POISSAOLOHERÄTTEET

Palvelurajapinta ja asiakassovellus

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tieto- ja Viestintätekniikan koulutusohjelma

Riihimäki, kevät 2019

Matti Palmén

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Riihimäki
Tieto- ja viestintätekniikka
Ohjelmistotekniikka

Tekijä Matti Palmén **Vuosi** 2019

Työn nimi Poissaoloherätteet

Työn ohjaaja/t Petri Kuittinen

TIIVISTELMÄ

Tämä opinnäytetyö on syntynyt asiakastarpeesta rakentaa sovellus palkkahallinnon ja henkilöstöhallinnon sovellusten väliseen kommunikointiin osana HRM-työhyvinvointiprosessia. Työhyvinvoinnilla tarkoitetaan tässä yhteydessä poissaoloihin liittyvien tapahtumien seurantaa ja niiden käsittelyä.

Asiakastarpeita tällaisesta toiminnasta ovat esittäneet Pohjois-Savon sairaanhoitopiiri, HUS ja Hyvinkään kaupunki. Esitysten pohjalta on mietitty tuotteistettavaa ratkaisua, joka voidaan konfiguroida kulloisenkin asiakkaan tarpeita vastaavaksi.

Poissaoloherätteet on CGI:n toteuttama palvelurajapintakokonaisuus, jossa asiakaspuolen sovellus lukee ajastetusti joka yö palkkajärjestelmän tietokannasta poissaoloihin liittyvää dataa, muodostaa lähetettävistä tiedoista JSON-muotoisen sanoman ja lähettää sanoman HRM-järjestelmässä pyörivälle REST-palvelulle, joka lukee sanoman, tarkistaa datan sisällön ja hakee esimiestiedot, joille lähetetään sähköpostiviestit tilanteesta. Viesti tallentuu myös HRM-järjestelmän tietokantaan myöhemmin tapahtuvan raportoinnin tarpeisiin.

Ohjelmointikieli on Java ja versio 8. Client osa on Java-sovellus, joka on paketoitu jar-tiedostoksi ja sitä ajetaan shell-skriptin kautta ja se saa tarvitsemansa lähtötiedot asiakaskohtaisesti määritetystä properties-tiedostosta. Service osa on REST-tyyppinen palvelu, joka on paketoitu war-tiedostoksi ja voidaan asentaa sovelluspalvelimelle esim. Tomcat. JSON-sanoman skeema toimii rajapintakuvauksena, josta on muodostettu Java-luokat.

Avainsanat HRM, JSON, REST, Varhaisen välittämisen prosessi

Sivut 28 sivua ja kolme demovideota

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Riihimäki
Information Technology
Programming Technology

Author Matti Palmén **Year** 2019

Subject Absence alarms

Supervisors Petri Kuittinen

ABSTRACT

This thesis was created as a document of the REST application between payroll and HR administration applications as a part of the HRM well-being process of the commissioners. Occupational well-being in this context refers to events related to absences and dealing with them.

Customer needs were presented by the Northern Savo Hospital District, HUS and the City of Hyvinkää, on the basis of which a productive solution was considered that could be configured to meet the needs of each customer.

Absence alarms is a service interface product made by CGI. The Client application records the absence data from the database of payroll system as scheduled every night, creates a JSON message of that data and sends it to the REST-service running next to the HR system. The service reads the message and checks the content of the message. It also searches manager information and sends email messages to them. The data of a JSON message is stored into the HR database for reporting purposes.

The programming language here is Java8. The Client part is a Java application that has been packed into a jar file and is run through a shell script. It gets the required information from a specified properties file. The Service part is a REST-type service that has been packed into a war file and can be installed on an application server, e.g. Tomcat. The schema of the JSON message serves as an interface description that has been converted into java classes.

Keywords HRM, JSON, early care process, REST

Pages 28 pages and three demo videos

SISÄLLYS

1. JOHDANTO

Työssä rakennetaan sovellus HRM-työhyvinvointiprosesseihin liittyen. Puhutaan varhaisen välittämisen sähköisestä prosessista, joka mahdollistaa hyvinvointikeskustelun esimiehen ja henkilön välillä, sekä keskustelun raportoinnin. Varhaisen välittämisen sähköinen prosessi etenee siten, että esimies saa palkkajärjestelmästä herätteen toistuvista poissaoloista ja sitten käynnistää tarvittaessa keskustelun HRM-järjestelmässä.

Poissaoloherätteet sovellus on syntynyt Pohjois-Savon sairaanhoitopiirin (PSSHP) aloitteesta ja vastaavaa toiminnallisuutta ovat kaivanneet myös Helsingin ja Uudenmaan sairaanhoitopiiri (HUS) ja Hyvinkään kaupunki. Niinpä tavoitteena onkin ollut tuottaa tuotteistettu ratkaisu asiakkaiden erilaiset tarpeet ja ajoympäristö huomioiden.

Työn tavoitteena on määritellä ja toteuttaa liityntä kahden tietojärjestelmän välillä, jossa palkkajärjestelmä lähettää tietoa HRM-järjestelmään ja paluuviestin perusteella päivittää oman tietovaraston taulua.
Liittymän toteutukseen tarvitaan siis rajapintakuvaus, palvelusovellus ja palvelua käyttävä asiakassovellus ja niihin tietokanta- ja sähköpostitoiminnallisuudet.

Määrittelyosiossa käydään läpi tuotettavan sovelluksen rakenne ja mihin asioihin tässä työssä on keskitytty. Muutama osio kokonaisprosessista on jätetty tämän työn ulkopuolelle, pääpainon ollessa tiedon siirrossa järjestelmien välillä.

Ohjelmointiosiossa kerrotaan esim. toteutuksen vaiheista ja työkaluista sekä miten haluttu lopputulos on saavutettu. Järjestelmän toiminta ja ohjelmoinnin toteutus esitellään demovideoissa, joita voi käydä lataamassa katselua varten kappaleen 7. ”Demovideot” alta.

Työssä esiintyy joitakin tuttuja ja tuntemattomampia lyhenteitä, joita on selitetty alla olevassa taulukossa.

HRM	Human Resource Management, Henkilöstöhallinnan tietojärjestelmä
Prima	Palkkajärjestelmä
JSON	JavaScript Object Notation
XML	Extensible Markup Language
SKEEMA	Sanoman rakennekuvaus, tietomalli
REST	REpresentational State Transfer
IDE	Integrated Development Environment
API	Application programming interface

2. POISSAOLOHERÄTTEIDEN IDEA JA MÄÄRITTELY

Poissaoloherätteillä osana työhyvinvointiprosessia tarkoitetaan esimiehelle välitettävää tietoa työntekijöiden eri pituisista poissaoloista, jotta niihin voidaan puuttua ja selvittää syyt ja ryhtyä tarvittaviin toimenpiteisiin.

CGI Suomi on aikojen saatossa rakentanut palkkahallinnon sovelluksia ja henkilöstönhallinnan sovelluksia asiakkailleen. Yrityskauppojen myötä saman tyyppisiä sovelluksia löytyy nykyään tarjoamasta varsin laaja kirjo ja niitä on kehitetty eri asiakassegmenttien (valtio, kunnat, sairaanhoitopiirit, yksityiset yritykset) tarpeita silmällä pitäen. Eri tietojärjestelmistä löytyy vähän erilaista tietoa ja sen vuoksi pitää rakentaa palveluita järjestelmien väliseen kommunikointiin. Poissaoloherätteet-sovellus on esimerkki sellaisesta. Tekstissä esiintyy palkkajärjestelmän tuote Prima ja henkilöstöhallinnon järjestelmä HRM, joita käytetään lyhyiden nimien vuoksi.

Alla olevassa kuvassa on kerrottu hiukan poissaoloherätteiden tarkoituksesta ja miten se eri yhteisöillä toteutuu.

