Arvo Koskikallio

HELP MANAGEMENT SYSTEM

Tekniikka
2022

ALKUSANAT

Viimeiset 4 vuotta Vaasan Ammattikorkeakoulussa (VAMK) ovat opettaneet minulle paljon sekä tietotekniikasta että elämästä itsestään. Vaikka koronapandemia ja sen tuoma vaihtelevan tasoinen etäopiskelu usein pisti kapuloita rattaisiin, niin oppimisen kuin elämisenkin kannalta, voin sanoa olevani tänään paljon vakaampi, onnellisempi ja tyytyväisempi ihminen kuin ikinä ennen aikaani VAMKissa ja toivon, että tämä oppimisen prosessi jatkuu yhtä lailla tulevaisuudessakin.

Haluan osoittaa kiitokseni projektinjohtaja Jesse Ikolaa ja kehittäjäkollegaani Juhana Itkosta kohtaan, joiden kanssa projekti tuli kunnialla loppuun. Haluan myös kiittää softaguru Jesse Båtmania, jota ilman olisin täyttänyt työhakemuksia paljon pitempään.

Lopuksi haluan myös kiittää perhettäni, ilman perheen kannustusta ja ohjausta en näitä sanoja tänään kirjottaisi.

Vaasassa 30.3.2022

Arvo Koskikallio

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Arvo Koskikallio
Opinnäytetyön nimi	Help Management System
Vuosi	2022
Kieli	suomi
Sivumäärä	34
Ohjaaja	Anna-Kaisa Saari

Help Management System on Jubicin ja Prima Powerin kehittämä alusta, jota käytetään Prima Powerin sisäisten ohjetiedostojen, ns. ”Help-tiedostojen”, käsittelyyn. Help-tiedostot ovat yksinkertaistettuna HTML-pohjaisia tiedostoja, jotka kertovat Prima Powerin koneiden toiminnasta koneen ja koneeseen liittyvien ominaisuuksien, ns. ”Parametrien” perusteella.

Idea Help Management Systemin suunnitteluun alkoi halusta tuoda help-tiedostojen käsittelyssä käytettävät ominaisuudet yhdelle alustalle. Alustan tarkoitus oli tuoda kaikki help-tiedostot yhteen paikkaan ja tehdä niiden hallinnoimisesta mahdollisimman helppoa sekä tiedostojen luomiselle että niiden lukemiselle työtilanteessa.

Alustan frontend-puoli rakennettiin TypeScriptillä ja JavaScriptin React-kirjastolla, ja backend-puoli .NETillä, C#illa. Sovelluksen tietokantaa hallinnoitiin Microsoftin SQL Serverillä. Alustaan ladattavat tiedostot säilytettiin Microsoftin Azuren Blob Storage-pilvipalvelussa.

Tässä työssä kuvailtu suunnitteluprosessi johti toimivaan alustaan, jota voidaan käyttää uusien help-tiedostojen luomiseen ja vanhojen muokkaamiseen ja hallinnoimiseen. Tähän projektiin tullaan tulevaisuudessa lisäämään toiminnallisuuksia, jotka parantavat Prima Powerin toiminnan kulkua entisestään.

Avainsanat ohjelmistokehitys, .NET, Azure, SQL Server, GrapesJS

VAASAN AMMATTIKORKEAKOULU, UNIVERSITY OF APPLIED SCIENCES
Tietotekniikka

ABSTRACT

Author	Arvo Koskikallio
Title	Help Management System
Year	2022
Language	Finnish
Pages	34
Name of Supervisor	Anna-Kaisa Saari

Help Management System is a software platform developed by Jubic and Prima Power. The platform is used to manage Prima Power's instruction files, known within the company as helpfiles. Helpfiles are HTML files that display information about Prima Power's devices based on parameters linked to them.

The idea for the development of Help Management System was formed from a desire to bring all the tools used to edit helpfiles onto a single, unified platform. The goal of the platform was to bring all the helpfiles to one place and make managing them as easy as possible both for creating new helpfiles and viewing them in a work environment.

The frontend of the platform was developed using TypeScript, as well as the JavaScript library React. The Backend was developed with .NET, using C# as the programming language. The application database was managed using Microsoft SQL Server. Media uploaded to the platform was stored in Azure Blob Storage, which is a cloud storage solution developed by Microsoft.

The development process described in this document resulted in a working platform that can be used to create new helpfiles and edit and maintain existing ones. This project will be expanded on in the future to include more functionalities that further improve the workflow for Prima Power's operations.

Keywords software development, .NET, Azure, SQL Server, GrapesJS

SISÄLLYS

LYHENNELUETTELO

API	Application Programming Interface. Ohjelmointirajapinta.
Blob	Binary Large Object. Binääridataa.
CI/CD	Continuous integration/Continuous deployment. Jatkuva integraatio/Jatkuva toimitus.
CLR	Common Language Runtime. .NET Frameworkin virtuaalikonekomponentti.
EULA	End User Licence Agreement. Loppukäyttäjän lisenssisopimus.
GUID	Globally Unique Identifier. Universaalinen uniikki tunniste.
HTTP	Hypertext Transfer Protocol. Hypertekstin siirtoprotokolla.
HTTPS	Hypertext Transfer Protocol Secure. Hypertekstin siirtoprotokolla, joka salaa datan käyttäen SSL:ää.
JS	JavaScript. Ohjelmointikieli.
MPE	Master Parameter Editor. Prima Powerin ohjelmisto.
REST	Representational State Transfer. Ohjelmointirajapinta-arkkitehtuuri.
SSL	Secure Sockets Layer. Salausprotokolla.
VNet	Virtual Network. Virtuaalinen verkko.
YAML	YAML Ain’t Markup Language. Merkintäkieli.

KUVALUETTELO

1 JOHDANTO

1.1 Yleiskuvaus

Tämä opinnäytetyö esittelee Jubicin ja Prima Powerin yhteistyöllä suunnitellun ja toteuttaman Prima Powerin sisäiseen käyttöön tarkoitetun Help Management System-alustan.

Alustan suunnitteluprosessissa käytettiin hyväksi suunnittelun Agile-menetelmää. Suunnitteluprosessi koostui lukuisista kahden viikon sprinteistä, joiden jälkeen suunnitteluryhmä kokoontui käymään läpi aikaansaannokset ja suunnittelemaan seuraava sprint. Suunnitteluryhmä piti kokouksia myös keskellä sprinttiä, joissa keskusteltiin työn etenemisestä ja mahdollisista ongelmista. Näiden säännöllisien kokouksien lisäksi pidettiin myös pienempiä palavereita, kun joitain suurempia ongelmia syntyi. Suunnittelun aikana pidettiin myös säännöllisin aikavälein palavereita Prima Powerin kanssa. Näissä palavereissa käytiin läpi uusia ominaisuuksia sekä toiminnallisuudessa ilmentyneitä epäkohtia.

1.2 Jubic Oy

Jubic Oy on vuonna 2015 perustettu yritys, joka tällä hetkellä työllistää yli 15 työntekijää pääasiassa Vaasan alueella. (Jubic, 2022) Vuonna 2021 yrityksen liikevaihto oli 933 000 euroa. Liikevaihdon keskimääräinen vuotuinen kasvuprosentti vuodesta 2016 vuoteen 2021 on 33,4 %. (Suomen Asiakastieto, 2022)

Ylivoimaisesti suurin osa Jubicin liiketoiminnasta liittyy ohjelmistokehitykseen. Yritys suorittaa projekteja avaimet käteen -periaatteella, sekä tarjoaa R&D-palvelua muille yrityksille. Jubic tarjoaa ohjelmistokehityksen lisäksi myös konsultointia esimerkiksi projektinhallintaan, ohjelmistohankintoihin ja tietoturvaan liittyvissä asioissa. Tämän lisäksi yritys tarjoaa asiakkailleen myös pilvipalvelua.

1.3 Prima Power

Prima Power on italialaisen Prima Industrie -konsernin tytäryhtiö, joka toimii laserkoneiden ja ohutlevyjen työstökoneiden alalla. Yritys sai alkunsa Suomessa vuonna 1969 nimellä Finn-Power. Prima Industrie osti Finn-Powerin vuonna 2008, ja vuonna 2011 oston seurauksena syntyi uusi Prima Power -brändi.

