HAMK
Hämeen ammatti-
korkeakoulu

Mallitietokannan suunnittelu ja toteutus 3D-tulostamisen tarpeisiin

Ammattikorkeakoulututkinnon opinnäytetyö

Tieto- ja viestintätekniikka, insinööri (AMK)

Syksy, 2024

Joona Karhu, Joel Mieskonen

HAMK
Hämeen ammatti-
korkeakoulu

Tieto- ja viestintätekniikka Tiivistelmä
Tekijät Joona Karhu, Joel Mieskonen Vuosi 2024
Työn nimi Mallitietokannan suunnittelu ja toteutus 3D-tulostamisen tarpeisiin
Ohjaaja Petri Kuittinen

Opinnäytetyön tarkoituksena oli toteuttaa tietokantapalvelu, johon voi tallentaa kolmiulotteisia malleja, niiden metatietoja ja muita tarvittavia tietoja. Palvelun käyttäminen tapahtuu käyttöliittymän kautta käyttäjäystävällisesti ja se antaa ajankohtaista tietoa tallennetuista malleista, esimerkiksi luomis- ja muokkauspäivämääristä, kuvauksesta, tulostamiseen käytettävästä materiaalista ja tulostusmenetelmästä. Työssä käsiteltiin Django-verkkokehyksen käyttöä ja sen käyttökelpoisuutta työn toteutuksessa. Hyödynnettiin tämän tarjoamia tekniikoita mallitietokannan toteuttamiseen. Käytiin läpi eri vaiheita suunnittelusta, toteutuksesta sekä ylläpidosta.

Opinnäytetyö alkaa web-sovelluksen teoriaosuudesta, jossa käsitellään työhön liittyviä asioita, kuten kehitystä, sovelluksen sisältöä, web-sovelluskehyksiä, sekä viimeiseksi teoriaa Django-verkkokehyksestä. Django-osiossa käsitellään, kuinka palvelinta ylläpidetään, mitä tulee huomioida tietoturvaan liittyen, kun kehittää Django-sovelluksia ja kyseisen kehyksen elinkaarta, kuinka se on kehittynyt ja mihin suuntaan se on menossa. Tämä antaa hyvän kuvan, kuinka web-sovelluskehykset toimivat yleisellä tasolla. Lisäksi Djangosta saadaan selville mikä se on ja kuinka se toimii.

Mallitietokannan osuudessa ohjaudutaan käsittelemään suunnittelua ja toteutusta. Aluksi suunnittelussa käydään läpi mallitietokannan käyttöliittymää perustasolla ja sen ominaisuuksia. Työ syvenee tästä käymällä yksityiskohtaisesti jokaisen sivun ominaisuudet, sekä toiminnallisuudet. Lopuksi työstä löytyy palvelinpuolen toteutusta, kuinka tämä projekti käyttöönotetaan ja kuinka tieto liikkuu palvelinpuolelta käyttöliittymään.

Opinnäytetyössä tutkittiin alustavan konseptitodistuksen avulla, minkälaisia vaatimuksia mallitietokannan kehittämisessä ja ylläpitämisessä tulee huomioida. Saavutettiin kehityksen aikana keskeiset ominaisuudet, mitä palvelusta pitää löytyä. Lisäksi pohdittiin työssä ilmenneitä jatkokehityksen kohteita, mitkä parantaisivat palvelun käyttöä. Työn tilaajan kanssa tehdyn alustavan suunnitelman sekä tutkimuskysymysten avulla, lähdettiin toteuttamaan ratkaisua, joka voisi toimia tulevaisuudessa mallina kehitettäessä uutta ratkaisua työstä.

Avainsanat 3D-malli, 3D-tulostus, Django, Palvelin, Tietotekniikka
Sivut 47 sivua ja liitteitä 1 sivu

HAMK
Hämeen ammatti-
korkeakoulu

Information and Communication Technology — Abstract
Author Joona Karhu, Joel Mieskonen — Year 2024
Subject Design and Implementation of a Model Database for 3D Printing needs
Supervisor Petri Kuittinen

The purpose of this thesis was to implement a database service that allows storing three-dimensional (3D) models, along with their metadata. The service is accessed through a user-friendly interface and provides the latest information about the stored models, such as creation and modification dates, description, material used for printing and printing method. The thesis discusses the use of Django web framework and its usefulness in the implementation of the project. The techniques it offers for implementing the model database were used. The different stages of design, implementation and maintenance are presented.

The thesis starts with a theoretical part of the web application, where topics related to the project such as development, application content, web application frameworks, and finally the theory of the Django web framework are discussed. The Django section covers how to maintain the server, what to consider in terms of security when developing Django applications and the lifecycle of the framework, how it has evolved and where it is heading. This serves as a description of how web application frameworks work in general. It also gives a basic understanding of what Django is and how it works.

The model database section guides through the design and implementation. First, the design goes through the user interface of the model database and its features. The thesis then goes into more detail by describing the features of each page, as well as the functionalities. Eventually, the thesis covers the backend implementation, how to roughly deploy this project, and how the data is transferred between the front end and back end.

The thesis studied, through a preliminary proof of concept, what requirements need to be considered in the development and maintenance of the model database. The key features that the service should have were successfully accomplished. In addition, the thesis identified areas for further development that would improve the use of the service. Based on the initial design and the research questions, a solution was developed that could serve as a reference for the future development of a new implementation.

Keywords 3D model, 3D printing, computer science, Django, server
Pages 47 pages and appendices 1 page

Sisällys

Kuvat, taulukot ja kaavat

Liitteet

1 Johdanto

Opinnäytetyön tarkoituksena on toteuttaa Django-verkkokehyksellä verkkosivulla hallittava tietokantapalvelu, johon voi lisätä 3D-mallinnusohjelmistojen avulla tuotettuja kolmiulotteisia malleja. Palvelua käytetään verkkosivuilla olevan käyttöliittymän kautta, minkä avulla käyttäjä voi tallentaa kolmiulotteisia malleja sekä niihin kytkeytyviä metatietoja ja muita oheistietoja. Näihin tietoihin lukeutuu muun muassa luomis- ja muokkauspäivämäärät, kuvaukset, tulostuksessa käytettävästä materiaalista ja tulostusmenetelmästä. Opinnäytetyö toteutetaan parityönä, joten työnjako on seuraavanlainen: Käyttöliittymäosuuden toteuttaa Joona Karhu ja palvelinpuolen Joel Mieskonen.

Verkkosivulla olevassa taulukossa näkyvät kaikki palveluun tallennetut mallit, sekä osa niihin lisätyistä tiedoista. Käyttäjä voi halutessaan tarkastella tallennettujen mallien ajankohtaisia tietoja valitsemalla taulunäkymästä haluamansa mallin. Mikäli malleja on taulussa runsaasti, malleja voidaan hakea käyttöliittymässä olevan hakutoiminnon avulla, syöttämällä siihen erilaisia tunnistetietoja, kuten esimerkiksi nimen, materiaalin tai päivämäärän mukaan.

Aihe on rajattu siten, että opinnäytetyössä ei oteta huomioon suorituskykytestatusta tai manuaalisia SQL-hakuja. Palvelu olisi tilaajan näkökulmasta käytettävissä vain rajatuille henkilöille tietyillä käyttöoikeuksilla, esimerkiksi uusien mallien lisäämisen, muokkaamisen, tulostamisen ja poistamisen muodoissa. Palvelun kieli on englanti, joten se taipuu myös kansainvälisesti.

Työssä havaittuja ratkaisuja toteutettaessa mallitietokantaa ja siihen liittyvien mallien metatietojen rakennetta voitaisiin hyödyntää esimerkiksi tulevaisuuden 3D-mallien teollisessa tuotannossa ja kirjanpidossa. Tämä mahdollistaa perusteellisen ylläpito ja analysointi mahdollisuuden.

Työssä tutkitaan ja haetaan vastauksia seuraaviin tutkimuskysymyksiin:

- Mitä vaatimuksia 3D-tulostamisen mallitietokannan kehittämiselle ja ylläpitämiselle on huomioitava?
- Miten mallitietokanta toteutettiin ja miksi tähän toteutustapaan päädyttiin?
- Miten muita tietokantoja, jakelupalveluita tai 3D-tulostimia voidaan integroida mallitietokantaan?

2 Teoria

2.1 Web-sovelluskehitys

Web-sovelluskehitys on kokonaisuus, jossa kehitetään verkkosivuja tai web-sovelluksia, mitkä ovat saatavilla joko julkisessa tai suljetussa verkossa käyttäjän verkkoselaimen kautta (Webb, 2024).

Kehityksen kokonaisuuteen sisältyy lukuisia vaiheita:

- Suunnitteleminen sisältää alustavan suunnitelman kehitettävän työn käyttötarkoituksesta, jossa muun muassa pohditaan ja kerätään tietoa, esimerkiksi työn tavoitteesta ja kohderyhmästä (Codecademy, 2021).

- Ulkoasussa suunnitellaan verkkosivujen rakenne ja kuinka visuaaliset elementit tullaan sijoittelemaan niihin (Codecademy, 2021).

- Kehitysvaiheessa front-end- ja back-end-kehittäjät toteuttavat ohjelmallisesti suunnitelman mukaisen web-sovelluksen, johon sisältyvät verkkosivujen käyttöliittymä ja sen välisen toiminnallisuuden palvelinpuolen kanssa (Codecademy, 2021).

- Testaamisessa tutkitaan, että web-sovellus toimii halutulla tavalla. Etsitään mahdollisia virheitä lähdekoodista ja kokeillaan sovelluksen yhteensopivuutta eri selaimien käytössä. (Webb, 2024)

- Julkaisussa web-sovellus tuodaan käyttäjille saatavaksi verkkoselaimen kautta testaamisvaiheen jälkeen (Codecademy, 2021).

- Ylläpito sisältää sovelluksen julkaisemisen jälkeisiä toimenpiteitä, kuten esimerkiksi turvallisuudenseuranta, verkkosivujen ulkonäön ja näytettävän sisällön päivitystyöt (Indeed, 2024a).

Web-sovelluskehityksen ohjelmallisesta kehittämisestä vastaavat web-kehittäjät. Kehittäjät työskentelevät yhdessä ja heidän työpanoksensa muodostavat konkreettisen palvelun toiminnallisuuden, mikä on käyttäjille saatavilla. (Webb, 2024)

Seuraavia kehitysrooleja ovat:

Front-end-kehittäjät, eli web-sovelluksen käyttöliittymän kehittäjät vastaavat verkkosivun ulkoasusta, kuinka esimerkiksi elementit, kontrasti ja käytettävyys verkkosivulla on toteutettu (Rouse, 2013).

Back-end-kehittäjät, eli web-sovelluksen logiikan ja palvelun toiminnallisuuksien kehittäjät vastaavat linkityksestä käyttöliittymään ja siinä olevien elementtien välillä. Kehittäjät käsittelevät ohjelmallisesti käyttöliittymästä tullutta dataa ja kutsuja, varmistavat tiedonkulun palvelussa. He esimerkiksi ylläpitävät ja testaavat palvelua sekä kehittävät rajapintoja. (Rouse, 2017)

Full-stack-kehittäjät omaavat niin front-end- kuin back-end-kehitykseen vaadittavan osaamisen. He esimerkiksi testaavat ja optimoivat palvelun yhteensopivuutta eri selaimille, kehittävät rajapintoja ja vastaavat turvallisuudesta. (Codecademy, 2021)

2.2 Web-sovelluksen sisältö

2.2.1 Käyttöliittymä

Käyttöliittymän tehtävänä on mahdollistaa käyttäjän ja tietokoneen välinen toiminta tietokoneella, verkkosivulla tai sovelluksessa. Hyvä käyttöliittymä parantaa käyttökokemusta ja toteuttaa käyttäjän tarpeet vaivattomasti. (Indeed, 2024b)

Erilaisia käyttöliittymiä on muun muassa;

- Graafiset käyttöliittymät, jotka kattavat tietokoneen, verkkosivujen ja sovellusten käyttäjälle siinä olevia visuaalisia ja audiovisuaalisia elementtejä, kuten esimerkiksi ikkunat, painikkeet, valikot, kuvat, äänet ja videot. (Interaction Design Foundation, 2016; Hashemi-Pour & Churchville, 2024)

- Ääniohjatut käyttöliittymät, jotka mahdollistavat vuorovaikutuksen sovelluksessa käyttäjän puheen avulla, mitä hyödyntävät esimerkiksi virtuaaliavustajat Applen Siri ja Amazonin Alexa (Interaction Design Foundation, 2016).

- Eleisiin pohjautuvat käyttöliittymät, joissa käyttäjän kehon liikkeitä hyödynnetään virtuaaliympäristöissä (Interaction Design Foundation, 2016).

Käyttöliittymän keskeisimpänä tehtävänä web-sovelluksessa, on vastata palvelun sujuvan käytettävyyden avulla käyttäjien odotuksiin. Hyvin suunniteltu käyttöliittymä noudattaa loogista rakennetta, jossa esimerkiksi tekstin ja taustavärin kontrasti, kuvat, interaktiiviset elementit, tarkkaan harkitut visuaaliset efektit sekä verkkosivun responsiivisuus eri laitteilla parantavat palvelun käytettävyyttä ja käyttäjäkokemusta. (Indeed, 2024b)

Web-sovelluskehityksessä käytetty Bootstrap on avoimeen lähdekoodiin pohjautuva front-end-sovelluskehys, jonka avulla voidaan luoda tehokkaasti responsiivisia ja mobiiliystävällisiä verkkosivuja. Bootstrap sisältää valmiiksi luotuja HTML-, CSS- ja JavaScript-komponentteja, mitkä nopeuttavat verkkosivujen kehitystyötä. (Alexandrea, 2023) Toteutuksessa olevissa taulukoissa hyödynnetään Bootstrap 4 -version muotoiluja.

2.2.2 Tietokanta

Tietokanta tallentaa, järjestää ja hallitsee tietoa. Tietokantoja on suunnattu erilaisiin käyttötarkoituksiin, kuten esimerkiksi optimointiin, yhteensopivuuteen, skaalattavuuteen tai helppokäyttöisyyden tukemiseen. (Microsoft, n.d.) Seuraavassa kappaleessa käsitellään tietokannan eri suunnitteluvaiheita.

Ensimmäinen vaihe tietokannan suunnittelussa on sen tarkoituksen määrittäminen. Pohditaan mitä toimintoja tietokanta pitää sisällään ja millaisia tietoja siinä tullaan käsittelemään. Suunnitteluvaihe on yksi tärkeä osuus projektien tavoitteiden varmistamiseksi. Aluksi kerätään tarvittavat tiedot, jotka halutaan tallentaa tietokantaan. Tähän voi kuulua esimerkiksi tuotteiden nimet, tilausnumerot, asiakastiedot ja muut olennaiset tiedot. Tietojen haku auttaa selventämään, millaista tietokannan rakennetta tarvitaan. (Microsoft, n.d.)

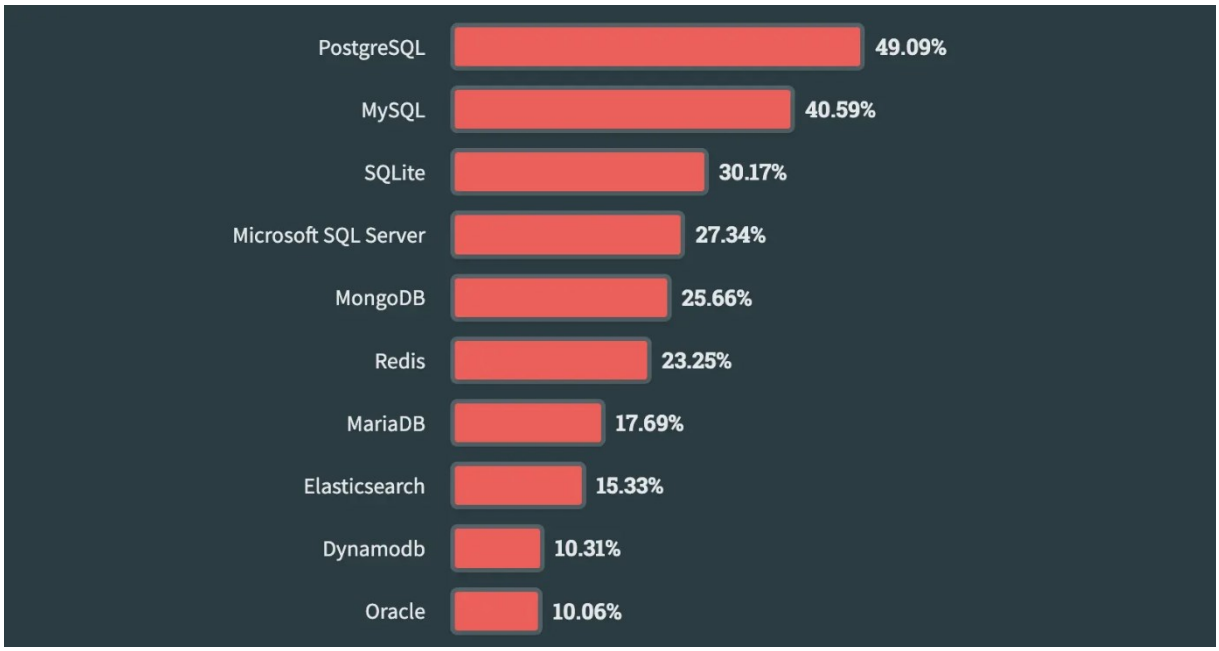
Haetut tiedot jaetaan pääkohteisiin tai -aiheisiin. Esimerkiksi tuotteet ja tilaukset voivat olla omia taulukoita. Jokaiseen taulukkoon jako auttaa pitämään tiedot järjestyksessä ja helposti hallittavina. Kyseiset tiedot päätetään mihin taulukkoihin ne halutaan tallentaa. Jokaisesta kohteesta tulee oma taulukon kenttä, joka näkyy sarakkeena. Esimerkiksi työntekijätaulukossa voi olla kentät sukunimi, etunimi ja aloituspäivä. Tämä vaihe varmistaa, että kaikki tiedot tallennetaan oikein järjestelmällisesti. (Microsoft, n.d.)

Jokaiselle taulukolle valitaan perusavain. Sarake, jossa on perusavain, yksilöi kunkin rivin. Perusavain voi olla esimerkiksi tuotetunnus tai tilaustunnus. Tämä sarake on erottuva, sillä se on uniikki ja sen avulla voidaan tunnistaa tietueet toisistaan. Tämän jälkeen tarkastellaan ja päätetään, miten taulukoiden tiedot liittyvät toisiinsa. Tarvittaessa lisätään taulukkoihin uusia kenttiä tai uusia taulukoita yhteyksien selventämiseksi. (Microsoft, n.d.)