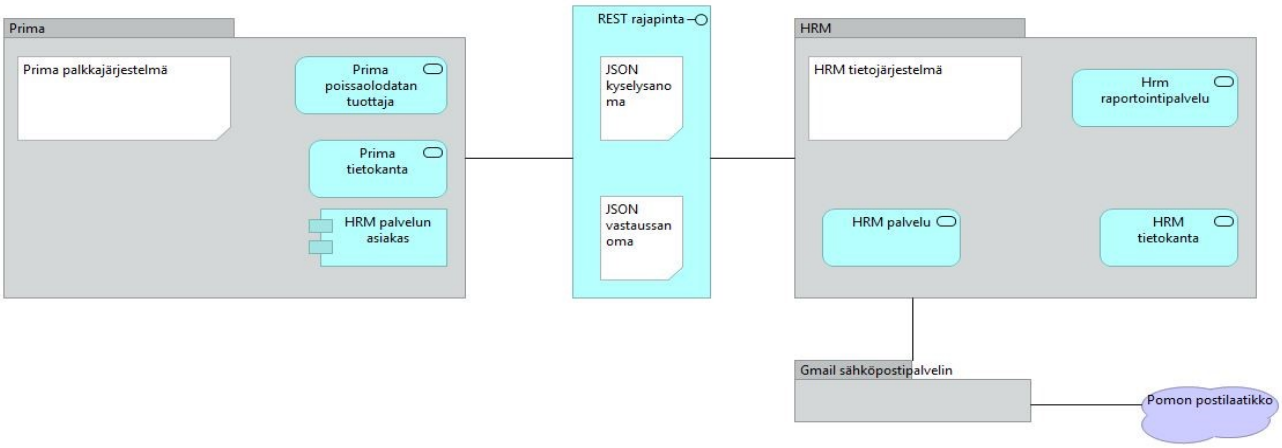
Kuva 1. Poissaoloherätteet prosessina

2.1 Kokonaiskuva kommunikaatiosta

Poissaoloherätteiden kokonaiskuvaa on pyritty havainnollistamaan seuraavassa kuvassa 2, johon on myös tuotu palkkajärjestelmän kuvausta asiakassovelluksen roolissa. Tuossa kuvassa palkkajärjestelmä on vasemmalla puolella ja siihen on laitettu poissaolodatan tuottava ohjelma, tietokanta ja HRM-asiakassovellus, joka lähettää poissaolodataa HRM-järjestelmän suuntaan HRM-palvelulle.

HRM-järjestelmä on kuvattu oikealla puolella ja siihen on liitetty tähän työhön liittyviä komponentteja esim. HRM-palvelu, järjestelmän tietokanta ja raportointipalvelu.

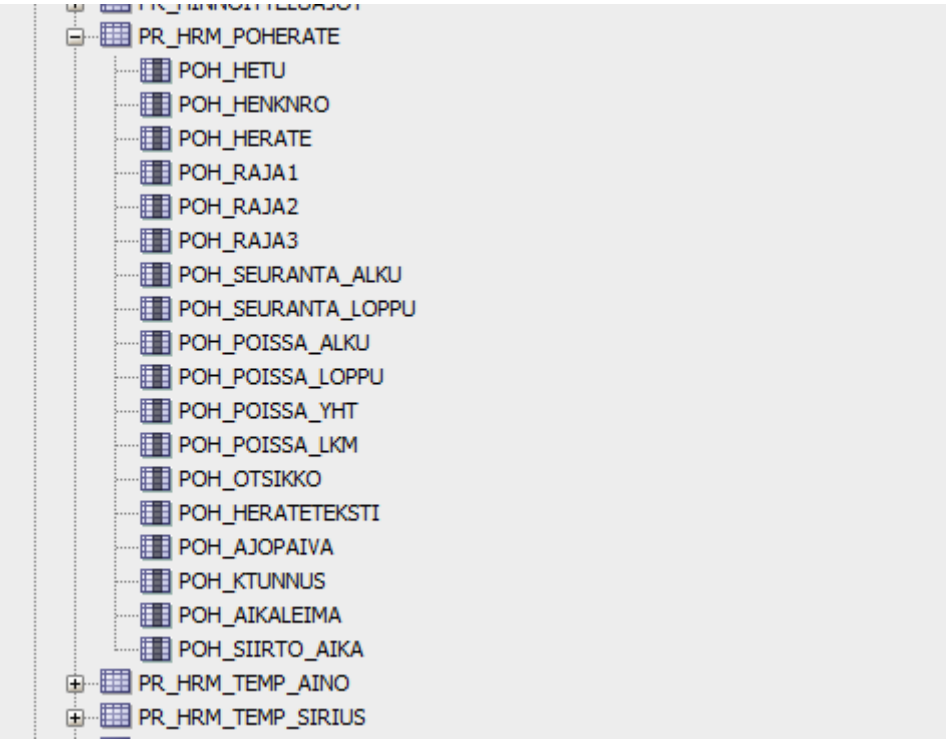
Näiden järjestelmien väliin on laitettu REST-rajapinta JSON-sanomineen. REST on lyhenne sanoista REpresentational State Transfer (Hewitt, 2009, s. 380). Lisäksi kuvassa on sähköpostipalvelin ja esimiehen postilaatikko.



Kuva 2. Poissaoloherätteet, pelkistetty kokonaiskuva

2.2 Lähtötiedot palkkajärjestelmästä

Poissaoloherätteiden asiakassovellus (HRM-asiakas) lukee palkkajärjestelmänä olevan Priman tietokannan taulua PR_HRM_POHERATE (kuva 3), jonne erillinen poissaolojen poimintasovellus kerää joka yö ajettavassa ajossa dataa HRM-asiakassovelluksen lähtötiedoiksi. Tuo lähtötietoa tuottava poimintasovellus on rajattu tämän työn ulkopuolelle. Ideana siinä on laskea henkilöiden poissaoloista kertymiä ja katsoa sitten kuinka suuri kertymä vastaa asiakkaiden kanssa sovittuja sanallisia kuvauksia ja muodostaa sitten kertymille sanalliset heräteselitteet. Nuo rivit poimintasovellus sitten tallentaa em. tietokannan tauluun.



Kuva 3. Priman lähtötiedot PR_HRM_POHERATE taulusta

HRM-asiakassovelluksen ideana on, että sovellus lukee taulusta tietoja riveiltä, joissa POH_SIIRTO_AIKA on null ja päivittää sitten ajon lopussa onnistuneesti siirretyistä riveistä ajohetken aikaleiman tuohon kenttään. Näin ollen tuota riviä ei seuraavassa ajossa enää poimita mukaan lähetettäviin viesteihin. Alla on esimerkki lähtötilanteesta, jossa on demoa varten toteutettu keksitty data.

POH_HETU	POH_HENKNRO	POH_OTSIKKO	POH_HERATETEKSTI	POH_SIIRTO_AIKA
010101-0033		ei huvita	heppu lintsaa duuneista	(null)
160170-1246		Minnan_testi_20181005	Henkilöllä ESITESTAUS-ESARA ESSI on 6 kappaletta alle 5 kalenteripäivän sairaus...	(null)
010255-123D		Minnan_testi_20181005	Henkilöllä UUSI-POISSAOLOKOODI UNTO on 30 kalenteripäivää tai yli sairauspoissa...	(null)
010277-456U		Minnan_testi_20181005	Henkilöllä POISSAOLO PÄIVI on 30 kalenteripäivää tai yli sairauspoissaoloja, hu...	(null)
010374-4611		Minnan_testi_20181005	Henkilöllä VEROPÄIVÄ VIILIS on 30 kalenteripäivää tai yli sairauspoissaoloja, h...	(null)
010470-2577		Minnan_testi_20181005	Henkilöllä LOMARAHAVAPAA LAURI on 30 kalenteripäivää tai yli sairauspoissaoloja...	(null)
010555-789A		Minnan_testi_20181005	Henkilöllä POISSAOLO OSSI on 30 kalenteripäivää tai yli sairauspoissaoloja, huo...	(null)
010660-234H		Minnan_testi_20181005	Henkilöllä LOMARAHAVAPAA LEENA-MAIJA on 30 kalenteripäivää tai yli sairauspoiss...	(null)
011077-127N		Minnan_testi_20181005	Henkilöllä TESTI TEUVO on 30 kalenteripäivää tai yli sairauspoissaoloja, huomio...	(null)
040771-407N	10881	Minnan_testi_20181005	Henkilöllä Tilistö Ville on 30 kalenteripäivää tai yli sairauspoissaoloja, huom...	(null)
170172-171V	10888	Minnan_testi_20181005	Henkilöllä LOPPUTILIÖ LATE on 30 kalenteripäivää tai yli sairauspoissaoloja, hu...	(null)
210577-123D		Minnan_testi_20181005	Henkilöllä TOIVEIKAS TOIVO on 30 kalenteripäivää tai yli sairauspoissaoloja, hu...	(null)
241170-124D		Minnan_testi_20181005	Henkilöllä LOMARAHA LIISI on 30 kalenteripäivää tai yli sairauspoissaoloja, huo...	(null)
300860-125A		Minnan_testi_20181005	Henkilöllä WI-POLO VINSKI on 30 kalenteripäivää tai yli sairauspoissaoloja, huo...	(null)
300860-234U		Minnan_testi_20181005	Henkilöllä POISSAOLO PÄIVIKKI on 30 kalenteripäivää tai yli sairauspoissaoloja,...	(null)
300860-235V		Minnan_testi_20181005	Henkilöllä POLOLIITTYMÄ PASIPEKKA on 30 kalenteripäivää tai yli sairauspoissaol...	(null)
301170-133K		Minnan_testi_20181005	Henkilöllä POLO REIJO on 30 kalenteripäivää tai yli sairauspoissaoloja, huomioi...	(null)
301170-912P	10323	Minnan_testi_20181005	Henkilöllä POISSAOLO PAULA on 30 kalenteripäivää tai yli sairauspoissaoloja, hu...	(null)
010160-484J		Minnan_testi_20181005	Henkilöllä TESTI TIINA-MAIJA on 6 kappaletta alle 5 kalenteripäivän sairauspois...	(null)
010170-1124		Minnan_testi_20181005	Henkilöllä POLO PEPPI on 6 kappaletta alle 5 kalenteripäivän sairauspoissaoloa,...	(null)
010255-124E		Minnan_testi_20181005	Henkilöllä POISSAOLOKOODI PIRJO on 6 kappaletta alle 5 kalenteripäivän sairausp...	(null)
010267-254R		Minnan_testi_20181005	Henkilöllä VPK-116 VILHELMIINA on 6 kappaletta alle 5 kalenteripäivän sairauspo...	(null)
010680-457V		Minnan_testi_20181005	Henkilöllä LOMARAHAVAPAA LARS on 6 kappaletta alle 5 kalenteripäivän sairauspoi...	(null)

