Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

MICTIS15

2017

Kirsi Jokimies

DOKUMENTTIKÄSITTELIJÄN OHJELMOIMINEN JAVA -KIELELLÄ

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikan koulutus

2017 | 30

Kirsi Jokimies

DOKUMENTTIKÄSITTELIJÄN OHJELMOIMINEN JAVA -KIELELLÄ

Työn tarkoituksena on luoda ohjelmistokomponentti joka osaa täyttää PDF lomakkeet automaattisesti. Lomakepohjan kenttiin täytetään saadut tiedot ja dokumentti voidaan tämän jälkeen tallentaa tai tulostaa. Järjestelmään on pystyttävä lisäämään uusia lomakepohjia tarpeen mukaan, ja niiden kentät on pystyttävä linkittämään tietokannan kenttiin. Lomakkeen kenttien linkitystieto tallennetaan tietokantaan. Lisäksi ohjelmistokomponentin on osattava hallita tilanne, jolloin annettu tieto ei mahdu sille varattuun kenttään. Toteutuksessa luodaan yleiskäyttöinen palvelu, jota voivat kutsua kaikki palvelua tarvitsevat ohjelmistokomponentit. Opinnäytetyö tehdään Mediconsult Oyj:n toimeksiantona.

Kehitystyö aloitetaan valitsemalla käytettävä dokumenttiformaatti ja sitä tukeva Java -kirjasto. Tämän jälkeen luodaan loogiset kuvaukset käyttötilanteista ja vaadituista toiminnollisuuksista. Lisäksi suoritetaan toiminnallisuuden tarkempi määrittely. Ohjelmointityössä käytetään Java EE:tä ja PDFBox Java -kirjastoa. Ohjelmiston toiminnallisuus varmistetaan kirjoittamalla toiminnollisuuksien testitapaukset rinnakkain varsinaisen ohjelmakoodin kanssa.

Työn tuloksena on palvelu, jonka avulla voidaan käsitellä ja täyttää annettuja PDF -lomakkeita. Lomakkeiden analysointi ja lomakkeen sekä tietokannan kenttien linkittäminen toisiinsa suoritetaan pääkäyttäjän toimesta ja linkityksen jälkeen ohjelmisto osaa täyttää tiedot lomakkeelle itsenäisesti. Ohjelmisto osaa lisäksi käsitellä tilanteen, jossa kirjoitettava tieto ei mahdu lomakkeella sille varattuun tilaan. Tällöin mahdollisia toimintatapoja on kaksi. Joko tiedolle luodaan uusi kenttä lomakkeen loppuun, tai varattua alkuperäistä kenttää venytetään niin, että tieto saadaan mahtumaan.

Toteutuksella saadaan useita etuja. Lomakkeiden täyttäminen tulostusta varten on helppoa ja nopeaa. Järjestelmään pystytään lisäämään mikä tahansa PDF lomake linkittämällä lomakkeen kentät tietokannan kenttiin. Lisäksi niissä tilanteissa, joissa annettu tieto ei mahdu sille varattuun kenttään, ohjelmakomponentti tarjoaa käyttäjälle paremman käyttäjäkokemuksen siirtämällä tiedon kokonaisena lomakkeen loppuun, tai venyttämällä alkuperäistä kenttää.

ASIASANAT:

ohjelmointi, java, sähköiset lomakkeet, ohjelmistosuunnittelu

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Degree programme

2017 | 30

Kirsi Jokimies

AUTOMATED DOCUMENT MANIPULATION

The purpose of this thesis was to implement a software module which would fill up PDF forms automatically. It had to be possible to add further form templates into the system and perform configuration for new templates. All configuration data was saved into a database. In addition, the software module needed to know how to handle situations where given data does not fit into the reserved field. This assignment was commissioned by Mediconsult Oyj.

Development work was started with selecting used document format and Java library for document handling. Work continued with creating use cases and logical graphs of the required functionality. Requirements of the software module was written down in a greater detail. The programming work was done using Java EE and PDFBox Java -library. Test driven development was used as a method to ensure the development quality.

As a result, a software module which can fill up pre-defined PDF forms was created. It supports mapping document fields and database fields together and saving mapping data into a database. Additionally, it can handle situations where data does not fit into the given field. It either moves data to the end of the document, or stretches the original field – along with the rest of the document. Implemented software module is a service which is available for all other software components in the system.

This Implementation has several benefits. Filling up a form with customer's information is fast and easy. Implementation offers an easy way to enter new documents into a system and configure them so that they can be filled up automatically. Furthermore, it offers a great benefit in cases where too small space is reserved for information in the form template. In such cases, the software module can stretch the field so that the information fits. Another option is to create new field to the end of the document for this information. Thus, end user enjoys an improved user experience.

Click here to enter text.

KEYWORDS:

programming, java, software development, electronic forms

SISÄLTÖ

LIITTEET

KUVAT

SANASTO

AcroForm	PDF Formaatin alkuperäinen lomaketeknologia
AGPL	Affero General Public License, Lisenssi määrittely joka edellyttää ohjelmakoodin julkaisua avoimen lähdekoodin kehittäjäyhteisölle.
Apache (ASF)	Apache Software Foundation on avoimeen lähdekoodiin keskittynyt ohjelmistokehittäjien yhteisö.
Back End	Palvelinpuolen ohjelmistokomponentti
BPMN	Business Process Model and Notation, prosessikaavio jolla kuvataan toiminnallisuutta, suoritusta tai operaatiota.
Clean Code -periaate	Ohjelmistokehityksen periaate, jolla pyritään varmistamaan tuotetun koodin luettavuus ja ylläpidettävyys
CSS	Cascading Style Sheets, ohjelmointikieli jolla kuvataan HTML -kielellä kirjoitetun dokumentin ulkoasu
Dto	Data Transfer object, on olio, joka paketoi datan siirrettäväksi yhdeltä järjestelmältä toiselle. Tyypillisesti palvelutasolta käyttöliittymälle.
Eclipse	Integroitu kehitysympäristö jota käytetään usein Java -kielen kehittämiseen. Tukee muitakin ohjelmointikieliä.
Front End	Käyttöliittymäkomponentti
GET	HTTP -kutsu jota käytetään yleisesti resurssin hakua varten.
GNU LGPL	GNU Lesser General Public Licence, ohjelmistolisenssi, jolla voidaan lisensoida ilmaiset ohjelmakomponentit siten, että lähdekoodia ei tarvitse paljastaa.
Hibernate	Java -kielen olio-relaatioiden linkitystyökalu. Hibernaten avulla ohjelmointikielen oliot linkitetään relaatiotietokannan kenttiin.
HTML5	HTML5 on web teknologia, joka mahdollistaa entistä näyttävämmät web sivut ja palvelut, ja joka takaa niiden toimivuuden mobiililaitteissa.
Java EE	Java Enterprise Edition, Oraclen yritysratkaisuille tarjoama ohjelmointialusta
JavaScript	Korkean tason tyypittämätön ohjelmointikieli.
JPA	Java Persistence API, rajapintamäärittely joka määrittelee, miten tietoa käsitellään Java -sovelluksissa, ja miten se tallennetaan tietokantaan

JUnit	Java -kielen kirjasto, jonka avulla voidaan toteuttaa yksikkötestejä varsinaiselle ohjelmakoodille.
MariaDB	Avoimen lähdekoodin tietokantahallintajärjestelmä, joka perustuu MySQL tietokantahallintajärjestelmään
Maven	Ohjelmistoprojektin hallintatyökalu joka hallitsee projektin kääntämisen, raportoinnin ja dokumentoinnin.
MySQL	Oraclen tarjoama tietokantahallintajärjestelmä
PDF	Portable document format, Adoben kehittämä dokumenttiformaatti
PDFBox	Apachen kehittämä avoimen lähdekoodin Java kirjasto, jonka avulla voidaan käsitellä PDF tiedostoja.
REST	Representational State Transfer, arkkitehtuurityyli jonka avulla toteutetaan verkkosovelluksia. RESTkäyttää tiedonsiirtoon yksinkertaisia HTTP komentoja.
SVN	Subversion, versionhallintatyökalu
Wildfly	Java -kielellä toteutettu sovelluspalvelin.
XML	Extensible Markup Language, on kieli, joka määrittelee joukon sääntöjä, joiden avulla voidaan luoda sekä ihmisen, että koneen ymmärtämiä dokumentteja.
XFA	PDF Formaatin kehittyneempi lomaketeknologia
POST	HTTP -kutsu joka käytetään esimerkiksi täytetyn lomakkeen tietojen lähettämiseen palvelimelle.
ODF	Open Document Format, Avoin, XML -pohjainen dokumenttiformaatti jolla voidaan toteuttaa taulukkolaskentadokumentteja, kaavioita, esityksiä ja tekstinkäsittelytiedostoja
ODFDom	Apachen Java -kirjasto ODF dokumenttien käsittelyä varten

1 JOHDANTO

Digitalisaatio on saavuttamassa kuntien ja valtion virastot, terveydenhuollon toimipisteet sekä yksityiset palveluntarjoajat. Sähköisen asioinnin lisääntymisestä huolimatta edelleen on useita asiakirjoja, jotka halutaan tulostaa paperille; virallisia todistuksia sekä erilaisia yhteenvetoja ja lomakkeita.

Tämän opinnäytetyön toimeksiantona on dokumenttikäsittelijän toteuttaminen. Dokumenttikäsittelijän avulla asiakkaan tiedot täytetään valmiiseen lomakepohjaan myöhempää tulostamista varten. Ohjelma toimii osana isompaa tietokantajärjestelmää ja tarjoaa palvelun kaikille järjestelmän osa-alueille.

Työn tavoitteena on luoda ohjelmisto joka analysoi pdf-muotoisen lomakkeen ja tallentaa pohjan sekä tulostusohjeen myöhempää käyttöä varten. Käytettäessä valmista pohjaa, tulostusohje tarkoittaa dokumentin kenttien linkittämistä käyttöliittymän kenttiin. Koska kyseessä on palvelu, ei työ varsinaisesti sisällä käyttölogiikkaa, vaan ainoastaan dokumentin valmisteluun tarvittavat erilliset operaatiot. Tulostusmoduulin käyttäjä on toisen ohjelmistokomponentin ohjelmoija, joka pyytää tulostuspalvelua dokumenttikäsittelijältä. Käyttäjille luodaan erillinen ohjeistus siitä, miten palvelun ominaisuuksia käytetään.

Dokumenttikäsittelijä toteutetaan tulostusoperaatioiden yhdenmukaistamiseksi. Aikaisemmin jokainen ohjelmiston osa-alue on luonut oman tulostusratkaisunsa omiin tarpeisiinsa sopivaksi. Nyt tarkoituksena on toteuttaa tulostuksen valmistelu siten, että mikä tahansa ohjelmiston osa-alue voi ottaa sen tarvittaessa käyttöönsä.