Tänä päivänä Prima Industrie -konsernilla on liiketoimintaa yli kahdeksassakymmenessä maassa, joista Prima Power toimii yli kolmessakymmenessä. Priman pääkonttori sijaitsee Italian Collegnossa ja Suomen toimipaikka on Seinäjoella. (Prima Power, 2022)

2 TEKNOLOGIAT

Tässä luvussa käydään läpi projektissa käytettyjä teknologioita. Alustan backend kehitettiin .NETiin C#illa, frontend kehitettiin käyttäen Reactia ja TypeScriptiä. Alustan vaatiman datan hakemiseen käytettiin kahta Microsoft SQL serverin tietokantayhteyttä, yhtä Prima Powerin tietokantaan ja toista itse alustaa kehitettyyn tietokantaan. Alustaa varten kehitettyä tietokantaa pystyttiin kehitysvaiheessa hallinnoimaan Dockerin avulla. Alustan käyttäjät voivat muokata alustalle tallennettuja help-tiedostoja suoraan selaimessa GrapesJS-kehyksen avulla.

2.1 .NET

.NET (aiemmin .NET Core) on vuonna 2014 julkaistu avoimen lähdekoodin ohjelmistokehys, joka perustuu Microsoftin vuonna 2002 luomaan .NET Frameworkiin. .NET Frameworkin ja .NETin suurin ero on se, että .NET Framework tukee ainoastaan Windowsia, kun taas .NET tukee useampia käyttöjärjestelmiä.

.NET koostuu CLR-suoritusympäristöstä sekä luokkakirjastoista. Luokkakirjastot sisältävät uudelleenkäytettävää koodia, joita voidaan käyttää hyväksi ohjelmistokehityksessä. .NET tukee monia eri ohjelmointikieliä, joista käytetyimmät ovat C# ja Visual Basic. (Microsoft, 2021)

2.2 Azure Blob Storage

Azure Blob Storage on Microsoftin pilvipalvelu, jota voidaan hallinnoida HTTP- tai HTTPS-protokollaa käyttäen REST APIn avulla. (Microsoft, 2022) Datan siirtäminen Blob Storageen ja sen vastaanottamisen Blob Storagesta toteutetaan käyttämällä hyväksi Microsoftin .NETin luokkakirjastoihin kuuluvaa Azure Storage Blobs -kirjastoa. Blob Storage on optimoitu rakenteettoman datan tallentamiseen. Rakenteeton data on dataa, joka ei sitoudu tiettyyn datamalliin. Esimerkiksi binääridata on rakenteetonta dataa. (Microsoft, 2021) Datan rakenteettomuuden ansiosta Blob

Storageen on vaivatonta ladata erityyppisiä tiedostoja ja erottaa tiedoston metadata, esimerkiksi tiedoston datatyyppi, latausvaiheessa ja pitää nämä tiedot lokaalissa tietokannassa.

Tiedostojen organisointiin Blob Storagessa käytetään containereita, joiden avulla data organisoidaan tiedostojärjestelmän kansioiden tapaan. Kuva 1 C#-koodista esittelee tiedoston sisään ja ulos lataamista Azure Storage Blobs-kirjastolla. Funktioille annetaan arvot *connectionString,* jolla määritetään ja autentikoidaan yhteys Blob Storageen, ja *containerName*, jolla määritetään containerin nimi. Lisäksi upload-funktiolle annetaan ladattava tiedosto ja download-funktiolle ladattavan tiedoston nimi.

Lataamiseen käytetään hyväksi Azure Storage Blobs-kirjaston *BlobContainerClient*- ja *BlobClient*-luokkia. *BlobContainerClient*-luokka mahdollistaa containerin hallinnan ja *BlobClient*-luokka itse blobin hallinnan. Esimerkissä käsitellään tiedostoa *Stream*-objektina ja se nimetään GUIDilla latausprosessin aikana. GUID on 128-bittinen numero, jota käytetään tiedostojen identifioimiseen. (Beal, 2002)

```
static void UploadToBlobStorage(
    string connectionString,
    string containerName,
    Stream fileStream
)
{
    BlobContainerClient containerClient =
      new BlobContainerClient(connectionString, containerName);
    Guid guid = Guid.newGuid();
    blockBlob.UploadBlob(guid.ToString(), fileStream);
}

static MemoryStream DownloadFromBlobStorage(
    string connectionString,
    string containerName,
    Guid guid
)
{
    BlobContainerClient containerClient =
      new BlobContainerClient(connectionString, containerName);
    BlobClient blobClient = containerClient
      .GetBlobClient(guid.ToString());
    MemoryStream returnStream = new MemoryStream();
    blobClient.DownloadTo(returnStream);
    returnStream.Seek(0, System.IO.SeekOrigin);

    return returnStream;
}
```

Kuva 1. C# Azure Blob Storage-esimerkki.

2.3 SQL Server

SQL Server (Microsoft SQL Server) on relaatiotietokantojen luomiseen ja hallinnointiin tarkoitettu ohjelmisto. SQL Server on Microsoftin tuote, joten sen integroiminen .NET-backendiin on helppoa Microsoftin tarjoamien kirjastojen avulla. Tietokannan datan hakeminen ja lisääminen suoritetaan SQL-kyselykielellä. Kuvassa 2 on esimerkki SQL-lauseista, joiden avulla voidaan lisätä ja hakea tietoa tietokannasta. Koodin ensimmäinen komento lisää tietokannan Customers-tauluun uuden rivin lähetetyn datan mukaan. Seuraava komento taas hakee kaikki sarakkeet Customers-taulusta, missä taulun City-sarakkeen arvo on sama kuin kysytty

arvo, ja järjestää palautettavat sarakkeet FirstName-sarakkeen mukaan aakkosjärjestyksessä.

```
INSERT INTO Customers (FirstName, LastName, City) VALUES (
    'George',
    'Carter',
    'Newcastle'
);

SELECT * FROM Customers WHERE City='Newcastle' ORDER BY
FirstName;
```

Kuva 2. SQL-esimerkki

2.4 Docker

Docker on avoimen lähdekoodin alusta, joka mahdollistaa erilaisten ohjelmistojen pakkaamiseen kontteihin (containers). Kontit ovat standardisoituja komponentteja, jotka yhdistävät ohjelmiston lähdekoodin käyttöjärjestelmän kirjastoihin sekä tarvittavat riippuvuuksiin. Docker mahdollistaa näin ohjelmistojen siirtämisen tietokoneelta toisille ilman tarvetta huolehtia eri koneiden yhteensopivuuksista. (IBM, 2021)

Dockerin käyttö lokaalin tietokannan hallinnointiin onnistuu dockerin compose-työkalun avulla. Compose on joustava työkalu monikonttisten docker-ohjelmien suorittamiseen ja compose-työkalun konfigurointia voidaan muuttaa ohjelman juureen luotavan YAML-tiedoston avulla. (Docker, 2022) Kuva 3 demonstroi compose-työkalun konfigurointia tietokannan hallitsemiseen. Tiedosto asettaa SQL Server-tietokannan ohjelman yhdeksi, ja tässä esimerkissä ainoaksi palveluksi. Tietokantapalvelun asetetaan kuuntelemaan tietokoneen porttia 1433, sille asetetaan salasana ja hyväksytään EULA-sopimus. Konfiguraation jälkeen tietokantaa voidaan hallinnoida komentoriviltä ”docker-compose” -komentoa käyttäen.

```yaml
version: '3.4'

services:
    database:
        image: "mcr.microsoft.com/mssql/server"
        ports:
            - "1433:1433"
        environment:
            SA_PASSWORD: "testingpassword"
            ACCEPT_EULA: "Y"
```

