OPINNÄYTETYÖ
KOSTI NEVANLINNA 2012

PLANNER-TAITONSUUNNITTELUTYÖKALU

Rovaniemen
ammattikorkeakoulu
University of Applied Sciences

TIETOTEKNIIKAN KOULUTUSOHJELMA

Rovaniemen
ammattikorkeakoulu
University of Applied Sciences

ROVANIEMEN AMMATTIKORKEAKOULU

TEKNIIKKA JA LIIKENNE

Tietotekniikan koulutusohjelma

Opinnäytetyö

PLANNER-TAITONSUUNNITTELUTYÖKALU

Kosti Nevanlinna

2012

Toimeksiantaja PolarCode Oy

Ohjaaja Aku Kesti

Rovaniemen
ammattikorkeakoulu
University of Applied Sciences

Tekniikka ja liikenne
Tietotekniikan
koulutusohjelma

Opinnäytetyön
tiivistelmä

Tekijä Kosti Nevanlinna Vuosi 2012

Toimeksiantaja PolarCode Oy
Työn nimi Planner-taitonsuunnittelutyökalu
Sivu- ja liitemäärä 38

Opinnäytetyö käsittelee web-pohjaista Planner-taitonsuunnittelutyökalua sekä ohjelmistona että kehitysprosessina. Ohjelmisto rakennettiin PolarCode Oy:n PolarCore-alustaan moduuliksi, joka hyödyntää muita alustan toiminnallisuuksia.

Työn tavoitteena oli toteuttaa PolarCode Oy:n asiakkaalle Polarlehdet Oy:lle työkalu, jolla aikakauslehden taitto voitaisiin suunnitella. Aiemmin suunnittelu tehtiin käsin kynällä ja paperilla. Lehden elementit, kuten jutut ja ilmoitukset, päivittyisivät reaaliajassa järjestelmään ja niiden paikkoja voitaisiin vaihtaa. Lopuksi työkalusta voitaisiin viedä taittajalle dokumentti lehden varsinaista graafista toteutusta varten.

Toteutustapana oli ohjelmistoprosessin vesiputousmalli. Teknisen alustan tietokantana käytettiin MySQL:ää, palvelinpuolen kielenä PHP:tä ja asiakaspuolen kielenä JavaScript:iä ja sen jQuery-kirjastoa. Työkalun ulkoasun looginen runko toteutettiin XHTML-merkkauskielellä ja visuaalinen ilme CSS-tyyliohjeilla.

Työn tuloksena PolarCoden asiakas sai käyttönsä työkalun, jolla voidaan graafisesti suunnitella tulevan lehden taitto, sivumäärät ja ilmoituksien ja artikkeleiden sijoittelu.

Avainsanat taiton suunnittelu, ohjelmistotuotanto, web-kehitys

Rovaniemen
ammattikorkeakoulu
University of Applied Sciences

School of Technology
Degree Programme in
Information
Technology

Abstract of Thesis

Author Kosti Nevanlinna

Year 2012

Client PolarCode Oy
Subject of thesis Planner – Layout Design Tool
Number of pages 38

This thesis is about Planner, a web-based tool for layout designing. The tool was discussed both as a computer application and as a process for software engineering. The application was developed as a module for PolarCore, a software engine developed by PolarCode.

The objective for the thesis was to develop a tool for designing magazine layouts. The tool was to be developed for Polarlehdet Oy, a client of PolarCode. The planning was made before by hand with pen and paper. The elements of the magazine, such as articles and advertisements, would be updated in real-time and they could be freely arranged. When the plan would be finished, the document could be exported to the graphics artist for the actual graphic implementation.

The method for implementation was the waterfall model for software engineering. The database for the system was MySQL. The server-side scripting was done with PHP and the client-side scripting with JavaScript together with its library, jQuery. The structure was implemented with the XHTML markup language and the visual layout used CSS style sheets.

As a result for the thesis, PolarCode's client received a tool for designing magazine's layout, number of pages and the arrangement of articles and advertisements.

Keywords layout design, software engineering, web development

SISÄLTÖ

KUVIOLUETTELO **1**
KOODIESIMERKIT **2**
KÄSITTEET JA TERMIT **3**
1 JOHDANTO **7**
2 KÄYTETYT TEKNOLOGIAT **8**
2.1 UML 8
2.2 PHP 9
2.3 MySQL 10
2.4 JavaScript 11
2.5 XHTML 12
2.6 CSS 13
3 POLARCORE-ALUSTA **15**
3.1 Yleistä 15
3.2 Arkkitehtuuri 16
3.3 Tietorakenteet 17
3.4 Rajapinnat 18
4 JÄRJESTELMÄN KUVAUS **20**
4.1 Määrittely 20
4.2 Käyttötapaukset 21
4.3 Käyttöliittymäsuunnitelma 25
4.4 Toteutunut järjestelmä 26
5 KEHITYSPROSESSI **31**
5.1 Prosessi käytännössä 31
5.3 Ongelmat 32
5.4 Prosessin kehitys 33
6 YHTEENVETO **36**
LÄHTEET **37**

KUVIOLUETTELO

Kuvio 1. UML:n erilaisten kaavioiden välinen hierarkia 8
Kuvio 2. PolarCore-arkkitehtuurin rakenne 17
Kuvio 3. Plannerin käyttötapaukset UC001-UC006 20
Kuvio 4. Plannerin käyttötapaukset UC007-UC012 21
Kuvio 5. Plannerin käyttöliittymän suunnitelma 25
Kuvio 6. Plannerin päänäkymä 26
Kuvio 7. Lehden asetukset 27
Kuvio 8. Ilmoituksen asettaminen ilmoituspaikkaan 28
Kuvio 9. Kontekstivalikko 28
Kuvio 10. Sivujen eri tilat 29
Kuvio 11. Plannerin muiden toiminteiden linkit 29

KOODIESIMERKIT

Koodiesimerkki 1. Hello World -ohjelma voidaan suorittaa joko suoraan tai HTML-koodiin upotettuna 10
Koodiesimerkki 2. JavaScriptillä ja jQuery poikkeavat toisistaan dokumentin oliomallin käsittelyn suhteen 12
Koodiesimerkki 3. Otsikon 1-tason muotoilu CSS-säännöllä 14

KÄSITTEET JA TERMIT

Ajax

Ajax on joukko asiakaspuolen web-kehitystekniikoita, joilla voidaan toteuttaa reaaliaikaisia verkkosovelluksia.

Alikysely

Alikysely on SQL-kyselystä muodostettu sisäkkäinen kysely.

Asiakaspuolen kieli

Asiakaspuolen kieli on käyttäjän omassa tietokoneessa suoritettava ohjelmisto tai sen osa, jolla ei ole suoraa pääsyä palvelimelle.

Dokumentin oliomalli

Dokumentin oliomalli (DOM, Document Object Model) on alusta- ja kieliriippumaton käytäntö HTML-, XHTML- ja XML-dokumenttien esittämiseen.

Heikosti tyypitetty kieli

Heikosti tyypitetty kieli on ohjelmointikieli, jonka muuttujien tyyppi määräytyy niihin sijoitettavan datan mukaan.

HTTP

HTTP (HyperText Transfer Protocol) on pyyntö-vastaus-periaatteeseen pohjautuva tiedonsiirtoprotokolla.

Kirjasto

Kirjasto on ohjelmistokehitykseen käytettävä resurssikokoelma, joka voi sisältää esimerkiksi esikirjoitettua koodia ja valmiita luokkia.

Luokka

Luokka on ohjelmistorakenne, jonka tarkoituksena on toimia mallina olioiden luomiseen.

Moduuli

Moduulilla tarkoitetaan sellaista järjestelmän osaa, joka voidaan lisätä tai poistaa ilman, että järjestelmän muu toiminta häiriintyy.

Määrittely

Ohjelmistoprosessin vaihe, jossa päätetään järjestelmän ominaisuudet yhdessä asiakkaan kanssa.

Nimiavaruus

Nimiavaruus on abstrakti säiliö tai ympäristö, joka sisältää uniikkeja tunnisteita tai symboleja eli nimiä.

Näkymä

Näkymä on tallennettu SQL-kysely virtuaalisessa taulussa.

ODBC

ODBC (Open Database Connectivity) on standardirajapinta tietokantajärjestelmiin ja sitä voidaan käyttää riippumatta tietokannan tyypistä tai käyttöjärjestelmästä.

Olio

Olio tai objekti on luokan esiintymä, joka sisältää joukon loogisesti yhteenkuuluvaa tietoa ja toiminnallisuutta.

Palvelinpuolen kieli

Palvelinpuolen kieli on ohjelmointikieli, joka ajetaan palvelimella irrallaan järjestelmän käyttöliittymästä.

Planner

Planner on PolarCode Oy:n asiakkaalle Polarlehdet Oy:lle toteutetun Internet-selainpohjaisen aikakauslehden suunnittelutyökalun nimi.

PolarCore

PolarCore on PolarCode Oy:n kehittämä järjestelmäalusta, jonka räätälöityä versiota Planner käyttää.

Rajapinta

Rajapinnalla tarkoitetaan komponenttien välistä interaktiopistettä, joka mahdollistaa komponenttien välisen kommunikaation määrätyllä protokollalla.

Relaatiotietokanta

Relaatiotietokanta on tietokanta, jonka sisältö on jaettuina toisiinsa viittaaviin tauluihin.

RSS

RSS (Really Simple Syndication) on verkkomuotoisen ja usein päivitettävän tiedon esittämiseen käytettävä formaatti.

Serialisointi

Serialisoinnilla tarkoitetaan tietorakenteen tai objektin muuntamista tallennettavaan muotoon.

Solmu

Solmu on tallenne joka koostuu tietokentästä ja yhdestä tai useammasta viitteestä toisiin solmuihin.

SQL

SQL (Structured Query Language) on relaatiotietokantojen sisällön hallinnoimiseen tarkoitettu ohjelmointikieli.

SyncML

SyncML on tiedon synkronoimiseen mobiililaitteen ja tietokoneen välillä käytettävä standardi.

Taitto

Taitolla tarkoitetaan lehden graafista toteutusta, jossa lehden sisältö sijoitetaan lehden sivuille visuaalisesti toimivaksi kokonaisuudeksi.

Triggeri

Triggeri on automaattisesti suoritettava proseduraalinen koodi, joka suoritetaan määrättyjen tapahtumien yhteydessä.

Transaktio

Transaktio on yhdestä tai useammasta käskystä koostuva tietokantatapahtuma, jonka jokaisen osa-alueen tulee suorittua loppuun asti läpäistäkseen tarkistuksen.

Unicode

Unicode on tekstin johdonmukaiseen esittämiseen ja muotoiluun käytettävä standardi.

XML

XML (eXtensible Markup Language) on merkkauskieli, jota käytetään tiedon tallettamiseen määrätyssä muodossa.

1 JOHDANTO

Toiminnallinen opinnäytetyöni käsittelee PolarCode Oy:n asiakkaalle Polarlehdet Oy:lle toteutettua Internet-selainpohjaista aikakauslehden suunnittelutyökalua, Planneria. Työkalu rakennettiin osaksi laajempaa toiminnanohjausjärjestelmää, matkalehti.fi:tä, joka sisälsi Plannerin lisäksi muun muassa julkisen www-sivuston, verkkokaupan ja asiakkuuksien hallinnan. Planner perustui PolarCode Oy:n omaan räätälöityyn järjestelmäalustaan, PolarCoreen. Tästä johtuen työssä käytetyt teknologiat olivat jo ennalta määriteltyjä järjestelmäalustan ominaisuuksien mukaisesti.