Dokumenttikäsittelijään toteutettavia osa-alueita ovat pdf käsittelijä, tietokantakäsittelijä, XML -käsittelijä sekä REST–rajapintakäsittelijä. Työn keskeisin osio on PDF käsittelijä, jonka toteuttaminen vaatii PDF standardin ja rakenteen hyvää ymmärtämistä.

Dokumenttikäsittelijän avulla kunkin organisaation tekninen asiantuntija syöttää järjestelmään dokumenttipohjat ja suorittaa kenttien parittamisen. Tämä mahdollistaa organisaation omien dokumenttipohjien käytön, jolloin myös organisaation omat logot ja muut organisaatiokohtaiset muutokset pystytään toteuttamaan. Organisaation omia dokumenttipohjia käytettäessä tekninen asiantuntija luo halutun dokumenttipohjan ja syöttää sen järjestelmään.

Dokumenttikäsittelijän toteutus on koettu tärkeäksi asiakasyrityksessä. Mikäli dokumenttikäsittelijää ei toteuteta, ohjelmistokokonaisuudessa käytetään edelleen moninaisia erilaisia tulostustapoja ja dokumenttien yhtenäistäminen saattaa olla hankalaa. Pahimmassa tapauksessa jokainen lomakepohja joudutaan mallintamaan järjestelmään erikseen.

Työn lähdeaineistona käytetään Java EE Dokumentaatiota, PDF dokumentaatiota, Java –kirjastojen dokumentaatiota, XML spesifikaatiota, Apache PDFBox dokumentaatiota, REST–dokumentaatiota, Hibernate –dokumentaatiota sekä työkalukohtaisia dokumentaatioita. Suurin osa dokumentaatioista on sähköisessä muodossa ja varsinkin uusimpien versioiden dokumentaatiot löytyvät ainoastaan sähköisenä.

Ennen varsinaisen työn toteuttamista on päätettävä mitä tiedostoformaattia työssä käytetään, sekä mitä Java -kirjastoa dokumenttipohjan käsittelyssä käytetään. Lisäksi on mietittävä mihin formaattiin tulostettava dokumentti luodaan, ja tuetaanko vapaita formaatteja. On myös ratkaistava, miten tiedoston kentät linkitetään tietokannan kenttiin ja miten linkitystieto tallennetaan tietokantaan. Lopullisessa toteutuksessa kenttien linkittäminen tapahtuu tarkoitusta varten rakennetussa käyttöliittymässä. Käyttöliittymän toteuttaminen ei kuulu annettuun toimeksiantoon.

Lisäksi tulevissa ratkaisuissa tarvitaan yksinkertainen tapa määritellä tulostuksen ulkoasu niin, että järjestelmän ylläpitäjä pystyy sen helposti tekemään. Tämä ulkoasun määrittely koskee tilanteita, joissa valmista lomakepohjaa ei ole käytettävissä. Tulosteessa tietyt kentät on ryhmiteltävä loogisiksi kokonaisuuksiksi esimerkiksi kenttien sijainnin ja kehystyksen avulla. Esimerkki tällaisista tiedoista on henkilön perustiedot, Nimi, osoite, puhelinnumero, jotka halutaan ryhmitellä dokumentissa omaksi tiiviiksi ryhmäkseen.

Toteutuksen vaativin osa liittyy staattisen PDF pohjan muuntamiseen dynaamiseksi. Vaatimuksena on, että lomakepohjan on reagoitava dynaamisesti kenttiin syötettävän tiedon pituuteen. Käytännössä tämä tarkoittaa, että kentän kokoa on kasvatettava, mikäli syötettävä tieto ei siihen mahdu. Dokumentin muiden kenttien sijaintia saatetaan joutua muuttamaan niissä tilanteissa, joissa kentän kokoa joudutaan muuttamaan. Kentän venyttämisen ja muiden kenttien siirtämisen lisäksi dokumenttikäsittelijään rakennetaan toiminto, jonka avulla kenttien siirtäminen sivulta toiselle on mahdollista siten, että kenttien järjestys ja sijainti pysyvät edelleen oikeana.

Toimeksianto perustuu olettamukseen, että dokumentin muokkaamiseen on tarjolla runsaasti erilaisia edullisia tai jopa ilmaisia rajapintakirjastoja. Lisäksi oletetaan, että valittu

dokumenttiformaatti on riittävän joustava, niin että dokumenttien täyttö ja muokkaaminen voidaan toteuttaa yleisluontoisesti, eli että yhdellä ja samalla toteutuksella pystytään käsittelemään kaikkia tarvittavia dokumentteja.

Tulostusmoduuli toteutetaan Java EE ympäristöön. Tietokantakäsittelyssä käytetään Hibernate -rajapintaa, ja web -sovellusten suuntaan rajapinta toteutetaan REST-määrittelyn mukaisena.

Tämä dokumentti koostuu neljästä loogisesta osasta. Ensimmäisessä osassa kuvataan opinnäytetyön toimeksianto ja taustoitetaan työn lähtökohdat, vaatimukset ja tavoitteet. Toisessa osassa käydään läpi toteutuksessa käytetty teoria ja työkalut. Kolmannessa osassa kerrotaan työn toteutuksesta, teknisistä yksityiskohdista ja valituista ratkaisuista. Neljännessä osassa käydään läpi tehdyn työn tulokset, verrataan niitä tavoitteisiin sekä pohditaan jatkokehitysmahdollisuuksia ja tehdään yhteenveto saavutetuista tuloksista.

2 WEB -SOVELLUKSEN PALVELINOSION OHJELMOIMINEN

Merkittävä osa nykypäivän ohjelmistoista toteutetaan selaimella käytettävinä web -sovelluksina. Web -sovellusten etuja ovat niiden riippumattomuus ajasta ja paikasta, helppo ylläpidettävyys, käyttöjärjestelmäriippumattomuus ja saavutettavuus.

Toimeksiannossa toteutettava ohjelmisto perustuu kerroksittaiseen arkkitehtuuriin. Kerroksittainen arkkitehtuuri mahdollistaa joustavien ja uudelleenkäytettävien sovellusten toteuttamisen.

Monikerroksisen arkkitehtuurin tunnusmerkkejä ovat:

- Kerrokset ovat toisistaan erillisiä kokonaisuuksia ja niiden toiminnot ja ominaisuudet eroavat toisistaan
- Mikään kerros yksinään ei muodosta toimivaa kokonaisuutta
- Sovelluksen kaikki kerrokset vaaditaan toimivan kokonaisuuden aikaansaamiseksi.
- Kerrosten välillä tulee olla jokin kommunikointimetodi
- Kukin kerros on oma itsenäinen kokonaisuutensa
- Samat toiminnot sisältävä kerros voidaan korvata toisella samat toiminnot sisältävällä kerroksella, eikä sovelluksen toiminta muutu.

Ylin taso (Front End) toteuttaa käyttöliittymän sekä määrittelee sovelluksen ulkonäön ja asettelun. Keskimmäinen taso huolehtii tiedonsiirrosta tasojen välillä, siirrettävien tietojen prosessoinnista sekä käyttöliittymälle tarjottavien tietojen kokoamisesta. Alin taso (Back End) huolehtii tietokantayhteyksistä sekä muista sovelluksen vaatimista yhteyksistä ja loogisista operaatioista. (What is the 3-Tier Architecture By Tony Marston, 2017)

Web -sovellusten käytetyin arkkitehtuuri on 3 -kerroksinen rakenne, jolloin sovelluksen tarvitsemat toiminnot on erotettu omiksi selkeästi erillisiksi kokonaisuuksiksi. Lisäksi kerrokset ovat tekniikaltaan varsin erilaisia ja kunkin kerroksen toteuttaminen vaatii erilaisia taitoja.

Tässä työssä toteutetaan web -sovelluksen palvelinosion kerrokset; kommunikointiin tarkoitettu välikerros sekä tietokantayhteyksiä ja muita loogisia operaatioita sisältävä alin kerros.

Sovelluksen alimman kerroksen toteutuksessa tärkeimpiä asioita ovat tietokantarajapinnan toteuttaminen sekä PDF tiedostoformaatin käsittelyn toteuttaminen. Kyseessä olevan toteutuksen vaatima välikerros on hyvin suoraviivainen, sisältäen käyttöliittymän tarvitsemat REST-rajapinnan määrittelyt. Välikerros sisältää myös käyttöliittymälle tarjottavien tietojen koostamisen sekä käyttöliittymältä saatavien tietojen valmistelu tietokantaan tallettamista varten.

2.1 PDF

PDF (Portable Document Format) on Adoben kehittämä ja määrittelemä tiedostomuoto, jonka tavoitteena on mahdollistaa dokumenttien helppo, luotettava, ympäristöstä ja näytön resoluutiosta riippumaton siirto ja katselu. PDF tiedostomuotoa on kehitetty vuodesta 1993 lähtien. Vuonna 2007 Adobe Systems Incorporated haki ISO 32000 standardoinnin PDF spesifikaatio versiolle 1.7.

PDF tiedosto koostuu neljästä osasta: datarakenteesta, tiedostorakenteesta, dokumenttirakenteesta sekä lohkomäärittelystä. Datarakenne määrittelee käytetyn merkistön sekä muita syntaksisia elementtejä. Tiedostorakenne määrittelee, miten objektit on tallennettu PDF tiedostoon, miten niitä käsitellään ja päivitetään. Dokumenttirakenne puolestaan määrittelee miten perustyypit kuten sivut ja fontit on esitetty dokumentissa. PDF dokumentin monimutkaisin osa on lohkomäärittely, joka määrittelee miten tekstit, sivut tai muut graafiset elementit näytetään dokumentissa. Lisäksi lohkomäärittely määrittelee joitakin datastruktuureita sekä sisäänrakennettuja perusrakenteita.

PDF lomakkeet, AcroForm

Vuorovaikutteiset lomakkeet ovat joukko kenttiä joiden avulla käyttäjältä kerätään tietoa. PDF standardissa on määritelty kaksi erilaista lomakemuotoa. Alkuperäinen AcroForm, sekä modernimpi ratkaisu XFA. AcroForm -tyyppisissä lomakkeissa lomakkeen kaikki kentät määritellään kenttähakemistossa. Kenttähakemistoon kentät voidaan määritellä hierarkkisesti.

PDF Lomakkeet voivat sisältää seuraavan tyyppisiä kenttiä: Tekstikenttä, valintakenttä, painikekenttä sekä allekirjoituskenttiä.

Tekstikenttä on kenttä, jossa on laatikko tai tila tekstin syöttämiselle. Tekstin pituus voidaan rajoittaa tiettyyn merkkimäärään, yhdelle riville, tai kenttä voidaan määritellä moniriviseksi.

