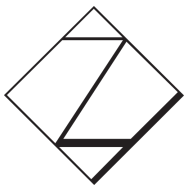

Westsächsische Hochschule Zwickau

Fakultät Physikalische Technik / Informatik

Studiengang Informatik

Bachelorarbeit

Aufbau einer generischen Schnittstelle zur Anbindung und Verwaltung von High-Performance-Clustern

Lukas Balzereit

Matrikelnummer: 37679

Seminargruppe: 162079

1. Gutachter

Prof. Dr.-Ing. Rainer Wasinger

2.Gutachter

Prof. Dr.-Ing. Thomas Franke

16. Oktober 2020

Kurzfassung

Mit der größer werdenden Nachfrage von rechenintensiven Simulationen und der gleichzeitig stagnierenden Entwicklungen im CPU-Markt werden größere Ressourcen für High Performance Computing benötigt. Neben den groß angelegten Serversysteme werden werden auch immer mehr Cloud Ressourcen für HPC angeboten.

Bedingt durch die große Anzahl verschiedener Plattformen und Arten der Interaktion kann es zu Schwierigkeiten in der Anwendung von HPC für Anwender kommen

In dieser Arbeit wird eine Architektur für eine Anwendung entwickelt, die den Zugriff auf HPC Ressourcen für Anwender erleichtert. Anhand dieser Architektur wird ein Produkt entwickelt mit dem die Jobeinreichung und Jobübersicht über eine Weboberfläche und eine REST-Schnittstelle ermöglicht werden.

Diese Bachelorarbeit wurde in Kooperation mit der Firma GNS Systems GmbH in Braunschweig und der Westsächsischen Hochschule Zwickau verfasst.

Danksagung

Für die Unterstützung während meiner Bachelorarbeit möchte ich allen danken, auf die ich mich fachlich und private während der Erstellung verlassen konnte. Das gilt insbesondere, da wir alle den widrigen Umstände durch die Coronavirus Pandemie trotzen mussten.

Mein erster Dank gilt Herrn Prof. Dr.-Ing. Rainer Wasinger für seine Unterstützung und die Betreuung meiner Arbeit.

Des Weiteren bedanke ich mich bei meinen Kollegen der GNS Systems GmbH, die mir jederzeit bei Fragen und mit Anmerkungen zur Seite standen. Insbesondere gilt das für Martin Raunest und Malte Lachnit, die mir durch konstruktive Zusammenarbeit bei der Umsetzung des Projektes geholfen haben.

Abschließend danke ich meiner Familie und meiner Freundin, die mir stets vollen Rückhalt gewährt haben.

Inhaltsverzeichnis

Abkürzungsverzeichnis

HPC High Performance Computing

CFD Computational Fluid Dynamics; numerische Berechnung von Strömungsmechanik

CSM Computational Structural Mechanics; numerische Strukturanalysen mittels finite element method

SaaS Software as a Service

CAE Computer Aided Engineering

SET Software Engineering und Transformation

REST Representational State Transfer

LDAP Lightweight Directory Access Protocol

SSH Secure Shell

TLS Transport Layer Security

UUID Universally Unique Identifier

SPA Single Page Application

ASGI Asynchronous Server Gateway Interface

HATEOAS Hypermedia as the Engine of Application State

JWT JSON Web Token

MVP Minimal Viable Product

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einleitung

1.1 Unternehmensprofil

Das Unternehmen GNS Systems GmbH wurde 1997 als Tochterunternehmen der GNS mbH gegründet. Die circa 80 Mitarbeiter werden an Standorten unter anderem in Braunschweig, Wolfsburg und Sindelfingen beschäftigt. Der Name GNS steht für „Gesellschaft für numerische Simulation“ und deutet auf den Fokus des Unternehmens hin: IT Dienstleistungen im Rahmen von Berechnungen in der Produktentwicklung und Engineering. Infolgedessen stammen die meisten Kunden aus dem produzierenden Sektor, insbesondere aus der Automobil- und Maschinenbaubranche.[GNSa]

1.2 Abteilung SET

Die Abteilung Software Engineering und Transformation (SET) ist für die Softwareentwicklung in der GNS Systems GmbH zuständig. Firmeneigene Softwareprodukte wie lEYEcense zur Überwachung und Verwaltung von Softwarelizenzen oder JGen zur Erstellung von Berechnungsjobs runden das Portfolio des Unternehmens neben dem Anwendersupport ab. Des Weiteren werden verschiedene Projekt im Umfeld des High Performance Computing als Dienstleistung für Kunden neu- und weiterentwickelt.[GNSb]

1.3 Hintergrund und Motivation

Die GNS Systems GmbH hat mit dem Softwareprodukt **JGen** (**J**ob**Gen**erator) die Möglichkeit geschaffen, Berechnungsjobs zu erstellen, ohne das der Anwender dazu eigene Shellscripte erstellen muss. Zudem kann durch diese Anwendung das im Backend eingesetzte Batch-Queueing-System abstrahiert werden, wodurch der Anwender ohne weiteres Fachwissen eigene Jobs erstellen kann. [GNSb]

Im Rahmen einer Neuentwicklung des JGens soll durch Einführung von neuen Schnittstellen neben dem Desktop Client die Zielgruppe um weitere Anwender erweitert werden.

1.4 Zielsetzung

Im Rahmen dieser Arbeit soll eine Minimal Viable Product (MVP) entstehen, mit dem in Zusammenarbeit mit Endanwendern festgestellt werden kann, ob und wie die Anwendung von High Performance Clustern eingängiger gestaltet werden kann. Das Ziel eines MVP ist es, durch Entwicklung eines Produktes mit eingeschränktem Funktionsaufwand mit relativ geringem Aufwand den Markt für ein solches Produkt zu testen und die weitere Entwicklung gezielt an Kundenwünschen zu orientieren.[Moo12]

Als Grundlage soll dazu eine Architektur entwickelt werden die möglichst universell zu einem umfassenden Softwareprodukt weiterentwickelt werden kann. Ziel dieser Anwendung ist das Monitoring sowie die Verwaltung und Verwendung von Server-Clustern an einer einheitlichen, nutzerfreundlichen Schnittstelle im Webbrowser zu ermöglichen. Zu diesem Zweck ist es notwendig, viele verschiedene Anwendungen die momentan zum Einsatz kommen über eine einzelne Schnittstelle verfügbar zu machen. Als Grundlage wird diese Anwendung mit dem Arbeitstitel *hpc-tool* entstehen. Das bedeutet eine für Veränderung offene Architektur mit der Anbindung an den JGen zu entwickeln, um das Potenzial aufzuzeigen, dass in einer einheitlichen Anwendung liegt und wie in Zukunft weitere Anwendungen wie der JGen angebunden werden könnten.

1.5 Aufbau der Arbeit

Im ersten Teil der Arbeit werden die Grundlagen von High Performance Computing (HPC) aufgezeigt, um beim Leser ein erstes Verständnis für HPC aufzubauen und den Kontext dieser Arbeit zu verdeutlichen.

Anschließend werden die Anforderungen erhoben, damit eine nachhaltige Architektur entwickelt werden kann und sich das Produkt an den Vorstellungen der Stakeholder orientiert. Damit das Ergebnis der Arbeit übertragbar ist, wird die Architektur außerhalb des Kontextes HPC entwickelt.

Abschließend wird diese Architektur inklusive der Anbindung des JGens umgesetzt. Dazu soll eine Weboberfläche entstehen, die die rudimentäre Bedienung des JGen erlaubt.

2 High Performance Computing

High Performance Computing (deutsch „hochleistungsrechnen“) bezeichnet die Berechnung auf einem High Performance Computer. Dies ist wie der Begriff „Supercomputer“ ein generischer Überbegriff für besonders schnelle und spezialisierte Computersysteme.[FH11] Heute sind dies häufig keine einzelnen Rechner, sondern eine in einem Verbund („Cluster“) zusammengefasste Vielzahl an Computern die über Hardwareschnittstellen, die speziell zur Hochgeschwindigkeitsübertragung konzipiert sind (z. B. Infiniband[Gru10]) verbunden sind. [DS10]

Anwendung findet HPC häufig in Form von Simulation. Die Simulation von Wetter, Erdbeben oder auch menschlichem Verhalten werden oft von Regierungen oder Wissenschaftlern auf High Performance Computern durchgeführt. Auch in der Privatwirtschaft findet HPC Anwendung. Abbildung 2.2 zeigt, dass die häufigste Anwendung in der Produktentwicklung sowie in der naturwissenschaftlichen Forschung liegt, wobei Computational Fluid Dynamics (CFD) und Computational Structural Mechanics (CSM) genutzt werden.[Sne20]

Die Anwendung von HPC für Simulation findet weltweit einen großen Abnehmer in der Autoindustrie. Mittels CFD kann zum Beispiel Aerodynamik von neuen Modellen berechnet werden ohne diese prototypisch umsetzen zu müssen um sie realen Testszenarien im Windkanal zu unterziehen. Durch CSM können Änderungen an Materialien

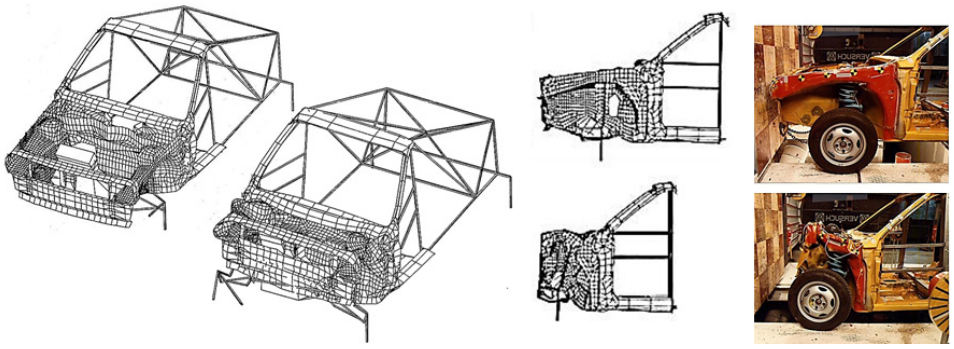

Abbildung 2.1: Simulation und Crashtest eines VW Polo
Quelle: `https://www.esi-group.com/pam-crash`

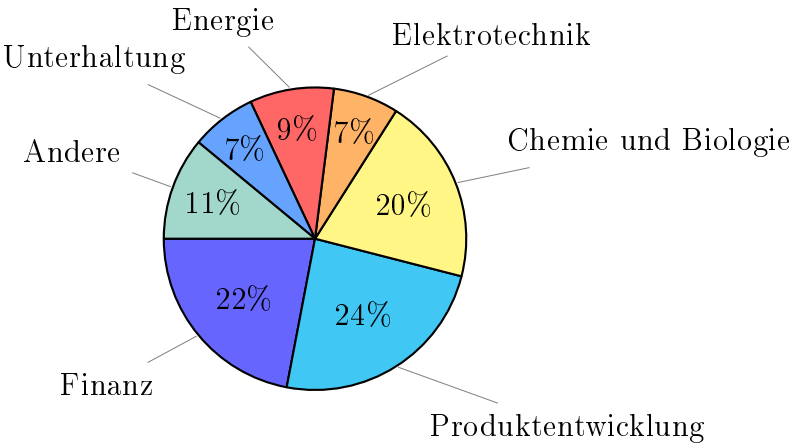

Abbildung 2.2: Marktverteilung HPC nach Industrie [Sne20]

und Konstruktion eines Modells durchgeführt werden werden um sie in Simulationen von Crashtests darauf hin zu untersuchen, ob die Umsetzung in einem Prototyp für einen realen Crashtest zur Verifikation der Ergebnisse lohnenswert ist. Durch die Anpassung der Entwicklungsprozesse als Folge auf die Einführung von Simulationen seit den 1980er Jahre hat dazu geführt, dass neue Entwicklungen günstiger und schneller umgesetzt werden können. Folglich konnte der Produktentwicklungszyklus verkürzt werden, wodurch die die schnellere „time to market“ von neuen Produkten und die Überarbeitung von alten zu einen Wettbewerbsvorteil durch die Verwendung von HPC werden konnten.[Nat05]

Grundbegriffe

Bei einem Auftrag zum Besipiel eine Simulation durchzuführen, spricht man von einem **Job**. High Performance Cluster werden in der Regel von einem **Queuing System** verwaltet, denen die Jobs übergeben werden damit die Rechenressourcen nach bestimmten Kriterien an die Jobs zugewiesen werden können. Häufige Kriterien sind zum Beispiel eine vom Anwender gesetzt Priorität oder die vermutete Länge eines Jobs, wobei insbesondere die Vorhersage der Berechnungsdauer ein eigenes Forschungsfeld ist, in dem künstliche Intelligenz Anwendung findet.

