnollisesti monta käyttäjää. Työtunteja merkittäessä tunnit merkitään aina automaattisesti järjestelmään kirjautuneelle käyttäjälle, joka vastaa kirjausta tekevää työntekijää. Peruskäyttäjällä ei myöskään tule olla oikeutta nähdä muiden työntekijöiden tekemiä tunteja, sillä työtuntien sisältö saattaa olla salaista, henkilökohtaista tai ne saattavat sisältää joitain muita erityisjärjestelyitä, joita muiden työntekijöiden ei tarvitse tietää.

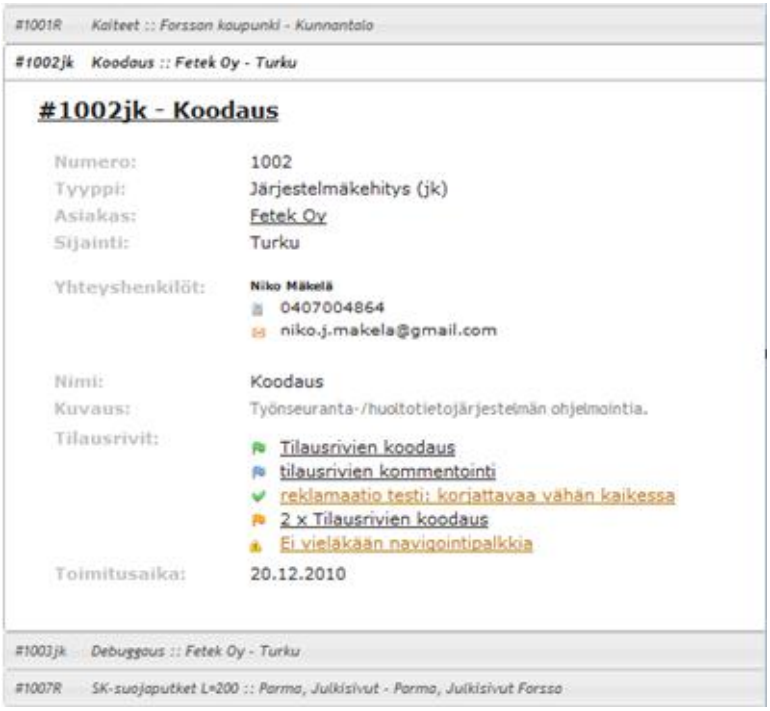
Työnseurantajärjestelmä tarjoaa myös käyttäjäryhmien ja -tasojen määrittelyn. Perusjako käyttäjäryhmien välillä on jakaa käyttäjät peruskäyttäjiin ja ylemmän tason käyttäjiin. Ylemmän käyttäjätason käyttäjät omaavat sellaisia hallintaan ja seurantaan liittyviä toimintoja joita peruskäyttäjillä ei ole käytössä eikä edes nähtävillä. Toisin kuin peruskäyttäjät, ylemmän tason käyttäjät pääsevät näkemään myös muiden työntekijöiden kirjaamia työtunteja. He voivat aina rajata työtuntilistan myös työntekijöiden mukaan, mikä helpottaa esim. laskutuksen tekoa ja yksittäisen työntekijän työaikojen seurantaa.

7.1 Päätoimintojen toteuttaminen

7.1.1 Työt ja työtunnit

Työt-sivulla luetellaan kaikki järjestelmään lisätyt työt. Työt voidaan järjestää niille asetetun tilan mukaan niin, että työt, jotka ovat jo valmiita tai ovat pidossa, listataan eri listaan aktiivisten, eli uusien tai keskeneräisten töiden kanssa. Oletuksena työt lajitellaan kaikki samaan listaan tilan mukaan väreillä erottaen.

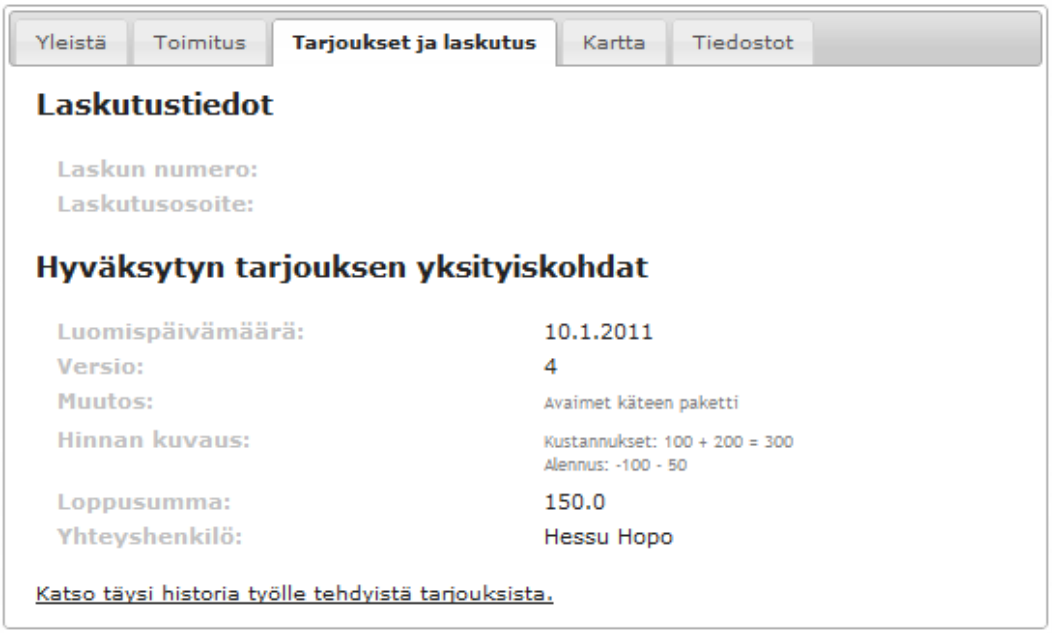
Töiden listaaminen on toteutettu ns. haitarilistana eli työn tiedoista luetellaan aluksi vain otsikkorivi, jota napsauttamalla työstä avautuu työhön liittyvä tietokortti. Haitarin otsikkorivi sisältää työn tunnistamiseen vaadittavan tiedon, ja sitä napsauttamalla aukeava työkortti puolestaa työhön liittyvän tarkemman tiedon (kuva 4). Alleviivattu teksti kuvaa linkkiä työhön liittyviin tieto-objekteihin. Kuvassa 4 asiakkaaksi merkittyä Fetek Oy:tä napsauttamalla siirryttäisiin suoraan asiakasnäkymään. Napsauttamalla tilausriviä siirryttäisiin tilausrivin muokkaussivulle. Eriväriset liput tilausrivien edessä ilmoittavat tilausrivien edistymisasteen.



Kuva 4. Töiden listausnäkymä ja haitarilistaus.

Työkohtaisella sivulla listataan kaikki työhön liittyvä tieto. Myös työn omalla sivulla käytetään haitarilistaa työhön liittyvien tuntien esittämiseen. Kuten ei koko järjestelmässä, eivät myöskään työt-sivulla normaalitason käyttäjät pääse näkemään kuin omat työtuntinsa, kun taas ylemmän tason käyttäjät pääsevät valitsemaan, katselevatko kaikkia työlle tehtyjä tunteja vai vain tietyn työntekijän. Listan tulokset on rajattavissa myös päivämäärän tai aikajakson mukaan.

Koska työ-tietomalli on järjestelmän keskeisimpiä tietomalleja, listataan työ-sivulla erittäin runsaasti tietoa. Työ-sivulla voi työtuntien selailun lisäksi hallita ja katsella työhön liittyviä tilausrivejä, karttapisteitä, tiedostoja, tarjouksia, toimitustietoja sekä kommentteja. Jotta tieto olisi helposti selattavissa, WWW-sivulla käytetään välilehtiä tiedon ryhmittelyyn. Kuva 5 on esimerkki työ-sivun välilehdistä. Täsmälleen samanlaista välilehtinäkymää käytetään myös järjestelmän muillakin tietosivuilla.



Kuva 5. Suuri määrä tietoa ryhmitellään välilehtiä käyttämällä. Esimerkkikuva työ-sivulta, tarjoukset ja laskutus -välilehdestä.

Työ-sivun toteutukseen liittyvät ongelmat ovat kehityksen aikana olleet lähinnä käyttöliittymän käytettävyyteen, tietokannan nopeuteen ja ulkoasuun liittyviä ongelmia. Koska sivulla käsitellään erittäin suurta määrää tietoa, myös tietokannan käyttö on ajoittain raskasta. Työ-tietomallin ollessa keskeinen malli, siihen liittyy paljon tietoa muista tietomalleista mikä kuormittaa tietokantahakua runsaasti. Suurin sivun latausnopeuteen vaikuttava tekijä onkin runsas tietokannan käyttö. Tietokannasta tehtävien hakujen lukumäärää on pyritty pitämään mahdollisimman pienenä kokoamalla tietoa osittain myös tietokoneen välimuistista uusien tietokantahakujen sijaan.

Työmallin sisältö ja tilausrivit

Työ-tietomallin pakollisiin perustietoihin sisältyvät mm. työn numero, nimi ja tehtäväkuvaus. Tehtäväkuvauksen sisältö saa olla mitä tahansa, joten siihen on helppo määritellä kuvauksia esimerkiksi työn eri vaiheista. Työstä on tällöin kuitenkin vaikeata erotella tiettyyn työvaiheeseen käytetyt työtunnit, sillä kirjatuissa työtunneissa olisi viittaus ainoastaan työhön, ei työn vaiheeseen. Työhön voidaan merkitä työvaiheita, jotta voitaisiin pysyä paremmin perillä siitä, miten työ etenee ja mihin osaan työstä kuluu eniten aikaa. Siksi järjestelmään on lisätty tietomalli myös tilausrivistä, joka kuvaa työn vaihetta, reklamaatiotehtävää tai tehtävänlistan merkintää. Tilausrivin tehtävä voidaan kuvata esimerkiksi seuraavilla eri tavoilla:

- "Metalliputki 2mm 20 kpl"
- "Rosterilevy 30 x 30"
- "Kaiteiden kulman korjaus"
- "Uudelleen 10 x 10 kiinnityslevyt, 20kpl".