Painikekentät ovat tyypillisesti painonappeja, valintalaatikoita tai radionäppäimiä. Painonapit ovat painikkeita, jotka käynnistävät jonkin toiminnon. Tällaisia toimintoja voivat olla lomakkeen tulostus, tai syötettyjen tietojen tyhjennys. Valintalaatikoilla puolestaan on kaksi tilavaihtoehtoa: ”Päällä” ja ”Pois päältä”. Radionäppäimet koostuvat monesti useasta valintapainikkeesta, joilla kaikilla on tilat ”Päällä” ja ”Pois päältä”. Radionäppäimet rakennetaan usein ryhmiksi, jolloin yhden näppäimen tilan asettaminen ”Päällä” –tilaan, muuttaa ryhmän muiden nappien tilan automaattisesti ”Pois päältä” –tilaan. (Adobe Systems Incorporated 2008, Document mangerment – Portable document format – Part 1: PDF 1.7, 2008.)

PDF lomakkeet, XFA

XFA on PDF lomakkeiden modernimpi toteutus. XFA formaatin on alun perin kehittänyt JetForm Corporation, joka yrityskauppojen seurauksena siirtyi Adoben omistukseen. XFA on lisätty PFD spesifikaatioon versiosta 1.5 lähtien. XFA toteuttaa AcroFormia paremmin lomakkeille nykyisin asetetut vaatimukset, erityisesti lomakkeiden dynaamisuuden osalta. XFA lomakkeita on kahta eri tyyppiä, staattinen sekä dynaaminen. Dynaaminen lomake osaa nimensä mukaisesti mukautua automaattisesti kenttään syötetyn tiedon pituuden mukaan, jolloin kenttien venyttämisestä tai siirrosta sivulta toiselle ei tarvitse huolehtia. Valitettavasti Adobe on luopumassa dynaamisten XFA lomakkeiden tukemisesta, joten tätä formaattia ei voida tässä työssä käyttää. (Adobe XFA specification, 2016.)

PDF Content Stream

PDF dokumentti koostuu kokoelmasta tekstiä, grafiikkaa ja kuvia. Sivun ulkonäön määrittelee PDF Content Stream, joka sisältää graafiset sivulle tulostettavat elementit. PDF Content Stream koostuu lohkoista, joissa on aloitus ja lopetusoperaattorit. Lohkomäärittelyn sisällä ovat varsinaiset piirto-ohjeet koordinaatteineen ja muine määrittelyineen.

```
BT
  /TT0 1 Tf
  7.9966 0 0 7.9966 119.9472 678.449 Tm
  (Sukunimi ja etunimet ) Tj
  24 13.5 Td
  (Todistuksen tarkoitus) Tj
ET
```

Kuva 1: Esimerkki tekstilohkosta

Kuvassa 1 on esimerkki tekstilohkosta. Tekstilohko alkaa operaattorilla BT (Begin Text) ja päättyy operaattoriin ET (End Text). Tekstilohkon sisällä määritellään lohkon ominaisuudet. Tf (Text font) operaattori määrittelee käytettävän fontin, sekä fontin asetukset. Tm (Text Matrix) operaattori määrittelee lohkon sijainnin ja mahdollisen skaalauksen. Tj -operaattori tulostaa sille annetun tekstin asetettuun kohtaan ja Td -operaattori siirtää kohdistimen haluttuun kohtaan.

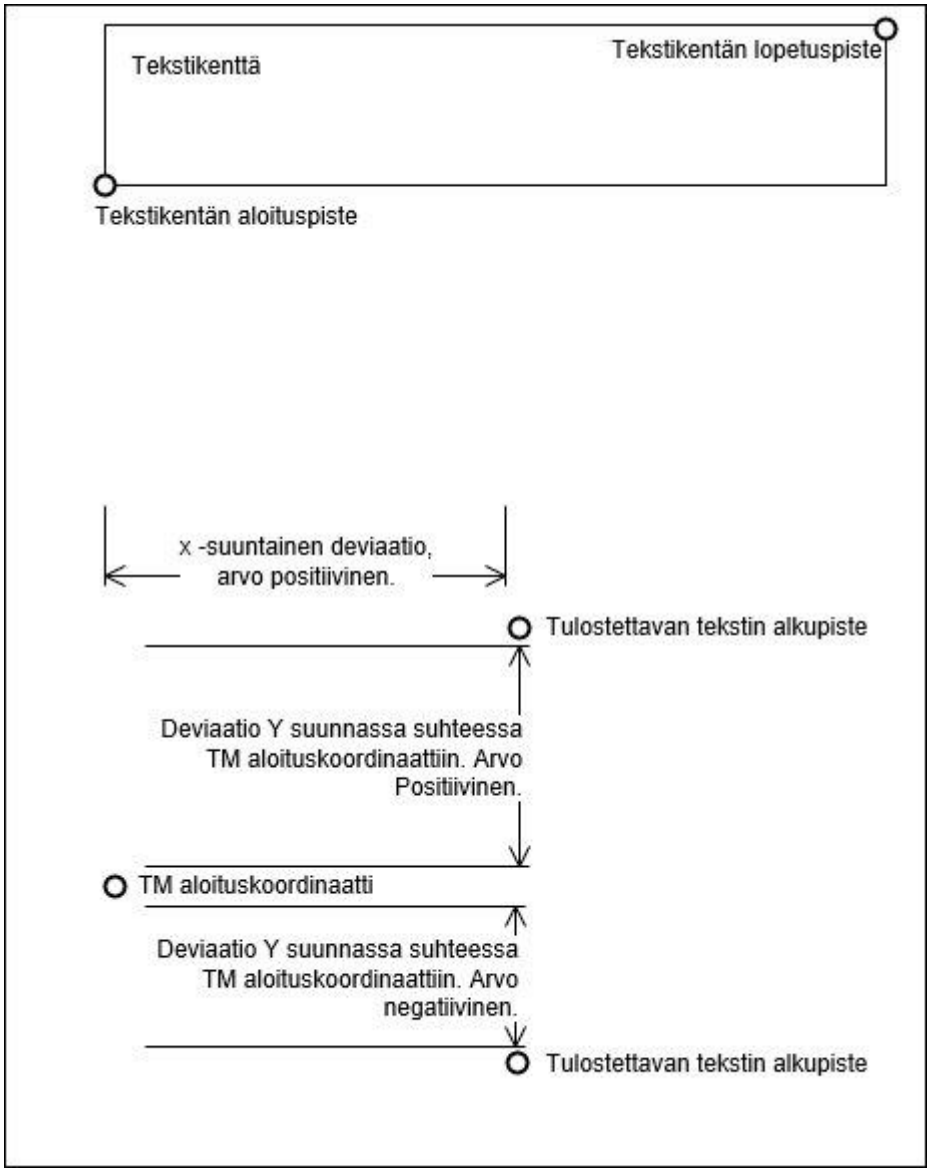
Lohkojen järjestyksellä Content Streamin sisällä ei ole merkitystä, vaan PDF:lle tulostettavien graafisten elementtien sijainti riippuu kunkin lohkon sisällä olevasta sijaintimäärittelystä. Lohkojen järjestys riippuu myös dokumentin luomisessa käytetystä ohjelmasta.

Tekstilohkot saattavat olla hyvinkin suuria, sisältäen runsaasti tekstejä, joiden sijainti suhteessa muihin saman tekstilohkon sisällä oleviin teksteihin saattaa vaihdella hyvinkin paljon. Tekstilohkojen erikoisuus on, että Td ja TD operaattoreiden sijainnin määrittelyt ovat suhteellisia edellisiin määrittelyihin. Niinpä tekstilohkon sisällä olevien operaattoreiden järjestyksen on pysyttävä samana koko käsittelyn ajan.

Tekstilohkot saattavat myös sijaita osittain päällekkäin, siten, että osa toisen tekstilohkon teksteistä sijoittuu toisen tekstilohkon tekstien lomaan.

Text Matrix ja erityyppiset tekstit

Tekstiblokin sisällä Text Matrix (TM) operaattori määrittelee tekstimatriisin alkupisteen. Alkupiste on alueen vasemman alakulman koordinaatti, ja samaan tekstilohkoon kuuluvien tekstien sijainti määritellään suhteessa tähän aloituspisteeseen sekä Y, että X -koordinaatin suuntaan.



Kuva 2: PDF:n tekstimatriisin alkupiste sekä eri tekstilohkojen sijainti dokumentissa.

Kuva 2 esittää yksinkertaistetusti tekstimatriisin alkupisteen sekä tekstikenttien sijainnin dokumentissa. Steam -olion sisällä tekstiblokin alussa määritellään tekstimatriisin aloituskoordinaatti (kuvassa TM aloituskoordinaatti). Tekstimatriisin aloituskoordinaatti ilmoitetaan suhteessa dokumentin aloituskoordinaattiin (0,0). Näytettäville teksteille annetaan koordinaatit, jotka ovat puolestaan suhteellisia tekstimatriisin aloituskoordinaattiin. Yksittäisten tekstien sijainti määritellään td tai TD -operaattoreilla, antamalla alkupisteen koordinaatit. Nämä koordinaatit voivat olla positiivisia tai negatiivisia riippuen siitä, sijaitseeko teksti tekstimatriisin ylä- vai alapuolella.

Tekstikenttien sijainti määritellään eri tavalla. Tekstikenttä on osa AcroFormia ja sille annetaan vasemman alakulman koordinaatti aloituspisteeksi ja oikean yläkulman koordinaatti lopetuspisteeksi. Tekstikentät ovat suhteellisia sivun aloituspisteeseen, joka sijaitsee sivun vasemmassa alakulmassa ja on koordinaatiltaan (0,0).

2.2 REST

REST-termi tulee sanoista Representational State Transfer. REST on arkkitehtuurityyli, joka perustuu koneiden väliseen yksinkertaiseen HTTP kommunikaatioon. REST-rajapintaa käyttävät sovellukset käyttävät HTTP pyyntöjä datan lukemiseen, poistamiseen sekä lähettämiseen.

HTTP -kutsuissa pyynnön identifioiminen tapahtuu kutsun osoitteen mukaan (URI). Jokaiselle kutsulle on oma osoitteensa, ja kutsuun vastataan sille osoitetun toimintalogiikan mukaan. Yleensä kullekin kutsulle toteutetaan oma metodi, johon suoritettavat toimenpiteet määritellään. Koska REST-arkkitehtuuri on tilaton, kutsun tila ilmenee palvelimelle lähetetystä kutsusta.

REST-arkkitehtuuri piilottaa käytetyt resurssit, esimerkiksi tietokantakutsuissa palvelin ei palauta käyttöliittymälle tietokantaa, vaan paluuarvo saattaa olla esimerkiksi HTTP, XML tai JSON -muodossa.