HPC in der Cloud

Der HPC-Markt, der 2019 39 Milliarden Euro umfasste, macht gemeinsam mit dem gesamten Markt von Serverequipment und Softwaredienstleistungen eine Entwicklung in Richtung Cloud Computing, sodass Wachstumsprognosen insbesondere HPC in der Cloud als Sektor der Zukunft hervorheben.[Sne20] Die speziellen Anforderungen von

HPC an die Hardware- und Softwareumgebung erschwert allerdings diese Entwicklung: Zum einen sind die Computer Aided Engineering (CAE)-Anwendungen nicht darauf ausgelegt in Cloud Umgebungen parallelisiert zu laufen, da Software as a Service (SaaS) Deployments durch die häufig verwendete Virtualisierung nicht den Anforderungen an den schnellen Speicherzugriff und Netzwerkverbindungen zwischen den einzelnen Computern eines Clusters entsprechen.[Dow] Zum anderen sind viele Softwarelizenzen für CAE-Anwendungen nicht für die Anwendung in der Cloud, sondern nur für lokale Installationen vorgesehen und verbieten deshalb die Verwendung in in Serverclustern.[LRW18]

HPC und COVID-19

In der 2019 ausgebrochenen Coronavirus Pandemie, die während der Erarbeitung dieser Arbeit eine große Einschränkung des öffentlichen Lebens bewirkt hat[Rob20], spielt auch HPC auf der Suche nach einer Lösung ein Rolle. Das US-amerikanische Energieministerium und der Technologiekonzern IBM haben mit dem Ziel Wissenschaftlern bei der Bekämpfung des Virus zu unterstützen eine Kooperation ins Leben gerufen. Dabei werden den Forschern die Rechenressourcen zur Verfügung gestellt, die sie für ihre Untersuchungen benötigen, ohne auf die wirtschaftlichen Faktoren achten zu müssen, die üblicherweise wissenschaftliche Forschung einschränken.

Die von Mitgliedern des Konsortiums durchgeführten Projekte, befassen sich zum Beispiel mit molekularen Simulationen und künstlicher Intelligenz um den Aufbau des Virus und die Interaktion mit den menschlichen Zellen besser zu verstehen um daraus Erkenntnisse zur Entwicklung eines Impfstoffes oder Behandlungsmethoden zu gewinnen. Neben der Forschung auf der chemischen Ebene fördert das Konsortium auch andere Arten um die Pandemie zu bekämpfen. So kann die Verteilung kritischer Gesundheitsinfrastruktur wie Intensivbetten und Beatmungsgeräten in Simulationen vorhergesehen werden um Engpässe zu identifizieren und die Koordination von Ressourcen zu optimieren um weitere Menschenleben zu retten.[GD20]

3 Anforderungen

Damit die Anforderungen den fachlichen Ansprüchen nach korrekt aufgestellt werden, orientiert sich das *Requirements Engineering* an dem von Balzert[Bal09] und Vogel[VAC+09] vorgeschlagenen Vorgehen. Aufgrund der agilen Natur eines MVP fokussiert sich die Anforderungsanalyse auf die längerfristigen nicht funktionalen Anforderungen, während die funktionalen Anforderungen als weniger formale User Storys festgehalten werden.

3.1 Produktvision

Das hpc-tool soll grundlegend dafür ausgelegt sein in Zukunft zu einem umfassenden Portal für die vollständige Verwaltung ganzer High Performance Cluster zu werden. Das bedeutet das neben der Anbindung von JGen auch weitere Dienste von Drittanbietern oder der direkte Zugriff auf die Cluster ermöglicht werden, wodurch eine große Anzahl von HPC Funktionen in einer einheitlichen Software für den Anwender aufbereitet werden könnten. Beispielsweise wäre die Einbindung des Lizenzmanagements ein Schritt um das Monitoring eines Clustern zu vereinfachen.

3.2 Problem

In Listing 3.1 ist ein einfaches Script für einen Job im Queueing System PBS darge-

```
#!/bin/bash -l
#PBS -l walltime=8:00:00,nodes=1:ppn=8,mem=10gb
#PBS -m abe
#PBS -M sample_email@umn.edu
cd ~/program_directory
module load intel
module load ompi/intel
mpirun -np 8 program_name < inputfile > outputfile
```

Listing 3.1: Job Script für PBS Queing System
Quelle https://www.msi.umn.edu/content/job-submission-and-scheduling-pbs-scripts

stellt. Die verfügbaren Befehle muss der Autor auswendig kennen oder in der Dokumentation nachschlagen. Während das für Anwender, die sehr häufig Jobs einreichen oder im Arbeitsalltag bash Scripte schreiben akzeptabel ist, ist das für andere Anwender weniger der Fall. Das könnte dann ein Hinderungsgrund für die Anwendung von HPC sein.

Das Problem lässt sich in folgender Kurzform zusammenfassen:

Problem	High Performance Cluster sind schwer zu bedienen
Betrifft	CAD/CAE Ingenieure HPC Support
Auswirkungen	Hohe Aufwände im Support, daher höhere Kosten für den Kunden. Weniger Nutzer als möglich.
Erfolgreiche Lösung	HPC ist leichter zu verwenden, daher weniger Aufwand im Support und mehr Nutzer, daher geringere Kosten für den Kunden. Verkaufsargument für die Dienstleistungen von GNS Systems.

Tabelle 3.1: Problembeschreibung hpc-tool

3.3 Rahmenbedingungen

3.3.1 Produktumgebung

Software

Beim hpc-tool handelt es sich um eine Anwendung, die auf einem Server ausgeführt wird und unter anderem eine Oberfläche im Browser ausliefert. Die Anwendung muss auf einem Server mit dem Betriebssystem CentOS 7 installiert und ausgeführt werden können, wobei keine vorinstallierten Softwarepakete angenommen werden sollten. Für die Installation kann der Zugriff auf Administratorrechte angenommen werden, zur Ausführung der Anwendung sollten das nicht der Fall sein.

Die Weboberfläche muss mindestens die Browser Edge, Firefox, Chrome und Safari in aktuellen Versionen unterstützen. Genauso wie aktuellen Produkten von Microsoft wird eine Unterstützung von Internet Explorer nicht garantiert.[1]

Die Anwendung benötigt Zugriff auf einen im Netzwerk verfügbaren LDAP-fähigen Authentifizierungsdienst. Zur Entwicklung wird ein Windows AD verwendet. Des Weiteren besteht die Abhängigkeit zu einer Instanz des in Abschnitt 1.3 vorgestellten JGen

[1] `https://techcommunity.microsoft.com/t5/microsoft-365-blog/microsoft-365-apps-say-farewell-to-internet-explorer-11-and/ba-p/1591666`

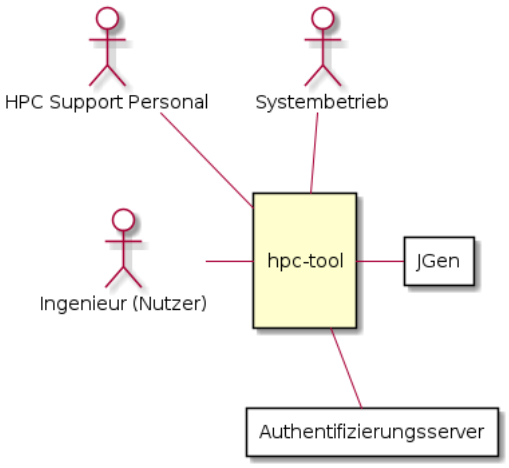

Abbildung 3.1: Kontext des hpc-tools

der ebenfalls im Netzwerk erreichbar sein muss. Auf den Server, auf dem der JGen installiert ist, muss der Endanwender mittels konfigurierter public key Authentifizierung via Secure Shell (SSH)[2] passwortlos zugreifen können, dazu ist einmalig das Passwort des Endanwenders oder die Konfiguration durch einen Administrator notwendig.

Hardware

An die Hardware des Servers werden keine Anforderungen gestellt. Endanwender der Webanwendung können diese auf einem mobilen Gerät mit Touch Interface (Tablet, circa 10"; Smartphone, circa 5") oder auf klassischen Computern (Laptop, circa 14"; Desktop, größer 14") nutzen.

[2] `https://www.ssh.com/ssh/public-key-authentication`

Interessenvertreter

Interessenvertreter	Rolle	Beschreibung
Product Owner	Vertritt Produktvision	Priorisiert die Entwicklungsaufgaben und stellt sicher das die Anwendung den Anforderungen entspricht
HPC Support	Vertritt Interessen des HPC Supports	Unterstützt den Systembetrieb und behandelt eskalierte Fehler
HPC Support[3]	Vertritt Interessen des Systembetriebs	Verwalten die Ausführung des hpc-tools (Installation, Start, Stop, Fehlerbehandlung: Aufnahme, ggf. Behandlung und Eskalation)
HPC Support[3]	Vertritt Interessen von Nutzern	Verwenden das hpc-tool zur Interaktion mit HPC-Cluster
Abteilungsleiter	Vertritt Interessen des Unternehmens	Rechtfertigung von Entwicklungskosten

Tabelle 3.2: Interessenvertreter des hpc-tools

3.3.2 Geschäftschancen

Damit neue Anwender schneller und einfacher mit der Verwendung von High Performance Computing starten können soll die Interaktion mit HPC mittels des hpc-tools vereinfacht werden, indem Jobs über ein Webfrontend abgegeben und verfolgt werden können. Dadurch werden Arbeitsabläufe für Nutzer vereinfacht und auch weniger versierte Nutzer werden in die Lage versetzt HPC für Simulationen zu nutzen. Somit können neue Zielgruppen erschlossen werden und die Verwendung von HPC im bestehenden Kundenstamm durch eine verbesserte Usability angehoben werden. Durch ein Portal wird die Nachfrage nach Softwareprodukten der GNS Systems GmbH wie dem JGen angehoben.

[3] Die Mitarbeiter aus dem HPC Support dienen als Ansprechpartner für Kundeninteressen und Interessen des Systembetrieb da die Supporter die meiste Erfahrung im Umgang mit den Nutzern und Systemadministratoren haben und daher deren Interessen kennen. Reale Kunden beziehungsweise Nutzer sind in dieser Projektphase nicht beteiligt.

3.4 Nicht-Funktionale Anforderungen

Durch Befragung der Interessenvertreter ergab sich, dass die Anwendung folgende Qualitätsmerkmale nach ISO 9126 implementieren soll:

Systemqualität	sehr gut	gut	normal	nicht relevant
Funktionalität			x	
Zuverlässigkeit		x		
Benutzbarkeit			x	
Effizienz				x
Wartbarkeit	x			
Portabilität	x			

Tabelle 3.3: Übersicht über angestrebte Qualitätsziele

3.4.1 Funktionalität

Durch die Bereitstellung einer Representational State Transfer (REST)-Schnittstelle soll eine **Interoperabilität** mit anderen Systemen gewährleistet werden, da Anwender durch eine Vielzahl von Anwendungen und Programmier- bzw Scriptsprachen mit diesen Kommunizieren können.

Die **Sicherheit** ist durch die Verfügbarkeit von Transport Layer Security (TLS) Verschlüsselung für Verbindungen aus dem Internet sicherzustellen. Des weiteren sollten Nutzer authentifiziert werden um die unsachgemäße Nutzung von HPC-Ressourcen zu vermeiden.

3.4.2 Zuverlässigkeit

Fehlertoleranz ist ein wichtiger Aspekt bei der Verwendung jeder Software. Damit eine möglichst Durchgängige Verfügbarkeit gewährleistet wird, muss die Anwendung entweder Fehler tolerieren beziehungsweise transparent an den Nutzer weitergeben oder bei kritische Fehlern nach dem *fail fast*(schneller Abbruch) Verfahren die Ausführung direkt abbrechen damit Infrastrukturdienste einen Neustart durchführen können, damit ein längeres Verweilen in einem unbestimmten aber nicht betriebsfähigen Zustand zu verbringen.[Wig11]

Durch eine möglichst weitreichende Zustandslosigkeit kann nach dem selben Prinzip die **Wiederherstellbarkeit** durch einfaches neustarten der Anwendung sichergestellt werden.