Tilausrivi-tietomalli pitää sisällään tilausrivin tehtäväkuvauksen, tehtävien toistojen määrän, tilakuvaksen sekä valinnaisen siihen liittyvän tuotteen ja valinnaisen merkinnän siitä, onko tilausrivi reklamaatiomerkintä vai tavallinen merkintä työn vaiheesta. Aivan kuin kokonaiset työtkin, myös tilausrivit lukitaan, kun ne on merkitty valmiiksi. Valmiille tilausriville ei voida enää syöttää uusia tunteja jolloin se näkyy myös työn tilausrivien listauksessa valmiiksi merkittynä.

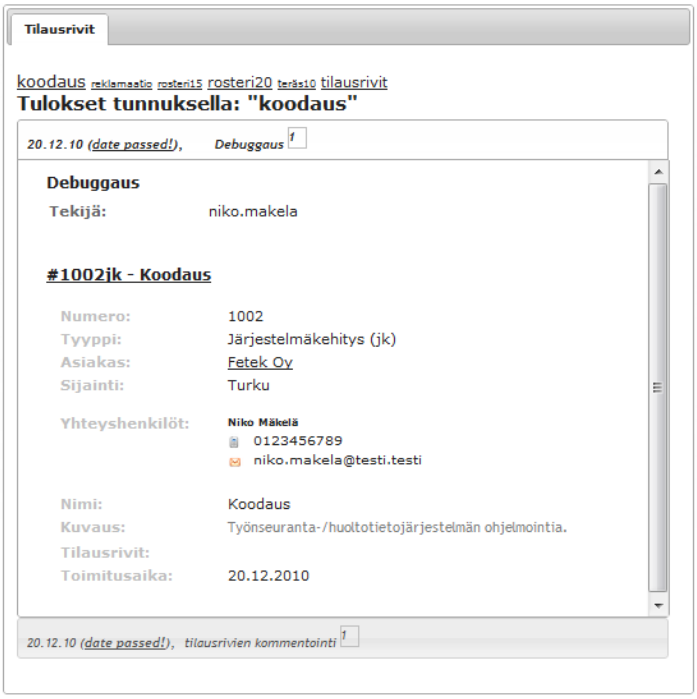
Tilausriveihin tulee luetella myös sen sisällön helpoksi tunnistamiseksi ja hakutoimintoja auttamaan vähintään yksi tunnistetieto eli materiaalitunniste.

Reklamaatiotilaukset/reklamaatiorivit käyttäytyvät täysin tilausrivien kaltaisesti. Vaikka reklamaatiot listataan järjestelmässä tilausriveistä eroteltuina, todellisuudessa reklamaatiorivi on aivan normaali tilausrivi, joka on vain merkitty reklamaatioksi. Siksi myös reklamaatiolla voi olla tilausrivien tapaan merkittynä toistojen määrä ja siihen liittyvä tuote. Reklamaatiotilauksia syötetään samasta lomakkeesta normaalien tilausrivien kanssa, mutta laitetaan merkintä reklamaatio kohtaan.

Työn tilausriveihin on yhtä helppoa merkitä tunteja kuin työhön ilman tilausrivejä. Kun työlle on osoitettu tilausrivejä, voidaan työlle uusia tunteja syötettäessä valita myös tilausrivi, jolle tunnit on tehty. Mikäli tilausrivejä ei ole työlle olemassa, voidaan niitä lisätä työn omalta sivulta tai hallintasivuston kautta.

7.1.2 Tilausrivien materiaalitunnisteet

Tilausrivien materiaalitunnisteet auttavat listamaan tilausrivejä työn suorittamiseen käytettävien materiaalien perusteella. Toiminnosta on suurta hyötyä esimerkiksi vesileikkausosastolla, jossa vaihdetaan materiaalilevyä monesti päivässä työn määritysten mukaisesti. Mikäli samana päivänä tehdään monta työvaihetta johon käytetään materiaalina esimerkiksi rosteria, on paljon aikaa vievää vaihtaa rosterilevy pois vesileikkauskoneesta, jos sitä tarvitaan leikkurissa myöhemmin samana päivänä. Materiaalitunnisteiden avulla voidaan listata päivän samaa materiaalia käyttävät työt samaan listaan (kuva 6). Materiaalien vaihtelun sijaan voidaan heti tarkastaa leikkurissa olevan materiaalin kaikki työtehtävät kerralla, suunnitella töiden suoritusjärjestys ja viedä levy pois vasta sitten, kun kaikki materiaalia tarvitsevat työvaiheet on tehty. Tilausriville voidaan listata useita materiaalitunnisteita kerrallaan pilkulla tai välilyönnillä erotellen.



Kuva 6. Kuvassa on esitettynä hakutulokset töistä haitarinäkymässä ”koodaus” -materiaalitunnisteella haettuna.

7.1.3 Omat työtunnit

Työtuntien kirjaaminen on järjestelmän käytetyin ja merkittävin toiminnallisuus. Työtuntien tekemiseen liittyy monia lainsäädännöllä säädettyjä sääntöjä ja niiden kirjaamisessa on monia erilaisia tapoja.

Työnseurantajärjestelmän tuntirivi sisältää

- kuvauksen tehdystä työstä
- työn ajankohdan
- merkinnän työstä johon tunnit liittyvät
- merkinnän työn tilausrivistä eli tehtävävaiheesta
- matkatuntien määrän
- matkaan liittyvät tiedot, 50- ja 100-prosenttisten päivä- ja viikkoylityötuntien määrän
- raskaantyönlisän merkinnän
- päivärahan tai ateriakorvauksen

- viikkolepokorvauksen
- arkipyhäkorvauksen
- aikapankkituntien määrän.

Työtuntien oikeellisuuden tarkistaminen, on monimutkaista sillä tuntiriviin liittyy monia eri viiteavaimia ja tietokantatauluja. Pienikin muutos tai lisäys tuntiriviin voi vaikuttaa moneen tietokantatauluun kerrallaan. Luomalla sääntöjä tuntirivin täyttämiseen, voidaan varmistua syötettävän tiedon eheydestä ja luotettavuudesta, mutta myös automatisoida lomakkeen täyttöä ja siten saada tuntirivin täyttämisestä mielekkäämpää.

Tuntirivi sisältää muutaman pakosta täytettävän lomakekentän: työn, aloitus- ja lopetusajan, päivämäärän ja tehtäväkuvauksen. Joidenkin lomakekenttien täydentäminen luo myös lisää sääntöjä pakollisuuksista. Tästä esimerkkinä esimerkiksi matkatunnit: Kun matkatuntikenttää syötetään jokin arvo, aukeaa lomakkeeseen lisälomake matkasta täytettävien tietojen täyttöön. Matkalomakeosuus sisältää omat pakollisuutensa matkan suorittamiseen käytetystä ajoneuvosta riippuen. Mikäli ajoneuvo on yrityksen oma, riittää että merkitään reitti, syy ajoneuvon valinnalle, matkan aloitus- ja lopetusajat sekä lähtö- ja lopetusmittarilukemat kilometreinä. Mikäli matka kuljetaan työntekijän omalla ajoneuvolla, tulee matkasta merkitä myös kaikki tiedot käytetystä ajoneuvosta, esim. rekisterinumero, omistaja ja autonmalli.. Jos ajoneuvo taas on asiakkaan, ei matkasta tarvitse merkitä muuta kuin reitti.

Matkan aloitus ja lopetusajan erotus tulee olla sama kuin lomakkeeseen merkittyjen matkatuntien määrä. Matkan ajankohdan tulee myös sijoittua tuntirivin aloitus ja lopetus aikojen väliin. Lopetusaikojen tulee olla merkittyjä samalle päivälle kuin aloitusajat eivätkä ne saa olla pienempiä kuin aloitusajat ovat.

Normaalit työtunnit lasketaan automaattisesti tuntirivin lopetus- ja aloitusajan erotuksesta. Mikäli riville on merkitty erikoistunteja eli ylitöitä, matkatunteja tai aikapankkitunteja, tulee kyseiset tunnit vähentää ennen normaalituntien laskua tuntirivin aikavälistä. Merkittyjen erikoistuntien yhteenlaskettu arvo ei saa siis

olla koskaan olla suurempi kuin tuntirivin aikaväli on. Aikaväli voidaan täyttää myös kokonaan erikoistunneilla jolloin rivin normaalituntien määräksi merkitään 0:00.

Päiväkohtaista 100% ylityötä ei voi tehdä ennekuin samana päivänä on tehty 2 tuntia 50% ylityötä. Viikottaista 100% ylityötä puolestaan ei saa ennekuin on tehty 8 tuntia 50% viikkoylityötä.

Tunnit merkitään vartin tai puolentunnin tarkkuudella. Minuutit eivät kentästä riippuen saa siis poiketa koskaan olla muuta kuin joko 00, 15, 30 tai 45.

7.1.4 Tuntirivin muokkaaminen

Henkilökohtaisia tuntirivejä on myös pystyttävä muokkaamaan ja korjaamaan. Tuntiriviä voidaan muokata siihen asti kunnes se on merkitty laskutetuksi. Laskutetun tuntirivin muokkaaminen sekoittaisi hyväksytyn ja valmiin kirjanpidon. Järjestelmän käyttäjä on erehtyväinen, joten lisätyssä tuntirivissä saattaa joskus olla kirjoitusvirheitä, vääriä kellonaikoja tai virheellisiä erikoistuntivalintoja esim. työn suorituksesta aiheutuvia korvauksia valittaessa. Vaikka tuntirivin syöttöön liittyykin monenlaisia tarkistuksia, ei kaikkia syöttövirheitä ja inhimillisiä erehdyksiä pystytä tarkistamaan. Tuntirivin muokkaamiseen siirrytään napsauttamalla hiirellä tuntirivin otsikkoa tapahtuman ajankohta -merkinnän kohdalta.