HTTP -kutsuissa voidaan välittää parametreja esimerkiksi tietokantahakuja varten. Nyrkkisääntö on, että lyhyet pyynnöt välitetään GET -tyyppisissä kutsuissa ja pidemmät pyynnöt POST -tyyppisissä pyynnöissä. Tämä on käytännöllistä siksi, että POST -tyyppisissä kutsuissa pidempien parametrien siirtäminen on helpompaa. POST -tyyppisiä kutsuja käytetään tyypillisesti myös välitettäessä palvelimelle tietokantaan talletettavaa dataa. (Learn REST: A Tutorial, 2016.)

2.3 XML

XML kirjainlyhenne tulee sanoista Extensible Markup Language. XML on kieli, jolla määritellään dokumentille täsmällinen tietokoneen sekä ihmisen ymmärtämä rakenne. XML muodostaa siis aina dokumentin. XML dokumentti on jaettu erilaisiin lohkoihin tagimäärittelyjen avulla. Aloitustagille täytyy aina löytyä dokumentista myös lopetustagi. Tagit ovat muotoa <tag></tag>, joka muistuttaa hyvin paljon HTML -formaattia. Ero HTML:n

ja XML:n välillä on se, että HTML:ssä tägien nimet ovat tarkkaan ennalta määriteltyjä, ja määrittelystä poikkeavat tagit jäävät huomiotta. XML:ssä tagien nimet puolestaan määritellään tarpeen mukaan. Tagien nimien täytyy kuitenkin olla yksilöllisiä ja samaa tagin nimeä ei tule käyttää useassa eri merkityksessä. Lisäksi tageilla voi olla hierarkkinen rakenne sen mukaan, miten tietoa on tarve erotella. (Extensive Markup Language (XML W3C recommendation, 2008.)

Tageille voidaan antaa myös attribuutteja. Attribuutit ovat nimi – arvo -pareja ja ne esiintyvät aloitustagissa elementtinimen jälkeen. Esimerkiksi <tag attr=”true”>

Sisällön rakenteen määrittelevien tagien lisäksi dokumentti voi sisältää myös kommentteja, prosessointiohjeita tai dokumentin tyyppimäärittelyjä. (XML.com, A Technical Introduction to XML, 2016.)

2.4 Käytetyt työkalut ja kehitysympäristö

Document Writer -moduulin toteutuksessa käytetään useita työkaluja, joista tärkeimmät ovat Java EE, Hibernate tietokantakäsittelyihin, PDFBox PDF dokumenttien käsittelyyn sekä REST -rajapinta jonka avulla kommunikoidaan nettisovellusten kanssa. Lisäksi XML tiedostojen käsittelyyn tarvitaan Javan XML käsittelyyn tarkoitettu kirjastokokoelma.

Varsinaisten ohjelmisto -osien lisäksi on useita työkaluja, joita työn tekemisessä on tarvittu. Tällaisia ovat ohjelmistokehitysympäristö Eclipse, versionhallinta SVN, sovelluspalvelin Wildfly sekä projektikonfiguraationhallintatyökalu Maven.

2.4.1 Java EE

Java EE on ohjelmointikieli, joka tarjoaa standardiin perustuvan alustan netti- ja yritys-sovellusten kehittämiseen. Nämä sovellukset ovat tyypillisesti monikerroksisia koostuen nettirajapinnasta, tietoturva- ja tiedonsiirtopalveluita tarjoavasta välikerroksesta sekä serveritoteutuksesta, joka tarjoaa tietokanta- tai muut palvelut. (Arun Gupta, Java EE 7 Essentials, 2013.)

Java EE:n alkuperäinen spesifikaatio on Sun Microsystemsin kehittämä ja julkaistu vuonna 2001. Java EE spesifikaatio sisältää useita eri tarkoituksiin soveltuvia rajapintas-

pesifikaatioita ja määrittelee, miten rajapintoja käsitellään. Java EE:n avulla voidaan rakentaa sovelluksia jotka ovat yhteensopivia vanhojen versioiden kanssa ja ovat käytettävissä useilla eri alustoilla. Java EE:llä voidaan rakentaa sekä pieniä sovelluksia, että isoja järjestelmiä.

Java EE yhdistettynä HTML5:een, JavaScriptiin ja CSS :ään mahdollistaa dynaamisten ja interaktiivisten sovellusten luomisen. Sovellukset voivat tarjota käyttäjille reaaliaikaisia interaktiivisia syötteitä, kuten uutisia, pörssikursseja tai muita sovelluskohtaisia ilmoituksia. Lisäksi HTML5:n käyttö helpottaa merkittävästi sovellusten mukauttamista eri kokoisille näytöille.

Java EE:n etuja ovat robustisuus, tuki verkkosovelluksille ja käyttöönoton helppous. Java -kieltä uudistetaan jatkuvasti Javan kehittäjäyhteisöltä saadun palautteen mukaisesti. Uusimmilla Java EE:n versioilla sovellusten kehittäminen on varsin tehokasta ja joustavaa. Lisäksi Java EE tukee avoimen lähdekoodin ohjelmistoympäristöjä. (Oracle Inc, Java EE whitepaper, 2016.)

2.4.2 JUnit

JUnit on yleisesti käytetty testausympäristö, jolla toteutetaan yksikkötestit kullekin ohjelmakoodin toiminnolle. Testit ajetaan erillisinä testiajoina ja ne on mahdollista lisätä myös automaattiseen CI järjestelmään, niin että ne suoritetaan säännöllisin väliajoin, esimerkiksi kerran vuorokaudessa, tai aina kun uutta ohjelmakoodia lisätään koodilinjaan.

JUnit testeillä voidaan testata kaikki ohjelmakoodin suoritustilanteet. Tyypillisiä testattavia tilanteita ovat operaatioiden onnistumiset, epäonnistumiset sekä erilaiset poikkeustilanteet. Lisäksi, ennen testauksen aloittamista voidaan tarvittavat muuttujat ja ohjelmiston osa-alueet asettaa testauksessa käytettäviin arvoihin. Testauksen loputtua testauksessa tarvitut arvot siivotaan pois, niin ettei järjestelmään jää ylimääräistä testauksesta aiheutuvaa roskaa. (JUnit, 2017.)

2.4.3 Apache PDFBox

Apache PDFBox on avoimen lähdekoodin kirjasto PDF muotoisten tiedostojen luomiseen, käsittelyyn ja tulostukseen. PDFBox lisensoidaan Apache lisenssin v2.0. alaisena.

PDFBox on laaja kirjasto, jonka avulla mahdollistetaan PDF dokumenttien käsittely usein eri tavoin. Dokumentteja voidaan lukea, luoda, muuttaa, tulostaa ja tallentaa kirjaston avulla. Se sisältää rajapinnan useiden PDF dokumenttien yhdistämiseen, PDF lomakkeiden täyttämiseen sekä muuntamiseen eri PDF formaatteihin (mm. suojatuiksi tai arkistoitaviksi), tai kuvaksi. PDF dokumenttiin voidaan lisätä XML tai XFA sisältöjä tai allekirjoituksia

Yleisimmille PDF:n käsittelyyn tarvittaville toiminnoille on omat, tiettyyn tarpeeseen tehdyt rajapinnat. Tällaisia ovat PDF dokumentin luominen, lomakkeiden täyttö ja dokumentin tulostus. Sen sijaan dokumentin ulkoasun muokkaaminen täytyy tehdä yleiskäyttöisen stream -rajapinnan avulla. Koska Streamin käsittely tarkoittaa PDF:n ulkoasua määrittävien komentojen muuttamista, vaatii se PDF:n komentorakenteen hyvää ymmärrystä.

2.4.4 JPA ja Hibernate

JPA on spesifikaatio, joka määrittelee, miten dataa käsitellään, tallennetaan ja hallitaan Java objektien, luokkien ja relaatiotietokannan välillä. Näin ollen JPA määrittelee Java kehittäjille menetelmät Java sovellusten sisältämän relaatiotiedon käsittelyä varten. JPA spesifikaatio koostuu neljästä eri osiosta: tietokantarajapinnasta, kyselykielestä, tietokantamäärittelyjen kriteeristöstä sekä tietokantayhteyden metadatasta.

Hibernate on JPA spesifikaation toteutus Java -kielelle. Se on tehokas ja suorituskykyinen oliorelaatioiden tallennus, kysely ja hakupalvelu, jota pystytään hyödyntämään millä tahansa Java -sovelluksella, riippumatta sovelluksen arkkitehtuurista tai toimintaympäristöstä. Hibernate linkittää Java -luokat tietokantatauluiksi sekä Java tietotyypit SQL -tietotyypeiksi vapauttaen kehittäjän suurimmalta osalta yleisimmistä datan tallennukseen liittyvistä ohjelmointitehtävistä. (Java Platform, Enterprise Edition, Java EE Tutorial: Persistence, 2016.)

Hibernate sijoittuu perinteisten Java olioiden sekä tietokantapalvelimen väliin ja käsittelee tiedon tallentamiseen ja hakuihin liittyvät tehtävät. Hibernaten avulla Java luokkien sisältämä tieto voidaan tallentaa tietokantaan ilman ohjelmakoodin kirjoittamista. Hibernate tarjoaa yksinkertaisen rajapinnan Java -olioiden tallentamiselle ja lukemiselle. Tietorakenteiden muuttaminen on yksinkertaista, sillä Hibernaten avulla SQL tietotyypit saadaan piilotettua ja tiedon käsittely tapahtuu Java -olioiden avulla. Hibernate hallitsee monimutkaisetkin tietorakenteet ja minimoi tietokantakäsittelyt älykkäiden hakutoimintojen

ansiosta sekä tarjoaa yksinkertaiset tietokantahaut. Hibernaten käyttö ei vaadi erillistä sovelluspalvelinta. (Hibernate overview, Java persistence framework, 2016.)

Hibernate on LGPL -lisenssin alainen kirjasto, jolloin Hibernaten lähdekoodit ovat vapaasti käytettävissä ja muokattavissa. Hibernatea käyttävän sovelluksen lähdekoodeja ei tarvitse julkaista.(Hibernate community documentation, 2016.)

2.4.5 MySQL ja MariaDB

MySQL on erittäin suosittu avoimen lähdekoodin tietokantapalvelu. Sen suorituskyky, luotettavuus ja helppokäyttöisyys ovat tehneet siitä merkittävän tietokantaratkaisun web -sovelluksille. MySQL:n omistaja on Oracle Incorporation. (MySQL homepage, 2017.)

MariaDB on MySQL:n kehityshaara, jonka omistaa MariaDB Foundation -säätiö. Säätiön tavoitteena on taata tietokantapalvelun jatkuvuus ja avoimuus sekä kehittää tietokantapalvelua edelleen nykyvaatimuksia vastaavaksi. (MariaDB homepage, 2017.)

Yhteisestä historiasta johtuen MySQL ja MariaDB ovat rajapinnoiltaan tällä hetkellä käytännössä identtiset.

