SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
LUONNONTIETEIDEN ALA

OPINNÄYTETYÖ

Työntekijän perehdyttämisjärjestelmä TyPe:n käytettävyyden parantaminen viestijonolla ja videosoittimella

TEKIJÄ/T: Robert Brandt

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Luonnontieteiden ala			
Koulutusohjelma/Tutkinto-ohjelma Tietotekniikan koulutusohjelma			
Työn tekijä(t) Robert Brandt			
Työn nimi Työntekijän perehdyttämisjärjestelmä TyPen käytettävyyden parantaminen viestijonolla ja videosoittimella			
Päiväys	21.9.2020	Sivumäärä/Liitteet	40 / 3
Ohjaaja(t) Jussi Koistinen, Veijo Pitkänen			
Toimeksiantaja/Yhteistyökumppani(t) Pinja Digital Oy			

Tiivistelmä

Tämän opinnäytetyön tavoitteena oli etsiä tapoja parantaa työntekijän perehdyttämisjärjestelmä TyPe:n käytettävyyttä ja toiminnallisuutta viestijonotekniikkaa käyttäen. Opinnäytetyö toteutettiin yhteistyössä Pinjan kanssa. Viestijonotekniikan tuli olla mahdollisimman yleiskäyttöinen, jotta toteutusta voitaisiin jatkossa käyttää muualla yrityksen palveluissa.

TyPe:n käytettävyyttä ja toiminnallisuutta heikentävä komponentti oli tiedossa opinnäytetyön alussa. Viestijonosovelluksen valinnan ja konfiguraation jälkeen työ keskittyi komponentissa ilmenneiden ongelmien ratkaisuun. Viestijono toteutettiin käyttäen PHP -ohjelmointikieltä Symfonyn sovelluskehyksen ja kirjastojen puitteissa. Viestijonoksi valikoitui oletusasetuksin käytettävä RabbitMQ -instanssi, jota käytettiin Symfonyn rajapinnan kautta.

Opinnäytetyön tuloksena saatiin selvitettyä viestijonotekniikoita ja niiden toteuttamisperiaatteita. Viestijonosovelluksen demon toteutus ei onnistunut TyPe:n ongelmakomponentissa käytetyn rekursiivisen koodaustekniikan vuoksi. Toteutusta jälkikäteen tarkastellessa todettiin, että TyPe:n komponentissa käytetyn rekursion purkamiseen, tai sen puitteissa toimimiseen, olisi tarvittu järjestelmään erikoistuneen asiantuntijan apua. Selvitystyön perusteella kuitenkin havaittiin, että vaikka viestijono olisi muuttanut front-end toteutuksen asynkroniseksi ja parantanut käyttökokemusta, se ei olisi ratkaissut ongelmakomponentin rekursiivisuuden tuottamaa ongelmaa.

Viestijonon lisäksi TyPeen vaadittiin toimiva videosoitin. Eräälle TyPen suuremmista käyttäjistä videoiden esitys työturvallisuuskursseilla oli tärkeää, ja videosoittimen puute heikensi TyPen markkina-arvoa. Soitinprojektin tuloksena saatiin TyPeen lisättyä toimiva videosoitin, joka toistaa YouTube-linkkejä ja pystyy esittämään n kappaletta videoita jokaisessa kurssikysymyksessä.

Avainsanat
Viestijono, Message Queue, RabbitMQ, Symfony, Videosoitin

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Natural Sciences			
Degree Programme Degree Programme in Information Technology			
Author(s) Robert Brandt			
Title of Thesis Improving the usability of employee orientation software TyPe through message queue and video player			
Date	September 21, 2020	Pages/Appendices	40 / 3
Supervisor(s)			
Client Organisation /Partners Pinja Digital Oy			

Abstract

This bachelor's thesis was made in cooperation with Pinja. The goal of this thesis was to research and develop a message queue solution to improve the usability and user experience of the employee orientation service TyPe.

The specific component hindering the speed and usability of the TyPe service was known in the beginning of this project. After researching message queue services and techniques, the coding efforts were focused on resolving the slowdowns caused by the component. The message queue was made using the PHP coding language within Symfony framework utilizing its libraries and components. After researching different message queue services, RabbitMQ was selected for use in this project.

The result of this thesis was a more comprehensive understanding of message queue techniques and their implementation. The development of a functional message queue demo was not achieved due to the recursive coding technique used in the problem component. When evaluating the outcome of this thesis, it was noted that to undo or to work around the recursion used in the problem component, assistance of an expert specialized in the backend solution of TyPe would have been required. As a result of the initial research into message queues, however, it was understood that while the asynchronicity of the message queue would have improved the user experience of the end user, it would not have resolved the underlying issue stemming from the implementation of the recursive component.

In addition to the message queue, a working video player was required for TyPe. One of the largest clients of TyPe valued using videos in their employee orientation courses, and the lack of a video player lessened the market value of TyPe. As a result of the video player project, a video player capable of playing and displaying multiple YouTube videos was added to TyPe.

Keywords
Message Queue, RabbitMQ, Symfony, Video Player

SISÄLTÖ

1 JOHDANTO

1.1 Työn tilaaja

Työn tilaajana oli Pinja Digital Oy. Pinja-konserni on moniosaaja teollisuuden ja digitalisaation saralla. Pinjan toimialoihin kuuluvat työkoneet ja liikenteen sovellukset, energia ja kiertotalous, hyvinvointi- ja terveysteknologia, meriteollisuus, eri palveluyritykset sekä puunjalostusteollisuus.

Pinja-konserni perustettiin vuonna 1990. Tänä päivänä Pinja työllistää noin 500 henkilöä Suomessa yli kymmenellä eri toimipisteellä. Pinja myös tarjoaa eri palveluitaan yli kolmessakymmenessä eri maassa tekniikan, hyvinvoinnin, automaation, suunnittelun ja tietotekniikan saroilla. Pinjan liikevaihto oli 40 miljoonaa euroa vuonna 2019. (Pinja Group Oy, ei pvm)

1.2 Työn tarkoitus

Opinnäytetyön tarkoituksena on tutkia viestijonopalveluja, valita yksi jonokandidaatti ja soveltaa sitä Pinjan TyPe-palvelun käyttäjäkokemuksen parantamiseen. TyPe käsittelee saamansa kutsut synkronisesti, mikä voi johtaa käyttäjäkohtaisen palvelun hidastumiseen tai kaatumiseen, jos käyttäjän pyytämä toiminne on jostain syystä liian monimutkainen käsiteltäväksi.

Synkronisuus johtaa myös siihen, että käyttäjän kokemus palvelun laadusta kärsii, mikäli ohjelma jumiutuu siksi aikaa, kun käyttäjän pyytämää palvelua käsitellään. Viestijonopalvelun avulla on tarkoitus muokata yhtä TyPe:n elementtiä asynkroniseksi, jonka odotetaan parantavan palvelun toimivuutta ja responsiivisuutta.

1.3 Työn tavoite

Opinnäytetyön tavoitteena on tuottaa yksinkertainen viestijonodemo, joka toimii Pinjan TyPe-palvelun taustalla ja käsittelee palveluun saapuvia pyyntöjä asynkronisesti. Aluksi, ennen varsinaisen työn aloittamista, tutkitaan eri viestijonopalveluita ja -menetelmiä. Eri palveluista valitaan yksi kandidaatti, jota sovelletaan viestijonototeutuksessa.

Viestijonon tavoitteena on osittain vapauttaa front-end toteutus taustalla pyörivistä toiminteista, jotta palvelu ei jumiudu liian pitkäksi aikaa. Käyttäjän pyyntö voidaan sitten käsitellä hallitusti taustalla. Viestijonon tulisi myös olla mahdollisimman yleiskäyttöinen, jotta sitä voidaan jatkossa soveltaa muissa Pinjan palveluissa, jotka hyötyisivät asynkronisuudesta.

Lisäksi opinnäytetyössä lisättiin TyPeen tuki videoille ja luotiin videosoitin, joka pystyy toistamaan n kappaletta YouTube -videoita.

1.4 Lyhenteet ja määritelmät

1.4.1 TyPe

TyPe, eli työntekijän perehdytysjärjestelmä on web-pohjainen ohjelmisto, jonka avulla yritykset ja konsernit voivat valvoa työntekijöidensä, sekä alihankkijoiden työntekijöiden koulutus- ja osaamistasoa. TyPen tarkoituksena on tehostaa osaamisen hallintaa ja valvontaa, sekä parantaa työturvallisuutta. (Pinja Group Oy, ei pvm)

1.4.2 Viestijono

Viestijonot ovat palveluja, jotka mahdollistavat eri palveluiden tai palvelun osien välisen asynkronisen viestinnän, eli datan siirron eri prosessien, palvelujen, palvelimien ja ohjelmien välillä. Viestijonossa viestijonopalveluun saapuvat viestit asetetaan joidenkin määriteltyjen kriteerien perusteella jonoon tai jonoihin odottamaan käsittelyä. (Amazon, ei pvm)

Viestijono on yksinkertainen taulu tai kokoelma, jossa viestit erotetaan ja jaetaan määritellysti viestien otsikoiden ja attribuuttien avulla. Asynkronisuus on erittäin toivottava ominaisuus monissa ohjelmissa, jotta käyttäjän ei tarvitse jäädä odottamaan ohjelman etenemistä ja pystyy jatkamaan muuta työskentelyä. Asynkroniset viestijonot auttavat myös sietämään virheitä säilyttämällä ja lähettämällä viestit uudelleen, kunnes ne voidaan käsitellä. (Johansson, What is message queuing?, 2019)

1.4.3 RabbitMQ

RabbitMQ on avoimen lähdekoodin viestijonopalvelu. RabbitMQ tarjoaa tuen useille viestijonoprotokollille, viestin saapumiskuittaukset, useita vaihtotyyppejä sekä tuen useille koodauskielille. (Pivotal, ei pvm)

1.4.4 PHP

PHP: Hypertext Preprocessor on yleisesti käytetty koodikieli, jota käytetään erityisesti web-kehityksessä HTML:n kanssa. Käyttäjä ei ole suorassa vuorovaikutuksessa PHP:n kanssa, vaan koodi suoritetaan palvelimella ennen kuin sen tuloste välitetään käyttäjän selaimelle. (PHP.net, ei pvm)

1.4.5 Symfony

Symfony on avoin PHP -ohjelmistojen runkopalvelu, joka sisältää valmiita ja räätälöitäviä komponentteja ja kirjastoja. Symfonyä käytetään nopeuttamaan PHP -ohjelmistojen luontia tarjoamalla uudelleenkäytettäviä osia ja komponentteja. (Symfony.com, ei pvm)

1.4.6 Angular

Angular on web-ohjelmistojen -sivujen suunnittelun runkopalvelu ja kehitysympäristö. Angular sisältää komponentteja ja kirjastoja Symfonyn tapaan, mutta on tarkoitettu sivustojen *front end* toteutuksia varten. (Angular Docs, ei pvm)

1.4.7 Docker

Docker on ohjelmistoalusta, joka mahdollistaa standardisoitujen konttien luomisen ja pystyttämisen. Kun kontti on kerran luotu, Docker tallentaa kaikki työvaiheet, jotka toistetaan samassa järjestyksessä aina, kun kontti täytyy luoda uudestaan, kuten esimerkiksi toiselle työasemalle. (Amazon, ei pvm)

