Tampereen ammattikorkeakoulu

Verkkosovelluksen sisällönhallinta osana julkaisujärjestelmää

Niko Mattila

OPINNÄYTETYÖ
Kesäkuu 2020

Tietojenkäsittely
Ohjelmistotuotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittely
Ohjelmistotuotanto

MATTILA, NIKO:
Verkkosovelluksen sisällönhallinta osana julkaisujärjestelmää

Opinnäytetyö 32 sivua
Kesäkuu 2020

Opinnäytetyön tavoite oli tutkia sisällönhallinnan tarkoitusta yleisestä näkökulmasta sekä syventyä ohjelmistokehityksen sisällönhallintaan käyttäen esimerkkinä Liferay-portaaliohjelmistoa. Tämän lisäksi opinnäytetyössä tutkitaan eri ohjelmistokehityksen ympäristöjä ja niiden käyttötarkoituksia sekä olemassa olevia sisällönsiirron ratkaisuja.

Opinnäytetyön teoriatutkimuksen tarkoituksena oli tukea ja antaa pohjatietoa opinnäytetyössä toteutetulle työkalulle. Työssä suunniteltiin ja luotiin Export-/Import-työkalu sisällönsiirtoa varten. Sen tehtävä on helpottaa ja nopeuttaa kehittäjien työtä, kun he siirtävät sisältöä ympäristöstä toiseen. Työkalun avulla siirto pystytään automatisoimaan suurelta osalta, minkä ansiosta lopputuloksen laatua on helpompi hallita.

Opinnäytetyön toimeksiantajana toimi Visma Consulting Oy. Opinnäytetyön aihe syntyi tarpeesta kehittää joustava sisällönsiirtoratkaisu ympäristöjen välille. Teknistä toteutusta tuki olemassa oleva Java-kirjasto, joka antoi raamit työlle. Lopputuloksena syntyi työkalu, joka on Liferay-portletti. Portletin käyttöliittymä toteutettiin hyödyntäen Angular-kehystä ja backend-osuus tehtiin Javalla käyttäen Liferay MVC Portlet -kehystä.

Asiasanat: sisällönhallinta, järjestelmäympäristö, liferay, portaali

ABSTRACT

Tampere University of Applied Sciences
Bachelor’s degree in business administration
Option of Software Development

MATTILA, NIKO
Web Content Management as a Part of Release System

Bachelor's thesis 32 pages
June 2020

The purpose of this thesis was to study content management from a general viewpoint, and to focus more closely on software development content management using Liferay Portal as an example. The thesis also covers development environments, their use cases and existing solutions for content transfer.

The theoretical study conducted for this thesis forms the base knowledge before the actual tool of this thesis is introduced. The actual product of this thesis is an Export/Import tool for content transfer, which aims to make content transfer faster and easier between development environments. Using this tool the content transfer process is mostly automated, which means that the result is more manageable.

This thesis was assigned by Visma Consulting Oy. The subject of this thesis came from a need for flexible content transfer between development environments. There was a Java library already built that supported and provided guidelines for this work. The result was a tool that was built to be a portlet. The user interface or frontend of the tool was built using Angular framework and the backend was built using Liferay MVC Portlet framework.

Key words: content management, development, liferay, portal

SISÄLLYS

ERITYISSANASTO

Angular	Javascript-ohjelmoinnin kehys
CMS	Sisällönhallintajärjestelmä (engl. content management system)
DAM	Digitaalisen omaisuuden hallinta (engl. digital asset management)
DTAP	Kehitys, testaus, hyväksyntä ja tuotanto (engl. development, test, assurance and production)
ECM	Yrityksen sisällönhallinta (engl. enterprise content management)
Java	Ohjelmointikieli
JSON	JavaScript Object Notation, avainarvopareihin perustuva datansiirrolle suunnattu tiedostoformaatti
Liferay Portal	Liferay yrityksen portaali ohjelmisto
Portal	Palveluntarjoajan alusta sisäisille tai ulkoisille palveluille
QA	Laadunvarmistus (engl. quality assurance)
WCM	Web-sisällönhallinta (engl. web content management)

1 JOHDANTO

Sisällönhallinta on monipuolinen aihe, joka kattaa muutakin kuin verkkosivut. Yksinään sisällönhallinta on laaja kokonaisuus, mutta nykypäivänä sisällönhallinta ajatellaan enemmän digitaalisena konseptina ja yhdistetään usein suoraan web-sisällönhallintaan. Web-sisällönhallinta sisältää jo itsessään monta eri järjestelmää ja ratkaisua, joita tulee ottaa huomioon modernissa sovelluskehityksessä. Tämän lisäksi laajalle sovellusprojektille löytyy mahdollisesti useampia järjestelmäympäristöjä, joilla jokaisella on toisestaan poikkeava käyttötarkoitus. Esimerkiksi paikallisen kehitysympäristön tarkoitus on erilainen, kuin testiympäristön taikka tuotantoympäristön. Kaikki tämä on erittäin tärkeä osa hallittua sovelluskehitystä ja juuri sen takia sisällönhallinta projektin sisällä nousee tärkeään rooliin.

Opinnäytetyön tarkoituksena oli suunnitella ja kehittää sisällönsiirtoon tarkoitettu työkalu, jonka tavoitteena on helpottaa kehittäjän työtä, kun sisältöä siirretään useamman kehitysympäristön välillä. Kyseinen työkalu toteutettiin Liferay-portaalialustalla käytettäväksi portletiksi. Työkalu voidaan jakaa kahteen osaan, joista ensimmäisenä on frontend-osuus ja toisena backend-osuus. Frontend-osuus toteutettiin käyttäen Angular-kehystä ja backend puolestaan toteutettiin käyttäen Liferay MVCPortlet -kehystä.

Opinnäytetyössä tutkitaan sisällönhallintaa ja sen järjestelmiä yleisellä tasolla ja tuodaan esiin sisällönhallinnan laajuus ja tärkeys. Tämän lisäksi opinnäytetyössä käydään läpi ohjelmistokehityksen eri järjestelmäympäristöjä ja tuodaan esiin tarve sisällön siirtämisen automatisoinnille, näiden eri järjestelmäympäristöjen välillä.

2 SISÄLLÖNHALLINTA

Tässä luvussa käsitellään sisällönhallintaa yleisesti sekä tutustutaan sisällönhallinnan tyyppeihin. Lisäksi selvitetään Liferay Portal -nimisen ohjelmiston sisällönhallinnan ominaisuuksia.

2.1 Sisällönhallinnan tarkoitus

Sisällönhallinnalla on monta eri tarkoitusta ja riippuu näkökulmasta mitä sillä halutaan saavuttaa. Yhdestä näkökulmasta sisällönhallinnan avulla saadaan liikearvoa, toisesta taas sisällönhallinta yhdistää sisältöön liittyviä oppeja ja kolmannesta sisällönhallinta on tekninen rakenne. Nämä kaikki pitävät paikkansa ja sitä varten sisällönhallinnalle ei ole yksiselitteistä määritelmää. Prosessinäkökulmasta sisällönhallinta kuitenkin on prosessi, jonka avulla kerätään, hallitaan ja julkaistaan sisältöä. (Boiko 2005, 66, 72.)

Sisällönhallinta ajatellaan usein digitaalisena konseptina, mutta sisällönhallinta on ollut olemassa niin kauan kuin sisältökin. Siitä lähtien, kun sisältöä on tuotettu, sen hallitsemiseen on etsitty ratkaisuja. Kirjastot toimivat hyvänä esimerkkinä alkuperäisestä sisällönhallinnan ratkaisusta ja kirjastonhoitajia voidaan pitää ensimmäisinä sisällönhallinnoijina. (Barker 2016.)

2.2 Sisällönhallintajärjestelmät

Nykyään sisällönhallintajärjestelmällä (engl. content management system tai CMS) viitataan ohjelmistoon, jonka avulla digitaalista sisältöä hallitaan. Nämä ohjelmistot sisältävät työkaluja, joiden tarkoitus on helpottaa sisällönhallinnan tehtävien suorittamista. Modernit sisällönhallintajärjestelmät nojautuvat enemmän siihen suuntaan, että sisällöntuottajalta tai hallinnoijalta ei vaadita ohjelmoijan kokemusta suoriutuakseen tehtävistään. (AAIM n.d.) Tämä on hyvä suunta, sillä se helpottaa sisällöntuottamista, kun sitä voi tehdä miltei kuka vain.