Kuva 4. Taulun sisältöä, rivit jotka haetaan sanomaan

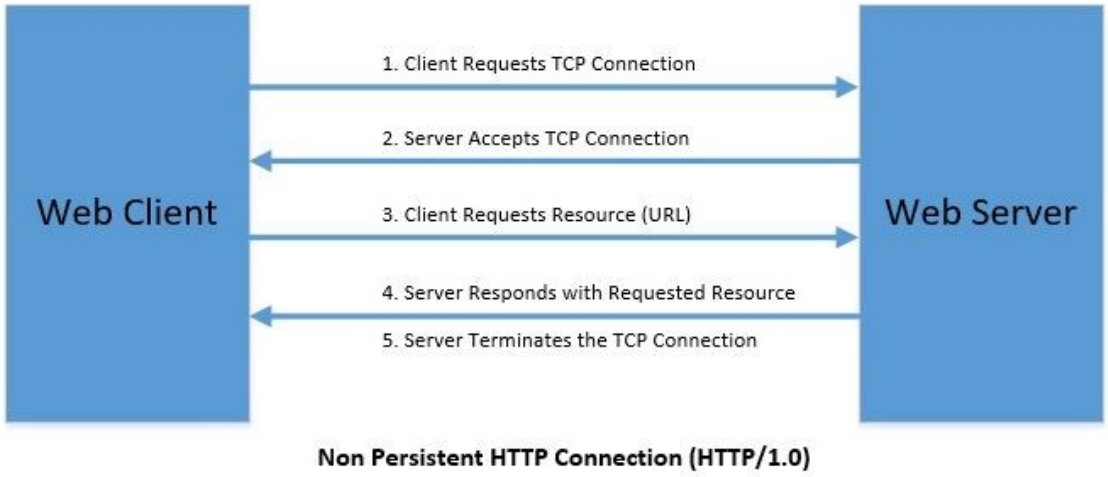
Lähetettävään viestiin poimittavia kenttiä em. taulusta ovat:

- poh_hetu (henkilötunnus) tai
- poh_henknro (henkilönumero) HRM järjestelmästä riippuen
- poh_otsikko (viestin aihe)
- poh_heräteteksti (poissaoloherätteen viestin sisältö)

2.3 Tietojen lähetys HRM-järjestelmään

HRM-järjestelmä pitää sisällään REST-tyyppisen palvelun (HRM-palvelu), jolle lähetetään JSON-muodossa oleva viesti, jonka sisältö on määritelty kappaleessa 4.1 olevan rajapintakuvauksen (skeema) mukaan.

Kuvassa 5 on kuvattu REST-tyyppinen sanoman välitys yleisesti, jossa yhteys pitää aina avata jokaista sanoman lähetystä varten. Kun sanoma palautuu takaisin asiakassovellukselle niin yhteys ”kuolee”. Niinpä puhutaan myöskin tilattomasta kommunikaatiosta.



Kuva 5. REST sanomavälityksen periaate (Baeldung, 2019)

Rajapintakuvausta (skeema) vasten tehdään Java-luokat, joiden avulla itse viesti rakennetaan. Tällöin viestin rakenne ja tietosisältö voidaan validoida palvelua vastaavaksi ja sama pätee myöskin vastaanottavassa päässä varsinaisen palvelun puolella.

2.4 Tietojen käsittely HRM-järjestelmässä

HRM-palvelu odottaa saavansa kappaleessa 4.1 määritellyn rajapintakuvauksen mukaisen sanoman, josta tiedot luetaan eteenpäin käsittelyä varten. HRM-palvelu lukee jokaisen viestin data-alkion, tarkistaa löytyykö järjestelmästä esimiestiedot henkilölle ja tallettaa viestin omaan kantaansa myöhempiä mahdollisia raportointitarpeita varten. HRM-palvelu lähettää

henkilön esimiehelle sähköpostiviestin poissaoloista, mikäli esimiestiedot löytyvät.

2.5 Paluuviesti takaisin palkkajärjestelmään

HRM-palvelu lähettää paluuviestin takaisin HRM-asiakkaalle. Paluuviesti noudattaa paluuviestin skeemaa (kuva 9). Käytännössä kyselysanomaan lisätään rivikohtainen statuskoodi ja statusviesti, jotka kertovat miten sanoman sisäänluku on onnistunut.

2.6 Lopputoimet asiakassovelluksessa

Asiakassovellus päivittää pr_hrm_poherate-taulun poh_siirto_aika-kenttää ajohetken aikaleimalla niiden rivien osalta, jotka on HRM-palvelussa onnistuneesti vastaanotettu. Viestisanoman statuskoodista voidaan päätellä tämä tieto.