Työkalulla pyrittiin vastaamaan Polarlehdet Oy:n tarpeeseen saada helppokäyttöinen työkalu korvaamaan lehden vanhentunut kynään ja paperiin perustunut tapa suunnitella sen ulkoasu. Kyseisen tavan suurin heikkous oli sen päivitettävyys, sillä lehden rakenne ja sisältö elää lähes aina ilmestymispäivään asti. Käytännössä tämä tarkoitti peruuntuneiden ilmoitusten yliviivaamista, siirtyneiden sisältöjen uudelleensijoittelua, nuolien piirtämistä uusille sivuille ynnä muuta lisätyötä. Pahimmassa tapauksessa sivusuunnitelma tuli korjausmerkintöineen liian täyteen ollakseen enää ymmärrettävä ja prosessi jouduttiin aloittamaan kokonaan alusta.

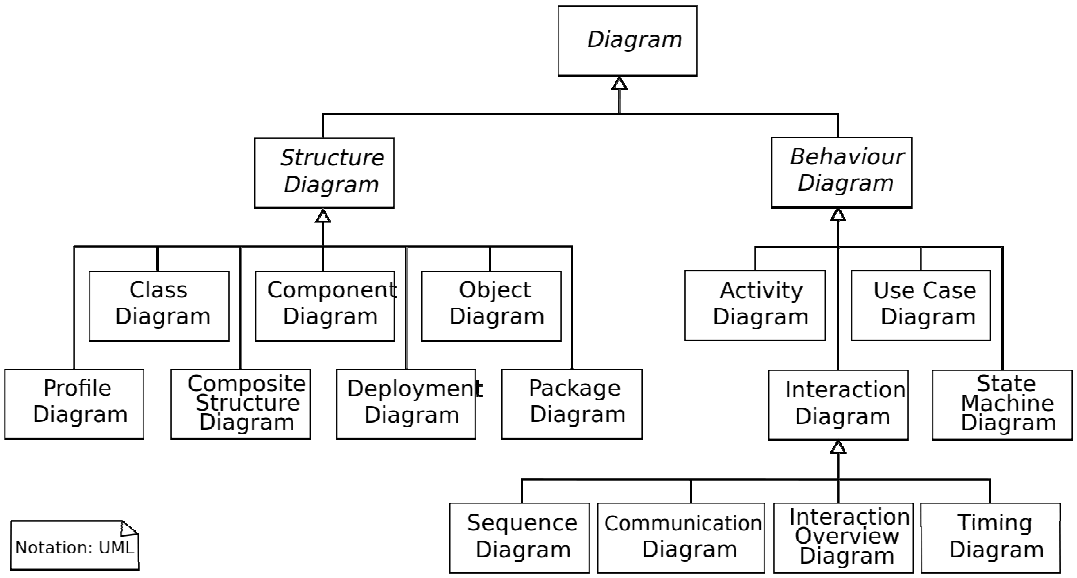
Plannerilla voitaisiin suunnitella lehden rakenne sivupohjineen ja sisältöineen siten, että rakenne ei hajoaisi, vaikka suunnitelmaan tulisi suuriakin muutoksia. Käytettävissä olevat jutut ja ilmoitukset päivittyisivät reaaliajassa järjestelmään ja niiden paikkoja olisi vapaasti siirreltävissä sivujen välillä. Lopuksi suunnitelmasta voitaisiin viedä ulos dokumentti, josta taittaja näkisi kootusti kaiken, mitä millekin lehden sivulle tulee.

Raportin ensimmäisessä luvussa asetetaan projektin lähtökohdat ja tavoite. Toisessa ja kolmannessa luvussa perehdytään lyhyesti käytettyihin teknologioihin ja PolarCore-alustaan. Neljäs luku käsittelee järjestelmän toimintaa ja sen käyttöliittymää. Viidennessä luvussa kerrotaan työprosessista itsessään ja sen haasteista: lisäksi pyritään esittämään vaihtoehtoisia toimintamalleja prosessin tehostamiseksi.

2 KÄYTETYT TEKNOLOGIAT

2.1 UML

UML (Unified Modeling Language) on mallinnuskieli, jolla voidaan viestiä ohjelmistotuotantoprosessin menetelmiä. Se määrittelee sekä notaation eli malleissa käytettävän graafisen esityksen että metamallin, joka puolestaan määrittelee notaation. Suurimmassa osassa menetelmiä ei käyttäjän ole kuitenkaan välttämätöntä ymmärtää metamalleja UML-notaation hyödyntämiseksi. (Fowler–Scott 2002, 2–4.)



Kuvio 1. UML:n erilaisten kaavioiden välinen hierarkia (Wikimedia Commons 2012)

UML:n tärkein funktio on viestintä: sen avulla voidaan selkeästi esitellä määrätyt termit verrattuna luonnollisten kielien epätäsmällisyyteen teknisessä suunnittelussa. Koska ohjelmistoprojekteissa toimitaan asiakkaiden kanssa, voidaan UML:n käytöllä löytää yhteinen kieli eri ammattislangien käytön sijaan. (Fowler–Scott 2002, 7–10.)

UML-kaaviot jakautuvat kolmeen eri kategoriaan: yleisiä käyttäytymismalleja kuvaavat käyttäytymiskaaviot, erilaisia vuorovaikutuksia kuvaavat vuorovaikutuskaaviot ja staattisia ohjelmarakenteita mallintavat rakennekaaviot (Object Management Group 2012).

Käyttäytymiskaavioihin kuuluvat erilaisia skenaariojoukkoja kuvaavat käyttötapauskaavio, olion mahdollisia tiloja kuvaavat tilakaaviot ja tehtävien tapahtumajärjestystä kuvaavat aktiviteettikaaviot (IBM 2012).

Vuorovaikutuskaaviot sisältävät olioiden välisiä toimintoja kuvaavat sekvenssikaaviot, viestejä ja niiden osia kuvaavat kommunikaatiokaaviot, olion tilan tai elämänviivan muutosta ajan kuluessa kuvaavat ajoituskaaviot sekä muiden vuorovaikutuskaavioiden etenemistä kuvaavat kokoavat vuorovaikutuskaaviot (IBM 2012).

Rakennekaavioihin kuuluvat fyysisia koodimoduuleja kuvaavat komponenttikaaviot, järjestelmän osien viestiyhteyksiä kuvaavat koostekaaviot, olioiden välisiä suhteita kuvaavat luokkakaaviot, luokkien ilmentymiä kuvaavat oliokaaviot, pakettien välisiä riippuvuuksia kuvaavat pakkauskaaviot ja fyysisiä solmuja kuvaavat sijoittelukaaviot (IBM 2012).

Plannerissa UML:ää käytettiin käyttötapauskaavioiden osalta mahdollistamaan toteuttajan ja asiakkaan välistä viestintää eri toiminteiden määrittelyssä.

2.2 PHP

PHP (PHP Hypertext Preprocessor) on palvelinpuolen skriptikieli. Se on tarkoitettu erityisesti web-pohjaiseen sovelluskehitykseen. Laajassa merkityksessään PHP:lla viitataan paitsi PHP-kieleen itsessään myös PHP-ohjelmien suorittamiseen mahdollistaviin ratkaisuihin: voidaan puhua PHP-ympäristöstä. (Rantala 2002, 12.) PHP on avoimen lähdekoodin ohjelmisto ja se on saatavilla ilmaiseksi.

PHP voidaan asentaa kaikkiin yleisimpiin käyttöjärjestelmiin ja se tukee useimpia suosituimpia palvelinratkaisuja, kuten Apachea ja IIS:ää. PHP:lla voidaan tulostaa HTML:n lisäksi myös PDF- ja Flash-dokumentteja. Tietokantatuki on saatavilla tietokanta- ja ODBC-laajennuksien sekä dataobjektien avulla. Lisäksi PHP:hen on saatavilla useita muita laajennuksia erilaisiin käyttötarpeisiin. (The PHP Group 2012.)

Kielen syntaksi pohjautuu C-kieleen ja siinä on ominaisuuksia Java-, Perl- ja C++-kielistä. PHP on heikosti tyypitetty kieli eikä sen muuttujia voida alustaa

joitakin poikkeuksia lukuunottamatta. PHP:ssa on myös käytössä valmiiksi määriteltyjä ympäristömuuttujia, joista saadaan listaus metodilla *phpinfo()*. (Rantala 2002, 52, 60.) Erillisten PHP-tiedostojen lisäksi koodia voidaan kirjoittaa myös suoraan HTML-tiedostoihin upotettuina merkinnällä *<?php ?>*.

```
// Suoraan tulostus
<?php echo “Hello, World!”; ?>

// HTML-koodiin upotettuna
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
   <head>
      <title>Otsikko</title>
   </head>
   <body>
      <p><?php echo “Hello, World!”; ?></p>
   </body>
</html>
```

Koodiesimerkki 1. Hello World -ohjelma voidaan suorittaa joko suoraan tai HTML-koodiin upotettuna

Usein PHP:tä käytetään yhdessä muun avoimen lähdekoodin arkkitehtuurin kanssa. Linux-käyttöjärjestelmän, Apache-palvelimen, MySQL-relaatiotietokannan ja PHP:n muodostamaa ympäristöä kutsutaan yleisesti nimellä LAMP. Tässä nimenomaisessa muodossa PHP on myös käytössä PolarCore-alustassa ja näin ollen myös Plannerissa.

2.3 MySQL

MySQL (My Structured Query Language) on eniten käytetty avoimen koodin relaatiotietokantajärjestelmä. Se voidaan asentaa kaikille yleisimmille käyttöjärjestelmille ja siitä on saatavissa sekä ilmaiseen että kaupalliseen käyttöön tarkoitettuja lisenssejä. MySQL:n omistaa tällä hetkellä Oracle. (Oracle Corporation 2012.)

MySQL:ää käytetään monissa laajoissa Internet-sovelluksissa, kuten Wikipediassa, Googlessa, Facebookissa ja Twitterissä. (Wikimedia Foundation 2012; Oracle Corporation 2012; Facebook 2012; Twitter 2012) Useissa ohjelmointikielissä on olemassa kirjastoja MySQL:n käyttöä varten ja niissä,

joissa sitä ei ole, voidaan yhteys muodostaa käyttäen MySQL:n omaa ODBC-rajapintaa, MyODBC:tä. MySQL:n mukana ei tule varsinaista käyttöliittymää, mutta sitä voidaan käyttää joko komentokehotteella tai jonkin kolmannen osapuolen käyttöliittymän avulla.

Järjestelmän vahvuuksia ovat muun muassa nopeus, jäljennettävyys ja useat tietovarastoalustat, kuten InnoDB ja MyISAM. Lisäksi MySQL tukee yleisimpiä ominaisuuksia, kuten alikyselyitä, näkymiä, triggereitä ja transaktioita. Rajoitteena MySQL:llä on heikohko SQL-standardin noudatus ja XML-tuki. (Kofler 2005, 5–8.)

Planner hyödyntää MySQL:ää PolarCore-alustan käyttämien tietovarastojen lisäksi myös omissa, sivujen sisäistä logiikkaa ja hierarkiaa sisältävissä tauluissa.

2.4 JavaScript

JavaScript on asiakaspuolen ohjelmointikieli, jolla voidaan lisätä toiminnallisuuksia verkko-ohjelmistoihin ja -sivuihin. Se on tarkoitettu interaktiivisuuden lisäämiseen ja käytettävyyden parantamiseen. Nimestään huolimatta JavaScriptillä ja Javalla ei ole kuitenkaan keskenään tekemistä, vaan ne ovat kaksi eri ohjelmointikieltä. (Smith–Negrino 2007, 2–6.)

Usein JavaScriptiä käytetään verkkosivun dokumentin oliomallin käsittelyyn (Smith–Negrino 2007, 13). Tällaisia tilanteita ovat esimerkiksi käyttäjän antaman syötteen validointi, uusien ikkunoiden avaaminen ja kuvan vaihto hiiren kursorin siirtyessä sen päälle (ns. rollover-efekti). Toinen yleinen käyttökohde on Ajax, jonka avulla voidaan siirtää valtaosa ohjelman toimintalogiikasta käyttäjän selaimelle ja siten minimoida palvelimelta haettavan ja sinne siirrettävän tiedon määrä. (Smith–Negrino 2007, 8–9.) JavaScript voidaan myös asentaa palvelimelle käsittelemään HTTP-pyyntöjä ja -vastauksia.