Kuva 3. Docker-compose.yml, esimerkki

2.5 ReactJS

ReactJS (tai pelkkä React) on Facebookin kehittämä avoimen lähdekoodin JavaScript-kirjasto käyttöliittymien rakentamiseen. Reactilla tehdyt käyttöliittymät rakentuvat React-komponenteista, jotka voivat hallita omaa tilaansa (state). Esimerkiksi Reactilla tehty käyttäjän profiilikäyttöliittymä koostuu form-elementistä, joka pitää sisällään käyttäjän tiedot. Kyseisellä elementillä on sisäkkäisinä (nested) elementteinä kaikki esitettävän datan tekstikentät. Form-elementti kommunikoi oman tilansa tekstikentille, jotka esittävät datan käyttäjälle. Form-elementin ja tekstikenttien välillä olevaa suhdetta kutsutaan parent-child-suhteeksi, jossa form-elementti on parent-elementti ja tekstikentät child-elementtejä. (Codeacademy, 2021)

2.6 TypeScript

TypeScript on Microsoftin kehittämä, JavaScriptiin ominaisuuksia lisäävä ohjelmointikieli, joka mahdollistaa staattisien tyyppien käyttämisen. TypeScript on tätä ominaisuutta lukuun ottamatta lähes identtinen JavaScriptin kanssa, joten jo kirjoitettu JavaScript-koodi on helppo muuttaa TypeScriptiksi. (Typescriptlang, 2022)

TypeScriptin staattisien tyyppien luominen ja toiminta on havainnollistettu kuvassa 4. Koodissa luodaan objekti tyyppiä *user*, jolla on elementit *firstName*, *lastName* ja *role*. Objektilta yritetään tulostaa konsoliin *name*-elementti. Koska tällaista elementtiä ei *user*-tyypillä ole, TypeScript palauttaa tästä kertovan virheilmoituksen.

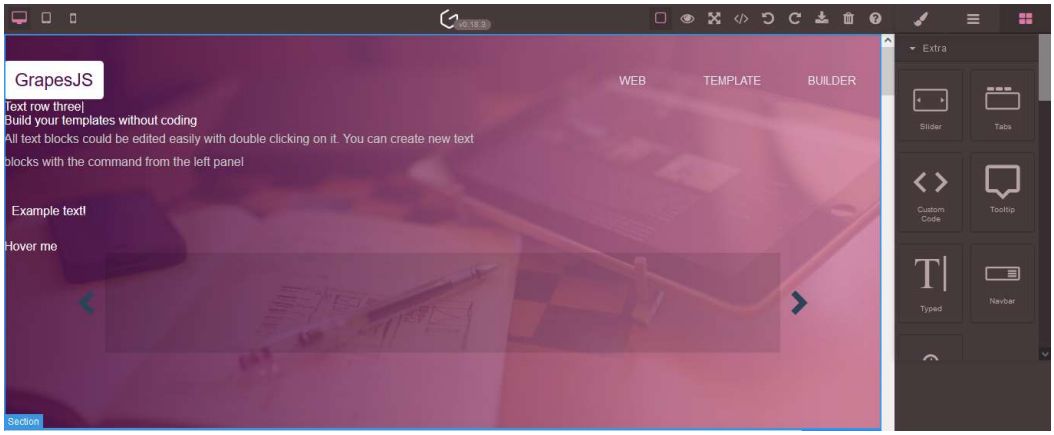
```
const user = {
    firstName: "Angela",
    lastName: "Davis",
    role: "Professor",
  }
  console.log(user.name)

Property 'name' does not exist on type '{ firstName: string;
 lastName: string; role: string; }'.
```

Kuva 4. TypeScript-esimerkki (Typescriptlang, 2022)

2.7 GrapesJS

GrapesJS on avoimen lähdekoodin ohjelmistokehys, jota voidaan käyttää HTML-rakenteiden luomiseen kirjoittamatta riviäkään koodia. GrapesJS tarjoaa käyttäjälle erilaisia HTML-komponentteja, joita hän voi lisätä ja muokata vapaasti raahaamalla niitä eri paikkoihin HTML-muokkausikkunan sisällä. Kuva 5 esittää GrapesJS:n editointinäkymää. Kuvan oikeassa reunassa sijaitsee valikko, jolla käyttäjä voi lisätä komponentteja editorissa näkyvään HTML-sivuun. Valikosta näkyy myös paljon muuta tietoa sivusta.



Kuva 5. GrapesJS-editointi (GrapesJS, 2022)

3 HELP MANAGEMENT SYSTEM -ALUSTAN KUVAUS

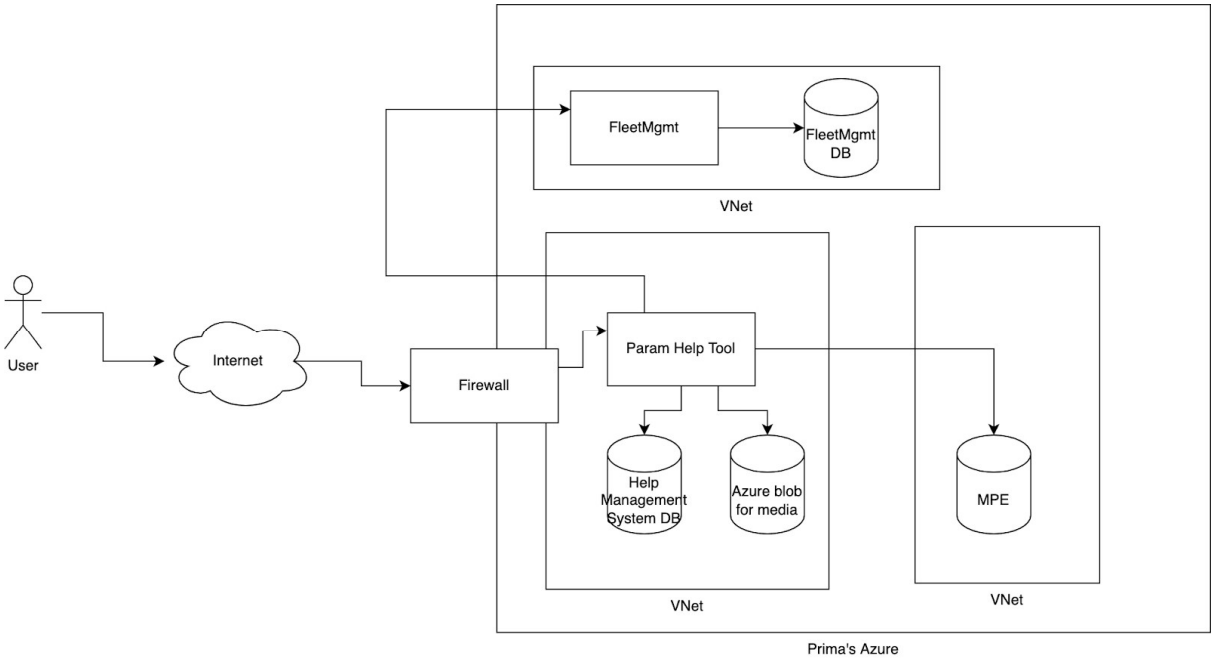
Tämä luku käy läpi alustan rakennetta enimmäkseen käyttäjän näkökulmasta. Tässä luvussa käydään läpi myös ohjelman eri toimintoja sekä integraatiota muihin alustoihin.

3.1 Arkkitehtuurikaaviot

Arkkitehtuurikaaviot ovat kaavioita, jotka auttavat kuvaamaan ohjelmistoalustan rakennetta ja alustan eri komponenttien suhteita toisiinsa. Seuraavat kaaviot havainnollistavat alustan yhteyksiä muihin alustoihin, sekä itse Help Management Systemin rakennetta.

3.1.1 Integraatiokaavio

Kuvassa 6 on esitelty alustan integraatiot muihin alustoihin ja tietokantoihin. Käyttäjä luo yhteyden Help Management Systemiin (kaaviossa nimellä Param Help Tool) Internetin kautta. Help Management System on käynnissä Azuren VNet-verkossa. VNetit ovat pilvessä sijaitsevia virtuaalisia verkkoja. Kaavion kolme VNet-verkkoa ovat kaikki laajemman Prima Powerin Azure-pilviympäristön osia.