Diese Aspekte sind nicht alle in dieser Entwicklungsphase mit eingeschränkten Testkapazitäten zu erfüllen und die Vollständige **Reife** kann mit Hilfe von agiler Entwicklung mit einem automatisierten *Deployment*-Prozess (CI/CD-Pipeline) im Betrieb erreicht werden.

3.4.3 Benutzbarkeit

REST Schnittstelle

Durch Implementierung des RESTful Ansatzes und Einhaltung der üblichen Standards und best practices soll die Benutzbarkeit der REST-Schnittstelle sichergestellt werden. Die verwendeten Parameter sollen für Anwender, die kein tieferes Verständnis für HPC Anwendungen haben leicht ohne tiefgreifende Einarbeitung in die Dokumentation verständlich sein.

Webanwendung

Die Bedienbarkeit der Webanwendung steht zu diesem Zeitpunkt nicht im Fokus, da ohne Usability Tests mit Anwendern(die aus Zeitgründen nicht in diesem Projekt durchgeführt werden können) wenige aussagekräftige Feststellung über Attribute wie **Erlernbarkeit**, **Bedienbarkeit** und **Attraktivität** getroffen werden können. Das stünde im Widerspruch zur nutzerzentrierten und agilen Entwicklung die in diesem Projekt angestrebt wird. Die Webanwendung ist daher nur ein Vorschlag auf deren Grundlage in Zusammenarbeit mit Nutzern und Kunden weitere Entwicklungen ansetzen kann.

3.4.4 Effizienz

Die Anwendung sollte dem Anwender und insbesondere bei Verwendung durch mehrere parallele Nutzer ein angemessenes **Zeitverhalten** besitzen. Das bedeutet, dass die Anwendung in einem für den Nutzer verträglichem Tempo arbeitet um Ausgaben in der Weboberfläche nach einer erträglichen Zeit ausgibt. Diese Wartezeiten sollte sich auch bei der Verwendung der Anwendung durch mehre Nutzer nicht drastische verschlechtern. Für diese Kategorie werden bewusst keine klaren Ziele definiert, da diese nicht im Rahmen dieses Projektes ausreichend verifiziert werden können, da die Anwendung nicht mit realen Nutzern in einem realen Szenario, also Produktivsystem, getestet werden kann und Last- beziehungsweise Performancetests nicht im Rahmen dieser Implementierung stattfinden. Aufgrund der geringen Limitierung von Rechenkapazität im Einsatzgebiet

auf größeren Servern oder in der Cloud, erhält das erhält das **Verbrauchsverhalten** eine niedrige Priorität.

3.4.5 Wartbarkeit

Für die **Analysierbarkeit** soll durch die Ausgabe von Meldungen für Anwender und Entwickler(Logging) transparent gemacht werden, welche Probleme bei der Ausführung der Software aufgetreten sind. Beim erfüllen dieser Anforderung ist insbesondere Verwendung in der Cloud zu beachten, wo oftmals *postmortem*, also nach dem Ausfall einer Anwendung anhand von festgehaltenen Ausgaben analysiert werden muss, worin der Ausfall begründet war.[Atl20]

Die Anwendung sollte durch entsprechendem Architektur und Design eine möglichst gute **Änderbarkeit** besitzen, das heißt, es soll durch Modularität und das Einhalten von Softwareentwurfsmustern und -prinzipien verständliche und kleine Funktionseinheiten geschaffen werden.[Gol19]

Des weiteren sollte die Anwendung in einer Weise umgesetzt werden, dass eine gute **Testbarkeit** gegeben ist. Wenn Softwaretests in einem ausreichenden Maße umgesetzt sind folgt aus der Testbarkeit auch eine gute **Stabilität**, weil unerwartetes Verhalten von Änderungen am Programmcode durch Tests offengelegt wird und die technische Schulden möglichst gering gehalten werden, wodurch die Anwendung länger unterstützt und weiterentwickelt werden kann und zukünftige Änderungen einem kleineren Aufwand bedeuten.

3.4.6 Portabilität

Eins der wichtigen Qualitätsmerkmale für die Anwendung ist die Portabilität: Sie sollte in der Lage sein sowohl auf einem On-Premise System ausgeführt zu werden oder in einer Cloud-Umgebung. Um Vendor Lock-In zu verhindern, können keine für einen Anbieter spezifischen Dienstleistungen in Anspruch genommen werden. Um Anbieterunabhängigkeit sicherzustellen reicht es die Lauffähigkeit in einem Container sicherzustellen.

Um den einfachen Betrieb durch Systemadministratoren zu erlauben muss eine gute **Installierbarkeit** gewährleistet sein. Gleichzeitig muss auch durch ausreichende Kapselung **Koexistenz** mit bereits installierten Softwareanwendungen möglich sein.

3.5 Zielgruppen

Um die Visionen der Zielgruppen zu verdeutlichen werden Personas erstellt. Dadurch werden die erstellten Anforderungen für reale Profile modelliert und die Kommunikation der Interessenvertreter vereinfacht und vereinheitlicht.[Coo08]

Automatisierer

Annika arbeitet seit 10 Jahren CAD-Ingenieurin und verwendet mehrfach am Tag HPC Systemen, um Simulationen von ihr erstellten Modellen berechnen zu lassen. Sie verwendet lieber textbasierte als grafische Benutzeroberflächen und versucht alle wiederkehrende Aufgaben so gut wie möglich zu automatisieren. Dazu verwendete sie die Skriptsprachen *bash* sowie *python*. Da sie viel beschäftigt ist, will sie sich nicht immer wenn sie eine neue HPC-Applikation verwendet mit den extrem detaillierten Kommandozeilenparameter auseinandersetzen, sondern lieber eine für alle Anwendungen einheitliche Schnittstelle verwenden.

Täglicher Nutzer

Theo berechnet jeden Tag zum Feierabend seine CAE-Modelle. Da er schon länger mit HPC arbeitet kennt er sich ein wenig mit den Parametern aus und möchte diese anpassen können um sicherzustellen das die Ergebnisse am nächsten Morgen korrekt berechnet wurden. Auf dem Heimweg will Theo noch kurz auf seinem Diensthandy checken ob sein Job schon angelaufen ist.

Oberflächlicher Nutzer

Olaf ist Produktdesigner und hat erst vor kurzen von HPC erfahren und ist widerwillig sich mit den Systemen tiefgehend auseinanderzusetzen. Deswegen will er so wenig Zeit und Interaktion wie möglich mit dem Hochladen seiner Jobs verbringen wie er kann. Da Olaf nur unregelmäßig Jobs abgibt ist es jedes Mal als würde er die Anwendung zum ersten Mal nutzen und Arbeitsabläufe kann er sich nicht merken. Olaf nutzt am meisten Apple Produkte, da er auf Design und Aussehen großen Wert legt.

3.6 Funktionale Anforderungen

Die Funktionalen Anforderungen wurden während der Projektdurchführung in agiler Vorgehensweise, also Sprint Plannings und Refinements in Zusammenarbeit von Product Owner und Entwickler mit Unterstützung des Mentors formuliert.

3.6.1 Job-Submit

Nutzer sollen in der Lage sein Jobs abzugeben.

- Als Automatisierer will ich die möglichen Parameter für eine Job Abgabe von der REST-Schnittstelle abfragen können, um diese für eine Job-Abgabe verwenden zu können
- Als Nutzer der Webanwendung will ich die verfügbaren Parameter im Browser auswählen können.
  - Als Nutzer der Webanwendung will ich auf dem Fileserver browsen können, um benötigte Dateien interaktiv auszuwählen und Pfade nicht auswendig kennen zu müssen.
  - Als Oberflächlicher Nutzer der Webanwendung will ich mit möglichst wenigen Schritten einen Job einreichen, alle Parameter bei denen das möglich ist sollten voreingestellte Werte haben damit ich mich nicht mit den Anwendungen und Servern „unter der Haube“ auseinandersetzen muss.

3.6.2 Job-Liste

Nutzer sollen sich die Job die sich im Queueing System befinden mit zusätzlichen Informationen auflisten lassen können.

- Als Oberflächlicher Nutzer will ich in der Lage sein die Jobs im Queueing System auflisten zu lassen, damit ich feststellen kann, ob mein Job läuft oder schon abgeschlossen ist.
- Als Täglicher Nutzer will ich zusätzliche Informationen wie Output, Jobfile und Directory auf dem Queueing Server zu Jobs einsehen können, damit ich das Verhalten des Jobs nachvollziehen kann und Fehler gegebenenfalls selber beheben kann.
- Als Nutzer der Webanwendung will ich einen Job einfach Abbrechen können.
- Als Automatisierer will ich einen Job abbrechen können.

3.6.3 Authentifizierung

- Als Nutzer der Webanwendung will ich mich mittels meiner Unternehmensweiten Anmeldeinformationen authentifizieren können, damit ich mir nicht noch weitere Passwörter merken beziehungsweise in meinem Passwortmanager speichern muss.
- Als Automatisierer will ich mich mit meinen unternehmensweiten Anmeldeinformationen authentifizieren können ohne eine eien grafische Oberfläche nutzen zu müssen, damit ich auch diesen Schritt automatisieren kann.

4 Architekturentwurf

4.1 Generalisierung des Problems

Anhand der nicht funktionalen Anforderungen der Anwendung die in Abschnitt 3.4 aufgestellt wurden, können die Anforderungen an die Architektur verallgemeinert werden um eine nachhaltige Erkenntnis aus diesem Projekt zu gewinnen.
Eine fundamentale Anforderung ist **Skalierbarkeit**, also die Leistungssteigerung einzelner Komponenten und der gesamten Architektur durch Vervielfältigung der Ressourcen.[FH11] Des Weiteren ist die **Änderbarkeit** einzelner Module und die leichte **Erweiterbarkeit** durch hinzufügen neuer Module von integraler Wichtigkeit.

Insgesamt lassen sich die funktionalen Anforderungen auf 3 Kernanforderungen herunterbrechen:

- Darstellung im Browser
- Bereitstellung einer Schnittstelle für Nutzer und Browser
- Kommunikation mit Externen Anwendungen im Backend

4.2 Architekturansätze

4.2.1 Monolith

Falls alle Anforderungen an die Architektur in einem einzelnen Systembaustein umgesetzt wird, ist von einem Monolithen die Rede.[VAC+09] Dieser Monolith kann auch in der Lage sein dynamische Webinhalte auszuliefern, ein weit verbreiteter Ansatz für Java-Anwendungen sind JavaServer Pages (JSP)[1]. Da das hpc-tool auch ohne Webfrontend, also nur mit einer REST Schnittstelle auslieferbar sein soll, kommt nur ein Monolith in Frage, der eben diese Schnittstelle bereitstellt mit einer Webanwendung die getrennt funktioniert und ausgeliefert wird.

[1] `https://www.oracle.com/java/technologies/jspt.html`

Ein Vorteil einer monolithischen Software ist die einfache manuellen Verwaltung, da nur eine Anwendung gestartet, verwaltet und aktualisiert werden muss. Liegt allerdings ein System zum automatisierten Deployment („CI/CD-Pipeline“) vor, ist der Aufwand eine Änderung durchzuführen höher als wenn aller Funktionalitäten in kleinere Anwendungen ausgelagert wären.

Ein Nachteil einer monolithischen Software liegt in der begrenzten Skalierbarkeit. Obwohl eine monolithische Anwendung vertikal skalierbar ist, das bedeutet, dass die Leistung durch Erweiterung der Hardware um Speicher oder Rechenleistung zu einer verbesserten Leistung der Anwendung führt, mangelt es an horizontaler Skalierbarkeit. Wenn eine Anwendung horizontal Skalierbar ist, kann sie auf mehrere Geräte verteilt werden, wobei die Replikation einzelner Dienste zu einer verbesserten Gesamtleistung des Systems führt.[TVS17]

4.2.2 Serviceorientiert/Microservices

Obwohl Martin Fowler [Fow15] zur Vorsicht aufruft bevor ein Projekt mit einer verteilten Architektur begonnen wird und die Frage aufwirft, ob es besser sei, mit einer monolithischen Architektur zu starten und diese anschließend in eine verteile Architektur herunter zu brechen, decken sich die oben genannten Kernanforderungen deutlich mit Designzielen die van Steen und Tanenbaum in ihrem Buch "Distributed Systemsfür verteilte Systeme identifizieren.[TVS17] Daher ist die bessere Grundlage für künftige Anpassungen, von Beginn an eine verteilte Architektur zu entwickeln, die es ermöglicht alle beteiligte Komponenten schnell auszuwechseln und neue hinzuzufügen.