7.1.5 Asiakkaalle tehdyt työtunnit

Työtuntirivejä ja töitä voidaan etsiä myös asiakkaan perusteella. Asiakkaan sivulle päästään asiakasluettelosta ja tuntirivien listauksen kautta asiakkaan nimeä napsauttamalla. Asiakassivulla listataan asiakkaalle tehdyt työt asiakkaan toimipisteiden mukaan lajiteltuna ja työn tilanteen mukaan. Tuntirivien luettelointi on samankaltainen kuin muillakin sivuilla, mutta listassa ovat vain kyseiselle asiakkaalle tehdyt työtunnit. Normaalitason käyttäjät näkevät vain omat asiakkaalle merkkaamansa tuntirivit ja ylemmän tason käyttäjät pääsevät valitsemaan työntekijän, jonka tekemiä tunteja he haluavat tarkastella tai listata kaikkien työntekijöiden tunnit yhdellä kertaa. Tunteja

voidaan rajata myös ajankohdan mukaan. Myös kaikkien asiakkaalle tehtyjen tuntien määrä on nähtävillä yhteenlaskettuna.

Asiakkaalle tehtyjen tuntirivien ja töiden lisäksi asiakassivulla näytetään asiakkaan yhteystiedot ja listataan asiakkaan toimipisteet, jotka näytetään myös kartalla Googlen karttapalvelua hyödyntäen. Yksittäiset yhteyshenkilöt listataan toimipisteittäin. Sivulla voidaan selata myös asiakkaalle tehtyjä tarjouksia ja tarkastella mitkä niistä on toteutuneet. Myös asiakassivulle voidaan jättää kommentteja, asiakkaaseen liittyen.

Asiakas-tietomalliin yhdistyy monta muuta tietomallia mikä hankaloittaa sivun luomista ja pidentää sivunlatausaikaa. Kun asiakkaalle tehtyjen töiden määrä lisääntyy, lisääntyy myös esitettävien tuntirivien määrä. Jos asiakkaan työtä tekee esim. 5 työntekijää joista jokainen merkitsee työhön 5 tuntirivimerkintää, tulee yhteen työhön liittyen 25 merkintää. Ongelmaa kaikkien tuntirivien yhdenaikaisesta esittämisestä ei siis synny työkohtaisella sivulla, jossa esitetään tuntirivit vain yhdestä työstä, mutta ongelmia syntyy asiakassivulla, kun siellä esitetään kaikki asiakkaan työt ja niihin liittyvät tuntirivit. Esimerkin mukaisesti 5:n asiakkaalle tehdyn työn jälkeen sivulla listattaisiin 125 tuntiriviä (joissa jokaisessa viitataan useisiin muihin tietomalleihin) mikä on iso määrä merkintöjä näytettäväksi. Tästä syystä kerrallaan näytettävien tai ladattavien merkintöjen määrää tulee myös pystyä rajaamaan.

7.1.6 Tehdyt ja toteutuneet tarjoukset

Työhön liittyviä tehtyjä ja toteutuneita tarjouksia voidaan tarkastella työn sivulla ja asiakkaalle kaikista töistä tehtyjä tarjouksia asiakassivulla. Molempiin sivustoihin sisältyy tarjouksien lisäksi erittäin paljon muuta myös edellä mainittua tietoa, joiden lataaminen pidentää sivun latausaikoja. Koska tarjouksia selataan harvoin eivätkä ne ole tärkeysjärjestyksen huipulla olevaa tietoa, tarjoukset ladataan sivulle Ajax-pyynnöllä vasta niitä pyydettäessä.

Tarjoukset saadaan listattua sivuille ”näytä tarjoukset” -linkkiä napsauttamalla, jonka tuloksena tarjoukset listataan sivujen "tarjoukset"-välilehdille. Tarjouksista

esitetään niihin liittyvä tarjoushinta, yhteyshenkilö, tarjouksen numero sekä tarjouksen luontipäivämäärä.

7.1.7 Huollot

Huollot-sivu on tarkoitettu opinnäytetyön toimeksiantajayrityksen oman ja ylläpitosopimusasiakkaiden kaluston säännölliseen huoltamiseen ja kunnon sekä toiminnan tarkistamiseen sekä huoltojen ylöskirjaamiseen. Koska huoltotöiden kohteena ovat erilaiset koneet, laitteet, ajoneuvot, yms. sekä niihin kuuluvat osat ja varaosat, on huolto-tietomalliin liityttävä myös tietomallit koneesta ja koneen osasta.

Työnseurantajärjestelmä ohjelmoitaessa on päätetty, että tehtävät huollot tehdään aina jollekin laitteen osalle eikä koskaan suoraan laitteelle. Mikäli huoltotyö liittyy laitteen huoltoon osaa mainitsematta tai laite sisältää vain yhden osan, tulee huollettavaksi laitteen osaksi nimetä termi joka viittaa koko huollettavaan laitteeseen, esim. osa jonka nimi on "koko laite". Osaan liittyy tietokenttä joka määrittää osan huoltojen välissä olevan aikajakson päivinä. Jos osan tarkistusajanjaksoksi määritetään esim. 180 päivää, järjestelmä muistuttaa osalle tehtävästä huollosta järjestelmän hälytystaulussa puolenvuoden kuluttua viimeisestä tarkistushuollosta. Osa-tietomalliin sisältyy myös tiedot siitä miten osa tulisi huoltaa: miten osan kunto tarkistetaan, miten se huolletaan ja mitä voiteluaineita osalle käytetään. Kone-tietomalli puolestaan kerää tiedot koneen sijainnista toimipisteen ja toimipisteen osaston mukaan.

Uutta huoltomerkintää lisätessä voidaan valita aloitetaanko osan huollon tarkistusajanjaksopäivien laskenta alusta napsauttamalla "tarkistus"-kenttä valituksi. Laitteen osille voidaan siis tehdä huoltomerkintöjä ilman että ennalta määrättyyn tarkistusajanjaksoon tulisi muutoksia. Esimerkiksi merkittäessä jollekin laitteelle tai laitteenosalle suoritettua voitelua, ei huoltosykliä nollata koska toimenpide ei riitä esimerkiksi vuotuisen huoltotarkistuksen suorittamiseen.

Huoltojen pääsivulla listataan kaikki huollettavat laitteet toimipisteiden mukaan. Listaa voidaan rajata valitsemalla asiakas tai toimipiste rajausvalinkentästä. Yksi listan rivi sisältää laitteen nimen, laitteeseen kuuluvat osat, osan viimeisin huoltopäivä, viimeisin tarkistus sekä seuraavan tarkistuksen ajankohta.

Laitteen nimeä napsauttamalla siirrytään laitteen omalle sivulle jossa huoltomerkinnät on kirjattu tarkemmin ja josta näkee kaikkien laitteeseen liittyvien osien viimeisimpien tehtyjen huoltojen ajankohdat. Osan nimeä napsauttamalla puolestaan siirrytään osan omalle sivulle jossa on listattuna kaikki osaan liittyvät huoltomerkinnät ja niiden ajankohdat.

7.1.8 Profiilisivu

Jokaiselle työntekijälle on Työnseurantajärjestelmässä myös oma profiilisivu jossa työntekijä voi jakaa hänen yhteystietonsa, koulutuksensa sekä voimassaolevat luvat ja lisenssit. Profiilisivulla listataan myös työntekijän henkilökohtaiset aikapankkitunnit, työajanlyhennykset, sairauspoissaolot sekä työtapaturmat. Työntekijä voi myös merkitä aikapankkitunteja, sairauspoissaoloja ja työajanlyhennyksiä. Aikapankkitunteja ei voida merkitä käytettäväksi enempää kuin mitä pankissa on tunteja säilössä merkittynä. Työajanlyhennyksissä puolestaan on sovittu, että niiden kuukausittainen lisääntyminen unohdetaan ja tietokantaan merkitään vain käytetyt lyhennykset.

Profiilisivun toiminnot ovat kaikki yksinkertaisia Djangon automaattisilla tietomallilomakkeilla luotuja toiminnallisuuksia. Vaikein osuus profiilisivun toiminnoiden toteuttamisessa on pankissa olevat työtunnit jotka tallennetaan tietokantaan 2 eri tauluun: tuntiriviin ja aikapankkituntiin. Aikapankkitunnit liittyvät aina yhteen tuntiriviin, mutta tuntien arvo merkitään tietokantataulun denormalisoinnin vuoksi myös aikapankkimerkintään. Pankissa vapaana olevien tuntien määrän tulee siis muuttua kun niihin liitetyn tuntirivin aikapankki tunteja muokataan.

Työtapaturmia ei työntekijä pysty itse merkitsemään, vaan niiden lisäys ja muokkaus on mahdollista ainoastaan hallintasivuston kautta ylemmän tason käyttäjien toimesta.

7.2 Toissijaiset toiminnot

7.2.1 Ilmoitustaulu

Ilmoitustaulu on koko järjestelmän pääsivuna ja tiedotuskanavana käytettävä uutissivu. Ilmoitustaululle voidaan merkitä erityyppisiä ilmoituksia aina saunailloista ja yrityksen pikkujouluista yrityksen tilanne katsaukseen. Ilmoituksesta tai uutisesta on mahdollista luoda etusivulla näytettävä lyhennelmä sekä uutista tarkemmin luettaessa aukeava koko artikkeli. Niihin on mahdollista myös lisätä kuvia, jotka näytetään ilmoitusta tarkasteltaessa.

Ilmoituksia voidaan kirjoittaa myös etukäteen ja asettaa ilmestyväksi automaattisesti näytille valittuna päivämääränä. Niille voidaan asettaa myös voimassaoloaika mikä määrittää ajanjakson jonka ilmoitukset ovat näkyvissä ilmoitustaululla. Vanhoja ilmoituksia voidaan selata ja etsiä myös voimassaoloajan jälkeenkin ilmoitusten arkistosta. Viimeisimmät uutiset listataan myös oikeassa reunassa olevaan sivupalkkiin.

7.2.2 Henkilökohtaiset muistiinpanot

Koska järjestelmän on tarkoitus olla jatkuvassa käytössä päivittäin, se on hyvä väline myös omien henkilökohtaisten muistiinpanojen kirjaamiseen. Muistiinpanot listataan oikeassa reunassa olevaan sivupalkkiin ylimmäiseksi, josta ne ovat helposti löydettävissä. Muistiinpanot näkyvät järjestelmän sivupalkissa ylimpänä jokaisella sivulatauksella ja ovat siten heti luettavissa aktiivisesta sivusta riippumatta.