Docker on avoimen lähdekoodin projekti, joka toimii Linux ja Windows -alustoilla. Dockeria voidaan käyttää paikallisesti tai pilvipalvelussa. Docker eroaa virtuaalikoneesta siten, että virtuaalikoneet ovat erillisiä työasemia, ja Dockerin kontit käsittävät työnasemaan asennettuja ohjelmistoja tai ohjelmistokokonaisuuksia. (Microsoft, 2018)

1.4.8 Kontti, container

Container, eli kontti, on standardisoitu tietokoneohjelman tai -ohjelmiston yksikkö. Kontissa ohjelma tai ohjelmisto, sekä kaikki sen riippuvuudet paketoidaan yhdeksi kokonaisuudeksi, joka voidaan myöhemmin asentaa mihin tahansa muuhun työasemaan, jossa kaikki vaadittavat toimenpiteet suoritetaan aina samalla tavalla. Täten joka asennuskerralla saadaan samalla lailla toimiva ympäristö tai ohjelma, riippumatta siitä minne tai milloin se asennetaan. (Docker Inc., ei pvm)

1.4.9 Dockerfile

Dockerfile on Dockerin käyttämä tekstitiedosto, joka sisältää käskyt, parametrit, asennettavat ohjelmat ja riippuvuudet, joiden pohjalta Docker tuottaa käytettävän levykuvan. (Butler, 2015)

1.4.10 Ubuntu MATE

Ubuntu MATE on Debian-pohjainen avoimen lähdekoodin Linux-jakelu, jossa on sisäänrakennettu helppokäyttöinen graafinen työpöytä ja käyttöliittymä. (Ubuntu MATE Team, ei pvm)

1.4.11 Git ja Github

Git on avoimen lähdekoodin komentorivipohjainen versionhallintajärjestelmä ja GitHub on pilvipohjainen varasto. GitHubia käytetään Git:n kautta, tai jonkin muun kolmannen osapuolen ohjelmaan tai palveluun integroidun Git -toiminnallisuuden kautta. Sovelluskehittäjät voivat tallentaa työnsä GitHubiin, josta muut henkilöt, kuten muut kehitystiimin jäsenet, voivat ladata tallennetun koodin. Versionhallintaominaisuuksiensa avulla voidaan havaita ja ratkaista ristiriitaisuuksia koodissa, sekä luoda ja yhdistää eriäviä sovellushaaroja. (Brown, 2019)

1.4.12 Oracle VM VirtualBox

Oracle VM VirtualBox on Oracle -yhtymän kehittämä avoimen lähdekoodin x86 ja AMD64/Intel64 arkkitehtuurien kanssa yhteensopiva virtualisointituote. VirtualBoxia tarjotaan yksityis- sekä yrityskäyttöön. (Oracle Corporation, ei pvm)

2 TYÖN ALOITUS

2.1 Aloituspalaveri

Ennen työn aloittamista käytiin aloituspalaveri esimiehen osoittamien asiantuntijoiden kanssa. Palaverissa keskusteliin työn luonteesta, tavoitteista, rajauksista, tarpeesta ja selvennettiin viestijonon periaatetta. Ennen varsinaisen selvitystyön aloittamista, aloituspalaverin jälkeen, tutustuttiin viestijonon periaatteisiin sekä käyttötapauksiin ja -tarkoituksiin.

Aloituspalaverissa sovittiin seuraavat asiat:

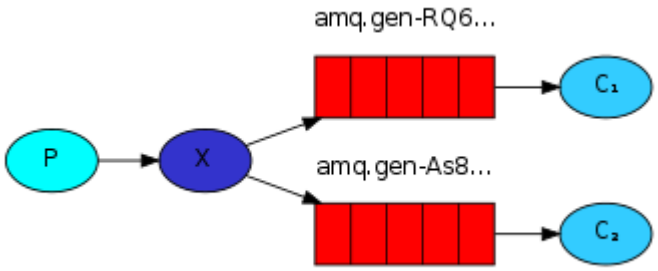
- Vertaillaan viestijonopalveluita, joista valitaan yksi
- Viestijonopalvelun ja sen riippuvuuksien on oltava ilmaisia
- Selvitetään valitun viestijonopalvelun viestijonotyypit ja tutustutaan valitun palvelun konfiguraatiomalleihin
- Tutustutaan Kubernetes -palveluun
- Tutustutaan Docker -palveluun ja sen käyttämien konttien periaatteisiin
- Projektin etenemistä seurataan kahden viikon välein pidettävissä tilannekatsauspalavereissa

2.2 Viestijonon teoriaa

Viestijono, tai viestin välityspalvelu, toimii välikätenä kahden tai useamman ohjelman, ohjelmiston tai muun palvelukokonaisuuden kesken. Viestijonot ovat luonteeltaan asynkronisia, eli ohjelman toiminteen suoritus ei lukitse ohjelman käyttäjän rajapintaa, vaan toiminne suoritetaan taustalla ja ohjelma sittemmin ilmoittaa käyttäjälle lopputuloksesta, kun toiminteen suoritus päättyy. Tämä tarkoittaa sitä, että palvelun tai ohjelmistojen tuottajien on otettava huomioon viestien välittämisessä tapahtuva viive. (James, 2013)

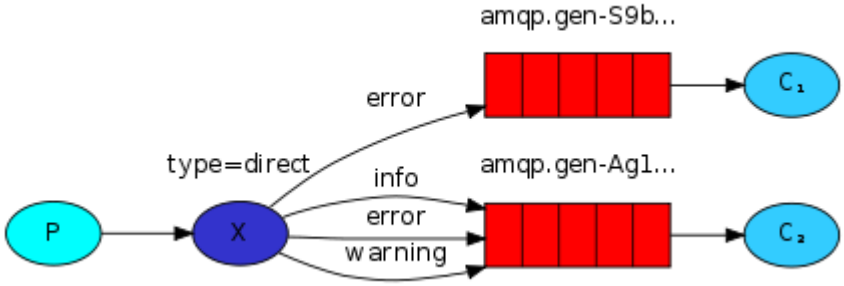
Viestijonoja käytetään useista syistä; oikein käytettyinä ne tehostavat suorituskykyä ja luotettavuutta, sekä tarjoavat mahdollisuuden dynaamiseen skaalaamiseen. Viestien tuottajat ja päätepisteet eivät ole välittömässä vuorovaikutuksessa toistensa kanssa, vaan käsittelevät viestejä viestijonon kautta. Viestien tuottajat voivat siis lisätä viestejä jonoon odottamatta päätepisteiden käsittelijöitä, ja käsittelijät työstävät viestejä vain silloin kun niitä on tarjolla. Oikein rakennettuna viestijono vähentää merkittävästi järjestelmän komponenttien tyhjäkäyntiä ja joutoaikaa. Viestijonoissa odottava data on yleensä kestävää, jolloin tieto ei häviä joidenkin järjestelmien osien sammuessa tai kaatuessa. Kun järjestelmä paloitellaan pienemmiksi, toisistaan riippumattomiksi osasiksi, parantuu myös järjestelmän virheensietokyky ja mahdollisista virheistä toipuminen on helpompaa, kun yksittäinen virhe ei välttämättä seisauta koko järjestelmää, vaan vaikuttaa vain yhden komponentin toimintaan. (Amazon, ei pvm)

Viestijonoja on muutamaa eri tyyppiä. **Point-to-point** -tyylinen viestijono on nimensä mukaisesti kahden eri pisteen (lähettäjän ja vastaanottajan) välinen jonotyyppi. Point-to-point tekniikkaa käytettäessä lähettävän osapuolen on tiedettävä mihin päätepisteeseen viestin on lähdettävä. Riippuen toteutustavasta ja viestijonosovelluksesta, lähettäjän voi olla tarpeen tietää jonon nimi, jonon managerin nimi tai muuta jonon yksilöivää tietoa. (IBM, 2020)



KUVA 1. Esimerkki Point-to-Point -viestijonosta (Pivotal, ei pvm)

Publish/Subscribe -tyyli puolestaan levittää lähetettävän viestin kaikille viestistä kiinnostuneille osapuolille, joita voi olla yksi tai useampi, tai ei yhtään. Viestit kopioituvat siis kaikkiin jonoihin, joiden kriteerit viestin tunniste tai otsikko täyttää. Kriteereitä voi olla yksi tai useampi, ja viesti voi kopioitua useaan jonoon samanaikaisesti. (IBM, 2020)



KUVA 2. Esimerkki Publisher/Subscribe -viestijonosta. (Pivotal, ei pvm)

Joissain viestijonosovelluksissa on myös **Fanout** -tyyli, joka levittää viestin kaikkiin viestijonoihin, joilla on viestiin jokin kytkös välittämättä tunnisteista tai kriteereistä. Fanout -tyyliä käytetään lähinnä joko erittäin simppeleissä viestijonoissa, joissa ei tarvita viestien erottelua, tai vain sellaisissa viesteissä, joiden halutaan tavoittavat kaikki vastaanottajat.

2.3 Viestijononsovellusten vertailu

Tässä kappaleessa esitellään viestijonosovelluksia, joita harkittiin käytettäväksi projektissa. Viestijonoja vertailtiin niiden dokumenttien, saatavilla olevien teknisten tietojen ja aikaisemmin mainittujen kriteerien perusteella.

2.3.1 Azure Queue

Azure Queue Storage on Microsoftin tarjoama palvelu, joka säilöö viestejä ja mahdollistaa asynkronisen viestinnän pilvessä muiden palvelujen ja komponenttien kesken. Azuren viestijonoihin pääsee käsiksi mistä tahansa, josta voidaan tehdä autentikoituja HTTP ja HTTPS -kutsuja. Azure Queue vaatii Azure Storage käyttäjätunnuksen. Viestijonopalvelua käytetään ja konfiguroidaan Microsoftin muiden tuotteiden kautta, mikä tekee siitä epäsopivan Linux -distribuutiota käytettäessä. (Microsoft Corporation, 2020)

2.3.2 Amazon Simple Queue Service

Amazon Simple Queue Service (SQS) on Amazonin tarjoama viestijonopalvelu, joka on ilmainen aina miljoonaan transaktioon kuukaudessa asti, jonka jälkeen Amazon veloittaa palvelun käytöstä. SQS tarjoaa kaksi erilaista viestijonotyyppiä: vaihtoehtoina ovat perusviestijono sekä First-In-First-Out (FIFO) -viestijono. Amazon tarjoaa muutaman kurssin SQS:n käyttöä varten. Amazonin muita web-palveluita voidaan myös soveltaa SQS:ä käytettäessä. (Amazon AWS, ei pvm)