Diese Architekturen werden auch als Serviceorientierte Architektur bezeichnet, da alle Komponenten lose gekoppelt sind und über eine auf Nachrichten aufbauende(„message oriented“) Schnittstelle kommunizieren.[FH11]

Die Kommunikation soll dazu dienen Anfragen asynchron Anfragen an Services im Backend weiterzuleiten und die Antworten unabhängig wieder an den Anwender zurückzugeben. Antworten können sowohl mehrere Nachrichten in einem Strom als auch einzelne Nachrichten sein.

Da die Schnittstelle des hpc-tools für Anwender auf einem synchronen Protokoll(REST) aufbauen soll, muss der Wechsel von asynchroner zu synchroner Kommunikation abgebildet werden. Dazu müssen Clients wiederholt auf neue Nachrichten überprüfen(eng. „polling“), während der Zeiten zwischen den Anfragen empfangene Nachrichten, werden in einer Message Queue zwischengespeichert.

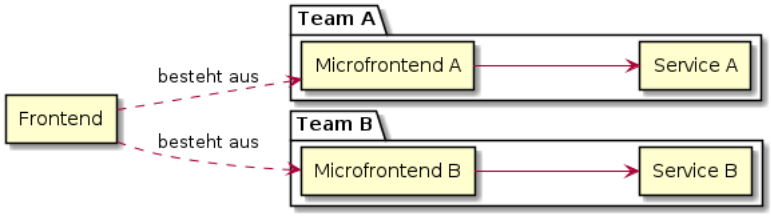

Abbildung 4.1: Aufbau von Micro Frontends

Micro Frontends

Ein Ansatz um modulare Architekturen mit einem serviceorientierten Architekturgedanken mit der Entwicklung eines Frontends verbinden wurde 2016 im ThoughtWorks Technology Radar[2] vorgestellt: Micro Frontends.

In ihrer Essenz verteilen Micro Frontends die Entwicklung von kleinen Teilen einer Webanwendung an die Teams, die auch für die Entwicklung des entsprechenden Dienstes welcher die Logik serverseitig bereitstellt. Diese von getrennten Teams entwickelten Teilanwendungen werden erst zur Auslieferung an den Nutzer zu einer zusammenhängenden Oberfläche zusammengeführt.[Gee20]
Diese Technik bietet einige Vorteile: Durch die Trennung von den unterschiedlichen Diensten und Codebasen erhalten diese untereinander eine losere Kopplung und Modularität sowie eine höhere Kohäsion in sich, was wiederum Prinzipien sind, die in einer Softwarearchitektur anzustreben sind damit eine eine nachhaltige und den Anforderungen entsprechende Software entstehen kann.[VAC+09]

Allgemein besitzen Micro Frontends zwar einen ähnlichen Anwendungsfall wie er in diesem Projekt existiert, in einigen wichtigen Punkten kommt es allerdings zu einer gewissen Diskrepanz. In diesem Anwendungsfall soll ebenfalls eine Anwendung entstehen, in der voneinander unabhängige Dienste zu einem gemeinsamen Nutzererlebnis führen. Dem gegenüber ist aber nicht nur wie bei Micro Frontends vorgesehen eine zusammengeführte Webanwendung sonder auch eine zentrale REST-API gefordert. Des weiteren ist ein zentraler Punkt von Micro Frontends die Skalierbarkeit der Entwicklung in mehrere Fullstack Teams, die in diesem Projekt nicht gegeben ist und auch in Zukunft nicht absehbar ist. Daher würde der Einsatz von Micro Frontends insbesondere einen höheren Entwicklungsaufwand für die Zusammenführung der Komponenten und mehr organisatorischen Aufwand in Form von mehreren Repositories und Build-Pipelines führen. Folglich sind zwar die Gedanken die mit dem Micro Frontend Muster dokumentiert werden

[2] `https://www.thoughtworks.com/radar/techniques/micro-frontends`

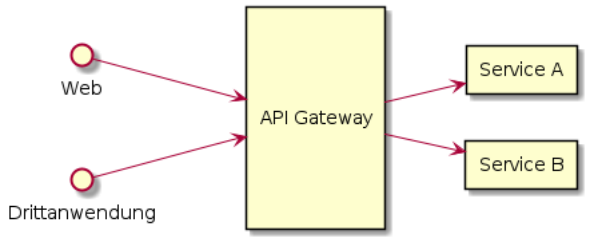

Abbildung 4.2: Aufbau eines API Gateways

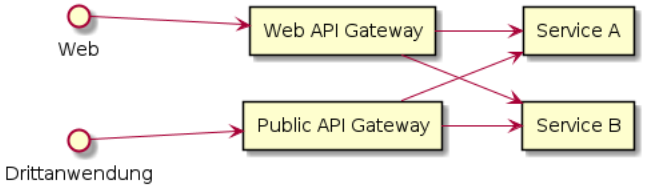

Abbildung 4.3: Aufbau von „Backends for Frontends“

interessant, der Umsetzung steht hier aber das in diesem Fall ungünstige Kosten/Nutzen Verhältnis im Weg.

API Gateway

Um den Gedanken einer serviceorientierten Architektur mit der Anforderung einer einheitlichen API zu Verbinden ist ist eine weitere Komponente notwendig. Diese Aufgabe wird durch ein „API Gateway“ erfüllt. Abbildung 4.2 veranschaulicht wie ein Gateway es allen Clients ermöglicht, über eine einheitliche Schnittstelle Dienste im dahinterliegenden Backend aufzurufen. Die Gateway-Komponente verfüllt demnach mehre Aufgaben: Sicherheit und Authentifizierung von Anfragen sicherzustellen, Kommunikationsprotokolle von Client und Service zu übersetzen sowie die Kommunikation an die mittels Load-Balancing, Message Queueing oder Service Discovery verwalteten Service-Instanzen weiterzuleiten.[MW16] Ein Problem eines einzelnen API-Gateways liegt darin, das es an keinen Client angepasst ist und daher nicht die optimale Kommunikation gewählt werden kann.[Jac12]

Backends für Frontends

Eine Spezialisierung des „API Gateway“ Musters findet sich in der Anwendung von „Backends for Frontends“. Wie die Abbildung 4.3 zeigt, besitzt jede Anwendung eine dedizierte Schnittstelle, beziehungsweise es wird eine öffentliche Schnittstelle für Drittanwendungen angeboten. Diese Schnittstelle können auch in einer großen Komponente

vereint sein, wie es zum Beispiel bei der erneuerten API von Netflix, einem großen Anwendungsfall einer Microservices Architektur, der Fall ist.[Jac12] Durch Einsatz dieses Musters können die verschiedenen APIs genau an den jeweiligen Client angepasst werden, wodurch

4.2.3 Kommunikation zwischen Komponenten

Damit Clients mit und verschiedene Komponenten untereinander Kommunizieren können, müssen die Netzwerkadressen der angesprochenen Services bekannt sein. Da Komponenten durch Ausfall nicht zwingend an der gleichen Adresse erreichbar, oder durch Replikation mehrere Adressen gültig sein können, sollte dem Aufrufer ein betriebsfähiger Kommunikationspartner vermittelt werden.[MW16]

Service Discovery

Eine Möglichkeit um dem Aufrufer die Adresse eines verfügbaren Dienstes zukommen zu lassen ist die Methode des „Service Discovery“, dabei registrieren sich Dienste bei einem Discovery Service. Dieser gibt dann auf Anfrage durch einen Client die registrierte Adresse heraus, worauf hin der Client seine Anfrage direkt an den entsprechenden Dienst richten kann. Gegebenenfalls besitzt der Discovery Service eine Funktion, die Anfragen gleichmäßig auf replizierte Instanzen des Dienstes verteilt(„Load Balancing“).

Der entscheidende Nachteil dieser Technik ist, das Clients die Logik implementieren müssen, wodurch die Adresse nachgefragt wird. Das führt zu Dopplung von Code, was generell zu vermeiden ist um Wartbarkeit des Codes zu gewährleisten. Um dem entgegen zu wirken, gibt es auch Methoden zur serverseitigen Service Discovery. Typischer Weise wird Service Discovery zur Kommunikation mittels RESTful, daher Request-Reply basierten Kommunikationsarten verwendet, um die Anzahl lange bestehende Verbindungen, die insbesondere, die insbesondere bei asynchroner Kommunikation auftreten, zu reduzieren. [MW16]

Message Queue

Eine weitere Möglichkeit der Vermittlung von Aufrufen zu Microservices, ist die Nutzung einer Message Queue. Anfragen werden dabei von dem Aufrufer in einer Warteschlange gespeichert, die vom angesprochenen Dienst abgefragt wird. Folglich übernimmt der angesprochene Dienst selber die Aufgabe des Load Balancing indem nur Anfragen abgerufen werden wenn die Rechenkapazität für die Bearbeitung vorhanden ist. Daraus folgt

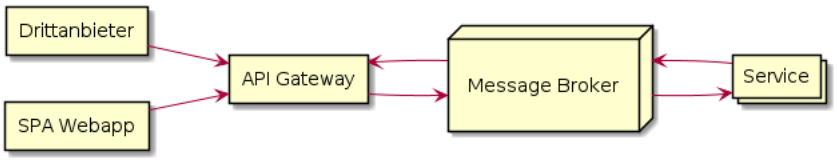

Abbildung 4.4: Architekturentwurf

ebenfalls das dem Empfänger und dem Sender der jeweils Andere unbekannt bleibt und eine einfache Replikation möglich ist. Da es sich um eine unidirektionale Kommunikation handelt kann der Sender nicht feststellen, ob die Anfrage abgerufen oder bearbeitet wurde: Dazu sind weitere Systeme nötig. [TVS17]

4.3 Architekturentwicklung

Im vorhergegangenen Abschnitt Architekturansätze, wurden einige Ansätze wie (Teil-)Probleme der generalisierten Anforderungen (Abschnitt 4.1) gelöst werden könnten. Da keine der vorgestellten Muster für diesen Anwendungsfall genau passend sind, muss eine besser angepasste Architektur entwickelt werden, die sich bei Bedarf auf die vorgestellten Architekturmuster zurückgreift.

Die Architektur wird von einem zentralen Bestandteil aus entwickelt: dem *Message Broker*. Er dient dazu, das alle Komponenten ohne vorher bekannt zu sein, unabhängig miteinander kommunizieren können. Dadurch wird direkt die Skalierbarkeit und Erweiterbarkeit der Architektur sichergestellt. Tatsächlich besteht die Message Broker (Abbildung 4.5) aus mehreren einzelnen Message Queues: Eine Message Queue für jeden Service, um die Aufgaben vom API Gateway weiterzuleiten. Wenn der Service während der Bearbeitung der Aufgabe eine Ausgabe erzeugt die für den Client interessant sein könnte, wird diese in einer eigens für diese Aufgabe angelegte Message Queue abgelegt. Wenn der Nutzer eine Aufgabe absendet fügt er einen generierten Universally Unique Identifier (UUID) an. Anhand dieser UUID kann im Anschluss die Message Queue identifiziert werden, in der Ausgaben des Services gesendet werden, wobei die Message Queue als Datenbank dient bis die Antworten abgefragt wurden. Dieser Aufbau des Message Brokers als Grundlage der Architektur ermöglicht insbesondere eine schnelle Veränderung der Architektur wenn sich die Anforderungen in Zukunft Ändern sollten, sie folgt dem Ansatz des *evolutionary designs*[3]. Auf einigen Möglichkeiten der Anpassung liegt dazu besonderer Augenmerk: Das API Gateway kann beliebig skaliert werden oder dem