JavaScript pohjautuu C-kieleen ja se on heikosti tyypitetty kieli. Kuten PHP:tä, myös JavaScriptiä voidaan sekä upottaa suoraan HTML-koodiin että linkittää koodi erilliseksi tiedostoksi. JavaScript sisältää tiettyjä esimääriteltyjä objekteja, kuten taulukoiden käsittelyyn tarkoitettu Array ja tiedon serialisointiin tarkoitettu JSON. (Mozilla Developer Network 2012.) JavaScriptiin on myös saatavissa kirjastoja, jotka lisäävät siihen ominaisuuksia. Esimerkkinä

tällaisesta kirjastosta on jQuery, joka on tarkoitettu yksinkertaistamaan hankalahkoa dokumentin oliomallin käsittelyä. (The jQuery Foundation 2012.)

```
// JavaScript
window.onload = function() {
   alert(“Hello, World!”);
}

// jQuery
$(document).ready(function() {
   alert(“Hello, World!”);
}
```

Koodiesimerkki 2. JavaScriptillä ja jQuery poikkeavat toisistaan dokumentin oliomallin käsittelyn suhteen

Planner käyttää JavaScriptiä runsaasti käyttöliittymälogiikan toteutuksessa. Myös suuri osa sen graafisista efekteistä on toteutettu jQuery-kirjastoa ja sen efektilaajennuksia hyväksi käyttäen.

2.5 XHTML

XHTML (eXtensible HyperText Markup Language) on HTML-merkkauskielen versio, joka on standardisoitu käyttämään XML-merkkauskielen muotomääreitä. Sen avulla voidaan paitsi rakentaa verkkosivun runko myös yhteiskäyttää XML-pohjaisia kieliä.

Vaikka XHTML ei varsinaisesti tuo HTML:ään uusia ominaisuuksia, on se useimpien asiantuntijoiden mielestä standardoinnin vuoksi paras ratkaisu verkkosivujen toteuttamiseen. (Korpela–Linjama 2004, 26–29). Suurimmat erot HTML:n ja XHTML:n välillä ovat muotosäännöissä merkkauksen kirjoituksessa. XHTML:ää kehittää World Wide Web Consortium ja se on aktiivinen toimija standardien määrittelyissä ja tarjoaa kehittäjille suosituksia toteutustavoista laadukkaan lopputuloksen aikaansaamiseksi. (W3C 2012.)

XHTML:ää voidaan tuottaa sekä erilaisilla editoreilla että suoraan kirjoitettuna. Koska editorien välillä voi olla suuriakin eroja, miten merkkaus luodaan ja minkälaisia CSS-tyylejä sisällytetään, on web-kehittäjien yleisimmin käyttämä tapa käyttää jotakin edellisten yhteensulautumaa: osa merkkauksesta kirjoitetaan käsin ja osa luodaan automaattisesti. (Korpela–Linjama 2004, 32–43.)

XHTML ei itsessään ota kantaa sisällön visuaaliseen esittämistapaan, vaan se riippuu CSS-määrittelyistä ja käytettävästä selaimesta.

XHTML-dokumentin juurielementti on html ja se sisältää xmlns-attribuutin liittyäkseen XHTML-nimiavaruuteen. Lisäksi dokumentti sisältää <!DOCTYPE>-ilmoituksen, joka määrittelee, mitä dokumenttityyppimääritystä noudatetaan. Validoituakseen dokumentin tulee lisäksi noudattaa eräitä muita muotosääntöjä. (Korpela–Linjama 2004, 109–114.)

XHTML on käytössä kaikissa niissä Plannerin kohdissa, joissa tuotetaan käyttäjälle visuaalista sisältöä. Lisäksi XHTML on tiedon esittäjänä niissä kohdin käyttöliittymää, joissa Plannerilla ja PolarCorella on yhteisiä elementtejä.

2.6 CSS

CSS (Cascading Style Sheets) on merkintäjärjestelmä dokumenttien ulkoasujen muotoiluja varten. Yleensä CSS:stä puhuttaessa tarkoitetaan verkkosivujen tyyliohjeita. (Korpela 2003, 2.)

CSS:ää voidaan käyttää joko yksittäisten elementtien muotoiluun tai sivun ulkoasun toteuttamiseen kokonaisuudessaan. Osaa sen ominaisuuksista voidaan käyttää myös suoraan HTML:llä sellaisenaan, kuten fonttien värejä. Selkein ero HTML:n muotoiluominaisuuksiin monipuolisuuden ohella on rakenteen ja muotoilun erottaminen: voidaan toteuttaa verkkosivun looginen esitysasu ottamatta kantaa sen visuaaliseen ilmeseen. (Korpela 2003, 20–22.) Lisäksi etuna on parempi mahdollisuus saada ulkoasu näyttämään samalta selaimesta riippumatta.

Tyyliohjeen yksittäisen säännön perusmuoto koostuu selektorista ja deklaraatiosta ja niitä voidaan käyttää joko erillisinä tiedostoina tai suoraan HTML:ään upotettuina. Sääntö voi sisältää useita deklaraatioita puolipisteellä erotettuina ja usein tämä onkin selkein tapa säännön esitykseen. (Korpela 2003, 51–52.) Selektoreita voidaan myös yhdistellä erottamalla ne pilkuilla toisistaan. Eri selaimet tulkitsevat sääntöjä eri tavoin ja tämän vuoksi on pyritty antamaan suosituksia niiden käytöstä. (W3C 2011.)

```
// Perusmuoto: selektori {deklaraatio;}
h1 {
   font-family: Arial, sans-serif;
   font-size: 1.5em;
   color: blue;
}
```

Koodiesimerkki 3. Otsikon 1-tason muotoilu CSS-säännöllä

Plannerin ulkoasu on toteutettu CSS:llä sekä suoraan HTML:ään upotettuna että erillisinä tiedostoina. Osa ulkoasusta rakentuu myös dynaamisesti käyttäjän toiminteiden mukaan muuttamalla dokumentin oliomallia sekä JavaScriptillä suoraan että sen jQuery-kirjaston avulla.

3 POLARCORE-ALUSTA

3.1 Yleistä

PolarCore on PolarCode Oy:n järjestelmämoottori. Sitä käytetään pohjana kaikissa PolarCoden tuotteissa. Näitä tuotteita ovat www-sivujen hallintajärjestelmä Online Marketing, verkkokaupan- ja tilaustenhallinta Online Sales ja asiakkuuksienhallinta Online CRM. Lisäksi saatavilla on lehtialan toiminnanohjausjärjestelmä Online Paper.

Koska PolarCore toimii ainoastaan teknisenä alustana eikä ota kantaa sisällön esittämistapaan, voidaan tuotteita vapaasti yhdistellä toisiinsa. Tyypillistä onkin toteuttaa esimerkiksi julkiset www-sivut ja verkkokauppa yhdistämällä Online Marketing ja Online Sales tai hallinnoida yrityksen tilauksia ja asiakkuuksia käyttämällä Online Salesin ja Online CRM:n yhdistelmää. Kaikkien edellisten kombinaatiota kutsutaan nimellä Online MCS.

Kaikki PolarCoreen pohjautuvat tuotteet ovat www-pohjaisia ja niitä voidaan käyttää tavallisella Internet-selaimella.

Järjestelmässä on oletuksena kolme käyttäjäryhmää. Näitä ovat pääkäyttäjä, rekisteröitynyt käyttäjä ja anonyymi käyttäjä, missä anonyymillä tarkoitetaan ei-sisäänkirjautunutta käyttäjää. Käyttäjäryhmiä voidaan luoda lisää ja niiden oikeudet ovat vapaasti määriteltävissä moduulitasolle asti hallintatyökalujen avulla. Voidaan esimerkiksi määrittää anonyymeille käyttäjille pääsy ainoastaan julkisille www-sivuille ja rekisteröityneille käyttäjille lisäoikeus muokata kyseistä sisältöä. Käyttäjien määrää ei ole rajoitettu.

Julkisia ja intranet-tyyppisiä www-sivustoja voidaan hallinnoida järjestelmän työkalujen avulla. Sivuston visuaalinen ulkoasu on vapaasti määriteltävissä ja sivupohjia voidaan luoda yksi tai useampia eri käyttötarpeita varten. Sivujen näkyvyys voidaan myös määritellä sivutasolla edellä mainittujen käyttäjäryhmien mukaan.

PolarCoren varsinainen ohjelmarunko on toteutettu PHP-kielellä ja se toimii yhteydessä MySQL-tietokannan kanssa. Käyttöliittymä on toteutettu XHTML:llä ja CSS:llä JavaScriptiä ja sen laajennusta, jQueryä, hyödyntäen.

Seuraavaksi käydään läpi PolarCore-alustan tekninen toimintaperiaate, tietokannan peruslogiikka ja rajapinnat muihin järjestelmiin.

3.2 Arkkitehtuuri

PolarCore-alusta jakautuu karkeasti kahteen osaan: käyttäjälle näkyvät, varsinaisia toiminnallisuuksia toteuttavat moduulit ja näille metodeja tarjoava, matalamman tason ydin eli Core.

Moduulit toimivat pääsääntöisesti toisistaan irrallaan eivätkä siten vaadi muita moduuleja toimiakseen. Poikkeuksena tähän ovat toisista moduuleista periytetyt niin sanotut alimoduulit, kuten tiedostonhallinta-moduulista periytetty, tiedostoja itsessään käsittelevä tiedostomoduuli. Core sen sijaan on itsenäinen kokonaisuus, jonka yksittäisten elementtien muokkaus vaikuttaa laajasti koko järjestelmän toimintaan.

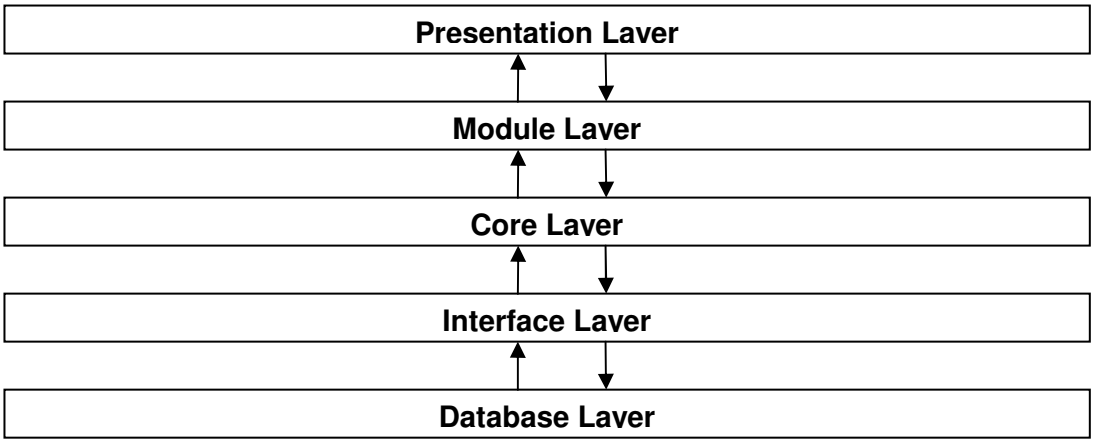
Moduulit voivat myös keskustella keskenään Coren kautta. Tällaista funktiota voidaan hyödyntää esimerkiksi moduulissa, joka tarvitsee tiedostojen käsittelytoimintoja. Tällöin tiedosto-moduulista voidaan tarvittaessa luoda esiintymä eikä kyseisiä toimintoja tarvitse toteuttaa uudelleen moduulia varten.

