TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Annabel Põder 164049IABB

ONBOARDING RAKENDUSE ARENDAMINE

Bakalaureusetöö

Juhendaja: Inna Švartsvam
MSc

Tallinn 2020

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Annabel Põder

[24.05.2020]

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua veebirakendus, mis kogub arenduses olevatele projektide keskkonna loomiseks vajalikud nõuded, mille põhjal luuakse Jira keskkonda vastavad *Taskid*. Loodava lahenduse abil lihtsustatakse ja kiirendatakse andmete edastamise ning *Taskide* loomise protsessi.

Autor annab ülevaate olemasolevast lahendusest ettevõttes ja toob välja selle tugevad ning nõrgad küljed. Lisaks analüüsib autor projekti *onboardimiseks* vajaminevaid andmeid. Välja on toodud ka nõuded loodavale rakendusele, mis on kirja pandud ettevõtte poolt määratud ning analüüsist saadud tulemuste põhjal. Töös on kirjeldatud ettevõtte jaoks loodud veebirakendust.

Töö tulemusena valmis töötav veebirakendus, mis kogub vormi abil vajalikud andmed ja saadud andmete põhjal loob ülesanded ettevõtte poolt kasutatavasse keskkonda.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 29 leheküljel, 6 peatükki, 6 joonist.

Abstract

Onboarding application development

The aim of this bachelor's thesis is to create a web application that collects the requirements necessary for the development of the environment of the projects under development, on the basis of which the Tasks corresponding to the Jira environment are created. The created solution simplifies and speeds up the process of data transfer and creation of Tasks.

The author gives an overview of the existing solution in the company and highlights its strengths and weaknesses. In addition, the author analyzes the data needed to onboard the project. The requirements for the application to be created are also set out, which is written on the basis of the results determined by the company and obtained from the analysis. The paper describes the web application created for the company.

As a result of the work, a working web application was completed, which collects the necessary data using the form and, based on the received data, creates tasks in the environment used by the company.

The thesis is in Estonian and contains 29 pages of text, 6 chapters, 6 figures.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*. Rakendusliides
Bitbucket link	Lähtekoodi repositooriumi asukoht
Confluence	Confulence programmi link, kus on projekti dokumentatsioon
CPU	*Central processing unit*, protsessor
DNS	*Domain name system* ,domeeni nimi
HTTP	*Hypertext Transfer Protocol*, hüperteksti edastusprotokoll
Jira	Atlassiani projektihaldus vigade jälgimiseks ja kirja panemiseks
JSON	*JavaScript Object Notation*, lihtsustatud andmevorming
Onboarding	Uue rakenduse kasutusele võtt
PAI	*Platform and Infrastructure*, platvorm ja infrastruktuur
REST	*Representational State Transfer*, veebiteenuse arhitektuuri ülesehitamise viis
Subtask	Alamülesanne, mis luuake Jira keskkonda põhiülesande valmimise abistamiseks
Task	Põhiülesanne, mis luuakse Jira keskkonda

Sisukord

Jooniste loetelu

1 Sissejuhatus

1.1 Taust ja probleem

Eesti Post OÜ (edaspidi ettevõte) Platvormi ja Infrastruktuuri (edaspidi PAI) osakond tegeleb, nagu nimigi ütleb platvormide haldamise ja arendamisega. Üheks nende ülesannetest on arendusfaasis kui ka arendatud rakenduste platvormide loomine ja haldamine. Selleks, et luua platvormi, kus rakendus tööle hakkab on vaja kirja panna nõuded keskkonnale.

Ettevõttes on kasutusel Jira nimeline ülesannete haldus keskkond, mis aitab ettevõtte meeskondadel erinevaid tööülesandeid jagada ning hallata. Seda kasutatakse nii veateadete raporteerimiseks kui ka tarkvaraarenduse nõuete ning kasutusjuhtude kirjeldamiseks. [1]

Siiamaani toimubki keskkonna nõuete ehk *onboardingu* kirjapanek läbi Jira. Arendustiimid kes on rakendust valmis saamas vajavad keskkonda, kus oma rakendus tööle panna. Selleks loob rakenduse administraator (edaspidi Tellija) Jira keskkonnas vastava *Taski*, milles kirjeldab nõudeid mis on vajalikud loodava rakenduse keskkonnaks. Keskkonna nõuete kirjeldus on aga vabas vormis, mis omakorda ongi nõrgaks lüliks. Paljudel juhtudel on nõuete kirjeldus ebapiisav ning vastutav isik (Assignee), kellele määratakse sissetulev *Task*, peab kontakteeruma uuesti Tellijaga ja täpsustama nõudeid. Selleks aga kulub väärtusliku aega nii Tellija kui vastutava isikul, mil oleks võimalik aega paremini kasutada.

1.2 Töö eesmärk

Käesoleva bakalaureusetöö eesmärgiks on luua veebirakendus, mis kogub arenduses olevatele projektide keskkonna loomiseks vajalikud nõuded, mille põhjal luuakse Jira keskkonda vastavad *Taskid* ja *Subtaskid*. Loodava lahenduse abil lihtsustatakse ja kiirendatakse andmete edastamise ning *Taskide* loomise protsessi.

1.3 Töö struktuur

Käesoleva töö teises peatükis on välja toodud olemasoleva lahenduse kirjeldus ning analüüs. Autor toob välja selle nõrgad kui ka tugevad küljed. Lisaks kirjeldab ja analüüsib rakenduse vormi jaoks vajaminevaid andmeid ja keskkondi.

Kolmandas peatükis kirjeldab autor veebirakendusele esitatud nõudeid ning tehnoloogiad mida loodava veebirakenduse puhul kasutatakse.

Neljandas peatükis kirjeldab autor veebirakenduse jaoks loodud reaalset rakendust piltide ja kirjelduse.

Viiendas peatükis toob autor välja kuidas veebirakendust veelgi täiendada, et muuta rakendust mugavamaks ja kiiremaks.