2.3.3 RabbitMQ

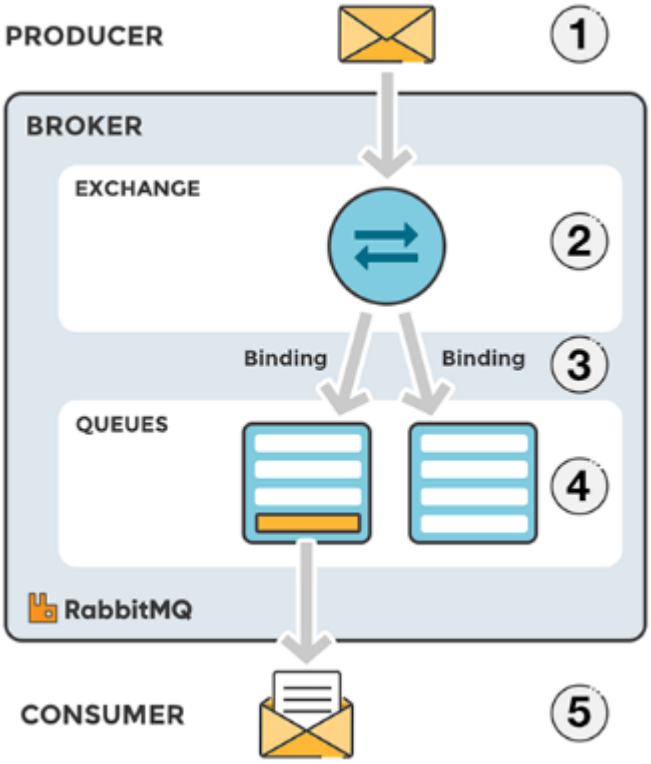
RabbitMQ on avoimen lähdekoodin viestivälittäjä, joka on yksi suosituimmista ja käytetyimmistä niin pienissä yrityksissä kuin laajemmissa yhtiöissä. RabbitMQ:lla on pieni jalanjälki ja se on valmiina käytettäväksi pilvi- ja yritysympäristöissä. RabbitMQ:lla on myös useita liitännäisiä, jotka mahdollistavat sen räätälöinnin tarpeen mukaan. RabbitMQ:n ominaisuuksiin kuuluu myös web-pohjainen käyttöliittymä, josta jonojen suorituskykyä ja kuormaa, sekä viestiliikennettä voidaan valvoa. RabbitMQ on ilmainen käyttää ja kehittää, mutta Pivotal Software tarjoaa myös kaupallisia vaihtoehtoja ja lisäominaisuuksia. RabbitMQ:lle on myös tarjolla useita alkeiskursseja, joissa käydään läpi RabbitMQ:n ominaisuuksia, sekä annetaan esimerkkejä viestijonon konfiguraatiosta. (Pivotal, ei pvm)

2.3.4 Google Cloud Queue

Google Cloud Tasks on palvelu, joka tarjoaa tehtäväjonoja, jotka voivat sisältää useita eri mikropalveluja ja pilvessä suoritettavia tehtäviä. Google Tasks tarjoaa komentokehotteen sekä oman käyttöliittymänsä, jonka kautta Google Taskin jonoja voidaan hallita. Cloud Tasks on ilmainen ensimmäiseen miljoonaan operaatioon asti, jonka jälkeen Google veloittaa $0.40 dollaria miljoonalta operaatiolta. (Google, ei pvm)

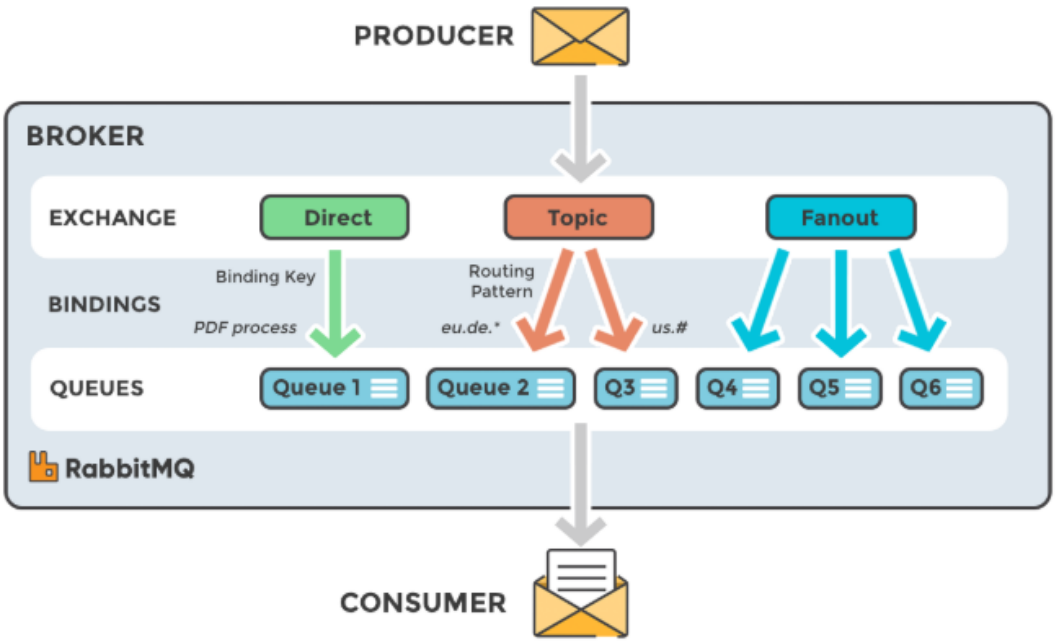
2.4 Viestijonosovelluksen valinta

Lopulliseksi viestijonosovellukseksi valittiin RabbitMQ. RabbitMQ:n valintaperusteina olivat sen helppokäyttöisyys, pieni jalanjälki ja konfiguraation monimuotoisuus, kurssimateriaalien runsaus sekä suurimpana vaikuttajana yrityksessä jo olemassa oleva kokemus RabbitMQ:sta. Pinja Digital Oy:n muissa projekteissa oli käytetty RabbitMQ:n viestijonosovellusta, joka mahdollisti opiskelun ja tiedonhaun muiden projektien sekä RabbitMQ:n kokemusta omaavien henkilöiden kautta.



KUVA 3. RabbitMQ:n vaihteet, sidokset ja reititykset (Johansson, 2019)

RabbitMQ:lla on neljää eri vaihdetyyppiä: **Direct**, **Topic, Headers** ja **Fanout**. Jokainen vaiheista toimii hieman eri tavalla, mutta kaikilla on sama tarkoitus: ohjata tuotettu viesti oikeaan jonoon, josta viestin käsittelijä sittemmin hakee sen. Vaihteen tyyppi vaikuttaa viestin tunnisteen tyyppiin ja siihen, kuinka viesti ohjautuu. Viestin tuottajat eivät siis sijoita viestejä suoraan viestijonoon, vaan viestien lopullinen sijoitus ja ohjaus tehdään vaihteiden kautta.



KUVA 4. RabbitMQ:n vaihdetyypit ja jonorakenne. (Johansson, 2019)

Kirjoittamassaan RabbitMQ:n alkeiskurssissa Johansson kuvailee vaihdetyyppejä seuraavasti:

- **Direct:** Viesti ohjataan vain niihin viestijonoihin, joiden sidosavain on täsmälleen sama kuin viestin reititysavain. Esimerkiksi viesti, jonka reititysavain on *pdfprocess* ohjautuu vain siihen viestijonoon, jonka sidosavain on myös *pdfprocess*.
- **Topic:** Vaihde suorittaa jokerivertailun reititysavaimen ja reititysmallin kesken, ja ohjaa viestin sopivaan jonoon. [Esimerkiksi viesti, jonka reititysavain on *posti.kuopio.jynkkä* ohjautuisi kaikkiin viestijonoihin, joiden malli vastaa *posti.kuopio.jynkkä* reititysavainta, muttei jonoihin, joiden malli olisi siitä eriävä, kuten *posti.kuopio.neulamäki*]
- **Headers:** Vaihde käyttää viestin otsikkotietoja viestin reitittämiseen. [Esimerkiksi viesti, jonka otsikkona on *{format: pdf, type: report}* ohjautuisi jonoihin, jotka ottavat vastaan *pdf* -formaattisia viestejä ja/tai *report* -tyyppisiä viestejä.]
- **Fanout:** Vaihde reitittää viestit kaikille vaihteeseen kytketyille viestijonoille.

Vaihteille ja jonoille voidaan myös määrittää useita eri parametreja, kuten *durable*, *auto-delete* ja *temporary*. Durable -parametriset vaihteet tallennetaan muistiin, jolloin ne eivät katoa palvelimen kaatuessa tai käynnistyessä uudelleen, ja tuhoutuvat vain silloin kun niiden käsketään poistuvan. Temporary -tyyppiset vaihteet puolestaan olevat olemassa vain niin kauan kuin RabbitMQ on käynnissä, eivätkä tallennu palvelimelle. Auto-delete -parametri puolestaan aiheuttaa vaihteen tuhoutumisen, kun viimeinen vaihteeseen kytketty objekti – jono, viestin tuottaja tai muu entiteetti – irrotetaan vaihteesta. (Johansson, 2019)

3.1 RabbitMQ:n alkeiskurssit

RabbitMQ tarjoaa verkkosivuillaan (www.rabbitmq.com) seitsemän alkeiskurssia. Kurssien esimerkkitehtävät on käännetty Python, Java, Ruby, PHP, C#, JavaScript, Go, Elixir, Objective-C, Swift ja Spring AMQP -koodikielille. RabbitMQ ei siis ole spesifinen yksittäisille koodikielille, vaan tuki löytyy monelle käytetyistä kielistä. Alkeiskursseilla esitellään viestijonon periaate ja tärkeimmät käsitteet yksinkertaisin esimerkein ja vertauskuvin. Vaikka esimerkkejä ei suoraan käytetty työssä, periaatteiden sisäistäminen helpotti työn laajuuden kartoittamista ja paransi valmiutta konfiguroida viestijonoa. Ennen työn aloittamista alkeiskursseja käytiin läpi ensimmäiset viisi kappaletta.

Kurssi 1: ”Hello World!”

Ensimmäisellä kurssilla esitellään viestijonon tärkeimmät komponentit ja selitetään kuinka ne pääpiirtettäin toimivat. Komponentteja ovat: **producer** eli viestin tuottaja, **queue** eli viestijono itse, sekä viestin käsittelijä **consumer**. Riippuen koodikielestä, RabbitMQ saattaa tarvita erillisen kirjaston komponentteja toimiakseen. Työssä käytettiin PHP -koodikieltä, joten alkeiskurssien tarkastelu keskittyi PHP -esimerkkeihin.

Kurssi 2: Työjonot

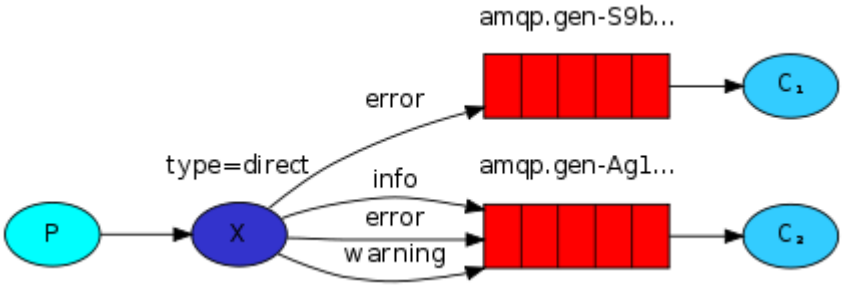
Seuraavalla kurssilla keskitytään viestijonon, tai työjonon, keskimmäiseen ideaan: viestijonon tarkoituksena on välttää raskaiden tehtävien tekeminen välittömästi synkronisessa järjestyksessä, joka voisi lukkiuttaa ohjelman suorituksen ajaksi. TyPe:n ongelma oli juuri vastaavanlainen: intensiivinen prosessi jumiutti ohjelman, kunnes suoritettava tehtävä aikakatkaistiin tai suoritus kaatui. Viestijonon toinen tehtävä on jakaa suoritettavia tehtäviä ja tasoittaa suoritettavien tehtävien kuormaa käsittelijöiden kesken.

Kurssi 3: Publish/Subscribe -malli

Kolmannella kurssilla viestijonoa laajennetaan yksittäisestä säikeestä useampaan viestin käsittelijään ja esitellään viestijono vaihde. Esimerkkinä käytetään yksinkertaisia lokitusjärjestelmää, joka lähettää lokitiedon sisältävän viestin vaihteelle, josta viestit kulkevat käsittelijöihin. Kurssia käydessä läpi, lisättiin vaihtoehtoinen lisäkriteeri TyPe:n viestijonosovellukselle: jos demo saadaan toimivaksi tavoiteajassa, voidaan viestijonoon lisätä myös lokituspalvelu. Lokitietojen perusteella vikatilanteiden selvittäminen ja sovelluksen jatkokehitys olisi tulevaisuudessa helpompaa.