2.4.6 Eclipse

Eclipse on laajasti käytetty integroitu graafinen kehitysympäristö, jota käytetään Java ohjelmoinnissa erittäin yleisesti. Eclipse sisältää yleisimmin tarvittavat työkalut sekä ympäristömäärittelyt ja sitä voidaan laajentaa erilaisten laajennusten avulla.

Eclipsen laajennukset sisältävät tyypillisesti erilaisia koodausavustajia, konfiguraatiotyökaluja sekä koodin laadullisia tarkastustyökaluja. Erilaisia laajennuksia Eclipseen on saatavilla erittäin runsaasti. (Eclipse homepage, 2017.)

2.4.7 Wildfly

Wildfly on sovelluspalvelin, joka voidaan integroida ohjelmistokehitysympäristöön. Wildfly toteuttaa Java EE:n spesifikaatiota ja on siksi turvallinen valinta Java -kehittäjille. Wildfly koostuu internetpalvelin -yhteyksistä, tietokantayhteyksistä, ajonaikaisista kirjastoista sekä pääkäyttäjän palveluista.

Sovelluspalvelimen avulla paikallisia sovelluksia tai nettisovelluksia voidaan ajaa paikallisesti pc -ympäristössä. Tämä on erityisen tärkeää ohjelmistokehityksen aikana silloin, kun halutaan testata ohjelmiston toimintaa ennen muutosten siirtoa yrityksen pääkoodilinjaan. Testien avulla sovelluskehittäjä voi varmistaa, etteivät hänen koodimuutoksensa riko muiden kehittäjien ohjelmointiympäristöä. Erityisen hyödyllinen sovelluspalvelin on tilanteissa, joissa halutaan testata tietokantakäsittelyä. Sovelluspalvelimen avulla voidaan luoda todellisen kaltainen palvelinyhteys internetpalvelimen ja tietokantapalvelimen välille. (Wildfly homepage, 2016.)

2.4.8 Maven

Maven on työkalu, jonka avulla voidaan hallita ja kääntää laajoja Java -projekteja. Mavenin tavoitteita ovat käännösprosessin helpottaminen, ohjelmiston käännösprosessin yhtenäistäminen, projektin tietojen kerääminen ja tarjoaminen selväkielisenä, parhaiden käytäntöjen kehittäminen ja ohjeistaminen sekä uusien ominaisuuksien läpinäkyvä hallinta.

Mavenin avulla uudet tarvittavat kirjastot määritellään osaksi järjestelmää, ja Maven osaa automaattisesti hakea kirjaston komponentit oikealta palvelimelta. Samoin Maven osaa aika ajoin tarkastaa kirjaston komponenttiversiot palvelimelta, ja päivittää ne uuteen versioon, mikäli sellainen on tarjolla. (Apache Maven Project, 2017.)

2.4.9 Apache Subversion

Apache Subversion (SVN) on versionhallintajärjestelmä. SVN tukee yleisimmin tarvittavia perusominaisuuksia, kuten hakemistojen, tiedostojen ja linkkien versiointi ja tiedostojen yhdistämisten seuraaminen. Lisäksi SVN tukee ohjelmakoodiversioiden haarauttamista ja haarojen nimeämistä.

SVN on helppokäyttöinen versionhallintajärjestelmä ja sille on saatavilla oma laajennus Eclipseen. (Apache Subversion documentation, 2016.)

2.4.10 Javan XML tuki

Javalle on olemassa monipuoliset kirjastot XML:n käsittelyyn, JAPX (The Java API for XML processing). Tämä kirjasto tarjoaa monipuoliset parserit ja rajapinnat, joiden avulla XML dokumenttia voidaan käsitellä. Kirjaston avulla XML dokumenttien käsitteleminen on hyvin suoraviivaista. (Oracle Incorporated Java Tutorials: Lesson: Introduction to JAXP, 2016.)

2.4.11 Jira

Jira on tehtävänhallintaan, ohjelmistokehityksen seuraamiseen ja julkaisuun tarkoitettu ohjelmisto, jota yleisesti käytetään ketterissä ohjelmistoprojekteissa. Jira on Atlassianin tuote, johon on saatavilla runsaasti erilaisia lisäosia, jotka mahdollistavat toteutuksen katselmoinnin, virheidenhallinnan ja muut ohjelmistoprojektissa tarvittavat toiminnot. (Atlassians Jira software page, 2017.)

2.4.12 Jenkins

Jenkins on työkalu, jonka avulla projektin ohjelmakoodi käännetään ja testataan automaatisesti. Jenkins voidaan konfiguroida suorittamaan ohjelmakoodien käännös aina uusien koodiversioiden julkaisun jälkeen ja ajamaan tietyn testisetin heti käännöksen jälkeen. Jenkins nopeuttaa ohjelmistokehitystä, koska ohjelmakoodien yhteensopivuus ja virheet havaitaan mahdollisimman nopeasti. (Jenkins, build great things at any scale, 2017.)

3 TYÖN TOTEUTUS

3.1 Mediconsult Oy:n asettamat vaatimukset

Työ tehdään Mediconsult Oy:n toimeksiantona ja yritys on toimeksiannossaan määritellyt työn vaatimat toiminnallisuudet.

Järjestelmään on pystyttävä syöttämään lomakepohjia, jotka tallennetaan tietokantaan lomakkeen nimellä. Lomakkeen syöttämisen yhteydessä lomake analysoidaan, ja suoritetaan lomakkeen kenttien linkittäminen sisäisiin tietotyyppeihin.

Lomakkeelle on oltava mahdollista lisätä organisaation logo. On oltava mahdollisuus tallentaa logo tietokantaan, ja haluttaessa logo voidaan lisätä annetulle dokumenttipohjalle.

Lomakkeen täyttämistä aloitettaessa listataan tietokannassa olevat lomakepohjat, joista käyttäjä voi valita tarvitsemansa. Joissain tilanteissa tulostuskomennon yhteydessä dokumenttikäsittelijälle tulee lomakkeen nimi parametrina, jolloin lomakkeen täyttö ja tulostus suoritetaan ennalta määrätyn pohjan perusteella.

Tulostustilanteita on kaksi. Tilanne, jossa käsitellään valmista lomakepohjaa, eli täytetään tiedot valmiiseen PDF dokumenttiin. Toisessa tilanteessa on olemassa ainoastaan kenttien ryhmittelyn sekä muotoilun määrittely, ja tulostettava PDF luodaan ajonaikaisesti. Tällöin tietokentät on eroteltava loogisesti omiksi ryhmikseen. On myös mahdollista, että joissain tilanteissa osa dokumentista on määritelty PDF pohjan avulla, ja osa dokumentista luodaan ajonaikaisesti.

Järjestelmän on otettava huomioon myös tilanne, jossa lomakepohjassa oleva tila on riittämätön kenttään syötettävälle tiedolle. Näissä tilanteissa on kaksi ratkaisuvaihtoehtoa, joista molempia on tuettava. Ensimmäinen ja käytettävyyden kannalta parempi ratkaisuvaihtoehto on, että kenttää laajennetaan niin, että tieto mahtuu. Tällöin lomakepohjan muita kenttiä siirretään vastaavasti eteenpäin, tai seuraavalle sivulle mikäli alkuperäisen sivun alamarginaali tulee vastaan.

Toinen ratkaisuvaihtoehto on, että tietokentän sisältö siirretään uudelle sivulle dokumentin jälkeen, ja alkuperäiseen tietokenttään lisätään viittaus dokumentin loppuun.

Ohjelmistomoduuliin rakennetaan REST-rajapinta lomakkeiden syöttämistä, hakua, linkitystä ja tulostuskomentoja varten.

Ohjelmamoduulin toteutuksesta luodaan suunnitelma Jira -työkaluun ja aina yhden osa-alueen valmistuessa se katselmoidaan yrityksen muiden työntekijöiden toimesta.

3.2 Kehitysympäristö

Ohjelmointikielenä käytetään Java EE versiota 8. Ohjelmointiympäristöön kuuluvat integroitu kehitysympäristö Eclipse, SVN versionhallintatyökalu, Maven konfiguraatiohallintatyökalu sekä Jenkins integrointiympäristö. Tehtävälistan ja työskentelyjärjestyksen suunnittelu tehdään Jira -tehtävienhallintatyökalussa ketterän kehityksen prosessimäärittelyn mukaisesti.

Työ tehdään Mediconsult Oy:n ohjelmistokehityskäytäntöjen mukaisesti. Tärkein noudatettava periaate on ns Clean code -periaate, joka määrittelee ohjelmakoodin laadullisia seikkoja kuten koodin kompleksisuuteen, rakenteeseen ja koodin ulkonäköön liittyviä seikkoja. Clean Code -periaatteen tarkoituksena on varmistaa koodin luettavuus ja ylläpidettävyys. (Clean code cheat sheet, 2016.)

3.3 Käytetyt työmenetelmät

Varsinainen toteutus tehdään yrityksen kehitysympäristöön. Koodin kirjoittaminen aloitetaan luomalla ensin JUnit testi kyseiden metodin testaamista varten, jonka jälkeen kirjoitetaan varsinainen koodi, joka toteuttaa alkuperäisen JUnit testin. Metodin valmistuttua luodaan useampi JUnit -testi, jolla varmistetaan metodin toiminta myös erilaisissa virhetilanteissa.

Koska tulostusmoduuli toteutetaan palveluna, muu ohjelmistoympäristö ei aseta toteutukselle juurikaan rajoituksia. Välitettävien tietojen formaatti, komponenttien välinen kommunikointi ja muut rajapinnat ovat täysin tulostusmoduulin päätettävissä.

Työn tekeminen aloitetaan analysoimalla annettu tehtävänanto. Toiminnan mukaisiin osa-alueisiin jaetuista työtehtävistä piirretään BPMN (Business Process Model and Notation) -kaaviot, jotka käydään läpi työn ohjaajien kanssa. Lisäksi suunnitellaan alustava tietokantarakenne. Osa-alueiden läpikäynnin yhteydessä vaatimus asiakkaan logon lisäämisestä lomakkeelle voidaan poistaa, koska asiakkaalla on joka tapauksessa mahdollisuus syöttää järjestelmään omalla logollaan varustettu PDF -muotoinen lomakepohja.

3.4 Dokumenttien käsittelyyn käytettävän kirjaston ja dokumenttiformaatin valinta

Ennen varsinaisen implementointityön aloitusta on valittava työssä käytettävä Java -kirjasto, sekä tiedostoformaatti jolla tulostus suoritetaan. Tiedostomuodon valintaan vaikuttaa saatavilla olevien kirjastojen laatu sekä hinta. Myös tulostusformaatin tuen kattavuus vaikuttaa valittavaan tiedostoformaattiin. Tutkittavia tiedostoformaatteja ovat PDF sekä ODF. ODF on houkutteleva vaihtoehto sen ilmaisuuden ja avoimuuden vuoksi.