POH_HETU	POH_HENKNRO	POH_OTSIKKO	POH_HERATETEKSTI	POH_SIIRTO_AIKA
010101-0033		ei huvita	heppu lintsaa duuneista	(null)
160170-1246		Minnan_testi_20181005	Henkilöllä ESITESTAUS-ESARA ESSI on 6 kappaletta alle 5 kalenteripäivän sairaus...	2019041814560
010255-123D		Minnan_testi_20181005	Henkilöllä UUSI-POISSAOLOKOODI UNTO on 30 kalenteripäivää tai yli sairauspoissa...	2019041814560
010277-456U		Minnan_testi_20181005	Henkilöllä POISSAOLO PÄIVI on 30 kalenteripäivää tai yli sairauspoissaoloja, hu...	2019041814560
010374-4611		Minnan_testi_20181005	Henkilöllä VEROPÄIVÄ VIILIS on 30 kalenteripäivää tai yli sairauspoissaoloja, h...	2019041814560
010470-2577		Minnan_testi_20181005	Henkilöllä LOMARAHAVAPAA LAURI on 30 kalenteripäivää tai yli sairauspoissaoloja...	2019041814560
010555-789A		Minnan_testi_20181005	Henkilöllä POISSAOLO OSSI on 30 kalenteripäivää tai yli sairauspoissaoloja, huo...	2019041814560
010660-234H		Minnan_testi_20181005	Henkilöllä LOMARAHAVAPAA LEENA-MAIJA on 30 kalenteripäivää tai yli sairauspoiss...	2019041814560
011077-127N		Minnan_testi_20181005	Henkilöllä TESTI TEUVO on 30 kalenteripäivää tai yli sairauspoissaoloja, huomio...	2019041814560
040771-407N	10881	Minnan_testi_20181005	Henkilöllä Tilistö Ville on 30 kalenteripäivää tai yli sairauspoissaoloja, huom...	2019041814560
170172-171V	10888	Minnan_testi_20181005	Henkilöllä LOPPUTILIÖ LATE on 30 kalenteripäivää tai yli sairauspoissaoloja, hu...	2019041814560
210577-123D		Minnan_testi_20181005	Henkilöllä TOIVEIKAS TOIVO on 30 kalenteripäivää tai yli sairauspoissaoloja, hu...	2019041814560
241170-124D		Minnan_testi_20181005	Henkilöllä LOMARAHA LIISI on 30 kalenteripäivää tai yli sairauspoissaoloja, huo...	2019041814560
300860-125A		Minnan_testi_20181005	Henkilöllä WT-POLO VINSKI on 30 kalenteripäivää tai yli sairauspoissaoloja, huo...	2019041814560
300860-234U		Minnan_testi_20181005	Henkilöllä POISSAOLO PÄIVIKKI on 30 kalenteripäivää tai yli sairauspoissaoloja,...	2019041814560
300860-235V		Minnan_testi_20181005	Henkilöllä POLOLIITTYMÄ PASIPEKKA on 30 kalenteripäivää tai yli sairauspoissaol...	2019041814560
301170-133K		Minnan_testi_20181005	Henkilöllä POLO REIJO on 30 kalenteripäivää tai yli sairauspoissaoloja, huomioi...	2019041814560
301170-912P	10323	Minnan_testi_20181005	Henkilöllä POISSAOLO PAULA on 30 kalenteripäivää tai yli sairauspoissaoloja, hu...	2019041814560
010160-484J		Minnan_testi_20181005	Henkilöllä TESTI TIINA-MAIJA on 6 kappaletta alle 5 kalenteripäivän sairauspois...	2019041814560
010170-1124		Minnan_testi_20181005	Henkilöllä POLO PEPPI on 6 kappaletta alle 5 kalenteripäivän sairauspoissaoloa,...	2019041814560
010255-124E		Minnan_testi_20181005	Henkilöllä POISSAOLOKOODI PIRJO on 6 kappaletta alle 5 kalenteripäivän sairausp...	2019041814560
010267-254R		Minnan_testi_20181005	Henkilöllä VPK-116 VILHELMIINA on 6 kappaletta alle 5 kalenteripäivän sairauspo...	2019041814560
010680-457V		Minnan_testi_20181005	Henkilöllä LOMARAHAVAPAA LARS on 6 kappaletta alle 5 kalenteripäivän sairauspoi...	2019041814560

Kuva 6. Lopputilanne päivityksen jälkeen

3. POISSAOLOHERÄTTEIDEN RAJAPINTAKUVAUKSET

Poissaoloherätteet on toteutettu viestipohjaisena tilattomana ratkaisuna, jossa on palvelupyynnön käsittelevä komponentti (HRM-palvelu) ja palvelua pyytävä sovellus (HRM-asiakas). Jotta molemmat sovellukset tietävät minkälaista viestiä liikutellaan ja miten viesti välittyy asiakassovellukselta palvelulle ja takaisin, tarvitaan jokin yhteisesti sovittu tapa toimia.

Palvelurajapintatoteutukset vaativat siis jonkinlaisen rajapintakuvauksen eli skeeman, jossa määritellään viestin rakenne ja sisältö. Skeema voidaan lisäksi versioida, jotta pystytään helpommin kertomaan myöhemmin mitä skeeman versiota kulloinenkin palvelu noudattaa. Viestin rakenne on vanhemmissa sovelluksissa usein Xml-muotoista, mutta nykyisin yleisempi tapa on käyttää JSON-muotoista dataa. Tässä työssä käytetään JSON-muotoista sanomarakennetta.

Olio-ohjelmoinnissa skeemasta muodostetaan yleensä luokkamalli, jonka avulla luetaan ja kirjoitetaan skeemaa noudattavaa sanomarakennetta. Tässä työssä skeema muodostetaan luokkarakenteeksi Jackson ohjelmointirajapintaa (API) hyödyntäen (Jenkov, 2016).

Luokkamallia ei tarvitse kirjoittaa itse vaan yleensä se generoidaan jollakin työkalulla. Verkosta löytyy lukuisa joukko työkaluja tähän tarkoitukseen. Tässä työssä on käytetty jsonschema2pojo-osoitteesta löytyvää apuvälinettä (Littlejohn, 2012-2018)

Rajapintakuvaukset ja esimerkkisanomat on kuvattu seuraavissa luvuissa.

3.1 HRM järjestelmään lähetettävän viestin rajapintakuvaus


```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "title": "Initiations send request",
  "description": "HTTP request for sending initiations",
  "type": "object",
  "properties": {
    "document": {
      "description": "Object holding document data",
      "type": "object",
      "properties": {
        "id": {
          "description": "Id of document data type, use constant value 'initiation'.",
          "type": "string"
        }
      },
      "required": ["id"]
    },
    "items": {
      "description": "List of initiations",
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "id": {
            "description": "Initiation id, fixed value will be provided during initial setup.",
            "type": "string"
          },
          "personId": {
            "description": "Person id or ssn.",
            "type": "string"
          },
          "ssn":{
            "description": "Whether personId attribute is ssn or plain number. Possible values true|false",
            "type": "boolean"
          },
          "subject": {
            "description": "Subject of a initiation message.",
            "type": "string"
          },
          "message": {
            "description": "Body of a initiation message.",
            "type": "string"
          }
        },
        "required": ["id","personId","ssn","subject","message"]
      }
    }
  },
  "required": ["document"]
}
```


Kuva 7. Kyselysanoman JSON-skeema

Alla on esimerkkisanoma, joka toteuttaa em. kyselysanoman rajapintakuvauksen.


```
{
    "document": {
        "id": "initiation"
    },
    "items": [{
            "id": "absenceInitiation",
            "personId": "123456",
            "ssn": false,
            "subject": "Heräte poissaolosta",
            "message": "Henkilön X poissolo aikavälillä ...."
        },
        {
            "id": "absenceInitiation",
            "personId": "123456-789A",
            "ssn": true,
            "subject": "Heräte poissaolosta",
            "message": "Henkilön Y poissolo aikavälillä ...."
        }
    ]
}
```


Kuva 8. Kyselysanoman esimerkki

3.2 HRM järjestelmän lähettämän paluuviestin rajapintakuvaus


```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "title": "Initiations send response",
  "description": "HTTP response for sending initiations",
  "type": "object",
  "properties": {
    "data": {
      "description": "List of initiations",
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "id": {
            "description": "Initiation id, fixed value will be provided during initial setup.",
            "type": "string"
          },
          "personId": {
            "description": "Person id or ssn.",
            "type": "string"
          },
          "ssn":{
            "description": "Whether personId attribute is ssn or plain number. Possible values true|false",
            "type": "boolean"
          },
          "subject": {
            "description": "Subject of a initiation message.",
            "type": "string"
          },
          "message": {
            "description": "Body of a initiation message.",
            "type": "string"
          },
          "status": {
            "description": "validation status of current initation item.",
            "type": "string"
          },
          "statusMessage": {
            "description": "Validation status messag current initation item.",
            "type": "string"
          }
        },
         "required": ["id","personId","isSsn","subject","message"]
      }
    },
    "message": {
      "description": "HTTP Response status message.",
      "type": "string"
    },
    "status": {
      "description": "HTTP Response status",
      "type": "integer"
    }
  },
  "required": ["data","message","status"]
}
```


Kuva 9. Vastaussanoman skeema

Alla on esimerkki vastaussanomasta, jossa käytännössä kyselysanomassa olevaan yksittäiseen data-alkioon lisätään status ja statusviesti siitä onnistuttiinko kyseisen tiedon vastaanotossa ja löytyikö henkilölle esimiestietoa.

```
{
    "data": [
        {
            "id": "primaAbsence",
            "personId": "100208",
            "ssn": false,
            "subject": "Heräte poissaolosta",
            "message": "Henkilön X poissolo aikavälillä",
            "status": 400,
            "statusMessage": "Superior user code missing"
        },
        {
            "id": "primaAbsence2",
            "personId": "200291-984R",
            "ssn": true,
            "subject": "Heräte poissaolosta",
            "message": "Henkilön Y poissolo aikavälillä",
            "status": 200,
            "statusMessage": "OK"
        }
    ],
    "status": 200,
    "message": "OK with validation error(s)"
}
```

Kuva 10. Vastaussanoman esimerkki

3.3 Rajapintakuvaus Java-luokiksi

Rajapintakuvaus kertoo, minkälainen on viestin muoto ja sisältö. Jotta sanoman rakentaminen ja lukeminen itse sovelluksessa olisi helpompaa, niin tuota varten voidaan hyödyntää verkosta löytyviä työkaluja, joilla skeemasta voidaan muodostaa luokkamalli. Luokkamallin avulla pystytään näppärästi luomaan skeeman mukainen viesti. Kun viesti on luotu skeemasta generoidun luokkamallin avulla, niin samalla tulee suoritettua myös rakenteellinen validointi.