Muistiinpanoja voidaan lisätä sivupalkkiin niin monta kuin niitä on tarve lisätä "uusi muistiinpano"-painiketta painamalla. Uusi muistiinpano -linkki aukaisee

yksinkertaisen ja nopeasti täytettävän lomakkeen uuden muistiinpanon kirjaamiseen.

7.2.3 Tulostustoiminnot

Järjestelmän tietojen on oltava myös tulostettavissa jotta mahdollistettaisiin tietojen helppo monistaminen ja esittäminen. Työskentelytilat voivat usein olla likaiset ja ahtaat tai muuten sellaiset jossa tietokoneen, kännykän tai muun elektronisen laitteen käyttö on mahdotonta ja tällöin työnkuvauksen on hyvä olla taskuun taitettavalla paperilomakkeella.

Tietojen tulostusmuodon on oltava mahdollisimman yksinkertainen ja helppolukuinen sekä taustakuvaton, yleensä kuvaton ja pelkkää tekstiä ja taulukoita sisältävä lomake. Tästä syystä tulostustoiminnolle on luotava jokaisesta järjestelmän sivupohjasta myös tulostuspohjaversio. Koska järjestelmä käyttää MVC-arkkitehtuuria eikä tulostettavan sivun sisältö poikkea näytöllä näytettävästä WWW-käyttöliittymäversiosta juurilainkaan ei tulostusnäkymän mahdollistamiseksi tarvitse muuttaa kuin sivunesityspohjaa (template). Esityspohjan rakenne muutetaan mahdollisimman yksinkertaiseksi, tekstit vasemmasta tai molemmista reunoista tasatuiksi sekä tekstin väri kokonaan mustaksi.

7.2.4 Hälytystaulu ja kaaviot

Hälytystaulun tehtävänä on varoittaa ennalta asetettujen määräaikojen ja aikarajojen lähestymisestä sekä ilmoittaa uusien kommenttien julkaisuista. Hälytykset listataan oikeassa reunassa olevaan sivupalkkiin josta hälytysilmoitusta napsauttamalla, ilmoitus aukeaa JavaScriptillä toteutettuun ponnahdusikkunaan.

Suurimmassa roolissa hälytystaulu on töiden ja huoltojen kanssa. Uusia töitä syötettäessä, työn arvioitu valmistumis- eli luovutusajankohta on yksi lomakkeen pakollisista syötettävistä kentistä. Hälytystaulu seuraa työn arvioitua valmistumisajankohtaa ja päivämäärän lähestyessä varoittaa sen lähestymisestä. Samaan tapaan toimii myös huoltotaulu ja sen huollettavien

osien huoltoaikajaksot. Kun huoltosykliksi kirjattu ajanjakso on kulunut edellisestä tarkistushuollosta, ilmestyy siitä hälytystauluun huomautus, mikä kertoo, että laite tai sen osa tulisi käydä jälleen huoltamassa ja tarkastamassa.

Töiden arvioitua valmistumispäivää käytetään myös työsivulla aikajanapylväsdiagrammin piirtämiseen, jonka tehtävänä on näyttää kuinka paljon sillä työllä on aikaa jäljellä ja kuinka iso osa arvioidusta ajasta on jo käytetty.

Opinnäytetyön valmistumiseen mennessä hälytystaulua ei ehditty toteuttaa, mutta sen toiminnot ehdittiin suunnitella. Hälytystaulun toiminnot eivät tule olemaan hankalia toteuttaa, sillä se tulee toimimaan lähes samalla tapaa kuin jo käytössä olevat kalenteritoiminnot, joiden avulla rajataan esim. tuntirivien määrää ja aikajaksoa.

7.2.5 Palautteen antaminen

Järjestelmän käyttöliittymän alalaidassa sijaitsee tekstikenttä, jolla voidaan antaa palautetta järjestelmän toiminnasta, sekä ehdotuksia järjestelmän uusiksi toiminnoiksi. Palautteenannosta on tehty mahdollisimman helppoa näyttämällä palautekenttä järjestelmän kaikilla sivuilla, jolloin se on aina saatavilla ja nopeasti täytettävissä. Palautteet kirjataan tietokantaan ja ne lähtevät myös sähköpostiviestillä järjestelmän ylläpitäjille. Palautetta syötettäessä valitaan palautteen tyyppi kehitysehdotuksen ja virheilmoituksen väliltä ja kirjoitetaan vapaan sanan tekstikenttään järjestelmästä annettava palaute.

7.3 Järjestelmässä käytetyt 3. osapuolen ohjelmat

Työnseurantajärjestelmään sisältyy myös muutama kolmannen osapuolen Django-ohjelma. Ohjelmat on lisätty osaksi Työnseurantajärjestelmää niiden integroituessa järjestelmään hyvin ja niiden täydentäessä työnseurantajärjestelmän kokonaisuutta tuomalla siihen kattavan annoksen lisää hyödyllisiä toimintoja. Järjestelmään liitetyt ohjelmat ovat suosittuja Django-ohjelmia joiden kehitys on ollut jatkuvaa ja kiivasta kyseisten ohjelmien

ensiaskeleista lähtien. Ne ovat eheitä ja toimivia, mutta parasta niissä on niiden tulevaisuuden päivitysmahdollisuudet.

7.3.1 Django-taggit

Django-taggit on BSD-lisensoitu eli vapaan ohjelmistolisenssin alainen ohjelma. Lisenssi sallii koodin muokkaamisen ja uudelleenkäytön myös muissa tuotteissa, kunhan lisenssin teksti säilyy lähdekoodissa. Ohjelman päätekijä ja lisensoitu ohjelman omistaja on Alex Gaynor ja projektiin osallistuneet yksittäiset osallistujat. Ohjelman Github.com sivusto on 3.4.2011 ollut yksi ohjelman päälevityskanavista (https://github.com/alex/django-taggit). [30]

Django-taggit ohjelman toimintaa on ehostettu Työnseurantajärjestelmään tunnisteiden syöttön osalta. Siihen on lisätty automaattinen tekstin täydennys toiminto, jQuery:n autocomplete JavaScript-funktiota hyödyntäen.

7.3.2 Django-taggit-templatetags

Django-taggit-templatetags on laajennus Django-taggit ohjelmalle. Se tuo Django-taggit ohjelmaan joukon valmiita esityspohja koodipätkiä (templatetag). Niihin lukeutuu mm. tunnistelistan automaattinen luominen 3 eri tavalla, joista yksi esimerkki on suosittu tunnistepilvi (tag cloud). Ohjelma vaatii Django-taggit ohjelman toimiakseen. [31]

Django-taggit-templatetags ohjelmaa käytetään Työnseurantajärjestelmässä listamaan materiaalitunnisteita ja nopeuttamaan tuntirivien ja töiden hakutoimintoja.

Ohjelman on lisensoitu BSD-lisenssillä ja sen päätekijä ja lisensoitu ohjelman omistaja on Julian Moritz ja projektiin osallistuneet yksittäiset osallistujat. Ohjelman Github.com sivu on 3.4.2011 ollut yksi ohjelman päälevityskanavista. [31]

Oletuksena tunnistepilvi ei sisällä linkkejä mihinkään vaan tunnisteet vain listataan niiden suosion mukaan. Ohjelmaa on muokattu

Työnseurantajärjestelmään sopivaksi lisäämällä hakulinkit listattaviin tunnisteisiin.

7.3.3 Django-reversion

Django-reversion on laajennus Djangoon, joka tarjoaa kokonaisvaltaisen paketin versionhallinan ominaisuuksia. Ohjelman toimintoihin sisältyy mm. tietueiden aikasempien versioiden ja poistettujen tietueiden palauttamisen. Ohjelma on myös integroitu Djangon hallintasivuston kanssa yhteensopivaksi jolloin ohjelmaan saadaan runsaasti lisätoimintoja. Muutoksia voidaan myös ryhmittää jolloin voidaan palauttaa kokonainen ryhmä muutoksia kerrallaan järjestelmän sisällön aikaisempaan tilaan. Tietueiden sisällön muutoksia tallennettaessa Django-reversion tallentaa automaattisesti versiotiedot tietokantaan. [32]

Ohjelman tekijä on David Hall, ja ohjelma on lisensoitu BSD-lisenssillä. Yhtenä ohjelman päälevityskanavana 3.4.2011 on toiminut ohjelman Github-sivu. [32]

7.3.4 Ohjelmat kehitysympäristöön

7.3.4.1 Django-rosetta

Django-rosetta on tehty helpottamaan Djangon kielikäännösprosessia Django-projekteissa. Django-rosettaan ei kuulu mitään tietomallia, joten se ei myöskään luo tietokantatauluja tietokantaan. Se ei ole lainkaan tietokantariippuvainen.[33]

Koska Django-rosetta vaatii kirjoitusoikeudet kielikäännöstiedostoihin, Django-projektissa on ohjelmaan syytä asettaa käyttöoikeuksia. Ohjelmaan suositellaan luotavaksi jokin käännöskäyttäjäryhmä, jolloin vain luotetut ryhmän jäsenet saisivat kirjoitusoikeudet käännöstiedostoihin.

Django-rosetta käyttää ulkoasutyylinään Djangon omaa hallintasivuston ulkoasua. Se listaa kaikki käännettävät sanat ja niiden käännökset sivustolla jossa olemassa olevia käännöksiä voidaan muokata tai lisätä uusia käännöksiä valitulle kielelle.