2 Olemasolev lahendus ja vajaminevate andmete analüüs

Käesolev peatükk kirjeldab olemasolevat lahendust. Välja tuuakse selle tugevad ja nõrgad küljed. Viiakse läbi projekti *onboardingu* jaoks vajamineva informatsiooni analüüs ettevõtte olemasoleva lahenduse analüüsist saadud tulemuste ja artiklites leiduva info alusel. Lisaks uuritakse ettevõtte poolt kasutatavate keskkondade loomiseks vajaminevaid andmeid.

2.1 Olemasoleva lahenduse kirjeldus

Siiani on keskkonna nõuete edastamiseks PAI administraatoritel kasutusel olnud Jira keskkond. Tellija loob uue *Taski*, mis on ülesanne, hõlmates endas informatsiooni ülesande kohta mis tuleb lahendada. [2]

Taski loomisel küsitakse ülesandele loodavat nime, mis tüüpi probleem (Issue) on, kellele loodav *Task* suunatakse, ehk kes on vastutav isik sellele *Taskile*, kokkuvõte ning kirjelduse tekstiväli. Tellijale ei ole ette antud ühtegi juhist ega malli mille põhjal keskkonna nõudeid kirjeldada. See valik on antud Tellijale vabas vormis esitamiseks. Väljade täitmise järel luuakse vastav *Task* loodud nimega ning suunatakse määratud vastutavale isikule.

Task ilmub vastutava isiku profiilile, kes määrati *Taski* luues. Sellele vajutades on näha *Taski* kirja pandud probleemi tüüp, kokkuvõte ja kirjeldus.

Projekti *onboardingu* puhul tuleb kirjeldusse kirja panna keskkonna nõuded. Kuna paljudel juhtudel on nõuded ebapiisavalt kirja pandud, tuleb vastutaval isikul ühendust võtta Tellijaga, et täpsustada vajalikke andmeid. Kui kogu informatsioon on olemas asub vastutav isik nõuete põhjal *Subtaske* looma. *Subtask* on alamülesanne, mis on vajalik ülesande täitmiseks. [2] Igale *Subtaskile* määratakse omakorda vastutav isik, kes hakkab selle ülesandega tegelema. Seda kõike teeb vastutav isik käsitsi, mis on aeganõudev tegevus. Selline protsessi käik on väga ajakulukas kõigile pooltele. Sellest tekkis ka vajadus luua uus lahendus.

2.2 Olemasoleva lahenduse analüüs

Olemasoleva lahenduse idee on väga hea. Jira on keskkond, mida kasutatakse ettevõtte siseselt ja selle abil on võimalik pidada ülevaadet tehtavatest töödest nii projektide kui meeskondade raames. *Taskide* loomine käib väga lihtsalt ja kiirelt. Kirja pandud informatsioon on kergesti kättesaadav ja vajadusel on võimalik muudatusi sisse viia.

Kuigi keskkonnana on Jira väga sobilik, jääb hetkel lahenduse toimimiseks puudu andmete täpsustamine. Kuna Tellijale on ette antud ainult kirjelduse tekstiväli, siis paljudel juhtudel pannakse kirja ainult need andmed, mida Tellija ise vajalikuks peab. Paljudel juhtudel jääb aga saadavast informatsioonist väheks. Vastutav isik, kellele määratakse loodud *Task*, peab uuesti kontakteeruma, et täpsustada vajaminevaid andmeid.
Selleks, et *Taski* loomisel oleks olemas kõik vajaminev informatsioon, oleks mõistlik luua vorm erinevate tekstiväljade, rippmenüü valikute ja märkeruutudena, mis vajaksid täitmist vastavalt küsitud andmete kohta. Sellisel juhul on Tellijal palju lihtsam kirja panna kõik vajalikud andmed, kuna neid küsitakse antud vormis.

Taski kättesaamisel loob vastutav isik käsitsi vastavad *Subtaskid* olemasolevate andmete põhjal, määrab need järgmistele vastutavatele isikutele, kes on PAI administraatorid ning hakkavad määratud ülesandega tegelema.

Selline käsitsi *Subtaskide* loomine on aeganõudev. Selle protsessi kiirendamiseks oleks mõistlik kogutud andmete põhjal luua automaatselt vastavad *Subtaskid* ning määrata need õigele PAI administraatorile. Probleem seisneb selles, et ilma kindla protsessita tekib olukordi kus hilisemas staadiumis tuleb välja, et mõni eeldus on täitmata. Näiteks andmebaas on puudu kui hakatakse rakendust keskkonnas käivitama. See aga pikendab kasutuselevõtu protsessi asjatult.

2.3 Onboardingu nõuete analüüs

Hea oleks teada ka kes on antud projektiga seotud isikud ning mille eest on nad vastutavad. Välja tasuks tuua äripoolel olev isik, kes on vastutav arendusmeeskond, rakenduse haldaja ning kas projekti raames tehakse kellegagi koostööd. [3]

Kui eelnimetatud isikutel on juba ligipääs arendusprojektile olemas oleks seda vaja ka PAI administraatoritel, kellele suunatakse loodavad ülesanded andmete põhjal, mille Tellija esitab. Kõige kiirem viis on selleks anda vormis kaasa link, mis suunab projekti materjalidele. [4]

Tähtis aspekt on ka ajal. Kirja tasub panna ajakava, millal on arendusprojekti tähtajad või kuupäevad mis on seotud keskkonna üles seadmisega. [5]

2.4 Kubernetese keskkonna analüüs

Ettevõtte poolt üks kasutatav keskkond on Kubernetes. See on avatud lähtekoodiga konteinerite haldussüsteem automatiseerimaks kasutuselevõttu, skaleeruvust ning konteineripõhiste rakenduste haldust. See grupeerib rakenduse loogilisteks üksusteks lihtsamaks haldamiseks. [6]

Kubernetese keskkonna üles seadmisel tuleb kausta, ehk klastris töötava protsessi, määramisel täpsustada kui palju iga vajaminevat ressurssi konteiner vajab. Põhilised ressursid mis vajavad täpsustamist on mälu ja protsessori andmed. Andmete täpsustamisel kasutab planeerija seda teavet otsustamaks, millisele sõlmele kaust paigutada. Kui konteinerile määrata mäluline limiit, siis rakendatakse selle piirangud nii, et töötaval konteineril ei lubata kasutada rohkem ressurssi kui talle määratud on. Võimalik on monitoorida kaustade tegevust tehes päringuid sobiva APIga või kasutades monitoorimistööriistu. [7]
Keskkonna nõuete kirja panemisel on sel juhul kõige vajalikumad andmed mälu piirangud, kui ka palju kaustasid läheb keskkonna loomise andmete talletamiseks vaja. Lisadena on võimalik lisada ka monitooringut.