Sisällönhallintajärjestelmä yksinkertaisimmillaan on järjestelmä, jonka avulla useampi käyttäjä voi tehdä yhteistyötä, olla vuorovaikutuksessa ja hallinnoida dataa yhteisen rajapinnan kautta (Johnston n.d.). Loogisesti sisällönhallintajärjestelmä voi koostua useammasta eri osasta, kuten sisällön muokkaukseen käytettävästä käyttöliittymästä, tietokannasta, julkaisujärjestelmästä, käyttäjienhallintajärjestelmästä ja niin edelleen. Nämä kaikki voivat toimia omina autonomisina osina järjestelmän uumenissa mutta yleisestä näkökulmasta nämä kaikki yhdessä ovat sisällönhallintajärjestelmä. (Barker 2016.)

2.3 Sisällönhallintajärjestelmien kategorioita ja portaalit

Sisällönhallintajärjestelmiä on monenlaisia, ja yhä useammin ne nykyään palvelevat useampaa kuin yhtä käyttötarkoitusta. Vaikka niiden käyttö onkin hyvin joustavaa, on jokaisella ohjelmistolla ja järjestelmällä omat vahvuutensa sekä heikkoutensa. Sisällönhallintaratkaisua hankittaessa on hyvä ottaa huomioon oma henkilökohtainen, yhteisön tai yrityksen tarve sekä ohjelmiston tarjoamat ominaisuudet. Esimerkiksi yksinkertaisen kotisivun rakentaminen ei vaadi monimutkaista järjestelmää taustalle, joka saattaa olla normaalisti suunnattu yrityksille.

Alla on lueteltuna muutamia suosittuja sisällönhallintajärjestelmien kategorioita, jonka perusteella ohjelmiston käyttötarkoituksesta saa karkean käsityksen. Nämä kategoriat ovat kuitenkin hyvin suuntaa antavia käsitteitä moderneissa sisällönhallinnan alustoissa ja yhä useampi ohjelmisto voidaan luokitella useampaan kuin yhteen näistä. Eri kategoriat eivät myöskään sulje pois toisiaan. (Barker 2016.)

2.3.1 Web-sisällönhallintajärjestelmä

Web-sisällönhallintajärjestelmä (engl. web content management system tai WCMS) on monipuolinen, mutta yksinkertainen sisällönhallintajärjestelmä, sillä se on hyvin joustava siinä mitä käyttötapausta se palvelee. Pääasiallinen tarkoitus on sisällöntuottaminen ja sen esittäminen sivustolla. Web-sisällönhallintajär-

jestelmän määritelmä on hyvin karkea, sillä tämän kategorian alle valuvat järjestelmät saattavat sisältää muiden kategorioiden kaltaisia ominaisuuksia, kuten videoiden ja kuvien tallentamista, missä esimerkiksi digitaalisen omaisuuden hallintajärjestelmä DAM (Digital Asset Management) loistaa. (Barker 2016.) Tämän lisäksi esimerkiksi DAM ja ECM (Enterprise Content Management) yleisesti sisältävät web-sisällönhallinnalle suunnatun ratkaisun, joten web-sisällönhallintajärjestelmä voidaan ajatella niiden osajoukkona.

Web-sisällönhallintajärjestelmälle ei siis löydy yksiselitteistä tarkoitusta ja niin kuin muutkin kategoriat, sen määritelmä painottuu enemmän siihen suuntaan mihin käyttötapaukseen sen ajatellaan olevan suunnattu. (Barker 2016.) Suosittuja esimerkkejä järjestelmistä, joiden katsotaan lukeutuvan tämän kategorian alle ovat WordPress ja Drupal.

2.3.2 Digital Asset Management (DAM)

DAM-sovellus on suunnattu digitaalisen omaisuuden hallintaan. Digitaalisella omaisuudella tarkoitetaan kuvia, videoita, logoja, ääniä yms. Tarve tämänkaltaiselle sisällönhallinnalle tulee, kun halutaan yrityksen tai organisaation kaikki digitaalinen sisältö samaan paikkaan talteen. Moderni DAM-järjestelmä on käytettävissä kaikilta laitteilta, data on saavutettavissa koska tahansa sekä sisältää integraatioita muiden järjestelmien kanssa. (Bynder n.d.) Esimerkkinä DAM-sisällönhallintajärjestelmästä toimii Bynder.

2.3.3 Enterprise Content Management (ECM)

Enterprise Content Managementin kaltaisen järjestelmän tarkoitus yleisesti on hallita liiketoimintasisältöä organisaation sisällä. ECM-käsitettä voi kuvata koko organisaation kattavaksi sisällönhallinnaksi. ECM:n tavoitteena on yhdistää erillään olevat ”siilot” yhteisin periaattein ja menetelmin hallinnoitavaksi kokonaisuudeksi. (Juha Anttila n.d.) Juha Anttilan mukaan ECM koostuu tyypillisesti seuraa-

vista osa-alueista: dokumenttien, asiakirjojen, työnkulkujen ja liiketoimintaprosessien sekä verkkosisällön hallinta, ryhmätyöskentely, haku, digitointi, tekstintunnistus, sähköiset lomakkeet, sähköpostin hallinta ja arkistointi ja niin edelleen. ECM kuvastaa siis enemmänkin järjestelmää, joka pyrkii kattamaan useamman sisällönhallinnan ominaisuuden yhdistämällä useita kokonaisuuksia yhdeksi. Niin kuin aiemmin mainittiin, ECM sisältää myös jonkin sortin web-sisällönhallintajärjestelmän, jolloin ECM:n voi ajatella ylemmän tason sisällönhallintajärjestelmänä. Tämä ei kuitenkaan tarkoita sitä, etteikö web-sisällönhallintajärjestelmänä ajateltua järjestelmää voisi käyttää hallinnoimaan liiketoimintasisältöä organisaation sisällä. (Barker 2016.) Esimerkkinä ECM:ksi luokitellusta järjestelmästä toimii Alfresco.

2.3.4 Portaalit

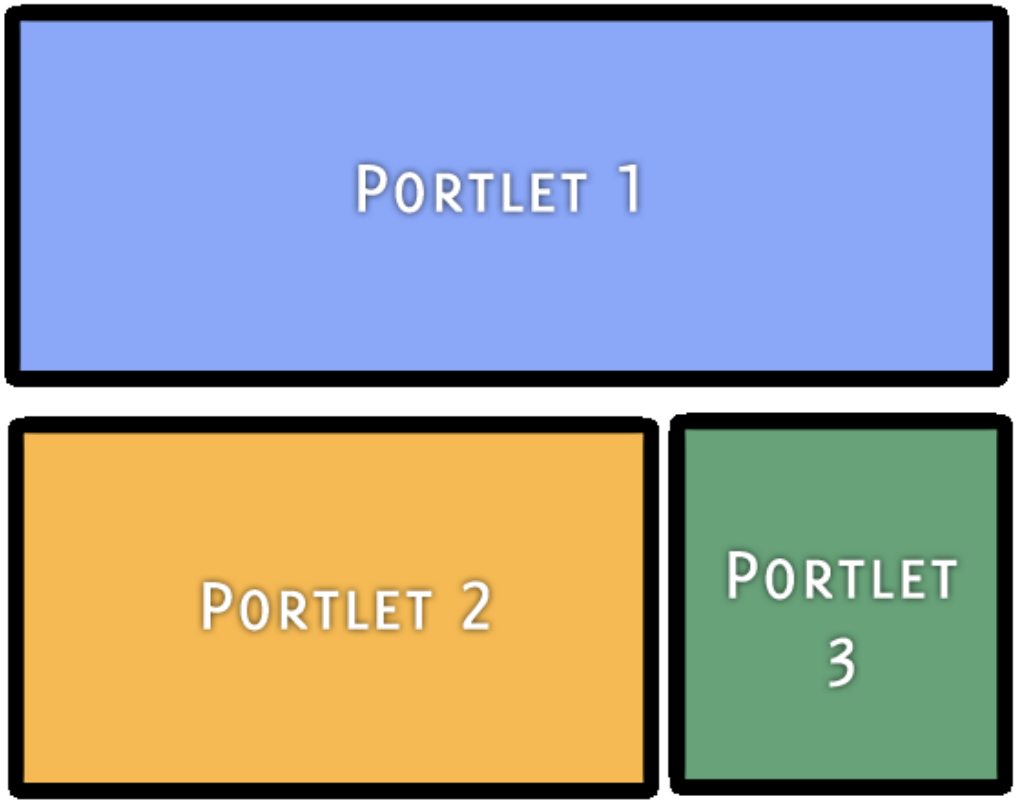
Portaalin tarkoitus on toimia palveluntarjoajan alustana joko omille tai ulkoisille palveluille. Portaali voidaan mieltää julkaisujärjestelmänä ja usein portaalijärjestelmiin on pakattuna mukaan myös sisällönhallintaratkaisu. Tämä ei kuitenkaan välttämättä rajoita sisällönhallinnoijaa käyttämään tarjottua ratkaisua eli CMS ja portaali elävät samassa ympäristössä eivätkä ne sulje toisiaan pois. Portaali voidaan tulkita ennemminkin kehyksenä sisällönesittämiselle. Opinnäytetyön kannalta tärkeänä esimerkkinä tästä toimii Liferay Portal.