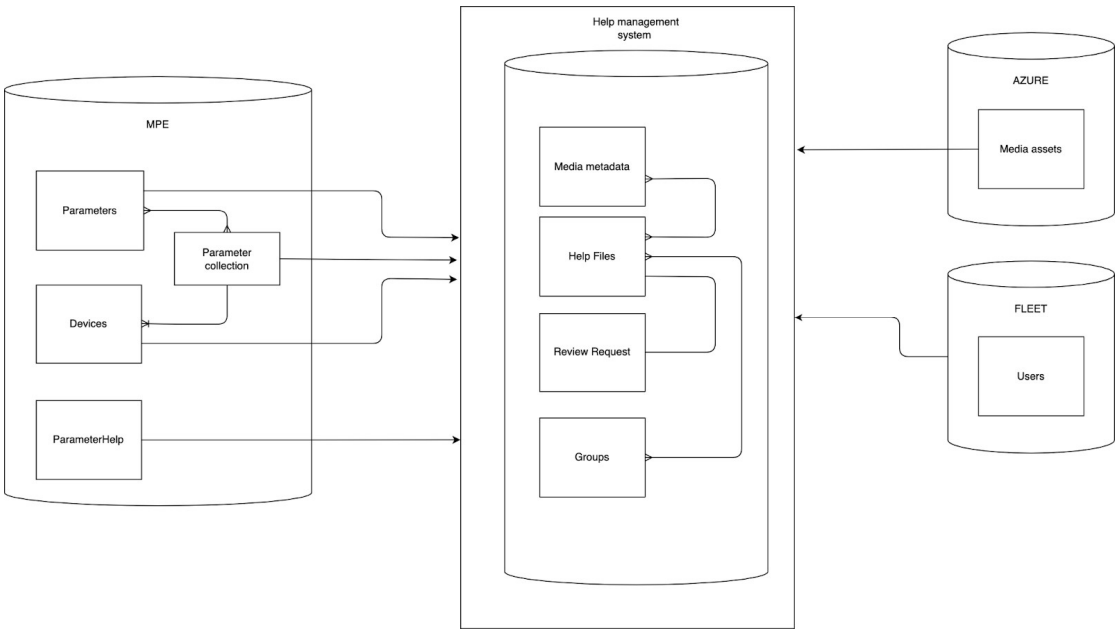
Kuva 6. Integraatiokaavio (Ikola, 2022)

Help Management Systemillä on sisäisen VNet-verkon sisällä Azure Blob Storage-yhteys. Blob Storagessa säilytetään alustan käyttämä media ja help-tiedostot. Samassa VNet-verkossa on myös sisäinen tietokanta, johon tallennetaan median ja help-tiedostojen metadata sekä tiedot tiedostojen välisistä suhteista. Help Management Systemillä on myös yhteydet Prima Powerin Master Parameter Editor-alustan (MPE) tietokantaan. MPE:n tietokanta pitää sisällään tietoja Prima Powerin koneista. Help Management Systemille relevanttia dataa on näiden koneiden ja niihin liittyvien parametrien metadata. Help Management System käyttää myös hyväkseen PrimaPowerin Fleet-ohjelmiston APIa käyttäjien autentikointiin. Fleet on Prima Powerin sisäinen alusta, joka hallinnoi käyttäjien oikeuksia Prima Powerin muihin alustoihin. Help Management Systemiin kirjautuminen suoritetaan myös Fleetin ylläpitämillä käyttäjätunnuksilla.

3.1.2 Tietokantakaavio

Kuvassa 7 näkyvä kaavio kuvaa MPE:n, Help Management Systemin, Azure Blob Storagen ja Fleetin tietokantojen välisiä suhteita. MPE-yhteyden avulla saadaan

Help Management Systemiin parametrirakenne sekä yhteydet parametrien ja koneiden välillä. MPE:ssa sijaitsee myös Prima Powerin vanhat help-tiedostot, jotka siirretään Help Management Systemin Azure Blob Storageen erillisellä import-skriptillä. Skripti käy läpi kaikki MPE:ssä sijaitsevat help-tiedostot, järjestää tietorakenteen uudelleen sellaiseen muotoon, jota Help Management Systemin help-tiedostojen lataamiseen tarkoitettu API-endpoint osaa käsitellä ja lähettää tiedostot endpointille.



Kuva 7. Tietokantakaavio (Ikola, 2022)

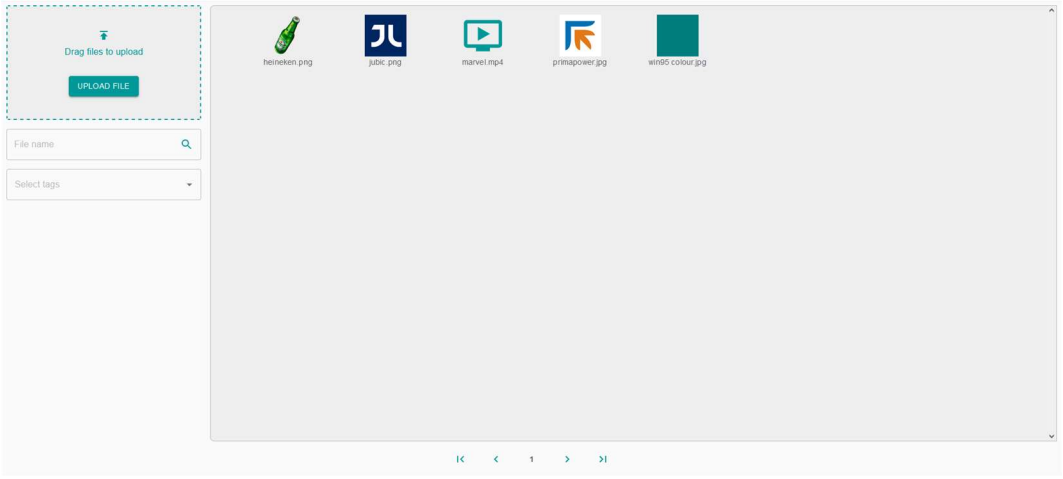
3.2 Ominaisuudet

Tämä osio selittää alustan eri osien toiminnot ja demonstroi niiden käyttöä käyttäjän näkökulmasta. Alusta koostuu käyttäjän näkökulmasta kolmesta eri osasta:

Mediakirjastosta, help-tiedostojen hallinnasta, ja arvioinnista. Alustaan on myös rakennettu API export-endpoint help-tiedostojen viemistä varten.

3.2.1 Mediakirjasto

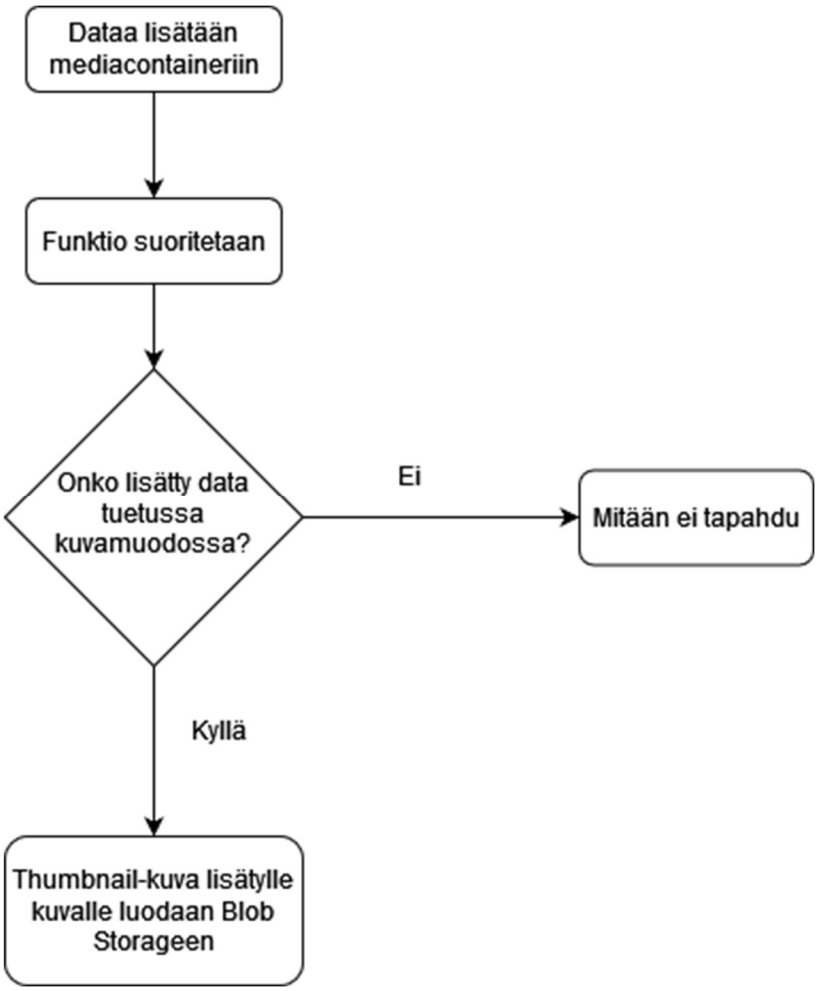
Mediakirjasto on Prima Powerin mediatiedostojen hallinnointiin tarkoitettu kokonaisuus. Mediakirjastoon tallennettava media tallennetaan Azure Blob Storageen. Käyttäjän näkökulmasta mediakirjastoa käytetään mediatiedostojen lataamiseen ja niihin liitettävien tagien hallinnoimiseen. Mediakirjastossa käyttäjä voi myös päivittää olemassa olevaa kuvaa, sekä muuttaa jo ladattujen kuvien metadataa. Kuvia voidaan hakea kuvan nimen tai tagien perusteella. Kuva 8 esittää mediakirjaston käyttöliittymän perusnäkymää.