Django-rosetta on MIT-lisensoitu ja sen lisensoitutekijä on Marco Bennetti.Ohjelman pääasiallinen jakelukanava on code.google.com -sivusto. [33]

7.3.4.2 South

Migraatiot ovat tapa tehdä muutoksia tietomallien muutosten pohjalta tietokantoihin eri tietomallimuutosversioiden välillä. South on ohjelma joka tuo tietokantamigraatiot Django projekteihin. Sen päätehtävänä on automatisoida tietokantataulujen muuntaminen tietomallimuutoksia vastaavaksi. Southin automaattitoimintoja ajettaessa se huomaa muutokset tietomalleissa, luo automaattisesti tietokantataulujen muutoslausekkeet ja ajaa muutokset läpi tietokantaan. South tukee kaikkia Djangon tukemia tietokantatyyppejä. Southin migraatiot ajetaan ohjelmakohtaisesti, joten Django projektin kaikkien ohjelmien ei ole pakko käyttää Southin tarjoamia migraatioita jonkun ohjelman niitä käyttäessä. Mikäli Django ohjelmalla on monta kehittäjää, South huomaa myös eri kehittäjien luomat ristiriitaiset muutokset kirjaten ongelmat ylös vertailussa ratkaistaviksi. South on erittäin hyödyllinen laajennus Django projekteihin ja ketterään ohjelmointiin, se tekee tietokantamuutosten tekemisen todella helpoksi. [34]

South on alun perin Torchboxin vuonna 2008 kehittämä ratkaisu puuttuville migraatio-ominaisuuksille kehitystyössä. Vuoden 2008 DjangoCon-tapahtuman jälkeen se on kuitenkin ollut avoimen lähdekoodin Django-projekti. [34]

South on lisensoitu Apache License 2.0-lisenssillä joka on vapaan ohjelmiston lisenssi jonka on luonut Apache Software Foundation. Se vaatii tekijänoikeuden huomautuksen säilyttämisen ja erottamislausuman. Se sallii lähdekoodin käytön vapaan ja avoimen lähdekoodin kehittämiseen kuten myös omistetun ja suljetun koodin kehittämisen. [34]

7.3.4.3 Django-debug-toolbar

Django-debug.toolbar tarjoaa hyödyllisen testaustyökalun Djangon ohjelman kehitysvaiheeseen. Sen avulla voidaan seurata HTTP-pyyntöjä ja vastauksia,

evästeitä ja istuntoja, lomakkeiden tietojen lähettämistä sekä tietokantakyselyitä. Se myös listaa Python-koodissa lisätyt lokitiedot. Ohjelma esittää tiedot JavaScriptillä WWW-sivuston päälle tehdyssä sivupalkissa jäsennellysti ja helposti löydettävästi.

Ohjelman lisensoitu pääkehittäjä on Rob Hudson ja sen yksi päälevityskanavista on ohjelman Github-sivu. Ohjelma on lisensoitu BSD-lisenssillä. [35]

7.3.5 AJAX-toiminnot

7.3.5.1 Google Maps

Työn toimeksiantajan toimeksiannon pohjalta järjestelmään ei tarvinnut lisätä minkäänlaisia karttoja hyväksikäyttäviä toimintoja. Järjestelmään on kuitenkin lisätty mahdollisuus hyödyntää Google Maps JavaScript ohjelmointirajapinnan tarjoamaa karttapalvelua asiakkaan toimipisteiden ja työn suorituspaikkojen sijainnin merkitsemiseen, sekä matkojen pituuden laskentaan tai tarkistamiseen. [36]

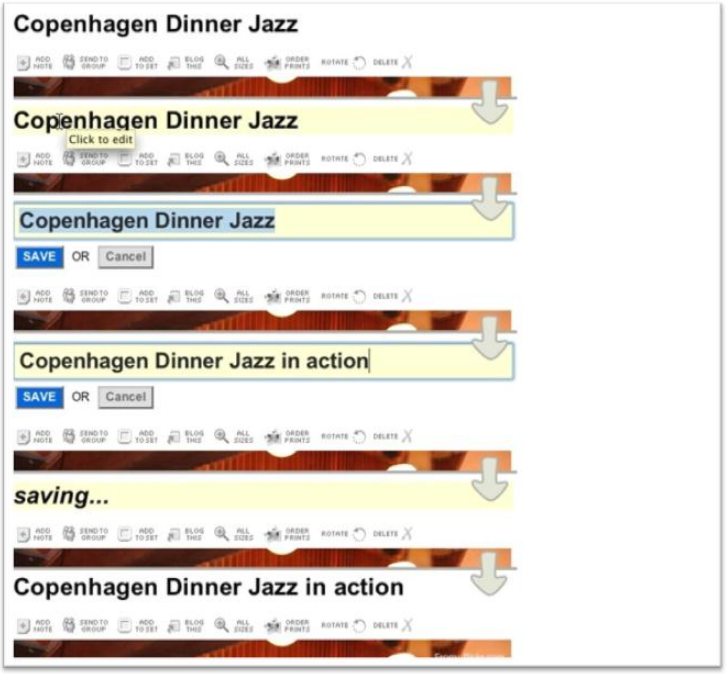
Google Maps -kartat ovat dynaamisia, interaktiivisia karttoja, joilla voi tarkastella mitä tahansa aluetta tai osaa helposti kuin tavallista katukarttaa käyttämällä. Karttaa voi selata hiirellä napsauttamalla ja vetämällä karttaikkunassa. Karttaa voi myös lähentää ja loitontaa liukusäätimin, kartan plus- ja miinusnäppäimiä käyttämällä tai hiiren rullaa kartan päällä pyörittämällä. Sijainnista riippuen Google Mapsista voi etsiä tietyn alueen yrityksiä ja saada niistä näkyviin esim. nimen, osoitteen ja puhelinnumeron. [36]

Perinteisen katukarttanäkymän lisäksi karttoja voi tarkastella myös ns. lintuperspektiivissä hyödyntämällä kartan tarjoamia satelliitti-ilmakuvia. Joissakin kohteissa voi katsella jopa katutason kuvia napsauttamalla ihmiskuvaketta tai sinisellä reunuksella merkittyä katua. Katutasokuvan ympäristöä voi jälleen tarkastella vetämällä panoraamakuvia tai napsauttamalla ikkunan nuolia. Googlen katutasonäkymiä kutsutaan Google Street View'ksi.[36]

Google Maps -karttojen hyödyntäminen Työnseurantajärjestelmässä käy saumattomasti. Asiakkaan uuden toimipisteen lisäyslomakkeessa, matkatunnit lomakkeessa ja työn suorituspisteet lomakkeessa on kaikissa kartan käyttäminen yhtä helppoa kuin hiirellä napsauttaminen. Kun asiakkaalle lisätään uusi toimipiste tai työlle uusi suorituspaikka, karttaa kerran napsauttamalla siihen ilmestyy ns. nuppineula napsautettuun sijaintiin. Napsautuksen yhteydessä Google Maps ohjelmointirajapintaa hyödyntämällä saadaan kartasta poimittua nuppineulan koordinaatit kartalla ja sitä kautta myös katuosoite. Nämä tiedot täydennetään automaattisesti lomakkeeseen napsautuksen yhteydessä ja uutta kohdetta lisättäessä.

7.3.5.2 Jeditable

Jeditable on in place -muokkaustyökalu eli ns. paikallaan muokkaustyökalu jonka avulla voidaan muokata tekstiä ilman erillisiä lomakkeita, siinä paikassa jossa teksti sijaitsee. Kuva 7 esittelee paikallaan muokkaustyökalun toimintaa: Kun napsauttaa hiiren osoittimella "Copenhagen Dinner Jazz" -otsikkoa, teksti vaihtuu muokattavaksi ja tallennettavaksi. Tallennuksen jälkeen muokattu teksti on vaihtunut heti tietokantaan ja WWW-sivulle.



Kuva 7. Jeditable muokkaustyökalun toiminta.

Työnseurantajärjestelmässä Jeditablea käytetään mm. profiilisivulla käyttäjän omien tietojen muokkaamiseen. Vaikka Jeditable onkin innovatiivinen ja toimiva tekstin muokkain, ei sitä voida käyttää joka paikassa. Yksi syy Jeditable-kentien karsimiseen on se että lomakkeiden tulee toimia yhteneväisesti. Jeditable ei myöskään ole toiminnaltaan yhtä vakaa kuin normaali HTML-lomakkeen lähetys ja sivun uudelleen latauksen yhdistelmä. Sen käyttöä on myös hankalampi hallita ja se lisäisi lomakkeiden lähetysten (POST-kutsujen) määrää moninkertaiseksi. Koska jokainen kenttä vaatisi oman käsittelynsä lähdekoodissa, se ei myöskään ole tietoturvaltaan samalla tasolla perinteisen ratkaisun kanssa. Myös sen ulkoasua ja tyyliä on vaikeata hallita, mikäli jeditable-kenttä sijaitsee syvällä monien HTML-elementtien sisällä. Mainion muokkaustyökalun Jeditablesta tekee kuitenkin tilanteet joissa lomakkeiden sisältöä muutetaan useimmiten vain yksi kenttä kerrallaan, jolloin on turha avata erillistä lomaketta ja tehdä sivun uudelleen latausta tai tiedon ajantasaisuuden tarkistusta.

7.3.6 Lisensointi

Työnseurantajärjestelmään liitetyt 3. osapuolen ohjelmat ovat kaikki vapaiden lisenssien alaisia, vapaasti käytettäviä ohjelmia. Lisensiointi on erittäin tärkeää varmistaa, sillä lisenssien sisältö kertoo, miten ohjelmaa saadaan käyttää, jakaa ja julkaista. Järjestelmään liitetyt 3. osapuolen ohjelmat ovat kaikki erittäin suosittujen BSD tai MIT lisenssien alaisia.