2.4 Liferay-portaali

Liferay-portaali sisältää paljon valmiita ominaisuuksia, mutta tässä alaluvussa keskitytään kolmeen opinnäytetyön toteutukselle tärkeään osuuteen: portletteihin (engl. portlet), web-sisältöön (engl. web content) sekä dokumentti- ja mediakirjastoon (engl. document & media library).

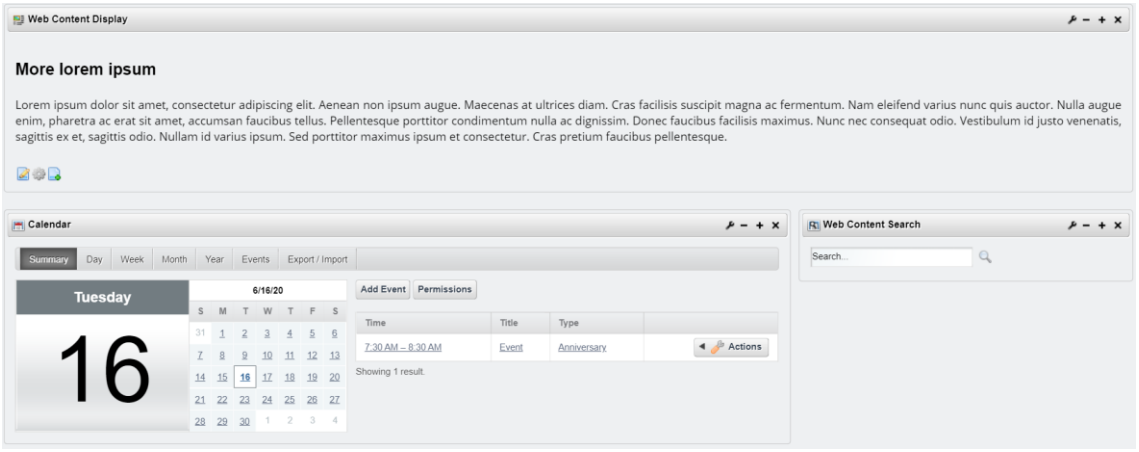
2.4.1 Portletit

Portletit ovat sovelluspalikoita, joita voidaan asettaa sivustolle yksi tai useampi. Jokainen portletti toimii omana yksikkönään ja nämä yksiköt yhdessä muodostavat portaalissa sivun. Liferay Portal käyttää portletteja toiminnallisuuksiin kuten kalenteriin, kirjanmerkkeihin, sisältöhakuun ja moneen muuhun. (Liferay n.d.a) Kuviossa 1 on hahmotelma sivusta, jolle on asetettu kolme eri portlettia, joista jokainen on oma sovelluksensa ja palvelee tiettyä käyttötarkoitusta.



Kuvio 1. Hahmotelma useammasta portletista yhdellä sivulla (Liferay n.d.a)

Alla olevan kuvan 1 tarkoitus on havainnollistaa tarkemmin miltä näyttää, kun useampi portletti esiintyy sivulla. Kuvassa olevat portletit ovat: Web-sisällön esitys, kalenteri sekä web-sisällön haku. Nämä portletit tulevat mukana valmiiksi asennettuna Liferay Portalissa.



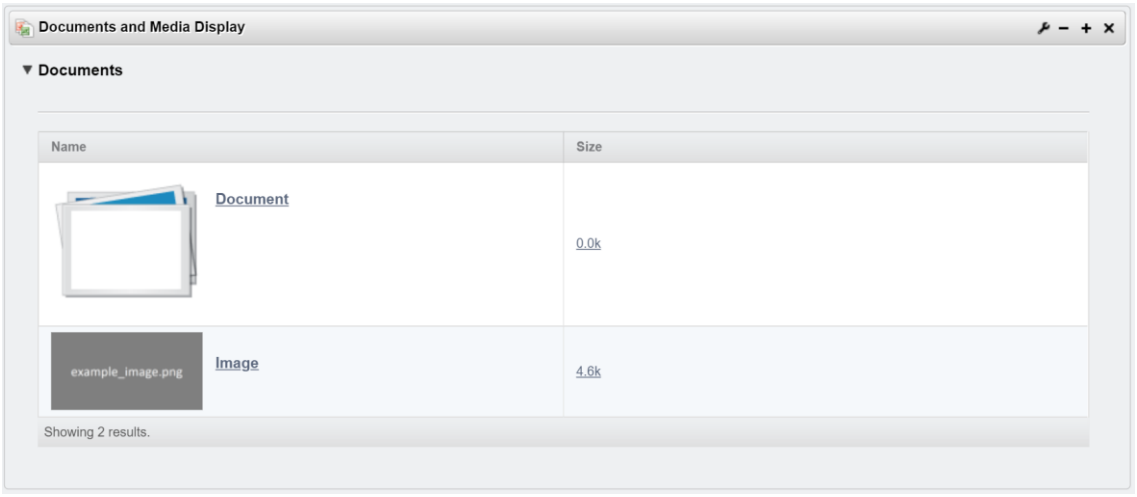
Kuva 1. Esimerkki Liferay sivustosta, jossa kolme portlettia (kuvakaappaus)

2.4.2 Web-sisältö

Web-sisällönhallinta (engl. web content management tai WCM) on Liferayn sisäänrakennettu järjestelmä, jolla voidaan muokata, versioida ja lavastaa (stage) web-sisältöä ilman teknistä osaamista (Liferay n.d.a). Tämä on sisällöntuottajan hyödyllisin ominaisuus, kun puhutaan sisällönhallinnasta Liferayn kontekstissa. Tätä sisältöä esitetään portletissa nimeltään ”Web Content Display” Kyseessä on portletti, jonka tarkoitus on esittää tuotettua sisältöä sekä helpottaa sisällöntuottamista sivustolle. Tämän lisäksi portletissa on mahdollistettu sisällönmuokkaus suoraan portletin käyttöliittymässä. Tämän käyttäminen ei kuitenkaan ole pakollista, mutta valmiiden työkalujen hyödyntäminen säästää aikaa.

2.4.3 Dokumentti ja mediakirjasto

Liferayn dokumentti- ja mediakirjasto tarjoaa mekanismin tiedostojen tallentamiseen verkossa käyttäen samantyyppistä rakennetta, jota käytetään tiedostojen tallentamiseen paikallisesti (Liferay n.d.a). Näihin kuviin ja dokumentteihin voidaan viitata suoraan esimerkiksi web-sisällön esityksessä. Dokumentti ja mediakirjaston sisältöä voidaan myös esittää alla olevan kuvan 2 mukaisesti portletissa ”Documents and Media Display”.



Kuva 2. Esimerkkikuva dokumentti ja mediakirjasto portletista (kuvakaappaus)

3 JÄRJESTELMÄYMPÄRISTÖT

Tässä luvussa käydään läpi erilaisia sovelluskehityksessä käytettyjä järjestelmäympäristöjä ja niiden tarkoitusta sekä esitellään tarve optimaaliselle tavalle siirtää sisältöä ympäristöstä toiseen. Lisäksi luvussa käydään läpi eri tapoja siirtää sisältöä ympäristöstä toiseen.