Tyypillinen tiedonkulkutilanne tapahtuu järjestelmässä seuraavasti:

1. Käyttäjä antaa järjestelmälle herätteen (esimerkiksi pyytää määrätyn asiakkaan kaikkien tilausten listan).
2. Käyttöliittymä lähettää hakupyynnön moduulille.
3. Moduuli kokoaa pyydetyistä tiedoista SQL-hakulausekkeen ja lähettää sen edelleen Corelle käsiteltäväksi.
4. Core lisää pyyntöön tarvittavat lisälausekkeet ja pyytää tiedot rajapintatoiminteiden avulla fyysisestä tietokannasta.
5. Rajapinta palauttaa vastauksen Corelle, joka käsittelee vastauksen ja palauttaa tuloksen moduulille (tai jos kyseessä on vikatilanne, virheilmoituksen).
6. Moduuli käsittelee vastauksen oman toimintalogiikkansa mukaan ja muuntaa vastauksen sopivaan muotoon esitystasolle.
7. Käyttäjä saa haun tulokset visuaalisena esityksenä ruudulleen (esimerkkitapauksessa kyseisen tilausluettelon).

Coren ja moduulien lisäksi järjestelmässä on käytettävissä erilaisia työkaluja yksinkertaisia mutta usein toistettavia tapauksia varten. Näitä ovat esimerkiksi taulukoiden käsittelyyn tarkoitettu ArrayUtils, solmujen välisiä hierarkioita selvittävä NodeUtils ja XML-tiedon muotoiluun tarkoitettu XMLUtils. Lisäksi

voidaan käyttää muita pienempiä apuvälineitä, kuten pikahakuja, järjestettäviä taulukoita ja graafien esityksiä.



Kuvio 2. PolarCore-arkkitehtuurin rakenne

Useimmissa käyttötapauksissa järjestelmän kaikkien tasojen ominaisuuksia tarvitaan toiminteiden toteuttamiseen. Kuviossa 2 on havainnollistettu eri abstraktiotasojen välinen hierarkia, jossa tietokantataso on alimpana. Tieto kulkee rajapintatason kautta Corelle, joka välittää tiedon käsittelyä varten moduulitasolle. Lopuksi esitystaso muokkaa tiedon käyttäjälle esitettävään muotoon.

Järjestelmätasolla Planner on alimoduuli tilaustenhallinnan päämoduulista. Kyseinen päämoduuli itsessään ei sisällä toteutusta, vaan se toimii ainoastaan mallina siitä periytettäviin moduuleihin. Planner käyttää myös muiden, sitä tukevien moduulien tietoja: näitä ovat esimerkiksi tilaustenhallinta- ja asiakaskontaktimoduuli.

3.3 Tietorakenteet

PolarCoren käyttää MySQL-relaatiotietokantaa tiedon tallettamiseen. Jokaista yksittäistä tuoteasennusta kohden luodaan oma tietokanta, joka sisältää kaiken järjestelmän tarvitsevan tiedon. Käytännössä tämä tarkoittaa sitä, että mitä useampia moduuleja kyseisessä asennuksessa on käytössä, sitä laajempi tietokanta asennukseen toteutetaan. Laajimmassa muodossaan PolarCore sisältää lähes 200 eri taulua.

Tietokannan pienin yksikkö on solmu eli noodi. Noodiin itseensä ei tallenneta tietoa: se sisältää uniikin indeksinumeron lisäksi ainoastaan metatietoa. Tällaista tietoa on esimerkiksi luontiajankohta, tila tai noodin sijainti muihin noodeihin nähden. Lisäksi jokaiselle noodille määritellään sen tyyppi, joka ilmaisee noodin viittaaman tiedon muodon (kuten sisältösivu, asiakas tai tilaus).

Noodeja voidaan myös ryhmitellä. Tällöin ryhmälle määritetään isäntänoodi, ryhmän jäsenet ja liitoksen tyyppi. Esimerkiksi tiettyyn tilaukseen kiinnitetyt yksittäiset tuotteet muodostavat noodiryhmän liitostyypillä ”tuote tilaukseen”. Ryhmät voivat myös muodostaa osajoukkoja muista ryhmistä.

Tapauksissa, joissa noodiryhmät eivät riitä määrätyn toiminnallisuuden toteuttamiseen, voidaa käyttää lisäksi niin sanottuja virtuaalisia noodeja. Tällöin noodilla on kaksi hierarkiaa: toinen määrittää noodin sijainnin varsinaisessa puurakenteessa ja toinen kyseiseen noodiin liittyvät syvyysakselille mallintuvat, kolmannen ulottuvuuden noodit. Esimerkkinä tällaisesta rakenteesta voisi olla sisältösivukokonaisuus, joka liittyy toisiin sisältösivuihin.

Tietosisällöt itsessään tallettuvat moduulien määräämiin tauluihin. Tieto tallennetaan unicode-muodossa, jotta se on helposti muunnettavissa. Tiedon talletukseen pyritäänkin käyttämään mahdollisimman alkuperäistä ja yksinkertaista muotoa ja toteuttaa tiedon muuntaminen moduulissa kunkin tilanteen vaatimalla tavalla. Yksittäinen moduuli sisältää tyypillisesti yhdestä viiteen taulua.

Plannerin eniten käyttämä taulu mallintaa lehden rakennetta. Se sisältää tietoa määrätyn lehden numeron sivuista, sivujen sisältämistä aineistoista ja aineistojen välisistä yhteyksistä. Lisäksi Planner käyttää muita tauluja toimintalogiikkansa tukemiseen, kuten ilmoituksien tilaajien tietoja ja juttujen sisältöjä.

3.4 Rajapinnat

Edellä mainittujen ominaisuuksien lisäksi PolarCore sisältää rajapintoja muihin järjestelmiin. Rajapinnat jakaantuvat kolmeen ryhmään: tiedostojen vientiin käytettävät rajapinnat, sähköpostin käsittelyyn käytettävät rajapinnat ja tiedon muuntamiseen toiseen muotoon käytettävät rajapinnat. Lisäksi järjes-

telmässä on mahdollisuus yhteystietojen ja kalenterin synkronoimiseen SyncML-rajapinnan avulla.

Järjestelmä esittää useita tietoja listamuotoisena ja nämä tiedot voidaan viedä Excel-muotoon. Tekstimuotoiset sisällöt voidaan viedä Word-muotoisiksi: esimerkiksi laskun esitys muotoillaan jatkokäsittelyä ja tulostusta varten. Kyseiset tiedot ovat myös tulostettavissa lukittuun PDF-muotoon, jolloin tiedostoa ei voida enää muokata ilman erityisiä ohjelmistoja.

Sähköpostin lähetys tapahtuu muusta järjestelmästä irrallisen rajapinnan kautta väärinkäytösten estämiseksi. Viesti voidaan toimittaa vapaasti määriteltävältä lähettäjältä yhdelle tai useammalle vastaanottajalle. Viestejä voidaan myös tallettaa viestipohjiksi. Liitteet toimitetaan linkkinä järjestelmän julkisiksi määriteltyihin tiedostoihin viestin koon pienenä pitämiseksi.

Tieto voidaan viedä yksinkertaistetun tulostusmuodon lisäksi RSS-syötteeksi tai XHTML-dokumentiksi. RSS-muotoa käytetään esimerkiksi uutisten esittämiseen.

Planner käyttää edellä mainituista rajapinnoista tiedostojen vientiin käytettäviä rajapintoja. Lehden suunnitelman ollessa Plannerissa valmis viedään suunnitelmasta Word-dokumentti, joka kokoaa yhteen lehdessä käytettävän aineiston ja niiden sijoittelusuunnitelman. Kyseinen dokumentti toimitetaan edelleen taittajalle varsinaista ulkoasun toteutusta varten.

4 JÄRJESTELMÄN KUVAUS

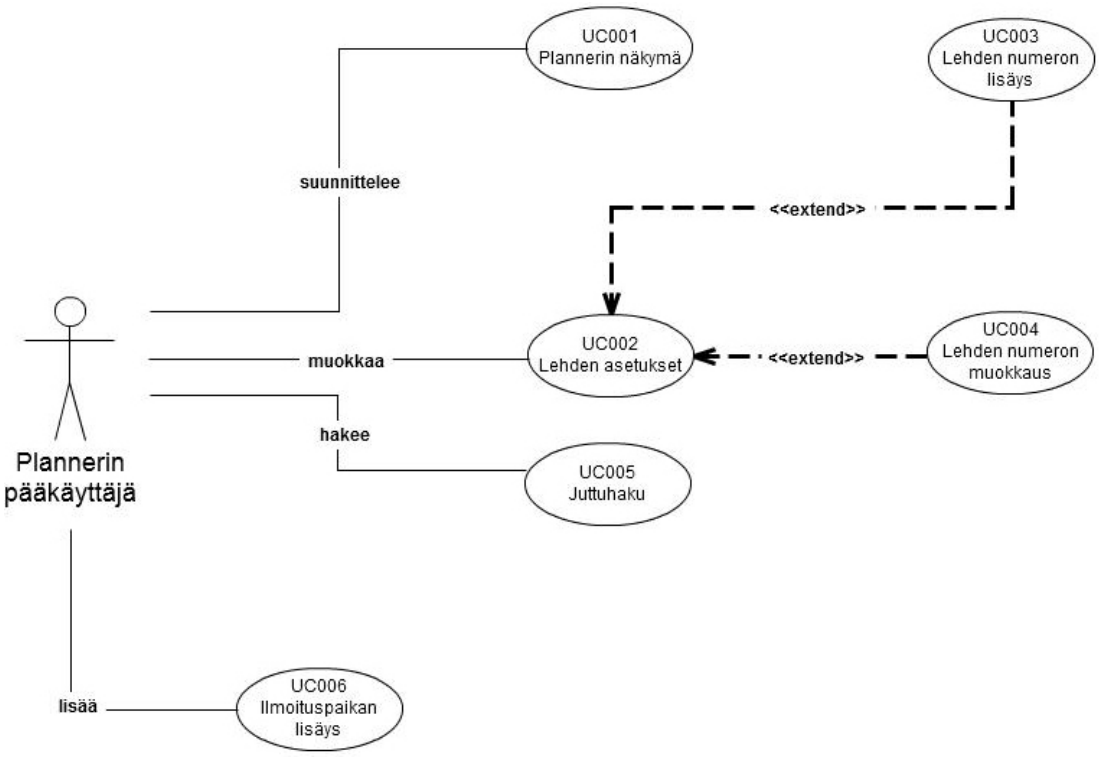
4.1 Määrittely

Planner määriteltiin toimituksen juttu- ja sivutuotannon suunnittelu- ja hallintatyökaluksi. Plannerin avulla suunniteltaisiin lehden rakenne ja kiinnitettäisiin jutut ja ilmoitukset lehden rakenteeseen.