Kuva 8. Mediakirjaston perusnäkymä

Esillä olevat kuvien nimet tuodaan käyttöliittymään Help Management Systemin sisäisestä tietokannasta. Itse kuvien hakeminen Azuren Blob Storagesta voi olla kovinkin hidasta, varsinkin jos kuvat ovat korkealaatuisia. Tämä ongelma ratkaistiin luomalla Azure-Funktio thumbnail-kuvien generointiin. Thumbnail on käytännössä pienikokoinen versio suuremmasta kuvasta, jonka hakeminen Blob Storagesta on huomattavasti nopeampaa. Joka kerta, kun Help Management Systemiin ladataan sisään kuva, funktio luo sille thumbnail-kuvan, jonka se tallentaa erilliseen Blob Storage Containeriin samalla nimellä kuin alkuperäinen kuva. Tämä tekee thumbnail-kuvien yhdistämisestä alkuperäisen kuvien tietoihin helppoa.

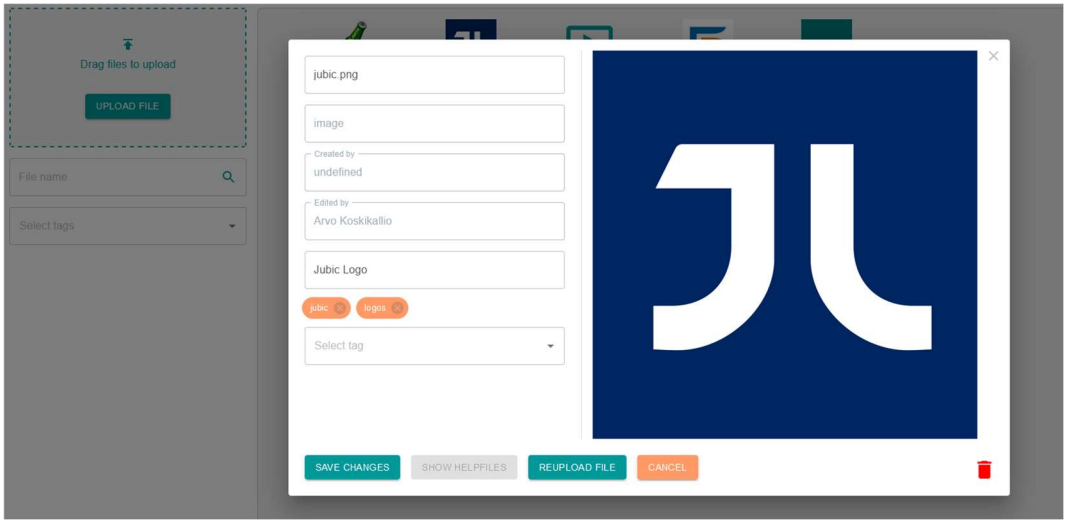
Azure-Funktiot on pilvipalvelu, joka voidaan konfiguroida suoritettavaksi tietyn asian tapahtuessa. Tässä toteutuksessa funktio suoritetaan joka kerta, kun Help Management Systemin Blob Storage -mediacontaineriin lisätään uutta dataa. Ulkoisen Azure-Funktion avulla thumbnail-kuvien luomiseen käytetyt resurssit erotetaan Help Management Systemin resursseista, ja ohjelmalle jää enemmän resursseja muihin toimintoihin. Kuvassa 9 oleva vuokaavio visualisoi Funktion toimintaa.



Kuva 9. Thumbnail-funktion suorittaminen

Kuten kuvasta 10 näkyy, median metadatasta käyttäjä voi muokata kuvan nimeä sekä kuvatekstiä. Viimeinen kuvaa muokanneen käyttäjän käyttäjänimi tallenne-

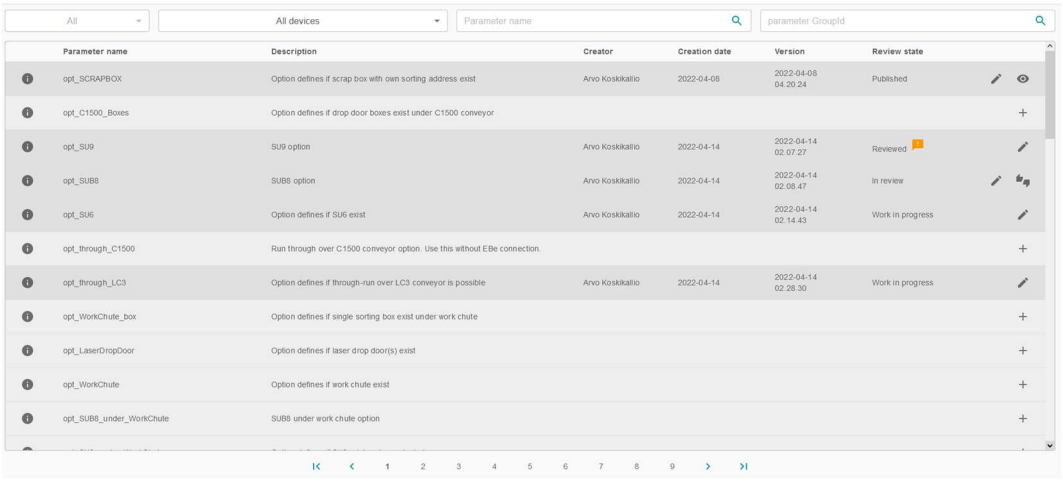
taan kuvan metadataan. Kuvaan voi myös lisätä ja poistaa tageja. Uusia tageja voidaan kirjoittaa vapaasti tag-kenttään, tai valita jo olemassa olevia pudotusvalikosta.