7.3.6.1 MIT-lisenssi

MIT-lisenssi on vapaa ohjelmistolisenssi joka on kehitetty Massachusettsin Massachusettsin teknillisessä korkeakoulussa (Massachusetts Institute of Technology, MIT) joka on yksi maailman arvostetuimmista teknillisistä korkeakouluista. Se ei ole copyleft-lisenssi, joten se sallii teoksen käytön myös kaupallisissa suljetun lähdekoodin ohjelmistoissa. [37]

MIT-lisenssin ehdot ovat yksinkertaiset (kuva 8). Se antaa käyttäjälle oikeudet muokata, kopioida ja käyttää teosta vapaasti omassa projektissa sillä ehdolla, että lisenssin teksti säilyy lähdekoodissa. Karkeasti sanottuna ohjelmaa saa siis vapaasti käyttää miten haluaa, kunhan alkuperäinen tekijä käy lähdekoodista ilmi. [37]

```
Copyright (c) <vuosi> <tekijä>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

Kuva 8. MIT-lisenssin sisältöä saa muokata vapaasti.

7.3.6.2 BSD-lisenssi

BSD-lisenssi (kuva 9) on yksi käytetyimpiä vapaita ohjelmistolisenssejä, joka tunnetaan parhaiten Barkley Software Distribution (BSD) lisenssinä. Se antaa käyttäjälle lähes Public Domain oikeudet eli se antaa vapaat oikeudet asettaa teos yleiseen käyttöön ja tekijä luopuu tekijänoikeuksistaan siinä määrin kuin se on lain mukaan mahdollista. BSD-lisenssi ei myöskään vaadi lähdekoodin julkaisemista lisensoitua ohjelmaa edelleen levittämisessä, kuten esimerkiksi GPL-lisenssi vaatii. [38]

```
* Copyright (c) 1998, Regents of the University of California
* All rights reserved.
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions are met:
*
*     * Redistributions of source code must retain the above copyright
*       notice, this list of conditions and the following disclaimer.
*     * Redistributions in binary form must reproduce the above copyright
*       notice, this list of conditions and the following disclaimer in the
*       documentation and/or other materials provided with the distribution.
*     * Neither the name of the University of California, Berkeley nor the
*       names of its contributors may be used to endorse or promote products
*       derived from this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND ANY
* EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
* DISCLAIMED. IN NO EVENT SHALL THE REGENTS AND CONTRIBUTORS BE LIABLE FOR ANY
* DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
* (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
* ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
* (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

Kuva 9. BSD-Lisenssin sisältö

7.4 Järjestelmän oma JavaScript ohjelmointirajapinta

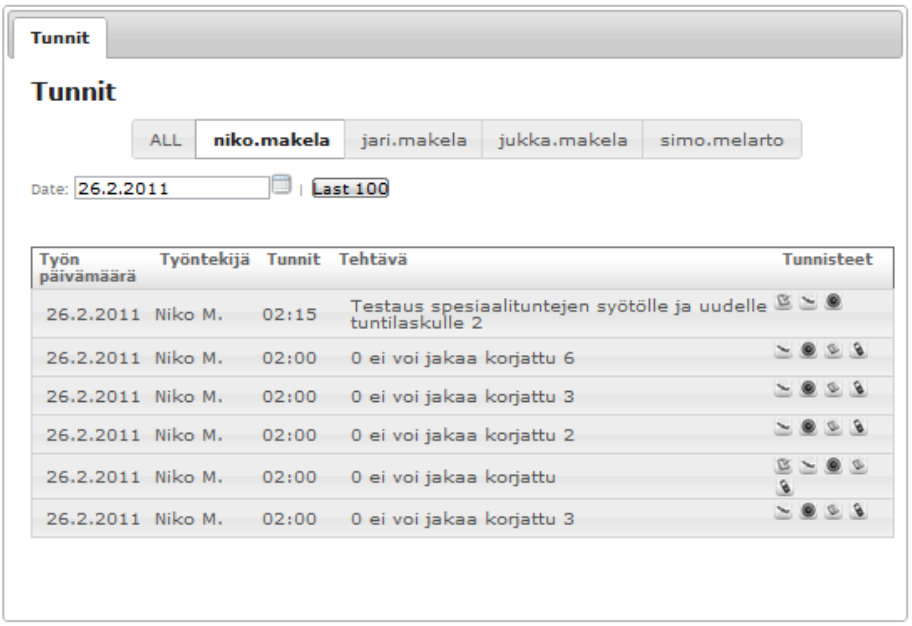
Työnseurantajärjestelmään on kirjoitettu pieni JavaScript-ohjelmointirajapinta dynaamisen ja interaktiivisen tiedon lataamiseksi. Tällaisia toimintoja ovat esim. hakutulosten kuten työtuntien rajaus tietyltä aikaväliltä, vain tietyn työntekijän tuntien näyttäminen, työhön liittyvien tarjousten listaaminen ja karttapisteiden tallentaminen.

7.4.1 Kalenterin käyttö

Syöttökenttien täytön apuna käytettävä jQuery:n kirjastoja apuna käyttäen toteutettu kalenteri toimii apuna listauksien sisällön ajanjaksojen rajaamiseen ja päivämäärien valitsemiseen. Kalenterin päivää napsauttamalla lähetetään JavaScript valitun päivämäärän sisältävä POST-pyyntö palvelimelle, joka palauttaa päivää vastaavan tiedon esitettäväksi. Palautettava tieto on toteutettu sekä kaikille järjestelmille sopivalla JSON tiedonsiirtomuodolla että valmiina HTML-muotoisena jQuery haitarilistana. Käytännössä

7.4.2 Suodatus työntekijällä

Järjestelmän ylemmän tason käyttäjille on oletuksena asetettu käyttäjäoikeudet joiden avulla he voivat tarkastella myös muiden työntekijöiden merkkaamia työtuntirivejä. Tällöin heidän käyttöliittymään ilmestyy myös JavaScriptillä ja AJAXilla toteutettu työntekijäsuodatin työkalu, jota voidaan käyttää suodattamaan listattavaksi rajatun aikavälin tuntirivit ainoastaan valitulta työntekijältä. Kuvassa 10 listattaviksi tuntiriveiksi on rajattu käyttäjän ”niko.makela” tunnit yhdeltä päivämäärältä.

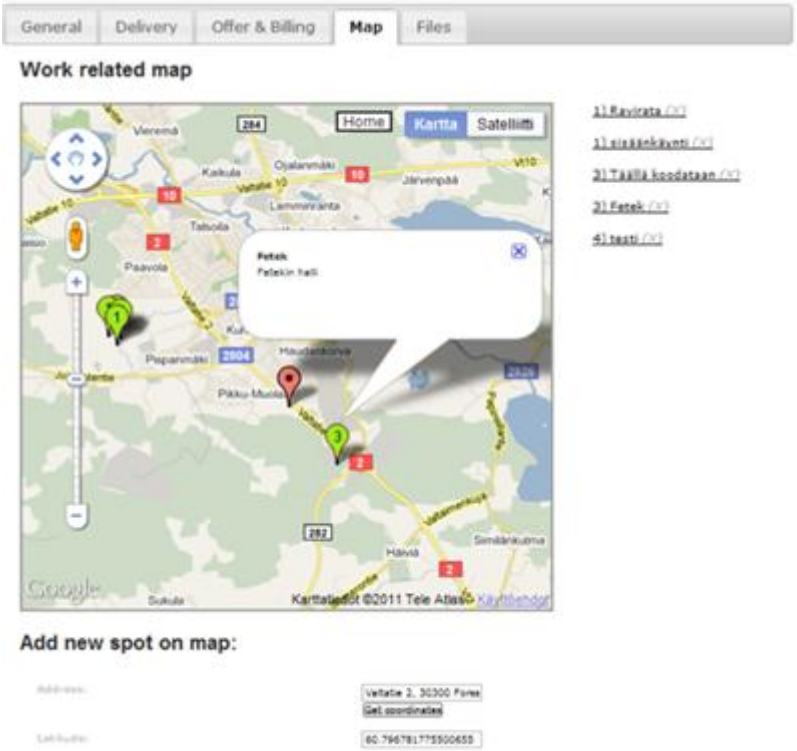


Kuva 10. Näytettävien tuntirivien suodattaminen työntekijän mukaan.

7.4.3 Karttapisteet

Karttapisteillä voidaan merkitä työhön liittyviä sijainteja kartalle. Samaa karttaominaisuutta käytetään myös näyttämään asiakkaan toimipisteiden sijainnit kartalla, mutta asiakkaan toimipistekarttaan ei voida luoda uusia karttapisteitä koska toimipisteitä voidaan lisätä vain järjestelmän hallintasivulta tiedon luotettavuuden varmistamiseksi ja turhien merkintöjen estämiseksi.

Kuvassa 11 on merkittynä karttaan 5 työhön liittyvää pistettä. Pisteet voidaan numeroida, jolloin numeroa voidaan hyödyntää merkkaamaan pisteissä käynti järjestystä tai merkitsemään samanarvoiset karttapisteet. Vihreät merkkaukset ovat avattavissa puhekuplaan vihreää merkkiä tai oikeassa reunassa sijaitsevasta listassa olevaa kohdetta napsauttamalla. Uudet pisteet merkitään karttaan punaisella merkillä.



Kuva 11. Työhön liittyvät karttapisteet.

7.4.4 Hallintasivusto

Djangon mukana tuleviin lisätyökaluihin sisältyy yksi Djangon tehokkaimmista ja hyödyllisimmistä ominaisuuksista, automaattinen järjestelmän hallintasivusto. Hallintasivusto lukee tietomallien meta-tietoja ja tarjoaa niiden pohjalta automaattisen tuotantovalmiin käyttöliittymän järjestelmän sisällön hallintaan.

Käyttö

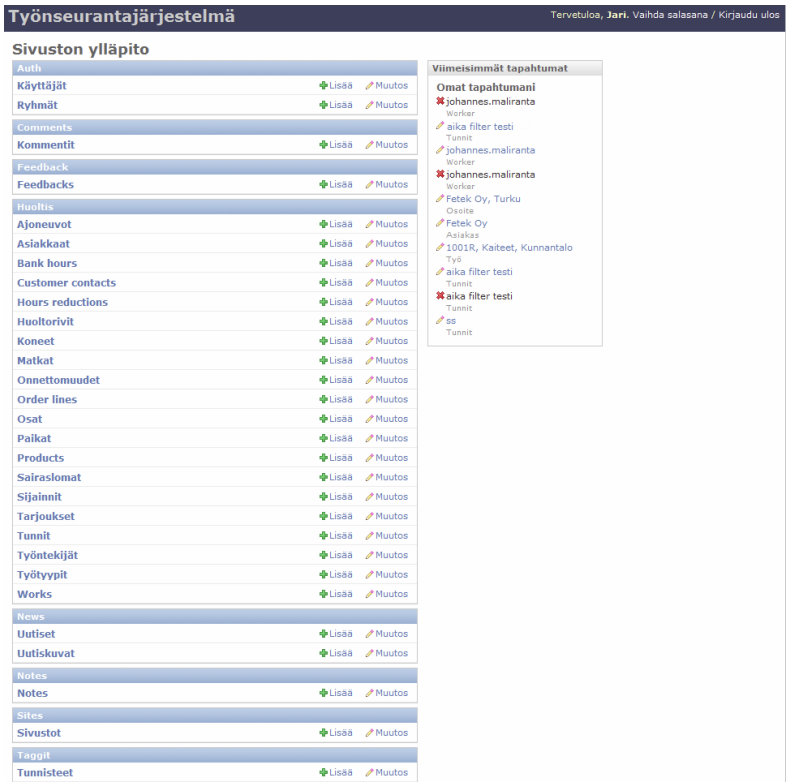
Vaikka myös hallintasivuston käyttöoikeuksia voidaan määritellä niin että kaikkien ohjelmien ja mallien hallitseminen ei ole kaikkien käyttäjien ulottuvilla, ei sitä tule käyttää ilman varsinaista ohjelmaa ja käyttöliittymää. Hallintasivusto on tarkoitettu käytettäväksi vain ylimmäntason käyttäjien tietojärjestelmän hallintaan ja aina hallintatyökaluna varsinaiselle Django-ohjelmalle. Hallintasivusto on tarkoitettu kaiken järjestelmällä kerätyn tiedon näyttämiseen ja kaiken tietokannan ja ohjelmien tallentaman tiedon muokkaamisen mahdollistamiseen.