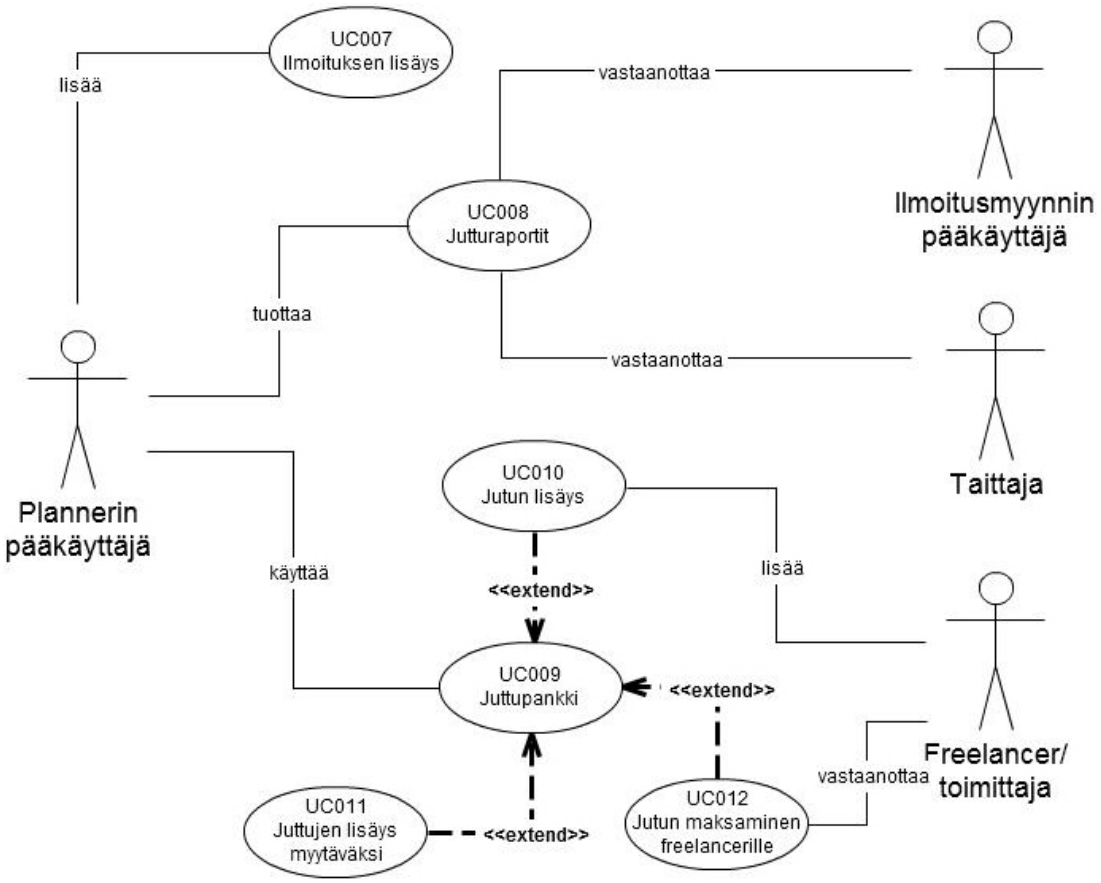
Prosessiin sisällytettiin seuraavat käyttötapaukset:

UC001 Plannerin näkymä
UC002 Lehden asetukset
UC003 Lehden numeron lisäys
UC004 Lehden numeron muokkaus
UC005 Juttuhaku
UC006 Ilmoituspaikan lisäys
UC007 Ilmoituksen lisäys
UC008 Jutturaportit
UC009 Juttupankki
UC0010 Jutun lisäys
UC0011 Juttujen lisäys myytäväksi
UC0012 Jutun maksaminen freelancerille

Planneriin määriteltiin käyttäjärooleiksi Plannerin pääkäyttäjä, ilmoitusmyynnin pääkäyttäjä, taittaja ja toimittaja/freelancer.



Kuvio 3. Plannerin käyttötapaukset UC001-UC006



Kuvio 4. Plannerin käyttötapaukset UC007-UC012

Kuten kuvioista 3 ja 4 nähdään, on rooli Plannerin pääkäyttäjä mukana lähes kaikissa käyttötapauksissa. Ilmoitusmyynnin pääkäyttäjä ja taittaja ovat Plannerin osalta mukana vain vastaanottamassa erityyppisiä jutturaportteja. Freelancer/toimittaja lisää juttuja juttupankkiin sekä vastaanottaa maksuja juttujen lisäyksestä.

Seuraavaksi käydään läpi käyttötapausten kuvaukset samassa muodossa kuin ne määrittelydokumentissa esiintyvät.

4.2 Käyttötapaukset

UC001 Plannerin näkymä

Lehden miniatyyrinäkymässä sivut esitetään kaksi aukeamaa vierekkäin, kansi ja takakansi omilla riveillään. Lehden sivuilla esitetään sivujen ilmoituspaikat omilla ikoneillaan. Sivun tila esitetään erinäköisinä sivukuvakkeina. Tiloja ovat tyhjä, juttu kiinnitetty ja materiaali puuttuu.

Sivun yläreunassa ovat lehden numeron tiedot, joissa esitetään tyhjien sivujen määrä, ilmoituspaikkojen määrä ja tarvittavat kontrollit. Miniatyyrinäkymän vieressä esitetään lista elementeistä, joita lehdessä voidaan käyttää.

UC002 Lehden asetukset

Lehden asetuksissa määritellään kaikille lehdille oletusasetukset sivumäärille, ilmoituspaikkojen määrille ja lehden kiinteille sivuille. Tapahtumien kulku:

1) Avataan asetukset-lomake.
2) Valitaan tai lisätään lehti.
3) Syötetään oletusarvot ja lisätään kiinteät sivut.
4) Tallennetaan muutokset.

UC003 Lehden numeron lisäys

Lisätään valitulle lehdelle uusi numero. Tapahtumien kulku:

1) Valitaan yläreunan alasvetovalikosta lehti.
2) Valitaan ”Lisää numero”.
3) Täytetään avautuvalle lomakkeelle numeron nimi, sivumäärä, ilmoitusten määrä, aineistopäivä ja kansien ja sivujen painopäivä.
4) Tallennetaan muutokset.

UC004 Lehden numeron muokkaus

Muokataan valitun lehden numeron tietoja. Tapahtumien kulku:

1) Valitaan yläreunan alasvetovalikosta lehti ja sen numero.
2) Valitaan ”Muokkaa”.
3) Muutetaan avautuvalle lomakkeelle jotakin tai kaikkia seuraavista tiedoista: numeron nimi, sivumäärä, ilmoitusten määrä, aineistopäivä ja kansien ja sivujen painopäivä.
4) Tallennetaan muutokset.

4.2.3 UC005 Juttuhaku

Miniatyyrinäkymän vieressä olevalla juttuhakuelementillä voidaan hakea järjestelmästä sisältöä, juttuja, artikkeleita ja uutisia liitettäväksi lehden sivuille. Tapahtumien kulku:

1) Valitaan haettavat sisältötyypit ja kirjoitetaan hakusana.
2) Valitaan hakutuloksista sisältö ja raahataan se lehden sivulle.

UC006 Ilmoituspaikan lisäys

Miniatyyrinäkymän vieressä olevalla ilmoituspaikkaelementillä voidaan hakea järjestelmästä mediakortissa määriteltyjä ilmoituspaikkoja liitettäväksi lehden sivuille. Tapahtumien kulku:

1) Kirjoitetaan hakusana.
2) Valitaan hakutuloksista ilmoituspaikka ja raahataan se lehden sivulle.

UC007 Ilmoituksen lisäys

Miniatyyrinäkymän vieressä olevalla ilmoituselementillä voidaan hakea järjestelmästä myytyjä ilmoituksia liitettäväksi ilmoituspaikkaan. Tapahtumien kulku:

1) Kirjoitetaan hakusana.
2) Valitaan hakutuloksista ilmoitus ja raahataan se lehden sivulla olevaan ilmoituspaikkaan.

UC008 Jutturaportit

Raportteja on kolme erilaista: lista puuttuvista jutuista numeroittain tai valituista numeroista, juttulista ilmoitusmyyntiin ja juttulista taittajalle sisältäen taitto-ohjeen Ilmoitusmyynnin juttulistaraportti on myös saatavilla ilmoitusmyynnin puolella. Tapahtumien kulku:

1) Valitaan raporttityyppi.
2) Valitaan lehti ja lehden numero.
3) Valitaan ”Luo raportti”.
4) Esitetään raportti.

UC009 Juttupankki

Juttupankki sisältää kaikki sekä julkaistut että julkaisemattomat järjestelmään suoraan syötetyt ja freelancereiden syöttämät jutut. Juttuja voidaan lisätä juttupankkiin, siirtää myytäväksi ja maksaa freelancereille ostetuista jutuista.

UC010 Jutun lisäys

Tällä toiminteella omia juttuja lisätään juttupankkiin. Tapahtumien kulku:

1) Valitaan ”Lisää juttu”.
2) Hyväksytään sopimusehdot.
3) Syötetään jutun nimi, synopsis, raakateksti, kirjoittaja, kuvaaja, kuvat ja kohteessakäyntipäivämäärä.
4) Tallennetaan muutokset.

UC011 Juttujen lisäys myytäväksi

Voidaan lisätä juttupankin juttuja verkkokaupan puolelle myytäväksi. Tämä toiminne on sisällönhallinnan puolella. Tapahtumien kulku:

1) Valitaan jutut juttulistasta.
2) Valitaan ”Lisää myytäväksi”.

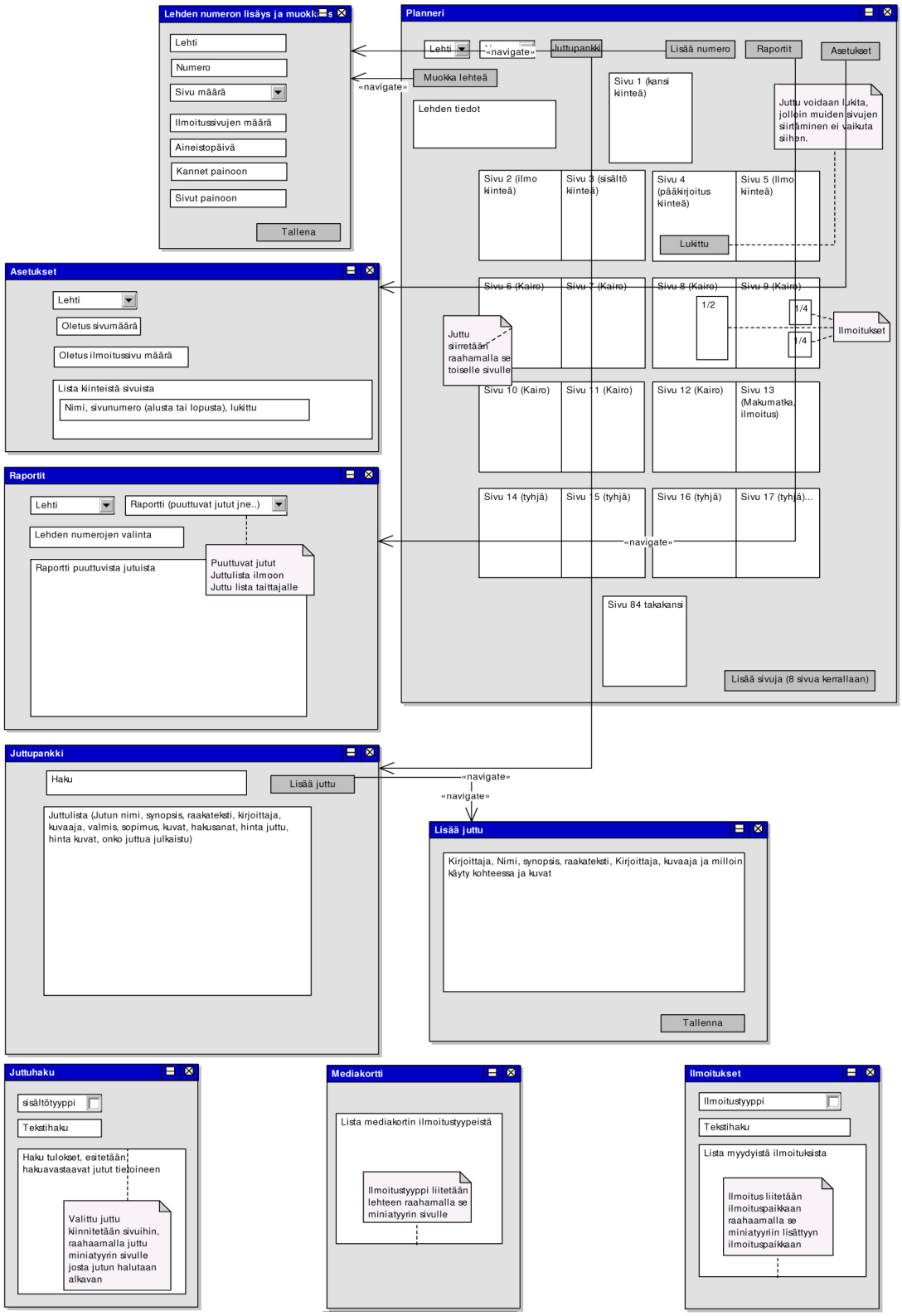
UC012 Jutun maksaminen freelancerille

Juttupankista saadaan lista freelancereilta ostetuista ja maksamattomista jutuista laskutietoineen. Tapahtumien kulku:

1) Valitaan ”Hae maksamattomat jutut”.
2) Valitaan ”Tulosta laskuraportti”.