Kuva 10. Median hallinnointinäkymä

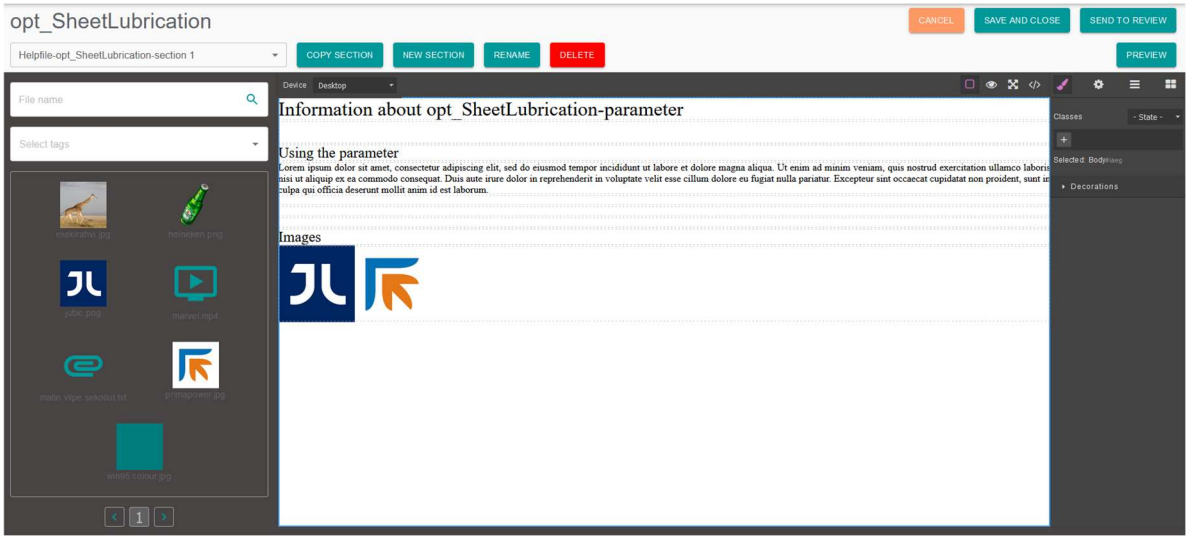
3.2.2 Help-tiedostojen hallinta

Help-tiedostojen hallintanäkymässä voidaan lisätä ja muokata help-tiedostoja. Help-tiedostot erotellaan toisistaan parametrin perusteella. Parametriin voi olla liitettynä korkeintaan yksi lopullinen help-tiedosto ja yksi keskeneräinen muokattava help-tiedosto. Kuva 11 esittää help-tiedostojen hallintanäkymää.



Kuva 11. Help-tiedostojen hallintanäkymä

Parametrejä voidaan hakea monen filtterin perusteella, kuten sillä onko siihen liitettynä lopullista help-tiedostoa, parametrin nimen perusteella sekä parametrin groupID:llä. GroupId on Prima Powerin sisäinen tapa ryhmittää tietyt samanlaiset parametrit. Parametreille voi tästä näkymästä lisätä help-tiedoston tai tehdä muokkauksia jo olemassa olevaan help-tiedostoon. Kuvassa 12 esitetään help-tiedoston muokkausnäkymä.



Kuva 12. Help-tiedoston osiovälilehtien muokkausnäkymä

Help-tiedosto jakautuu datasheet-taulukkoon, joka esittää käyttäjälle parametrin yleisarvoja että parametrille relevanttien koneiden arvoja, ja HTML-pohjaisiin

vapaasti muokattaviin osiovälilehtiin (sections). Help Management Systemin muokkausnäkymä on suunniteltu ainoastaan osiovälilehtien ylläpitoon. Osiovälilehdet tallennetaan Azure Blob Storageen, ja Help Management Systemin tietokantaan tallennettava metadata ylläpitää osiovälilehtien ja help-tiedoston suhteita. Datasheet-taulukon tiedot tulevat suoraan MPE:n tietokannasta, jonka muokkaamiseen ei Help Management Systemiä käytetä. Kuva 13 esittää erään parametrin datasheet-taulukkoa.

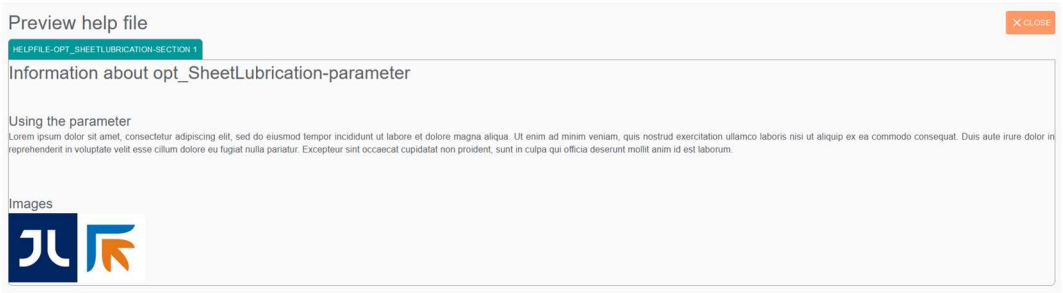
Q.CompensationInPosition4	
Description	Amount of stations in turret. Normally 16, 20, 24 or 30.
Scale	
Target unit	Basic machine turret
Activation	Execute the PARAM_EDIT_CONFIG.spf Beckhoff -> Immediate

	Default value	Default minimum value	Default maximum value	Default location in control	Default location in control 2
Head	-5.5				
Head	-5.5				
Head	-5.5				
Head	1.8				

Kuva 13. Datasheet-taulukko, esimerkki

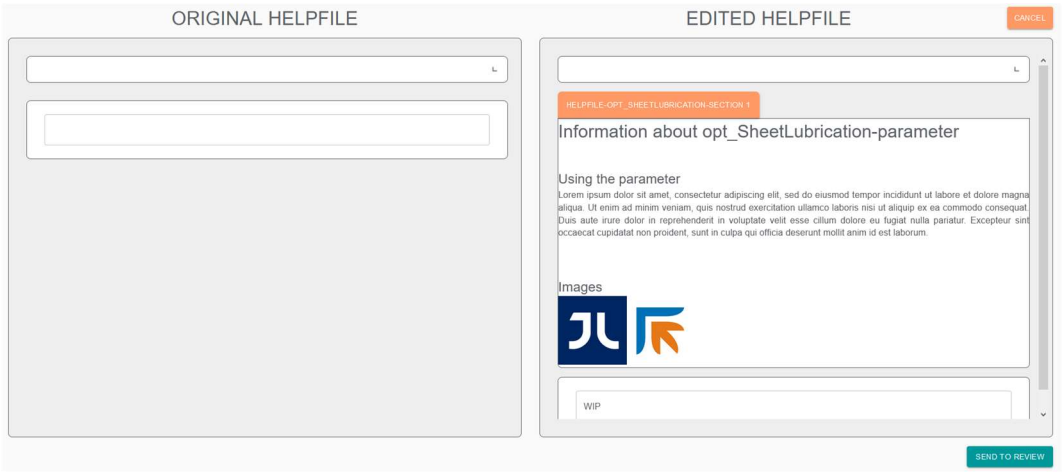
Osiovälilehdet ovat HTML-tiedostoja, joita voidaan luoda ja muokata vapaasti GrapesJS:llä. Prima Powerin vanhat osiovälilehdet koostuvat enimmäkseen tekstistä ja mediasta, mutta Help Management System mahdollistaa GrapesJS:n

avulla myös monipuolisempien osioiden luomisen eri tarkoituksiin. Kuva 14 esittää esikatselunäkymää, jossa nähdään help-tiedoston osiovälilehdet.