Django-ohjelma rekisteröidään hallintasivuston käyttöön yleisesti admin.py nimisellä tiedostolla mikä sijaitsee ohjelman kansiorakenteen juuritasolla. Admin.py tiedostossa kerrotaan mistä tietomallista sivu luodaan ja mitä sisältöä siellä päästään hallitsemaan.

7.4.4.1 Hallintanäkymät

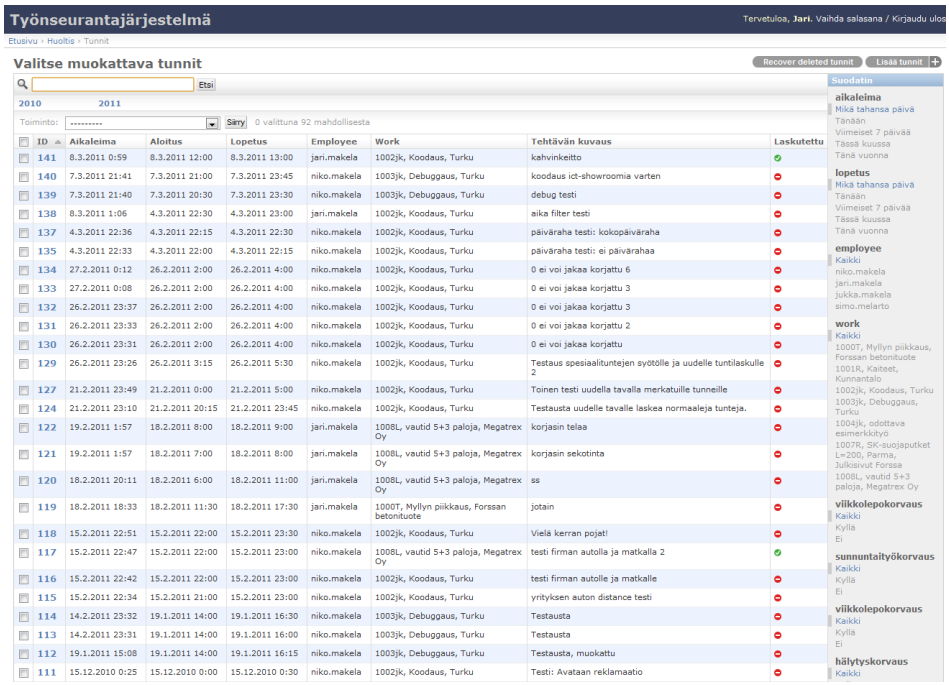
Hallintasivusto koostuu vakiona kolmesta eri listausnäkymästä: pääsivusta joka listaa hallintasivustoon rekisteröidyt tietomallit ohjelmittain lajiteltuna, tietomalliin kohdistuvasta sivusta jossa listataan kaikki tietomalliin liittyvä tietokanta sisältö, sekä sisällön muokkaus- ja lisäysnäkymästä joka näyttää tietomallia vastaavan lomakkeen, jonka avulla voidaan syöttää uusia tietueita tietokantaan.

Kuvassa 12 on kevyesti muokattu Djangon päänäkymä, jossa näytetään historialokia järjestelmän sisältöön koskevista muutoksista, sekä listataan hallintasivuston käyttöön rekisteröidyt järjestelmän tietomallit.

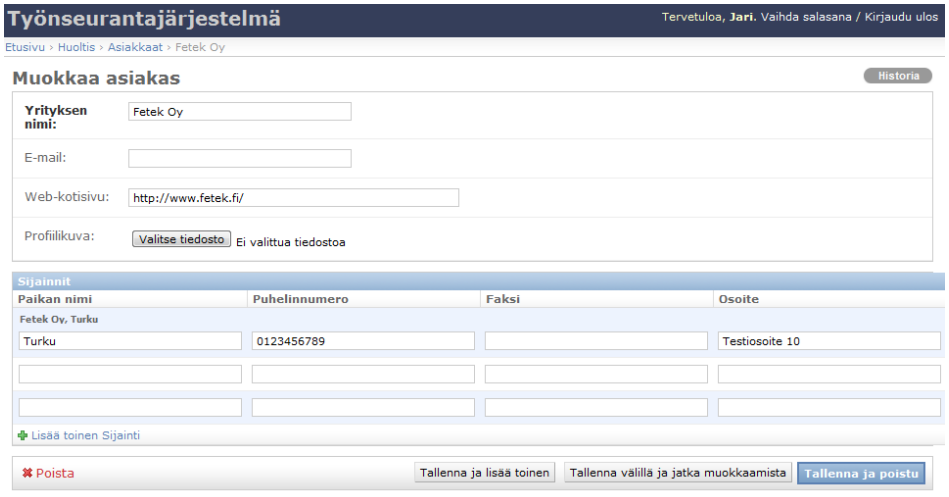


Kuva 12. Hallintasivuston päänäkymässä listataan hallintasivustolla hallittavat tietomallit Django-ohjelman mukaan eroteltuina.

Tietomallia napsauttamalla käyttöliittymään luetellaan tietokannan sisältö valittua tietomallia vastaavasta tietokantataulusta. Listausnäkymässä voidaan rajata esitettävää sisältöä hakusanoilla, erilaisilla aikarajauksilla, sekä sisällön suodattimilla. Suodattimiksi voidaan valita mikä tahansa tietokantakenttä, joka ei ole tietotyypiltään tietokannassa vierasavain. Kuvassa 13 on esimerkki tuntirivien listauksesta. Napsautettaessa rivin id-sarakkeessa olevaa tunnistenumeroa, aukeaa rivin muokkaussivu, josta on esimerkkinä kuva 14 asiakastietueen muokkauksesta. Vastaava sivu aukeaa myös uutta tietuetta luodessa.



Kuva 13. Hallintasivuston tuntirivien listaussivu.



Kuva 14. Hallintasivuston asiakkaan muokkaussivu.

7.4.4.2 Hallintasivuston muokkaus ja kehittäminen

Kuten kaikki muukin Djangon sisältö, myös hallintasivusto on kokonaan muokattavissa. Hallintasivuston yksittäisiä sivuja on helppo ylikirjoittaa luomalla Django-projektin esityspohjien (Templates) kansioon sopivankaltainen kansiorakenne. Ylikirjoittaessa hallintasivuston esitysasuja tulee

esitysasupohjien kansiorakenteen juureen eli ylimmälle tasolle luoda admin niminen kansio. Admin kansion juureen luoduilla tiedostoilla voidaan ylikirjoittaa Django-paketin admin-kansion juuren tiedostoja nimeämällä tiedostot niiden alkuperäisillä nimillä. Hallintasivuston ohjelmakohtaisia sivuja voidaan ylikirjoittaa luomalla admin-kansioon uusi kansio ylikirjoitettavan Django-ohjelman nimellä.

8 TULOSTEN TARKASTELU

Ketterän ohjelmointitavan tuomat nopeat ohjelmointijaksot toimivat projektissa mainiosti. Uusia ideoita toisiin toimintoihin heräsi usein vasta toista toimintoa toteutettaessa, jolloin ketterä menetelmä mahdollisti myös aikaisemmin tehdyn toiminnon uudelleen suunnittelun ja parantamisen. Tietokantamigraatiotyökaluilla myös tietokannan rakenteen muuttaminen oli helppoa ja mielekästä. Kuitenkin juuri suunnittelua ja etenkin testailua olisi kaivattu hieman lisää jokaiseen toteutettuun toimintoon liittyen. Kun perusteellinen ja luotettava testausprosessi uupuu kehityksestä, toimintojen toimivuus jää usein epävarmaksi. Toiminnoista ilmenikin ohjelmointivirheitä usein vasta toista toimintoa toteutettaessa, jolloin uusien toimintojen toteutus keskeytyi vanhan toiminnon korjaamiseen keskityttäessä.

Python ja Django olivat järjestelmän kehitykseen ehdottomasti oikeat työkalut. Tukea löytyi riittävästi sekä digitaalisessa että paperisessa kirjallisuudessa. Django vaikutti uutuudestaan huolimatta tuotantokäyttöön valmiilta ohjelmistokehykseltä, eikä se tai mikään mukaan järjestelmän kehityksessä käytetty ohjelma rajoittanut ratkaisevasti kehittämistä. Rajoittavin tekijä oli SQL-tietokannan käyttö, sillä jokainen sivulataus jakaa järjestelmässä suuren määrän tietoa, ja siten myös pakotti miettimään tehtävien SQL-kyselyiden määrää. Runsas tietokannan käyttö hidastaa sivujen latautumista merkittävästi. JavaScript mahdollisti tehokkaan reaaliaikaisten käyttöliittymän toimintojen toteuttamisen ja sivujen vaiheittaisen lataamisen, mutta lisäsi ongelmia järjestelmän jatkokehittämiseen ja tietoturvan ylläpitoon liittyen.