Kurssi 4: Reititys

Neljännessä kurssissa määritellään tarkemmin mihin jonoihin viestien on lähdettävä tiettyjen parametrien perusteella. Näin on mahdollista erotella esimerkiksi lokiviestit niiden kriittisyyden perusteella. Kappaleessa käsitellään myös vaihteen ja jonojen sidoksia, reitityksen ja reititysavaimien suhdetta, ja opetetaan tekemään useita rinnakkaisia sidoksia useisiin jonoihin.



KUVA 5. Esimerkki viestijonon reitityksistä (Pivotal)

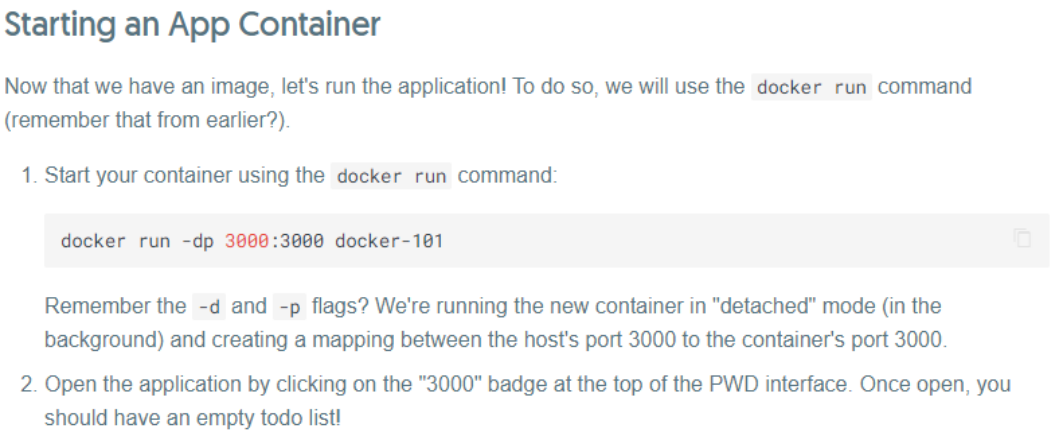
Kurssi 5: Viestien aiheet

Viidennessä, ja viimeisessä viestijonosovellukseen liittyvässä, kurssissa käsitellään **Topic Exchange** -vaihteita, eli aiheperusteisia vaiheita. Aiemmissa esimerkeissä käytetyt vaihteet lähettivät viestit suoraan haluttuihin jonoihin, mutta suora lähetys voi olla joissain tapauksissa liian rajoittava. Aiheperusteiset vaihteet lähettävät viestin työjonoon usean kriteerin perusteella, jolloin viestijonon konfiguraatiosta voi tehdä joustavamman ja geneerisemmän. Aiheperusteiset vaihteet käsittelevät viestit muodossa *<parametri1>.<parametri2>.<parametri3>*, jossa parametrejä on yksi tai useampi ja parametrit voidaan korvata jokerimerkein. Tätä vaihetyyppiä voitaisiin käyttää TyPe:n viestijonon jatkokehityksessä erottelemaan viestit niiden tyyppien ja tuottajien perusteella.

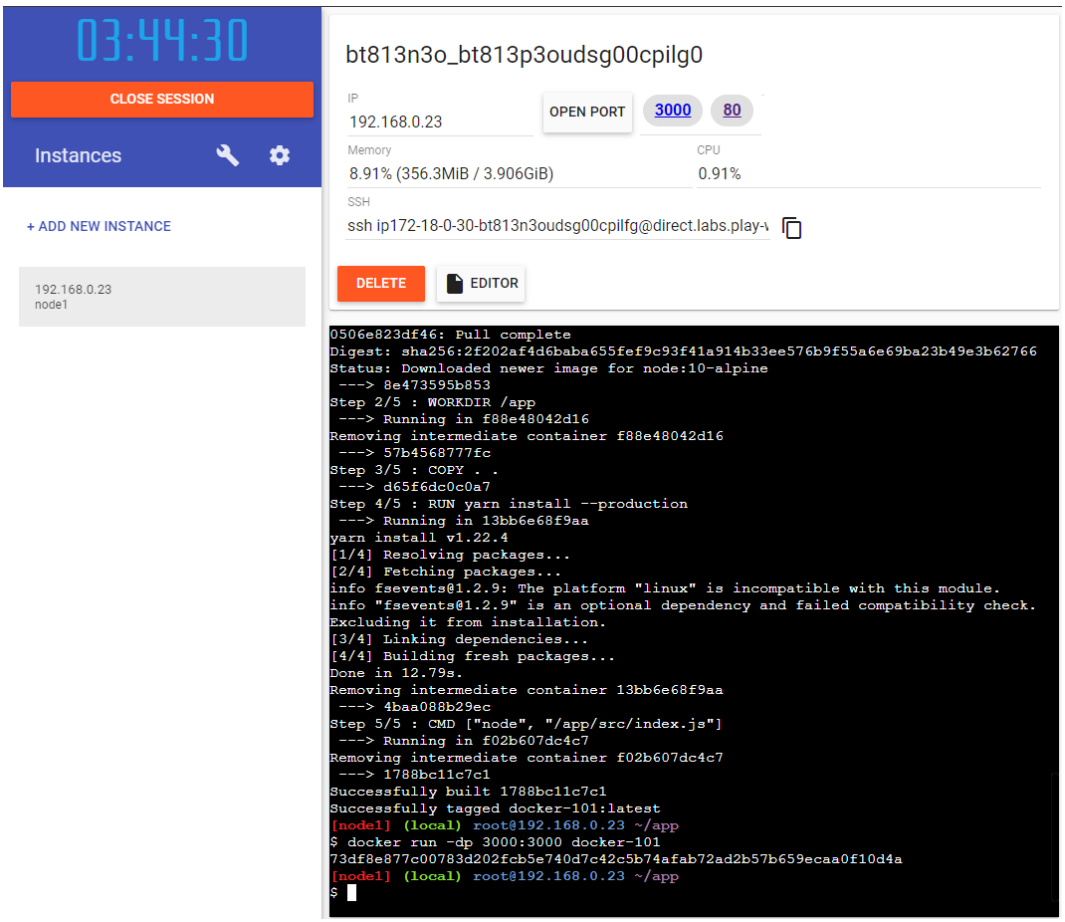
4 DOCKER: TUTUSTUMINEN JA YMPÄRISTÖN ASENNUS

4.1 Dockeriin tutustuminen

Docker tarjoaa interaktiivisia kursseja, joiden kautta perehtyä Dockerin toimintaan, periaatteeseen sekä sen yleisiin komentoihin ja käyttötarkoituksiin. Dockerin *Play with Docker* vaatii vain käyttäjätunnuksen luonnin, jonka vahvistuksen jälkeen käyttäjä voi kirjautua Dockerin pilvipalveluun opiskelemaan.



KUVA 6. Esimerkki alkeiskurssin sisällöstä (Docker, Inc.)



KUVA 7. Tilannekuva Docker Labs -palvelusta (Docker, Inc.)

Interaktiivisia kursseja on tarjolla useita, joiden kautta tutustua Dockerin toimintaan. Ennen työn aloittamista käytiin läpi tärkeimmät alkeiskurssit, joiden kautta selvitettiin tärkeimmät Dockerin ominaisuudet ja käyttötapaukset.

4.2 Ympäristön asennus

Alustana työlle toimi virtualisoitu Ubuntu Mate. Työaseman virtualisointiin käytettiin *Oracle VM VirtualBox* -virtualisointialustaa. Ennen ympäristön varsinaista asentamista tarvittiin salausavain Pinjan GitHub varastoon, josta ladattiin muutama asennukseen vaadittava tiedosto, muun muassa projektin Dockerfile. Tämän jälkeen ympäristön asentaminen suoritettiin yksinkertaisella Docker -komennolla, joka latasi ja asensi ympäristön ja vaadittavat riippuvuudet automaattisesti.

5 VIESTIJONON KONFIGURAATIO

Ympäristön asennuksen onnistuttua syvennyttiin RabbitMQ:n konfiguraatioon. RabbitMQ:n versiosta riippuen konfiguraatiotiedostot ovat joko .conf tai .config päätteisiä; uudemmat versiot käyttävät .conf tiedostopäätettä. Työssä keskityttiin perusominaisuuksien konfigurointiin, kuten ympäristömuuttujiin ja tiedostopolkujen määrittämiseen RabbitMQ:n sivustolla annettujen konfiguraatio-ohjeiden mukaisesti. Konfiguraatiotiedostoja luotiin kolme kappaletta:

- env.conf
- rabbitmq.conf
- definitions.json

Seuraavissa kappaleissa tarkastellaan konfiguraatiotiedostojen sisältöä sekä sisällettyjen konfiguraatioparametrien funktiota. Konfiguraatiotiedostot *rabbitmq.conf* ja *definitions.json*, jotka ovat tämän opinnäytetyöraportin liitteinä, esitetään aihealueittain tiedostojen pituuden vuoksi.

5.1 env.conf

```
# Overrides node name
NODENAME = rabbit@dev

# General config file location.
CONFIG_FILE = /etc/rabbitmq/rabbitmq.conf

# Environment config file location.
CONF_ENV_FILE = /etc/rabbitmq/env.conf
```

Tuotetuista konfiguraatiotiedostoista yksinkertaisin on *env.conf*. Se sisälsi vain kolme parametria, joiden tarkoituksena oli:

1. Nimetä ja yksilöidä RabbitMQ:n instanssi
2. Nimetä konfiguraatiotiedosto ja osoittaa sen tiedostopolku
3. Nimetä ympäristömuuttujien konfiguraatiotiedosto ja osoittaa sen tiedostopolku

Tiedoston tehtiin tarkoituksella mahdollisimman minimalistisesti, jotta ylimääräiseltä ja mahdollisesti tarpeettomalta konfiguraatiolta vältyttäisiin. Vain tärkeimpiä tietoja yritettiin listata *env.conf* tiedostoon, seuraten RabbitMQ:n konfiguraatio-oppaiden esimerkkejä.

5.2 rabbitmq.conf

RabbitMQ:n pääkonfiguraatiotiedosto on *rabbitmq.conf*, jonka avulla RabbitMQ:n palvelinta ja lisäosia voidaan konfiguroida. Se sisältää tärkeimmät asetukset, muun muassa mitä portteja RabbitMQ kuuntelee, kuinka montaa asiakasta voidaan palvella samanaikaisesti, sekä muuttujan virheiden lokitukselle. Tässä kappaleessa käydään läpi tuotettu *rabbitmq.conf* tiedosto aihealueittain.

```
## Default sockets
listeners.tcp.default = 5672
# listeners.ssl.default = 5671

## Number of Erlang processes that will accept connections for the TCP and TLS listeners.
# num_acceptors.tcp = 10
# num_acceptors.ssl = 10

## Maximum amount of time allowed for the AMQP 0-9-1 and AMQP 1.0 handshake
## (performed after socket connection and TLS handshake) to complete, in milliseconds.
##
handshake_timeout = 10000
```

Ensimmäiseksi konfiguraatiotiedostossa määriteltiin yleisimpiä verkkoyhteyden muuttujia. RabbitMQ:n kuuntelija osoitettiin kuuntelemaan TCP-porttia 5672. Tiedostoon määritettiin myös kuuntelija SSL-portille 5671 siltä varalta, että jossain vaiheessa ilmenisi tarve käyttää salattua yhteyttä, tosin asetus on tiedostossa toistaiseksi kommentoituna sen tarpeen puutteen vuoksi.