2.5 Virtual Machine keskkonna analüüs

Ettevõte kasutab ka keskkondade loomiseks virtuaalmasinaid. Virtuaalmasin on tarkvaraarvuti, mis pakub sama funktsionaalsust kui füüsiline arvuti. Nad käivitavad rakendusi ja operatsioonisüsteeme. Kuid nad on oma olemuselt hoopis arvutifailid mis töötavad füüsilises arvutis. [8]
Virtuaalmasina üles seadmisel küsitakse esmalt millist operatsioonisüsteemi soovitakse installida. Olenevalt valitud operatsioonisüsteemist on võimalik määrata sellele mäluhulk

mida ta kasutab. Virtuaalmasinale tuleb luua ka kõvaketas ning määrata tema suurus. Need on ka põhilised andmed mis on vajalikud keskkonna loomiseks. [9]

3 Veebirakenduse loomine

Arendatava veebirakenduse eesmärgiks on koguda kokku vajalik informatsioon mida läheb vaja, et tööülesannete jagamine ning teostamine sujuksid kiirelt.

3.1 Veebirakendusele esitatud nõuded

Antud veebirakenduse loomisel lähtub autor ettevõtte poolt kasutuses oleva lahenduse analüüsist, juurde otsitud informatsiooni ning ettevõtte PAI administraatorite soovide järgi kokku pandud nõuetest, mis on vajalikud selleks, et administraatoritel oleks kõik informatsioon kohe kättesaadav.

- Rakendus võimaldab pärida projektiga seotud isikute andmeid
- Rakenduses on võimalik lisada üks või rohkem keskkonna nõuete andmete kogumit
- Rakenduse andmete täitmisel ning edastamisel luuakse vastavad *Taskid* ja *Subtaskid*
- Väljade täitmisel millel on valikuvõimalused tuleb luua rippmenüüdena
- Vorm peab olema inglise keeles
- Üldandmed, mille põhjal luuakse *Task* peavad olema
    - Projekti nimi (Project Name)
    - Projekti lühinimi (Project Short Name)
    - Rakenduse administraator (Application Administrator)
    - Confluence link
    - Äripoole isik (Business Stakeholder)
    - Vastutav arendusmeeskond (Responsible Engineering team)
    - Müüja (Vendor)
    - Partner

    - Pre live kuupäev (Pre live date)
    - Live kuupäev (Live date)
    - Rakenduse juurdepääsu nõuded (Application Acces requirements)
    - Prioriteet (Priority)

- Projektiga seotud kuupäevalised valikud peab saama teha kasutades kalendri kuva
- Kubernetese keskkonna nõuded, mille põhjal luuakse vastav *Subtask* peavad olema
    - Juurutusnimi (Deployment name)
    - Kaustade arv (Number of pods)
    - Reserveeritud mälu (Memory reserved)
    - Mälu limiit (Memory limit)
    - Tüüp (Type)
    - Domeeni nimi (DNS)
    - Bitbucket link
    - Väline võrgu juurdepääs (External network access)
    - RabbitMQ
    - Uus andmebaas (New database)
    - Olemasolev andmebaas (Existing database)
- Virtual Machine keskkonna nõuded, mille põhjal luuakse vastav *Subtask*, peavad olema
    - Protsessori tuumade arv (CPU)
    - Mälu maht (Memory)
    - Ketta suurus (Disk size)
    - Operatsioonisüsteem (OP system)

- Juurutamise juhend (Deployment guide)
- Väline võrgu juurdepääs (External Network access)
- Domeeni nimi (DNS)
- MinIO
- Logi pidamine (Keep a log)
- Monitooring (Monitoring)
- Uus andmebaas (New database)
- Olemasolev andmebaas (Existing database)

3.2 Tehnoloogiate valik

Käesolevas peatükis toob autor välja veebirakenduses kasutatud tehnoloogiad ja teegid.

3.2.1 HTML

HTML on lühend sõnadest ***Hypertext Markup Language***. See võimaldab kasutajatel luua ja struktureerida sektsioone, paragrahv pealkirju, linke, blokk-tsitaate veebilehtede ja aplikatsioonide jaoks.
HTML ei ole programmeerimiskeel, tähendab seda et pole võimalik luua dünaamilisi funktsioone. Kuid, tänu sellel on võimalik organiseerida ja formuleerida dokumente, mis on sarnased Microsoft Word´ile. [10]

3.2.2 CSS

Cascading Style Sheets lühend CSS on stiililehe keel, mis dikteerib sinu veebilehe elemente kuidas need peaksid välja nägema. Sina kontrollid disaini, paigutust, fonti ja sinu veebilehe sisu manustades CSS faili sinu HTML dokumendiga. [11]

3.2.3 Javascript

Javascript on kõige populaarsem kliendi poolne (client - side) skriptimiskeel, , mis on standardiseeritud ECMAScript-i poolt. Skriptid on lisatud HTML dokumentidesse ja

suhtlevad DOMiga. Suurematel veebibrauseritel on sisse ehitatud Javascript mootorid mis käivitavad antud koodi kasutaja seadmes.[12]