[3] `https://martinfowler.com/articles/designDead.html`

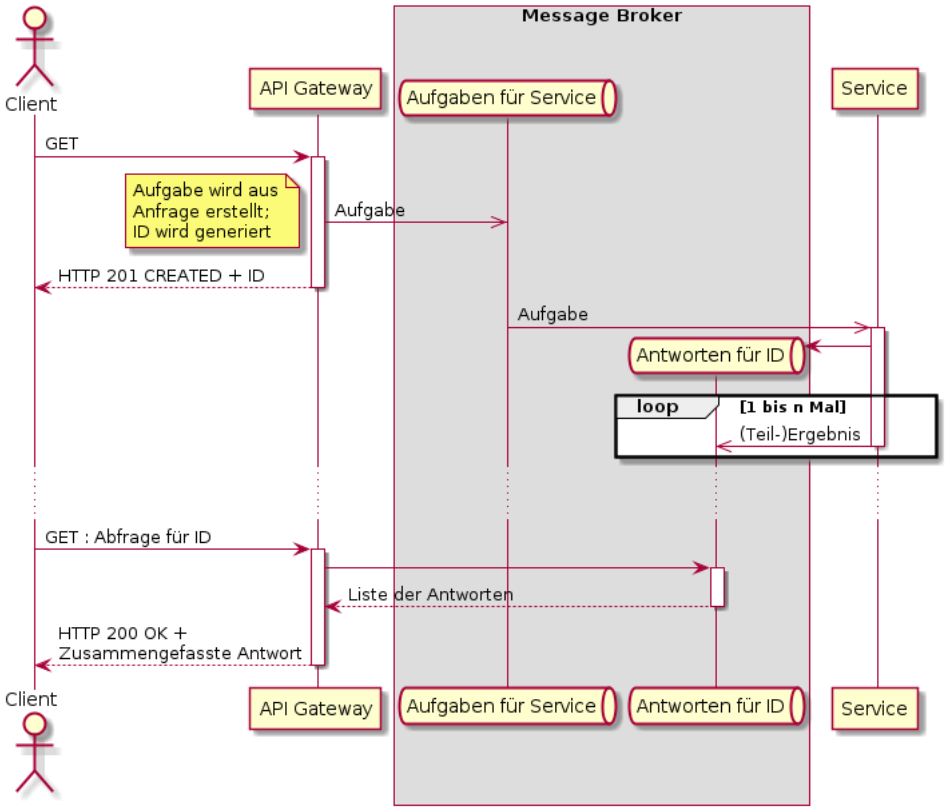

Abbildung 4.5: Anfragen und Antworten mittel Message Queueing

Backens for Frontends Prinzips (erläutert in Abschnitt 4.2.2). Das ist dann interessant wenn Clients neben der synchronen HTTP basierten Kommunikation noch eine asynchrone Kommunikationsart angeboten werden soll. Zum Beispiel könnte das weit verbreitete WebSockets-Protokoll verwendet werden um Antworten des Services direkt aus der Message Queue an den Client weiterzuleiten. Das würde die API Kommunikation für asynchrone Single Page Applications (SPAs) erleichtern während andere Anwendungen, die nur HTTP verwenden trotzdem unterstützt werden können.
Des weiteren kann der Service der die Anfragen bearbeitet beliebig repliziert werden, da keine direkte Verbindung zum API Gateway besteht. Ebenfalls können weitere Services angebunden werden indem der Message Broker um eine weitere Message Queue erweitert wird, die Aufgaben für den hinzugefügten Service entgegen nimmt.
Der größte Nachteil der Architektur liegt in dem hohen initialen Planungs- und Umsetzungsaufwand der im Widerspruch zum oben erwähnten *evolutionary design* steht. Außerdem ist der Message Broker ein „single point of failure“, das heißt wenn der Broker ausfällt, fällt das gesamte System aus. Daraus folgt, das der Broker eine robuste, also ausfallsichere und skalierbare Implementierung besitzen muss.

5 Implementierung

5.1 Übertragung der Architektur auf den Anwendungsfall

In der Umsetzung der in Abschnitt 4.3 entwickelten Architektur wird der erste Service ein Adapter zu der bestehenden JGen Applikation sein, wobei aus der Sicht der Anwendung irrelevant ist, ob es sich um einen vollständig eigenständigen Service handelt, oder um einen Adapter der ein bestehendes Programm verwaltet.

5.2 Message Broker

5.2.1 Aufgaben

Als zentrale Komponente der Architektur hat der Message Broker eine Schlüsselrolle. Er muss in der Lage sein Nachrichten zuverlässig in verschiedenen Message Queues zu speichern. Die Message Queues müssen dynamisch erstellt werden können, da die Antworten jeder Aufgabe in einer eigenen Message Queue versandt werden. Im besten Fall ist der Message Broker in der Lage skaliert zu werden, um zukünftigen Leistungsproblemen vorzugreifen. Damit der Entwicklungs- und Wartungsaufwand in diesem Projekt nicht zu groß wird, sollte eine bestehende Anwendung oder Framework verwendet werden.

5.2.2 Technologieauswahl

Es existieren viele verschiedene Implementierungen von Message Brokern, zum Beispiel Apache Kafka[1], RabbitMQ[2] oder Redis[3]. Redis ist zwar primär eine key-value in-memory Datenbank, unterstützt aber durch einen Listen Datentyp auch Message Queueing. Da zum initialen Deployment nur wenig Konfiguration notwendig ist, später aber durch

[1] `http://kafka.apache.org/`
[2] `https://www.rabbitmq.com/`
[3] `https://redislabs.com/solutions/use-cases/messaging/`

komplexere Konfiguration skaliert werden kann, passt es zu den Anforderungen die aktuell und absehbar in der Zukunft an den Message Broker gestellt werden. Des weiteren werden bereits Redis Instanzen in anderen Projekten der GNS Systems GmbH betrieben, wodurch sowohl bestehendes Wissen genutzt als auch der Wartungsaufwand im Betrieb reduziert werden kann.

5.3 API Gateway: „Messenger“

Um die geringe Geschäftslogik des API Gateway und die Hauptaufgabe, Nachrichten auszutauschen widerzuspiegeln, erhält es den Namen „Messenger“.

5.3.1 Aufgaben

Der Messenger überführt vor allem die HTTP Anfragen von Client in den Message Broker, von wo sie im Backend weiterverarbeitet werden. Dadurch wird auch die Vermittlung synchroner Verbindung zum Client und asynchronen Verbindungen über Message Queues vermittelt. Dafür wird beim Empfang einer HTTP Anfrage der enthaltene Authentifizierungstoken überprüft und ein UUID generiert der die HTTP Anfrage beantwortet und nach dem Hypermedia as the Engine of Application State (HATEOAS) Prinzip auf die Ergebnisse verweist. Während dessen wird die Anfrage in ein JSON-Format umgeformt und in die Message Queue des Redis eingespeist. Die User Storys aus Abschnitt 3.6 werden durch REST-Pfade abgebildet.

5.3.2 Technologie

Da für die Ansprechpartner im Projektkontext bessern Zugang zu python als zu anderen Programmiersprachen haben, wird diese Komponente in python entwickelt. Im python Kontext existieren einige populäre Frameworks, die für die Entwicklung von REST APIs zur Verfügung stehen. Einige der bekanntesten sind Django[4], Flask[5] und FastAPI[6]. Für diese Komponente wird FastAPI verwendet, da es ein Asynchronous Server Gateway Interface (ASGI) Framework ist, daher Nebenläufigkeit über *python asyncio*[7] unterstützt. Des weiteren wird eine automatische eine Dokumentation(Abbildung 5.1 nach OpenAPI

[4] `https://www.djangoproject.com/`
[5] `https://flask.palletsprojects.com`
[6] `https://fastapi.tiangolo.com/`
[7] `https://docs.python.org/3/library/asyncio.html`

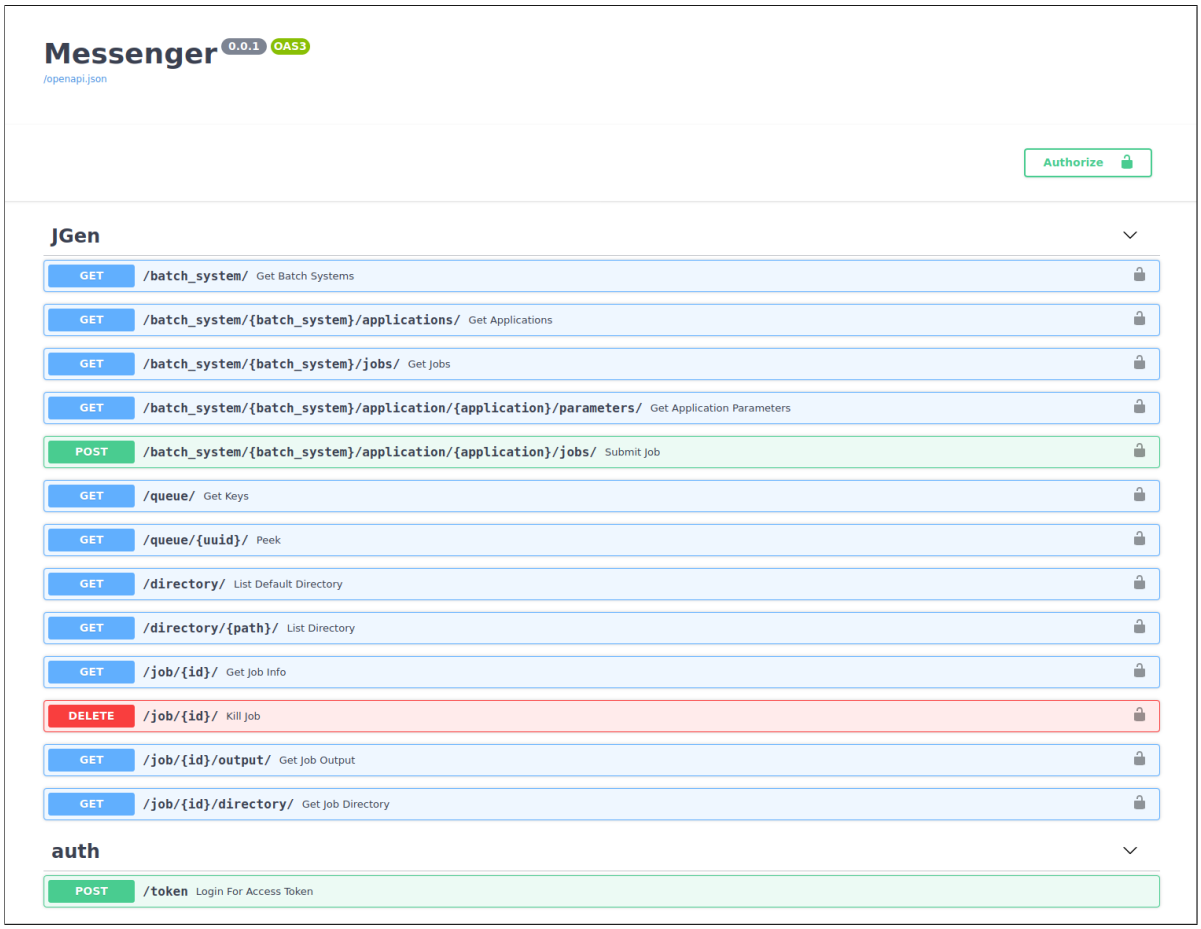

Abbildung 5.1: Screenshot Messenger OpenAPI Dokumentation

Spezifikation[8] bereitgestellt.

5.3.3 Authentifizierung

Damit nur bekannte Nutzer die HPC Ressourcen nutzen können müssen sie authentifiziert werden. Da die meisten Kunden der GNS Systems GmbH Windows Active Directory(AD) zur Verwaltung ihrer Anwender nutzen, wird Lightweight Directory Access Protocol (LDAP) verwendet um die Nutzer gegen einen Authentifizierungsserver zu authentifizieren. LDAP unterstützt nämlich eine Reihe von Open Source Verzeichnisdiensten, die somit ohne zusätzlichen Aufwand mit angebunden werden können.

Zur Authentifizierung wird der OAuth 2.0 Resource Owner Password Credential Grant(kurz: Password Grant)[Har12] verwendet: Nach Übermittlung der Anmeldeinformationen erhält der Client einen Access Token in Form eines JSON Web Token (JWT)

[8] `https://www.openapis.org/`

mit dem sich der Client für 24 Stunden ohne die erneute Angabe seiner Anmeldeinformationen authentifizieren kann. Bevor ein Access Token ausgestellt wird, stellt der Messenger durch eine Anfrage an den Windows AD Server fest, ob es sich um valide Anmeldeinformationen handelt. Durch Verwendung eines Access Tokens werden sowohl die Anfrage an den AD Server reduziert als auch die Nutzbarkeit verbessert, da nicht bei jeder Anfrage eine Nutzereingabe oder die unsichere Speicherung der Anmeldeinformationen im Klartext nötig ist.