3.1 Eri järjestelmäympäristöt

Nykyaikaiseen sovelluskehittämiseen kuuluu usein monta eri järjestelmäympäristöä, joiden tarkoitus on parantaa sovelluksen kehityksen hallintaa. Kaikki sovelluksen julkaisuvaiheet eivät välttämättä tapahdu erillisissä ympäristöissä, mutta se on kuitenkin mahdollista ja joissain tapauksissa jopa suotavaa. Tässä alaluvussa mainitut järjestelmäympäristöt toimivat esimerkkinä, ja näitä tyyppejä on muitakin ja ne saattavat kulkevat eri nimellä tai määritelmällä. Kuvio 2 esittää kontrolloidun kehitys-, testaus-, hyväksyntä- ja tuotantoputken (engl. development, test, acceptance and production), jonka tarkoitus on toimia testauksen ja käyttöönoton lähestymistapana. (Visser 2017.)

Kuvio 2. Controlled DTAP (Visser 2017.)

Paikallinen kehitysympäristö

Tämän ympäristön tarkoitus on toimia kehittäjän omana kehitysympäristönä. Ympäristöllä halutaan pystyä vapaasti kehittämään, testaamaan ja poistamaan virheitä (engl. debug) sovelluksesta tai sivustosta. Suurempien muutosten ja sovellusten testaus toteutetaan huolellisesti tässä suljetussa ympäristössä ennen sen

siirtämistä muihin ympäristöihin. Ympäristön tarkoitus on olla mahdollisimman produktiivinen kehittäjälle. (Visser 2017.)

Testausympäristö

Ympäristön tarkoitus on nimensä mukaan toimia testaamisen ympäristönä, jossa voidaan huolellisesti testata läpi kaikki mahdolliset testitapaukset, jotka ovat osana ominaisuuden kehityksen työnkulkua. Suurin osa sisäisestä testauksesta pyritään hoitamaan tässä ympäristössä. Testiympäristölle yleisesti on oma testidata, jolla pystytään varmistamaan rajatapausten toimivuus uusissa ominaisuuksissa, pitäen kuitenkin datan mahdollisimman todenmukaisena suhteessa tuotantoympäristöön.

Demonstraatioympäristö

Demonstraatioympäristön (engl. demonstration environment tai demo) tarkoituksena on uusien ominaisuuksien esittely asiakkaalle tai tuotteen esittely uudelle asiakkuudelle. Tätä ympäristöä voidaan myös vaihtoehtoisesti käyttää uusien ominaisuuksien testaukseen ennen julkaisua. Lisäksi tätä ympäristöä voidaan käyttää julkaisemattomien sisältöjen hallintaan. Tämä ympäristö ei välttämättä ole osa kehityksen työnkulkua, mutta toimii hyvänä ympäristönä markkinoinnille.

Hyväksymistestausympäristö

Hyväksymistestausympäristö (engl. acceptance environment) on hyvin yleinen ympäristö, joka on tärkeä osa hyvän verkkopalvelun kehittämisessä. Tämä ympäristö toimii viimeisenä varmistuksena ennen kuin sisältö tai ominaisuudet julkaistaan tuotantoympäristössä. Hyväksymistestauksen työnkulkuun kuuluu usein viimeiset sisäiset testaukset ja asiakkaan testaus, jonka perusteella päätetään ominaisuuksien viemisestä tuotantoon. Joskus tämän ympäristön tilalla voidaan käyttää laadunvarmistusympäristöä (engl. quality assurance environment tai QA), mutta periaate on miltei sama. Ympäristö on optimoitu vastaamaan tuotantoympäristöä (Visser 2007.). Hyväksymistestausympäristöstä siirretään usein sisältö sellaisenaan tuotantoon.

Tuotantoympäristö

Tuotantoympäristö (engl. production environment) toimii paikkana, johon aiemmissa ympäristöissä testattu ja viimeistelty data, sisältö ja ominaisuudet tuodaan. Tuotantoympäristö on se ympäristö, joka useimmiten palvelee loppukäyttäjää ja joka toimii ympäristönä, johon verkkosovellus julkaistaan.

3.2 Järjestelmäympäristöjen määrä ja tarve siirron automatisoinnille

Kuten aiemmassa alaluvussa huomasimme, järjestelmäympäristöjä voi olla useita. Tämä ei kuitenkaan tarkoita sitä, että kaikki ympäristöt ovat välttämättömiä ja niiden määrä usein riippuu käyttötapauksesta ja projektin laajuudesta. Jokaisella ympäristöllä on kuitenkin oma tehtävänsä ja tarkoitus, jota palvella. Usein sovelluskehityksen laadun varmistamiseksi on hyvä panostaa useampaan järjestelmäympäristöön, mutta itse sisällönhallinta ja tuottaminen hankaloituu. Tämän takia on olemassa useampi tapa siirtää sisältöä ympäristöstä toiseen. Käymme tässä läpi kolmea vaihtoehtoa, jotka kuitenkin poikkeavat toisistaan ja ovat hyvin käyttötapauskohtaisia.

3.3 Manuaalinen hallinta

Manuaalinen hallinta on periaatteena hyvin yksinkertainen. Sisältöä tuotetaan yhteen ympäristöön, jossa voidaan suorittaa testaukset ja hyväksyminen, jonka jälkeen sama sisältö tuotetaan seuraavaan ympäristöön. Tämä prosessi on hidas, sillä samat toimenpiteet joudutaan tekemään useampaan kertaan. Tämän lisäksi manuaalinen hallinta on hyvin virhealtis ja lopputulos saattaa poiketa eri ympäristöissä. Manuaalisen hallinnan positiivinen puoli tosin on se, että se ei vaadi ohjelmistoja taikka ominaisuuksia ympäristöltä. Tämän lisäksi joskus voi olla helpompaa siirtää sisältö manuaalisesti, jos ympäristöjä ei ole montaa tai sisältöä ei ole paljoa.

3.4 Liferay Staging

Staging on Liferay-portaalin yksi sisällönhallinnan työkalu, jonka tarkoitus on helpottaa sisällöntuottamisen ja julkaisun työnkulkua. Staging perustuu siihen, että sisältöä tuotetaan sivuston niin sanottuun työkopioon ja julkaistaan toiselle sivustolle haluttaessa. Tämä mahdollistaa monipuolisen työnkulun, johon sisältyy testaus ja katselmointi. (Liferay n.d.b.) Staging perustuu kahdenlaiseen ympäristöön: Staging-ympäristö ja Live-ympäristö, missä Staging-ympäristössä on työkopio sivustosta ja Live-ympäristössä on julkaistu, käyttäjien käytössä oleva versio sivustosta (Liferay n.d.b). Staging on näiden kahden eri ympäristön datan synkronointia. Tämän synkronoinnin aikana tapahtuu seuraavat asiat:

- Tiedon kerääminen. Suoritetaan tietokantakutsut, jotka hakevat tarvittavan sisällön.
- Siirto. Siirretään kerätyt tiedot kohdesivustoon. Tämä vaihe vain siinä tapauksessa, jos käytetään Remote Live Staging. Tästä tarkemmin alempana.
- Validointi. Validoidaan siirretty data ennen julkaisua.
- Julkaisu. Julkaistaan validoitu data Live sivustolla.

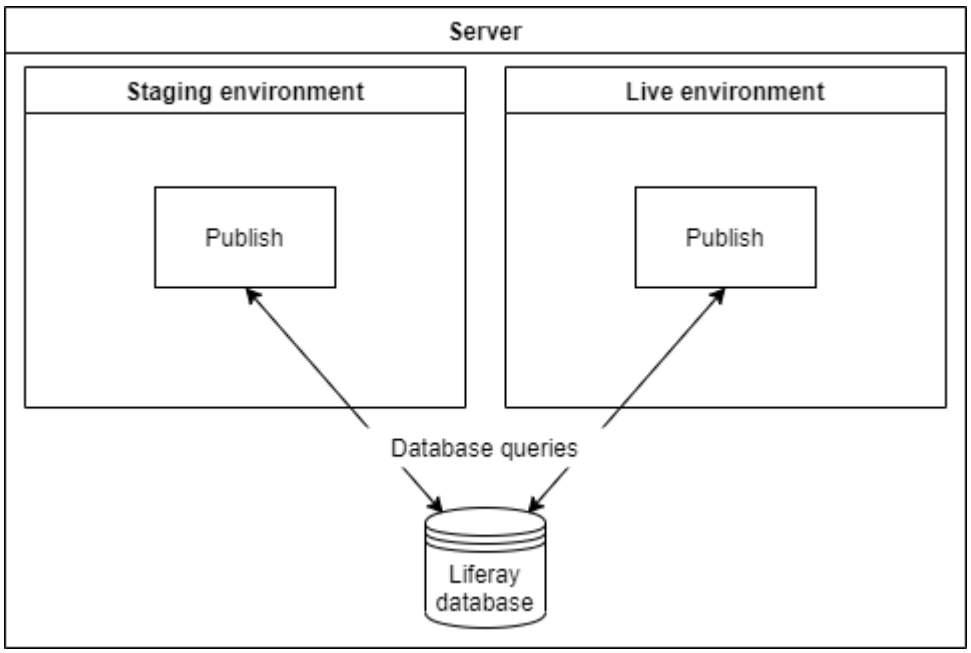
(Thurzo 2014.)