Seuraavaksi määriteltiin rajoitus samaan aikaan käynnissä olevien TCP- ja SSL-yhteydet hyväksyvien prosessien lukumäärälle. Rajoittamaton määrä prosesseja saattaisi jossain tapauksessa aiheuttaa järjestelmän hidastumista tai kaatumisen. Asetukset ovat toistaiseksi kommentoituna kehitysympäristössä. Viimeisenä verkkoyhteysmuuttujana määriteltiin maksimiaika AMQP-protokollan (Advanced Message Queuing Protocol) kättelylle, jotta estettäisiin prosessin roikkumaan jääminen kättelyn jostain syystä seisahtuessa.

```
## Memory threshold at which the flow control is triggered.
vm_memory_high_watermark.relative = 0.5

## Disk free space limit of the partition on which RabbitMQ is storing data.
## When available disk space falls below this limit, flow control is triggered.
disk_free_limit.relative = 3.0
```

Seuraava aihealue käsittelee RabbitMQ:n muistin ja levyn käyttöä. Ensimmäinen parametri käynnistää RabbitMQ:n virtauksenhallinnan. Virtauksenhallinta rajoittaa nopeutta sellaisilta yhteyksiltä, jotka julkaisevat dataa nopeammin kuin mitä palvelimen jonot voivat käsitellä, tai jos RabbitMQ:n palvelimen RAM-muistin tai levytilan ylittää sallitun rajan. Konfiguraatiotiedostossa virtauksenhallinnan käynnistävä muistin raja on 50% RAM-muistista ja levytilan raja on kolminkertainen RAM-muistin kokonaismäärään nähden.

```
## Level of errors logged.
## error = only errors are logged
## warning = only errors and warning are logged
## info = errors, warnings and informational messages are logged
## debug = errors, warnings, informational messages and debugging messages are
logged
# log.file = false
log.dir = /etc/rabbitmq/logs
log.file = rabbit.log
log.file.level = debug

## Rotate when the file reaches 10 MiB
log.file.rotation.size = 10485760

## Keep up to 5 archived log files in addition to the current one
log.file.rotation.count = 5

## Rotate the file based on date.
## Rotate file every month.
log.file.rotation.date = $M0
```

Seuraavat parametrit käsittelevät lokitiedoston ominaisuuksia. Ensimmäisenä määritetään lokitiedoston sijainti sekä tiedoston nimi. Lokituksen tasoksi on valittu *debug*, jolloin RabbitMQ lokittaa virheet, varoitukset, tiedotukset sekä testaukseen liittyvät viestit. Lopullisessa tuotantoversiossa *error* tai *warning* lokitustasot olisivat riittäviä. Seuraavaksi määritetään lokitiedostojen rotaatiota käsittelevät parametrit. Rotaatio, tai kierto, tarkoittaa lokitiedoston pakkaamista ja arkistointia, ja uuden lokitiedoston aloittamista. RabbitMQ konfiguroitiin kierittämään lokitiedosto, mikäli lokitiedoston koko ylittää kymmenen mebitavua, sekä kuukauden vaihtuessa. RabbitMQ myös säilyttää viisi viimeisintä lokitiedostoa arkistoituna.

```
## Maximum permissible number of channels to negotiate with clients, not including special channel 0 used in the protocol.
channel_max = 2047

## Frequency of heartbeat.
heartbeat = 60

## Virtual host to create when RabbitMQ creates a new database from scratch.
default_vhost = /

## Settings not suitable for production purposes.
default_user = dev
default_pass = dev
default_user_tags.administrator = true

## Loopback can access through localhost.
loopback_users.dev= true
default_permissions.configure = .*
default_permissions.read = .*
default_permissions.write = .*

## Broker, exchange, and queue definitions file location.
load_definitions = /etc/rabbitmq/definitions.json
```

Konfiguraatiotiedoston lopussa määritettiin maksimimäärä kanaville, joita RabbitMQ saa käyttää asiakkaiden kanssa viestimisessä, RabbitMQ:n palvelimen syke ja määritetään virtuaalinen isäntä. RabbitMQ:n sykeparametri aikakatkaisee yhteydet, jotka eivät anna signaalia minuutin aikana, jotta roikkumaan jäävät yhteydet eivät tuki palvelimen kanavia. Virtuaalinen isäntä on tässä tapauksessa jätetty oletukseksi.

Seuraavaksi asetettiin palvelimen oletusarvoinen käyttäjätunnus ja salasana, ja annettiin kyseisellä käyttäjälle ylläpito-oikeudet. Kuten tiedostossa mainittiin, nämä asetukset eivät ole tuotantokelpoisia, ja niiden tarkoituksena oli nopeuttaa testausta ja kehitystyötä. Seuraavaksi kehityskäyttäjälle annetaan pääsy palvelimelle localhost -yhteyden kautta, ja asetetaan tiedostojen käyttöoikeudet mahdollisimman laajasti kehitystä varten. Nämäkään asetukset eivät ole tuotantokelpoisia.

Viimeiseksi määritetään *definitions* -tiedoston sijainti.

5.3 definitions.json

Viimeinen käsiteltävä konfiguraatiotiedosto sisältää RabbitMQ:n käyttämien vaihteiden, viestijonojen ja sidosten määritykset sekä parametrit. RabbitMQ luo vaihteet, jonot sekä sidokset määritystiedoston mukaisesti ensimmäisen käynnistyskerran yhteydessä. Esiteltävä tiedosto on siistitty ja paloiteltu aihealueisiin. Vaihteet, jonot ja sidokset on tehty raportin liitteenä olevan opinnäytetyötä kuvaavan vuokaavion mukaisesti (liite 1).

5.3.1 Exchanges

```
"name": "acservice",
      "vhost": "/",
      "type": "topics",
      "durable": true,
      "auto_delete": false,
      "internal": false,
      "arguments": {}

"name": "dead_letter_exchange",
      "vhost": "/",
      "type": "direct",
      "durable": true,
      "auto_delete": false,
      "internal": false,
      "arguments": {}
```

Palvelimelle määritettiin kaksi vaihdetta, *acservice* sekä *dead_letter_exchange*. AC Service -vaihde on tarkoitettu pääasialliseksi vaihteeksi, jonka kautta ongelmatilanteita aiheuttavan komponentin viestiliikenne vastaisuudessa kulkisi. Dead Letter Exchange -vaihde (DLE) puolestaan käsittelee viestit, jotka eivät jostain syystä ole päässeet perille, tai joita ei olla pystytty käsittelemään.

AC Service -vaihteen tyypiksi määriteltiin reititysavaimia ja -malleja käyttävä *topics*, joka antaa mahdollisuuden tarvittaessa määritellä useita erilaisia reititysmalleja esimerkiksi eri käyttäjä- tai viestityyppien välillä. DLE -vaihde toimii sidosavainta käyttävällä *direct* -mallilla, koska vain palvelimen sisäinen viestintä ottaa yhteyttä DLE:een. Kumpikin jono on määritetty kestäväksi ja automaattinen poisto on kytketty päältä, jotta jonot eivät katoaisi palvelimen muistista uudelleenkäynnistyksen yhteydessä.

5.3.2 Queues

```
"name": "acservice",
    "vhost": "/",
    "durable": true,
    "auto_delete": false,
    "arguments": {}

"name": "dead_letter_queue",
    "vhost": "/",
    "durable": true,
    "auto_delete": false,
    "arguments": {}
```

RabbitMQ:lle luodaan kaksi jonoa, jotka vastaavat edellä luotuja AC Service ja Dead Letter Exchange -vaihteita. Kummankin isäntäpalvelun osoite määritetään palvelun juureen. Kuten vaiheita määrittäessä, jonoista tehdään kestäviä ja ne merkitään automaattisen poiston ulkopuolisiksi. Muita argumentteja jonoille ei annettu.

5.3.3 Bindings

```
"source": "acservice",
    "vhost": "/",
    "destination": "acservice",
    "destination_type": "queue",
    "routing_key": "*",
    "arguments": {}

"source": "dead_letter_exchange",
    "vhost": "/",
    "destination": "dead_letter_queue",
    "destination_type": "queue",
    "routing_key": "*",
    "arguments": {}
```

Lopuksi annetaan jonojen ja vaihteiden sidosparametrit. Kohteiksi asetetaan AC Service ja Dead Letter Exchange -viestijonot. Reititysavaimeksi annettiin jokerimerkki, jolloin jonot hyväksyvät kaikki niihin saapuvat viestit. Käytännössä tämä ei vaikuta DLE:n toimintaan, koska vaihteen välitystyyppinä on **direct**. Myöhemmin tuotannossa AC Servicelle voidaan määrittää tarkemmat reititysparametrit, mutta yhtä jonoa testatessa parametritykselle ei ollut tarvetta.

6 TYÖN TOTEUTUS

Tämä kappale käsittelee projektin osioiden toteutusta. Työ toteutettiin käyttäen PHPStorm ja Visual Studio Code -kehitysympäristöjä, ja viestijonosovelluksen osalta myös virtuaaliympäristöä hyväksi käyttäen. RabbitMQ -viestijonon ohella projektin aikana tehtiin myös videosoitin, joka lisättiin TyPen työturvallisuuskursseihin. Työn tilaajan pyynnöstä tuotettua koodiaineistoa ei esitellä tässä raportissa.

6.1 RabbitMQ:n toteutus

RabbitMQ:n konfiguraation valmistuttua siirryttiin koontiversion rakentamiseen ja varsinaisen työn toteuttamiseen. Dockerin avulla koontiversion luonti on helppoa, koska Docker käyttää annettuja konfiguraatioparametrejä ja asentaa automaattisesti kaiken tarvittavan. Koontiversion luonnissa kuitenkin ilmeni ongelmia, jotka estivät järjestelmän pystyttämisen ja projektin saattamisen eteenpäin.

```
rabbitmq    |
rabbitmq    | BOOT FAILED
rabbitmq    | ===========
rabbitmq    |
rabbitmq    | Config file generation failed:
rabbitmq    | 12:20:37.829 [error] load_definitions invalid, file doesn't exist or isn't readable
rabbitmq    | 12:20:37.838 [error] log.dir invalid, Cannot create file in dir
rabbitmq    | 12:20:37.848 [error] Error generating configuration in phase validation
rabbitmq    | 12:20:37.848 [error] load_definitions invalid, file doesn't exist or isn't readable
rabbitmq    | 12:20:37.848 [error] log.dir invalid, Cannot create file in dir
rabbitmq    | In case the setting comes from a plugin, make sure that the plugin is enabled.
rabbitmq    | Alternatively remove the setting from the config.
rabbitmq    |
rabbitmq    | {"init terminating in do_boot",generate_config_file}
rabbitmq    | init terminating in do_boot (generate_config_file)
rabbitmq    |
rabbitmq    | Crash dump is being written to: /var/log/rabbitmq/erl_crash.dump...rabbitmq exited
with code 0
```

Ensimmäinen ilmenevä virhe oli yllä kuvattu käynnistysvirhe. Vaikka aiemmin kuvatuissa konfiguraatiotiedostoissa oli määritetty *definitions.json* sekä lokitiedoston sijainti, Docker ei pystynyt käsittelemään annettuja parametreja. Virhettä yritettiin korjata muuttamalla tiedoston nimeä ja formaattia uudemmasta *.conf* -mallista vanhempaan *.config* -malliin, kansiorakennetta muuttamalla sekä konfiguraatiota muuttamalla. Ratkaisuyrityksistä huolimatta Docker jatkoi käynnistysvaiheessa kaatuilua.