Tiedostojen käsittely on varsinkin yritysmaailmassa aihepiiri, jossa on paljon tarpeita, ja myöskin paljon tilaisuuksia tiedostojen käsittelyyn erikoistuneille yrityksille. Kartoitettaessa erilaisia java -kirjastoja PDF sekä ODF formaateille, käy ilmi, että alalla on paljon toimijoita, ja hinnoittelu on sen mukaista. Ainoa varteenotettava ilmainen ratkaisu valintahetkellä on Apache Commons -lisenssin alainen PDFBox -Java-kirjasto. Apachella on rakenteilla myös ODF -formaatin käsittelyyn tarkoitettu ODFDom -niminen Java-kirjasto, mutta valintahetkellä tämä kirjasto on edelleen kehitysvaiheessa, eikä tuotantokäyttöön sopivaa versiota ole olemassa.

Vertailtaessa PDFBoxin kustannuksia ja ominaisuuksia muihin vaihtoehtoihin, selkeimmäksi tekijäksi nousee PDFBoxin ilmaisuus suhteessa kilpaileviin tuotteisiin. Kaupalliset tuotteet toki tarjoavat samat, tai ehkä jopa laajemmat ominaisuudet, mutta varsin kalliilla hinnalla. Ainoa edullinen vaihtoehto PDFBoxille on iText, joka puolestaan on lisensoitu AGPL -lisenssillä, joka edellyttää luodun ohjelmakoodin jakamista avoimen lähdekoodin yhteisölle. Kehitettäessä kaupallista ratkaisua, tämä ei kuitenkaan ole mahdollista.

3.5 Toteutuksen eri osiot

Lopullinen toteutus koostuu useasta eri osa-alueesta:

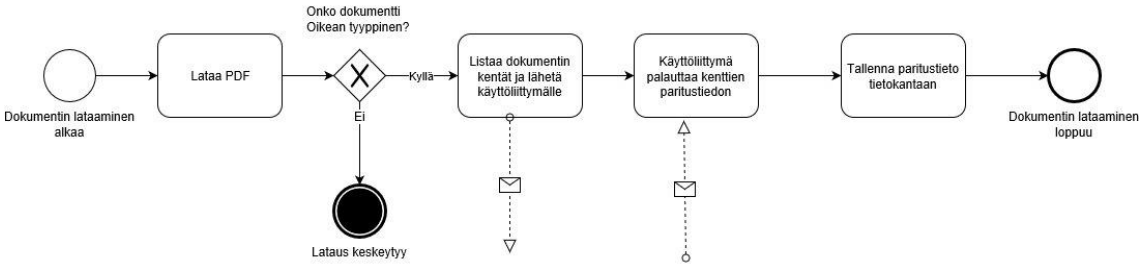
- Tietokantakäsittelystä joka toteutetaan Hibernate tietokantayhteydellä
- PDF:n käsittely PDFBox kirjaston metodeiden avulla,
- PDF:n dynaamisesta ajonaikaisesta muotoilusta sekä
- Tietojen bufferoinnista Dto:n avulla REST-rajapinnan saataville.

3.6 Dokumenttikäsittelijän toiminta eri tilanteissa

Dokumenttikäsittelijällä on kolme pääasiallista toimintoa. Dokumenttipohjan lataaminen järjestelmään, dokumentin täyttö sekä tekstitilan laajentaminen. Näistä viimeinen, tekstitilan laajentaminen käynnistyy dokumentin täyttö -toiminnon kutsumana, mikäli käyttäjän antama tieto ei mahdu tiedolle varattuun kenttään.

Dokumenttipohjan lataaminen järjestelmään

Dokumenttipohjan lataaminen suoritetaan tilanteissa, joissa järjestelmään ladataan valmiita dokumenttipohjia, tai asiakas haluaa ladata oman dokumenttipohjansa järjestelmään. Omien dokumenttipohjien käytöllä mahdollistetaan yrityksen tai organisaation logojen lisääminen tai muiden muutosten teko valmiisiin dokumenttipohjiin.



Kuva 3: Dokumenttipohjan lataaminen järjestelmään

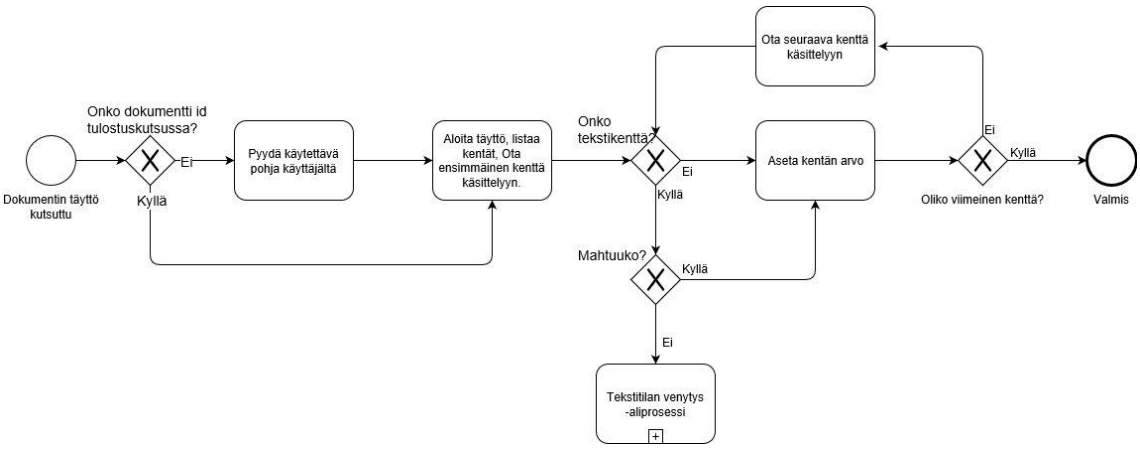
Dokumenttipohjan lataaminen järjestelmään (kuva 3) käynnistetään valitsemalla kyseinen toiminto käyttöliittymästä. Käyttäjä antaa ladattavan PDF dokumentin ja dokumenttikäsittelijä tarkastaa sen. Mikäli dokumentti ei ole oikeanlainen, esimerkiksi jos dokumentista ei löydy AcroFormia tai AcroFormissa ei ole yhtäkään kenttää, palautetaan virhe, ja toiminto päättyy. Mikäli dokumenttipohja täyttää asetetut vaatimukset, dokumenttikäsittelijä listaa löytämänsä AcroForm -kentät ja lähettää ne käyttöliittymälle linkittämistä varten.

Käyttöliittymässä käyttäjälle tarjotaan lista löydetyistä kentistä sekä tiedot jotka kenttiin voidaan asettaa. Käyttäjän linkitettyä kentät, käyttöliittymä palauttaa dokumenttikäsittelijälle saadut kenttäparit ja dokumenttikäsittelijä tallentaa ne tietokantaan. Yllä olevassa kuvassa prosessi on yksinkertaistettu selkeyden ja luettavuuden vuoksi. Esimerkiksi

kentän paritustietojen kirjoittamisen yhteydessä saattaa ilmetä tietokantavirheitä, ja näiden käsittelyä ei kuvassa ole huomioitu.

Dokumentin täyttö

Dokumentin täyttö -operaatiossa tietokantaan talletettu dokumenttipohja täytetään käyttöliittymältä saaduilla tiedoilla.



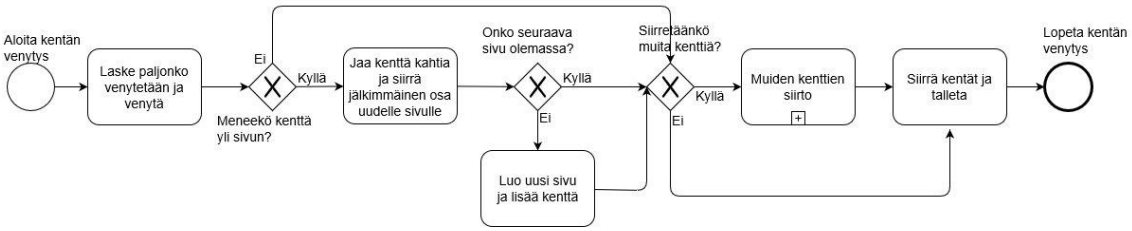
Kuva 4: Dokumentin täyttö

Dokumentin täyttö (Kuva 4) aloitetaan, kun käyttöliittymältä tulee tulostuskutsu Dokumenttikäsittelijälle. Tulostuskutsu pääsääntöisesti sisältää tiedon, mitä dokumenttipohjaa ollaan käyttämässä. Mikäli tätä tietoa ei tulostuskutsussa tule, kysytään se erillisellä dialogilla käyttäjältä.

Dokumentin täyttö aloitetaan lataamalla dokumenttipohja sekä kenttien paritustiedot tietokannasta. Tämän jälkeen käyttöliittymän antamat tiedot sijoitetaan määriteltyihin kenttiin. Tekstikenttien kohdalla tarkastetaan, mikäli annettu tieto mahtuu tiedolle varattuun tilaan dokumenttipohjassa. Mikäli tieto ei mahdu, käynnistetään kentän venytys -aliprosessi. Valintanappi -tyyppisten kenttien kohdalla tarkistusta ei tarvitse suorittaa, koska kenttiin kirjoitettava tieto on ennalta määritelty päälle/pois -tieto.

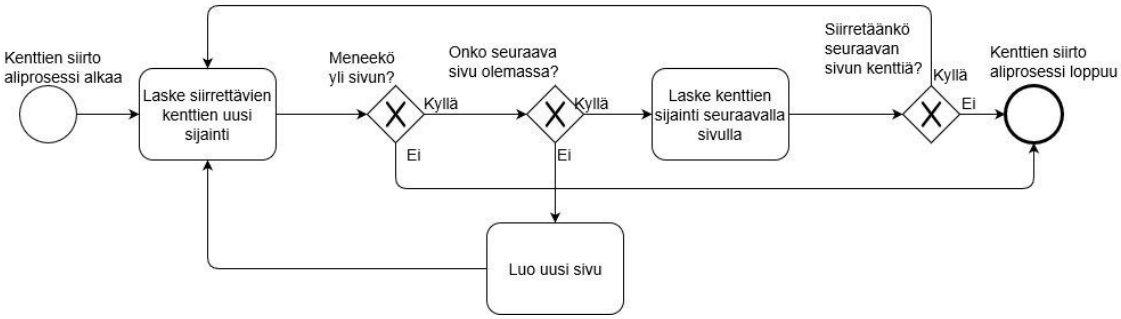
Tekstitilan laajentaminen

Tekstitilan laajentaminen (Kuva 5) koostuu loogisesti kahdesta eri toiminnosta. Yhden kentän laajentamisesta sekä laajennettua kenttää seuraavien kenttien siirrosta eteenpäin dokumentissa ja sivuilta toiselle. PDF Stream -rakenne määrittelee kenttien sijainnit sivu kerrallaan ja siksi myös kenttien siirto tehdään tutkimalla uudet sijainnit sivu kerrallaan.