Tämänkaltainen sisällönsiirto Liferayssä ei kuitenkaan kohdistu kustomoituihin portletteihin. Stagingia ei siis ole tarkoitettu viemään koodia ympäristöstä toiseen ja se tulee hoitaa muilla keinoilla. (Thurzo 2014.)

Staging voidaan ottaa sivustolla käyttöön kahdella eri tavalla: Local Live Staging ja Remote Live Staging. (Liferay n.d.b)

Liferay Local Live Staging

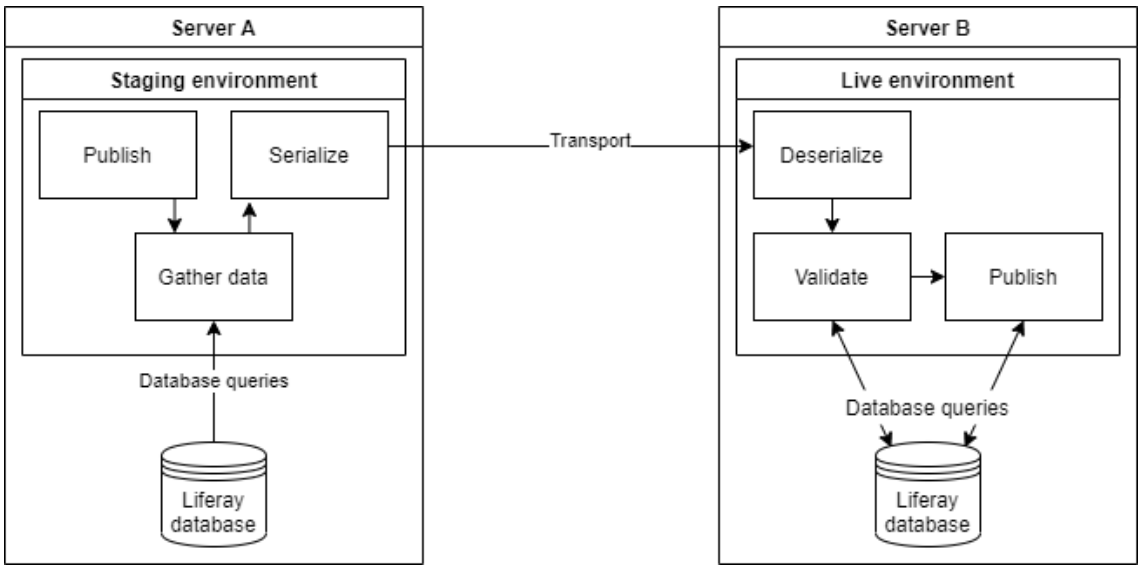
Local Live Staging tarkoittaa Stagingin käyttämistä paikallisesti. Tämä tarkoittaa sitä, että sivuston työkopio sekä Live-sivusto ovat samassa ympäristössä. Tämä on parempi vaihtoehto kevyille sivustoille, sillä se vaatii yksinkertaisemman palvelinarkkitehtuurin. (Liferay n.d.a) Kuviossa 3 on hahmotettuna tiedon kerääminen ja julkaisu käyttäen Local Live Stagingiä. (Thurzo 2014.)



Kuvio 3. Yksinkertaistettu havainnollistus Local Live Stagingistä

Liferay Remote Live Staging

Remote Live Staging tarkoittaa, että sivuston työkopio on eri palvelimella, kuin Live-ympäristö. Sen lisäksi, että ympäristöt ovat eri palvelimella, on niillä erilliset tietokannat. Tämä on hyvä vaihtoehto varsinkin useamman sivuston kanssa työskennellessä, sillä yhdellä työkopiolla voidaan hallita useampaa Live-sivustoa. (Liferay n.d.b) Kuviossa 4 on hahmotettuna tiedon kerääminen, siirto, validointi sekä julkaisu käyttäen Remote Live Stagingiä. (Thurzo 2014.)



Kuvio 4. Yksinkertaistettu havainnollistus Remote Live Stagingista

3.5 Export/Import

Export/Import-sisällönsiirrossa tarkoituksena on tuoda ympäristön A sisältö pakettina ulos ja viedä tämä sisältö ympäristöön B. Tämä sisällönsiirto toimii erittäin hyvin useamman ympäristön kanssa ja on täysin riippumaton suunnasta. Tämän järjestelmän avulla pystytään nopeasti siirtämään suuriakin sisältökokonaisuuksia hyvin vähäisellä vaivalla. Lisäksi siirtopaketti voidaan tallentaa versionhallintaan, joka mahdollistaa sisällön palauttamisen aikaisempaan tilaan. Työkalun toteuttaminen vaatii kuitenkin käytettävän sisällönhallintajärjestelmän tietorakenteiden syvällistä tuntemista.

4 TEKNINEN TOTEUTUS

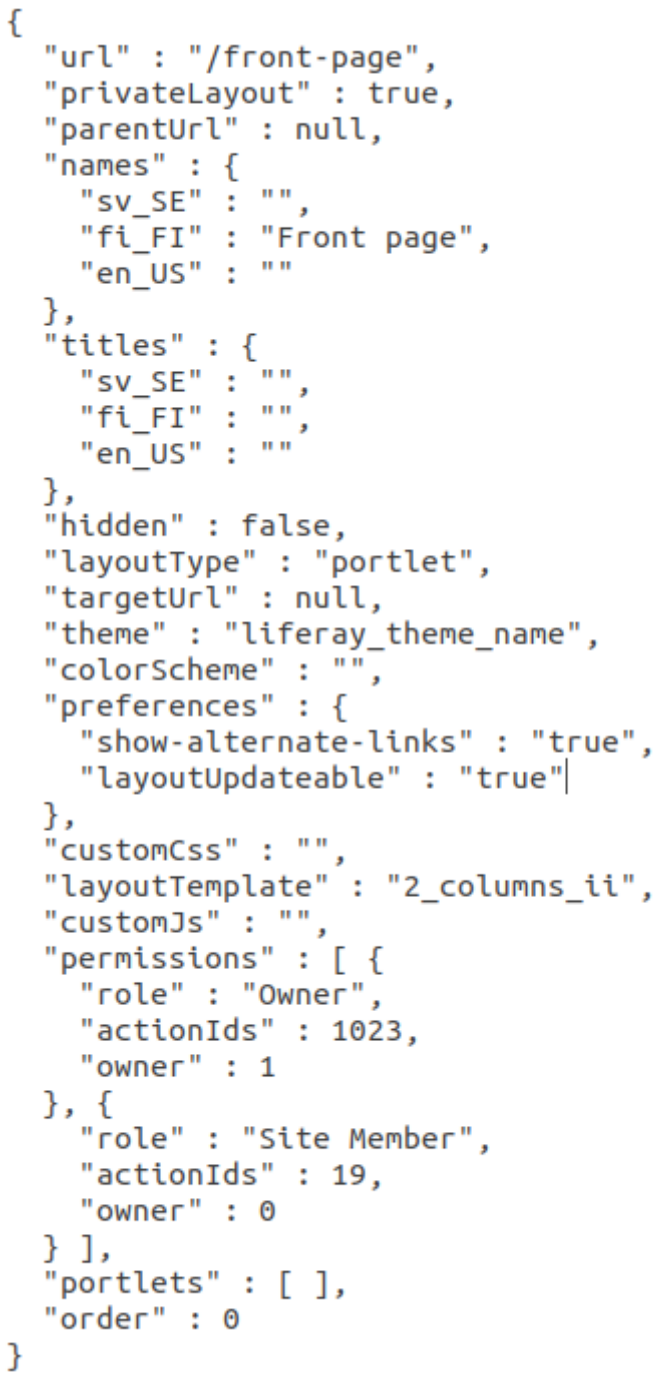
Opinnäytetyön teknisenä toteutuksena tehtiin työkalu, jolla voidaan siirtää sivuston sisältöä ympäristöstä toiseen. Työkalun toimintaperiaate pohjautui vientiin ja tuontiin. Sivustojen siirtotyökalu suunniteltiin tehostamaan ympäristöjen perustamista sekä synkronoimaan muutostenhallintaa. Lisäksi tavoitteena oli automatisoida siirron manuaalisia vaiheita, inhimillisten virheiden mahdollisuuden vähentämiseksi. Tässä luvussa käydään läpi työn tekniselle toteutukselle annetut vaatimukset, ratkaisukuvaukset sekä toteutus.