5.3.4 Umsetzung

```python
# Funktion auf Pfad /batch_system/ registieren
# erfolgreiche Antworten sind HTTP 201 mit UUID payload
@router.get(
    "/batch_system/", response_model=uuid.UUID, status_code=201, responses=responses
)
# request, response und user werden beim Aufruf durch FastAPI injiziert
async def get_batch_systems(
    request: Request, response: Response, user: User = Depends(get_current_user)
):
    query = Query(
        user=user.username,
        parameters={"action": "get_parameters", "client": settings.jgen_client},
    )
    # Query Objekt in die Message Queue senden
    uuid = await dispatch_request(request.app.redis, query)
    # Antwort enthält Link wo Antworten des Backends abzurufen sind
    response.headers["Location"] = f"/queue/{uuid}/"
    return uuid
```

Listing 5.1: Code für einen REST Pfad

Für jeden der REST Pfade in Abbildung 5.1 wird eine Funktion wie in Listing 5.1 angelegt. Der Programmfluss wird von FastAPI übernommen und leitet Aufrufe an die python Funktionen weiter. Durch die Typisierung des Antworttypen in der Annotation wird zum einen die Antwort geprüft, sodass der Client garantiert davon ausgehen kann, dass eine korrekt formatierte Antwort gesendet wird. Des weiteren kann dadurch die Dokumentation der Pfade inklusive Datentypen wie sie in Abbildung 5.1 abgebildet ist generiert werden. Alle Abhängigkeiten werden nach dem Dependency Injection Muster von FastAPI injiziert. Das Objekt das die Verbindung zu Redis wird ebenfalls im Request Objekt mit übergeben. Die Pfade sind in zwei Module unterteilt: Zum einen „auth“ mit einem Pfad um ein Access Token zu erlangen und zum anderen „jgen“ mit anderen Pfaden über die die Funktionen des JGens abgebildet werden. Für eine Anbindung weiterer Dienste kann daher ein neues Modul hinzugefügt werden, damit auch im Messenger eine

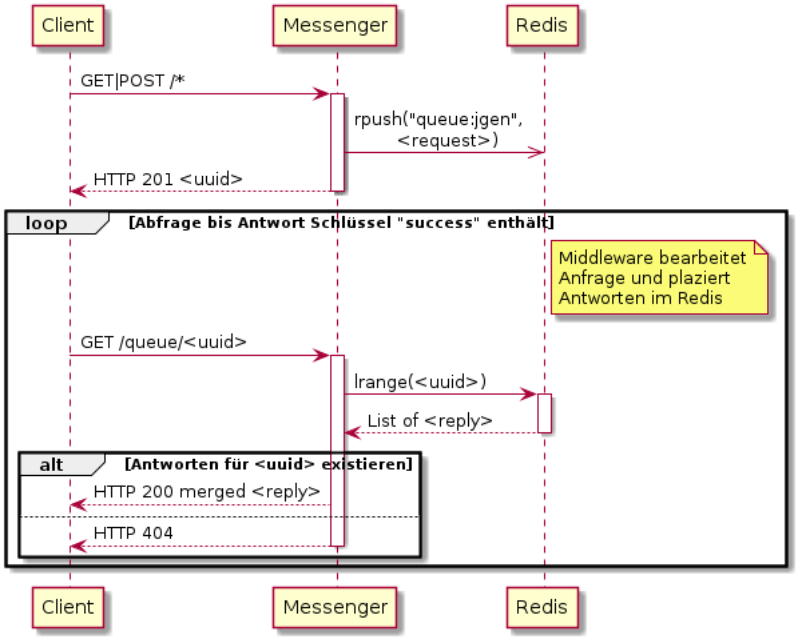

Abbildung 5.2: Sequenzdiagramm Messenger Interaktion

Trennung zwischen verschiedenen Aufgaben gewahrt bleiben kann.

Durch Orientierung an den Prinzipien der „twelve factor app“[Wig11] zur Entwicklung von Anwendungen die auf die Ausführung in der Cloud vorbereitet sind, ist der Messenger besonders für eine horizontale Skalierung und Orchestrierung in Containern angepasst. Begründet liegt dass in der Zustandslosigkeit der Komponente, weder Clients noch Message Queue stellen fest ob sie mit einer oder mehreren Instanzen des Messenger kommunizieren. Dadurch werden die Möglichkeiten der Leistungssteigerung durch Multi Threading des verwendeten *uvicorn* Webserver um die Möglichkeit der Replikation erweitert.

5.4 Nachrichtenformate

Damit Messenger und JGen Middleware kommunizieren können, ist vereinbartes Nachrichtenformat nötig. Dieses Format ist auch in der Lage eine einheitliche und für Softwareentwickler verständliche Kommunikation zwischen den Komponenten sicherzustellen. Mittels des Feldes „api_version“ ist die Versionierung möglich. Folglich kann sichergestellt werden dass Kommunikationsteilnehmer auch bei Veränderungen des Nachrichtenformats sicherstellen können dass der richtige Umgang mit Nachrichten gewählt wird. Da die Nachrichten primär von in Python entwickelten Anwendungen verwendet werden, sind Variablen mit mehrteiligen Bezeichnungen in Anlehnung an den Python Standard

durch Unterstriche getrennt.[RWC01] Die durch den Messenger generierte UUID wird verwendet um alle Nachrichten die sich auf eine Anfrage beziehen zu identifizieren. Damit der JGen im richtigen User Kontext verwendet wird, ist auch der Nutzername in den Nachrichten enthalten. Da sich die Struktur des JGens an einem Kommandozeilenaufruf orientiert, setzt sich dieser aus einem Befehl und Parametern, die die Aufgabe spezifizieren zusammen.

Das Antwortwortformat besitzt noch die zugehörige Task ID des JGen, damit eine Zuordnung von UUID zu Nutzer und Task ID transparent ist und dadurch die Nachvollziehbarkeit von Abläufen verbessert. Da ein Aufruf mehrere Antworteten nach sich ziehen kann, wird die letzte Nachricht durch das vorhanden sein eine „success“ Feldes angezeigt. In dem Fall, dass der Aufruf mehr als Misserfolg beziehungsweise Erfolg erzeugen kann, zum Beispiel eine Liste von Werten wird diese im Feld „payload“ transportiert währen `„success“:true` den Abschluss des Antwortstromes anzeigt. Damit Fehlerausgaben zum Anwender transportiert werden können um Fehler zu behandeln, werden diese in dem „message“ Feld abgespeichert.

```
{
    "api_version": "1.0",
    "user": "<username>",
    "uuid": "<uuid>",
    "command": "command|query|control",
    "parameters": {
        "<parameter>": "<value>"
    }
}
```

Listing 5.2: Nachrichtenformat für Anfragen

```
{
    "api_version": "1.0",
    "user": "<user>",
    "uuid": "<uuid>",
    "jgen_taskid": "<taskid>",
    "message": "<message>",     // optional
    "success": true|false,      // letzte Nachricht
    "payload": "<Text>"|[]|{}   // letzte Nachricht
}
```

Listing 5.3: Nachrichtenformat für Antworten

5.5 Service: „JGen Middleware“

Der Name setzt sich zusammen aus JGen, also der Anwendung die durch diese Komponente an den Messenger angebunden wird und Middleware, da diese Komponente effektiv

ein Adapter des JGens zu einem JSON Format, das auch für andere Anwendungen verwendet werden kann. Die Komplexität der darunter liegenden Anwendungen wird also verborgen, was auch als Middleware bezeichnet wird.[VAC+09][TVS17]

5.5.1 Aufgaben

Für jeden Nutzer wird vorausgesetzt das JGen in ihrem home directory auf dem File Server (Server auf dem HPC Quelldateien abgelegt werden) für sie installiert ist. Der Start der JGen Instanz findet über SSH statt und gibt als Ausgabe sowohl die Ports als auch den für die Curve Verschlüsselung benötigten öffentlichen Schlüssel als Ausgabe auf der Kommandozeile aus. Bis der JGen nach einem Zeitraum ohne Interaktion automatisch herunterfährt, kann über die vorher erhaltenen Ports mit dem Server kommuniziert werden. Bei den Ports handelt es sich um ZeroMQ Ports, einem Reply nach dem Request-Reply und einem Publisher nach dem Publish-Subscribe Muster[9]. Um dem JGen eine Aufgabe zu übermitteln wird diese in JSON Format an den Reply Port übermittelt, die wird durch eine Zahl beantwortet, der Task ID.
Damit die SSH Kommunikation passwortlos erfolgen kann, damit das Passwort des Nutzers nicht im Klartext kommuniziert werden muss, wird ein key file benötigt. Dadurch ist die JGen Middleware in der Lage nur mit dem Namen des Nutzers den Aufruf über SSH durchzuführen.

5.5.2 Umsetzung

Der gesamte Ablauf ist in Abbildung A.2 dargestellt. Die grundlegende Programmlogik wird durch die Klassen `RedisRequestObserver` und `JGenConnector` verwaltet. Wenn der `RedisRequestObserver` Nachricht in der Message Queue im Redis erhält, wird der `JGenConnector` mit dieser Nachricht aufgerufen. Der `JGenConnector` bietet eine Fassade der gesamten Programmlogik die zur Verwaltung der JGen Instanzen und der Kommunikation mit den Instanzen notwendig ist an.

Jede Anfrage wird durch die `Mapper` Klasse zwischen dem Nachrichtenformat aus Abschnitt 5.4 das auf der Message Queue empfangen wird mittels Skripten in der Skriptsprache *jq*[10] zu den tatsächlichen Befehlen die der JGen empfängt angepasst. Diese Informationen sind nicht im Python Quellcode enthalten, damit sie auch nach der In-

[9] `https://zguide.zeromq.org/docs/chapter2/#essaging-Patterns`
[10] `https://stedolan.github.io/jq/`

stallation der JGen Middleware an individuelle Änderungen des JGens angepasst werden können.

Um den JGen zu starten muss eine SSH Verbindung zu dessen Server aufgebaut werden. Die dabei erhaltenen Informationen werden als `JGenConnectionState` im `JGenConnectionStateStore` zusammen mit der Information über den Zeitpunkt der letzten Kommunikation gespeichert, wenn eine weitere Anfragen für den selben Nutzer empfangen wird, kann nach anhand dem Zeitpunkt der letzten Kommunikation festgestellt werden, ob angenommen werden kann das der JGen noch aktiv ist oder der zeitaufwändige Startaufruf über SSH stattfinden muss. Wird angenommen, das der JGen noch aktiv ist, wird die Kommunikation mittels der `JGenTask` Klasse vorgenommen. Diese Klasse abstrahiert die Kommunikation über zweit ZeroMQ Ports in einen einzelnen Methodenaufruf.

Die JGen Middleware ist auf die asyncio Python Bibliothek aufgebaut, das heißt das für jede empfangene Nachricht eine eigene *Coroutine* gestartet wird, sodass während zum Beispiel während auf Antworten des JGen gewartet wird, nebenläufig weitere Anfragen bearbeitet werden können.

5.6 Webanwendung

Um Stakeholdern und potentiellen Kunden die Möglichkeiten des hpc-tools aufzeigen zu können ist eine Weboberfläche von großer Bedeutung. Da es sich bei der Webanwendung um eine eigenständige Komponente handelt, ohne die das hpc-tool trotzdem ein vollständiges Produkt sein soll, erhält die Webanwendung eine eigenes Unterprojekt. Alternativ wäre auch möglich gewesen, das das API Gateway „Messenger“ die Weboberfläche an den Browser ausliefert.
Eine SPA ist in der Lage eine moderne und vom Backend möglichst unabhängige Lösung zu schaffen. VueJS ist ein JavaScript Framework [11] das durch Trennung der Komponenten in einzelne Dateien und einem zentralen Zustandsspeicher in Vuex[12] eine schnell Änder- und Erweiterbare SPA erzeugt. In Abbildung 5.3 sind alle Browserversionen aufgelistet die mit der Konfiguration von Vue unterstützt werden, darunter die geforderen Edge, Firefox, Chrome und Safari.