Kuva 5: Tekstitilan laajentaminen

Yksittäisen kentän venytys aloitetaan laskemalla, paljonko kenttää täytyy venyttää. Tämän jälkeen tarkastetaan, mahtuuko kenttä nykyiselle sivulle, vai siirretäänkö osa kentästä seuraavalle sivulle. Mikäli osa kentästä siirretään seuraavalle sivulle, tutkitaan, onko seuraava sivu olemassa, vai luodaanko uusi sivu kenttää varten. Kun venytettävän kentän käsittely on saatu valmiiksi, tutkitaan, onko kentän jälkeen kenttiä, jotka täytyy myös siirtää. Mikäli muita kenttiä joudutaan siirtämään, käynnistetään muiden kenttien siirto -aliprosessi (Kuva 6).



Kuva 6: Muiden kenttien siirto -aliprosessi

Kentän laajentamisen jälkeen tarkastetaan, onko muiden kenttien siirto tarpeellista. Mikäli kenttiä siirretään, lasketaan siirrettäville kentille uudet sijainnit ja tämän jälkeen tarkastetaan, meneekö jokin kenttä yli sivun alamarginaalista. Mikäli on tarpeen siirtää kenttä tai kenttiä seuraavalle sivulle, ensin tarkastetaan, onko seuraava sivu olemassa.

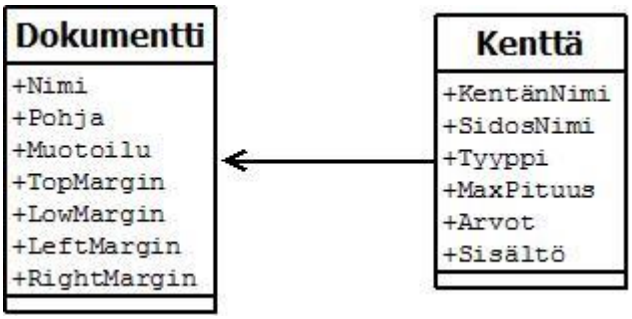
Mikäli ei ole, luodaan uusi sivu, lasketaan kenttien sijainnit uudella sivulla ja päätetään prosessi. Mikäli seuraava sivu on olemassa, siirretään kentät seuraavalle sivulle, siirretään seuraavan sivun kentät, ja tarkastetaan, meneekö jokin kenttä yli alamarginaalista. Tätä prosessia toistetaan niin kauan, kunnes kentät mahtuvat sivulle ja prosessi voidaan päättää.

Kenttien siirtojen päätyttyä kenttien uudet sijainnit, sekä niihin annetut tiedot tallennetaan uudeksi PDF dokumentiksi, joka on valmis tulostusta varten.

3.7 Tietokantayhteys

Hibernate -tietokanta määritellään luomalla luokka, jonka jäsenmuuttujat muodostavat tietokantataulun kentät. Kentät ja tietokannan käyttäytyminen määritellään @ -alkuisten annotaatioiden avulla.

Dokumenttikäsittelijä -moduulissa tarvitaan kaksi taulua (Kuva 7), joista toiseen tulee dokumentin tiedot: Dokumentin nimi, id ja muut perustiedot sekä lista joka sisältää dokumentin sisältämät kentät.



Kuva 7: Kuvaus tietokantataulujen rakenteesta.

Tietokantateknisesti kenttälista muodostaa uuden taulun, joka linkittyy dokumentiID:hen. Hibernate rakentaa taulujen välisen yhteyden automaattisesti, kun Java -luokat on määritelty niin, että toinen tietokantatauluksi määritelty luokka, tässä tapauksessa dokumentti, sisältää listan, jonka solut koostuvat toisesta Java -luokasta (kentät).

Tietokantarakenteen määrittelyn lisäksi määritellään kullekin kentälle Set- ja Get -metodit. Lisäksi tarvitaan SQL -muotoiset hakulausekkeet, joiden avulla tietokantahaut suoritetaan.

3.8 PDF:n käsittely PDFBox -kirjaston metodien avulla

PDFBox -kirjasto sisältää toteutukset PDF:n monipuoliselle käsittelylle. Työssä käytetään PDFBoxin AcroFormin käsittelyyn tarkoitettuja metodeita. Näiden metodeiden avulla PDF:stä saadaan luettua lista sen sisältämistä lomakekentistä, kenttien tyypit, sijainnin sekä useita muita tietoja. Kenttien sisällön asettaminen suoritetaan niin ikään AcroFormin käsittelyyn tarkoitettujen metodeiden avulla.

3.9 Staattisesta PDF:stä dynaaminen

PDF on perusformaattina staattinen, vaikkakin myöhemmin siihen on lisätty myös dynaamisia ominaisuuksia XFA formimäärittelyiden muodossa. Koska kenttien venyttämiseen tai kenttien siirtämiseen ei ole olemassa valmista ratkaisua PDFBox -kirjastossa, toteutetaan tämä ominaisuus muuttamalla PDF:n asettelua ohjelmallisesti syötettävien tietojen mukaan.

PDF:n asettelu on määritelty oliossa, jota kutsutaan nimellä Content Stream. Content Stream on yksinkertainen merkkijono sisältäen listan parametreja ja komentoja. Rivin alussa listataan ensin komennolle annettavat parametrit, ja viimeisenä suoritettava komento.

PDF:n staattisesta luonteesta johtuen Content Streamin elementtejä ei ole järjestetty Streamin tai tekstilohkon sisällä dokumentin mukaiseen loogiseen järjestykseen, vaan dokumentin luonnissa käytetty ohjelmisto määrittää sen mihin järjestykseen elementit tulevat. Niinpä Stream -olioiden käsittely vaatii varsin monimutkaista logiikkaa, jossa ensin on laskettava elementin sijainti ja joudutaanko elementtiä siirtämään toisten elementtien liikkumisen tai venymisen vuoksi. Seuraavaksi etsitään dokumentista elementin ympärillä olevat muut elementit ja lasketaan myös niiden uusi sijainti siirron jälkeen.

Elementin paikan laskentaa monimutkaistaa elementtien sijaintien suhteellisuus tekstilohkojen aloituspisteisiin, jolloin jälkimmäiseen tekstilohkoon kuuluvan elementin sijainti saatetaan määritellä ensimmäisen tekstilohkon elementtien keskelle.

Mikäli elementti siirtojen vaikutuksesta siirtyy seuraavalle sivulle, luodaan elementille uusi tekstilohko elementin uuden sijainnin mukaisesti.

Esimerkiksi komento Tm määrittelee tekstimatriisin koordinaatit, josta kyseisen tekstilohkon tietoa aletaan kirjoittaa. Osa koordinaateista on suhteellisia, eli niiden sijainti dokumentissa riippuu kyseisen tekstilohkon matriisin sijainnin alkupisteistä. Tähän tarkoitukseen rakennetaan monitasoinen algoritmi, joka tutkii kunkin tekstilohkon sijainnin dokumentissa suhteessa venytettyyn kenttään.

PDF lomakkeen tietojen kerääminen

Kun järjestelmään syötetään uusi dokumentti, se analysoidaan ja käyttäjää pyydetään määrittelemään taulukko, joka yhdistää dokumentin kentät vastaaviin käyttöliittymän kenttiin. Tämä taulukko on tulostuksessa ohjelmiston ohjenuora, jonka avulla luodaan tulostusmoduulille tarvittava taulukko joka sisältää kentän nimen ja kenttään syötettävän arvon.

Dokumentin sisältämien kenttien tiedot luetaan AcroForm -listasta, ja ne voidaan tulostaa näytölle tai kirjoittaa dokumentin kenttiin. Kenttiin tulostaminen on tarpeellista erityisesti tapauksissa joissa kenttien nimeäminen ei ole looginen sen suhteen missä järjestyksessä kentät sijaitsevat dokumentissa.

CheckBox -tyyppiset kentät ovat poikkeuksellisia, koska niiden lailliset arvot on määritelty täsmällisesti. CheckBox -tyyppisistä kentistä tulostetaan lista laillisista arvoista, jolloin käyttöliittymän on huolehdittava, ettei kenttään syötetä laittomia arvoja. Mikäli kenttään syötetään laiton arvo, aiheuttaa tämä poikkeuksen, jonka käsittelystä on huolehdittava asianmukaisesti.

PDF lomakkeiden täyttö

PDF lomake täytetään tulostusmoduulille annettavan HashMap -taulun avulla. Käyttöliittymä sijoittaa käyttöliittymässä olevat arvot kentän nimien mukaan vastaaviin kohtiin HashMapissa. Näin tulostusmoduuli voi kirjoittaa annetut tiedot suoraan dokumenttipohjaan huolehtimatta annettujen arvojen oikeellisuudesta.

CheckBox -tyyppisiin kenttiin kirjoitetaan annetut arvot, jolloin PDFBOX -kirjaston logiikka huolehtii rastin sijoittamisesta oikeaan ruutuun.

Mikäli tekstikenttään syötettävä arvo ei ylitä kenttään mahtuvien merkkien määrää, selvitään yksinkertaisella tiedon kirjoittamisella, mutta mikäli tiedon pituus ylittää kenttään mahtuvien tietojen pituuden, suoritetaan kentän venyttäminen, joka saattaa muuttaa koko loppudokumentin kenttien sijainteja. Kenttien venytyksen toteutus on kuvattu erillisessä osiossa: PDF lomakkeiden muokkaaminen.

PDF lomakkeiden muokkaaminen

Koska PDF on staattinen formaatti, joudutaan PDF purkamaan osiin, muuttamaan venytetyn kentän pituutta ja muuttamaan kentän jälkeisten kenttien ja muiden dokumentin yksiköiden sijainteja venytyksen suhteessa. Käytännössä tämä tehdään purkamalla dokumentti osiin, muokkaamalla kenttien sijainteja ja kokoamalla dokumentti uudelleen.

PDF dokumentti koostuu erillisistä sisältöobjekteista, joiden käsittely eroaa toisistaan merkittävästi. AcroForm objekti sisältää kenttämäärittelyt, ja rajapintametodit joiden avulla kenttien sijaintia voidaan muuttaa varsin suoraviivaisesti. Stream sen sijaan koostuu PDF:n muotoiluun vaikuttavista komennoista ja niiden parametreista. Streamin käsittelyyn ei PDFBOX -kirjastossa ole suoraa rajapintametodia, vaan Stream puretaan komentoihin, joille annettavia arvoja muokataan kenttien sijainnin mukaan. Streamin käsittelyn tekee erityiseksi myös se, että yksittäiset komennot ovat riippuvaisia paitsi omista parametreistaan, myös edeltävien komentojen parametreista.

