ins
INSTITUTE FOR
NETWORKED SOLUTIONS

OST
Ostschweizer
Fachhochschule

LTB Organizer

Studienarbeit

Studiengang Informatik
OST – Ostschweizer Fachhochschule
Campus Rapperswil-Jona

Herbstsemester 2021

Autor(en): Daniel Els, Loris Keller
Betreuer: Prof. Laurent Metzger

Abstract

Der Lab Topology Builder (LTB) ist eine Applikation des «Institute for networked solutions (INS)», welche für Forschungs- und Schulungszwecke benutzt wird, um virtuelle Umgebungen bereitzustellen. Das Tool ist eine strategische Ressource des INS und hat für die Zukunft eine hohe Relevanz. Die virtuellen Umgebungen werden anhand von Labs definiert, wobei diese durch Benutzer manuell gestartet und beendet werden. Dies bedingt, dass keine Planung zur Auslastung und zur Reservation von Ressourcen existiert und teilweise die verfügbaren Server überbeansprucht werden, bzw. nicht voll ausgelastet sind.

Ziel der Arbeit ist eine zentrale Organisation und Koordination der Ressourcen und deren Deployment im LTB. Es soll nie möglich sein, mehr Ressourcen in Anspruch zu nehmen, als dass verfügbar sind. Um eine Planung der Ressourcen zu ermöglichen, soll die Bereitstellung der Labs nur noch über die Erstellung von Reservationen möglich sein. Die Bereitstellung und die Entfernung der Labs sollen neu automatisch und anhand der definierten Reservationen erfolgen.

In der Arbeit wurde die bestehende Applikation erweitert. Dabei wurden die Labs um Kosten erweitert und eine Erfassung von Deployment Nodes mit ihren verfügbaren Ressourcen ermöglicht. Labs können nur noch über Reservationen gebucht werden, wobei eine Überwachung sämtlicher Ressourcen sichergestellt ist.

Management Summary

Ausgangslage

Der Lab Topology Builder (LTB) ist eine Applikation des «Institute for networked solutions (INS)», welche für Forschungs- und Schulungszwecke benutzt wird, um virtuelle Umgebungen bereitzustellen. Das Tool ist eine strategische Ressource des INS und hat für die Zukunft eine hohe Relevanz.

Die virtuellen Umgebungen sind durch Labs definiert, welche die Geräte (Virtual Machines (VM), Docker Container, etc.) und Verbindungen unter den Geräten definieren. Die Labs werden auf unterschiedlichen physischen Servern bereitgestellt. Das Deployment, sowie die Entfernung eines Labs wird durch den Benutzer manuell angesteuert. Dies bedingt, dass keine Planung zur Auslastung und zur Reservation von Ressourcen existiert und teilweise die verfügbaren Server überbeansprucht werden, bzw. nicht voll ausgelastet sind.

Ziel der Arbeit

Ziel der Arbeit ist eine zentrale Organisation und Koordination der Ressourcen und deren Deployment im LTB. Es soll nie möglich sein, mehr Ressourcen in Anspruch zu nehmen, als dass verfügbar sind. Um eine Planung der Ressourcen zu ermöglichen, soll die Bereitstellung der Labs nur noch über die Erstellung von Reservationen möglich sein. Die Bereitstellung und die Entfernung der Labs sollen neu automatisch und anhand der definierten Reservationen erfolgen.

In Zukunft soll es möglich sein, eine Art «Credit-System» zu implementieren. Dieses soll ermöglichen, die zeitliche Nutzung von Ressourcen für einen Benutzer zu beschränken. Zusätzlich soll es zukünftig möglich sein, die Dienste des LTB an externe Partner zu vermieten.

Ergebnis

In der Arbeit wurden die Ziele "Ressourcenbasiertes Deployment" und "Reservationen von Labs" konzeptionell erarbeitet und umgesetzt. Dabei wurde die bestehende Applikation erweitert.

Um ein ressourcenbasiertes Deployment umzusetzen, wurden zuerst Metriken definiert, welche die physischen Ressourcen repräsentieren. Diese sind: "Anzahl an logischen CPUs", "Arbeitsspeicher" und "Speicherplatz auf der Disk". Diese Ressourcen müssen pro Gerät definiert werden, um zu ermitteln, wie viele Ressourcen in Anspruch genommen werden. Zusätzlich sind die verfügbaren physischen Server als

"Deployment Nodes" in der Anwendung mit ihren verfügbaren Ressourcen abgebildet. Anhand der Kosten und den verfügbaren Ressourcen wird ermittelt, ob ein Lab bereitgestellt werden kann. Ebenso kann die Auslastung der einzelnen Nodes dargestellt werden.