Die Anwendung spaltet sich in zwei Fenster: „Generate“ und „Control“, die über eine Navigationszeile ausgewählt werden können. Im Generate Fenster wird eine dynamisch

[11] `https://vuejs.org/`
[12] `https://vuex.vuejs.org/`

```
and_chr 81    ie 11
and_ff 68     ios_saf 13.4
and_qq 1.2    ios_saf 13.3
and_uc 12.1   ios_saf 12.2-12.4
android 81    kaios 2.5
baidu 7.12    op_mini all
chrome 81     op_mob 46
chrome 80     opera 68
edge 81       opera 67
edge 80       safari 13.1
edge 18       safari 13
firefox 76    samsung 11.1
firefox 75    samsung 10.1
firefox 74
```

Abbildung 5.3: Unterstützte Browser und Versionen

vom JGen übermittelte Parameterliste angezeigt, wo Parameter für den zu startenden Job ausgewählt werden können (Abbildung 5.4). Durch Betätigung der „Submit“ Schaltfläche werden die Parameter abgesendet und der Job eingereicht.
Im „Control“ Fenster wird eine Liste der derzeit aktiven Jobs angezeigt und durch ausklappen werden weitere Funktionen für mehr Informationen für den jeweiligen Job angezeigt (Abbildung 5.5).

Durch Verwendung des Vuetify Frameworks [13] für die visuellen Komponenten besitzt die Anwendung ein *responsive design* und ist ohne erhöhten Entwicklungsaufwand für verschiedene Gerätetypen wie Laptop, Tablet oder Smartphone angepasst.

[13] `https://vuetifyjs.com/en/`

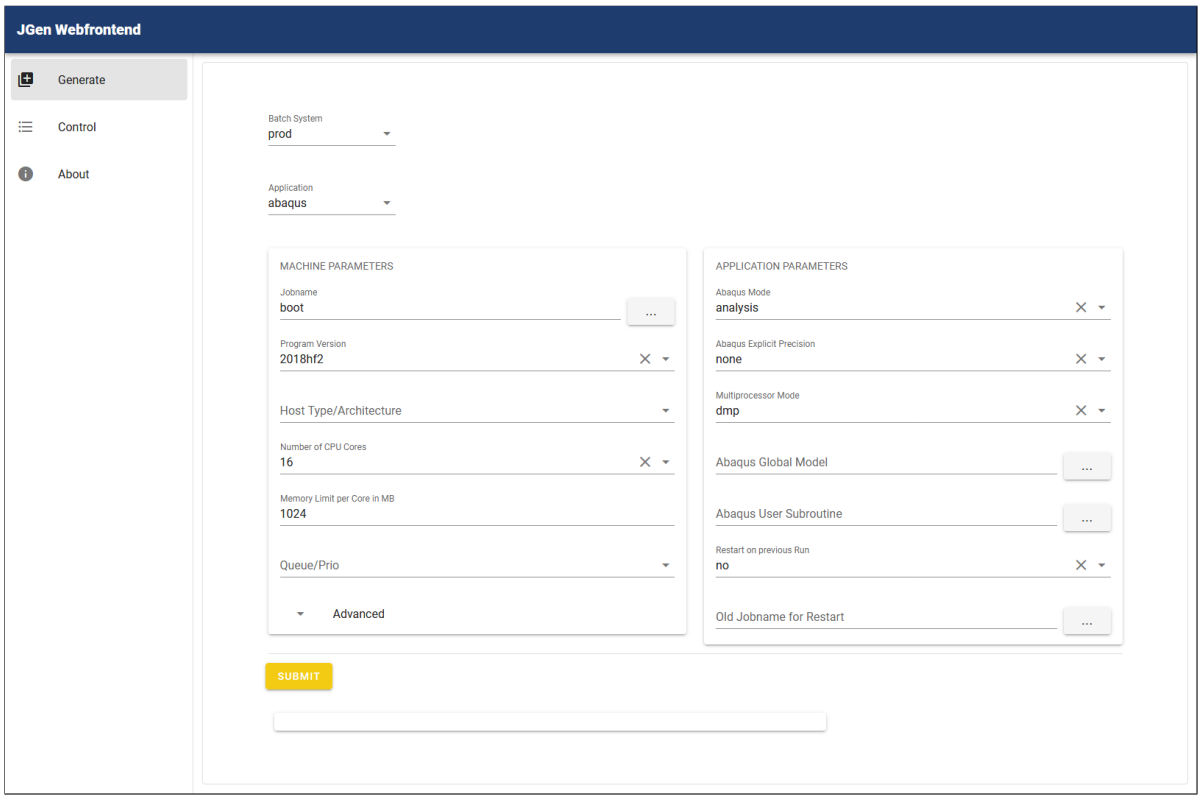

Abbildung 5.4: Screenshot der Joberstellung (Desktop)

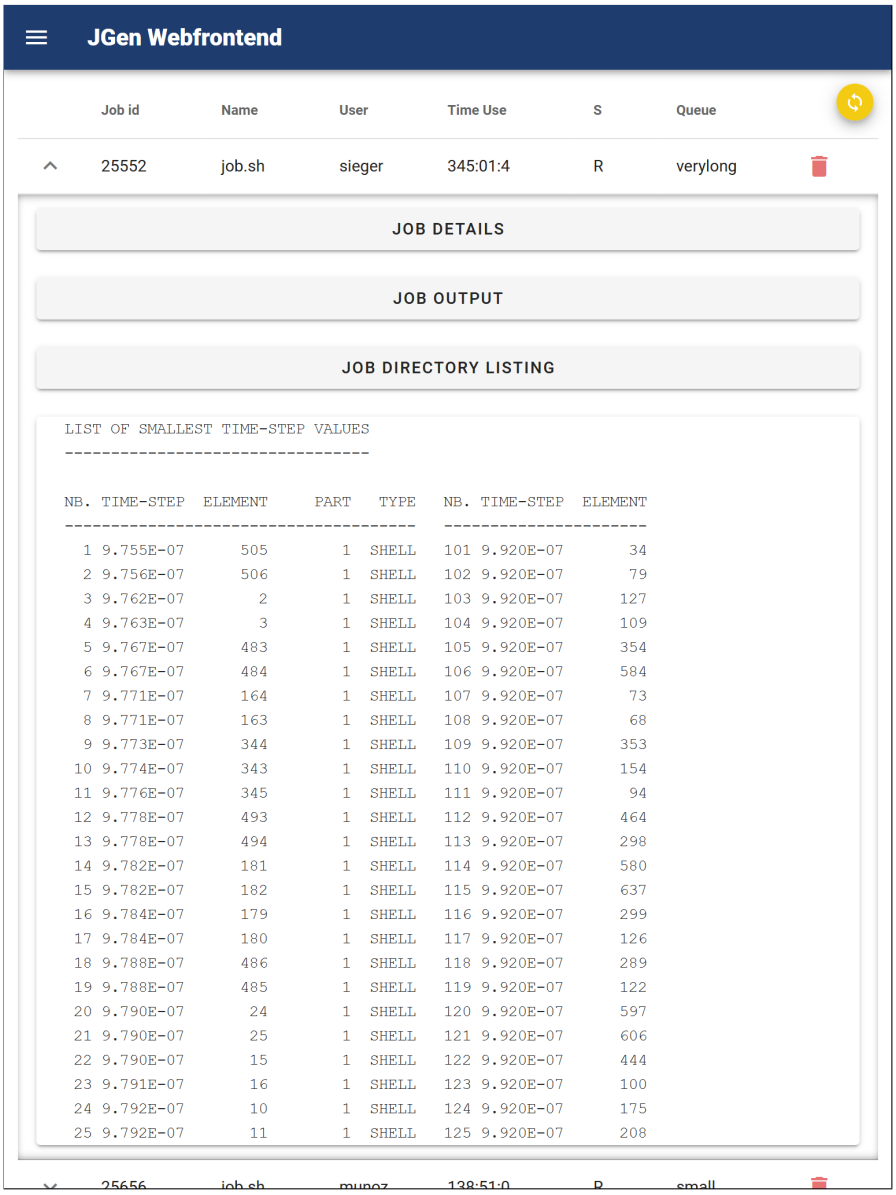

Abbildung 5.5: Screenshot der Jobübersicht (Tablet)

6 Einordnung der Ergebnisse

Im Laufe des Projektes wurde eine Analyse der derzeitigen Anforderungen an die Anwendung erstellt. Anhand der Ergebnisse konnte eine Architektur entwickelt werden, die moderne Architekturmuster umsetzt und auf für den Einsatz weiterer angepasst ist, sollte das benötigt werden. Somit wurde die Designphase aus dem klassischen Wasserfallmodell als Grundlage für ein agiles Projekt angepasst und umgesetzt.

Auf Grundlage der Architektur wurde ein MVP entwickelt auf dessen Grundlage Kunden für das Produkt akquiriert werden können, wodurch das finanzielle Risiko der GNS Systems GmbH gegenüber der vollständigen Entwicklung eines Produktes vor dessen Markteinführung reduziert ist.

Eine Validierung des Ergebnisses hat deshalb bisher nur mittels Präsentation vor den Interessenvertretern innerhalb des Unternehmens stattgefunden, zumal es schon Kontakt mit interessierten Kunden gab.

Alle funktionalen Anforderungen(Abschnitt 3.6 wurden sowohl in der REST API als auch in der Webanwendung umgesetzt.

Die geforderten nicht funktionalen Anforderungen sind gemäß ihrer Priorisierung(Tabelle 3.3) umgesetzt. Die Portabilität und Wartbarkeit ist durch die Modulariät der Architektur und der eingesetzen Technologien gegeben. Die in Python entwickelten Komponenten werden mittel virtual environemnts vom restlichen System gekaspelt, sodass Abhängigkeiten versioniert für jede Anwendung vorliegen, während die Abhängigkeiten des VueJS Projektes durch das Verhalten des Paketmanagers (yarn) im lokalen Order installiert und gemeinsam mit dem Quellcode paketiert wurden. Die Kompatibilität mit dem Zielbetriebssystem(CentOS 7) wurde ist sichergestellt, da sowohl die automatisierten Tests der Continuous Integration Pipeline in einem Docker Image als auch eine Testinstallation zu Präsentationszwecken in einer virtuellen Maschine mit Betriebssystem durchgeführt wurden. Durch Anwendung von der SOLID-Prinzipien[Mar14] im Quellcode ist dieser auch für andere Entwickler verständlich.

Mangels ausgiebiger Lasttests ist über die Zuverlässigkeit keine ausreichende Aussage zu treffen. Die Bedienbarkeit wurde bisher ohne formalen Nutzertest durch die Interes-

senvertreter als gut wahrgenommen, obwohl bei ihnnen zu beachten ist, das sie aufgrund der tiefgreifenden HPC Kenntnissen nicht der Zielgruppe entsprechen. Die Sicherheit durch https Verbindungen ist ebenfalls nicht umgesetzt, da dazu gültige Zertifikate nötig sind.

7 Ausblick

Mit dem Projekt wie es zu diesem Zeitpunkt besteht, können die Aufgaben eines MVP erfüllt werden: Es kann auf Kunden zugegangen werden um die Nachfrage nach dem Produkt zu prüfen. Wenn Nachfrage vorhanden ist, gibt es einige Punkte, an denen eine weitere Entwicklung ansetzen kann.

Die REST Anfragen an den Messenger werden derzeit mit einem Access Token authentifiziert, der mittels des OAuth 2 Password Grant erlangt wurde. Dieses Verfahren ist zwar generell sicher, insbesondere da sowohl dem Frontend als auch dem Backend vertraut werden kann. In den OAuth2 best practices wird allerdings davon abgeraten, diesen Ablauf zu nutzen. Begründet wird das zum einen mit der Schwierigkeit Zwei-Faktor-Authentifizierung zu implementieren und zum anderen das Nutzer sich angewöhnen an unterschiedlichen Stellen ihre Nutzerinformation anzugeben. Der übliche Ablauf von OAuth2 Authentifizierung sieht vor, dass der Access Token durch Login auf der Oberfläche einer vertrauten Instanz akquiriert wird.[LBLF20] Während die Rolle der vertrauten Instanz im Internet häufig durch Google oder Facebook übernommen wird, gibt es auch Lösungen dies innerhalb eines Unternehmens mit Windows AD oder die gesamte Authentifizierung mittels Windows Single Sing on zu erledigen.