4.3 Käyttöliittymäsuunnitelma

Plannerin käyttöliittymä suunniteltiin seuraavanlaiseksi:



Kuvio 5. Plannerin käyttöliittymän suunnitelma

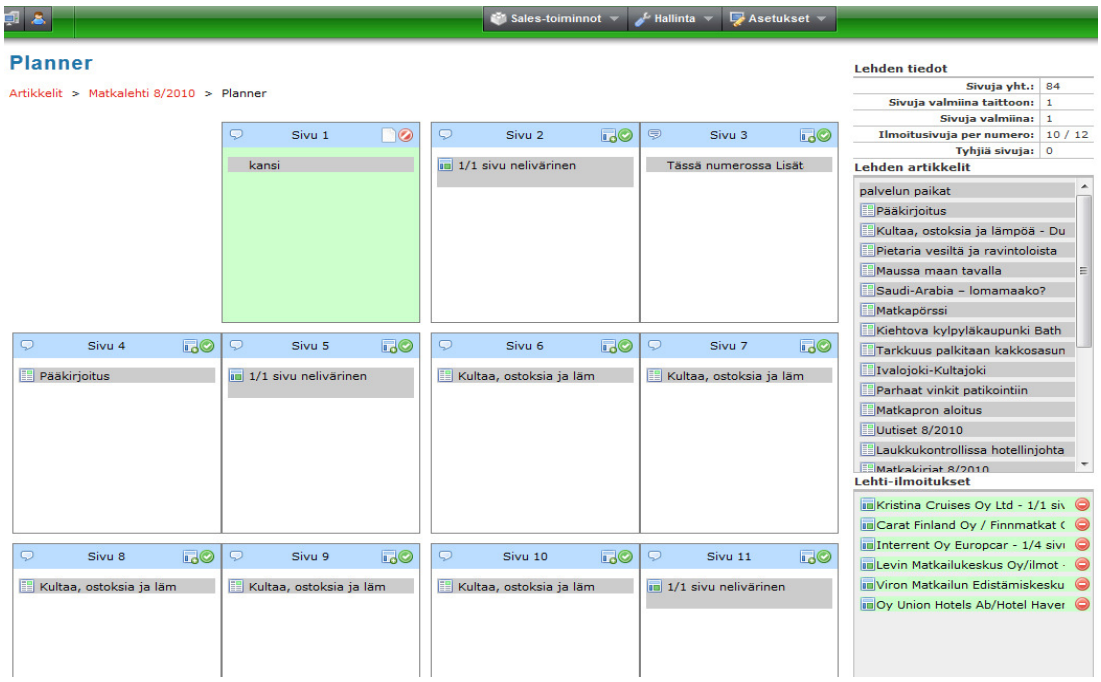
Oikean yläkulman iso kuva esittää käyttötapausta UC001 Plannerin näkymä, josta voidaan siirtyä sen alanäkymiin. Kuvassa ylimpänä nähdään siirtymävalikot lehdelle ja sen numeroille sekä linkit toiminteisiin Juttupankki, Lisää numero, Raportit, Asetukset ja Muokkaa lehteä. Vasen ylänurkka esittää lehden perustietoja. Päänäkymän keskellä esitetään varsinaiset sivut numerojärjestyksessä ja sivuilla niiden varsinainen sisältö.

Vasemman reunan toiminteet kuvaavat järjestyksessä ylhäältä alas käyttötapauksia UC003 Lehden numeron lisäys ja UC004 Lehden numeron muokkaus, UC002 Asetukset, UC008 Raportit, UC009 Juttupankki ja UC010 Lisää juttu.

Alimpana kuvassa näkyvät toiminteet ovat järjestyksessä vasemmalta oikealle käyttötapaukset UC005 Juttuhaku, UC006 Ilmoituspaikan lisäys ja UC007 Ilmoituksen lisäys. Kyseiset toiminteet suunniteltiin visuaalisesti päänäkymän UC001 oikealle puolelle.

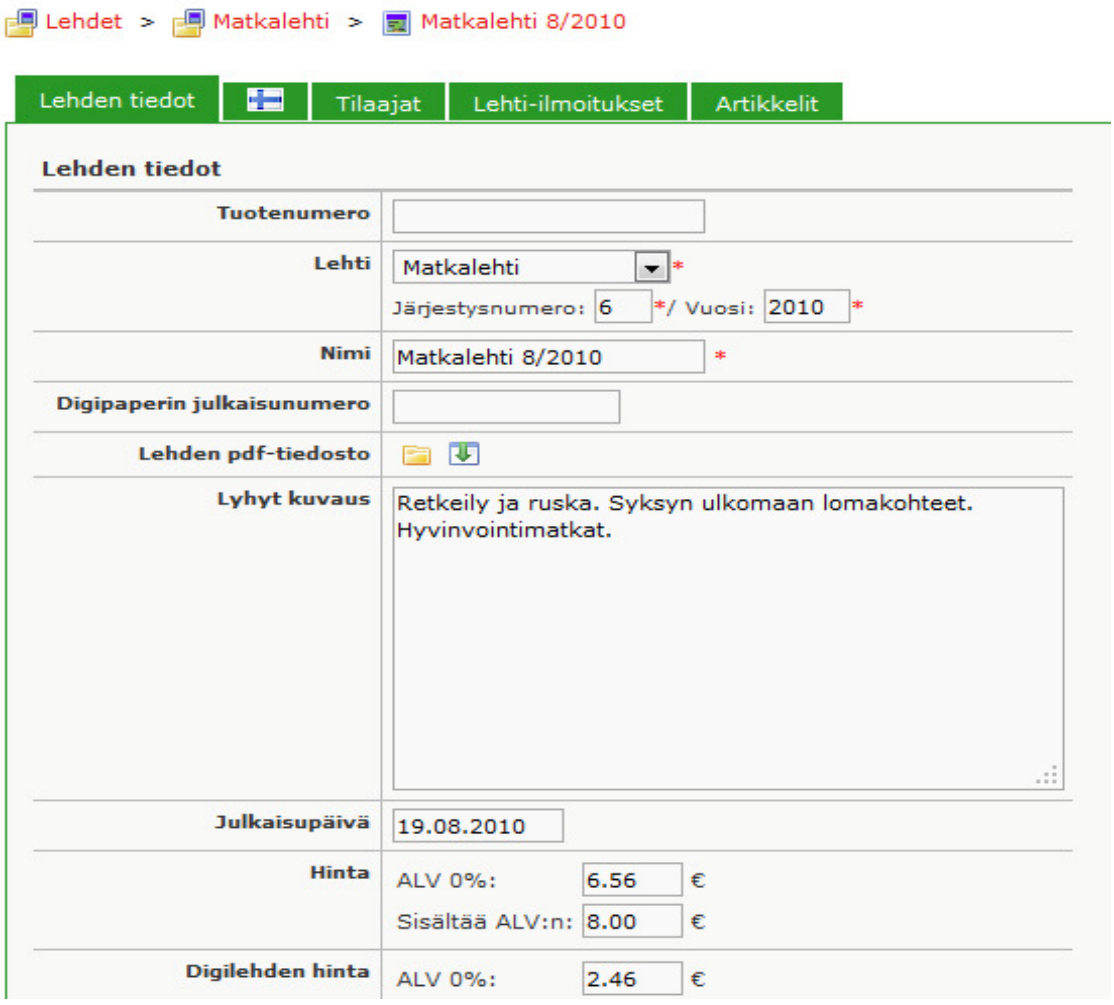
4.4 Toteutunut järjestelmä

Käyttöliittymän lopullinen muoto päätyi pääpiirteissään samankaltaiseksi kuin se suunniteltiin. Seuraavat kuvat havainnollistavat lopullista versiota ja sen eroja suunnitelmaan.



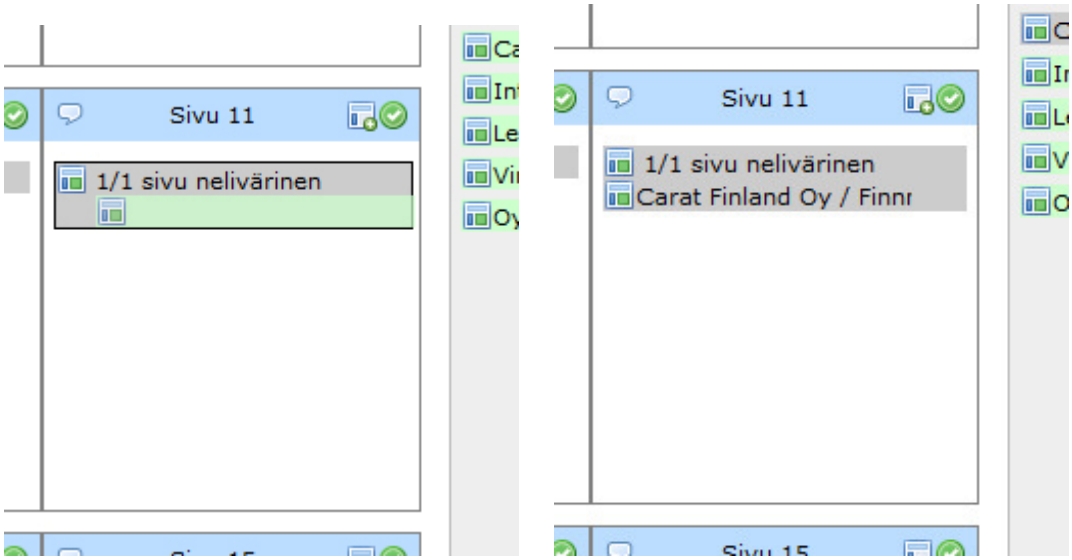
Kuvio 6. Plannerin päänäkymä

Kuviosta 4 saadaan yleiskatsaus Plannerin toteutuneeseen päänäkymään. Lehden valintakontrollit sekä linkit numeron lisäykseen, raportteihin, asetuksiin ja lehden muokkaukseen ovat siirtyneet ylävalikon Hallinta-valikon alle. Toisin kuin suunnitelmassa, ensimmäinen sivu esitetään samalla rivillä kuin muutkin sivut tilan säästämiseksi. Lehden tiedot on siirretty oikeaan yläkulmaan. Artikkelit ja lehti-ilmoitukset on haettavissa oikeasta reunasta mutta ne tulee määritellä Lehden asetukset -osiossa ennen ilmestymistään listaukseen.



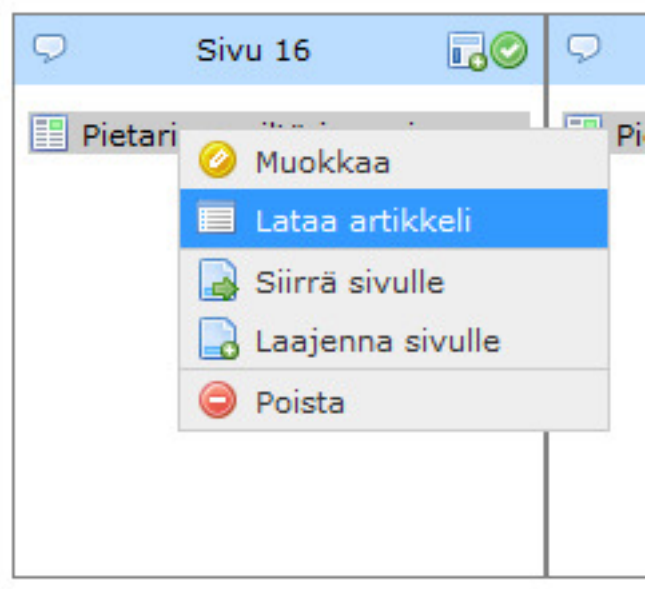
Kuvio 7. Lehden asetukset

Kuviosta 5 nähdään, kuinka tiedot syötetään lehden asetuksiin. Muuttuneina tietoina on Digipaperin julkaisunumero, lehden pdf-tiedosto ja digilehden hinta. Välilehdiltä voidaan määritellä lisäkieliä käyttöön suomen kielen lisäksi. Muut ominaisuudet ovat lehden tilaajien listaus ja lehden ilmoitusten ja artikkeleiden määritys.