Tietokannan rakennetta analysoidaan mahdollisten virheiden varalta. Tämän jälkeen luodaan tietokantaan taulukoita ja lisätään esimerkkitietoja tietueisiin. Taulukoista haetaan tietoja ja katsotaan, saadaanko halutut tulokset. Näiden jälkeen tehdään muokkauksia rakenteeseen, jotta tietokanta toimii halutulla tavalla. (Microsoft, n.d.)

Esitettyjen suunnitelmaratkaisujen perusteella, tulisi nyt olla käsitys tietokannan rakenteesta, sekä tämän suunnittelusta. Kuvassa 1 eritellään kyselytutkimuksen perusteella suosittuja tietokantoja.

Kuva 1. Vuoden 2023 kysely ammattilaiskehittäjien käyttämistä tietokannoista (Stack Overflow, 2023).



Toteutuksessa käytetään SQLite-tietokantaa helpon käyttöönoton vuoksi, sillä se tulee oletusarvoisesti konfiguroituna Djangossa. Tietokanta voidaan kuitenkin vaihtaa Djangon *settings.py*-tiedostosta. Työn tilaajan kanssa suunniteltiin mallien lisäämisrakenne, jota työssä käytetään.

2.2.3 Palvelin

Palvelin on tietokone, joka tarjoaa resursseja, dataa, palveluita tai ohjelmia verkon yli muille tietokoneille. Erilaisiin käyttötarkoituksiin valjastettuja palvelimia ovat, esimerkiksi web-, sähköposti- ja virtuaalipalvelimet. (Paessler, n.d.) Palvelin konfiguroidaan vastaanottamaan pyyntöjä käyttäjiltä, mihin se vastaa halutulla tavalla. Käyttäjän vieraillessa esimerkiksi verkkosivuilla, lähetetään kutsu palvelimelle, mikä palautetaan käyttäjälle takaisin pyynnön käsittelyn jälkeen. (Paessler, n.d.)

Web-palvelimella tarkoitetaan palvelinta, jolla ajetaan web-sovellusta, tyypillisesti verkkosivustoa ja sen resursseja, kuten esimerkiksi HTML-dokumentit, kuvat, CSS-muotoilut ja JavaScript-tiedostot (MDN Web Docs, 2024). Web-palvelinta voidaan ajaa esimerkiksi paikallisesti lähiverkossa oleville muille laitteille.

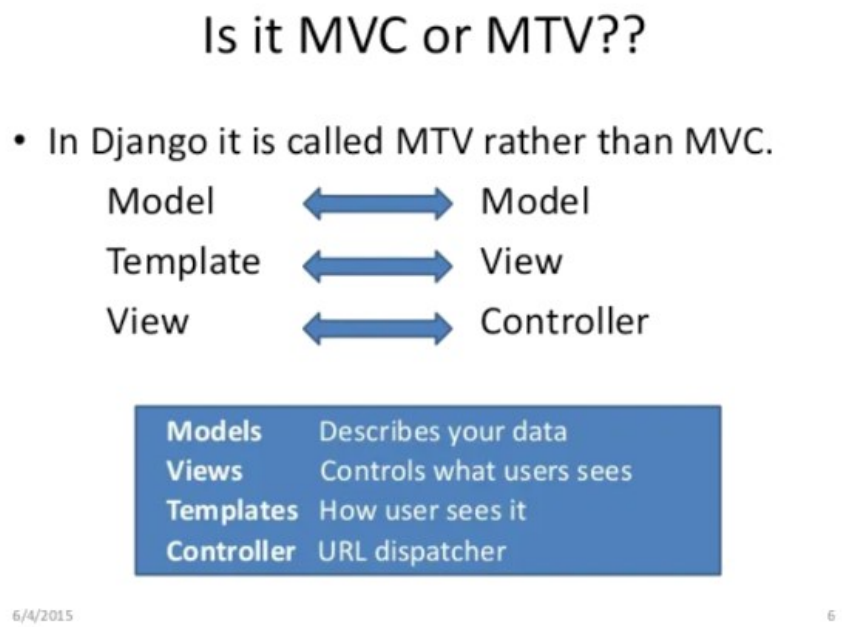
2.3 Web-sovelluskehykset

Web-sovelluskehykset ovat valmiiksi rakennettuja kehyksiä, joiden avulla luodaan ja ajetaan web-sovelluksia. Ne nopeuttavat kehityksen vaiheita, jolloin käyttäjän ei itse tarvitse ohjelmoida vastaavaa kehystä. (Intelegain Technologies, 2019) Ne voidaan jakaa palvelin- ja asiakaspuolen kehyksiin. Nämä ovat yleisesti rakennettu MVC (Model-View-Controller) -arkkitehtuuria noudattaen. (Oyom, 2017)

Kehysten ominaisuudet koostuvat seuraavista komponenteista: *Model* vastaa kaikesta dataan liittyvästä käsittelystä ja toiminnasta, eli se määrittää kuinka käyttäjän syöttämät tiedot päätyvät *Controller*-komponentista *View*-komponenttiin. *View* on tarkoitettu tietojen, kuten esimerkiksi kuvaajien tai kaavioiden graafiseen esittämiseen. Se muodostaa sovelluksen niin sanotun front-endin. *Controller* toimii välikappaleena *Model*- ja *View*-komponenttien välissä, hakien käyttäjän syötetyt tiedot *View*-komponentista ja välittää tiedot *Model*-komponenttiin. (Oyom, 2017)

Django on toteutettu hyödyntäen MVC-arkkitehtuuria, mutta sen sijaan se muodostuu arkkitehtuurista nimeltä MTV (Model-Template-View). Termit voidaan kuitenkin linkittää toisiinsa. (Oyom, 2017.) Kuva 2 osoittaa kuinka termit linkitetään toisiinsa.

Kuva 2. MVC- ja MVT-arkkitehtuurien vertailu (Oyom, 2017).



Palvelinpuolen kehyksen kehittäjän toteutus koostuu näistä aiheista: HTTP-pyynnöt, tietokannan hallinta ja reititys (Oyom, 2017). Kolme suosituinta palvelinpuolen kehyksiä pois lukien Django, ovat:

ASP.NET Core, Microsoftin kehittämä avoimen lähdekoodin kehys, joka on alustariippumaton ja sitä käytetään sovellusten luomiseen käyttäen .NET:llä. Kehyksellä pystytään kehittämään eri kielillä kuten esimerkiksi C#, F#, Visual Basic, NodeJS, ja JavaScript. Hyödyt kehyksen käyttämiseen ovat seuraavat: ASP.NET Core:n kanssa saadaan hyödynnettyä suosittua JavaScript-kieltä. Uudemmat kehysversiot mahdollistavat vähäisen koodauksen ja erinomaisen suorituskyvyn. (Gadhavi, 2024)

Laravel, suosittu valinta PHP-kielen käyttäjille. Kaikista verkkosivuista 75.7 % koostuu PHP:sta (W3Techs, n.d.). Se on avoimen lähdekoodin kehys, suorituskyvyltään nopea ja turvallinen, mikä tekee siitä web-kehittäjien suosikkivalinnan. Hyödyt kehyksen käyttämiseen ovat seuraavat: Se tarjoaa vahvat turvaominaisuudet hyödyntämällä algoritmeja turvallisiin salasanoihin, Widget-tuki, joka sisältää CSS- ja JS-widgetit PHP:n lisäksi, sopiva valinta

monimutkaisiin projekteihin, SwiftMailer-integraatio, joka mahdollistaa tämän helpon käytön. (Gadhavi, 2024)

Ruby on Rails, tunnettu full-stack-ominaisuuksista ja helppokäyttöisyydestä. Kehys tarjoaa infrastruktuurin, eli komponentit, joista rakennetaan ohjelmiston koodi. Hyödyt kehyksen käyttämiseen ovat seuraavat: Reaaliaikaiset päivitykset ja tietojen synkronointi, antaa kehittäjille täyden valtuuden projektien hallintaan, kuten tietokannan toiminnan määrittämisen. Tämä on mainio valinta aloittelijoille. (Gadhavi, 2024)

2.4 Django

Django-verkkokehyksen verkkosivuilla todetaan: “Django on korkean tason Python-verkkokehys, joka kannustaa nopeaan kehitykseen ja puhtaaseen, käytännölliseen suunnitteluun. Se on kokeneiden kehittäjien kehittämä, ja se huolehtii suurimmasta osasta web-kehityksen vaivasta – –.” (Django, n.d.-n) Kyseisessä toteutuksessa käytimme Django versiota 4.1.12.

Djangoa käytetään sovelluksissa kuten Instagram, joka on yksi suosituimmista valokuvien ja videoiden jakelupalvelusta. Spotify, yleisesti käytetty musiikkipalvelu suoratoistomusiikin kuunteluun. (Korsun, 2024) Django siis sopeutuu mainiosti sekä pienemmille että laajemmille projekteille.

2.4.1 Ylläpito

Ylläpidolla tarkoitetaan sovelluksen toteutuksen jälkeen olevaa vaihetta, jossa kyseistä sovellusta hallitaan, päivitetään ja korjataan. Tähän kuuluu esimerkiksi tärkeiden haavoittuvaisuus, suorituskyky ja uusien ominaisuuspäivityksien asennus. Ylläpito on myös yksi tärkeä osa sovellusten kehittämisessä, jolla varmistetaan sovelluksen luotettavuus ja toimivuus. (Indeed, 2024a)

Tässä toteutuksessa ajetaan Django-palvelinta paikallisella tietokoneella lähiverkossa. Django soveltuu myös hyvin komentorivi käytössä ilman graafista käyttöliittymää. Ohjelma voidaan ajaa esimerkiksi tmux:ia (terminal multiplexer) käyttäen, jolloin ohjelma pyörii taustalla. Vaihtoehtoisesti jos halutaan automatisoida ohjelman käynnistämistä, voidaan käyttää systemd.

Systemd on käyttöjärjestelmän ja siinä ajettavien palveluiden käynnistymistä ja sammuttamista ohjaava ns. init-järjestelmä. Useimmat jakelut ovat siirtyneet käyttämään systemd-järjestelmää. (Systemd, n.d.)

Migrations on tapa siirtää malleihin tehdyt muutokset, kuten *field*-kentän lisäämisen ja *model*-luokan poistamisen muutokset tietokantaan. Ne ovat suunniteltu pääsääntöisesti automaattisiksi, mutta tämä vaatii tietämyksen, milloin ne tulee tehdä, suorittaa ja millaisia yleisiä ongelmia voi esiintyä tämän yhteydessä. (Django, n.d.-d)

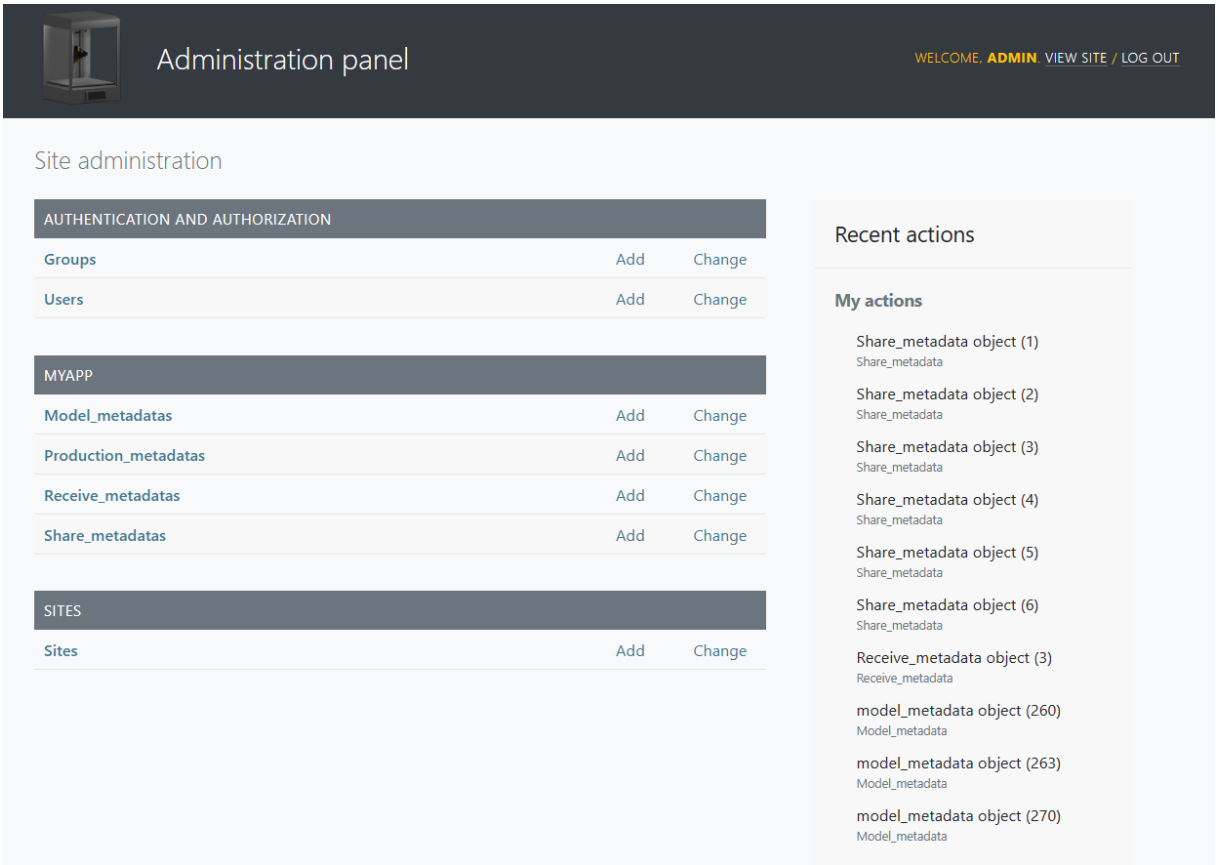
Koodissa Djangon models-päivityksen tai muokkauksen jälkeen tulee päivittää Django-tietokanta komennoilla *python3 manage.py makemigrations myapp* ja *python3 manage.py migrate*. Komento *makemigrations* vastaa uusista migraatioista malleihin muokkausten perusteella (Django, n.d.-d). Komento *migrate* vastaa migraatioiden käyttöönotosta ja poistamisesta, eli tietokannan päivittämisestä (Django, n.d.-d). Käynnistettäessä Django-palvelinta uudelleen, käytetään komentoa *python3 manage.py runserver* (Django, n.d.-i). Tietokannan pyyhkiminen voidaan toteuttaa *python3 manage.py flush* -komennolla, joka pyyhkii tiedot tauluista, mutta ei poista itse tauluja tai niiden rakennetta (Django, n.d.-j)

Mahdollisia ulkopuolisia palveluita ohjelman ajamiseen ovat esimerkiksi Amazon Web-service (AWS) ja Microsoft Azure. Kyseisissä palveluissa voidaan helposti graafisen käyttöliittymän avulla tarkastella tilastoja ja erilaisia statistiikkoja, ohjata koneen asetuksia sekä verkkoasetuksia. Virtuaalikoneille pystyy tarvittaessa lisäämään tai vähentämään esimerkiksi levytilaa (Microsoft, 2024).

Ylläpidossa on hyvä myös huomioida levytilan lisätarve, kun käyttäjä määrä kasvaa. Tämän lisäksi tietokanta kannattaa ottaa huomioon, sillä eri tietokannat ovat suunniteltu eri käyttötarkoituksiin, esimerkiksi jotkut ovat optimoitu pienemille tietomäärille, kun taas toiset suuremmille. (Cuello, 2024)

Djangosta löytyy oletusarvoisesti järjestelmänvalvojan hallintapaneeli. Täältä voidaan lisätä sekä muokata käyttäjiä ja tietokannan tietoja. Sivu pitää kirjaa muun muassa palvelussa olevista ryhmistä, käyttäjistä ja tietokantaan tallennetuista *models*-datasta. (Django, n.d.-k) Kuvassa 3 esitellään Django-järjestelmänvalvojan hallintapaneeli.

Kuva 3. Django-järjestelmänvalvojan hallintapaneeli.



2.4.2 Tietoturva

Django tarjoaa useita ratkaisuja tyypillisiin verkkohaavoittuvuuksiin Web-sovelluksissa. Nämä haavoittuvuudet ovat:

Cross-site scripting (XSS) -hyökkäys, jossa sovellukseen syötetään haitallisia asiakaspuolen skriptejä, joilla manipuloidaan sovelluksen toimintoja. Django Template-tiedostot oletusarvoisesti siivotaan haavoittuvuuden välttämiseksi. (StackHawk, 2022)

Cross-site Request Forgery (CSRF) -hyökkäys, joka yrittää hyödyntää käyttäjän todennettuja tietoja luvattomiin pyyntöihin. Django suojautuu tältä uhalta käyttämällä CSRF middleware:a ja luomalla CSRF-tokenin, joka on salainen arvo. Jokaisessa palvelinpyynnössä palvelin varmistaa tämän salaisen arvon pätevyyden. (StackHawk, 2022)

SQL-injektio tapahtuu, kun hyökkääjä lisää SQL-pyyntöön haitallisia kyselyitä, joilla voidaan manipuloida ja hakea tietoa tietokannasta. Riskilähde syntyy käyttäjän syötetyistä tiedoista,

jossa hyökkääjän SQL-komento syötetään ohjelmiston SQL-komentoon. Django erottaa SQL-koodin käyttäjän syötöistä ja prosessoi ne turvallisesti. (StackHawk, 2022)

HTTPS, verkkosivuilla liikkuvien tietojen salaus. On erittäin suositeltavaa käyttää tätä protokollaa Djangon yhteydessä. Django tarjoaa useita ominaisuuksia protokollan hyödyntämiseksi. Nämä ominaisuudet ovat seuraavat: HTTP-pyyntöjen automaattinen ohjaus HTTPS:lle, evästeiden lähettäminen vain kyseisellä protokollalla, HTTP Strict Transport Security (HSTS) -tekniikka, joka pakottaa selaimia toimimaan vain HTTPS-protokollalla. (StackHawk, 2022)

SSL-sertrifikaatti voidaan ottaa käyttöön esimerkiksi käyttäen Nginx tai Apache palvelinohjelmistoa. Tämän yhteydessä yleensä käytetään Gunicorn (Green Unicorn) -WSGI (Web Server Gateway Interface) HTTP-palvelinta.