Neu können Labs nur noch über Reservationen gebucht werden, welche einen gewünschten Start- und Endzeitpunkt definieren. Die möglichen Tageszeiten für einen Start oder ein Ende werden durch die INS-Administration definiert, um eine reibungslose Bereitstellung sicherzustellen. Einerseits wird geprüft, ob genügend freie Ressourcen im gewünschten Zeitraum zur Verfügung stehen, andererseits wird sichergestellt, dass der gewünschte Startzeitpunkt für eine Reservation eingehalten werden kann. Dafür werden für die Geräte ihre entsprechenden Deployment- und Cleanupzeiten definiert.

Ausblick

In einem ersten Schritt könnte die Lösung durch ein paralleles Deployment erweitert werden. Dies würde eine klare Optimierung im Prozess bringen, wobei initial sicher eine volle Parallelisierung auf unterschiedlichen Deployment Nodes erreicht werden kann.

In einem weiteren Schritt könnte ein Credit-System erarbeitet und umgesetzt werden. Dabei muss ein Credit auf die entsprechenden Kosten und die effektive Laufzeit eines Labs abgebildet werden. Ein weiterer Aspekt würde die Abrechnung bilden, wenn die Dienste des LTB an externe Parteien vermietet werden. Dafür ist auch ein Konzept notwendig, wie ein Nutzer im allgemeinen neue Credits erlangen kann.

Aufgabenstellung

The Lab Topology Builder (LTB) is the application used by the INS in the research and in the education to build virtual environements.

This tool is a strategic asset for the future of the INS.

The LTB deploys the labs on different physical servers part of the LTB pool. At this point, there is no planification of the ressources and no ressources reservation possibilities. This means that the LTB servers are sometimes over-utilized and we have issues starting new labs and sometimes unterutilized and we would like to know about it.

The goal of that thesis is to create an application that is the central point for the coordination of all the activities in LTB. The only way to access a lab hosted on LTB servers should be through the prior reservation of ressouces. The ressources should be monitored and the organizer should allow as many reservations as available ressources. The labs should be automatically started and automatically stopped when the booking is over.

The organizer should implement a credit system where a user is only allowed to use LTB for a limited amount of time and should request (buy) new credits if exhausted. This should eventually allow the INS to rent LTB out to external entities.

It is possible to start this thesis as a semester thesis and to extend it into a bachelor thesis.

Infrastructure: Kubernetes

Language: to be defined

Frontend: React, to be defined

Inhaltsverzeichnis

1 Einführung und Ziele

1.1 Zweck

Dieses Dokument soll eine Übersicht über das Projekt "LTB Organizer" geben. Einerseits soll beschrieben werden, was das Ziel der Studienarbeit ist, andererseits soll eine Übersicht über verwendete Ressourcen und Planungsaktivitäten gegeben werden.

1.2 Aufgabenstellung

Der Lab Topology Builder (LTB) ist eine Applikation des INS, welche für Forschungs- und Schulungszwecke benutzt wird, um virtuelle Environments bereitzustellen. Das Tool ist eine strategische Ressource des INS und hat für die Zukunft eine hohe Relevanz.

Die definierten Labs werden auf physischen Servern bereitgestellt, welche zusammen den LTB Pool bilden. Aktuell ist es nicht möglich, Labs im Voraus zu planen oder Ressourcen bereits im Voraus zu reservieren. Ebenso kommt es vor, dass der Pool teils über- und teils unterbeansprucht wird.

Ziel dieser Arbeit ist es einen zentralen Organizer für die Koordination im LTB zu definieren und zu entwickeln. Die Ressourcen sollen neu reserviert werden, bevor ein Lab gestartet werden kann. Der Organizer soll die Ressourcen überwachen und sicherstellen, dass nur so viele Ressourcen gebucht werden, wie auch vorhanden sind. Die Labs sollen anhand der Reservierungen automatisch gestartet und gestoppt werden. Zusätzlich soll es neu ein Credit-System geben, wobei die Credits anhand der gebuchten Ressourcen aufgebraucht werden und entsprechend neu beantragt werden müssen. In einem ersten Schritt geht es vor allem darum, alle mögliche Ansätze zu definieren und zu validieren. Danach sollen mittels Priorisierung die Features umgesetzt werden.