Kuvio 8. Ilmoituksen asettaminen ilmoituspaikkaan

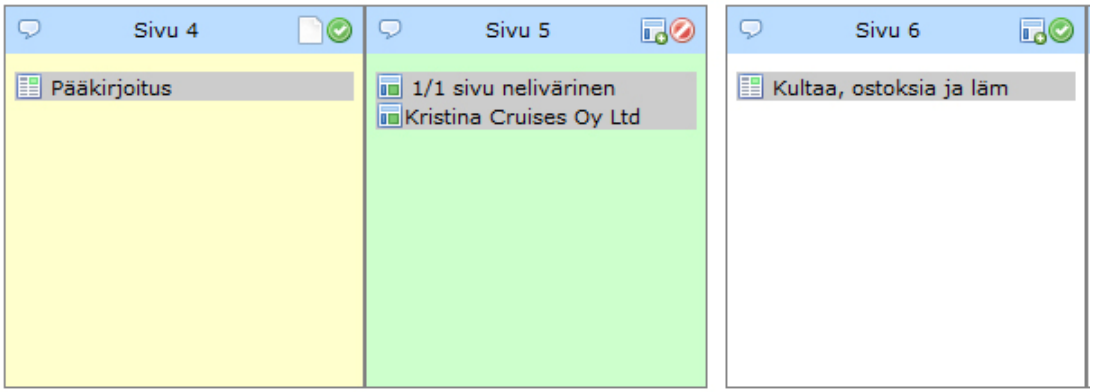
Oikean reunan ilmoituksista voidaan ilmoitus vetää hiirellä ilmoituspaikkaan. Kuvio 8 esittää kyseistä tilannetta. Ilmoitusta ei voida siirtää sivulle itsenäisenä, vaan se vaatii aina ilmoituspaikan sijoituspaikakseen. Jos käyttäjä yrittää kuitenkin tehdä näin, siirtyy ilmoitus animoituna takaisin oikean laidan listaukseen.



Kuvio 9. Kontekstivalikko

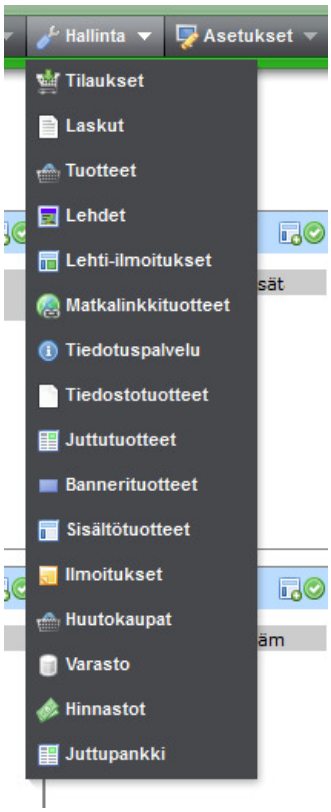
Alkuperäisestä suunnitelmasta poiketen Planneriin on toteutettu kontekstivalikko. Kuvion 9 mukainen valikko saadaan esille painamalla juttua, ilmoitusta tai ilmoituspaikkaa hiiren oikealla napilla. Avautuvasta valikosta voidaan siirtyä suoraan kyseisen elementin muokkaustilaan, ladata elementti tarkastelta-

vaksi tai siirtää tai laajentaa (eli kopioida) elementti suoraan määrätylle sivulle. Kontekstivalikko lisättiin, kun huomattiin lehden koon kasvavan ja tarvittiin vaihtoehtoinen tapa liikutella elementtejä sivujen välillä.



Kuvio 10. Sivujen eri tilat

Kuviosta 10 nähdään lehden sivuille lisätyt toiminteet. Vasemman ylänurkan puhekuplasta sivulle voidaan lisätä vapaa tekstikommentti. Oikean ylänurkan napeista sivu voidaan merkitä valmiiksi taittoon tai valmiiksi painoon. Sivun taustaväri muuttuu tilan mukaan: keltainen on valmis taittoon, vihreä valmis painoon ja valkoinen keskeneräinen sivu.



Kuvio 11. Plannerin muiden toiminteiden linkit

Kuviosta 11 nähdään Plannerin tukitoiminteiden sijainti. Linkit on siirretty Plannerin päänäkymästä järjestelmän ylävalikkoon samalle tasolle kuin muutkin tilauksenhallinnan toiminteet. Listassa on paitsi lehtien ja lehti-ilmoitusten hallinta, myös juttujen muunto juttutuotteiksi sekä juttupankki. Loput listan toiminteista ovat Planneriin suoraan liittymättömiä.

5 KEHITYSPROSESSI

5.1 Prosessi käytännössä

Planner määriteltiin projektisuunnitelmassa osaksi matkalehti.fi-toiminnanohjausjärjestelmää. Plannerin toteutus uskottiin minulle, järjestelmän muiden osien jäädessä muun projektiryhmän vastuulle. Projektin edetessä olin kuitenkin tarpeen vaatiessa mukana myös muiden, opinnäytetyöhön kuulumattomien, osien toteutuksessa. Plannerin tekninen määrittely tehtiin etukäteen, enkä ollut prosessin kyseisessä vaiheessa mukana.

Plannerin toteutukselle annettiin koko projektin suunnitelmassa aikaa yksi ja puoli kuukautta. Tätä ennen tulisi suunnitelman mukaan valmistua paitsi koko matkalehti.fi -järjestelmän perusarkkitehtuuri, myös järjestelmän kaikki ne toiminnallisuudet, joita Planner tulisi käyttämään hyväksi, kuten juttupankki ja tilaustenhallinta. Koska loput järjestelmän osat eivät olisi suoraan Planneriin kytköksissä, voitaisiin niitä kehittää Plannerin kanssa rinnakkain ennen järjestelmän viimeistelyä ja luovutusta.

Vaikka osa järjestelmän määrittelystä oli vielä kesken, tiukahkon aikataulun vuoksi projektia lähdettiin viemään eteenpäin. Määrittelyä jatkettaisiin projektin edetessä. Aikataulun edettyä Plannerin tekniseen suunnitteluvaiheeseen, eivät Plannerin vaatimat toiminnallisuudet kuitenkaan olleet vielä valmiita.

Muuttuneeseen tilanteeseen pyrin vastaamaan suunnittelemalla uudelleen Plannerin teknisen toteutuksen itsenäiseksi kokonaisuudeksi. Ne osat Plannerista, jotka tarvitsisivat muita järjestelmän osia, kuten juttupankkia käyttävä juttulista, jätettäisiin sisältöä odottamaan. Muut toiminteet toteuttaisin määrittelyn mukaisesti.

Ryhdyin toteuttamaan Planneria ja lyhyehkössä ajassa sen perustoiminnallisuus oli esitettävissä: lehden asetusten määritys, sivujen visuaalinen esitys ja elementtien siirtely sivujen välillä. Tarkoituksena oli katsoa yhdessä asiakkaan kanssa, vastasiko Planner määrittelyä ja voitaisiinko toteutusta jatkaa. Tässä vaiheessa prosessia asiakas oli kuitenkin siirtänyt huomionsa järjestelmän muihin, Planneriin suoraan liittymättömiin osiin. Uudelleenpriorisoinnin johdosta Plannerin toteutus keskeytettiin ja siirryin väliaikaisesti auttamaan muuta projektiryhmää.

Kun pääsin taas jatkamaan Planneria, aikataulusta oltiin jäljessä. Jotta projektin valmistuspäivästä voitaisiin pitää kiinni, määriteltiin Plannerin joitain ominaisuuksia suullisesti ja sähköpostin välityksellä uudelleen. Vaikka tavoitteena oli rajata toiminnallisuuksia ja siirtää niitä toteutettavaksi tuleviin versioihin, asiakkaan toiveesta toiminnallisuuksien määrää päinvastoin lisättiinkin entisestään.

Plannerin vaatimat järjestelmän muut osat eivät kuitenkaan olleet vieläkään valmiita, joten siirryin projektissa Plannerin ja järjestelmän muiden osien toteutuksen välillä useita kertoja. Tavoitteenani oli mahdollisuuksien mukaan nopeuttaa Plannerin etenemistä auttamalla Planneria tukevien moduulien toteutuksessa. Järjestelmän muiden osien tultua Plannerin kannalta valmiiksi pääsin viimeistelemään oman työni. Asiakkaalta saatujen kommenttien jälkeen hioin käyttöliittymän yksityiskohdat lopulliseen muotoonsa ja tuote luovutettiin asiakkaalle.

5.3 Ongelmat

Tuotteen lopullinen muoto poikkesi määritellystä ja projektin kokonaisaikataulu ylitettiin: voitaneen siis sanoa prosessissa olevan kehittämisen varaa. Näin siis siitä huolimatta, että asiakas sai lopulta tuotteensa ja oli siihen tyytyväinen. Missä siis tehtiin virheitä ja kuinka ne voitaisiin välttää?

Projektin vaihejakomalliksi valittiin vesiputousmalli. Kyseisen mallin kantava ajatus on ensiksi analysoida ongelma ymmärrettävään muotoon, suunnitella siihen toimiva ratkaisu ja lopuksi toteuttaa ja testata ratkaisu (Haikala–Märijärvi 2004, 41). Asiakkaalla ei ollut määrittelyvaiheessa tarkkaa kuvaa siitä, millainen järjestelmä tarvittiin, eikä asiakkaan liiketoimintamalli ollut toteuttajalle ennestään tuttu. Tässä vaiheessa olisi siis osapuolten ollut tärkeä huomata, että määrittely tulee hyvin todennäköisesti muuttumaan projektin edetessä. Toimivampaa olisi ollut jakaa kehitystyö pienempiin, muutamien viikkojen sykleihin ja tarkastella iteraatiota syklin päätteeksi. Näin olisi voitu tarkastella ratkaisujen käyttökelpoisuutta jo projektin alussa ja suurimpiin ongelmiin saada ratkaisumahdollisuus inkrementaalisen mallin mukaisesti. (Haikala–Märijärvi 2004, 45.)

Projektia ryhdyttiin toteuttamaan ennen määrittelyn valmistumista. Eräs tärkeimmistä syistä tuotantoprosessin vaiheistukseen on prosessin luonteessa:

järjestelmän vaatimuskuvaukset tulee muuntaa kuvauksiin vastaavaksi ohjelmistoksi. Koska kuvausten ero verrattuna varsinaiseen ohjelmistoon on abstraktiotasolla suuri, on luontevaa vaiheistaa kuvausprosessi. (Haikala–Märijärvi 2004, 63.) Ilman toimivaa spesifikaatiota on toteutus hyvin haastavaa ja erittäin aikaavievää: määrittely olisi ollut hyvä viedä loppuun ennen projektissa etenemistä. Huomattavaa on myös, että hyvin tehty määrittely aikaansaa ajallisia ja taloudellisia säästöjä projektin myöhemmissä vaiheissa.