Käyttäjän autentikointi onnistuu kätevästi Djangossa *authenticate*-funktion avulla. Kyseistä funktiota voidaan käyttää seuraavasti:

```
username = form.cleaned_data['username']
password = form.cleaned_data['password']
user = authenticate(request, username=username, password=password)
```

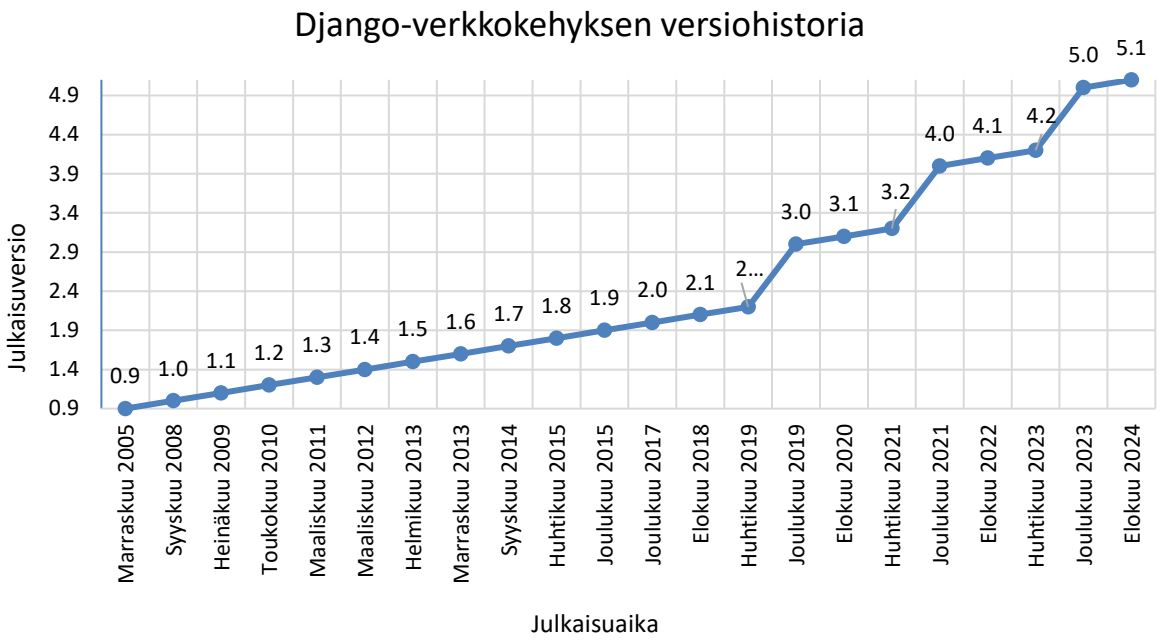
username sekä *password* tulevat Djangon muodostetusta form:ista.

2.4.3 Elinkaari

Django-verkkokehyksellä on pitkä kehityshistoria. Kehityksen alkuvaiheet ajoittuvat vuoteen 2005. Django on kasvattanut suosiotaan vuosien varrella, tarjoten lisää ominaisuuksia suorituskykyyn, turvallisuuteen ja käyttäjäystävällisyyteen. (Emanuilov, 2024)

Versiossa 3.1 tuli merkittävä parannus asynkronisiin *views*-funktioihin, jossa on mahdollista määritellä non-blocking verkkopyyntöjä. Seuraavassa versiossa 4.0, tuli päivitys asynkroniseen API:in, joka laajensi palvelinpuolen välimuistinhallintaa. Django 4.1 mahdollisti objekti-relaatiokartoituksen asynkronisesti, joka mahdollisti tietokantaoperaatioiden suorittamisen vaikuttamatta palvelimen vasteaikaan. (Emanuilov, 2024) Kuvassa 4 esitetään Django versiohistoriaa.

Kuva 4. Django-verkkokehyksen versiohistoria muodostettu lähteen mukaan (Django, n.d.-l).

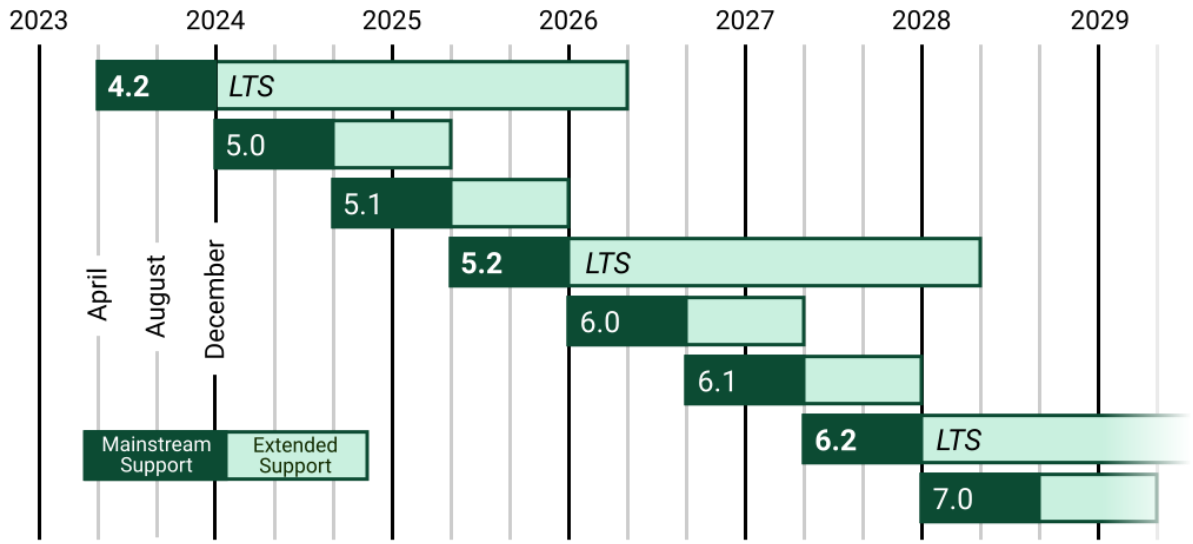


Tällä hetkellä Django-verkkokehys noudattaa melko vahvaa asemaa markkinoilla, joka on noin 25 000 yrityksen käytössä ja 32.08 % markkinaosuudella (6sense, n.d.). Tämä osoittaa, että Django on edelleen suosittu valinta. Päivityksiä lisätään siihen jatkuvasti ja se pysyy hyvin ajan tasalla uusimmista teknologioista ja trendeistä. Django on laajasti dokumentoitu, josta löytyy helposti tarvittavat välineet kehitykselle.

Tämänhetkisten kyselyiden perusteella Djangon käyttäjämäärä tulevaisuudessa vaikuttaisi pysyvän suosiossa. JetBrains:in vuonna 2022 tuottamaan verkkokehyskyselyyn otti kantaa yli 4000 Django-kehittäjää, missä mitattiin Django-verkkokehyksen suosiota. Kyselyssä suosio on hieman laskenut 83 %:sta käyttäjämäärästä 74 %:iin. (Letusheva, 2024)

Django jatkaa tulevaisuudessa asynkronista ohjelmoinnin kehitystä, tämä edistää ohjelmiston suorituskykyä, kun I/O operaatiot saadaan non-blokkaavaksi, jolloin ohjelma ei jumitu, kun esimerkiksi levyltä luetaan tietoa. (Emanuilov, 2024) Django suunnittelee julkaisevansa tulevaisuudessa 5.2 LTS -version huhtikuussa 2025, 6.0 joulukuussa 2025 ja 6.1-version elokuussa 2026 (Django, n.d.-m). Kuva 5 osoittaa arviota Djangon tulevaisuuden versioista.

Kuva 5. Djangon tulevat versiojulkaisut (Django, n.d.-m).



3 Mallitietokanta

Työssä lähdetään liikkeelle valitsemalla projektissa käytettävät sopivat työkalut ja tekniikat. Luvussa 2.3 kerrottiin erilaisista web-sovelluskehyksistä, joista työhön valikoitui Python-ohjelmointikieltä käyttävä Django, muun muassa sen helpon käytettävyyden, nopean tuotettavuuden, tietoturva ominaisuuksien ja valmiiksi integroidun SQLite-tietokannan vuoksi.

Pohdittaessa työn sisältöä, tavoitteita ja rakennetta tulee ensin rajata aihe. Työn tavoitteena on kehittää toimiva tietokantapalvelu halutuilla ominaisuuksilla. Vain pienellä käyttäjämäärällä on palveluun pääsy rajatuin oikeuksin. Palveluun asetetut erilaiset käyttöoikeudet vaihtelevat esimerkiksi mallien lisäämisen, muokkaamisen, tulostamisen ja poistamisen muodossa. Työ olisi tuotantovaiheessa julkaistavissa paikallisessa verkossa.

3.1 Suunnittelu

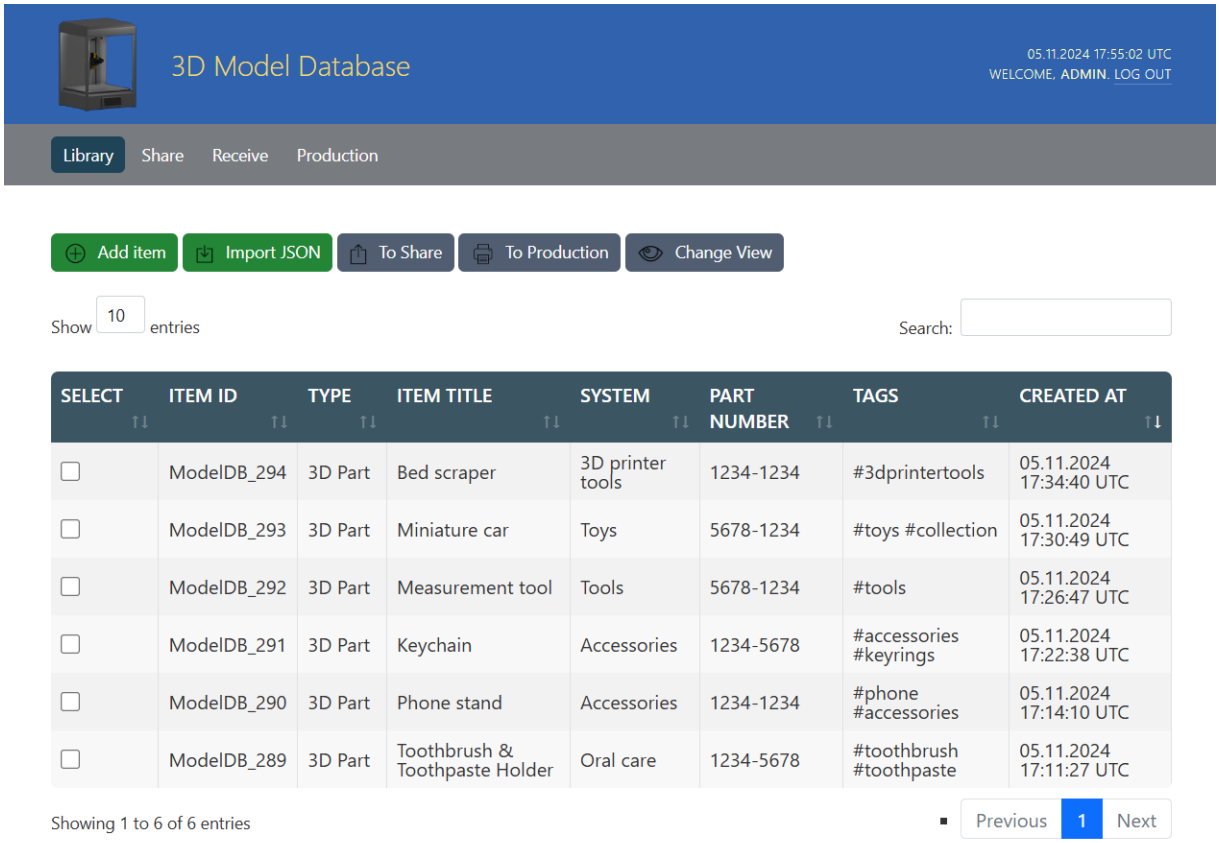
3.1.1 Käyttöliittymä

Luvussa käsitellään, kuinka palvelun verkkosivun käyttöliittymä ja siinä olevat elementit muodostetaan. Perehdytään Django back-end puolelta tuotavaan dataan, jota käytetään verkkosivujen Template-tiedostoissa Django Template -kielen syntaksin mukaisesti. Datan avulla rakennetaan verkkosivuilla olevia elementtejä, kuten taulukot ja niihin kytkeytyvät

painikkeet erilaisilla toiminnoillaan, joiden avulla voidaan olla vuorovaikutuksessa taulukossa oleviin tietoihin.

Lähdettäessä suunnittelemaan palvelulle käyttöliittymää, valikoitui haluttujen ominaisuuksien mukaan yksinkertainen ja kompakti ratkaisu. Käyttöliittymässä haluttiin pitää yleisilme neutraalina ja kontrasti niin tekstien kuin värien osalta. Palvelussa olevat verkkosivut noudattavat käyttöliittymän osalta samaa rakennetta. Keskeistä verkkosivuilla ovat taulukot, jotka ovat sivukohtaisesti luotuja niille suunniteltuja käyttötarkoituksia varten. Kuvassa 6 havainnoidaan mallitietokannan etusivun Library-näkymää, minkä jälkeen verkkosivun rakennetta aletaan purkamaan yksityiskohtaisesti, ja käymään läpi siinä käytettyjä tekniikoita.

Kuva 6. Mallitietokannan etusivu.



Django sisältää oman sisäänrakennetun Template-koodikielensä (Django Template Language). Kieltä käytetään template-tiedostoissa, joihin voidaan tuoda, esimerkiksi Python back-end-koodista dataa. Templateilla tarkoitetaan tekstidokumentteja, kuten esimerkiksi HTML-, XML-, CSV-tiedostoja, mitkä sisältävät muuttujia, jotka korvataan arvoilla templaten ajamisen yhteydessä ja tageja, jotka ohjaavat tiedoston logiikkaa (Django, n.d.-g). Syntaksi

noudattaa seuraavaa rakennetta muuttujissa *{{ muuttuja }}* ja tageissa *{{% %}}*, jonka sisälle voidaan syöttää, esimerkiksi ehtolauseita ja silmukoita (Django, n.d.-h).

Toteutuksessa olevien verkkosivujen kehittämisen runkona hyödynnetään Djangon lähdekoodissa olevia admin-verkkosivun template-tiedostoja. Palveluun kehitettävillä verkkosivuilla käytetään template-perintää (Template inheritance), jossa perittävästä tiedostosta voidaan jatkaa sen sisältö *extends*-toiminnon avulla, säilyttäen siinä olevat elementit uuteen tiedostoon (Django, n.d.-f). Suunniteltaessa template-tiedostojen rakennetta, on suositeltavaa ottaa huomioon toistuva koodi, joka voitaisiin yhdistää yhteen template-tiedostoon edellisellä mainitulla tavalla. Tämä auttaa koodin, sekä bugien vähentämisessä.

Käyttöliittymään on luotu navigointipalkki verkkosivun yläosaan palvelun logon alapuolelle. Se sisältää painikkeita linkkeineen, joiden avulla käyttäjä voi siirtyä palvelussa oleville eri verkkosivuille. Painiketta painettaessa sen taustaväri muuttuu tummansiniseksi, jolla ilmaistaan käyttäjälle tämänhetkinen verkkosivu. Perittävään template-tiedostoon on luotu Django template -kielellä tagien avulla *block*-elementti “breadcrumbs”, mikä sisältää navigointipalkissa olevat painikkeet. Jokainen verkkosivu perii template-tiedostosta oman sivun yhteyteen navigointipalkin sisällön. Seuraavassa koodipätkä käsittää painikkeiden luomista.

```
{% block breadcrumbs %}
<div class="breadcrumbs">
    <div class="main_buttons">
      {% for button in breadcrumbs_buttons %}
        <a
            {% if button.url %}
                href="{% url button.url %}
            {% endif %}"
            class="
                {% if button.class %}
                    {{ button.class }}
                {% endif %}
            "
            {% if button.type %}
                type="{{ button.type }}"
            {% endif %}
            {% if button.id %}
                id="{{ button.id }}"
            {% endif %}>
            <span>{% translate button.text %}</span>
        </a>
```

```
      {% endfor %}
    </div>
  </div>
{% endblock breadcrumbs %}
```

Koodipätkässä käydään for-silmukan avulla läpi lista sanakirjoista (dictionary), joka sisältää back-end-koodista tulleita painikkeita. Sanakirjasta haetaan jokaiselle painikkeen attribuuteille niille määritetyt arvot. Tämä toimii perustana verkkosivuilla luotaville painikkeille. Esimerkiksi palvelussa oleville taulukoille on yläpuolella luotu niihin kytkeytyvät painikkeet samalla periaatteella, mutta ne luodaan sivukohtaisissa HTML-dokumenteissa.

3.1.2 Taulukko

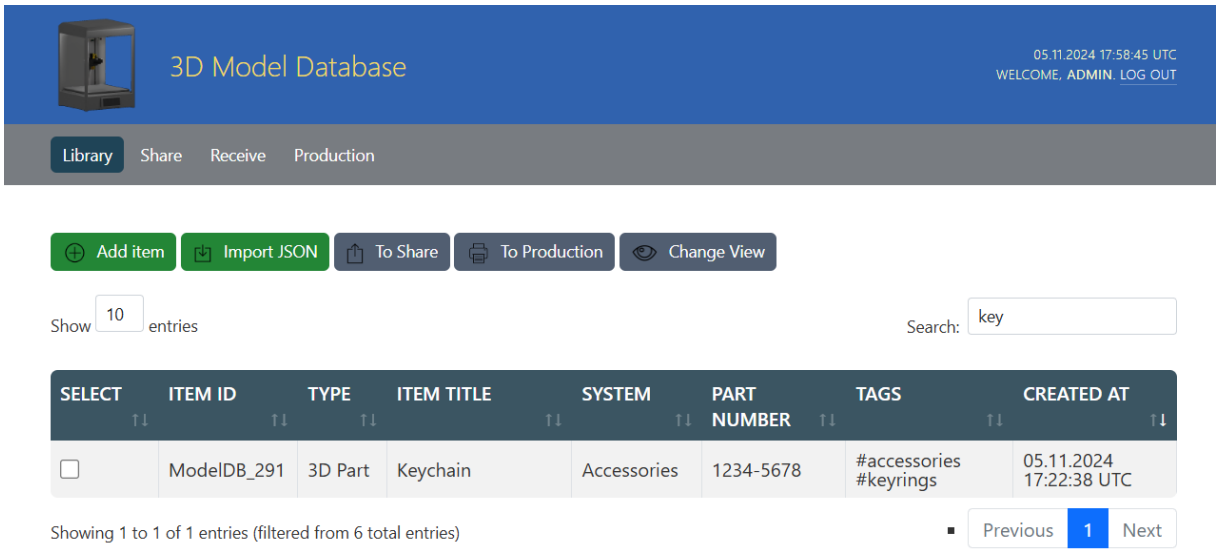
Luvussa käsitellään toteutuksessa olevien taulukoiden rakennetta ja rivien muodostumista niihin Djangosta tuodun back-end-datan perusteella. Aluksi esitellään taulukko ja siinä olevia ominaisuuksia, jonka jälkeen käydään läpi käytettyjä tekniikoita.