3.2.4 Golang

Go on kompileeritud, samaaegselt tegutsev, prügikorjaja (garbage-collector) ehk automaatne mäluhaldus, staatiliselt trükitud keel, mis on arendatud Google'is. Keel loodi 2007 ja esimene versioon anti välja 2012. [13]
Keelel on kaks nimetust tekkinud. Ametlik nimetus on Go, aga kuna seda on raske otsida, hakati laialdaselt viitavama keelele nimetusega Golang tema domeeninime järgi. [14]

Go üritab vähendada sõnarohkust. Kogu disaini vältel üritasid arendajad vähendada segadust ja keerukust. Puuduvad edastamisdeklaratsioonid ja päisefailid; kõik deklareeritakse täpselt ühe korra. Tema eelisteks on paindlikus mis väljendub täpsuses, lihtsuses ja kergesti loetavuses. Kompileerimise aeg on väga kiire, ning võimaldades samaaegselt mitme protsessi tõhusat toimimist. Go keeles on ka paljud tuntud rakendused arendatud nagu Docker, Kubernetes, Netflix, Openshift, Dropbox, CloudFlare. [15]

Go on ka laialdaselt populaarsust koguv programmeerimise keel. TIOBE andmete põhjal on Go keel vägagi populaarne, olles selle aasta mai seisuga 12ndal kohal. [16]

Autor valis Go keele backend arenduse jaoks kuna polnud kunagi veel selle keelega kokku puutunud, kuid pakkus palju huvi. Otsuse langetamisele aitas kaasa ka, et PAI osakonna siseselt kasutatakse arendustöödeks Go keelt.

3.2.5 Jira REST API

REST on olekuta klient-server kommunikatsiooni protokoll. [17]
Käesoleva arendatava rakenduse jaoks kasutatakse Jira REST API-t. Seda kasutatakse, et väliselt suhelda Jira serveri rakendustega. Jira serveri platvorm võimaldab kasutada erinevaid API päringuid laialdaselt Jira keskkonnas kasutatavatele funktsioonidele, näiteks probleemide ja töövoogude jaoks. Jira REST API kasutab JSONi HTTP meetodite – GET, PUT, POST ja DELETE tegemiseks. [18]

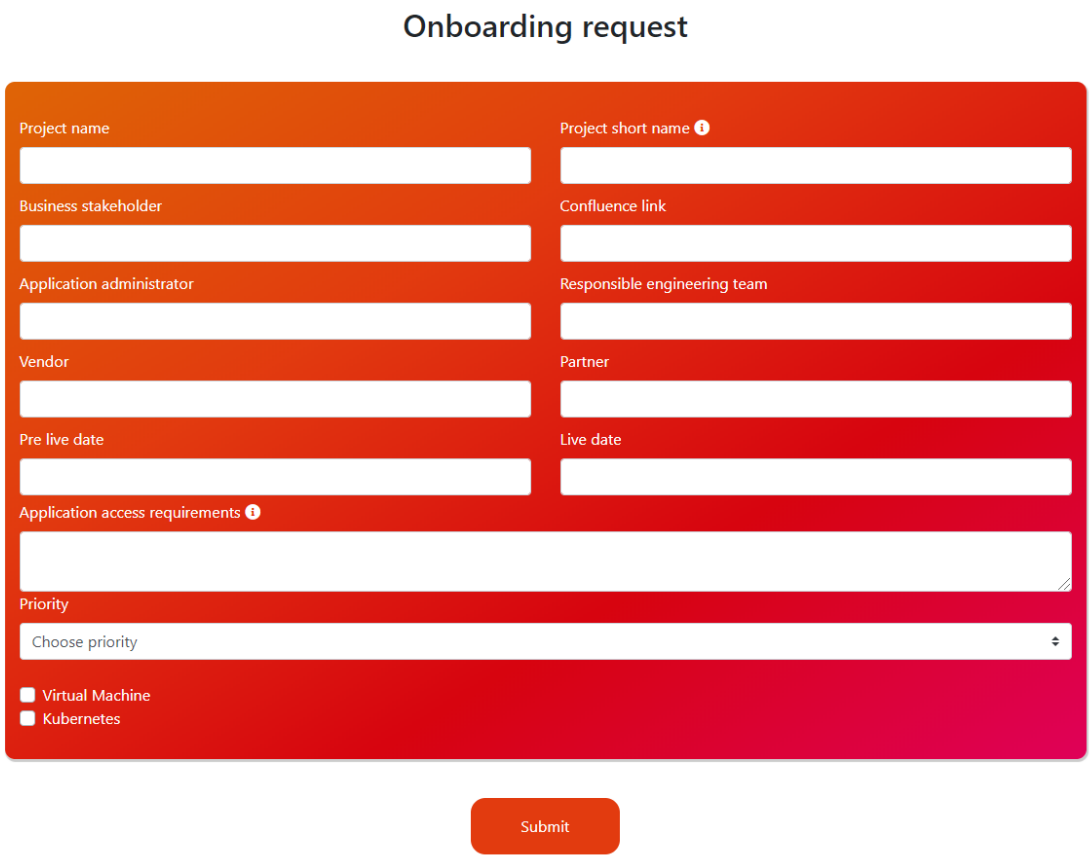
Autor kasutab arendatava rakenduse jaoks GET ja POST päringuid. GET päringut kasutatakse *onboarding* vormi põhiandmete täitmiseks. POST päringut kasutatakse vormi andmete põhjal *Task* ning *Subtaskide* loomiseks.

4 Veebirakendus

4.1 Üldandmed

Veebirakenduse avades kuvatakse Tellijale kogu vorm. Esimene pool vaatest koosneb üldandmetest ja need on kohe Tellijale nähtavad. Täita tuleb andmed projekti nime ning lühinime kohta, mida hiljem kasutatakse Jira keskkonda loodava *Taski* nimena. Tekstiväljad mis on valikuvariante pakkuvad kui alustada kirjutamist ning mis saavad endale sisendiks päringu abil saadud andmed on äripoole isik, rakenduse administraator, vastutav arendusmeeskond, müüja ning partner.