Kuva 14. Osiovälilehtien esikatselu

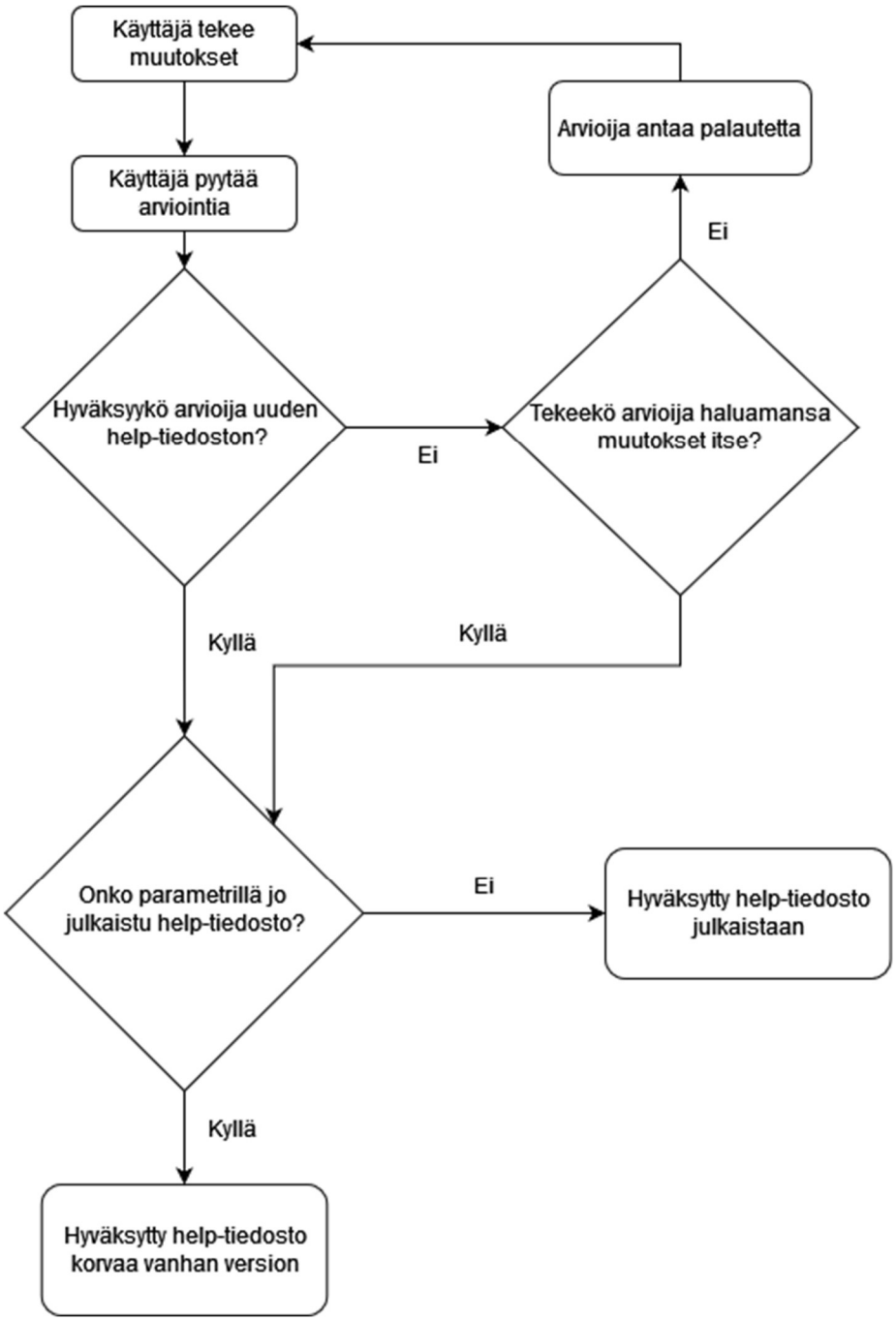
Help-tiedoston muokkaamisen tai lisäämisen jälkeen käyttäjä voi lähettää sen eteenpäin arvioitavaksi. Kuva 15 esittää arviointiin lähetettäviä esimerkkimuutoksia. Esimerkissä luodaan kokonaan uusi help-tiedosto, joten kuvan vasemmalla puolella näkyvä help-tiedoston alkuperäinen versio on tyhjä. Käyttäjä voi tarvittaessa kirjoittaa help-tiedoston alle myös kuvauksen, jonka tarkoitus on antaa arvioijalle lisätietoa muutoksista. Datasheet-taulukko pysyy jokaisen muutoksen jälkeen samana, joten oletuksena sitä ei tässä näkymässä näytetä.



Kuva 15. Help-tiedoston arviointiin lähetys

3.2.3 Arviointi

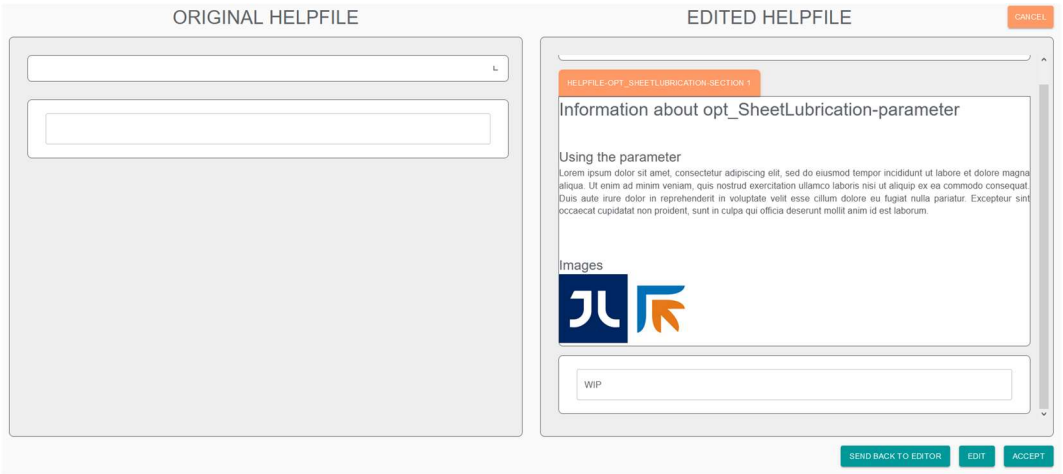
Arviointiprosessin logiikka on visualisoitu kuvassa 16 esitetyssä vuokaaviossa.



Kuva 16. Arviointiprosessi

Kuten kuvan 16 kaaviosta näkyy, arvioijan täytyy hyväksyä help-tiedostoon tehdyt muutokset, ennen kun uusi versio julkaistaan. Arvioijana voi toimia kuka tahansa,

jolla on alustaan moderaattori- tai administraattorioikeudet. Jos arvioija ei hyväksy muutoksia, voi hän lähettää help-tiedoston takaisin sen tekijälle ja kirjoittaa hänelle kuvauksen siitä, mitä tiedostosta puuttuu. Arvioija voi myös tehdä itse tiedostoon haluamansa muutokset, jotka hän voi itse hyväksyä. Kuva 17 esittää arviointinäkymää. Näkymä on lähes identtinen help-tiedoston arviointiin lähettämisessä käytettävän näkymän kanssa, mutta arvioijalla on painikkeet, joita hän käyttää arvioinnin antamiseen, muokkaamiseen tai hyväksymiseen. Muokkausten kuvaamiseen käytettyä tekstilaatikkoa käytetään arviointinäkymässä palautteen antamiseen.



Kuva 17. Arviointinäkymä

Help-tiedoston hyväksymisen jälkeen julkaistaan help-tiedostosta uusi versio. Mahdolliset help-tiedoston vanhat versiot ovat tämän jälkeen saatavilla ainoastaan help-tiedoston versiohistoriassa, jota pääsee katsomaan kuvassa 11 olevan valikon vasemmassa reunassa olevan infonapin avulla. Help-tiedostojen vanhoja versioita ei koskaan todellisesti poisteta, vaan tiedot pidetään tallessa varmuuskopioina. Näin vältetään tapauksia, jossa arvioija hyväksyy epähuomiossa kelvottomat muutokset ja halutaan palauttaa help-tiedoston vanha versio.

3.2.4 Helpfile export

Helpfile export on alustaan rakennettu API-endpoint, joka tullaan tulevaisuudessa integroimaan MPE-sovellukseen. Endpointtia käytetään Prima Powerin koneryhmiin (assembly) liittyvien parametrien tallentamiseen. MPE:n käyttäjä voi syöttää endpointille koneryhmän ID:n, jonka seurauksena Help Management System palauttaa MPE:lle zip-tiedoston, joka sisältää

- yhden help-tiedoston jokaista koneryhmään liittyvää parametriä kohden.
- index.html-tiedoston, joka sisältää hyperlinkin jokaiseen ladattuun help-tiedostoon.
- mediakansion, joka sisältää help-tiedostoihin liittyvän median.

Endpoint on suunniteltu Prima Powerin koneryhmän käyttäjän käyttöön. Koneryhmää käyttävä henkilö pystyy toimintoa käyttäen lataamaan tietokoneellensa tiedoston, joka sisältää paljon relevanttia tietoa hänen käytössään olevista koneista.

Toiminto muuttaa help-tiedostojen median Blob Storage-referenssit lokaaleiksi referensseiksi. Tämä varmistaa, että kerran ladattu help-tiedosto pysyy samanlaisena, vaikka siihen liittyvää media muokattaisiin jälkikäteen Help Management Systemillä.

4 KÄYTTÖÖNOTTO

Tässä luvussa käydään läpi alustan käyttöönottoprosessi. Käyttöönotto suoritettiin Azuren App Service -pilvipalvelun avulla. Käyttöönotossa alustasta luotiin kaksi versiota, yksi testaustarkoituksiin ja toinen Prima Powerin yrityskäyttöön. Tämän tapaisista ohjelmaympäristöistä käytetään yleensä nimityksiä ”staging” ja ”production”.