4.1 Vaatimukset

Työn tekeminen alkoi suunnittelusta, jonka pohjana toimi sille määritetyt vaatimukset. Työkalun export-osuudessa ensimmäisenä määrityksenä toimi mahdollisuus valita, mitkä sivut tuodaan ulos vietävään pakettiin. Tämän tarkoitus oli rajata vietävän tiedon määrää siinä tilanteessa, jos muutoksia on tehty vain osaan sivustoista. Lisäksi voidaan rajata myös muutoksia, joita ei vielä välttämättä haluta siirtää kohdeympäristöön.

Toisena vaatimuksena oli, että export-paketin tulee sisältää valitun sivun kaikki sisällöt, joihin lasketaan web-sisältö, sivustoasetukset, portlettimääritykset jne. Näiden tietojen ei kuitenkaan saa sisältää ympäristökohtaisia attribuutteja vaan datan tulee sisältää vain loogiset arvot, jotka pysyvät muuttumattomina eri ympäristöjen välillä.

Kolmantena vaatimuksena oli, että export-paketin tulee olla JSON-muodossa (Java Script Object Notation). JSON on paljon käytetty datansiirrolle omistettu tiedostoformaatti, joka perustuu avainarvopareihin. Tämän tiedostoformaatin hyöty on sen luettavuus, yleisyys sekä mahdollisuus versioida versionhallinnassa. Kuvassa 3 on esiteltynä esimerkki export-paketin sisältämästä JSON-tiedostosta.



```
{
  "url" : "/front-page",
  "privateLayout" : true,
  "parentUrl" : null,
  "names" : {
    "sv_SE" : "",
    "fi_FI" : "Front page",
    "en_US" : ""
  },
  "titles" : {
    "sv_SE" : "",
    "fi_FI" : "",
    "en_US" : ""
  },
  "hidden" : false,
  "layoutType" : "portlet",
  "targetUrl" : null,
  "theme" : "liferay_theme_name",
  "colorScheme" : "",
  "preferences" : {
    "show-alternate-links" : "true",
    "layoutUpdateable" : "true"
  },
  "customCss" : "",
  "layoutTemplate" : "2_columns_ii",
  "customJs" : "",
  "permissions" : [ {
    "role" : "Owner",
    "actionIds" : 1023,
    "owner" : 1
  }, {
    "role" : "Site Member",
    "actionIds" : 19,
    "owner" : 0
  } ],
  "portlets" : [ ],
  "order" : 0
}
```

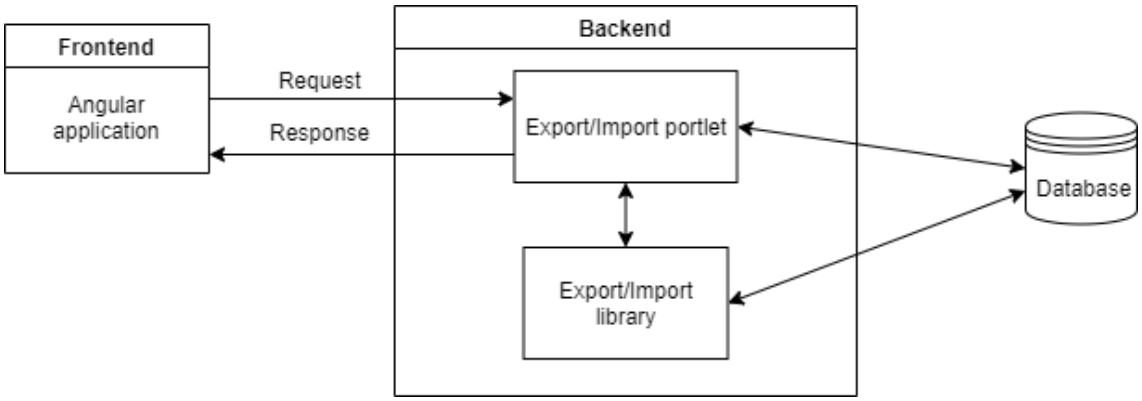
Kuva 3. Esimerkkikuva yhden siirrettävän sivun JSON-datasta (kuvakaappaus)

Import-toiminnallisuuden ensimmäisenä vaatimuksena toimi mahdollisuus valita, tullaanko kohdeympäristössä olevat sivut ja asetukset ylikirjoittamaan. Tämä on hyvin tärkeä ominaisuus tilanteissa, joissa sisältö muuttuu niin radikaalisti, että ne halutaan ylikirjoittaa kokonaan. Yksi esimerkki tällaisesta tilanteesta on se, kun halutaan tuoda sisältöä tuoreeseen ympäristöön, joka sisältää esimerkiksi vain vakiosivuja. Toisena vaatimuksena oli kyky osata yhdistää tuotava sisältö

olemassa olevaan sisältöön. Tämä tarkoittaa sitä, että mikäli aiemmin mainittu ylikirjoitus ei ole valittuna niin työkalu osaa lisätä sisältöä poistamisen sijaan.

4.2 Tekninen ratkaisukuvaus

Työn teknisenä ratkaisukuvauksena toimii sovellusarkkitehtuurikuvaus, joka antaa karkean kuvan työkalun toimintaperiaatteesta. Alla olevassa kuviossa 5 on esitelty frontendin, backendin ja tietokannan väliset yhteydet.



Kuvio 5. Karkea arkkitehtuurikuvaus

4.3 Työn toteutuksesta

Työ toteutettiin kustomoiduksi portletiksi joka voidaan asettaa mille tahansa Liferay Portal -alustan sivustolle. Portletin tulee pystyä toimimaan halutulla tavalla riippumatta siitä mille sivustolle se laitetaan. Toisin sanoen sivustosta tulee pystyä luomaan export-paketti mistä tahansa portaali-ilmentymässä olevasta sivustosta sekä tuomaan paketti samaan tapaan.

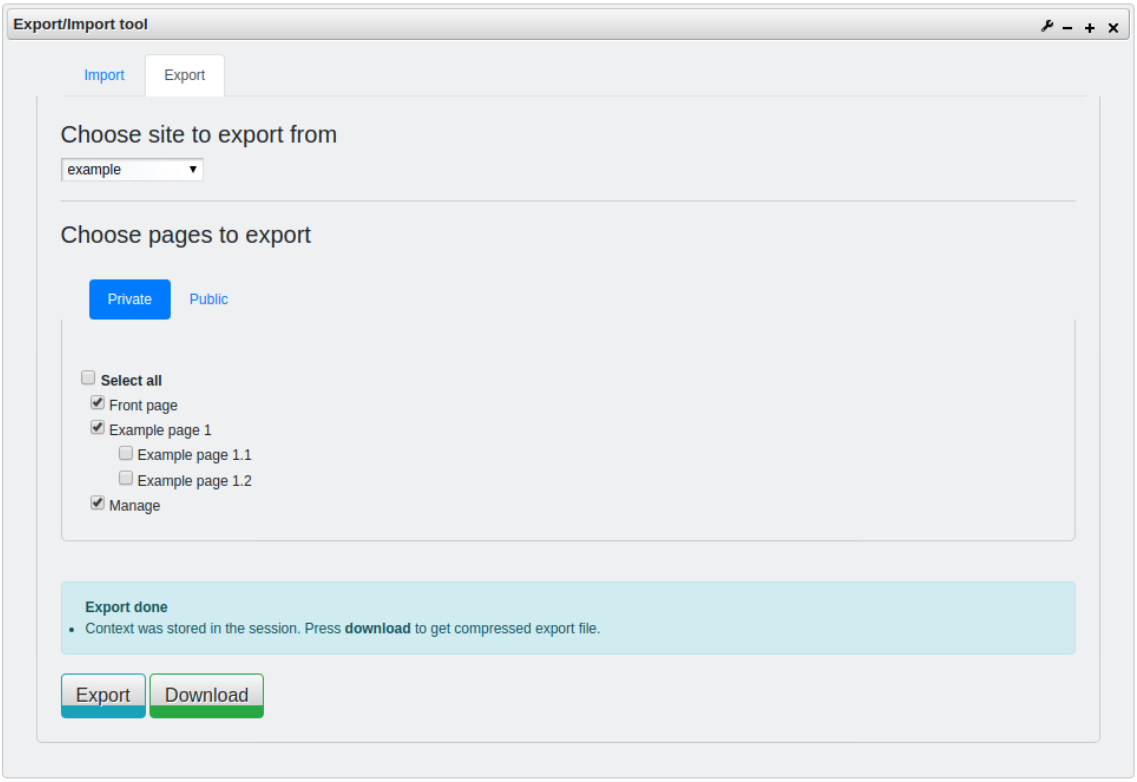
4.4 Käyttöliittymä

Työn toteutus alkoi käyttöliittymän puolelta. Vaikka työkalu onkin suunnattu kehittäjille, tuli käyttöliittymän olla selkeä ja informatiivinen. Käyttöliittymän toteutus voidaan jakaa kahteen erilliseen osaan: exporttiin ja importtiin.