Arendatava projekti materjalidele tuleb anda ka ligipääs Confluence lingi näol. Täpsustada saab kes vajavad rakendusele juurdepääsu, teha kuupäevalisi valikuid *pre-live* ning *live*. Viimasena saab teha valiku prioriteedi osas, kus saab valida nelja valiku vahel – kriitiline, kõrge, keskmine ja madal.



Joonis 1. Üldandmete vaade

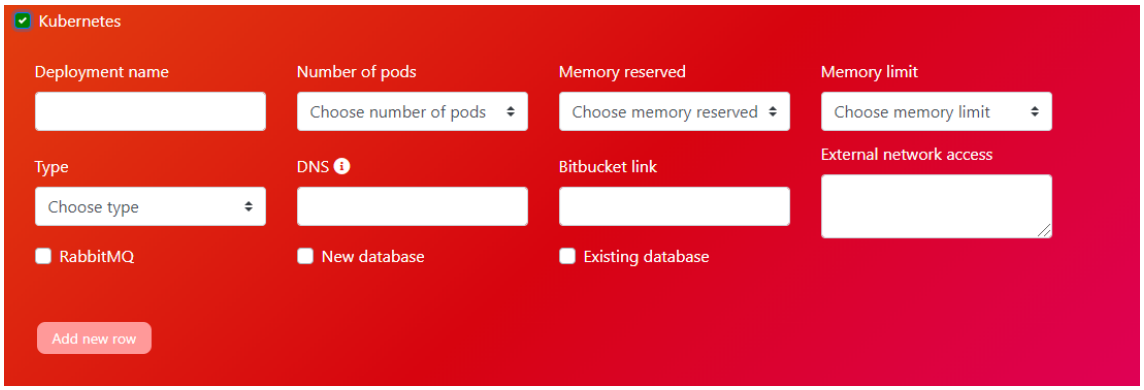
4.2 Kubernetes

Kubernetese andmete vaade ei ole kohe vormi avades nähtav. Tellijal on algselt näha ainult vastav märkeruut, millele vajutades avanevad antud keskkonna nõuded, mida tuleb täita.

Tellijale on ette antud kõik nõuded, mis tuleb täita, et vormi ära saates saaks nende andmete põhjal luua korraliku *Subtaski* kõikide vajalike andmetega, mida läheb PAI administraatoril vaja teada.

Esmalt küsitakse Tellijalt juurutusnime, kaustade arvu, mis määrab mitu protsessi võib klastris töötada. Täpsustada tuleb mälu hulka, mis on reserveeritud ning mälu limiiti ning nende tüüp. PAI administraatorile tuleb anda ka loodava keskkonna jaoks domeeni nimi, Bitbucket link, välise võrgu juurdepääsu saajad. Märkeruutude valikutena saab oma rakendusele valida kas soovitakse kasutada RabbitMQ, mis on avatud lähtekoodiga sõnumite järjekorda seadmise tarkvara või ka lihtsamini öeldes, järjekorrahaldur. [19] Ning lõpetuseks kas rakenduse jaoks on vaja luua uus andmebaas või anda ligipääs olemasolevale andmebaasile. Tellijal on võimalus ka valida mõlemad juhud.

Tellijale on antud võimalus lisada veel uusi Kubernetese keskkondi. Selleks on vaja lihtsalt vajutada nupule „Add new row“. Nupule vajutades lisatakse vaatele uued andmete kogumise väljad, mis kirjeldavad järgmist Kubernetese keskkonda.



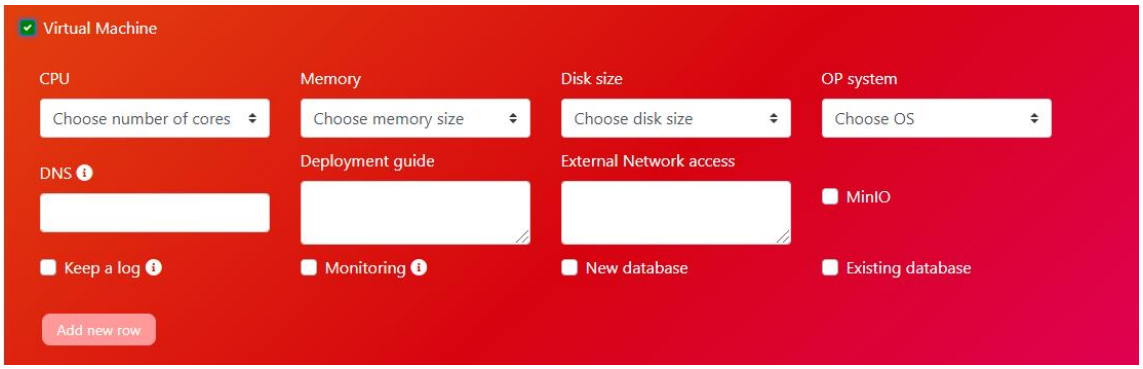
Joonis 2. Kubernetese andmete vaade

4.3 Virtual Machine

Virtual Machine vaade ei ole kohe vormi avades nähtav. Tellijal tuleb avada Virtual Machine vaade samamoodi nagu Kubernetese puhul.

Avanenud vaates küsitakse Tellijalt antud keskkonna nõudeid. Vaja läheb teada protsessori tuumade arvu, mälu hulka, ketta suurust, millist operatsioonisüsteemi läheb vaja rakenduse jaoks, domeeninime, välisele võrgule juurdepääsu saajad ning juurutamise juhendit.