Kun vian syy ei selvinnyt ja työhön käytettävissä oleva aika alkoi huveta, työtä valvova Pinjan työntekijä ehdotti *env.conf* -tiedoston poistamista. Kun kyseinen konfiguraatiotiedosto oli poistettu, Docker selvisi ensimmäisestä virheestä ja jatkoi koontiversion luontia.

```
Creating rabbitmq ...
Recreating typebackend_php_1 ...
Recreating typebackend_php_1
Recreating typebackend_php_1 ... done
Recreating typebackend_nginx_1 ...
Recreating typebackend_nginx_1 ... done
Attaching to typebackend_mysql_1, rabbitmq, typebackend_php_1, typebackend_nginx_1
rabbitmq    | sed: cannot rename /etc/rabbitmq/sedYsZiz9: Device or resource busy
php_1       | Loading composer repositories with package information
php_1       | Installing dependencies (including require-dev) from lock file
php_1       | Warning: The lock file is not up to date with the latest changes in composer.json. You
may be getting outdated dependencies. It is recommended that you run `composer update` or
`composer update <package name>`.
php_1       | Nothing to install or update
php_1       | Package phpunit/phpunit-mock-objects is abandoned, you should avoid using it. No
replacement was suggested.
php_1       | Package symfony/lts is abandoned, you should avoid using it. Use symfony/flex in-
stead.
php_1       | Generating autoload files
php_1       | ocramius/package-versions:  Generating version class...
php_1       | ocramius/package-versions: ...done generating version class
rabbitmq exited with code 4
php_1       | Executing script requirements-checker [OK]
```

Koontiversion luonti ei kuitenkaan onnistunut odotetusti. Docker ei enää kaatunut hallitsemattomasti, vaan loi muut työhön tarvittavat instanssit. Itse RabbitMQ:n palvelinta luodessa ilmeni seuraava ongelma. Virheilmoituksen mukaan RabbitMQ:n instanssi oli varattu, eikä pystynyt käsittelemään pyyntöä instanssin uudelleennimeämiseksi. Prosessi kuitenkin jatkui normaalisti, kunnes Docker sammui koodilla 4. Uudeksi ongelmaksi kehittyi virhekoodin tarkoituksen selvitys, sillä virhekoodia 4 ei ollut mainittu virhekoodilistassa. Työssä avustavat Pinjan työntekijät ehdottivat virheen syyksi Dockerille riittämättömiä käyttöoikeuksia.

Ongelmanratkaisuun kului suuri osa työhön jäljellä olevasta ajasta, mutta virheen syy ei selvinnyt. Dockerille annettiin korotetut oikeudet, kansiorakennetta yksinkertaistettiin ja konfiguraatiota muokattiin, mutta virhekoodista 4 ei päästy eteenpäin senhetkisellä konfiguraatiolla. Virheen syy ei myöskään selvinnyt. Viimeisenä mahdollisuutena eräs Pinjan työntekijä ehdotti konfiguraatiotiedos-

tojen hylkäämistä ja RabbitMQ:n ajamista täysin tehdasasetuksin, kuten eräässä toisessa Pinjan projektissa oli tehty. Ehdotus soti aikaisemmin saatuja konfiguraatio-ohjeita vastaan, joiden pohjalta RabbitMQ:n konfiguraatio oli luotu, mutta kyseinen työntekijä vakuutti konfiguraation olevan turhaa, koska viestijonon välikätenä voidaan käyttää Symfonyä.

Konfiguraatiotiedostot poistamalla virhekoodi poistui ja Docker viimein pystyi rakentamaan RabbitMQ:n instanssin. Sillä hetkellä projektiin oli kuitenkin käytettävissä enää noin viikko aikaa, joten projekti tulisi jäämään kehitysasteelle. Edellä mainittu työntekijä myös ehdotti toisten projektin viestijonototeutukseen liittyvien tiedostojen käyttämistä pohjana tämän viestijonoprojektin kehityksen nopeuttamiseksi. Esimerkkinä viestijonon käytöstä on alla oleva otos salasanan palauttamistoiminnon osasta, jota käytettiin pohjana viestijonosovellusta kehittäessä.

```
public function requestResetPassword(RequestPasswordResetDto $requestPasswordResetDto):
void
   {
      $user = $this->userResource->findOneBy(['email' => $requestPasswordResetDto-
>getEmail()]);
      if ($user !== null) {
         $this->messageBus->dispatch(new RequestPasswordResetAction($user->getId()));
      } else {
         $this->logger->warning(
            'Requested password reset for non existent account "' . $requestPasswordResetDto-
>getEmail() . '"'
         );
      }
   }
```

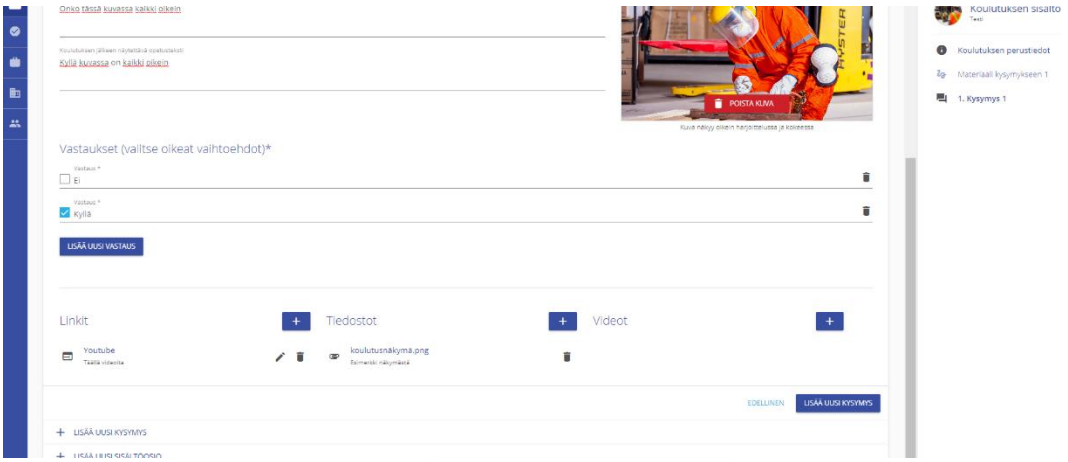
Projektiin varattu aika ei kuitenkaan enää riittänyt kuin toisesta projektista kopioitujen tiedostojen muokkaamiseen viestijonosovellusta varten. Ilman muuta konfiguraatiota RabbitMQ saatiin myös pystytettyä, tosin ilman spesifikaatiossa kuvattuja vaiheita ja jonoja. Uuden Symfonyyn pohjautuvan paradigman opiskeluun, viestijonon spesifikaation muokkaamiseen sen mukaiseksi ja viestijonon rakennus uuden spesifikaation pohjalta olisi kulunut huomattavasti enemmän aikaa kuin mitä projektissa oli jäljellä. Tästä johtuen projektin viestijono-osio jäi keskeneräiseksi.

6.2 Videosoittimen toteutus

Viestijonon ohella TyPen kursseihin tarvittiin tuki videosoittimelle. Pyyntö tuli eräältä suurelta TyPen asiakkaalta, ja vaatimukset soittimelle olivat yksinkertaiset: soittimen tulisi pystyä soittamaan videoita ainakin YouTube -linkkien kautta. Suunnittelupalaverissa videosoittimen spekseihin ehdotettiin myös mahdollisuutta ladata omia videoita palvelimelle, mutta lopulta soitin päätettiin luoda minimivaatimuksin.

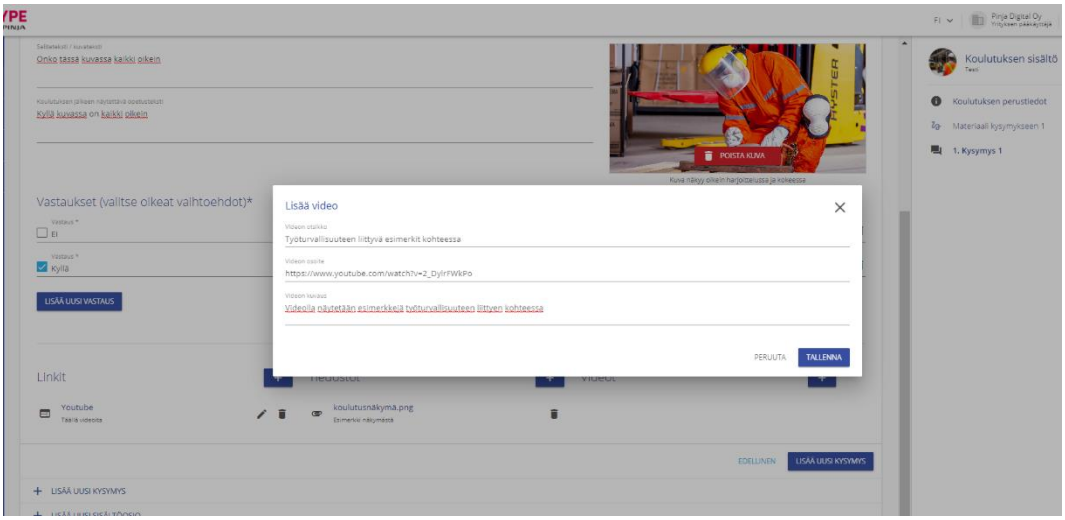
Koska videosoittimelle ei ollut olemassa olevaa tukea TyPen järjestelmässä, oli TyPeen luotava alusta pitäen täysin uusi komponentti. Aluksi etsittiin valmiita videosoittimia, jotka olisivat olleet yhteensopivia TyPen riippuvuuksien kanssa siltä varalta, että videosoitinkomponentti olisi ollut yksinkertaisesti liitettävissä järjestelmään. Valitettavasti TyPen riippuvuuksien versioiden kanssa yhteensopivaa videosoitinta ei ollut tarjolla kuin maksullisena versiona, ja TyPen spesifikaatiossa vaaditaan kaikkien käytettyjen komponenttien olevan ilmaisia.

Videosoittimen toteutus alkoi video -tyyppisten komponenttien ja tarkistusten lisäämisellä TyPen back-end toteutukseen, jotta järjestelmä ja tietokanta pystyvät käsittelemään uutta dataa. Komponentin alustamisen ja tietokannan muokkaamisen jälkeen luotiin front-end komponentit sekä yhdistettiin front- ja back-end toteutusten videokomponentti yhtenäistämällä järjestelmän käyttämät tiedon siirtämiseen vaadittavat datan alku- ja loppupisteiden formaatit.



KUVA 8. Työturvallisuuskurssin kysymyksen luonti/muokkausnäkymä.

Kun vaadittavat taustatyöt oli suoritettu, voitiin työssä siirtyä seuraavaan vaiheeseen, front-end toteutukseen. Kuvissa 8 ja 9 näkyy TyPessä käytetty visuaalinen formaatti, ja uuden videokomponentin oli täsmättävä ulkoasun kanssa sekä sovittava jo olemassa olevien ”Linkit” ja ”Tiedostot” sarakkeiden kanssa niille varattuun osioon. Koska TyPen front-end käyttää Angularia, uuden sarakkeen lisäämisessä ei ollut vaikeuksia.



KUVA 9. Videon lisäys kurssikysymykseen.