3.10 XML ja vapaasti tulostettavat kentät

Osa dokumenteista tulostetaan ilman valmista PDF pohjaa, tai osittaisen PDF -pohjan avulla. Näissä tilanteissa dokumentin muotoilu määritellään järjestelmään syötettävän XML -tiedoston avulla. XML -tiedostossa määritellään kenttien tulostusjärjestys ja ryhmittely, mahdolliset erottimet ja kehykset sekä PDF -pohjan sijainti, mikäli sitä kyseisten tietojen yhteydessä käytetään. Lisäksi XML tiedostossa määritellään kenttien otsikkotiedot sekä tarvittavat ohje, ja aputekstit.

Yksi dokumenttikäsittelijän vaatimuksista on tietojen tulostaminen ilman valmista PDF -pohjaa. Tämä on suunniteltu toteutettavaksi siten, että käyttäjä luo XML -tiedoston joka määrittelee tulostuksen rakenteen. XML -tiedostossa määritellään riville tulostettavien kenttien järjestys ja erikoistehosteet, kuten viivat, lihavoinnit ja kenttien kehystykset.

Kenttien arvojen lisäksi XML-tiedosto sisältää lomakkeen kenttien nimet ja selitetekstit sekä muut mahdolliset muotoilut. Kuten valintakenttien laatikot tai painonapit.

Documenttikäsittelijä lukee tulostusohjeet XML -tiedostosta ja luo sen mukaisen PDF dokumentin.

3.11 Dto ja REST rajapinnan toteuttaminen

DTO (Data Transfer Object) toimii tietokannan ja REST rajapinnan välissä suorittaen HTML -kutsujen tarvitsemat haut ja määritellen miten isoissa osajoukoissa hakujen tulokset annetaan käyttöliittymälle. DTO:n avulla voidaan säädellä ohjelman tarvitseman muistin suuruutta.

REST rajapintaan puolestaan määritellään käytettävissä olevat HTML -kutsut. REST-määrittely vastaanottaa määrittelyä vastaavan HTML -kutsun ja välittää pyynnön Dto:n kautta tietokannalle joka puolestaan palauttaa kutsun tarvitsemat tiedot.

REST rajapinta tarjoaa rajapinnan sekä POST, että GET -tyyppisille pyynnöille. POST -pyynnön avulla voidaan lisätä tietokantaan käyttäjän käyttöliittymälle antamia tietoja. GET -puolestaan suorittaa hakuja tietokannasta.

3.12 Ohjelmistomoduuli osana muuta sovellusta

Dokumenttikäsittelijä sijoittuu osaksi palvelinohjelmistoa. Kommunikointi web –sovelluksen suuntaan tapahtuu REST–rajapinnan avulla. Moduuli rakennetaan palvelemaan useita erilaisia tarpeita, joten ohjelmistokehitys ja suunnittelu on jaksotettava erilaisten vaatimusten kiireellisyyden mukaan.

Ensimmäisessä vaiheessa rakennetaan tietokantayhteydet ja tietokantakirjoitusten ja lukujen testaukseen käytettävät testit. Koska kaikkein kriittisin toiminto on PDF –asiakirjojen täyttäminen, työstetään tätä ominaisuutta samalla kun luodaan muidenkin ominaisuuksien vaatimia järjestelmiä. PDF asiakirjan käsittelyä tukevia muita järjestelmiä ovat REST–rajapinnan rakentaminen sekä XML –tiedoston käsittely.

Seuraavana toteutetaan vapaasti tulostettavien asiakirjojen muotoilun vaatimat ominaisuudet. Näitä ovat XML –tiedoston attribuuttien määrittely ja käsittely.

3.13 Tiedostojen tulostaminen

Dokumenttikäsittelijän toiminnot keskittyvät PDF dokumenttien luomiseen, täyttämiseen ja käsittelyyn. Niinpä varsinainen PDF dokumentin tulostus voidaan suorittaa esimerkiksi avaamalla dokumentti esikatseluun Adobe Readeriin ja käyttäjä voi suorittaa tulostuksen Adobe Reader -ohjelmiston avulla.

4 TULOSTEN ARVIOINTI JA MODUULIN JATKOKEHITYS

Toimeksiannon tavoitteena oli luoda ohjelmistokomponentti, joka tarjosi lomakekäsittelypalvelun ohjelmiston kaikille osa-alueille. Ohjelmistokomponentin keskeisin tehtävä oli analysoida PDF lomakkeen kentät, yhdistää ne tietokannan kenttiin ja tallentaa näin saatu tulostusohje tietokantaan myöhempää käyttöä varten. Lisäksi oli määriteltävä mitä ohjelmisto tekee siinä tapauksessa, että annettu tieto ei mahdu lomakkeessa sille varattuun tilaan. Toimeksianto ei määritellyt käytettävää tiedostoformaattia, tai Java -kirjastoa. Näin ollen ennen varsinaisen työn toteutusta oli vertailtava eri lomakeformaatteja ja valittava niistä tarkoitukseen parhaiten sopiva formaatti.

Työn tuloksena oli palvelu, joka käsitteli PDF lomaketiedostoja PDFBox -nimisen Java -kirjaston avulla. Toteutettu palvelu loi tulostusohjeen annetulle lomakkeelle ja tallensi sen tarkoitusta varten määriteltyyn tietokantarakenteeseen. Lisäksi toteutettiin käsittely tilanteille, jolloin annettu tieto ei mahtunut lomakkeella sille varattuun tilaan. Lopuksi luotiin dokumentaatio joka toimi myös käyttöohjeena palvelua käyttäville ohjelmisto-osa-alueille.

Työn tuloksista käy ilmi, että PDF lomakkeiden käsittely PDFBox -kirjaston valmiiden rajapintojen avulla on varsin suoraviivaista. Lisäksi uusien kenttien ja sivujen luominen onnistuu kirjaston avulla helposti. Sen sijaan lomakepohjan valmiiden kenttien venyttäminen ja tietojen siirtäminen sivulta toiselle, ei sisältynyt PDFBox-kirjaston ennalta tuettuihin toimintoihin ja sille jouduttiin luomaan kokonaan uusi toteutus PDFBox -kirjaston perustoiminnallisuuksien avulla. Työ osoitti, että tällainen toteutus on mahdollista tehdä, mutta on toteutuksena varsin työläs. Lisäksi erillisen yhtä tarkoitusta varten tehdyn toteutuksen luotettavuus ei ole yhtä hyvä kuin runsaasti käytetyn yleiskäyttöisen kirjaston.

Toteutuksen avulla lomakkeiden täyttäminen on helppoa ja nopeaa. Lisäksi järjestelmän ylläpitoon kuluva aika vähenee merkittävästi, kun jokaista lomaketta ei jouduta mallintamaan järjestelmään erikseen, vaan kenttien linkittäminen toisiinsa riittää. Koska valmiita lomakepohjia käytetään varsin runsaasti, toteutetun kaltainen lomakkeiden venyttäminen on varmasti yksi merkittävä käyttömukavuutta lisäävä seikka, ja saattaa siksi olla yksi ratkaiseva tekijä ohjelmistojen välisessä tiukassa kilpailussa.

Työn tuloksia voidaan soveltaa kaikissa PDF -lomakkeita käsittelevissä Java -ohjelmistoissa. Ohjelmistoteknisesti loogisin paikka PDF lomakkeiden dynaamisuuden toteuttamiselle on joko oma itsenäinen Java -kirjastonsa tai osana isompaa Java -kirjastoa. Tällöin toiminnallisuus voitaisiin pyytää valmiin rajapinnan kautta, ja sovelluksen toteuttaminen olisi nopeaa ja helppoa. Mikäli jossain vaiheessa vastaavanlainen toiminto tulee osaksi valmista Java -kirjastoa, lienee aiheellista harkita kannattaisiko käytettävä Java -kirjasto vaihtaa. Peruste vaihdolle on yleisesti käytetyn kirjaston parempi luotettavuus ja hyvä testikattavuus.

LÄHTEET

Adobe Systems Incorporated 2008, Document management - Portable document format - Part 1: PDF 1.7. Viitattu 22.8.2016. http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/PDF32000_2008.pdf

Adobe XFA specification. Viitattu 17.09.2016. http://partners.adobe.com/public/developer/en/xml/xfa_spec_3_3.pdf

Apache Maven Project. Viitattu 30.4.2017. https://maven.apache.org/

Apache Subversion documentation. Viitattu 08.12.2016. https://subversion.apache.org/docs/

Atlassians's Jira software page. Viitattu 25.02.2017. https://www.atlassian.com/software/jira

Clean code cheat sheet. Viitattu 10.12.2016. http://www.planetgeek.ch/wp-content/uploads/2013/06/Clean-Code-V2.1.pdf

Eclipse homepage. Viitattu 30.4.2017. https://eclipse.org/

Extensible Markup Language (XML) W3C recommendation 26 November 2008. Viitattu 08.12.2016. https://www.w3.org/TR/REC-xml/

Hibernate community documentation. Viitattu 27.11.2016. https://docs.jboss.org/hibernate/stable/annotations/reference/en/html/ch01.html.

Hibernate Overview, Java Persistence framework. Viitattu 12.11.2016 https://www.tutorialspoint.com/hibernate/hibernate_overview.htm

Java EE 7 Essentials, Arun Gupta, 2013

Java Platform, Enterprise Edition: Java EE Tutorial: Persistence. Oracle. Viitattu 27.11.2016. https://docs.oracle.com/javaee/7/tutorial/partpersist.htm

Jenkins, build great things at any scale. 25.02.2017. https://jenkins.io/

JUnit Homepage. Viitattu 30.4.2017. http://junit.org/junit4/

Learn REST: A Tutorial, Elkstein. Viitattu 15.09.2016. http://rest.elkstein.org/

Maria DB homepage. Viitattu 30.4.2017. https://mariadb.org/

MySQL homepage. Viitattu 30.04.2017. https://www.mysql.com/

Oracle Incorporated. Java EE whitepaper. Viitattu 5.12.2016 http://www.oracle.com/technetwork/java/javaee/javaee7-whitepaper-1956203.pdf

Oracle Incorporated Java Tutorials, Lesson: Introduction to JAXP. Viitattu 08.12.2016. https://docs.oracle.com/javase/tutorial/jaxp/intro/index.html

What is the 3-Tier Architecture By Tony Marston. Viitattu 25.2.2017. http://www.tonymarston.net/php-mysql/3-tier-architecture.html

Widlfly kotisivu. Viitattu 08.12.2016. http://wildfly.org/about/

XML.com, A Technical Introduction to XML, Norman Walsh. Viitattu 8.12.2016. http://www.xml.com/pub/a/98/10/guide0.html?page=2#AEN58