Tässä työssä on käytetty alla olevan linkin takaa avautuvaa generointityökalua Java-luokkien muodostamiseksi JSON-skeemasta:

http://www.jsonschema2pojo.org/

Työkalussa skeema kopioidaan vasemmalle puolelle laatikkoon, ja määritellään juuri-luokan nimi ja pakettirakenne, ja työkalu muodostaa luokat sen mukaan. Zip-pakettiin muodostetut luokat tai lähdekoodit voidaan kivasti liittää kehitettävään projektiin ja alkaa käyttää.

4. ASIAKASSOVELLUS TEKNINEN TOTEUTUS (CLIENT OSA)

Kun rajapintakuvaus on olemassa, niin palvelun ja asiakassovelluksen toteuttajat voivat vaipua kammioihinsa toteuttamaan osuuttaan. Tällainen rakenne ei sido tekijöitä mihinkään tiettyyn ohjelmointikieleen vaan toteutus voidaan periaatteessa valita vapaasti, ellei muuta vaatimusta aseteta.

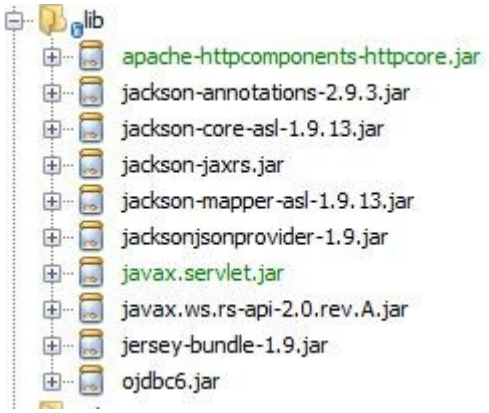
Kun asiakkaiden palvelin ja ajoympäristöt ovat poikkeuksetta IBM Aix-ympäristöjä, joissa sovellukset käyttävät Oracle-tietokantoja, niin sovellukset on päätetty koodata Javalla, varsinkin kun siihen on löytynyt osaamista omasta takaa. Koska Java on alustariippumaton ohjelmointiympäristö, ohjelmistokehitys voidaan tehdä kehittäjän omalla koneella ja sitten paketoida valmis sovellus tuotantoon sopivaksi.

Tässä työssä asiakasohjelma on tavallinen Java-sovellus, joka käyttää Jersey-ohjelmointirajapintaa HRM-palvelun kanssa kommunikointiin. Tässä sovelluksessa kyselyn oikeutus eli autentikointi asiakkaan ja palvelun välillä on toteutettu http-headerin sisällä kulkevassa merkkijonossa ja se käyttää ”basic access authentication” menetelmää (Wikipedia, 2019). Tämän on katsottu olevan riittävä, kun liikenne asiakassovelluksen ja palvelun välillä tapahtuu sisäverkossa.

Asiakassovellusta ajetaan tuotantoympäristöissä ajastetusti asiakkaan määrittelemin aikavälein. Käytännössä tuo tehdään jollakin komentoriviltä ajettavalla skriptillä, joka sitten ajaa varsinaisen ohjelman joka saa lähtötiedot jostain parametritiedostosta.

4.1 Asiakassovelluksen ohjelmointi

Asiakassovellus on tavallinen Java-ohjelma, jossa on main() metodi, josta kaikki lähtee liikkeelle. Toteutus on tehty NetBeans IDE 8.2 versiolla, Javasta on käytetty versiota 8. Lisäksi on käytetty alla olevassa kuvassa esitettyjä kirjastoja.



Kuva 11. Asiakassovelluksen kirjastot

Seuraavassa käydään läpi joitain pääkohtia itse koodista. Lähinnä miten yhteys palveluun muodostetaan ja mitä tietoja asiakassovellus tarvitsee kommunikointiin palvelun kanssa.

4.2 Asiakassovelluksen ohjelmakoodin pääkohtia

Alla olevassa koodissa on kommunikaatio asiakassovelluksen ja palvelun välillä. Sanoman generointi luokkarakenteeseen, palvelun osoitteen määrittäminen ja itse viestin lähetys.

```
import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.ClientResponse;
import com.sun.jersey.api.client.WebResource;
import com.sun.jersey.api.client.config.DefaultClientConfig;
import com.cgi.hrm.wakeup.service.WakeUp;
import com.sun.jersey.api.client.ClientResponse;
import com.cgi.hrm.wakeup.service.response.WakeUpResponse;


DefaultClientConfig defaultClientConfig = new DefaultClientConfig();
defaultClientConfig.getClasses().add(JacksonJsonProvider.class);

// JSON sanoma skeemasta generoituun luokkamalliin
WakeUp MyWakeUp = readTaulu1.getWakeup();

Client restClient = Client.create(defaultClientConfig);

// asetetaan endpoint url , jonne sanoma lähetetään
WebResource webResource = restClient.resource(hrmEndPointAdddress);

// sanoman lähetys POST metodilla palvelulle
ClientResponse response = webResource.type("application/json")
                    .header("Authorization", headerAuthentication)
                    .post(ClientResponse.class, MyWakeUp);
```

Kuva 12. Sanoman lähettäminen asiakassovelluksesta

Ennen paluuviestin lukemista tarkistetaan, onnistuiko tunnistautuminen palveluun, ettei turhaan lueta olematonta sanomaa.

```
status = response.getStatus();
  log.logger.info("status code returned "+status+"\n\n");

  if(status != HttpServletResponse.SC_UNAUTHORIZED)
  {
     getDataFromResponse(response);
  }
```

Kuva 13. Tunnistautumisen tarkistus

Seuraavassa on esitetty koodista kohta, jossa tehdään paluuviestin luku skeemasta generoituun luokkarakenteeseen ja lopputoimiin ryhtyminen, mikäli paluuviesti sisältää dataa.

```
// luetaan JSON paluuviesti skeemasta muodostettuun oliomalliin
WakeUpResponse myWakeupResponse = response.getEntity(WakeUpResponse.class);

PrimaDbTable Db = new PrimaDbTable();
// jos paluuviestistä löytyy data päivitetään lähtöaineiston taulua
if(myWakeupResponse!=null && myWakeupResponse.getData()!=null)
{
      Db.updatePohSiirtoAika(con, user, pass, myWakeupResponse.getData(),
                        this.itemCollection.getCollectionOfItems());
}
```

Kuva 14. Vastaussanoman lukeminen ja lopputoimet

4.3 Asiakassovelluksen lähtötietojen lukeminen ja kyselysanoman muodostus

Asiakassovellus lukee lähtötiedot hrm.properties-tiedostosta, jonka se saa kun ohjelmaa ajetaan. Properties-tiedoston sisältö on esitetty alla olevassa kuvassa. Demoa varten hrmEndpointAddress osoittaa paikalliseen koneeseen ja tietokantayhteys osoittaa CGI:n testiympäristön kantapalvelimelle.

```
connectingString=jdbc:oracle:thin:@10.158.21.76:1525:AVKUNTA
primaDatabaseUsername=pr
primaDatabasePassword=pr
hrmEndpointAddress=http://localhost:8680/HrmService/rest/json/persons
hrmHeaderAuthentication=Basic YWRtaW46YWRtaW4=
logFile=/tmp/hrmheratteet/PrimaRestClient.log
```

Kuva 15. Asiakassovelluksen lähtötiedot hrm.properties-tiedostosta

Tietokannan ohjaustiedoista taulusta PR_TAULU1 haetaan tieto, onko organisaatiolla käytössä henkilön yksilöivä suomalainen henkilötunnus vai käytetäänkö järjestelmissä olevaa henkilönumeroa. Tämän jälkeen luetaan taulusta PR_HRM_POHERATE kaikki taulussa olevat rivit, joissa kentän POH_SIIRTO_AIKA arvo on null. Tuosta tiedämme, että tietoa ei olla lähetetty aiemmin.

Samalla muodostetaan skeemaa vastaava luokkamalli lähetettävästä sanomasta. Kuten kuvasta 8 nähdään, niin kyselysanoma on verrattain yksinkertainen muodoltaan, käytännössä lista em. tietokannan taulun rivejä paketoituna Document-olion sisälle.