Export

Siirtotyökalun export-osuuden voi jakaa neljään loogiseen osuuteen, jotka ovat nähtävissä kuvassa 4:

1. Portaali-ilmentymän asetukset, joissa valitaan sivusto, josta export tehdään sekä tullaanko se tekemään yksityisistä vai julkisista sivuista.
2. Hierarkinen listaus kaikista sivuista ja niiden alisivuista, joita valittu sivusto sisältää. Jokaista sivua kohden on valintaruutu, jolla sivu merkataan otettavaksi mukaan exportiin.
3. Viestien esittämiselle tarkoitettu osio. Tämä osio sisältää kaikki mahdolliset viestit mitä työkalun backend lähettää käyttöliittymään. Näihin lasketaan virheviestit, varoitukset sekä onnistumiset.
4. Nappulat. Exporttia luotaessa on kaksi nappulaa. Ensimmäinen nappula luo siirtotiedoston valmiiksi kontekstiin. Toinen nappula lataa siirtotiedoston.

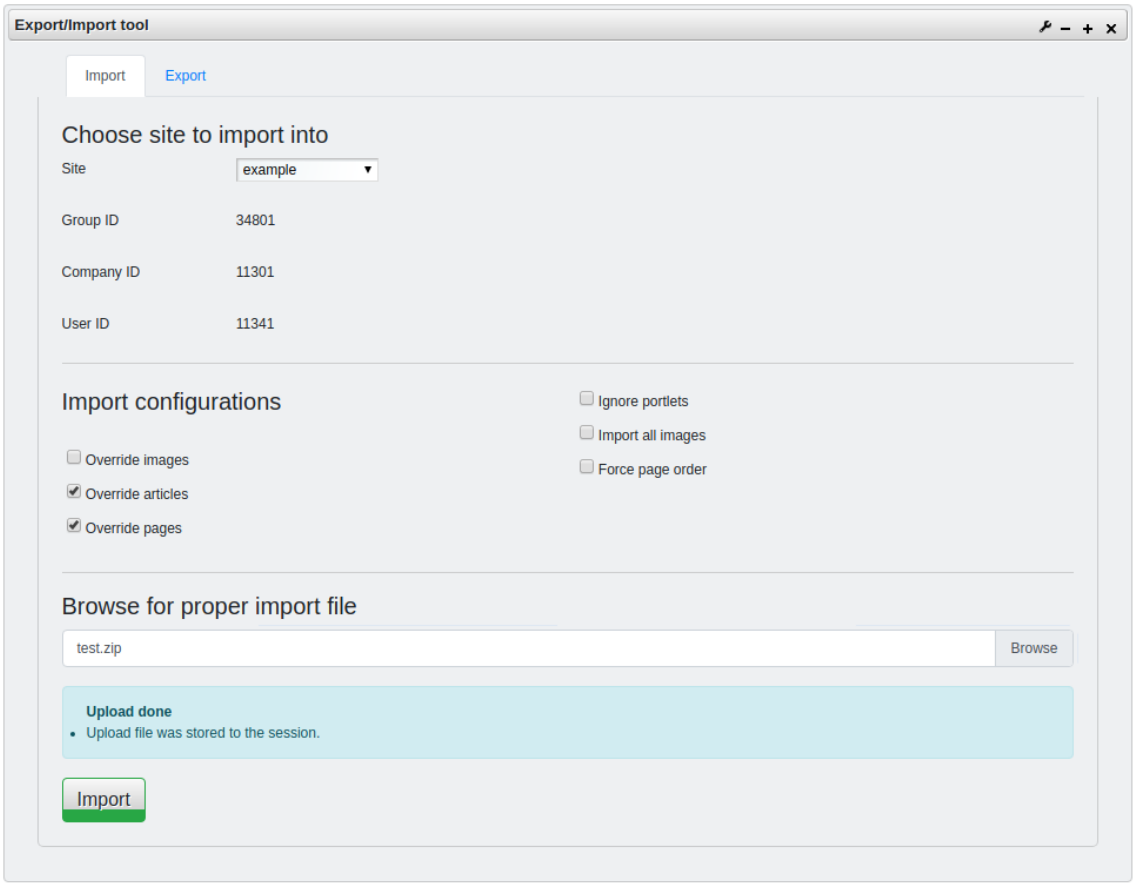


Kuva 4. Sisällönsiirtotyökalun export-osuuden käyttöliittymä (kuvakaappaus)

Import

Jokaista toiminnallista vaatimusta sekä siirtokirjaston ominaisuutta kohden tulee valintaruutu, jonka avulla määritetään millaisilla konfiguroinneilla sivuston data tuodaan kohdeympäristöön. Näihin ominaisuuksiin kuuluu vaatimusten lisäksi muitakin valintoja kuten dokumentti ja mediakirjastoon lisättävät tiedostot. Toisena tärkeänä ominaisuutena on syöttökenttä tiedostonhakemiselle. Lisäksi käyttöliittymässä on oma osio viesteille, joihin tulee mahdolliset virheviestit, joita backend lähettää käyttöliittymään mahdollisen virhetilanteen tullessa.

Nykyisellään siirto tehdään jokaiselle sisällölle erikseen ja virheen sattuessa siirrytään seuraavaan. Virheistä näytetään sisään/ulos viennin yhteydessä käyttäjälle jonkin verran tekninen, mutta kuitenkin järjestelmän käyttäjien ymmärrettävissä oleva virhe. Tätä voi käyttää selvittämään miten virheet on mahdollista korjata. Kuvassa 5 on kuvakaappaus toteutetusta käyttöliittymässä, jossa on näkyvissä import-osuuden ominaisuudet.



Kuva 5. Sisällönsiirtotyökalun import-osuuden käyttöliittymä (kuvakaappaus)

4.5 Backend

Työkalun backend-osuuden tärkein tehtävä on tuottaa halutunlainen export-paketti, jonka käyttäjä voi halutessaan ladata. Tämän lisäksi sen tulee pystyä vastaanottamaan samantyyppinen paketti ja muokkaamaan Liferayn tietokantaa siten, että paketissa tulleet sisällöt saadaan tuotua. Backendin tehtävä on myös hakea Liferayn tietokannasta tarvittavia tietoja exportin tai importin tekemiseen.

ID-arvojen haku

Ennen kuin tietokannasta voidaan hakea itse sisältöä, suoritetaan portaali-ilmentymän, sivustojen, käyttäjän ja sivujen ID-arvojen haku. Osaa näistä esitetään käyttöliittymässä kuten kuvassa 5 oleva ”Group ID” joka on tässä tapauksessa sivuston ID, jonka nimi näkyy yläpuolella. Sivuston ID vaaditaan siirtoon, jossa siirto suoritetaan eri sivustoon, kuin missä export/import-portletti sijaitsee.

Sivujen haku

Tietokannasta haetaan Liferay API -rajapinnan kautta tarvittavat sivut. Sivujen hakua käytetään hakemaan kuvan 4 mukaisesti kaikki valitun sivuston sisältämät sivut, näytettäväksi valintaruutulistassa.