Mallitietokannan etusivulla olevassa taulukossa näytetään palveluun tallennetut kolmiulotteiset mallit. Taulukon sarakkeet kertovat tallennetun mallin tietoja, kuten esimerkiksi sen nimen, tekijän, lisenssin, mihin kokoelmaan se on luokiteltu ja milloin se on lisätty palveluun. Mallit ovat oletuksella järjestetty nousevaan järjestykseen, ja sitä voi muuttaa painamalla haluttua saraketta – asettamalla sen joko nousevaan tai laskevaan järjestykseen.

Käyttäjät voivat etsiä taulukossa olevia malleja erilaisilla avainsanoilla käyttämällä hakukenttää taulukon oikeassa yläkulmassa. Tietoa voi hakea minkä tahansa sarakkeen tiedon mukaan, esimerkiksi tagin, luomispäivämäärän ja mallitunnuksen avulla.

Taulukon vasemmassa yläreunassa on *Show entries* -kohta, joka kertoo, kuinka monta riviä on kerrallaan näkyvillä taulussa. Painamalla lukuvalikkoa, avautuu näkyviin neljä eri vaihtoehtoa: 10, 25, 50 ja 100. Oletuksena taulussa näytetään 10 riviä. Mikäli tietoa olisi runsaasti saatavilla, voi taulukossa näytettävien rivien määrää kasvattaa *Show entries* -kohdassa, tai vaihtoehtoisesti vaihtamalla sivuja taulukon oikean alareunan painikkeiden avulla. Kuvassa 7 esitellään taulukosta haettu malli hyödyntäen hakutoimintoa.

Kuva 7. Taulukosta haettu malli hakutoiminnon avulla.



DataTables on avoimeen lähdekoodiin pohjautuva JavaScript-kirjasto, joka tarjoaa edistyneitä ominaisuuksia HTML-dokumenteissa oleviin taulukoihin (DataTables, n.d.-h). Kirjastossa on valmiina ominaisuuksina, esimerkiksi taulukon sivunvaihto painikkeiden avulla, hakutoiminto ja sarakkeiden järjestäminen joko nousevaan tai laskevaan järjestykseen (DataTables, n.d.-b). Kyseinen kirjasto valittiin sen sisältämien toimintojen ja kattavan dokumentaation perusteella.

Taulukossa käytetään kirjastosta olevaa 1.10.22-versiota Bootstrap 4 -muotoiluilla. DataTables-kirjaston lisääminen työhön vaatii siihen kuuluvat CSS- ja JavaScript-tiedostot, jQuery-riippuvuuden lisäksi toimiakseen (DataTables, n.d.-a). Seuraavalla koodipätkällä havainnoidaan kirjaston alustamista Library-sivun taulukkoon.

```
$(document).ready(function () {

  $('#table_div').DataTable({
    "order": [[{{table_count}}, "desc"]],
    "columnDefs": [
      { "orderable": true, "targets": "_all" },
      { "type": "date-euro", "targets": {{table_count}} }
    ]
  });
});
```

Koodipätkässä alustetaan HTML-dokumentin lataamisen jälkeen DataTable-funktio, johon asetetaan lisävalinnoilla taulukkoon kohdistuvia muutoksia. Ensimmäiseksi funktion

argumenttiin syötettyyn sanakirjaan (dictionary) asetetaan *order*, joka määrittää alustavan sarakkeen järjestämisen (DataTables, n.d.-c). Se asetetaan alkavan laskevaan järjestykseen taulukon viimeisestä sarakkeesta, missä muuttuja *{{ table_count }}* on sarakkeiden viimeinen indeksi. Kohta *columnDefs* on lista sarakkeelle määriteltävistä objekteista, minkä avulla voidaan määritellä yksittäisiin sarakkeisiin tehtäviä muutoksia *targets*-parametrin avulla (DataTables, n.d.-d). Parametrilla *orderable* määritellään, että onko sarake mahdollista järjestää (DataTables, n.d.-g). Tämä asetetaan arvoksi *true*, ja se kohdistetaan jokaiseen sarakkeeseen *_all*-merkkijonolla (DataTables, n.d.-d). Lopuksi asetetaan sarakkeelle *type*, jota käytetään sarakkeelle tehtävään merkkijonojen suodatukseen ja järjestämiseen (DataTables, n.d.-e). Tämä asetetaan viimeisen *Created at* -sarakkeen arvoksi *date-euro*, jolla määritellään Euroopan maille tyypillinen päivämäärän esittäminen muodossa päivä-kuukausi-vuosi ja kellonaika (DataTables, n.d.-f).

Palvelussa olevat lukuisat eri taulukot noudattavat samaa rakennetta muodostettaessa niihin tietoa. Kappale käsittelee yhden esimerkin avulla taulukoiden sarakkeiden ja rivien muodostamista. Seuraavan koodipätkän avulla selitetään, kuinka etusivun taulukkoon luodaan rivejä tietokannasta haettujen mallien perusteella.

```
<div class="table-responsive bdr">
  <table id="table_div" class="table table-striped table-bordered"
cellspacing="0" width="100%">
    <thead class="bg-head">
      <tr>
        <th class="th-sm">Select</th>
        {% for column in table %}
          <th class="th-sm">{{ column }}</th>
        {% endfor %}
      </tr>
    </thead>
    <tbody>
      {% for model_instance in model_datas %}
      <tr>
        <td>
          <input type="checkbox" name="selected_rows" value="{{
model_instance.0 }}">
        </td>
        {% for model_element in model_instance %}
          {% if forloop.counter > 1 %}
            <td id="td-class-id" onclick="location.href='{% url page_edit id=
model_instance.0 %}'">
              {{ model_element }}
            </td>
```

```
          {%endif %}
        {% endfor %}
      </tr>
      {% endfor %}
    </tbody>
  </table>
</div>
```

Taulukolla on *id*-arvolle määritetty *table_div*, johon alustettiin DataTable. Ensimmäisessä for-silmukassa määritetään taulukon sarakkeiden nimet, jotka saadaan *table*-muuttujasta render-funktion *context*-muttujasta. *model_datas* on lista kaikista tietokannan malleista, *model_instance* malliin liittyvä instanssi ja *model_element* viittaa Django models:in yksittäiseen muuttujaan. Kyseisessä koodissa käytetään *forloop.counter*, joka on Django-templaten sisäinen muuttuja, jossa *counter* inkrementoituu jokaisen silmukan jälkeen. Kohdassa *forloop.counter > 1* ohittaa mallin pääavaimen, koska sitä ei haluta näyttää taulukossa, mutta sitä tarvitaan mallin muokkaamisessa.

Template-kielessä saadaan listan indeksissä oleva arvo *.index*-syntaksilla, jossa *index* on listan indeksi. Kohta *page_edit id=model_instance.0*, kutsuu mallin muokkausfunktiota painettaessa taulukon solun td-elementtiä, jossa annetaan mallin pääavain *model_instance.0* funktion argumentiksi. Lopuksi annetaan arvo *{{ model_element }}* td-elementtiin.

3.2 Library

Taulukossa on useita erilaisia sarakkeita, joilla on oma merkityksensä. Ne kertovat osan malleihin tallennetuista tiedoista. Lähdettäessä liikkeelle taulukossa olevista sarakkeista – vasemmalta oikealle:

Select-kohdassa, kullakin rivillä on oma valintaruutu, jota painamalla voidaan valita taulukossa oleva rivi tai rivejä. Valinnan tehtyä – riveihin voi soveltaa palvelussa olevia toimintoja painamalla taulukon yläpuolella olevista tummansinisistä painikkeista. Näiden

painikkeiden sisältöön palataan yksityiskohtaisesti kunkin sivun kohdalla, omissa kappaleissa.

Item id -kohdassa, näytetään tietokannan nimi ja mallin uniikki tunnus eroteltuna alaviivalla. Esimerkkinä voisi olla *ModelDB_273*. Tietokantapalvelun jakeluversiossa palveluita voisi olla pystytetty useita, jolloin myös tietokantoja olisi useita erilaisia kukin omalla nimellään, mitkä saataisiin keskustelemaan toistensa kanssa.

Type-kohta, määrittää sen mihin kategoriaan malli kuuluu. Kategorioita on useita: *3D Part*, *3D Collection*, *2D Part* ja *2D Collection*. Luokittelu auttaa esimerkiksi mallien tuotannossa siten, että onko malli yksittäinen osa *Part*, vai *Collection*, joka taas kertoo, että kuuluuko se johonkin kokonaisuuteen. Kokoelmaan luokitellut mallit voivat kuulua jonkin *parent*:in, esimerkiksi auton alaisuuteen. Tämä mallin niin kutsuttu *parent* määritellään sarakkeen System-kohdassa.

Item title -kohta, kertoo oleellisimman tiedon, eli mallin nimen, joka on tallennettu tietokantaan. Lisättäessä malleja palveluun, tulee mallin nimeämiseen kiinnittää tarkkaan huomiota, jotta palvelu pysyisi mahdollisimman johdonmukaisena.

System-kohta, näyttää mallille luokitellun pääluokan, eli minkä kokonaisuuden alaisuuteen se kuuluu. Hyvänä esimerkkinä tähän voisi ottaa auton, joka on määritetty System-kohtaan. Autolla on erilaisia alaluokituksia, kuten esimerkiksi merkki, malli ja väri, mitkä kuuluvat Type-kohdan kokoelmaan *collection*.

Part number -kohta, eli mallin osanumero, joka voi olla valmistajan määrittämä tunnus. Sen avulla voidaan hakea tarkasti haluttu malli tietokannasta, tapauksessa, jossa palvelussa esiintyy samalla nimellä tai hyvin lähellä samaa nimeä oleva malli.

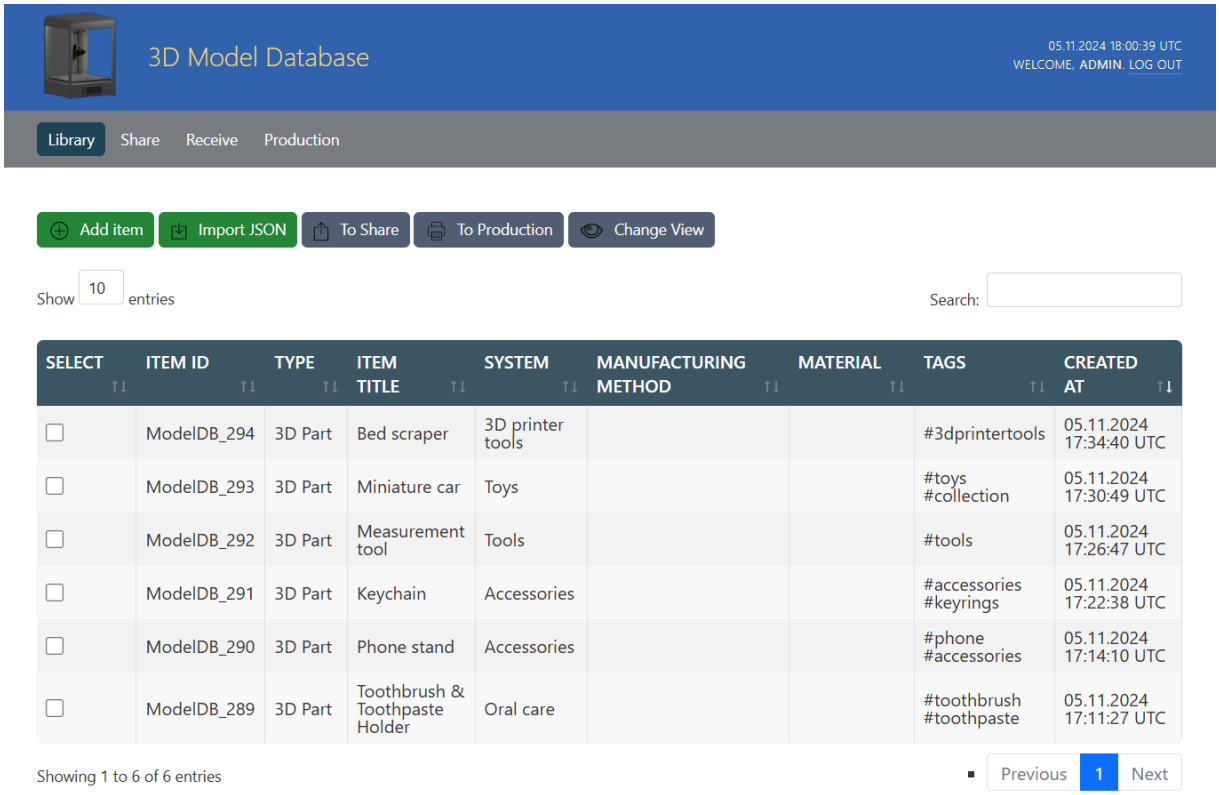
Tags-kohta, voidaan vapaavalintaisesti määritellä malleihin liittyviä avainsanoja. Malleja voidaan yksilöidä ja ryhmittää tagien avulla. Esimerkiksi liittämällä niitä tapahtumiin tai vuosiin, kuten “#tapahtuma2024”.

Created At -kohta kertoo ajankohdan, jolloin malli on tallennettu mallitietokantaan. Päivämäärä esitetään muodossa päivä-kuukausi-vuosi ja kellonaika UTC-aikavyöhykkeellä.

Taulukon sarakkeiden näkymää voidaan vaihtaa yläpuolella olevan *Change View* -painikkeen avulla, jolloin edellisten taulussa olleiden sarakkeiden lisäksi näkyy *Manufacturing*

method ja *Material*, missä edellinen on korvannut *Part number* -sarakkeen. Kuva 8 hahmottaa taulukon toista näkymää.

Kuva 8. Taulukon toinen näkymä.



3.2.1 Mallin lisääminen tietokantaan

Lisättäessä uutta mallia, tulee ensiksi painaa taulukon vasemmassa yläkulmassa olevaa *Add item* -painiketta. Näkymään avautuu lomake, joka koostuu useista erilaisista kohdista, joilla voidaan yksilöidä hyvin tarkasti lisättävään malliin kuuluvia tietoja, työn tilaajan määrittämien kohtien rakenteen perusteella. Liitteessä 1 esitetään tarkemmin lomakkeessa olevia kohtia.

Lomakkeen lopussa olevassa liitteet-kohdassa lisätään liitetiedostot tallennettavan mallin yhteydessä. Useissa kohdissa on työn tilaajan määrittämät sallitut eri tiedostomuodot. Näihin sisältyy, muun muassa metatietokortti-pdf, esikatselukuva, 3MF- ja STL-tiedostot ja projektitiedosto.

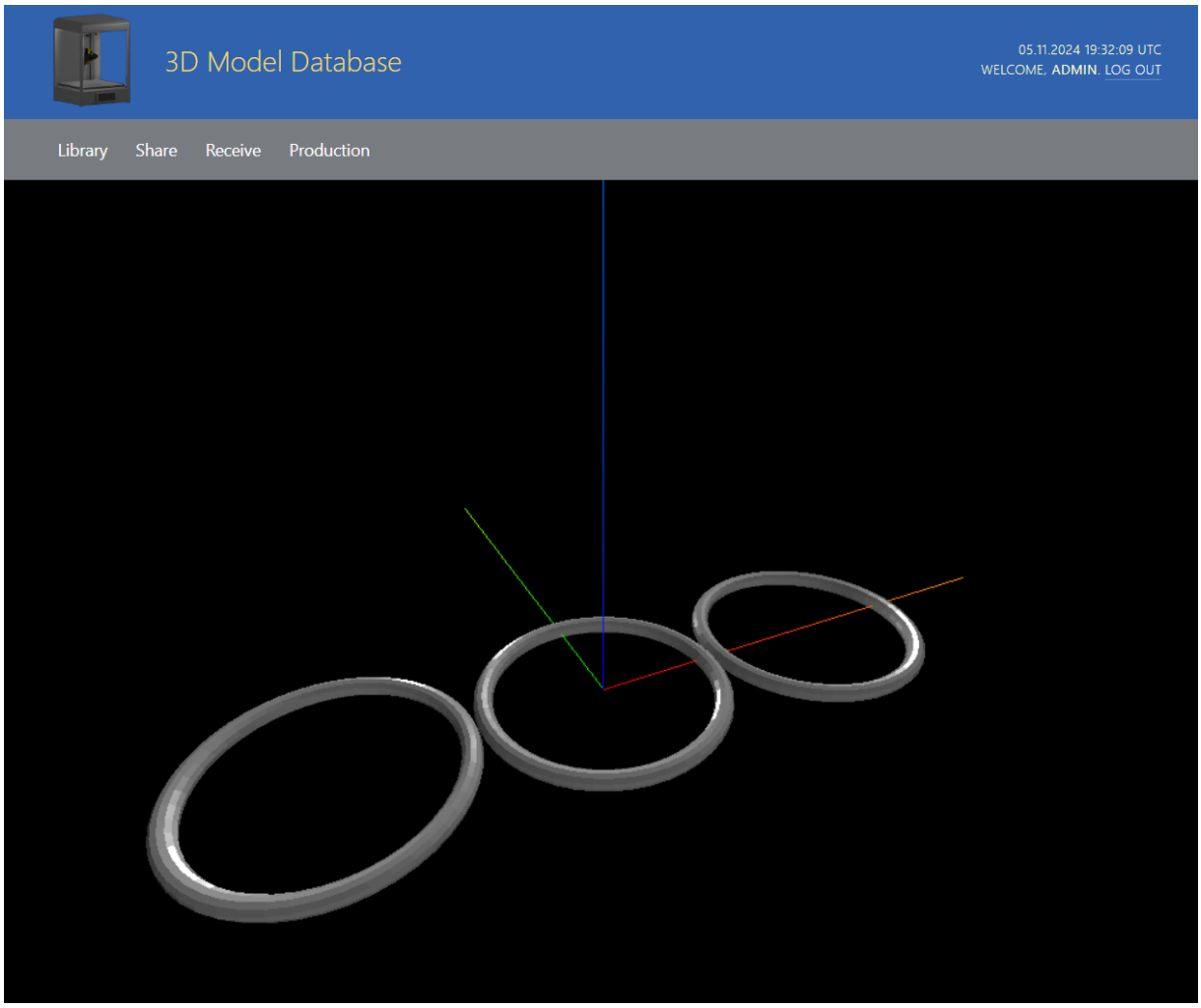
Käyttäjä voi halutessaan jättää osan liitteistä täyttämättä, ja tallentaa. Tallentamisen jälkeen ilmestyy Library-sivun taulukkoon uusi rivi ja osa siihen lisätyistä tiedoista näytetään rivin

sarakkeissa. Tallennettuun kohtaan voi palata takaisin muokkaamaan malliin kytkeytyviä tietoja tai lisäämään liitetiedostoja.