Alla on vielä esimerkki WakeUp.java lähdekoodina, josta nähdään minkälaista koodia generointityökalu tuottaa.

```java
package com.cgi.hrm.wakeup.service;

import java.util.List;
import com.fasterxml.jackson.annotation.JsonInclude;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.annotation.JsonPropertyDescription;
import com.fasterxml.jackson.annotation.JsonPropertyOrder;

@JsonInclude(JsonInclude.Include.NON_NULL)
@JsonPropertyOrder({
    "document",
    "items"
})
public class WakeUp {

    @JsonProperty("document")
    @JsonPropertyDescription("Object holding document data")
    private Document document;

    @JsonProperty("items")
    @JsonPropertyDescription("List of initiations")
    private List<Item> items = null;

    @JsonProperty("document")
    public Document getDocument() {
        return document;
    }

    @JsonProperty("document")
    public void setDocument(Document document) {
        this.document = document;
    }

    @JsonProperty("items")
    public List<Item> getItems() {
        return items;
    }

    @JsonProperty("items")
    public void setItems(List<Item> items) {
        this.items = items;
    }
}
```

Kuva 16. Skeemasta generoitu pääluokka

4.4 JSON-sanoman lähetys

Sanoma lähetetään Jersey-ohjelmointirajapinnan luokkia apuna käyttäen. Ennen sanoman lähettämistä POST-tyyppisenä, kerrotaan missä osoitteessa viestiä lukeva palvelu sijaitsee (hrmEndPointAdddress), minkä tyyppistä dataa ollaan lähettämässä (application/json) ja kyselyn tekemiseen oikeuttava merkkijono (headerAuthentication). POST-metodille kerromme vielä minkä niminen on JSON-sanoman juuriluokka tai pääluokka, jonka alta löytyy varsinainen data. Kaikki tämä näkyy kuvassa 12.

4.5 Paluuviestin lukeminen ja lopputoimet

Paluuviestistä katsotaan, onnistuiko palveluun tunnistautuminen. Kuvassa 13 tarkistetaan tuota ja mikäli onnistui, niin jatketaan.

Kun jäljempänä esiteltävä palvelusovellus on vastaanottanut ja lukenut sanoman, se lähettää asiakassovellukselle paluuviestissä tiedon mitkä rivit on onnistuneesti luettu ja saatu kohdistettua esimiestietoihin ja missä on epäonnistuttu. Paluusanomassa jokaisessa rivitiedossa on lisänä statuskoodi ja siihen liittyvä selite, jonka perusteella päätellään, kirjoitetaanko paluuviestin aikaleima tietokannan tauluun PR_HRM_POHERATE, kenttään POH_SIIRTO_AIKA.

Kuvassa 14 olevassa ohjelmassa luetaan vastaussanoma JSON-skeemasta luotuun Java-luokkaan aiemmin luodun ClientResponse-olion avulla seuraavasti:

WakeUpResponse myWakeupResponse = response.getEntity(WakeUpResponse.class);

4.6 Asiakassovelluksen paketointi ja suoritus tuotantoympäristössä

Asiakassovellus on tavallinen Java-ohjelma, jossa pääluokasta löytyy main metodi. Lisäksi on tarpeellinen määrä apuluokkia JSON-sanoman käsittelyyn ja tietokantaoperaatioihin. Javalla toteutettu asiakasprojekti paketoidaan Javan jar-paketiksi (PrimaRestClient.jar) ja suoritus tai ajo tapahtuu komennolla:

```
java –jar PrimaRestClient.jar hrm.properties
```

Ohjelma lukee tarpeelliset parametrit jo aiemmin kerrotusta hrm.properties-tiedostosta. Suorittaminen vaatii mm. tietokannan yhteystietoja, HRM-palvelun osoitetiedot, tunnistetiedot ja lokitiedoston tiedot. Alla olevassa laatikossa on kuvattu erään ympäristön tiedot.

```
connectingString=jdbc:oracle:thin:@10.158.21.76:1525:AVKUNTA
primaDatabaseUsername=pr
primaDatabasePassword=pr
hrmEndpointAddress=http://10.158.22.58:30600/HrmService/rest/json/persons
hrmHeaderAuthentication=qvT5WgWu5meLtv
logFile=/tmp/hrmheratteet/PrimaRestClient.log
```

Kuva 17. Asiakasohjelman tarvitsemat lähtötiedot

Tuotannossa ajo suoritetaan käytännössä ajastettuna skriptinä unix-järjestelmän crontab:iin laitettuna (Alexander , 2019). Alla esimerkki yhden ympäristön shell-skriptistä.

```
export PATH=/usr/java8_64/jre/bin:$PATH
arg1=/prima/pr006/hrmherate/hrm.properties
dir=/prima/pr006/hrmherate/dist
jar_name=PrimaRestClient.jar
java -jar $dir/$jar_name $arg1
```

Kuva 18. heratteet.sh komentoriviltä ajettava käynnistysskripti

5. PALVELURAJAPINTA TEKNINEN TOTEUTUS (HRM-PALVELU)

Palvelurajapintoja voidaan toteuttaa monilla eri tekniikoilla riippuen omista taidoista tai mieltymyksistä, mutta joskus myös ajoympäristö voi asettaa rajoituksia. Yhteistä kaikille on kuitenkin jonkinlainen rajapintakuvaus. Siitä löytyy tiedot, miten palvelurajapintaan liitytään ja mitä palvelusta saadaan ulos. Rajapintakuvauksista on kerrottu kappaleessa 4.

Tässä työssä palvelu on toteutettu osana isompaa henkilöstöhallinnon järjestelmäkokonaisuutta, mikä on antanut jonkinlaiset kehykset toiminnalle ja valitulle toteutustavalle.

HRM-järjestelmä on siis kokonaisuus henkilöstöhallinnon ratkaisuja ja poissaoloherätteet-palvelu on yksi palvelurajapinta muiden joukossa, joka on liitetty tuohon kokonaisuuteen. Toteutetussa palvelussa olennaisena osana on esimiestiedon liittäminen työntekijätietoihin, jolloin palveluun saapuvat poissaolotiedot voidaan välittää esimiehille, jotta varhaisen välittämisen proseduuri pääsee tarvittaessa alkuun.

5.1 Palvelun ohjelmointi

Rajapinta on toteutettu REST-tyyppisenä palveluna ja se on ohjelmoitu java8:lla. Sovelluspalvelimena on Apachen Tomcat versio 8 ja asiakasympäristöt ovat IBM unix käyttöjärjestelmiä tai Linux palvelimia. Tietokannat ovat Oraclen eri versioita 9-11.

Toteutus on tehty NetBeans IDE 8.2 versiolla ja projekti on dynaaminen web-projekti, joka käännetään Apachen Maven-työkalulla. Maven helpottaa kehitystyötä vähänkin isommissa projekteissa, joissa on paljon ulkoisia kirjastoja käytössä, kun riippuvuudet muihin haetaan automaattisesti verkosta. Maven käyttää pom.xml-tiedostoa, jossa kerrotaan käytettävät kirjastot ja versiot (The Apache Software Foundation, 2019).

5.2 Rajapinnan näkyminen ulospäin

Palvelurajapinnoissa tärkeää on kertoa julkaisuvaiheessa millä osoitteella rajapintaa kutsutaan ja mitä palveluja se tarjoaa. Käytännössä tämä on joku web-osoite (endpoint), joka käyttää tavallisesti http(s) protokollaa.

Kehitys ja demovaiheessa on helpointa asentaa palvelu paikalliselle koneelle jollekin web-sovelluspalvelimelle. Tässä työssä kehitys ja demoympäristö on asennettu Apachen Tomcat:lle, joka on asennettu windows:n paikalliseksi palveluksi porttiin 8680. Tuolle palvelimelle on sitten asennettu HrmService.war (java web archive paketti), josta syntyy osoitteen (endpoint) ensimmäinen osa.

http://localhost:8680/HrmService/rest/json/persons

Kuva 19. Palvelun osoite demoympäristössä.