Staging-ympäristön ylläpito koostui uusien testattavien ominaisuuksien lisäämisestä ympäristöön. Production-ympäristöön tehtiin CI/CD-integraatio, jotta testatut ominaisuudet voidaan lisätä Prima Powerin käyttöön nopeasti ja automaattisesti. CI/CD mahdollistaa jatkuvan ohjelman integraation ja käyttöönoton, joka tarkoittaa käytännössä sitä, että alustan versiohallinnassa päähaaraan (master/main branch) lisätyt ominaisuudet laukaisevat automaattisen käyttöönottoprosessin, joka tuo uudet ominaisuudet production-ympäristössä suoritettavan alustan käyttöön. Kehitysprosessissa eri ominaisuuksille luodaan omia haaroja, joihin tehdyt muutokset päivitetään säännöllisin väliajoin. Ominaisuuden valmiiksi tulemisen jälkeen ominaisuus käy kokeneemman kehittäjän koodiarviointiprosessin läpi. Vasta arvioinnin hyväksynnän jälkeen ominaisuus lisätään päähaaraan, mikä laukaisee CI/CD-käyttöönottoprosessin.

Ohjelman käyttöönotto vaati myös tietokannan käyttöönoton Azuren pilvipalveluun. Tietokannasta luotiin myös kaksi versiota, yksi staging- ja yksi production-ympäristöä varten. Tämän lisäksi alustan yhteyksien konfiguroinnit lisättiin Key-Value pareina Azuren App Service-konfiguraatiojärjestelmään. Ohjelman suunnitteluvaiheessa yhteyksien konfiguraatio on suoritettu lokaalin konfiguraatiotiedoston avulla. Kuva 18 näyttää esimerkkiversion lokaalista konfiguraatiotiedostosta.

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Debug",
      "System": "Information",
      "Microsoft": "Information"
    }
  },
  "MediaContainer": "<Media Container>",
  "ConnectionStrings": {
    "BlobStorage": "<BlobStorage Connection string>",
    "Database": "<Database Connection String>"
  }
}
```

Kuva 18. appsettings.json, esimerkki

Kuvan 18 ylimmäinen alue esittää konfiguraatiotiedoston automaattisesti generoitua osiota, joka määrittää, mitkä tiedot tulostetaan lokiin. Alhaalla ovat kehittäjän luomat konfiguraatiot. Kenttiin pitää tässä esimerkissä ennen ohjelman ajoa lisätä

- Azure Blob Storagen connectionstring-arvon, jolla ohjelma pystyy autentikoimaan itsensä Blob Storageen yhdistäessä.
- tietokannan connectionstring-arvon, jonka avulla päästään käsiksi tietokantaan.
- Azure Blob Storagessa sijaitsevan containerin nimen.

Oikeaoppinen konfiguraatiotiedostojen käyttö tekee ohjelmakoodin versiohallinnasta turvallisempaa, koska itse konfiguraatiotiedostoa ei koskaan viedä versiohallintaan. Näin jos ohjelman lähdekoodi vuotaa julkiseen verkkoon, potentiaaliset kyberrikolliset eivät saa napattua muiden alustojen salasanoja suoraan koodista.

Ohjelman käyttöönoton ohessa niin kehittäjät kuin asiakaskin suorittivat yksityiskohtaisen testausprosessin, jossa etsittiin ohjelman ongelmakohtia, joita korjattiin. Useimmat näistä korjauksista olivat nopeita ja pieniä muutoksia, mutta esille

ilmaantui myös muutama suurempi ongelma. Erityisesti GrapesJS-muokkausympäristön mediaintegraatio toi esille ongelman kuvien tallennuksessa ja niiden hallinnoinnissa itse muokkausnäkymässä. GrapesJS:ään lisättävä media tallennetaan oletuksena osiovälilehteen raakana datana, joka täytyi muuttaa Blob Storage-linkeiksi. Tämä osoittautui hankalaksi, koska Blob Storage-linkeissä tarvitaan mediaan viittaava GUID, jota median raa'assa datassa ei ole, sekä Blob Storagen containerin tiedot, johon frontend ei pääse käsiksi. Toimivien referenssien lisääminen vaati monia muutoksia sekä frontendin ja backendin puolella.

Työn loppuvaiheessa tehtiin myös konsultaatio kokeneemman kehittäjän kanssa, joka esitti epäkohtia ohjelmaan liittyen. Tämän seurauksena ohjelmakoodia uudelleenkirjoitettiin. Yksi ohjelmaan tehtävistä suurimmista muutoksista oli backendin I/O-operaatioita tekevien toimintojen, kuten tietokantakutsujen, muuttaminen asynkroniseksi eli ei-reaaliaikaiseksi. Asynkroniset operaatiot vähentävät toiminnan hidastumista, kun ohjelmaa käyttää samaan aikaan moni eri henkilö. Asynkronisten operaatioiden avulla ohjelma voi jakaa resursseja monille eri I/O-operaatioille samanaikaisesti.

5 YHTEENVETO

Tämän työn tuloksena saatiin tuotantoon uusi Prima Powerin ohjetiedostojen hallinta-alusta, joka otettiin käyttöön keväällä 2022. Työn vaatimuksiin kuuluivat help-tiedostojen luominen ja arvioiminen, sekä median hallinnointi.

Yleisellä tasolla projekti on ollut onnistunut, ja asiakkaalta tullut palaute on ollut päämääräisesti hyvin positiivista. Projekti on edennyt suunnitellussa tahdissa ja kaikki vaaditut ominaisuudet ovat tulleet valmiiksi. Kommunikaatio kehittäjien sekä asiakkaan välillä on ollut useimmiten sujuvaa, ja esille tulleet ongelmat on saatu ratkottua ilman suuria esteitä.

Suurimmat työhön liittyvät ongelmat syntyivät asiakkaan ja kehittäjätiimin toiminnallisista väärinymmärryksistä. Myös kokeneemman kehittäjän konsultaation suorittaminen suhteellisen myöhäisessä kehitysvaiheessa tarkoitti sitä, että uudelleenkirjoitettavia ominaisuuksia oli runsaasti.

Järjestelmätason suurimmat hidasteet liittyivät GrapesJS-kirjastoon. Kirjasto oli kehittäjille suhteellisen uusi haaste, ja aiheutti monia eri ongelmia sekä itse frontendissä että frontendin ja backendin integraatiossa.

Projektia jatketaan ja laajennetaan vielä ainakin kesään 2022 asti. Seuraavaksi lisättävä ominaisuus on kielikäännöksien integraatio, jossa yhdistetään Prima Powerin parametreihin liittyvien tekstien käännösprosessi Help Management Systemiin. Pidemmällä tulevaisuudessa on myös koneiden hälytysdatan hallinnointiominaisuuksien lisääminen.

LÄHTEET

Beal, V. 20.7.2002. Viitattu 27.2.2022. https://www.webopedia.com/definitions/guid/

Codeacademy. 13.9.2021. Viitattu 6.3.2022 https://www.codecademy.com/resources/blog/what-is-react/

Docker. Viitattu 5.3.2022. https://docs.docker.com/compose/

GrapesJS. Viitattu 15.3.2022. https://grapesjs.com/demo.html

IBM. 23.6.2021. Viitattu 5.3.2022. https://www.ibm.com/cloud/learn/docker

Ikola, J. 11.2.2022. Projektisuunnitelma.

Jubic. Viitattu 26.2.2022. https://jubic.fi

Microsoft. 30.3.2021. Viitattu 27.2.2022. https://docs.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction

Microsoft. 6.11.2021. Viitattu 26.2.2022. https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet

Microsoft. 21.2.2022. Viitattu 27.2.2022. https://docs.microsoft.com/en-us/azure/storage/common/storage-introduction

Prima Power. Viitattu 26.2.2022. https://www.primapower.com/fi

Suomen Asiakastieto. Viitattu 13.4.2022. https://www.asiakastieto.fi/yritykset/fi/jubic-oy/26552006/yleiskuva

Typescriptlang. Viitattu 5.3.2022. https://www.typescriptlang.org/