Sivujen export

Sivujen exportissa luetaan ensin pyynnön mukana tullutta sivujen dataa. Tästä datasta luodaan viitteet sivuihin, joita tullaan käyttämään itse export-paketin luomisessa. Käyttäen kuvaa 4 esimerkkinä pyynnöstä luodaan viitteet sivuihin: Front page, Example page 1 ja Manage. Näistä sivuista otetaan talteen sivun nimi, URL-osoite ja tieto siitä onko sivu julkinen vai yksityinen. Exportista luodaan myös tässä vaiheessa konteksti, joka asettaa pohjatiedot exportille kuten sivuston ID, josta export tehdään. Viitesivut ja konteksti lähetetään export/import-kirjastolle.

Kirjastoon tulee siis tieto kontekstista eli sivustosta, josta export ollaan tekemässä sekä viitesivut, jotka sisältävät tarvittavan datan tietokantahakuihin. Tietokannasta haetaan näillä viitesivuilla olevat portletit ja niiden konfiguraatiot, web sisällöt, kuvat, teemat ja templaatit sekä sivustokohtaiset kustomoinnit kuten kustomoitu css tai js. Tämä kaikki data kompressoidaan ZIP-tiedostoksi joka valmiiksi ladattavaksi. Mikäli jokin vaihe meni pieleen tai esiintyi virhe, lähetetään niistä selkokielinen viesti. Export-paketti on nyt ladattavissa.

Sivujen import

Importissa backendiin saapuu käyttöliittymän applikaatiosta kutsu, joka sisältää importissa tarvittavat tiedot. Näihin kuuluu kaikki kuvan 4 mukaiset valintalaatikoilla valittavissa olevat konfiguraatiot sekä sivuston yleiset tiedot. Tämän lisäksi kutsussa tulee olla mukana export-paketti, joka oltaisiin tässä tapauksessa tuomassa kohdeympäristöön. Tämän jälkeen aloitetaan hieman sivujen exportin kaltaiset toimenpiteet, mutta päinvastoin.

Kutsun mukana tulleesta ZIP-paketista parsitaan export/import-kirjaston avulla tarvittava sivustokohtainen data, joka asetetaan kontekstiin. Näiden lisäksi kontekstiin asetetaan kutsun mukana tulleet konfiguraatiot, jotka määrittävät, että import on halutunlainen. Myös yleisiä tietoja tarvitaan, sillä niitä käytetään Liferay API-kutsuissa, jotka päivittävät tietokantaa. Seuraavaksi kirjasto suorittaa kontekstin tietojen perusteella tarvittavat API-kutsut, joilla kohdeympäristön data saadaan vastaamaan export-paketissa tuotuja lisäyksiä. Import on nyt valmis. Mikäli importissa tapahtui virhe, näkyvät nämä käyttöliittymässä.

5 TOTEUTUKSEN TULOS

Tämän luvun tarkoitus on käydä läpi toteutuksen hyödyt, haasteet sekä jatkokehitysmahdollisuudet. Luvun pohjana toimii työkalun kehityksen tukena olleen kokeneemman kehittäjän kommentit.

5.1 Saavutetut hyödyt

Työkalulla on mahdollista siirtää kaikki tai vain valitut sivut asetuksineen ulos järjestelmästä, versioida ne versionhallintaan ja viedä toisiin ympäristöihin. Tämä toiminnallisuus toimii jo todistetusti ja on osin ollut myös tuotantokäytössä. Työkalu täyttää siis sille annetut tavoitteet. Työkalulla saavutetaan osittain automatisoitu sisällönsiirto, joka nopeuttaa kehityksen etenemistä ja säästää resursseja. Lisäksi siirron automatisoinnilla vähennetään inhimillisten virheiden mahdollisuutta.

5.2 Haasteet

Suurimpana haasteena oli käyttöliittymän suunnittelussa laaja joukko mahdollisia virhetiloja, joita ympäristöjen erilaisuus voi aiheuttaa ja miten näistä saadaan käyttäjälle järkevä ja ymmärrettävä ilmoitus. Lisäksi virheet eivät saa olla siirtoa kokonaisuudessaan estäviä, koska järjestelmään vietävät muutokset eivät ole transaktionaalisia (ja näin peruttavia) eikä myöskään tavallisesti estä muiden sisältöjen siirtoa. Tämän lisäksi käytettävät teknologiat olivat minulle uusia, joten niihin täytyi ensin tutustua. Tämä kuitenkin toimi hyvänä oppimiskokemuksena.

5.3 Jatkokehitys

Jatkokehitys kohdistuu pääsääntöisesti automatisoinnin parantamiseen ja työkalun ominaisuuksien laajentamiseen. Siirtotoiminnallisuus on kuitenkin sellaisenaan jo käyttökelpoinen ja käytössä muutaman asiakkuuden kanssa. Koska työkalulle on kysyntää jatkossakin, on siihen suunniteltu jo alustavasti laajennuksia.

Näihin mahdollisiin laajennuksiin kuuluu muun muassa kokonaan uusien ympäristöjen perustaminen. Uusia laajennuksia varten tarvitaan siis uusia toiminnallisuuksia taustalle kuin myös käyttöliittymään.

Opinnäytetyössä toteutettu työkalu toimii pohjana ympäristöjen välisen sisällönsiirron automatisoinnille, johon toimeksiantajalla on suuri kiinnostus käyttää resursseja ympäristöjen määrän lisääntyessä.

6 POHDINTA

Opinnäytetyön tavoite oli tutkia sisällönhallintaa yleisestä näkökulmasta ja vaikka se ei olekaan kovin yksinkertainen aihe, se tuli kuitenkin käsiteltyä karkeasti. Työssä käytiin läpi myös erilaisia sisällönhallintajärjestelmille tyypillisiä kategorioita ja niitten ominaisia piirteitä. Tämän lisäksi työssä sovellettiin Liferay Portal -portaalialustaa, jonka vuoksi käytiin läpi siihen liittyviä ominaisuuksia.

Opinnäytetyössä läpikäytiin myös DTAP-kehitykselle ominaisia järjestelmäympäristöjä, jonka tavoitteena oli tuoda esiin ympäristöjen mahdollinen määrä ja se, miten ne poikkeavat toisistaan. Tämä alustaa tarpeen sisällön hallitulle siirrolle, joka toimii pohjana opinnäytetyössä toteutetulle työkalulle.

Yhteenvetona opinnäytetyön voidaan sanoa täyttäneen tarkoituksensa ja opinnäytetyössä toteutettu työkalu täyttää sille asetetut tavoitteet. Työssä suunniteltiin ja toteutettiin Export-/Import-työkalu, joka helpottaa kehittäjiä siirtämään sisältöä ympäristöstä toiseen. Lisäksi jatkokehitykselle asetettuja määritelmiä on mietitty ja sisällönsiirron automatisointia voidaan aina parantaa. Tärkeintä kuitenkin on se, että osa sisällönsiirron työtä pystytään parantamaan ja se tuottaa hyötyä kehitykselle jatkossakin.

LÄHTEET

Associate of Information and Image Management (AAIM) N.d. What is Web CMS (or WCM)? Luettu 11.5.2020. https://www.aiim.org/What-is-Web-20#

Barker, Deane. Web Content Management: Systems, Features, and Best Practices. First edition. Beijing, China: O'Reilly, 2016. Print.

Boiko, Bob. *Content Management Bible*. 2nd ed. Indianapolis (Ind.): Wiley, 2005. Print.

Bynder N.d.Luettu 11.5.2020 https://www.bynder.com/en/what-is-digital-asset-management/

Juha Anttila. N.d. ECM – Enterprise Content Management. Luettu 3.5.2020 https://www.iitc.fi/ecm

Liferay N.d.a Introduction to Portlet Develoment. Luettu 3.5.2020 https://portal.liferay.dev/docs/7-0/tutorials/-/knowledge_base/t/portlets

Liferay N.d.b Staging Page Publication. Luettu 11.5.2020 https://help.liferay.com/hc/en-us/articles/360017873352-Staging-Page-Publication

Mike Johnston. N.d. CMS or WCM – Which is which? cmscritic.com. Luettu 25.5.2020.

Mate Thurzo, Liferay DEVCON 2014 – Staging in Liferay 6.2 – Mate Thurzo. Liferay. Youtube-video. Julkaistu 13.11.2014. Katsottu 26.4.2020. https://www.youtube.com/watch?v=RQolBqZLGiw

Visser, Joost. *Building Software Teams: Ten Best Practices for Effective Software*. First edition. Sebastopol, California: O'Reilly Media, 2017. Print.