Des Weiteren ist die JGen Middleware in ihrem jetzigen Zustand schlecht skalierbar. Es wäre nötig die Verwaltung von aktiven Verbindungen für verschiedene Instanzen des gleichen Dienstes zu vereinheitlichen. Wie viele Nutzer die Implementierung davon notwendig machen, müssen Lasttests zeigen. Im jetzigen Zustand wird die Leistung durch die asyncio Bibliothek eingeschränkt, die Multithreading erschwert.

Ein wichtiger Schritt in Richtung einer verlässlichen Anwendunge ist es, neben den schon erwähnten Usability- und Lasttests noch weitere Testarten umzusetzen. Zum einen ist die Abdeckung des Quellcodes durch Unit Tests zu gering und sollte erhöht werden, zum anderen sind Tests ganzer Komponenten nicht automatisiert. Zum anderen besitzt die Webanwendung noch keine Tests, diese sollten aber wenn die Oberfläche eine für Nutzer zufriedenstellende Form hat hinzugefügt werden. Damit die Anwendung sicher betrieben werden kann sind zudem auch Sicherheitstests durchzuführen.

Selbstständigkeitserklärung

Hiermit versichere ich, Lukas Balzereit, dass ich die vorliegende Bachelorarbeit mit dem Titel:

Aufbau einer generischen Schnittstelle zur Anbindung und Verwaltung von High-Performance-Clustern

selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Ort, Datum	Unterschrift

Literaturverzeichnis

[Atl20] Atlassian Incident Management: *Incident postmortems*. Version: 2020. https://www.atlassian.com/incident-management/handbook/postmortems, Abruf: 14.10.2020

[Bal09] Balzert, Helmut: *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering*. Heidelberg : Spektrum Akademischer Verlag, 2009. http://dx.doi.org/10.1007/978-3-8274-2247-7_1. http://dx.doi.org/10.1007/978-3-8274-2247-7_1. – ISBN 978–3–8274–2247–7

[Coo08] Cooper, Alan: *The origin of personas*. Version: 2008. https://www.cooper.com/journal/2008/05/the_origin_of_personas/, Abruf: 12.05.2020

[Dow] Downing, Chris: *How the Cloud Is Falling Short for HPC*. https://www.hpcwire.com/2018/03/15/how-the-cloud-is-falling-short-for-research-computing/, Abruf: 09.09.2020

[DS10] Dowd, Kevin ; Severance, Charles: *High performance computing*. Rice University, 2010

[FH11] Fischer, Peter ; Hofer, Peter: *Lexikon der Informatik*. Springer-Verlag Berlin Heidelberg, 2011. http://dx.doi.org/10.1007/978-3-642-15126-2_3. http://dx.doi.org/10.1007/978-3-642-15126-2_3

[Fow15] Fowler, Martin: *Monolith First*. Version: 2015. https://martinfowler.com/bliki/MonolithFirst.html, Abruf: 01.10.2020

[GD20] Gil, Dario ; Dabbar, Paul ; COVID-19 High-Performance Computing Consortium (Hrsg.): *The science behind the first proposals to fight COVID-19 with supercomputers*. Version: 2020. https://covid19-hpc-consortium.org/blog/the-science-behind-the-

first-proposals-to-fight-covid-19-with-supercomputers, Abruf: 14.10.2020

[Gee20] Geers, Michael: *Micro Frontends in Action.* Manning Publications, 2020 https://micro-frontends.org/

[GNSa] GNS Systems GmbH: *Das Unternehmen.* https://www.gns-systems.de/ueber-uns/, Abruf: 04.09.2020

[GNSb] GNS Systems GmbH: *SET.* https://www.gns-systems.de/it-dienstleistungen/software-engineering/, Abruf: 04.09.2020

[Gol19] Goll, Joachim: *Entwurfsprinzipien und Konstruktionskonzepte der Softwaretechnik: Strategien für schwach gekoppelte, korrekte und stabile Software.* Wiesbaden : Springer Fachmedien Wiesbaden, 2019. http://dx.doi.org/10.1007/978-3-658-25975-4. http://dx.doi.org/10.1007/978-3-658-25975-4. – ISBN 978–3–658–25975–4

[Gru10] Grun, Paul: *Introduction to InfiniBand for End Users.* https://www.mellanox.com/pdf/whitepapers/Intro_to_IB_for_End_Users.pdf. Version: 2010

[Har12] Hardt, D.: The OAuth 2.0 Authorization Framework / RFC Editor. Version: 2012. https://www.rfc-editor.org/info/rfc6749. 2012 (6749). – RFC

[Jac12] Jacobson, Daniel: *Embracing the Differences : Inside the Netflix API Redesign.* Version: 2012. https://netflixtechblog.com/embracing-the-differences-inside-the-netflix-api-redesign-15fd8b3dc49d, Abruf: 15.10.2020

[LBLF20] Lodderstedt, Torsten ; Bradley, John ; Labunets, Andrey ; Fett, Daniel: *OAuth 2.0 security best current practice.* Version: 2020. https://tools.ietf.org/html/draft-ietf-oauth-security-topics-16.html

[LRW18] Lachnit, Malte ; Riedel, Stephan ; Woll, Christopher: Cloud-Infrastruktur für High-Performance Computing. In: *iX Magazin für professionelle Informationstechnik* 09/2018 (2018), S. 96ff

[Mar14] Martin, Robert C.: *Clean Coder.* mitp Verlags GmbH & Co. KG, 2014

[Moo12] Moogk, Dobrila R.: Minimum Viable Product and the Importance of Experimentation in Technology Startups. In: *Technology Innovation Management Review* 2 (2012), 03/2012, 23-26. http://dx.doi.org/http://doi.org/10.22215/timreview/535. – DOI http://doi.org/10.22215/timreview/535. – ISSN 1927–0321

[MW16] Montesi, Fabrizio ; Weber, Janine: Circuit Breakers, Discovery, and API Gateways in Microservices. In: *CoRR* abs/1609.05830 (2016). http://arxiv.org/abs/1609.05830

[Nat05] National Research Council ; Graham, Susan L. (Hrsg.) ; Snir, Marc (Hrsg.) ; Patterson, Cynthia A. (Hrsg.): *Getting Up to Speed: The Future of Supercomputing*. Washington, DC : The National Academies Press, 2005. http://dx.doi.org/10.17226/11148. http://dx.doi.org/10.17226/11148. – ISBN 978–0–309–09502–0

[Rob20] Robert Koch-Institut: *Antworten auf häufig gestellte Fragen zum Coronavirus SARS-CoV-2 / Krankheit COVID-19*. Version: 2020. https://www.rki.de/SharedDocs/FAQ/NCOV2019/gesamt.html, Abruf: 14.10.2020

[RWC01] Rossum, Guido van ; Warsaw, Barry ; Coghlan, Nick: Style Guide for Python Code. Version: 2001. https://www.python.org/dev/peps/pep-0008/. 2001 (8). – PEP

[Sne20] Snell, Addison: Worldwide HPC Market 2019 Actuals, 2020-24 Forecast, Including Effects of COVID-19. (2020)

[TVS17] Tanenbaum, Andrew S. ; Van Steen, Maarten: *Distributed systems: principles and paradigms*. Maarten van Steen, 2017

[VAC+09] Vogel, Oliver ; Arnold, Ingo ; Chughtai, Arif ; Ihler, Edmund ; Kehrer, Timo ; Mehlig, Uwe ; Zdun, Uwe: *Software-Architektur*. Heidelberg : Spektrum Akademischer Verlag, 2009. http://dx.doi.org/10.1007/978-3-8274-2267-5_1. http://dx.doi.org/10.1007/978-3-8274-2267-5_1. – ISBN 978–3–8274–2267–5

[Wig11] Wiggins, Adam: The twelve-factor app. In: *The Twelve-Factor App* (2011). https://12factor.net

Anhang

```
|-- bachelorarbeit.pdf  # elektronische Version der Arbeit
|-- diagramme           # alle Diagramme zur Anwendung
|-- quellcode
|   |-- backend         # Unterprojekt Messenger
|   |-- ci
|   |-- copyright.txt
|   |-- frontend        # Unterprojekt Webanwendung
|   |-- Jenkinsfile     # Definition CI/CD Pipeline
|   `-- jgen_middleware # Unterprojekt JGen Messenger
|-- screenshots         # Screenshots der Webandwendung
|   |-- desktop
|   |-- smartphone
|   `-- tablet
`-- webquellen          # Ausdrucke von Internetquellen
```

Abbildung A.1: Inhalt des elektronischen Anhangs

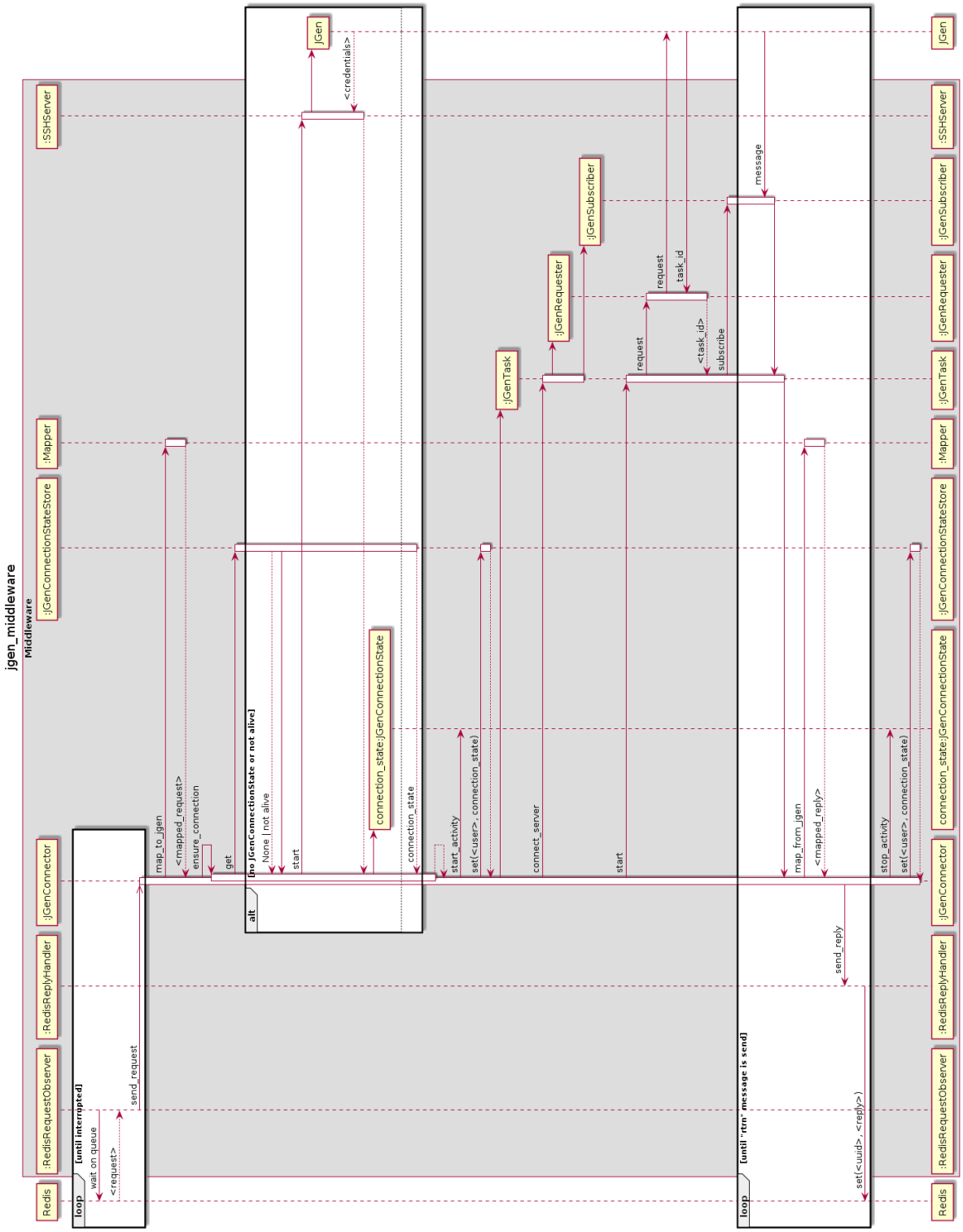

Abbildung A.2: Sequenzdiagramm Middleware

Abbildung A.3: Screenshot der Joberstellung (Desktop)

Abbildung A.4: Screenshot der Jobübersicht (Desktop)