3.2.2 Kolmiulotteinen näkymä verkkosivulla

Tietokantapalveluun tallennettujen mallien liitetiedostoihin on mahdollista lisätä STL-tiedosto, jota on mahdollista tarkastella kolmiulotteisessa avaruudessa, käyttäen Three.js-kirjastoa. Käyttäjä voi hallinnoida näkymää, esimerkiksi liikuttamalla tai suurentamalla sitä ja pyöriä kappaleen ympärillä. Kuvassa 9 tarkastellaan mallia kolmiulotteisessa näkymässä verkkosivulla.

Kuva 9. Mallin tarkasteleminen kolmiulotteisessa näkymässä.



Alla oleva koodipätkä kertoo, kuinka ikkunan koon vaihtaminen tulee huomioida Django CSS-elementtien kanssa. Se myös päivittää kameran projektiomatriisin huomioiden ikkunan kuvasuhteen. Lopuksi *renderer*, eli näkyvän kuvan koko asetetaan sopivaksi sivun mittoihin.

```
function on_window_resize() {
    const header_height = document.getElementById('header').offsetHeight;
    const nav_breadcrumbs_height =
        document.querySelector('nav[aria-label="Breadcrumbs"]').offsetHeight;
    const container = document.getElementById('content');
    camera.aspect = window.innerWidth / window.innerHeight;
    camera.updateProjectionMatrix();
    renderer.setSize(content.clientWidth, window.innerHeight - header_height +
nav_breadcrumbs_height);
}
```

3.2.3 Datan tuominen palveluun JSON-tiedostomuodossa

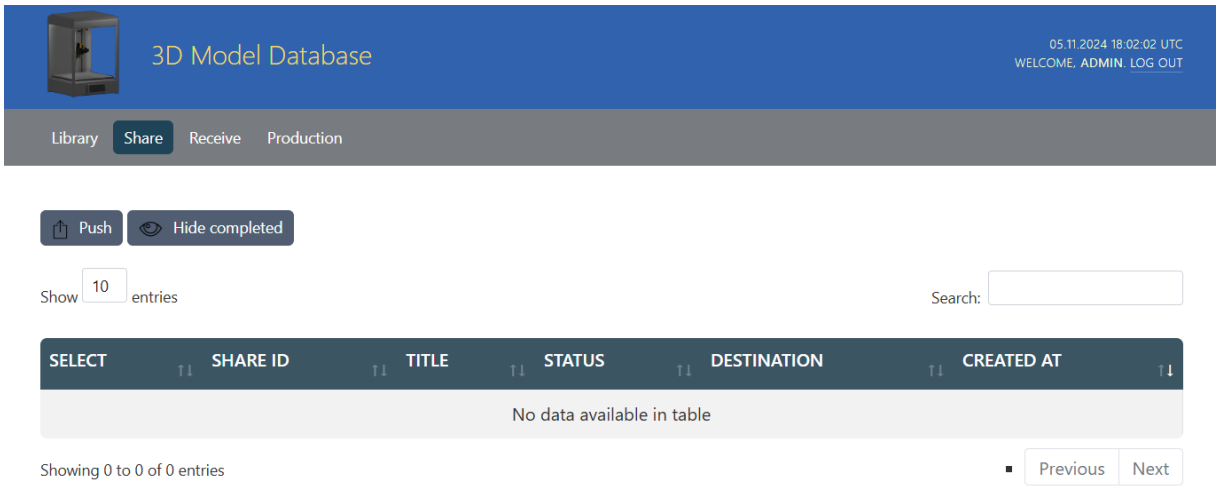
Tietokantapalvelun datan käsittelyn sujuvoittamista suunniteltaessa, kehitettiin palveluun ominaisuus, joka mahdollistaa ketterän datan tuomisen palveluun. Tavallisesti taulukkoon luotaisiin *Add item* -lomakkeen kautta uusi malli, kun sen sijaan voidaan se tuoda helposti JSON-tiedostossa olevan datan avulla.

Käyttäjän tarvitsee vain valita tuotava tiedosto tietokoneeltaan, näin ollen datan tuominen tauluun on yksinkertaista ja helppoa. Tämä mahdollistaa varmuuskopiointien sujuvan tuomisen tietokantaan ja samalla helpottaa yksittäisten mallien siirtämistä tietokannasta toiseen.

3.3 Share

Tietokantapalvelussa on jakotapahtumia sisältävä taulukko Share-sivulla. Taulukko koostuu Library-taulukon malleista luoduista jakotapahtumista. Niiden avulla voidaan määrittää, kuinka valittuja malleja halutaan käsitellä seuraavaksi. Malleja sekä niihin kytkettyjä liitetiedostoja voidaan esimerkiksi ladata ja jakaa toisen palvelun välillä tätä kautta. Kuvassa 10 esitellään jakotapahtuma verkkosivun ulkonäköä.

Kuva 10. Jakotapahtumien verkkosivu.



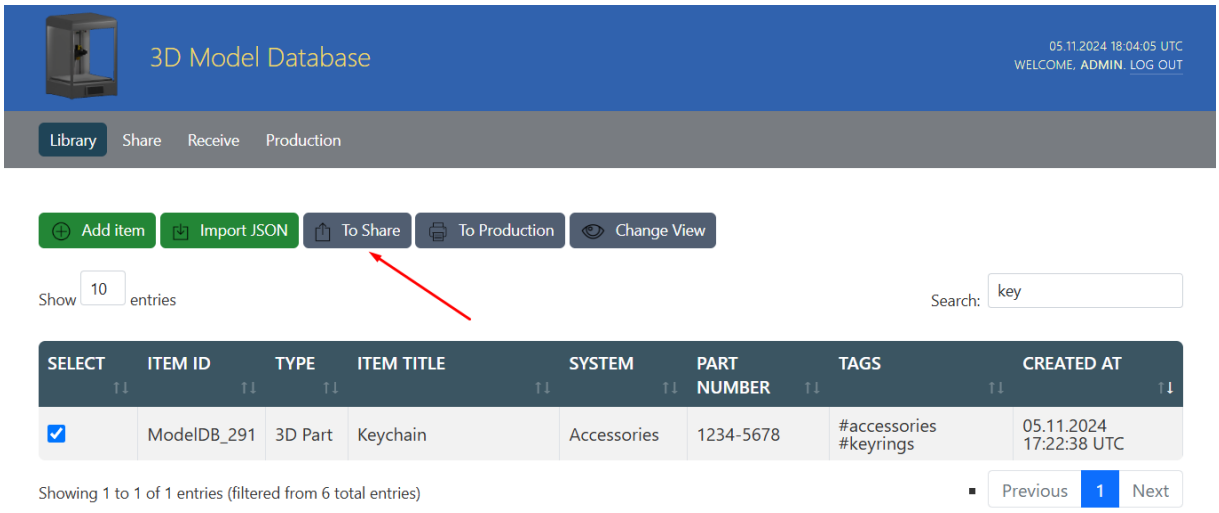
Taulukossa on useita sarakkeita, joiden avulla hahmotetaan jakotapahtumiin sisältyviä tietoja:

- Select-toiminto, jonka avulla voidaan valita ainoastaan yhden rivin kerrallaan toimintoa varten. Näin ennaltaehkäistään ylimääräisten tapahtumien suorittamista.
- Share ID -tunnus, eli jakotapahtumatunnus on uniikki ja se luodaan jakotapahtuman luonnin yhteydessä. Ilmoitetaan kokonaislukuna.
- Title, eli jakotapahtuman nimellä kerrotaan, mitä varten tapahtuma on luotu tai mihin se kohdistuu
- Status, kertoo tapahtuman sen hetkisen tilan. Jakotapahtumalla on kolme tilaa: Ready for Share, Sending ja Success.
- Destination kertoo, mitä tapahtumalle on määritetty toiminnoksi lähettämistä varten.

3.3.1 Mallin siirtäminen jakotapahtumiin

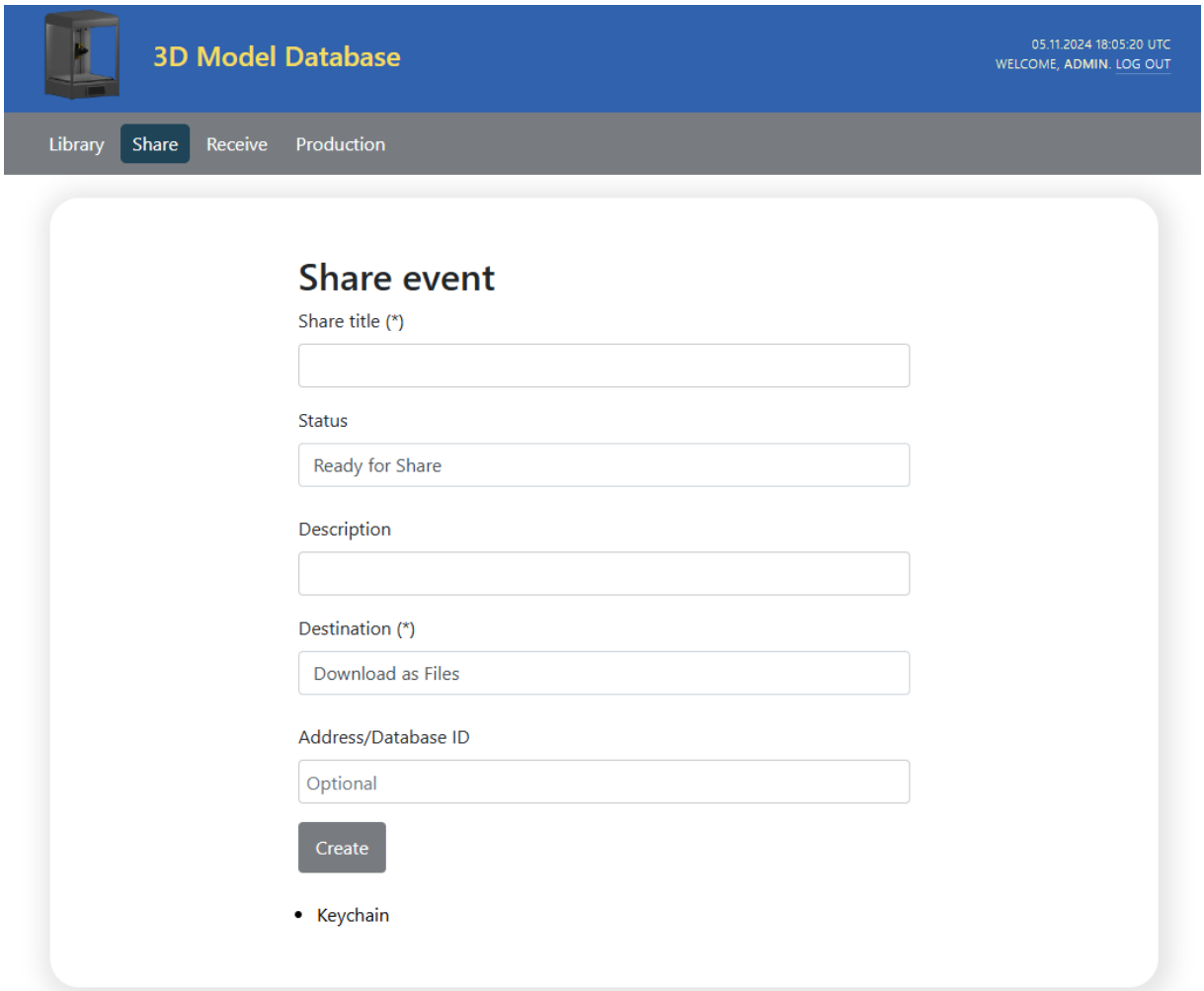
Tietokantapalvelun etusivun Library-taulukosta, voidaan valita yksi tai useampi rivi jakotapahtumiin siirtämistä varten. Käyttäjän tehtyä valintansa, voidaan hyödyntää tummansinisellä pohjalla olevaa To Share -painiketta mallin siirtämistä varten. Kuvassa 11 siirretään malli jakotapahtumiin.

Kuva 11. Mallin siirtäminen jakotapahtumiin.



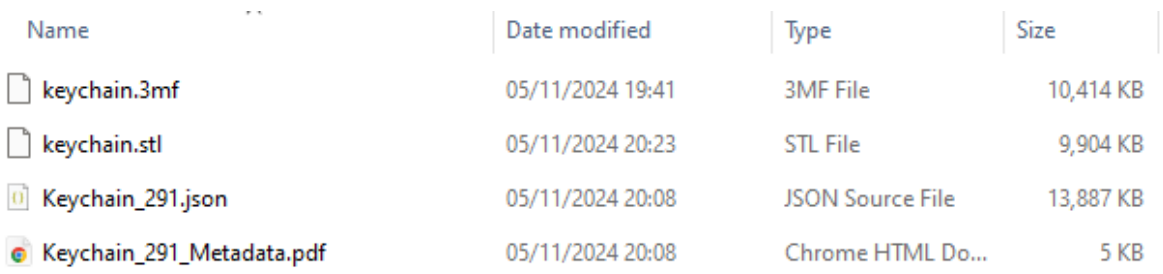
To Share -painikkeella luodaan taulukosta valituista riveistä Share-sivulle uusi jakotapahtuma. Painikkeesta aukeaa jakotapahtuman lomake, joka sisältää useita täytettäviä tekstikenttiä. Kuvassa 12 luodaan jakotapahtuma Library-taulukon rivistä, joka sisältää avaimenperän.

Kuva 12. Jakotapahtuman luominen.



Download as Files -kohdan avulla määritetään, että malli sekä kaikista siihen lisätyistä liitetiedostoista luodaan jakotapahtuma, jonka suorittamisen jälkeen on mahdollista ladata paikalliselle tietokoneelle. Kuva 13 sisältää ladatun mallin nimenmukaisen hakemiston sisällön ZIP-tiedoston purkamisen jälkeen.

Kuva 13. Ladatun tapahtuman hakemisto.



Download as JSON -kohdan avulla määritetään, että mallista sekä kaikista siihen liitetyistä liitetiedostoista luodaan JSON-tiedosto, joka on mahdollista tuoda Library-sivun taulukkoon Import JSON -painikkeen avulla.

Share folder -hakemisto pitää sisällään lähtevät tiedostot tietokannasta. Hakemistosta voidaan hakea tiedostoja eri tietokannasta esimerkiksi SFTP-protokollalla. Tässä JSON-tiedostossa on määritetty mistä tietokannasta tämä on lähtöisin, jolloin tuodessa tiedostoa palveluun tämä voidaan huomioida.

Status, kertoo tapahtuman sen hetkisen tilan. Jakotapahtumalla on kolme tilaa: Ready for Share, Sending ja Success.

- Ready for Share ilmaisee, että tapahtuma on valmis toteutettavaksi, ja voidaan lähettää palvelussa olevan Push-painikkeen avulla.
- Sending tarkoittaa, että tapahtuma on lähetetty eteenpäin ja on vielä kesken.
- Success tarkoittaa, että tapahtuma on onnistunut.

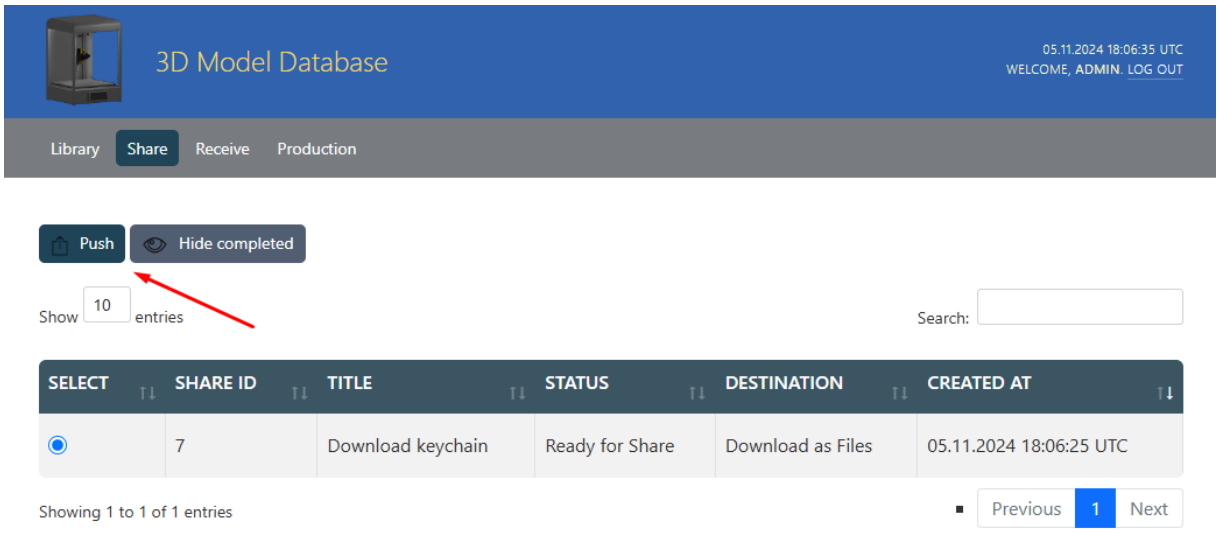
Address/Database ID, on valinnainen tapa lähettää malleja tiettyyn osoitteeseen.

Kun vaaditut kohdat lomakkeessa, on täytetty, voidaan viimein luoda jakotapahtuma painamalla Create-painiketta. Tapahtumalla on oletuksena Ready for Share -statustila. Käyttäjä voi tarkastella taulukossa olevia tapahtumia painamalla haluttua riviä. Jo luotua tapahtumaa ei voi jälkikäteen muokata. Vain järjestelmänvalvojalla on siihen oikeus. Näin on haluttu varmistaa, että sivustolla tapahtuvasta toiminnasta jää talteen tietoa malleista tehdyistä tapahtumista.

3.3.2 Jakotapahtuman suorittaminen

Tapahtumalle on määritelty täytettävässä lomakkeessa Destination-kohdassa, mikä toiminto tullaan suorittamaan jakotapahtumaan liitetylle mallille tai malleille. Malli voidaan joko ladata liitetiedostoineen Download as Files -valinnalla, pelkkänä JSON-tiedostona tai jakaa toiseen palveluun. Kuva 14 havainnollistaa suoritettavaa tapahtumaa.