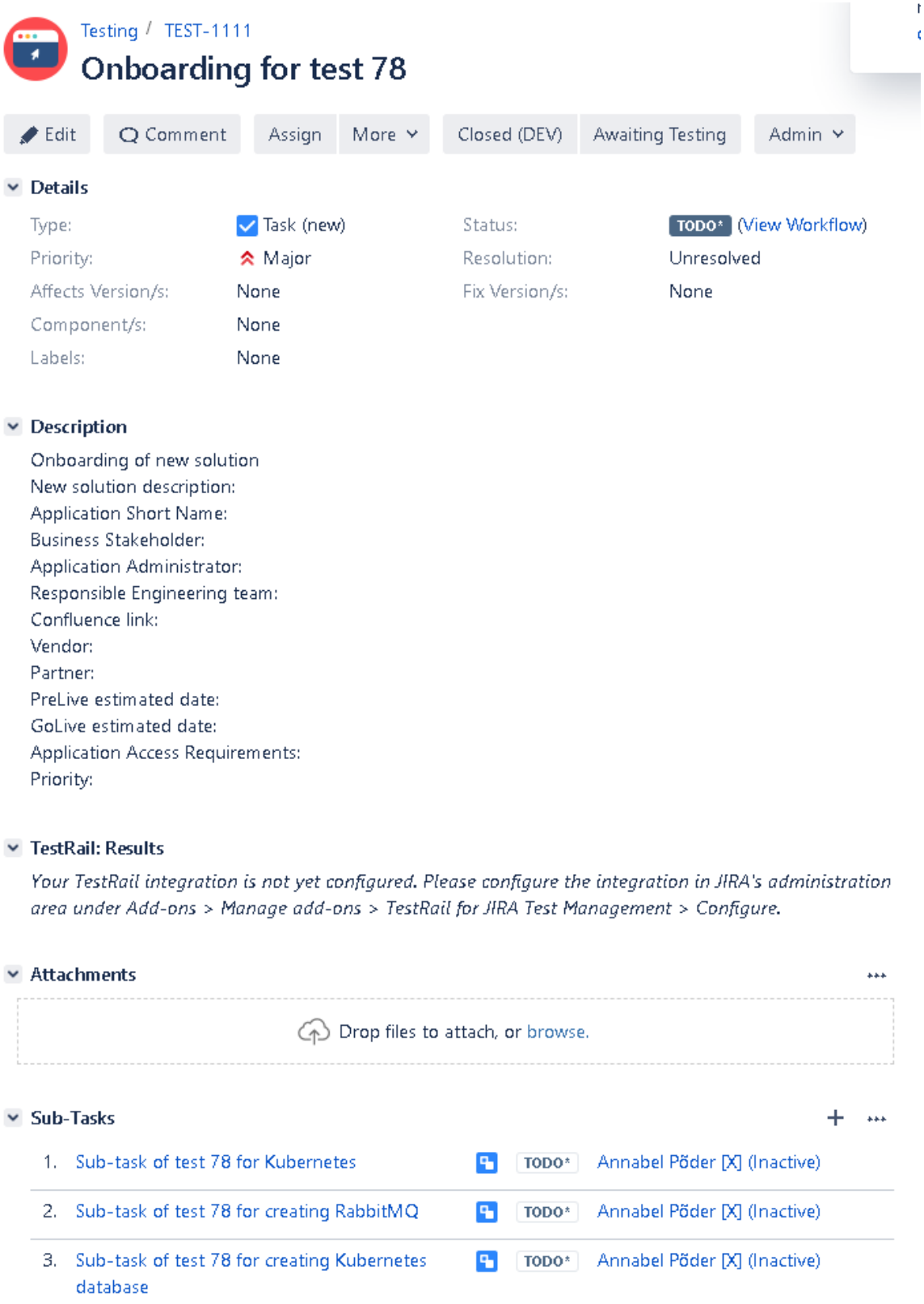
Märkeruutude valikutena saab oma rakendusele valida kas soovitakse monitoorida oma rakendust ning pidada logi et analüüsida andmeid. Võimalik on lisada rakendusele MinIO, mis on suure jõudlusega hajutatud objektide salvestussüsteem. See on tarkvara määratletud ning töötab tööstusstandardilises riistvaras ja on täiel määral avatud lähtekoodiga. [20]
Sarnaselt Kubernetesega on ka antud keskkonnale võimalik luua uus andmebaas või anda ligipääs olemasolevale andmebaasile. Ka Virtual Machine puhul on võimalik vajutades nupule „Add new row“ lisada uute andmete kogumise väljad.



Joonis 3. Virtual Machine andmete vaade

4.4 Loodud Task ja Subtaskid

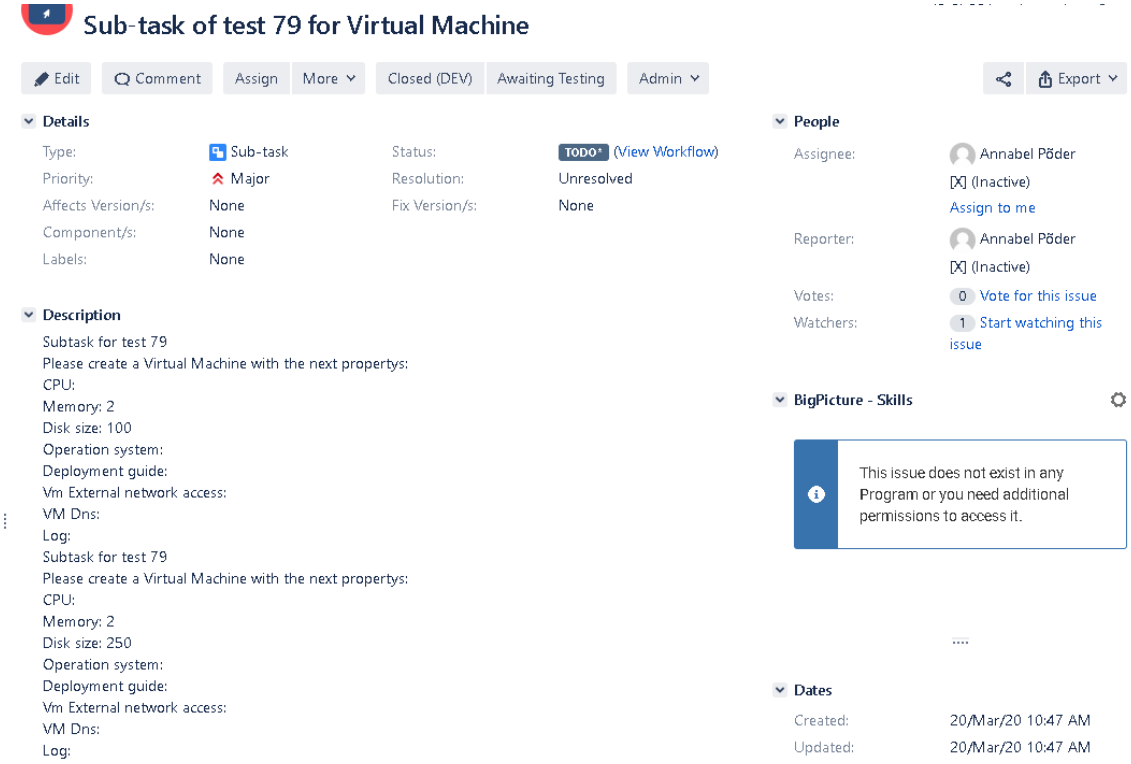
Kui projekti vorm on täidetud ja Tellija vajutab „Submit“ nuppu saadetakse vormi andmed, kasutades API päringut POST, Jira keskkonda. Saadud andmete põhjal luuakse *Task*, kus on kirjas andmed projekti kohta, mis on saadud vormi üldandmete osast. Antud *Taskile* määratakse vastutav isik automaatselt. Nii *Taskile* kui erinevatele *Subtaskidele* on määratud vastutavad.