Viimeisenä visuaalisena elementtinä videon lisäämisessä oli pop-up ikkuna, joka aukeaa ”Lisää video” -painiketta painamalla (kuvassa sininen plusmerkki). Videolle vaadittavat kolme tietoa vastaavat tietokantaan aikaisemmin lisättyjä videon datakenttiä. Videoita voidaan tallentaa samalla tavalla n kappaletta jokaiselle kysymykselle. Soitinprojektin aloituspalaverissa alun videoiden määrä rajattiin yhteen, mutta työn edetessä havaittiin, että videoiden määrän rajoittamiseen kuluisi enemmän aikaa. Lisäksi videoiden määrän keinotekoiselle rajoittamiselle ei ollut hyvää argumenttia, joten päätettiin, että käyttäjät voivat lisätä haluamansa määrän videoita jokaiselle kysymyksille. Useiden videoiden näyttämisen sivulla ei myöskään havaittu tuottavan ongelmaa Angularin joustavien tekniikoiden ansiosta.

7 TYÖN TULOKSET

7.1 Viestijonon tulokset

Tavoiteltuihin työn tuloksiin päästiin suurimmalta osin. Työn kriteerit olivat seuraavat:

1. Tutkitaan, millaisia ilmaisia viestijonopalveluita on saatavilla ja tuotetaan vertailluista palveluista vertailuaineistoa.
2. Asennetaan RabbitMQ Docker -ympäristöön ja konfiguroidaan viestijono siten, että viestijonolle on perusta ja sitä voidaan tarvittaessa jatkokehittää.
3. Luodaan karkea demo viestijonoa käyttäen, vaatimuksena yksinkertaisten viestien lähettäminen ja vastaanottaminen.

Edellä mainituista kriteereistä saavutettiin ensimmäiset kaksi, ja kolmas osittain. Tuotettua aineistoa käytettiin mittaamaan työn tekijän teoreettista tietämystä viestijonon perusteista, ennen kuin työn ohjaajat hyväksyivät työn konkreettisen puolen teon aloittamisen. Muuta tuotettua aineistoa käytettiin myös työn esittelyyn työssä avustaville tahoille, sekä karkeana mallina lopulliselle demolle.

Docker -ympäristö saatiin toimimaan ja useista virheistä ja konfiguraatio-ongelmista huolimatta myös RabbitMQ:n viestijonopalvelu saatiin lopulta pystytettyä. Lopullisen version konfiguraatio oli erittäin karsittu, lähes kokonaan tehdasasetuksilla toimiva viestijono. Tarkkaa syytä virheille, jotka johtuivat kehittyneestä konfiguraatiosta, ei saatu selvitettyä ajoissa. Lopulta päätettiin käyttää karsittua, Symfonyyn pohjautuvaa mallia viestijonosta.

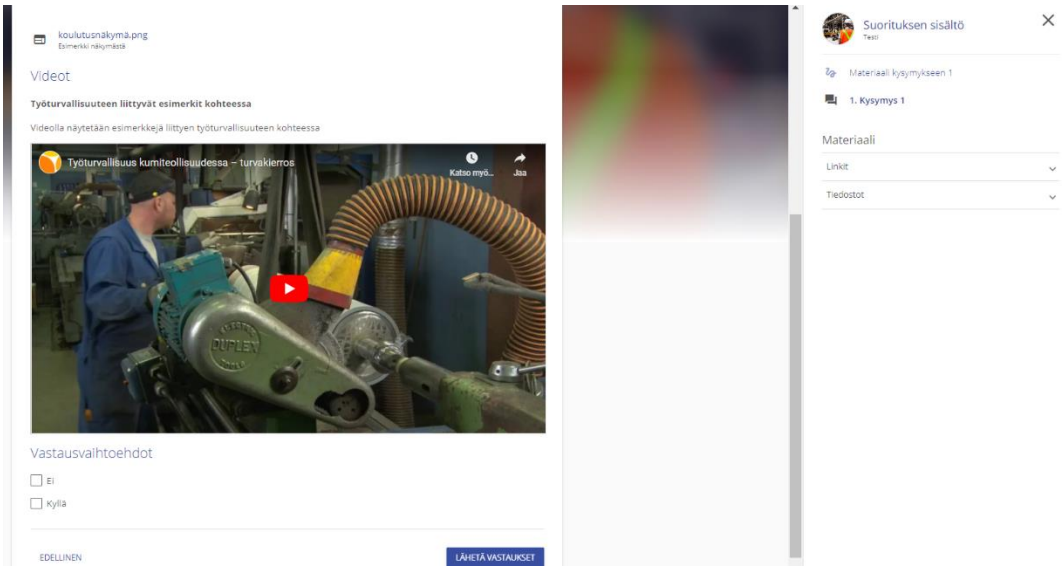
Työtä varten kehitettiin lopullista viestijonoa mallintava näytetyö, jonka pohjalta varsinainen TyPe-kokonaisuuteen liitettävä RabbitMQ palvelu luotaisiin. Lopullista toimivaa demoa ei pystytty tuottamaan edellä mainittujen virhetilanteiden selvittämiseen, sekä Symfonyn dokumentaatioon ja toiminnallisuuteen perehtymiseen kuluneen ajan vuoksi. Toimivan demon vaatimaa koodia saatiin tuotettua hieman muista projekteista otettujen esimerkkien avulla. Viestijonon työstämiseen varattu aika kuitenkin loppui ennen kuin toimiva demo oli valmis.

7.2 Videosoittimen tulokset

Videosoitin saatiin toimimaan asiakkaan antamien spesifikaatioiden mukaisesti. Työn kriteerit olivat seuraavat:

1. Tutkitaan eri vaihtoehtoja videosoittimille ja tuotetaan videosoitin parhaan mahdollisen tekniikan pohjalta.
2. Lisätään tuki videoille ja videosoittimille TyPeen uudella komponentilla.
3. Luodaan toimiva videosoitin, joka vastaa asiakkaan spesifikaatioita, ja lisätään se seuraavaan jakeluversioon.

Kaikkiin edellä mainittuihin tavoitteisiin päästiin videosoittimen osalta. Alustava tutkimistyö antoi projektiin osallistuneille henkilöille paremman kuvan TyPen kehityksen historiasta, riippuvuuksista ja rajoitteista. Alkupäässä videosoittimen kehitys oli hieman sekavaa vaillinaisen ohjeistuksen ja kommunikaation takia, mutta kun videosoittimen idea ja käytettävät tekniikat kiteytyivät, kehitys muuttui jouhevaksi. Projektin tekijä pääsi myös henkilökohtaisesti vaikuttamaan kehitysprosessiin, jonka vuoksi asiakkaille ei asetettu keinotekoisia rajoitteita videoiden määrälle siltä varalta, että asiakkailla olisi useita tiettyyn kysymykseen tai aihealueeseen liittyviä opetus- ja havainnollistamisvideoita.



KUVA 10. Kuvakaappaus videosoittimesta kurssin suoritusnäkymässä.

8 JATKOKEHITYS

Työn tuloksien ja tuotetun aineiston pohjalta työtä olisi helppo jatkokehittää. Ensimmäinen jatkokehityskohde olisi toimivan demon jatkokehitys ja viimeistely, jonka pohjalta työtä pystyttäisiin laajentamaan yhdestä ongelmametodista muihin ohjelman osiin, jotka aiheuttavat ajoittaista hidastusta TyPe:n toiminnassa. Tämä toimisi ainakin väliaikaisena ratkaisuna hidasteluongelmille, ennen kuin hitautta aiheuttavat komponentit voitaisiin ottaa käsittelyyn ja optimoida.

Jatkokehityksen ensimmäinen vaiheen jälkeen voitaisiin viestijonototeutusta laajentaa käsittämään suurta osaa TyPe:stä, jolloin koko ohjelmasta voitaisiin kehittää asynkronisesti toimiva kokonaisuus. Tällöin virhetilanteen sattuessa käyttäjä ei välttämättä huomaisi ongelmaa, eikä käyttäjän syöte välttämättä katoaisi, vaan jäisi talteen viestijonoon, kunnes back-end pystyisi käsittelemään käyttäjän suorittaman toiminteen. Sama virheensietokyky pätisi myös toisen suuntaiseen kommunikaatioon, jos käyttäjään ei jostain syystä saataisi yhteyttä.

Lopullisena kehityskohteena viestijonototeutusta voitaisiin soveltaa muissa Pinjan projekteissa ja tuotteissa. Mikäli viestijonosta saataisiin tuotettua toimiva geneerinen malli, jonka pohjalle rakentaa uusia tuotteita, voitaisiin Pinjan tulevat web-pohjaiset projektit rakentaa asynkroniselle viestijonoja hyödyntävälle alustalle.

Videosoitinkomponenttia puolestaan voitaisiin kehittää lisäämällä tuki omien videoiden lataamiselle. TyPessä on jo tuki kuvien lataamiselle palvelimelle, joten samaa tekniikkaa voitaisiin käyttää myös videoille. Kuitenkin on otettava huomioon videotiedostojen suuri koko, mikä saattaa rajoittaa palvelimille ladattavien videoiden laatua tai määrää.

9 LÄHDELUETTELO

Amazon AWS. (ei pvm). *What is Amazon Simple Queue Service?: Amazon AWS*. Haettu 26. elokuuta 2020 osoitteesta Amazon AWS:n verkkosivu: https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html

Amazon. (ei pvm). *Benefits of Message Queues: Amazon AWS*. Haettu 17. elokuuta 2020 osoitteesta Amazon Inc:n verkkosivu: https://aws.amazon.com/message-queue/benefits/

Amazon. (ei pvm). *Message Queues: Amazon AWS*. Haettu 13. maaliskuuta 2020 osoitteesta Amazon Inc:n verkkosivu: https://aws.amazon.com/message-queue/

Amazon. (ei pvm). *What is Docker?: Amazon AWS*. Haettu 3. huhtikuuta 2020 osoitteesta Amazon Inc:n verkkosivu: https://aws.amazon.com/docker/

Angular Docs. (ei pvm). *Introduction to the Angular Docs: Angular Team*. Haettu 13. maaliskuuta 2020 osoitteesta Angular.io dokumentaatio: https://angular.io/docs

Brown, K. (13. marraskuuta 2019). *What is Github and What Is It Used For?: How-To Geek*. Noudettu osoitteesta How-To Geek verkkosivu: https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/

Butler, T. (25. heinäkuuta 2015). *What is a Dockerfile?: Conetix*. Noudettu osoitteesta Contex:n verkkosivu: https://conetix.com.au/blog/what-is-a-dockerfile/

Docker Inc. (ei pvm). *What is a Container?: Docker Inc.* Haettu 7. elokuuta 2020 osoitteesta Docker Inc.:n verkkosivusto: https://www.docker.com/resources/what-container

Docker, Inc. (ei pvm). Our Application: Docker, Inc. Haettu 3. syyskuuta 2020 osoitteesta Docker Labs:n alkeiskurssi : http://ip172-18-0-30-bt813n3oudsg00cpilfg-80.direct.labs.play-with-docker.com/tutorial/our-application/

Google. (ei pvm). *Cloud Tasks: Google*. Haettu 26. elokuuta 2020 osoitteesta Googlen verkkosivu: https://cloud.google.com/tasks/

IBM. (28. heinäkuuta 2020). *Introduction to message queuing: IBM*. Haettu 19. elokuta 2020 osoitteesta IBM:n verkkosivu: https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_8.0.0/com.ibm.mq.pro.doc/q002620_.htm