Kuva 14. Jakotapahtuman suorittaminen painikkeella.



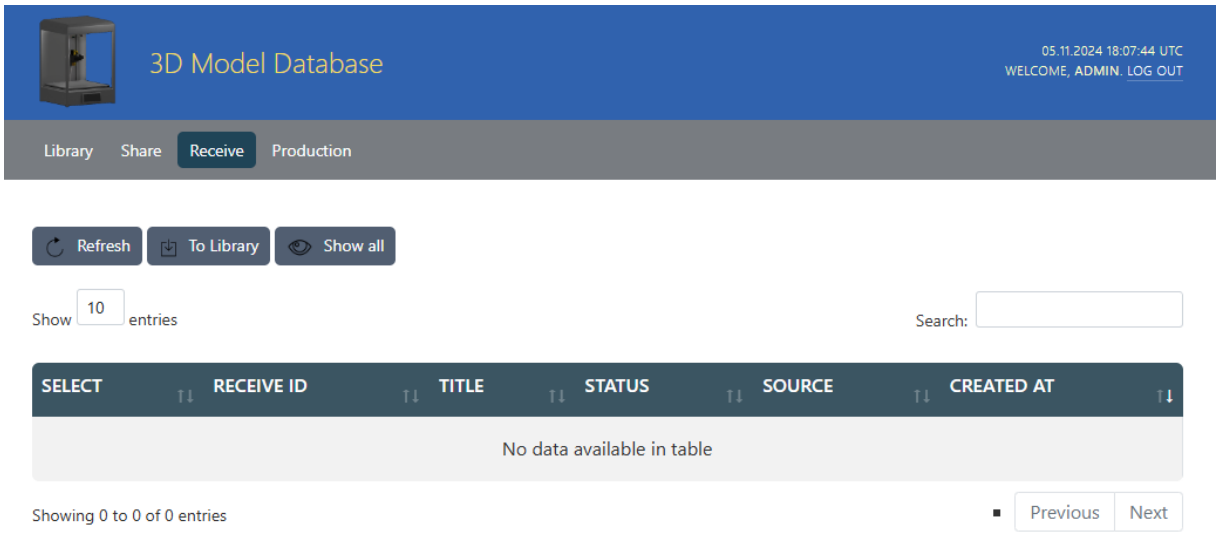
Taulukosta voidaan valita vain yksi rivi kerrallaan tapahtuman suorittamiseksi, minkä jälkeen se toteutetaan taulukon yläpuolella vasemmassa yläkulmassa olevan Push-painikkeen avulla. Näin varmistetaan, että jokainen tapahtuma on tarkkaan harkittu, eikä siten mahdollisia ylimääräisiä tapahtumia vahingossa siirtyisi toteutettavaksi samaan aikaan. Jatkossa tätä toimintoa voisi muuttaa tavalla, joka mahdollistaisi useampien eri jakotapahtumien suorittamisen. Esimerkiksi suoritettavaksi valituilla tapahtumilla olisi käytössä jono, jossa kukin kohta suoritetaan yksitellen.

3.4 Receive

Tietokantapalveluun on kehitetty mallien jakamisen lisäksi ominaisuus vastaanottaa malleja. Tämä on toteutettu Receive-sivulla, joka vastaa tiedon vastaanottamisesta palveluun rajapinnan kautta. Mallin tiedostomuoto tulee olla vain samankaltainen kuin mitä sallitaan liittää Library-taulussa olevien mallien liitetiedostoihin. JSON-tiedostoon sisältyy, esimerkiksi metatietokortti-PDF-, 3MF-, STL-, JSON- ja kuvatiedostot.

Vastaanotettaessa malleja, erityisesti muita malleja, tulee JSON-tiedoston rakenteen olla tietokantapalvelussa määritetyn rakenteen mukainen, jotta se saadaan onnistuneesti tuotua Receive-sivun vastaanottotaulukkoon. Kuvassa 15 näytetään vastaanottotapahtumien verkkosivun ulkonäköä.

Kuva 15. Vastaanottotapahtumien verkkosivu.

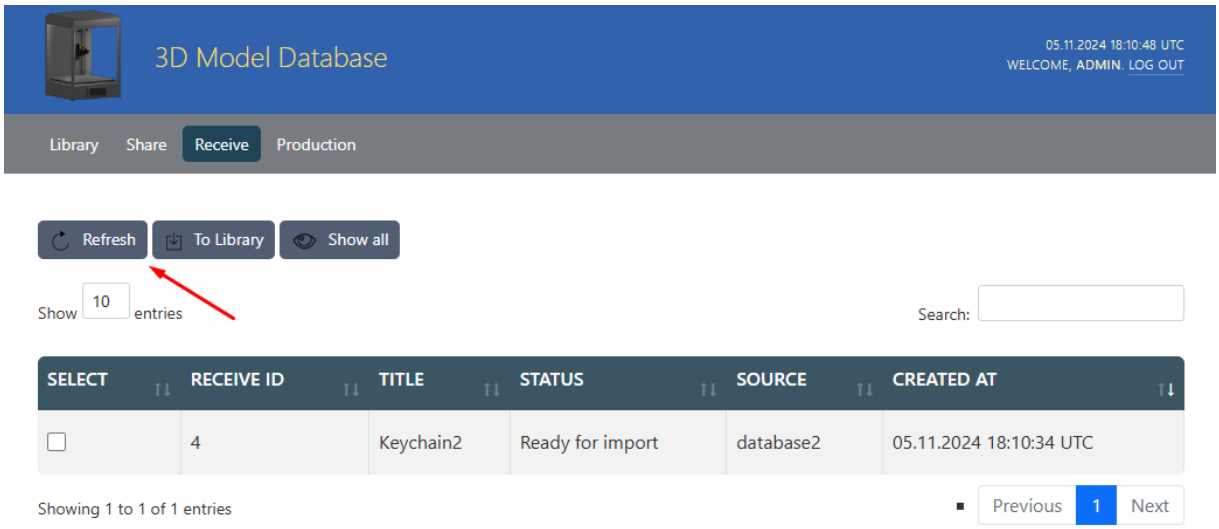


Sivulle on luotu vastaanottotaulukko, joka sisältää muutamia sarakkeita. Ne noudattavat pitkälti samaa rakennetta kuin muissa aiemmin esitellyillä sivuilla. Uusimpana tietona sarakkeissa ilmenee Source, eli lähde, jolla yksinkertaisuudessaan kerrotaan, mistä palvelusta tai tietokannasta kyseinen tuotu tieto on peräisin.

3.4.1 Datan vastaanottaminen palveluun esikatseltavaksi

Datan tuominen vastaanottotaulukkoon tapahtuu manuaalisesti taulukon yläpuolella vasemmassa yläkulmassa olevan Refresh-painikkeen avulla. Painike toteuttaa haun paikallisesta *receive*-hakemistosta, johon vastaanotetut tiedostot ovat saapuneet. Vastaanotettujen mallien JSON-tiedostot tuodaan taulukkoon, mikäli ne noudattavat palvelussa määritettyä tiedoston rakennetta. Kuvassa 16 esitetään Refresh-painike.

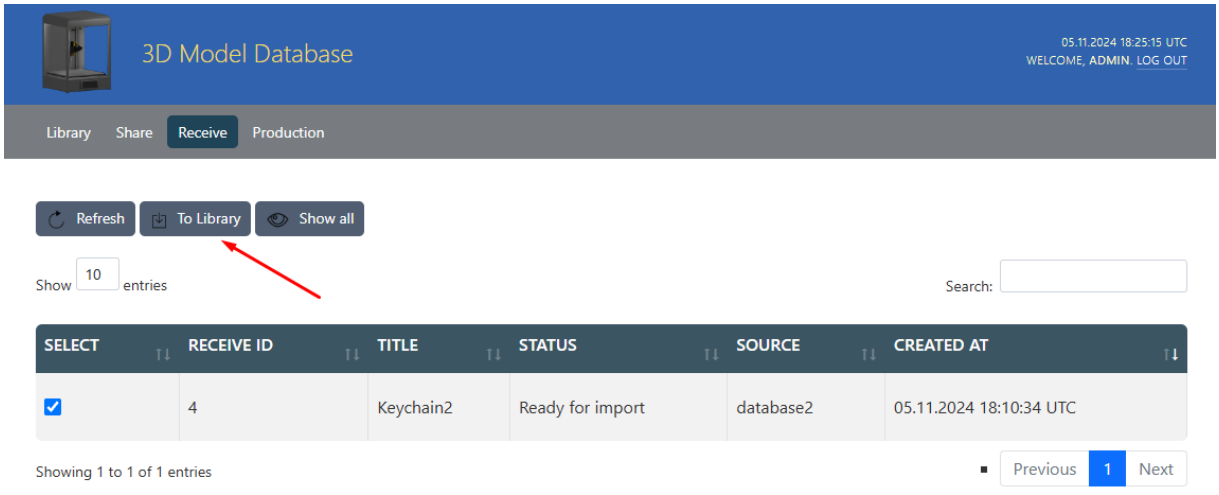
Kuva 16. Vastaanottotaulukko Refresh-painikkeen painamisen jälkeen.



3.4.2 Vastaanotetun datan siirtäminen tietokantaan

Vastaanottotaulukossa olevat tiedot malleista ovat nyt esikatseltavissa. Niitä ei ole vielä konkreettisesti tallennettu tietokantaan, vaan näistä taulukossa olevista riveistä voidaan valita, mitä halutaan palvelun tietokantaan tallentaa. Kuva 17 hahmottaa vastaanotetun datan tallentamista.

Kuva 17. Vastaanotetun mallin tallentaminen tietokantaan.



Datan lopullinen tallentaminen tietokantaan tapahtuu valitsemalla halutut rivit vastaanottotaulukosta, minkä jälkeen painetaan taulukon yläpuolella olevaa To Library -

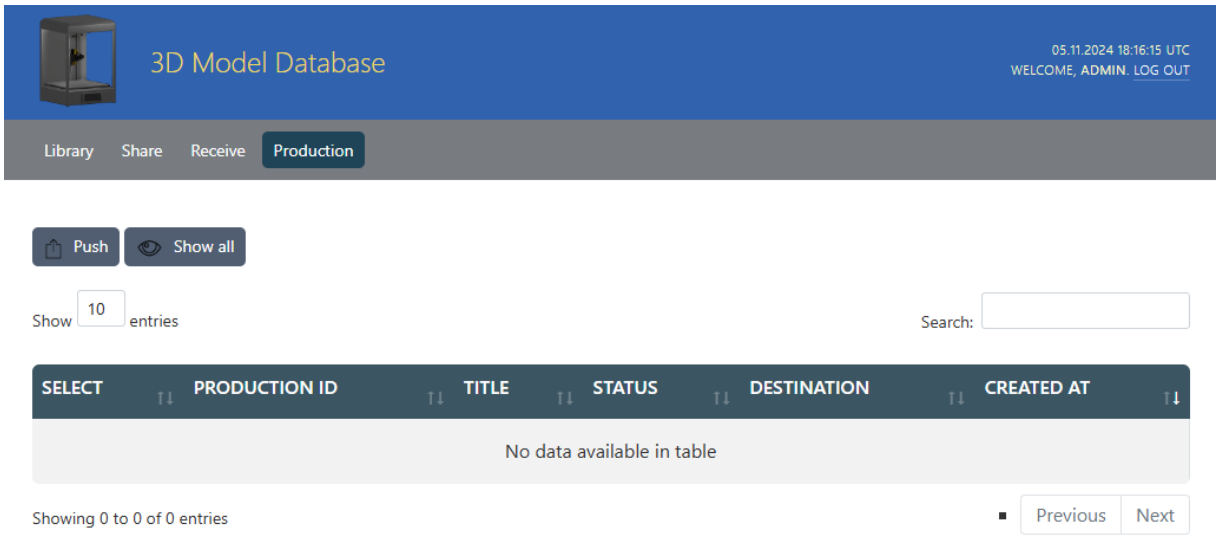
painiketta. Käyttäjän vahvistettua tallennettavat tiedot tietokantaan, haetaan palveluun liitetystä *receive*-hakemistosta malliin liittyvä JSON-tiedosto, joka sisältää mallin ja kaikki siihen lisätyt liitetiedosto. Toiminnon suoriuduttua, muuttuu vastaanottotaulukon rivin Status-tilaksi *imported*. Lopuksi verkkoselain uudelleenohjataan Library-sivulle, jonka taulukossa tallennettu malli näkyy.

3.5 Production

Tietokantapalveluun on haluttu kehittää mahdollisuus jakaa palveluun tallennettuja malleja muiden toimijoiden kesken. Näihin toimijoihin voisi kuulua esimerkiksi malleja valmistavan yrityksen palvelu tai toinen mallitietokanta.

On erityisen tärkeää lisättäessä mallitietokantaan uusia tietoja, tulee varmistua siitä, että malliin tulee olla liitettynä 3MF-tiedosto, ennen kuin se voidaan siirtää tuotantotapahtumiin. Työn tilaajan määrittämien ehtojen mukaisesti, on haluttu varmistaa, että jaettaessa malleja niitä valmistavalla yrityksellä on kaikki tarvittava data mallin tuottamiseen. Kuvassa 18 esitetään tuotantotapahtumien verkkosivun ulkonäköä.

Kuva 18. Tuotantotapahtumien verkkosivu.



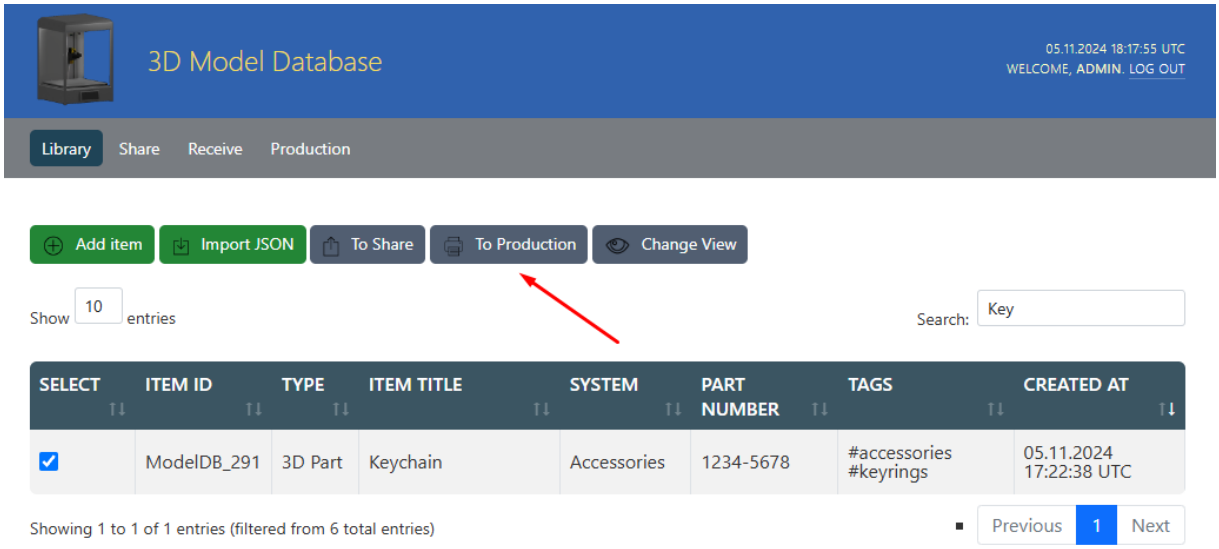
3.5.1 Mallin lisääminen tuotantotapahtumiin

To Production -painikkeen toiminto on hyvin samankaltainen kuin jakotapahtuman luomisessa. Sillä luodaan valituista Library-taulukon riveistä tuotantotapahtuma Production-

sivulle. Valitun mallin tai mallien yhteyteen tulee olla liitettynä 3MF-tiedosto kustakin mallista. Kyseistä tiedostomuotoa käytetään, sillä se on 3D-tulostinohjelmistojen hyväksymä.

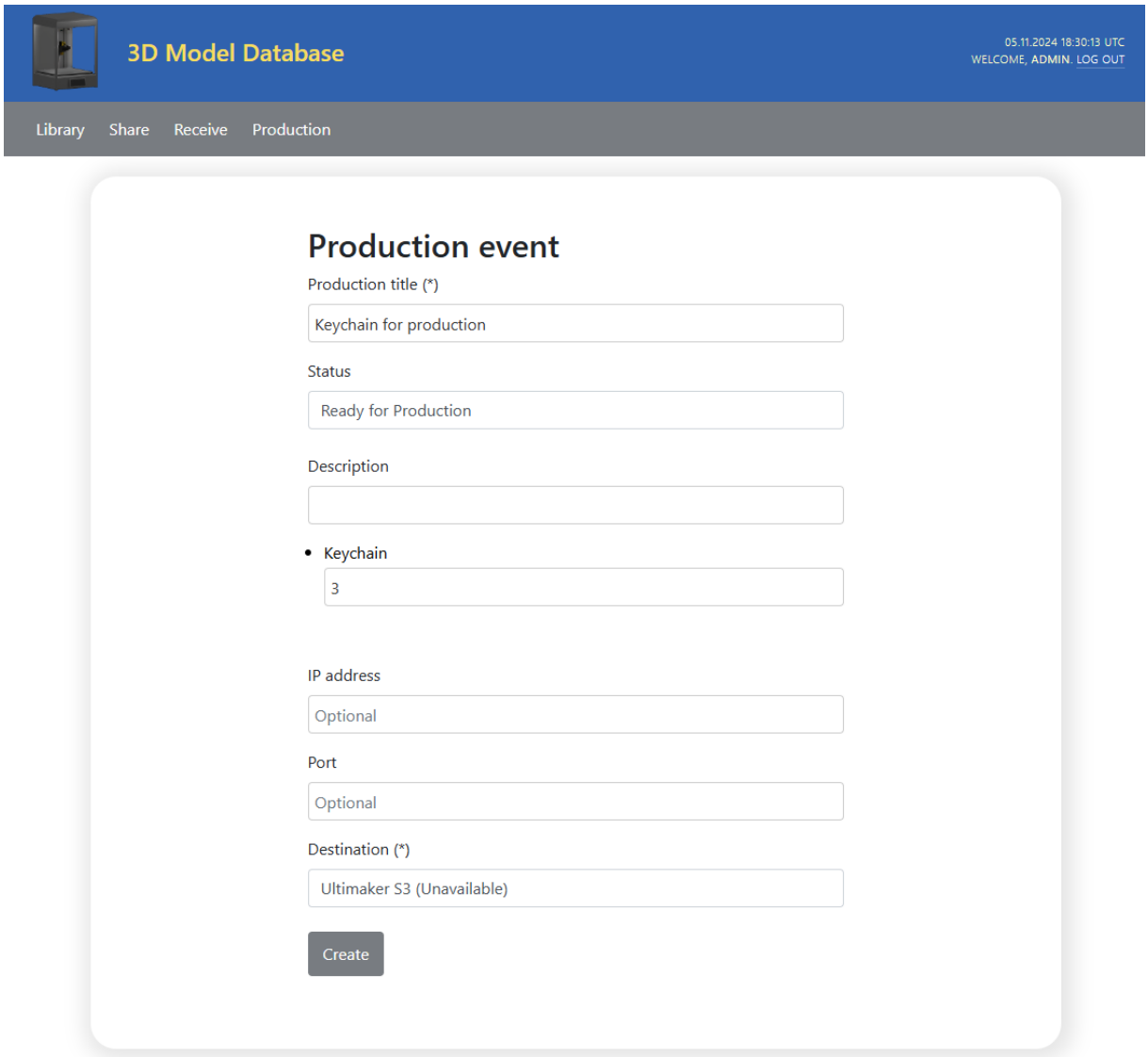
Tietokantapalvelun etusivun Library-taulukosta, voidaan valita yksi tai useampia rivejä tuotantotapahtumiin siirtämistä varten. Käyttäjän tehtyä valintansa, voidaan nyt hyödyntää tummansinisellä pohjalla olevaa To Production -painiketta mallin siirtämistä varten. Kuvassa 19 luodaan tuotantotapahtuma Library-taulukon avaimenperästä, johon on liitetty 3D-malli 3MF-tiedostomuodossa.