Osoitteen /rest osa tulee palvelun web.xml tiedostosta jonne palvelu on konfiguroitu seuraavasti

```
<servlet>
        <servlet-name>jersey-serlvet</servlet-name>
        <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>jersey-serlvet</servlet-name>
        <url-pattern>/rest/*</url-pattern>
</servlet-mapping>

<filter>
         <filter-name>AuthenticationFilter</filter-name>
         <filter-class>com.cgi.hrmservice.RestAuthenticationFilter</filter-class>
</filter>
<filter-mapping>
         <filter-name>AuthenticationFilter</filter-name>
         <url-pattern>/rest/*</url-pattern>
 </filter-mapping>
```

Kuva 20. Palvelun konfigurointi web.xml tiedostossa

/json osa tulee jo itse endpoint luokasta, joka on annotaatiolla osoitettu kuuluvan osaksi polkua.

```
@Path("/json")
public class JSONService {
```

Kuva 21. Javaluokan annotointi osaksi polkua

Osoitteen /persons osa tulee luokan metodista, joka ottaa palvelupyynnön vastaan ja käsittelee sen. Metodin annotaatioilla kerrotaan, minkä tyyppistä sanomaa tai palvelupyyntöä käsitellään ja mitä tuotetaan paluusanomana.

```
@POST
@Path("/persons")
@Consumes(MediaType.APPLICATION_JSON)
@Produces("application/json")
 public Object personData(@Context HttpHeaders httpHeaders,
                          InitiationAddRequest myrequest) throws SQLException{
```

Kuva 22. Metodin annotointi osaksi polkua

5.3 Käyttäjän tunnistus

Palvelurajapinta käyttää tunnistautumiseen Basic autentikointia, jossa tunnistautumisavain on muotoa ”Basic username:password”. Tuossa jälkimmäinen osa pitää olla Base64 koodattuna. Esimerkiksi jos tunnistautumisavain olisi selkokielisenä ”Basic admin:admin”, niin se tulisi välittää kyselysanomassa muodossa ”Basic YWRtaW46YWRtaW4=”

Käyttäjän tunnistusta varten sovellukseen on rakennettu erillinen filtteriluokka, joka ottaa kiinni kaikki /rest/ polussa olevat palvelut, jolloin tunnistautuminen tehdään yhdessä ja samassa luokassa ja varsinaiset sovelluslуokat voivat keskittyä varsinaiseen toimintaan. (Murach, 2014, p. 600)

Kappaleessa 6.2 ja kuvassa 20 olevassa web.xml-tiedostossa on tuo filtteriluokka mainittuna. Se on käytännössä luokka, joka toteuttaa javax.servlet.Filter rajapinnan. Filtteriluokasta ja sen toiminnasta kerrotaan enemmän demovideossa. Filtterin koodaus on toteutettu servlet-kontekstin mukaisena standarditoteutuksena (Kulandai, 2015).

5.4 Palvelurajapinnan toiminta

Ohjelmointi on tehty Java8:lla ja palvelu on Servlet-tyyppinen Java-sovellus, jossa metodeista on annotaatioiden avulla muodostettu erityyppisiä palveluita, joita kutsutaan Http post- tai Http get-tyyppisesti. Kappaleessa 6.2 olevassa kuvassa 22 on kerrottu, että metodi lukee sisään (Consumes) JSON-sanomarakennetta ja tuottaa ulos (Produces) vastaavassa muodossa olevaa dataa.

Parametreissa toisena parametrina saadaan JSON-sanomaa vastaava luokkarakenne, joka on muodostettu JSON-skeemasta ja siitä ylin luokka.

Palvelun toteutus käyttää hyödykseen javax.ws.rs-api:a (java api for RESTful web services), muita ohjelmointirajapintoja ovat javax.mail sähköpostiviestien lähetystä varten ja oraclen jdbc-ajurit tietokantaoperaatioita varten.

5.5 Esimerkkisanoma

Alla on vielä pieni esimerkkisanoma kyselystä, joka noudattaa kappaleessa 4.1 mainittua skeemaa, josta on sitten muodostettu luokkamalli sanoman käsittelyyn.

```
{
 "document": {"id": "initiation"},
 "items": [{
                "id": "absenceInitiation",
                "personId": " ",
                "ssn": false,
                "subject": "05092017_Minnan_uusi_testi",
                "message": "Henkilöllä POLO PEPPI on 15 kalenteripäivää tai yli sairauspoissaoloja, huo-mioi Aktiivisen tuen
                toimintamallin mukaiset toimet."
                }, {
                "id": "absenceInitiation",
                "personId": "10323",
                "ssn": false,
                "subject": "05092018_Minnan_uusi_testi",
                 "message": "Henkilöllä POISSAOLO PAULA on 15 kalenteripäivää tai yli sairauspoissaoloja, huomioi Aktiivisen
                toimintamallin mukaiset toimet."
          }]
}
```


Kuva 23. Toinen esimerkki kyselysanomasta

Yllä olevasta sanomasta nähdään, että kun ensimmäisestä alkiosta puuttuu personId, niin validoinnin pitää palauttaa virhesanoma tuohon riviin liittyen.

5.6 JSON sanoman sisäänluku ja validointi

Alla olevassa kuvassa 24 luetaan toisena parametrina tuleva JSON-objekti ja käydään sanomarakenne läpi. Items lista sisältää käytännössä kuvassa 23 näkyviä ”items” alkioita

```
List<InitiationAddRequest.Item> items = myrequest.getItems();

items.forEach(item -> {
                String personId =  item.getPersonId();
```

Kuva 24. Listan alkiot (items) JSON oliomallista haettuna

Jokainen listan alkio käydään läpi ja tarkistetaan vastaanottavassa järjestelmässä, onko henkilötunnus tyhjä, löytyykö henkilö järjestelmästä, onko hänelle esimiestietoja olemassa ja löytyykö esimiehelle sähköpostiosoitetta, johon herätetietoa lähetetään. Validointi tehdään lähinnä tuota personId-tietoa apuna käyttäen, joka yksilöi henkilön.

Viestin validoinnissa käytetään paluuviestiin liittyen seuraavia vakioituja merkkijonoja (Statusviesti). Tuossa on alusta otettu pois tyyppimäärittelyt jotka koodista löytyvät.

```
MISSING_ID_FOR_GIVEN_SSN = "Couldn't find matching person id for given ssn";
INVALID_PERSON_ID = "Person id is invalid";
MANAGER_EMAIL_MISSING = "Manager email missing";
MANAGER_USERCODE_MISSING = "Manager user code missing";
MESSAGE_OK = "OK";
```

Kuva 25. Tekstivakiot paluuviestiä varten

Statuksen ilmaiseva koodi on yksi seuraavista. Esim. 200 kun ei huomautettavaa ja listan alkio saatu onnistuneesti luettua, talletettua ja sähköposti lähetettyä. Virhetilanteessa käytetään koodia 400.

```
HttpStatus.SC_BAD_REQUEST  = 400
HttpStatus.SC_OK   = 200
```

Kuva 26. Paluuviestin virhekoodit

Palvelu hakee HRM-järjestelmästä esimiestiedot sen tarjoaman tietokantarajapinnan kautta.

5.7 Tietojen tallennus, sähköpostin lähetys

HRM-järjestelmä tallentaa saapuneet viestit omaan järjestelmäänsä tietokantarajapinnan metodeilla. Tarkistetaan henkilön olemassaolo ensin kannasta ja sitten tallennetaan henkilöön kohdistuva viesti kannan tauluun. Tämä menee normaalisti Javan Jdbc-rajapinnan kautta.

Sähköpostin lähetys tehdään JavaMail-ohjelmointirajapinnan (tutorialspoint, 2019) avulla ja sähköpostin lähetystä varten on luotu sovelluksen käyttöön omat gmail-tunnukset

```
public SLLMail(String to, String subject, String text)
{
   Properties props = new Properties();
   props.put("mail.smtp.host", "smtp.gmail.com");
   props.put("mail.smtp.socketFactory.port", "465");
   props.put("mail.smtp.socketFactory.class","javax.net.ssl.SSLSocketFactory");
   props.put("mail.smtp.auth", "true");
   props.put("mail.smtp.port", "465");

   try{
      Session session = Session.getDefaultInstance(props, new javax.mail.Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication() {
            return new PasswordAuthentication(username,password);
            }
            });

      Message msg = new MimeMessage(session);

      msg.setFrom(new InternetAddress(username));
      msg.setRecipients(Message.RecipientType.TO, InternetAddress.parse(to));
      msg.setSubject(subject);
      msg.setText(text);
      msg.setSentDate(new Date());
      Transport.send(msg);
      System.out.println("Message sent.");
   }
   catch (MessagingException e){
      System.out.println("Error, cause: " + e);
   }
}
```

Kuva 27. Sähköpostin lähettäminen

5.8 Vastaussanoman muodostus

Asiakassovellukselle lähetettävän paluuviestin sisältö kirjoitetaan vastaavan tyyppiseen listarakenteeseen kuin mitä kyselysanoman kanssa toimittiin.

```
List<InitiationAddResponse.Item> returnItems = new ArrayList<>();
```

Kuva 28. Paluuviestin lista-luokka

Listarakenteeseen luetaan käytännössä kyselysanoma täydennettynä jokaisen listan jäsenen (ao. kuvassa item) statuskoodilla ja statusviestillä, jotta asiakassovellus tietää mitkä on onnistuneesti luettu sisään.