James, M. (28. kesäkuuta 2013). *What is Asynchronous Programming?* Haettu 14. elokuuta 2020 osoitteesta www.i-programmer.info: https://www.i-programmer.info/programming/theory/6040-what-is-asynchronous-programming.html

Johansson, L. (23. syyskuuta 2019). *Part 1: RabbitMQ for beginners - What is RabbitMQ?: CloudAMQP*. Haettu 21. elokuuta 2020 osoitteesta CloudAMQP:n verkkosivu: https://www.cloudamqp.com/blog/2015-05-18-part1-rabbitmq-for-beginners-what-is-rabbitmq.html

Johansson, L. (23. syyskuuta 2019). RabbitMQ:n vaihdetyypit ja jonorakenne. *RabbitMQ Topic Exchange*. CloudAMQP. Haettu 21. elokuuta 2020 osoitteesta https://www.cloudamqp.com/blog/2015-05-18-part1-rabbitmq-for-beginners-what-is-rabbitmq.html

Johansson, L. (23. syyskuuta 2019). RabbitMQ:n vaihteet, sidokset ja reititykset. *RabbitMQ Exchanges, Bindings and Routing Keys*. CloudAMQP. Haettu 21. elokuuta 2020 osoitteesta https://www.cloudamqp.com/blog/2015-05-18-part1-rabbitmq-for-beginners-what-is-rabbitmq.html

Johansson, L. (25. lokakuu 2019). *What is message queuing?* Haettu 13. maaliskuuta 2020 osoitteesta CloudAMPQ: https://www.cloudamqp.com/blog/2014-12-03-what-is-message-queuing.html

Microsoft. (8. elokuuta 2018). *What is Docker?: Microsoft Corporation*. Haettu 5. huhtikuuta 2020 osoitteesta Microsoft korporaation verkkosivu: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/container-docker-introduction/docker-defined

Microsoft Corporation. (5. elokuuta 2020). *Get started with Azure Queue storage using .NET*. Haettu 26. elokuuta 2020 osoitteesta Microsoft korporaation verkkosivu: https://docs.microsoft.com/en-us/azure/storage/queues/storage-dotnet-how-to-use-queues?tabs=dotnet

Oracle Corporation. (ei pvm). *VirtualBox: Oracle Corportation*. Haettu 3. syyskuuta 2020 osoitteesta Oracle -yhtymän verkkosivu: https://www.virtualbox.org/

PHP.net. (ei pvm). *What is PHP?: The PHP Group*. Haettu 13. maaliskuuta 2020 osoitteesta php.net: https://www.php.net/manual/en/intro-whatis.php

Pinja Group Oy. (ei pvm). *Me olemme Pinja: Pinja*. Haettu 23. marraskuuta 2020 osoitteesta Pinjan verkkosivusto: https://www.pinja.com/me-olemme-pinja/

Pinja Group Oy. (ei pvm). *TyPe by Pinja: Pinja*. Haettu 12. maaliskuuta 2020 osoitteesta Pinjan verkkosivusto: https://www.pinja.com/teollinen-digitalisaatio/type/

Pivotal. (ei pvm). Esimerkki Point-to-Point -viestijonosta. *An example of bindings*. Haettu 19. elokuuta 2020 osoitteesta https://www.rabbitmq.com/tutorials/tutorial-three-php.html

Pivotal. (ei pvm). Esimerkki Publisher/Subscribe -viestijonosta. *An example of Publisher/Subscribe queue*. Haettu 19. elokuuta 2020 osoitteesta https://www.rabbitmq.com/tutorials/tutorial-four-python.html

Pivotal. (ei pvm). *Flow Control: Pivotal*. Haettu 15. syyskuuta 2020 osoitteesta Pivotal, RabbitMQ verkkosivusto: https://www.rabbitmq.com/flow-control.html

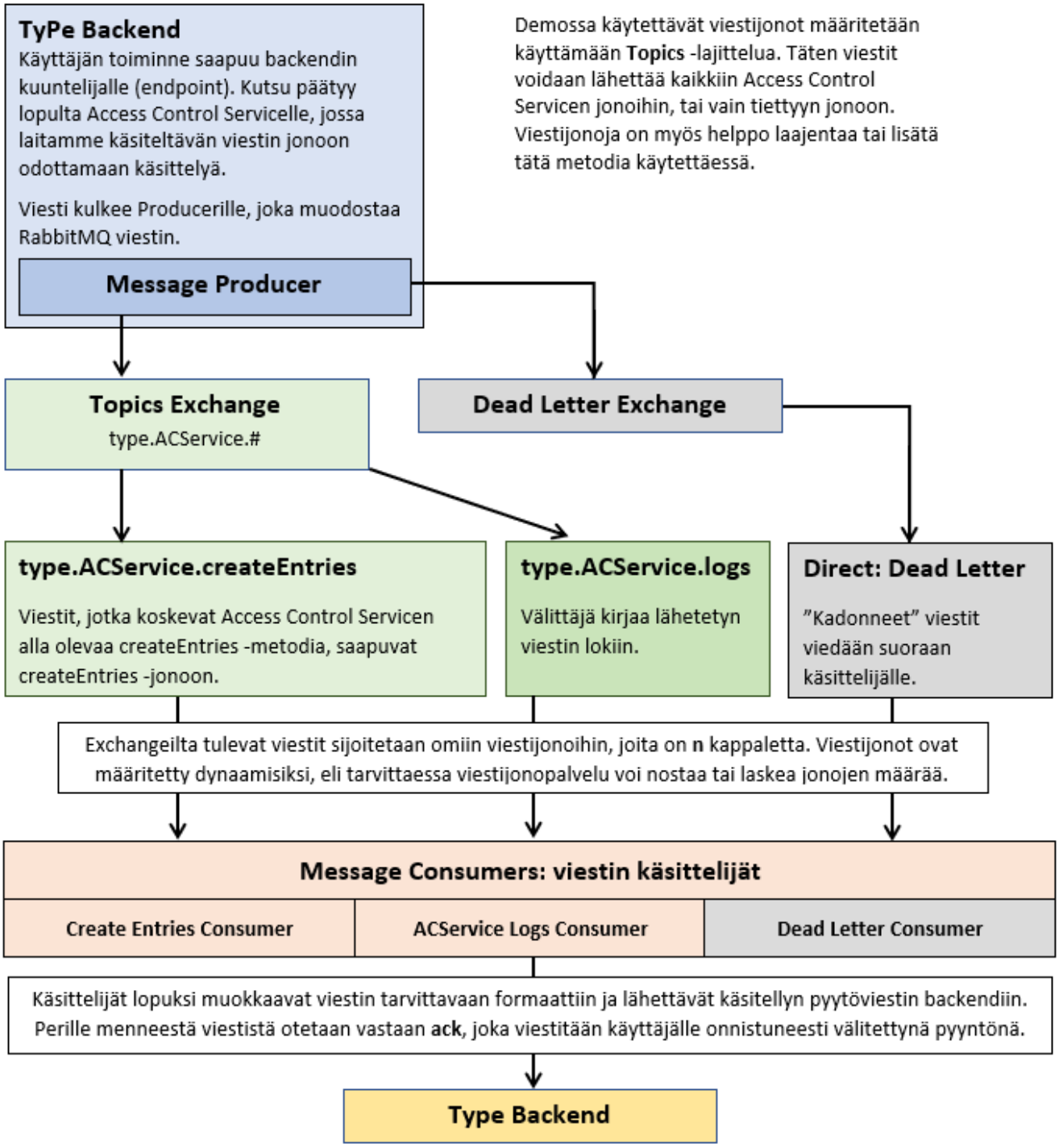
Pivotal. (ei pvm). Lopullinen esimerkki reititetystä viestijonosta. *Tutorials, Python four*. Haettu 27. elokuuta 2020 osoitteesta RabbitMQ:n alkeiskurssi 4: https://www.rabbitmq.com/tutorials/tutorial-four-php.html

Pivotal. (ei pvm). *RabbitMQ is the most widely deployed open source message broker: Pivotal*. Haettu 13. maaliskuuta 2020 osoitteesta Pivotal:n verkkosivusto: https://www.rabbitmq.com/

Symfony.com. (ei pvm). *What is Symfony: Symfony*. Haettu 13. maaliskuuta 2020 osoitteesta Symfony -yhteisön verkkosivusto: https://symfony.com/what-is-symfony

Ubuntu MATE Team. (ei pvm). *In a nutshell...: Ubuntu MATE Team*. Haettu 5. huhtikuuta 2020 osoitteesta Ubuntu-mate.org verkkosivu: https://ubuntu-mate.org/

LIITE 1: JÄRJESTELMÄN VUOKAAVIO



LIITE 2: RABBITMQ.CONF

```
# Default sockets
listeners.tcp.default = 5672
# listeners.ssl.default = 5671

## Number of Erlang processes that will accept connections for the TCP and TLS listeners.
# num_acceptors.tcp = 10
# num_acceptors.ssl = 10

## Maximum amount of time allowed for the AMQP 0-9-1 and AMQP 1.0 handshake
## (performed after socket connection and TLS handshake) to complete, in milliseconds.
##
handshake_timeout = 10000

## Memory threshold at which the flow control is triggered.
vm_memory_high_watermark.relative = 0.5

## Disk free space limit of the partition on which RabbitMQ is storing data.
## When available disk space falls below this limit, flow control is triggered.
disk_free_limit.relative = 3.0

## Level of errors logged.
## error = only errors are logged
## warning = only errors and warning are logged
## info = errors, warnings and informational messages are logged
## debug = errors, warnings, informational messages and debugging messages are logged
# log.file = false
log.dir = /etc/rabbitmq/logs
log.file = rabbit.log
log.file.level = debug

## Rotate when the file reaches 10 MiB
log.file.rotation.size = 10485760

## Keep up to 5 archived log files in addition to the current one
log.file.rotation.count = 5

## Rotate the file based on date.
## Rotate file every month.
log.file.rotation.date = $M0
```

```
## Maximum permissible number of channels to negotiate with clients, not including special channel 0 used in the protocol.
channel_max = 2047

## Frequency of heartbeat.
heartbeat = 60

default_vhost = /

## Settings not suitable for production purposes.
default_user = dev
default_pass = dev
default_user_tags.administrator = true
## Loopback can access through localhost.
loopback_users.dev= true
default_permissions.configure = .*
default_permissions.read = .*
default_permissions.write = .*

## Broker, exchange and queue definitions file location.
load_definitions = /etc/rabbitmq/definitions.json
```

LIITE 3: DEFINITIONS.JSON


```
{
  "exchanges": [
    {
      "name": "acservice",
      "vhost": "/",
      "type": "topics",
      "durable": true,
      "auto_delete": false,
      "internal": false,
      "arguments": {}
    },
    {
      "name": "dead_letter_exchange",
      "vhost": "/",
      "type": "direct",
      "durable": true,
      "auto_delete": false,
      "internal": false,
      "arguments": {}
    }
  ],
  "queues": [
    {
      "name": "acservice",
      "vhost": "/",
      "durable": true,
      "auto_delete": false,
      "arguments": {}
    },
    {
      "name": "dead_letter_queue",
      "vhost": "/",
      "durable": true,
      "auto_delete": false,
      "arguments": {}
    }
  ],
```

```
 "bindings": [
  {
   "source": "acservice",
   "vhost": "/",
   "destination": "acservice",
   "destination_type": "queue",
   "routing_key": "*",
   "arguments": {}
  },
  {
   "source": "dead_letter_exchange",
   "vhost": "/",
   "destination": "dead_letter_queue",
   "destination_type": "queue",
   "routing_key": "*",
   "arguments": {}
  }
 ]
}
```