Kuva 19. Mallin lisääminen tuotantotapahtumiin.



To Production -painikkeella luodaan taulukosta valituista riveistä Production-sivulle uusi tuotantotapahtuma. Painikkeen painamisen jälkeen aukeaa tuotantotapahtuman lomake, joka sisältää useita täytettäviä tekstikenttiä. Kuvassa 20 luodaan tuotantotapahtuma Library-taulukon rivistä, joka sisältää avaimenperän.

Kuva 20. Tuotantotapahtuman luominen.



Tuotantotapahtuman lomake noudattaa hyvin pitkälti samaa kaavaa kuin jakotapahtuman lomake. Lomake sisältää perustietoja, mutta erityisesti siinä nousee esille mallin nimen alapuolella oleva “Amount” ja “Destination” -kohdat.

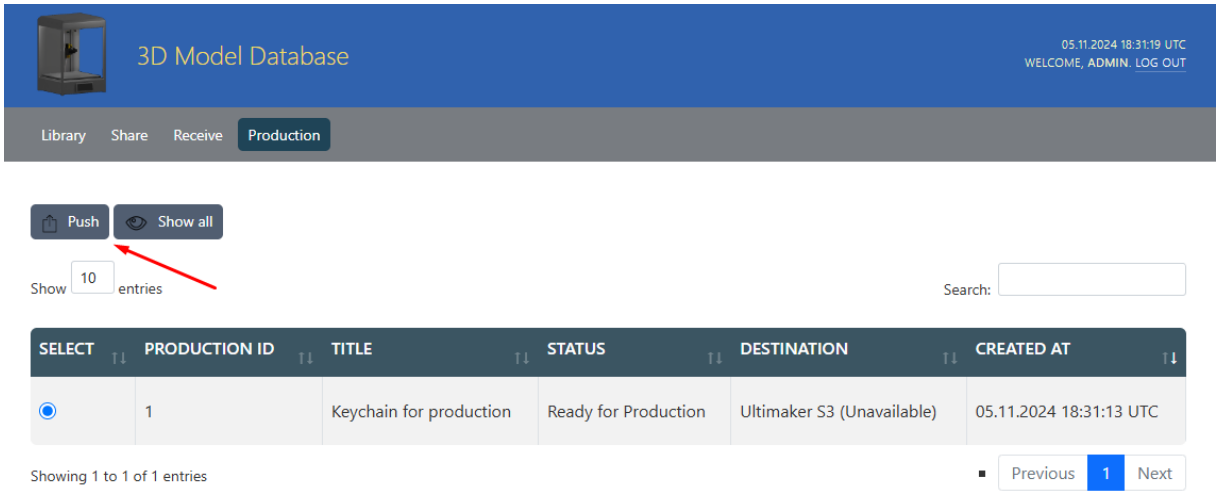
Amount, eli määrällä määritellään valmistajalle, että kuinka monta kyseistä mallia halutaan tuottaa tuotantovaiheessa.

Destination-kohdalla voidaan määritellä, että mille tulostimelle kyseinen malli halutaan lähettää. Yrityksellä voisi olla käytössään useita erilaisia 3D-tulostimia, joilla toteuttaa kyseisen mallin tulostaminen.

3.5.2 Tuotantotapahtuman suorittaminen

Tapahtumalle on määritelty Destination-kohdassa kohdetulostin, jolla kyseinen malli tullaan tulostamaan. Tuotantotapahtumataulukosta voidaan valita vain yksi rivi kerrallaan toteutettavaksi. Tällä ominaisuudella varmistetaan, että ei vahingossa toteutettaisi ylimääräisiä tapahtumia. Kuvassa 21 valitun tapahtuman suorittaminen.

Kuva 21. Tuotantotapahtuman suorittaminen Push-painikkeella.



Kun tapahtuma on valittu, tulee se toteuttaa Push-painikkeen avulla, mikä noudattaa hyvin samankaltaista toimintaperiaatetta kuin Share-sivulla jakotapahtumaa suoritettaessa. Toiminnon suoriuduttua, siirtyy malli 3D-tulostimelle tulostettavaksi, ja vastaanottotaulukon rivin Status-tila muuttuu *Success*.

3.6 Palvelinpuoli

3.6.1 Käyttöönotto

Toteutus aloitetaan tyhjästä Django 4.2.11-versiosta, ajamalla komento *django-admin startproject mysite* (Django, n.d.-o). Djangon konfigurointi tehdään *settings.py*-tiedostosta. Tässä tiedostossa määritellään Djangon eri asetuksia. s*ettings.py*-tiedostosta löytyy kohta *ALLOWED_HOSTS*, johon asetetaan *127.0.0.1*, joka kertoo, mitkä isäntänimet voivat käyttää sovellusta. Tässä tapauksessa sovellusta ajetaan paikallisesti, joten käytetään *localhost:*ia, eli *127.0.0.1*.

Sivujen toistuvan ulkonäön sekä rakenteen vuoksi, rakennetaan koodi tavalla, jossa jokainen sivu on samasta luokasta luotu olio. Näin koodin pituus lyhenee ja käytettävyys paranee. Tämän ansiosta koodin kehitys helpottuu ja nopeutuu, jolloin sivun pohja pysyy samana, mutta sisältö vain muuttuu. Luodun luokan nimeksi tulee *Builder.* Sivujen funktioiden etuliitteinä ovat *page_*, joissa on oletuksena *main, create ja edit,* mitkä määrittivät tietokannan objektin lisäämisen ja muokkaamisen tietokantaan. Sivu koostuu kyseisistä komponenteista: *model*, *form* ja *template*. *create*-funktio konstruktoi *form*-muuttujan *self.form*-luokkatyypin avulla, joka on käyttäjän määrittämä luokka, mikä perii *django.forms.Form* luokasta. Konstruktori ottaa *instance*-muuttujan, jolla päivitetään valmiina olevaa instanssia, jos tämän arvon asettaa muuksi kuin *None*.

```
form = self.form(request.POST, request.FILES, instance=model_id)
```

Luotaessa uutta verkkosivun näkymää, hyödynnetään *Builder*-luokkaa. *instance_page* määrittää verkkosivun URL-osoitteen. *instance_name* määrittää sivun nimen itsessään verkkosivulla, esimerkiksi painikkeen tekstin, joka ohjaa kyseiselle sivulle.

```
page_library = Builder()
page_library.instance_page = "library"
page_library.instance_name = "Library"
```

Sivun Django *context*-sanakirjasta löytyy muuttujat: *library, page_main, page_edit, page_download, page_delete ja extra_redirects.*

```
expanded_args = {
    'table': self.table,
    'page_main':f"{self.instance_page}",
    'page_edit':f"{self.instance_page}_edit",
    'page_download':f"{self.instance_page}_download",
    'extra_redirects':self.extra_redirects
}
```

Näiden argumenttien määritelmät ovat seuraavat: *table* määrittää pääsivulla näkyvän taulukon sarakkeiden otsikoiden nimet. Muuttujassa on kaksi listaa, sillä näkymää verkkosivulla voidaan vaihtaa, jolloin halutut muuttujat vaihtuvat. Vaihdosta kontrolloidaan *Builder*-muuttujalla *library_view.*

```
library_tables = [
    [
        "Item ID",
```

```
        "Type",
        "Item title",
        "System",
        "Part number",
        "Tags",
        "Created at",
    ],
    [
        "Item ID",
        "Type",
        "Item title",
        "System",
        "Manufacturing Method",
        "Material",
        "Tags",
        "Created at",
    ],
]

page_library.table = library_tables[page_library.library_view]
```

page_main, määrittää itse pääsivun Django *context*-sanakirjassa. Tämä muuttuja helpottaa uudelleen ohjausta pääsivulle templateissa. Nämä samat pätevät myös *page_edit* ja *page_download. extra_redirects* määrittää haluttujen sivujen uudelleenohjauksen. Näitä voidaan kutsua templatessa seuraavalla tavalla: *{% url extra_redirects.0 %}*, jossa numero määrittää uudelleenohjauksen indeksin.

```
page_library.extra_redirects = [
    'export_to_share',
    'export_to_production',
    'library_change_view'
]
```

Tämä on hyödyllinen, kun halutaan määrittää painikkeen uudelleenohjauksen toiselle sivulle ilman, että kovakoodaa URL:in templateen. Näin pystytään kontrolloimaan kaikki *pages.py*-tiedostosta, eli Python-tiedostosta.

Builder-luokasta löytyy *queryset*-muuttuja, joka on linkitetty yllä oleviin Model-muuttujiin. Nämä ovat Model-luokassa olevien muuttujien nimiä. Nämä merkkijonot tulee olla täysin samat kuin luokassa olevat muuttujien nimet. Näillä määritellään taulukko näkymän sarakkeet.

```
library_querysets = [
```

```
    [
        'id',
        'database_uid',
        'collection_type',
        'item_title',
        'system',
        'part_number',
        'tags',
        'created_at'
    ],
    [
        'id',
        'database_uid',
        'collection_type',
        'item_title',
        'system',
        'manufacturing_method',
        'material',
        'tags',
        'created_at'
    ]
]

page_library.queryset = library_querysets[page_library.library_view]
```

Lisättäessä painikkeita sivulle, ne sijoittuvat automaattisesti taulukon yläpuolelle ja kun niitä lisätään lisää, ne siirtyvät järjestyksessä edellisten painikkeiden oikealle puolelle. Painikkeen luomiseen tulee täyttää muutama muuttuja, jonka avulla voidaan luoda painikkeita ilman, että koskee Template-tiedostoihin.

Tämä esimerkki näyttää *Add item* -painikkeen toteutuksen Library-sivulle. Ensimmäinen täytettävä muuttuja on *url*, joka ohjaa painiketta painaessa tähän URL:iin. *text* määrittää verkkosivulla olevan painikkeen tekstin. *image_src* kertoo painikkeen kuvan tiedostopolun. *class* on HTML-dokumenttiin liittyvä painikkeen luokan nimi. *alt* on valinnainen kuvateksti, jos kuva ei ole näkyvissä.

```
page_library.buttons.append({
    "url":f"{page_library.instance_page}_create",
    "text":"Add item",
    "image_src": f'/{page_library.instance_page}_images/add.png',
    "class": "add-button",
    "alt": "Add",
})
```

Tähän mennessä on alustettu kaikki tarvittava tieto *Builder*-luokalle. Kun kaikki tarvittavat luokat on viimein määritelty, tulee sivun rakennus osio. Sivun komponentit rakennetaan alla olevan koodin mukaisesti.

```
page_library.create_model(model_metadata)
page_library.add_to_admin(ModelMetadataAdmin)
page_library.create_default_form("ModelMetadataForm", model_metadata)
```

create_model-funktio luo mallin *Builder*-olioon ja tekee tarvittavat toimenpiteet mallin luomiseksi. Vaihtoehtoisesti kutsuessa funktiota *add_to_admin*, se lisää luokan admin-paneliin. Viimeinen funktio luo lomakkeen kyseisestä luokasta.

Lisättäessä URL:eja *Builder*-olioon, tulee kutsua *add_to_urls*-funktiota, jossa ensimmäiseen funktion parametriin tulee haluttu osoite, jonka jälkeen määritetään funktio, joka kutsutaan URL:in yhdistäessä verkkoselaimella.

```
page_library.add_to_urls(f'{page_library.instance_page}/',
login_required(library))

page_library.add_to_urls(f'{page_library.instance_page}/create/',
login_required(library_create), f'{page_library.instance_page}_create')
```

Viimeiseksi sivun rakentamiseksi kutsutaan *add_to_builder*-funktiota alla olevan koodin mukaisesti.

```
add_to_builder(page_library)
```

3.6.2 Rakenne

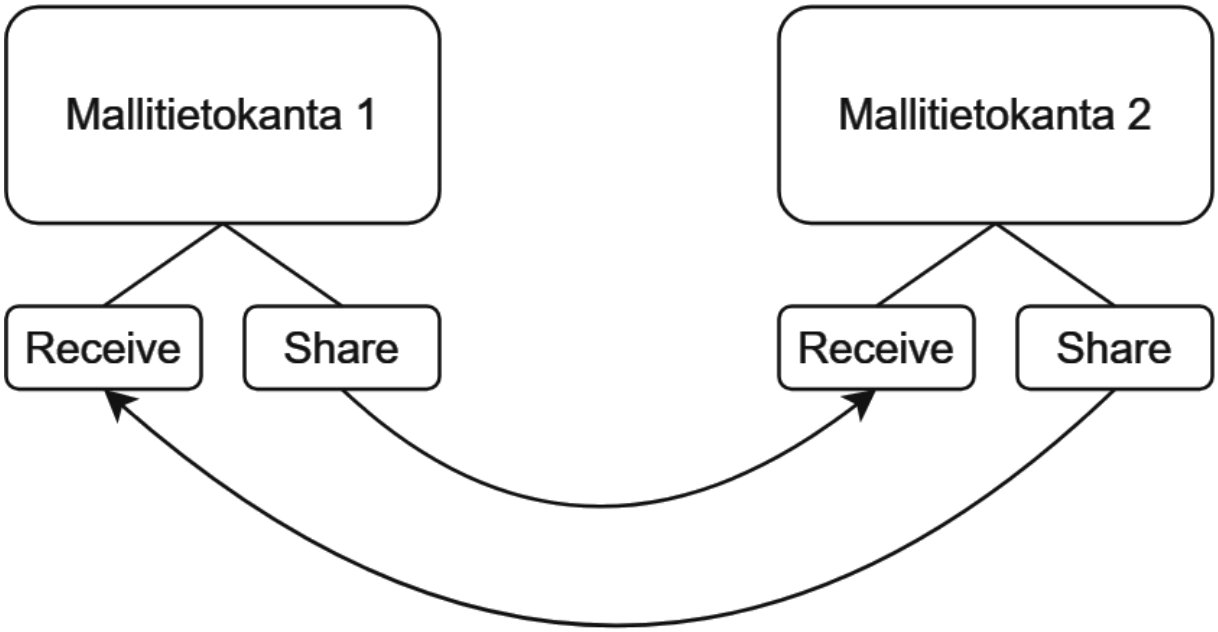
Samannäköisiä sivuja voidaan luoda helposti ilman, että joutuu muokkaamaan Template-tiedostoja. Kaikki löytyy yhdestä tiedostosta, jolloin kehittäminen on helpompaa pienemmissä projekteissa. Jokainen sivu voitaisiin myös jakaa omiin tiedostoihin, jolloin kaikki tuodaan *pages.py* tiedostoon. On suositeltu kuitenkin pitämään kaikki sivut yhdellä sivulla, sillä jos on tarve käyttää toisen sivujen *Builder*-muuttujia, pystytään kätevästi käyttää näitä samasta tiedostosta, mikä parantaa työn tuottavuutta.

Mallitietokannasta on mahdollista lähettää malleja toiseen mallitietokantaan. Tiedonsiirtoon on olemassa eri ratkaisuja, tässä avataan pari tekniikkaa. Ensimmäinen menetelmä mallien jakoon on SFTP-protokolla (SSH File Transfer Protocol). Aluksi tulee luoda SSH-palvelin, joka palvelee isäntänä. Pythonista löytyy kolmannen osapuolen kehittämä Paramiko-kirjasto, jolla voidaan yhdistää SFTP-protokollaa käyttäen isäntään (Paramiko, n.d.).

Projektista löytyy *site-config.json*-tiedosto, jossa voidaan määritellä erilaisia verkkosivun asetuksia. Tänne voidaan asettaa SFTP-protokollayhteyden eri parametreja ja nämä ovat seuraavat: ip, port, username, password, SSH-yksityisavain (SSH private key). Mallitietokantaa käynnistäessä, palvelin hakee tarvittavat tiedot tiedostosta ja käyttää niitä mallien lähettämisessä.

Toisessa menetelmässä löytyy hakemistosta *receive*- ja *share*-hakemisto, jossa saapuvat mallit tulevat receive-hakemistoon, josta voidaan nämä hakea painamalla verkkosivulla olevaa *refresh*-päivityspainiketta. s*hare*-hakemisto vastaa mallien lähettämisestä. Mallit voidaan esimerkiksi hakea tästä hakemistosta tiedonsiirtoprotokollalla. Kuvassa 22 esitetään tiedonsiirtoa kahden mallitietokannan välillä.

Kuva 22. Kahden mallitietokannan välinen tiedonsiirto.



Tietokannasta on mahdollista ladata malleja yksittäisessä JSON tai ZIP-muodossa käyttäen Pythonin JSON- tai ZIP-kirjastoa. Tietokannasta voidaan ottaa myös varmuuskopio, jossa mallien binääritiedostot ladataan ZIP-tiedostona. Koodi luo jokaisesta mallista hakemiston, josta löytyy lisätyt mallitiedostot, metatietokortti-PDF, sekä yksittäisen JSON-tiedoston. JSON-tiedosto voidaan tuoda tietokantaan, jolloin malli tai mallit pystytään lisäämään tietokantaan helposti.