Joonis 4. Jira keskkonda loodud *Task*

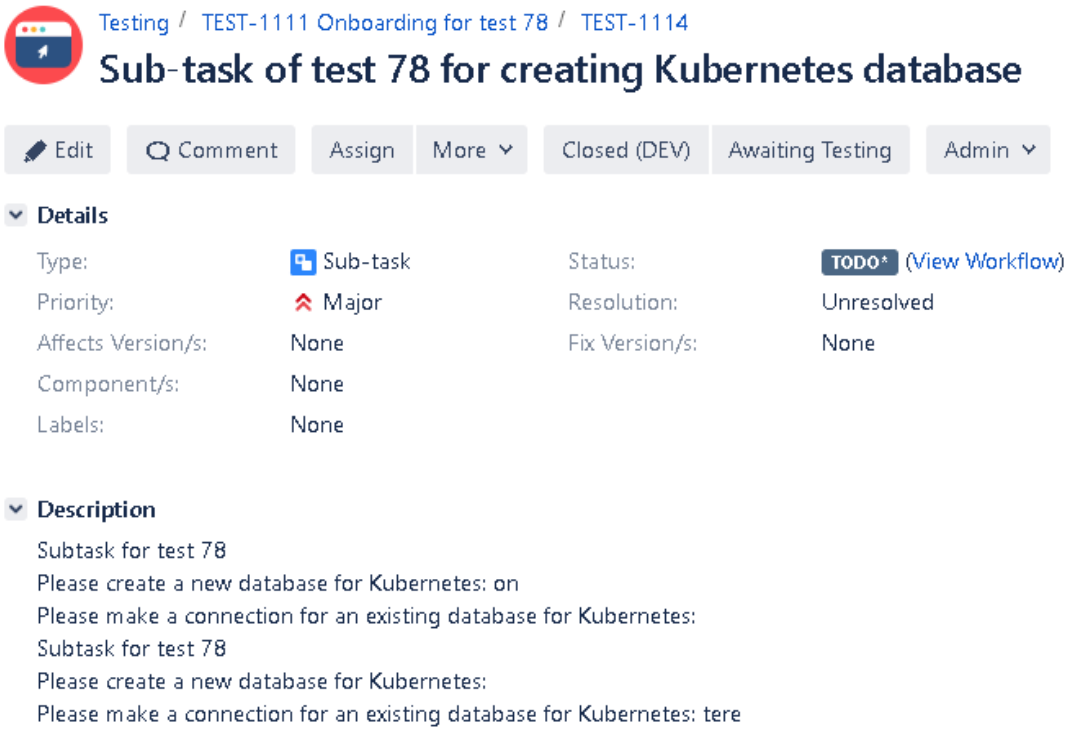
Eraldi *Subtask* luuakse Kubernetesele. Täidetud andmete põhjal luuakse kirjeldus, millised nõuded loodavale keskkonnale on määratud. Juhul kui Tellija on kirja pannud andmed rohkem kui ühe loodava keskkonna kohta, siis luuakse ühe Kubernetese *Subtaski* sisse kahe või rohkema keskkonna nõuete kohta kirjeldus.

Samamoodi luuakse *Subtask* ka Virtual Machine puhul. Andmete põhjal luuakse kirjeldus keskkonna kohta ning kui Tellija on täitnud andmed mitme keskkonna kohta, siis lisatakse kirjeldus mitme keskkonna kohta ühte *Subtaski*.



Joonis 5. Virtual Machine *Subtask*

Kubernetesest ja Virtual Machinest eraldi luuakse *Subtaskid* monitooringu, RabbitMQ ning andmebaaside jaoks. Juhul kui ühe keskkonna jaoks on vaja luua nii uus andmebaas kui ka anda ligipääs olemasoleva, pannakse nende kirjeldused ühte *Subtaski*. Erinevate keskkondade andmebaaside loomine aga tehakse erinevate *Subtaskide* alla, millele antakse keskkonna nimelise tunnusega nimetus.



Joonis 6. *Subtask* Kubernetese andmebaasi loomiseks

5 Hinnang rakendusele ja täiendused

5.1 Hinnang rakendusele

Loodud rakendust esitleti PAI osakonna juhile ning administraatoritele. Tagasisideks anti, et rakendus on lihtsasti kasutatav ning arusaadav. Toodi välja ka, et tekstiväljad mis täites valikuvariante päringute abil pakuvad on vägagi tänuväärt lahendus. Lisaks mainiti ka milliseid täiendusi võiks rakendusel olla.
Üheks edasiarenduseks nimetati, et esmalt võiks Tellijale kuvada valikuvõimalus kas täita uus *onboarding* andmete vorm või ühendada see juba olemasoleva *Taskiga*. Sel juhul tuleks kirja panna ainult *onboardingu* või *Taski* nimi, mis on Jiras olemas ning üldandmete osa sel juhul välja jätta.