Tämä tehdään samassa silmukassa, jossa kyselysanoma luetaan läpi.

```
//Create REST response item
InitiationAddResponse.Item returnItem = new InitiationAddResponse.Item();
returnItem.setId(initiationId);
returnItem.setPersonId(personId);
returnItem.setSsn(true);
returnItem.setSubject(subject);
returnItem.setMessage(message);
returnItem.setStatus(itemStatus);
returnItem.setStatusMessage(itemStatusMessage);
returnItems.add(returnItem);
```

Kuva 29. Paluuviestin listalle kirjoitettava jäsenluokka (item)

Alla on vastaussanoman muodostus. Metodi siis palauttaa objektin kyselevälle asiakassovellukselle

```
Map<String, Object> jsonMap = new LinkedHashMap<>();
jsonMap.put("data", returnItems);
jsonMap.put("status", HttpStatus.SC_OK);
jsonMap.put("message",MESSAGE_OK);

return jsonMap;
}
```

Kuva 30. Palautettava JSON viesti objektimallina

5.9 Palvelusovelluksen paketointi tuotantoympäristöön

Kun palvelusovelluksen kasaus tai käännös tuottaa Javan war-tiedoston, niin asennus tuotantoympäristöön on varsin helppoa. Ei tarvitse muuta kuin kopioida tuo tiedosto webserverin webapps-kansioon ja konfiguroida sovelluksen vaatimat tietokantaosoitteet ja mahdolliset sähköpostipalvelimet vastaamaan tuotantoympäristön asetuksia.

Tuotantoympäristöt ovat tätä kirjoitettaessa kahdella asiakkaalla Unix- tai Linux-palvelimella ja niille on asennettu Tomcat8-sovelluspalvelin ja Javasta käytössä on versio 8. Tietokantojen yhteystiedot on kirjoitettu erilliselle resurssitiedostolle.

6. DEMOVIDEOT

Työstä on tehty demovideoita, joissa kuvataan paremmin toteutettu ohjelmisto.

Demovideoita on kolme kappaletta. Client.mp4 (HRM-asiakas) ja service.mp4 (HRM-palvelu) käyvät läpi asiakas- ja palvelusovelluksen koodia ja itse sovellusta. Esimerkki yhdestä ajokerrasta on demossa ajot.mp4, joka sisältää operatiivisen osuuden eli alkutilanteen kuvauksen, demoajon ja tulokset.

Videot ovat ladattavissa ao. linkkien kautta. Katselua varten ne kannattaa ladata ja tallentaa omalle kiintolevylle ensin.

- client.mp4
- service.mp4
- ajot.mp4

7. KÄYTETYISTÄ TEKNIIKOISTA JA VÄLINEISTÄ

Työssä käytetyt välineet ohjelmistokehitykseen ja sovelluspalvelimiin liittyen ovat perinteikkäitä ja joidenkin arvioiden mukaan jo vanhaa tekniikkaa. Mielestäni Java on kuitenkin edelleen hyvä ohjelmointikieli sovelluskehitykseen, kun se integroidaan johonkin Java-sovelluskehykseen kuten esim. Spring-framework. Lisäksi Eclipse ja NetBeans tukevat hyvin Javaa ja selainpohjaista kehitystyötä hyvien virheenjäljitys mahdollisuuksien myötä. Suora kommunikointi versionhallintajärjestelmiin on myös mahdollista. Integrointi sovelluspalvelimiin onnistuu vaivattomasti ja kehitystyössä automaattinen käännös on toteutettavissa, kun koodia muutetaan.

Samat asiat olisi voinut varmasti toteuttaa nykysuuntauksen mukaisesti skriptauskielillä käyttämällä sellaisia kehitystyökaluja kuten node.js tai Python tai vaikkapa konfiguroimalla Apachen Camel palveluväyläratkaisua.

8. YHTEENVETO

Valitsin aiheeksi ohjelmointityön käytännönläheisyyden vuoksi. Ongelmasta tai ideasta pyritään rakentamaan toimiva kokonaisuus. Erilaiset integraatiot järjestelmien välillä ovat myös kiinnostavia, siksi valitsin integraatioratkaisun.

Tähän työhön liittyvät kaikki työvaiheet olivat yllättävän selkeitä määritellä ja toteuttaa kun siirrettävä aineisto oli tiedossa. Vaikka toteutettavia asioita oli lukuisa joukko, niin integraatiotestauksessa ei jouduttu enää kovinkaan paljon korjaamaan ohjelmia ja tekemään iterointia.

Rajapintakuvauksen tekeminen oli helppoa. Kun siirrettävää tietoa ei ole paljon niin rakenne muodostui aika yksinkertaiseksi ja siitä tuli ihan toimiva, kun listarakenteen läpikäynti on helppo toteuttaa objektimallin avulla.

Palvelun toteutus ja testaus olivat suuritöisin osuus tässä työssä. Java tuttuna työkaluna helpotti työtä, mutta silti aikaa tuhraantui varsin paljon tähänkin osioon. CygWin- ja Curl -työkalujen asennukset (Oracle, 2015) helpottivat palvelun yksikkötestaamista. Palveluun tunnistautumisen toteutus tuotti jonkin verran lisäpohdintaa, kun vaihtoehtoisia menetelmiä on paljon. Filtterin toteutus ja konfigurointi palvelun edustaksi syntyi viimeisenä osana tätä kokonaisuutta.

Asiakassovelluksen toteutustyössä minulle uusi asia oli JSON-objektimallin käyttö viestin välityksessä.

Työ oli mielenkiintoinen ja myös opettavainen. Ongelmia ja haasteita oli tuotantoympäristöissä jonkin verran lähinnä tietoliikenneyhteyksissä, mutta ne saatiin ratkaistua ja tuotannossa olevissa asiakasympäristöissä ohjelmistot toimivat tällä hetkellä varsin hyvin.

LÄHTEET

The Apache Software Foundation (2019). A*pache Maven Project*. Haettu 13.2.2019 osoitteesta https://maven.apache.org/

Baeldung, E. (18. April 2019). *Rest vs websockets*. Haettu 17.1.2019 osoitteesta Baeldung: https://www.baeldung.com/rest-vs-websockets

Wikipedia (2019). Basic access authentication. Haettu 16.3.2019 osoitteesta en.wikipedia.org/: https://en.wikipedia.org/wiki/Basic_access_authentication

Hewitt, E. (2009). Java SOA CookBook. Teoksessa *Java SOA CookBook* (s. 714). O'Reilly.

Jenkov, Jakob (2016). Java JSON Tutorial. Haettu 1.3.2019 osoitteesta tutorials.jenkov.com: http://tutorials.jenkov.com/java-json/index.html#jackson

tutorialspoint (2019). JavaMail API Tutorial. Haettu 22.3.2019 osoitteesta www.tutorialspoint.com: https://www.tutorialspoint.com/javamail_api/

Joel Murach, M. U. (2014). Murach's Java Servlets and JSP, 3rd Edition (Murach: Training & Reference). In M. U. Joel Murach, *Murach's Java Servlets and JSP, 3rd Edition (Murach: Training & Reference)* (p. 744). Murach.

Alexander, Alvin (2019) Linux crontab examples (every x minutes or hours). Haettu 12.3.2019 osoitteesta alvinalexander.com: https://alvinalexander.com/linux/unix-linux-crontab-every-minute-hour-day-syntax

Littlejohn, J. (2012-2018). *jsonschema2pojo*. Haettu 25.2.2019 osoitteesta jsonschema2pojo: http://www.jsonschema2pojo.org/

Oracle. (2015). *Installing cURL on Cygwin on Windows*. Haettu 2.2.2019 osoitteesta https://www.oracle.com/webfolder/technetwork/tutorials/obe/cloud/objectstorage/installing_cURL/installing_cURL_on_Cygwin_on_Windows.html

Kulandai, Joe (2015). RESTful Services HTTP basic Authentication. Haettu 1.4.2019 osoitteesta javapapers.com: https://javapapers.com/web-service/restful-services-http-basic-authentication/