Ladattaessa tiedostoja verkkosivun kautta, hyödynnettiin Djangon FileField-kenttää, joka tallentaa tiedoston koodissa määritettyyn tiedostopolkuun. Tiedostot tallennetaan paikalliselle tietokoneelle binäärinä.

3.6.3 Kolmiulotteisten kappaleiden valmisteleminen käyttöliittymään

Kolmiulotteisten mallien renderöintiin käytetään Three.js-kirjastoa. Se on helppokäyttöinen, kevyt ja usealle selaimelle yhteensopiva 3D-tietokonegrafiikka kirjasto, joka käyttää WebGL-renderöintiä. (Three.js, n.d.) Toteutuksessa hyödynnetään tätä kirjastoa kolmiulotteisten kappaleiden esittämiseen. Näkymää voidaan liikuttaa, suurentaa tai pyörittää kappaleen ympärillä. Näkymästä löytyy myös eri akseleiden koordinaatistovektorit. Pienentämällä näkymän *ambient lighting* ja lisäämällä muutama valonlähde, saadaan lopputuloksesta varjomainen kappale, josta heijastuu *specular lighting.* Tämä luo lopputuloksesta miellyttävän näköisen ja selkeästi erotettavan mallin.

Mallin binääritiedosto voidaan hakea käyttäen Django-mallin *pk*-pääavainta. Haun jälkeen saadaan yksi malli-instanssi, josta voidaan ladata Django FileField-kentän tallentaman binääritiedoston nimen avulla levyltä. Tämä tiedosto tulee enkoodata Base64-muodoksi, jonka jälkeen voidaan syöttää tämä arvo Django kontekstiin. Ladattaessa mallia HTML-tiedostossa, tulee malli dekoodata JavaScript *Uint8Array* tyyppiin, jonka jälkeen puskuri syötetään *STLLoader.parse*-funktioon *THREE.Mesh*-funktion käsittelyyn. Lopuksi saatu *mesh*, voidaan liittää *THREE.Scene*:en kutsumalla *scene.add(mesh)*, eli asetettuun kolmiulotteiseen avaruuteen.

```
const loader = new STLLoader();

var model_data = '{{model_data}}';
let binaryString = atob(model_data);
let len = binaryString.length;
let bytes = new Uint8Array(len);

for (let i = 0; i < len; i++) {
    bytes[i] = binaryString.charCodeAt(i);
}

var geometry = loader.parse(bytes.buffer);
var mesh = new THREE.Mesh(geometry, material);
scene.add(mesh);
```

Kolmiulotteisten mallien tarkasteleminen verkkoselaimessa on hyödyllinen ominaisuus tähän kokonaisuuteen. Tällä voidaan nähdä kyseinen malli ilman, että se täytyy ladata ja viedä erilliseen 3D-mallinusohjelmistoon.

4 Yhteenveto

Opinnäytetyön tavoitteena oli kehittää verkkosivuilla toimiva tietokantapalvelu, jota käytetään kolmiulotteisten kappaleiden tallentamiseen sekä hallintaan. Toteutuksessa päätettiin käyttää Django-verkkokehystä Pythonin yksinkertaisuuden, helppokäyttöisyyden, nopean kehityksen, sekä laajan kolmannen osapuolten kirjastokokoelman vuoksi. Työllä pyrittiin selvittämään, että millaisia vaatimuksia mallitietokannan kehittämisessä ja ylläpitämisessä tulee huomioida. Mallitietokanta toteutettiin alustavana konseptitodistuksena työn tilaajan toiveiden mukaisesti. Työhön sisältyi muutamia tutkimuskysymyksiä, joiden avulla haettiin ratkaisuja mallitietokannan toteuttamiseen ja jatkokehittämiseen.

Kehityksen parissa ilmeni erilaisia ominaisuuksia, joita palvelusta tulisi löytyä. Palvelussa tulee olla tallennusmahdollisuus sekä kommunikaatio tietokannan, käyttöliittymän ja palvelinpuolen välillä. Lisäksi mallien lisääminen, muokkaaminen ja poistaminen tietokannasta käyttöliittymän avulla on olennainen osa tätä palvelua. Palvelusta tulisi löytyä tapahtumarekisteri, eli loki, joka helpottaa ylläpitoa ja hallintaa. Malleja halutaan myös ladata, jakaa ja vastaanottaa esimerkiksi muista vastaavista tietokantapalveluista. Kokonaisuutta arvioiden tavoitimme nämä tarpeelliset toiminnallisuudet erinomaisesti.

Kolmiulotteisille malleille voitaisiin suorittaa *slice*-prosessi Ultimaker:in CuraEngine-konsoliapplikaation avulla. Prosessissa luodaan G-code käskyt STL-tiedoston perusteella. (Ultimaker, 2023). Tämän jälkeen tiedosto lähetettäisiin esimerkiksi verkon yli 3D-tulostimelle tulostettavaksi.

Palvelun kommunikoinnin toteuttaminen kolmannen osapuolen ratkaisuun olisi toteutettavissa REST API -integraation avulla. Ohjelmointirajapinnan avulla mahdollistetaan sujuva tiedonsiirto eri rajapintojen, sekä käyttöliittymän ja palvelinpuolen välillä. Djangolle on kehitetty kolmannen osapuolen toimesta REST-kehys, jolla voidaan ratkaista tämä tiedonvaihto. Tämä mahdollistaisi laajemmin eri verkkopalveluiden integroimisen palvelun yhteyteen.

Työssä opimme web-sovelluskehityksen keskeisistä työmenetelmistä, kuten esimerkiksi siihen kuuluvista suunnittelusta, kehitystyöstä, testaamisesta ja ylläpidosta. Lopputuloksena

työstä saatiin hyvin tietoa siitä, kuinka kyseistä mallitietokantaa tulisi jatkokehittää, sekä mahdollisista haasteista, joita tulee kehityksen yhteydessä. Työ voisi tulevaisuudessa toimia mallina kehitettäessä uutta ratkaisua.

Lähteet

Alexandrea, J. (19.5.2023). *What Is Bootstrap?*. Hostinger. Haettu 14.11.2024 osoitteesta https://www.hostinger.com/tutorials/what-is-bootstrap/

Cuello, C. (17.7.2024). *Understanding The Different Types Of Databases & When To Use Them.* Rivery. Haettu 15.11.2024 osoitteesta https://rivery.io/data-learning-center/database-types-guide/

Codecademy. (18.10.2021). *What Is Web Development?*. Haettu 14.11.2024 osoitteesta https://www.codecademy.com/resources/blog/what-is-web-development/

DataTables. (n.d.-a). *Installation*. Haettu 13.11.2024 osoitteesta https://datatables.net/manual/installation

DataTables. (n.d.-b). *Manual*. Haettu 12.11.2024 osoitteesta https://datatables.net/manual/

DataTables. (n.d.-c). *Order.* Haettu 13.11.2024 osoitteesta https://datatables.net/reference/option/order

DataTables. (n.d.-d). *columnDefs*. Haettu 13.11.2024 osoitteesta https://datatables.net/reference/option/columnDefs

DataTables. (n.d.-e). *columns.type*. Haettu 13.11.2024 osoitteesta https://datatables.net/reference/option/columns.type

DataTables. (n.d.-f). *Date (dd/mm/YYYY hh:ii:ss).* Haettu 13.11.2024 osoitteesta https://datatables.net/plug-ins/sorting/date-euro

DataTables. (n.d.-g). *columns.orderable.* Haettu 13.11.2024 osoitteesta https://datatables.net/reference/option/columns.orderable

DataTables. (n.d.-h). *DataTables*. Haettu 12.11.2024 osoitteesta https://datatables.net/

Django. (n.d.-a). *Models*. Haettu 24.10.2024 osoitteesta https://docs.djangoproject.com/en/5.1/topics/db/models/

Django. (n.d.-b). *Views*. Haettu 24.10.2024 osoitteesta
https://docs.djangoproject.com/en/5.1/topics/http/views/

Django. (n.d.-c). *Templates.* Haettu 24.10.2024 osoitteesta
https://docs.djangoproject.com/en/5.1/topics/templates/

Django. (n.d.-d). *Migrations*. Haettu 24.10.2024 osoitteesta
https://docs.djangoproject.com/en/5.1/topics/migrations/

Django. (n.d.-e). *FAQ: General*. Haettu 24.10.2024 osoitteesta
https://docs.djangoproject.com/en/5.1/faq/general/

Django. (n.d.-f). *Template inheritance*. Haettu 8.11.2024 osoitteesta
https://docs.djangoproject.com/en/5.1/ref/templates/language/#template-inheritance

Django. (n.d.-g). *Templates*. Haettu 10.11.2024 osoitteesta
https://docs.djangoproject.com/en/5.1/ref/templates/language/#templates

Django. (n.d.-h). *Syntax*. Haettu 11.11.2024 osoitteesta
https://docs.djangoproject.com/en/5.1/topics/templates/#syntax

Django. (n.d.-i). *Runserver.* Haettu 15.11.2024 osoitteesta
https://docs.djangoproject.com/en/5.1/ref/django-admin/#runserver

Django. (n.d.-j). *Flush*. Haettu 15.11.2024 osoitteesta
https://docs.djangoproject.com/en/5.1/ref/django-admin/#flush

Django. (n.d.-k). *The Django admin site*. Haettu 15.11.2024 osoitteesta
https://docs.djangoproject.com/en/5.1/ref/contrib/admin/

Django. (n.d.-l). *Tags*. GitHub. https://github.com/django/django/tags

Django. (n.d.-m). *Download* [kuva]. Haettu 5.11.2024 osoitteesta
https://www.djangoproject.com/download/

Django. (n.d.-n). *Django*. Haettu 2.8.2024 osoitteesta
https://www.djangoproject.com/

Django. (n.d.-o). *Writing your first Django app, part 1*. Haettu 18.11.2024 osoitteesta https://docs.djangoproject.com/en/5.1/intro/tutorial01/

Emanuilov, S. (25.2.2024). *Is Django dead?.* UnfoldAI. Haettu 8.11.2024 osoitteesta https://unfoldai.com/is-django-dead/

Gadhavi, M. (17.6.2024). *Most Popular Backend Frameworks for Web Development in 2024*. Radix. Haettu 9.7.2024 osoitteesta https://radixweb.com/blog/best-backend-frameworks

Hashemi-Pour, C. & Churchville, F. (2024, huhtikuu). *user interface (UI)*. TechTarget. Haettu 14.11.2024 osoitteesta https://www.techtarget.com/searchapparchitecture/definition/user-interface-UI

Indeed. (2.7.2024a). *What is website maintenance? (Reasons, components and costs)*. Haettu 14.11.2024 osoitteesta https://uk.indeed.com/career-advice/career-development/what-is-website-maintenance

Indeed. (11.9.2024b). *What Is a User Interface (UI)?*. Haettu 14.11.2024 osoitteesta https://www.indeed.com/career-advice/career-development/user-interface

Interaction Design Foundation. (2.6.2016). *What is User Interface (UI) Design?*. Haettu 14.11.2024 osoitteesta https://www.interaction-design.org/literature/topics/ui-design

Intelegain Technologies. (6.8.2019). *What are web frameworks and why you need them?*. Medium. Haettu 24.10.2024 osoitteesta https://intelegain-technologies.medium.com/what-are-web-frameworks-and-why-you-need-them-c4e8806bd0fb

Korsun, J. (5.8.2024). *10 Popular Django Websites That You Probably Know*. Django Stars. Haettu 15.11.2024 osoitteesta https://djangostars.com/blog/10-popular-sites-made-on-django/

Letusheva, V. (3.6.2024). *The State of Django 2024*. JetBrains Blog. Haettu 8.11.2024 osoitteesta https://blog.jetbrains.com/pycharm/2024/06/the-state-of-django/

MDN Web Docs. (23.7.2024). *What is a web server?*. Haettu 12.8.2024 osoitteesta https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server

Microsoft. (22.8.2024). *Scalability and performance targets for VM disks*. Haettu 18.11.2024 osoitteesta https://learn.microsoft.com/en-us/azure/virtual-machines/disks-scalability-targets

Microsoft. (n.d.). *Tietokannan suunnittelun perusteet*. Haettu 24.10.2024 osoitteesta https://support.microsoft.com/fi-fi/topic/tietokannan-suunnittelun-perusteet-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5

Oyom, A. N. (19.7.2017). Understanding the MVC pattern in Django. Medium. Haettu 31.10.2024 osoitteesta https://medium.com/shecodeafrica/understanding-the-mvc-pattern-in-django-edda05b9f43f

Paessler. (n.d.). *Server - Definition and details*. Haettu 12.8.2024 osoitteesta https://www.paessler.com/it-explained/server

Paramiko. (n.d.). *Paramiko*. Haettu 18.11.2024 osoitteesta https://www.paramiko.org/

Rouse, M. (15.10.2013). *Front-End Developer*. Techopedia. Haettu 7.8.2024 osoitteesta https://www.techopedia.com/definition/29569/front-end-developer

Rouse, M. (24.2.2017). *Back-End Developer*. Techopedia. Haettu 8.8.2024 osoitteesta https://www.techopedia.com/definition/29568/back-end-developer

6sense. (n.d.). *Market Share of Django.* Haettu 8.11.2024 osoitteesta https://6sense.com/tech/web-framework/django-market-share

StackHawk. (27.4.2022). *Guide to Security in Django*. Haettu 15.11.2024 osoitteesta https://www.stackhawk.com/blog/guide-to-security-in-django/

Stack Overflow. (19.5.2023). *Stack Overflow Developer Survery 2023* [kuva]. Haettu 11.11.2024 osoitteesta https://survey.stackoverflow.co/2023/#most-popular-technologies-database-prof

Systemd. (n.d.). *System and Service Manager*. Haettu 24.10.2024 osoitteesta https://systemd.io/

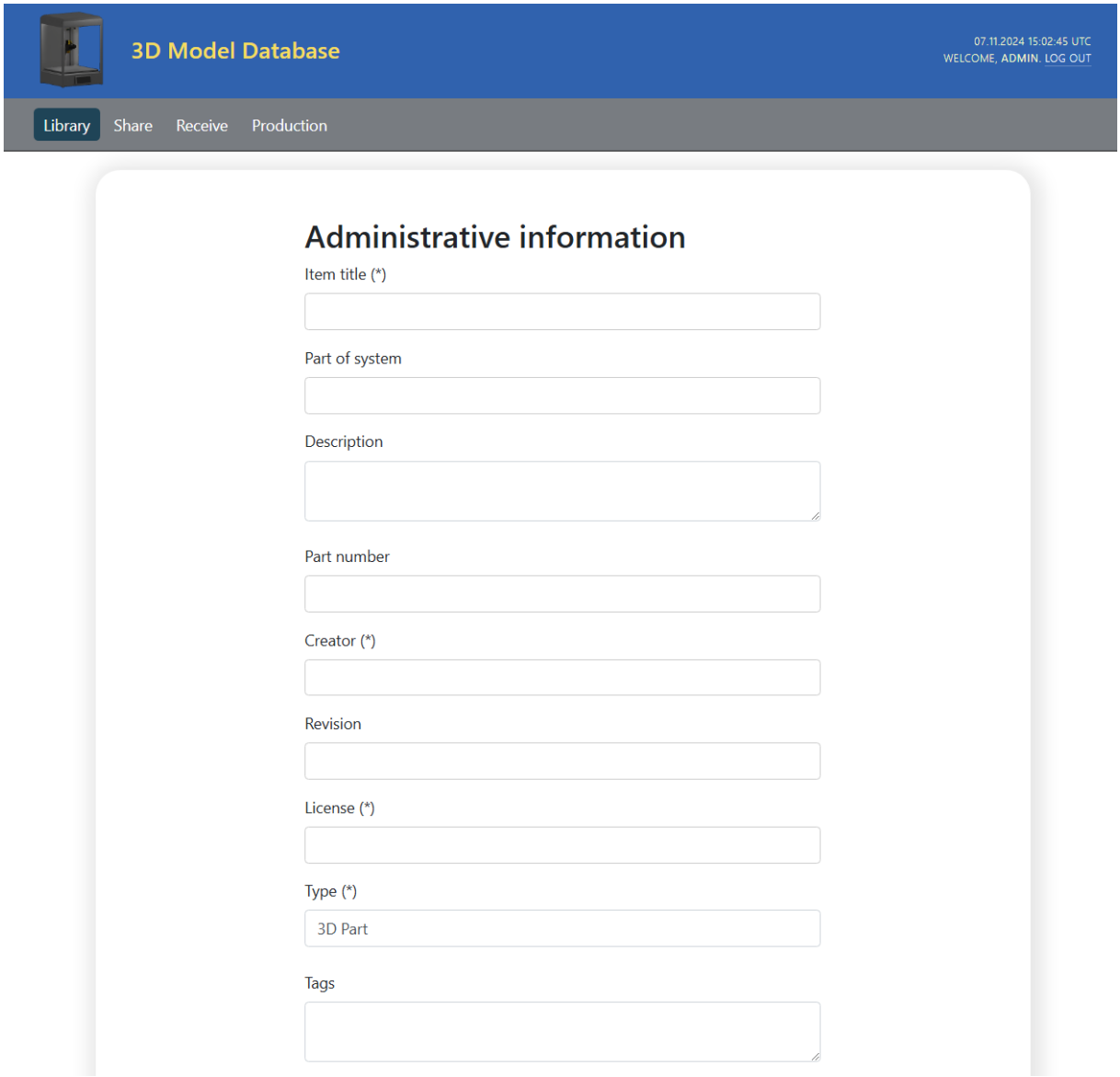
Three.js. (n.d.). *Three.js*. GitHub. Haettu 18.11.2024 osoitteesta https://github.com/mrdoob/three.js

Ultimaker. (17.2.2023). *Slicing*. GitHub. Haettu 14.11.2024 osoitteesta https://github.com/Ultimaker/CuraEngine/wiki/Slicing

Webb, M. (14.5.2024). *Web Development*. Techopedia. Haettu 2.8.2024 osoitteesta https://www.techopedia.com/definition/23889/web-development

W3Techs. (n.d.). *Usage statistics of PHP for websites*. Haettu 4.11.2024 osoitteesta https://w3techs.com/technologies/details/pl-php

Liite 1. Mallin luomislomake



X (mm)

Y (mm)

Z (mm)

Manufacturing method

Material

Additional information

Attachments

Manufacturing Info (PDF)

Choose a file Browse

Preview Image

Choose a file Browse

3MF File

Choose a file Browse

STL File

Choose a file Browse

Project File

Choose a file Browse

Submit