5.2 Rakenduse täiendused

Rakenduses on realiseeritud esialgselt planeeritud funktsionaalsus, kuid selleks et teha rakendus veelgi mugavamaks ja kiiremini kasutatavaks toob autor välja täiendused kuidas võiks rakendust edasi arendada:

- Esimese vaatena näidatakse Tellijale kahte valikuvõimalust, milleks on „Create new Onboarding“ või „Existing Onboarding“. Need saab luua märkeruutudena. Kui vajutatakse esimese valiku, ehk uue loomise võimalusele, kuvatakse Tellijale terve vorm, milleks on põhiandmete osa, Kubernetes ja Virtual Machine valik. Teisel juhul, kui valitakse Existing Onboarding, siis näidatakse Tellijale põhiandmete osa asemel ainult ühte tekstivälja. Sinna tuleb kirja panna olemasoleva projekti nimi või Jira poolt loodud *Taski* kood. Selle abil on seotakse keskkonna nõuete *Subtaskid* olemasoleva *Taskiga*.

- Äri osapoole isiku tekstiväljale on võimalik lisada mitu nimelist väärtust.

- Kubernetese ja Virtual Machine jaoks lisatavate andmete kogumi juurde lisada „Delete“ nupp, mis kustutaks lisatud andmete kogumi.

- Kubernetese ja Virtual Machine keskkonna nõuete valikutele lisada juurde hinnaväärtused. Iga valik, mis tehakse on välja mingisuguse väärtusega, mille

valimisel arvutatakse kokku maksumus, mis läheb sellise keskkonna loomiseks. Vormi saatmisel saadetaks automaatselt Tellija e-mailile keskkonna loomise maksumuse summa või ka arve.

Loetletud täiendused muudaksid rakenduse kasutamise veelgi mugavamaks ja kiiremaks.

6 Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli luua veebirakendus, mis kogub arenduses olevatele projektide keskkonna loomiseks vajalikud nõuded, mille põhjal luuakse Jira keskkonda vastavad *Taskid*. Loodava lahenduse abil lihtsustatakse ja kiirendatakse andmete edastamise ning *Taskide* loomise protsessi.

Töö käigus analüüsiti olemasolevat lahendust ja uuriti juurde, mis andmed on vajalikud projektide *onboardimiseks* ning mis nõuded peavad keskkondade loomiseks teada olema. Töös kirjeldati rakenduse jaoks antud nõudmisi, nii ettevõtte poolt antud kui analüüsi käigus saadud tulemustest ning kasutatud tehnoloogiaid.

Autori poolt seatud eesmärk luua töötav veebirakendus sai teostatud. Töös kirjeldati loodud rakendust ja selle funktsionaalsust. Välja toodi ka võimalikud täiendused, mis annaks loodud rakendusele lisaväärtust. Loodud rakenduse lähtekoodi omab ettevõte, kellele antud rakendus loodi.

Kasutatud kirjandus

[1] „What is Jira used for“ [WWW] https://www.atlassian.com/software/jira/guides/use-cases/what-is-jira-used-for (20.04.2020)

[2] „Issue types“ [WWW] https://confluence.atlassian.com/adminjiracloud/issue-types-844500742.html

[3] „ 6 steps to make project onboarding easier“ [WWW] https://www.sayonetech.com/blog/6-steps-to-make-project-onboarding-easier/ (29.01.2020)

[4] „Project onboarding“ [WWW] https://teamdeck.io/project-management/project-onboarding/ (29.01.2020)

[5] „Why Project Onboarding is Important“ [WWW] https://www.function1.com/2015/06/why-project-onboarding-is-important (29.01.2020)

[6] Kubernetes [WWW] https://kubernetes.io/ (15.04.2020)

[7] „Managing resources for containers“ [WWW] https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ (16.04.2020)

[8] Virtual Machine [WWW] https://www.vmware.com/topics/glossary/content/virtual-machine (15.04.2020)

[9] „Beginner Geek: How to create and use Virtual Machines“ [WWW] https://www.howtogeek.com/196060/beginner-geek-how-to-create-and-use-virtual-machines/ (16.04.2020)

[10] „What is HTML” [WWW] https://www.hostinger.com/tutorials/what-is-html (01.05.2020)

[11] „CSS Introduction“ [WWW] https://www.w3schools.com/css/css_intro.asp (01.05.2020)

[12] „What is JavaScript Used For?“ [WWW] https://www.hackreactor.com/blog/what-is-javascript-used-for (01.05.2020)

[13] „Go at Google: Language Design in the Service of Software Engineering“ [WWW] https://talks.golang.org/2012/splash.article (01.05.2020)

[14] „Go (programming language)“ [WWW] https://en.wikipedia.org/wiki/Go_(programming_language) (01.05.2020

[15] „Go Programming Language (Introduction)“ [WWW] https://www.geeksforgeeks.org/go-programming-language-introduction/?fbclid=IwAR1suiC4UBK6Lydp8heZCTW8JUFbKYbVPgbUmNpOA953idZv5P1T5hSRfC8 (01.05.2020)

[16] „ TIOBE Index for May 2020“ [WWW] [https://www.tiobe.com/tiobe-index/go/] (01.05.2020)

[17] REST [WWW] https://et.wikipedia.org/wiki/REST (01.05.2020)

[18] „REST APIs“ [WWW] https://developer.atlassian.com/server/jira/platform/rest-apis/ (01.05.2020)

[19] „Part 1: RabbitMQ for beginners - What is RabbitMQ?“ [WWW] https://www.cloudamqp.com/blog/2015-05-18-part1-rabbitmq-for-beginners-what-is-rabbitmq.html (05.05.2020)

[20] „MinIO Object Storage“ [WWW] https://min.io/product/overview (05.05.2020)