Planner pyrittiin suunnittelemaan ennen Planneria tukevien toiminnallisuuksien valmistumista. Suunnitteluprosessin tarkoitus on kääntää määrittely tekniseen muotoon ja sen yleisinä tavoitteina voidaan pitää selkeyttä ja ymmärrettävyyttä, tehokkuutta, luotettavuutta, ylläpidettävyyttä ja siirrettävyyttä (Haikala–Märijärvi 2004, 81–82). Plannerin tekninen suunnitelma jäi puuttuvien komponenttien vuoksi näistä ominaisuuksista vajaaksi. Plannerin suunnitelmasta – ja näin ollen myös toteutuksesta – olisi tullut parempi, jos suunnitelma olisi tehty vasta, kun muut erilliset osat olisivat olleet valmiit.

Projektin aikataulu oli melko haastava. Vaikka projektin osien aikatauluttaminen onkin usein sen vaikein osuus, työmääräarviot on hyvä pitää realistisina ja pyrkiä kasvattamaan projektisuunnitelman toteuttamiskelpoisuutta muilla tavoin (Haikala–Märijärvi 2004, 227). Suunnitteluvaiheessa olisi osapuolten ollut tärkeä havaita, ettei tavoitteiden saavuttaminen suunnitelluilla aikatauluilla tulisi olemaan todennäköistä. Vähintäänkin olisi ollut aikataulutavoiteiden ylittyessä hyvä miettiä yhdessä, siirretäänkö tavoitetta tuonnemmaksi vai rajataanko ominaisuuksia seuraavaan iteraatioon.

5.4 Prosessin kehitys

Vaikka projekti oli tarkoitus toteuttaa vesiputousmallin mukaisesti, kehittäjät toimivat käytännössä melko itsenäisesti. Järjestelmästä saatiin hyvä ja toimiva kokonaisuus määrittely- ja suunnitteludokumenttien yksinkertaisuudesta huolimatta. Voidaankin sanoa, että suunnitelmia ja virallisia sopimuksia tärkeämmässä asemassa olivat tiivis keskustelu asiakkaan kanssa ja tilanteeseen mukautuminen.

Ketterässä ohjelmistokehityksessä arvostetaan sen virallisen manifestin mukaisesti:

- yksilöitä ja vuorovaikutusta enemmän kuin prosesseja ja työkaluja
- toimivaa sovellusta enemmän kuin kokonaisvaltaista dokumentaatiota
- asiakasyhteistyötä enemmän kuin sopimusneuvotteluita
- muutokseen reagoimista enemmän kuin suunnitelman noudattamista.

Kyseisessä manifestissa painotetaan myös, että vaikka edellä mainitun listan oikeallakin puolella on arvoa, arvostavat ketterien menetelmien käyttäjät listan vasemman puolen asioita enemmän. (Agile Alliance 2012.)

Olisiko projekti kannattanut siis toteuttaa ketterin menetelmin? Kenties, sillä kuten huomattiin, oli prosessi jo hyvin lähellä kyseistä metodologiaa, eikä siihen mukautuminen olisi vaatinut juuri tarkemman metodin valitsemista enempää. Koko projektin tarkoitus oli tuottaa asiakkaalle mahdollisimman nopeasti toimiva, laadukas ja asiakkaan toiveita vastaava ohjelmisto ottaen vastaan mahdolliset muutokset myös projektin myöhemmissä vaiheissa – kaikki nämä ovat ketterien menetelmien periaatteita.

Eräs tällainen menetelmä on Scrum. Se perustuu empiiriseen prosessinhallintateoriaan, jossa painotetaan kolmea asiaa: osallisten välistä läpinäkyvyyttä, edistymisen jatkuvaa tarkastelua ja prosessin ja materiaalien sopeuttamista (Scrum.org 2012). Määritelmänsä mukaan Scrum on kevyt ja yksinkertainen ymmärtää, mutta erittäin vaikea hallita hyvin.

Scrum ei ole varsinaisesti tuotekehitysprosessi vaan viitekehys. joka tukee erilaisia tekniikoita. Scrumtiimi koostuu tuotteen vaatimuksista vastaavasta tuoteomistajasta, kehitystiimistä itsessään ja Scrummasterista, joka varmistaa, että tiimi pitäytyy Scrumin teoriassa, käytännöissä ja säännöissä. Scrummaster ei siis ole projektipäällikkö, vaan palveleva johtaja: hänen tehtävänsä on paitsi toimia tulkkina tuoteomistajan ja kehitystiimin välillä, myös valmentaa kehitystiimiä ja poistaa mahdolliset esteet etenemisen tieltä. (Scrum.org 2012.)

Tapahtumanhallinnan ydin on enintään kuukauden pituinen sprintti, jonka aikana toteutetaan uusi tuoteversio. Sprintin aikana ei oteta tavoitteeseen vaikuttavia muutoksia vastaan ja se voidaan keskeyttää vain tuoteomistajan päätöksestä. Väli- ja päiväpalaverien lisäksi tiimiä kannustetaan tarkastelemaan ja kehittämään prosesseja sprinttien välissä pidettävässä sprintin retrospektiivissä. (ATSC–Mountain Goat Software 2008.)

Tuote kehittyy Scrumissa vaiheittain kehitysjonossaan, joka priorisoidaan. Kehitysjonoa työstetään jatkuvasti tuoteomistajan ja kehitystiimin kesken ja se on olemassa yhtä kauan kuin itse tuotekin. (ATSC–Mountain Goat Software 2008.) Yksittäisen sprintin tehtävälista rakennetaan kehitysjonosta valituista kohteista.

Kun tuote on valmis, on läpinäkyvyyden turvaamiseksi kaikkien osapuolten tärkeä ymmärtää valmiin määritelmä. Oleellista on, että jokainen tuoteversio pohjautuu kaikkiin edellisiin ja että versioiden yhteensopivuus testataan. (Scrum.org 2012.) Usein määritelmä laajeneekin tarkoittamaan tiukempia kriteerejä korkealle laadulle.

Scrumin avulla Plannerin kaltainen projekti voidaan viedä hallitummin loppuun asti, eikä tällaiseen viitekehykseen mukautuminen vaadi suuria muutoksia. Projektipäällikkö on helposti muutettavissa Scrummasteriksi ja asiakkaan tehtäväksi jää valita sopiva yksittäinen edustaja joukostaan tuoteomistajaksi. Toiminnallisuuskokonaisuudet voidaan jakaa sprintteihin. On tyypillistä, että laajoissa projekteissa asiakkaan vaatimukset muuttuvat kesken projektin: tällöin on luontevaa mukauttaa prosessia vaatimusten mukaan, eikä toisinpäin. Kun asiakas-toimittaja-vastakkainasettelu poistetaan ja kaikki osapuolet sitoutetaan itse kehitysprosessiin, on edistyminen läpinäkyvämpää ja palaute välittömämpää.

6 YHTEENVETO

Opinnäytetyön tavoitteena oli toteuttaa Polarlehdet Oy:lle aikakauslehden suunnittelutyökalu Planner. Tavoitteena oli myös kytkeä työkalu osaksi asiakkaalle samanaikaisesti toteutettavaa, laajempaa toiminnanohjausjärjestelmää ja siten auttaa kehittämään asiakkaan liiketoimintaprosesseja.

Projektin lopputulos oli valmis tuote, joka helpotti asiakkaan toimintaa. Tuotteen ominaisuudet olivat lopullisessa versiossa suunniteltua laajemmat ja käyttöliittymä yleisesti ottaen selkeämpi. Asiakas oli lopputuotteeseen tyytyväinen.

Raportin teoreettiseen taustaan löytyi helposti tietoa sekä kirjallisista että Internet-lähteistä. Käytetyt teknologiat olivat järjestelmäalustan mukaan etukäteen määriteltyjä, mikä osaltaan edesauttoi projektin teknistä suunnittelua. PolarCore-alusta oli teknologioita haastavampi purkaa auki alustan suljetun luonteen vuoksi.

Vaikka projekti itsessään saatiin valmiiksi ja asiakas oli siihen tyytyväinen, kehitysprosessissa olisi runsaasti parannettavaa. Tärkeää olisi paitsi vaiheistaa projektin aikataulu realistisesti, myös valvoa työn vaiheiden loogista etenemistä. Myös sopivan vaihejakomallin valinnan merkitystä ei tule väheksyä.

Toteutettuani tämän opinnäytetyön olen oppinut paljon laajoista ohjelmistoprojekteista käytännössä. Erityisesti kiinnitin huomiota eroavaisuuksiin, joita teoreettisten prosessimallien ja reaalimaailman käytäntöjen välillä ilmenee. Tulevissa projekteissa arvelen osaavani paremmin kiinnittää huomiota niihin ongelmiin, joita projektin edetessä esiintyi.

Työn ohella toteutetun toiminnanohjausjärjestelmän julkinen sivusto on nähtävissä osoitteessa http://www.matkalehti.fi/.

LÄHTEET

Agile Alliance 2012. The Manifesto for Agile Software Development. Osoitteessa http://www.agilealliance.org/the-alliance/the-agile-manifesto/. 22.4.2012.

Facebook 2012. MySQL at Facebook. Osoitteessa http://www.facebook.com/MySQLatFacebook. 10.4.2012.

Fowler, M. – Scott, K. 2002. UML. ToolKit. Suom. Eero Sarkkinen. Jyväskylä: Docendo.

Haikala, I. – Märijärvi, J. 2004. Ohjelmistotuotanto. Helsinki: Talentum.

IBM Corporation 2012. Types of modeling diagrams. Osoitteessa http://publib.boulder.ibm.com/infocenter/rtnlhelp/v6r0m0/index.jsp?topic=/com.ibm.xtools.modeler.doc/topics/rdiagtype.html. 10.4.2012.

Kofler, M. 2005. The Definitive Guide to MySQL 5. New York: Apress.

Korpela, J. – Linjama T. 2004a. XHTML-käsikirja. Jyväskylä: Docendo.

– 2004b. CSS-tyylit. Jyväskylä: Docendo.

Mountain Goat Software – ATSC 2008. Scrum Overview. Osoitteessa http://epf.eclipse.org/wikis/scrum/. 20.8.2008.

Mozilla Developer Network 2012. JavaScript Reference. Osoitteessa https://developer.mozilla.org/en/JavaScript/Reference. 12.4.2012.

Negrino, T. – Smith D. 2007. JavaScript: tehokas hallinta. Suom. Mikko Kamppila. Helsinki: Readme.fi.

Object Managerment Group 2012. Introduction To OMG's Unified Modeling Language. Osoitteessa http://www.omg.org/gettingstarted/what_is_uml.htm. 8.3.2012.

Oracle Corporation. 2012a. Market Share. Osoitteessa http://www.mysql.com/why-mysql/marketshare/. 11.4.2012.

– 2012b. MySQL Customer: Google. Osoitteessa http://www.mysql.com/customers/view/?id=555. 11.4.2012.

Rantala, A. 2002. PHP: web-ohjelmoinnin peruskirja. Jyväskylä: Docendo.

Scrum.org 2012. jQuery: The Scrum Guide. Osoitteessa http://www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%20FI.pdf#view=fit. 12.4.2012.

The jQuery Foundation 2012. jQuery: The Write Less, Do More JavaScript Library. Osoitteessa http://jquery.com/. 12.4.2012.

The PHP Group 2012. Database Extensions. Osoitteessa http://fi.php.net/manual/en/refs.database.php. 11.4.2012.

Twitter 2012. MySQL at Twitter. Osoitteessa http://engineering.twitter.com/2012/04/mysql-at-twitter.html. 9.4.2012.

W3C 2011. Selectors Level 3 W3C Recommendation. Osoitteessa http://www.w3.org/TR/2011/REC-css3-selectors-20110929/. 29.9.2011.

W3C 2012. W3C Mission. Osoitteessa http://www.w3.org/Consortium/mission. 12.4.2012.

Wikimedia Commons 2011. Hierarchy of diagrams in UML 2.2. Osoitteessa http://en.wikipedia.org/wiki/File:UML_diagrams_overview.svg. 10.9.2011.