Rasittavimmat ongelmat liittyivät järjestelmän monikielisyyden tukemiseen. Käännöstoiminnot olivat osittain sekavasti dokumentoituja ja jatkuva tekstin muuntelu pakotti luomaan myös uudet käännöstiedostot. 3. osapuolen valmistama Django-Rosetta -ohjelma auttoi käännösten tekemisessä, mutta käännösten kohdistaminen oikeaan osaan tekstiä oli silti ajoittain arpapeliä.

Kokemus 3. osapuolen ohjelmista oli kaiken kaikkiaan erittäin kiitettävää. Ohjelmat sulautuivat järjestelmään paikoittain jopa erittäin hyvin, ja kaikkien ohjelmien kanssa selvittiin pienin muutoksin.

9 YHTEENVETO

Opinnäytetyössä toteutettiin kehittyneillä ja tuoreilla työkaluilla metalliteollisuuden alan yritykselle räätälöity tietojärjestelmä ratkaisu työnseurantaan. Työnseurantajärjestelmän käyttöönoton uskotaan säästävän aikaa useita tunteja kuukaudessa yrityksen laskutusta tehtäessä. Järjestelmän uskotaan myös parantavan yrityksen raportoinnin tasoa niin sisällöltään, saatavuudeltaan kuin säilyvyydeltäänkin. Koska tieto on digitaalisessa muodossa myös töiden tilastointi ja palautteen anto on huomattavasti helpompaa. Projekti on myös opettanut paljon ohjelmistokehitykseen ja ohjelmointiprojekteihin liittyen mm. asiakkaan kanssa tehtävän yhteistyön tärkeydestä, tietoturvaan liittyvistä ongelmista ja uusien ohjelmointitapojen käytöstä.

Toteutetut toiminnot on pyritty pitämään helposti järjestelmästä irrotettavina kokonaisuuksina, jolloin ne ovat suoraan käyttöönotettavissa tai ainakin yksinkertaisesti kopioitavissa myös muihin projekteihin. Järjestelmän jatkokehitysmahdollisuuksiin on panostettu koko kehittämisen ajan, jotta se olisi helppoa myös tulevaisuudessa. Kehittämistä helpottaa kolmikerrosarkkitehtuuri ja selkeä ohjelmarakenne, mutta myös se, että ohjelmakoodissa ei ole tyydytty tekemään kompromisseja missään vaiheessa. Järjestelmän kehitystä tullaan jatkamaan sekä harrastussyistä että taloudellista hyötyä tavoitellen. Jatkokehitysoikeudet säilyvät sekä järjestelmän toteuttajalla että asiakkaalla eli toimeksiantajalla. Tulevaisuuden kehitysongelmina saattaa esiintyä ainakin työhön käytettävien resurssien pula kuten tapahtui myös opinnäytetyöprojektin aikana.

LÄHTEET

[1] Emma Kauppi, Tietoviikko, 1.2.2011, ”Yritysten tietojärjestelmät ovat kivikautisia” [WWW-dokumentti]. Saatavilla: http://www.tietoviikko.fi/kaikki_uutiset/article571309.ece. (Luettu 16.5.2011)

[2] TeamexTime Controlin Kuvaus ohjelman toiminnasta -esite, TT-Teamex Oy 2009

[3] Jotbar Solutions Oy, ”Työajanseuranta” [WWW-dokumentti]. Saatavilla: http://www.jotbar.fi/tmp_jotbar_site_1.asp?sua=1&lang=1&s=4&q=9fi34h. (Luettu 16.5.2011)

[4] Likeit Solutions, ”Likeit Tasks” [WWW-dokumentti]. Saatavilla: http://www.likeit.fi/tasks.php. (Luettu 16.5.2011)

[5] Visma Avendo, ”Tuntikirjanpito” [WWW-dokumentti]. Saatavilla: http://www.vismaavendo.fi/art-4.asp?id=5&iPageID=9. (Luettu 16.5.2011)

[6] Visma Severa, ”Työnohjaus ja työajanseuranta” [WWW-dokumentti]. Saatavilla: http//www.severa.com/fi/psa/ratkaisu/tuntikirjaus-kuluseuranta. (Luettu 16.5.2011)

[7] Visma Severa, ”Asiakastyytyväisyyskysely” [WWW-dokumentti]. Saatavilla: http://www.severa.com/fi/psa/automatisointi. (Luettu 16.5.2011)

[8] Python 2.7.1 Documentation, Python Software Foundation, 2011

[9] Wikimedia Foundation, ”Python” [WWW-dokumentti], (2.3.2011). Saatavilla: http://fi.wikipedia.org/wiki/Python,. (Luettu 16.5.2011)

[10] Koulutus- ja konsultointipalvelu KK Mediat, ”JavaScript” [WWW-dokumentti]. Saatavilla: http://www.2kmediat.com/jscript/johdanto.asp. (Luettu 16.5.2011)

[11] Wikimedia Foundation, ”JavaScript” [WWW-dokumentti], (27.4.2011). Saatavilla: http://fi.wikipedia.org/wiki/JavaScript. (Luettu 16.5.2011)

[12] Wikimedia Foundation, ”HTML” [WWW-dokumentti], (4.4.2011). Saatavilla: http://fi.wikipedia.org/wiki/HTML. (Luettu 16.5.2011)

[13] Wikimedia Foundation, ”CSS” [WWW-dokumentti], (30.4.2011). Saatavilla: http://fi.wikipedia.org/wiki/Cascading_Style_Sheets. (Luettu 16.5.2011)

[14] Django Software Foundation,”Django” [WWW-dokumentti]. Saatavilla: http://www.djangoproject.com/. (Luettu 16.5.2011)

[15] The Definitive Guide to Django: Web Development Done Right, Apress 2007

[16] Wikimedia Foundation, ”Ohjelmistokehys” [WWW-dokumentti], (1.2.2011). Saatavilla: http://fi.wikipedia.org/wiki/Ohjelmistokehys. (Luettu 16.5.2011)

[17] Wikimedia Foundation, ”Ohjelmistokehys” [WWW-dokumentti], (3.4.2011). Saatavilla: http://fi.wikipedia.org/wiki/MVC-arkkitehtuuri. (Luettu 16.5.2011)

[19] Django Software Foundation “runserver”, [WWW-dokumentti]. Saatavilla: http://docs.djangoproject.com/en/dev/ref/django-admin/. (Luettu 16.5.2011)

[20]] Wikimedia Foundation, ”SQL” [WWW-dokumentti], (28.4.2011). Saatavilla: http://fi.wikipedia.org/wiki/MySQL. (Luettu 16.5.2011)

[21] Oulun seudun ammattikorkeakoulu, ”SQL” [WWW-dokumentti]. Saatavilla: http://www.ratol.fi/opensource/mysql/mysql.htm. (Luettu 16.5.2011)

[22] Oracle, ”MySQL”, [WWW-dokumentti]. Saatavilla: http://dev.mysql.com/doc/refman/5.5/en/introduction.html. (Luettu 16.5.2011)

[23] Wikimedia Foundation, ”Versionhallinta” [WWW-dokumentti], (16.12.2010). Saatavilla: http://fi.wikipedia.org/wiki/Ohjelmiston_versiohallinta. (Luettu 16.5.2011)

[24] Tuomas Kautto, ”Ohjelmistotekniikan seminaariesitelmä” [WWW-dokumentti] (2006). Saatavilla: http://www.mit.jyu.fi/opiskelu/seminaarit/bak/testaus/. (Luettu 16.5.2011)

[25] Wikimedia Foundation, ”Testaus” [WWW-dokumentti], (10.1.2011). Saatavilla: http://fi.wikipedia.org/wiki/Ohjelmiston_testaaminen#Regressiotestaus. (Luettu 16.5.2011)

[26] Sininen meteoriitti, ”Scrum” [WWW-dokumentti]. Saatavilla: http://www.ketteratkaytannot.fi/Menetelmat/Scrum/. (Luettu 16.5.2011)

[27] Reaktor Innovations Oy, ”Scrum” [WWW-dokumentti]. Saatavilla: http://www.reaktor.fi/web/fi/teknologia-ja-tutkimus/scrum. (Luettu 16.5.2011)

[28] Sanna Leskinen , ”Ken on maassa ketterin?”, Tietokone, 5/2011, s. 54 - 58

[29] Scrumwell.com, ”KanBan” [WWW-dokumentti]. Saatavilla: http://scrumwell.com/2009/09/26/mika-ihmeen-kanban/. (Luettu 16.5.2011)

[30] Alex Gaynor, ”Django-taggit” [WWW-dokumentti]. Saatavilla: https://github.com/alex/django-taggit. (Luettu 16.5.2011)

[31] Julian Moritz, ”Django-taggit-templatetags” [WWW-dokumentti]. Saatavilla: https://github.com/feuervogel/django-taggit-templatetags. (Luettu 16.5.2011)

[32] David Hall, ”Django-reversion” [WWW-dokumentti]. Saatavilla: https://github.com/etianen/django-reversion. (Luettu 16.5.2011)

[33] Marco Bonetti, ”Django-rosetta” [WWW-dokumentti]. Saatavilla: http://code.google.com/p/django-rosetta. (Luettu 16.5.2011)

[34] Andrew Godwin, ”South” [WWW-dokumentti] (2010). Saatavilla: http://south.aeracode.org/docs/about.html. (Luettu 16.5.2011)

[35] Rob Hudson, ”Django-debug-toolbar” [WWW-dokumentti]. https://github.com/robhudson/django-debug-toolbar. (Luettu 16.5.2011)

[36]Google, ”Maps” [WWW-dokumentti] (2011). Saatavilla: http://www.google.fi/help/maps/tour/. (Luettu 16.5.2011)

[37] Wikimedia Foundation, ”MIT-lisenssi” [WWW-dokumentti], (19.2.2011). Saatavilla: http://fi.wikipedia.org/wiki/MIT-lisenssi. (Luettu 16.5.2011)

[38] Wikimedia Foundation, ”MIT-lisenssi” [WWW-dokumentti], (10.11.2010). Saatavilla: http://fi.wikipedia.org/wiki/BSD-lisenssi. (Luettu 16.5.2011)