1.3 Qualitätsziele

Die meisten Qualitätsmassnahmen sind bereits vorgegeben. Ziel muss es sein, die aktuelle Qualität beizubehalten.

Massnahme	Zeitraum	Ziel
CI/CD	Bei jedem Merge-Request	Fehlerverhinderung, Verbesserung der Code Qualität
Retro	Fortlaufend	Verbesserung der zukünftigen Sprints
Dokumentation	Fortlaufend	Gewährleistung der Gültigkeit und Aktualität
Code Reviews	Bei jedem Merge-Request	Fehlerverhinderung, Verbesserung der Code Qualität
Automatisierte Tests	Fortlaufend	Fehlerverhinderung, Verbesserung der Code Qualität
Linter (Frontend)	Fortlaufend	Einhaltung des Codestyles

Tabelle 1: Qualitätsziele

1.4 Stakeholder

Die folgende Tabelle stellt die Stakeholder des LTB Organizer und ihre jeweilige Intention dar.

Funktion	Person	Erwartungshaltung
Projektteam	Daniel Els	Entwickelt die Applikation weiter, so dass diese allen Anforderungen der anderen Stakeholder gerecht werden.
Projektteam	Loris Keller	Entwickelt die Applikation weiter, so dass diese allen Anforderungen der anderen Stakeholder gerecht werden.
Betreuungsperson SA / Auftraggeber	Prof. Laurent Metzger	Überwacht den Arbeitsfortschritt und interveniert bei grossen Abweichungen bezüglich des Ziels dieser Arbeit
Mitarbeiter INS	Yannick Zwicker	Unterstützt das Projektteam mit Fachwissen und hat die Entscheidungsgewalt bei der Implementation
Mitarbeiter INS	Sebastian Hug	Unterstützt das Projektteam mit Fachwissen und hat die Entscheidungsgewalt bei der Implementation

Tabelle 2:Stakeholder

2 Rahmenbedingungen

Beim Lösungsentwurf waren verschiedene Rahmenbedingungen zu beachten, da sie in der Lösung fortwirken. Dieser Abschnitt stellt diese dar und erklärt auch, wo nötig, deren Motivation.

2.1 Organisatorische Rahmenbedingungen

Für die Studienarbeit werden pro Student 8 ECTS vergeben, wofür ein Aufwand von 240 Stunden erwartet wird. Insgesamt entspricht dies einem Aufwand von 480 Stunden.

Das Projekt begann mit der Kick-off-Sitzung am 22. September und endet am 24. Dezember mit der Abgabe aller Dokumente bis um 17 Uhr.

2.2 Technische Rahmenbedingungen

Für die lokale Entwicklung wird das "Windows Subsystem for Linux (wsl)" mit den folgenden Spezifikationen verwendet:

Randbedingung	**Erläuterung, Hintergrund**
Datenbank mit PostgreSQL	Entwicklung mit Version 12.3
Backend in Python/Django	Entwicklung mit Version 3.7
Frontend mit React und NodeJs	Entwicklung mit Version 10.24.1
CI / CD mit Docker Container	Das Projekt verwendet die CI/CD Pipelines von GitLab und nutzt dafür Docker Container

Tabelle 3: Technische Rahmenbedingungen

Das Projekt LTB Organizer baut auf dem bereits existierenden LTB Projekt auf. Die Applikationsarchitektur ist somit bereits vorgegeben und wird deshalb nicht ausführlich in diesem Dokument beschrieben.

3 Kontextabgrenzung

Dieser Abschnitt definiert das Umfeld des LTB Organizers. Er zeigt auf, welche Benutzer und mit welchen Systemen interagiert wird.

3.1 Fachlicher Kontext

Zurzeit wird die LTB Applikation nur von den INS Mitarbeitern genutzt. Sie kennen die Abläufe und haben das Fachwissen. Sie stellen externen Personen die Labs zur Verfügung.

Der LTB Organizer ist eine Weiterentwicklung des LTB selbst. Es ist ein erster Schritt, um die LTB Applikation für externe User ohne Fachwissen nutzbar zu machen.

3.2 Technischer Kontext

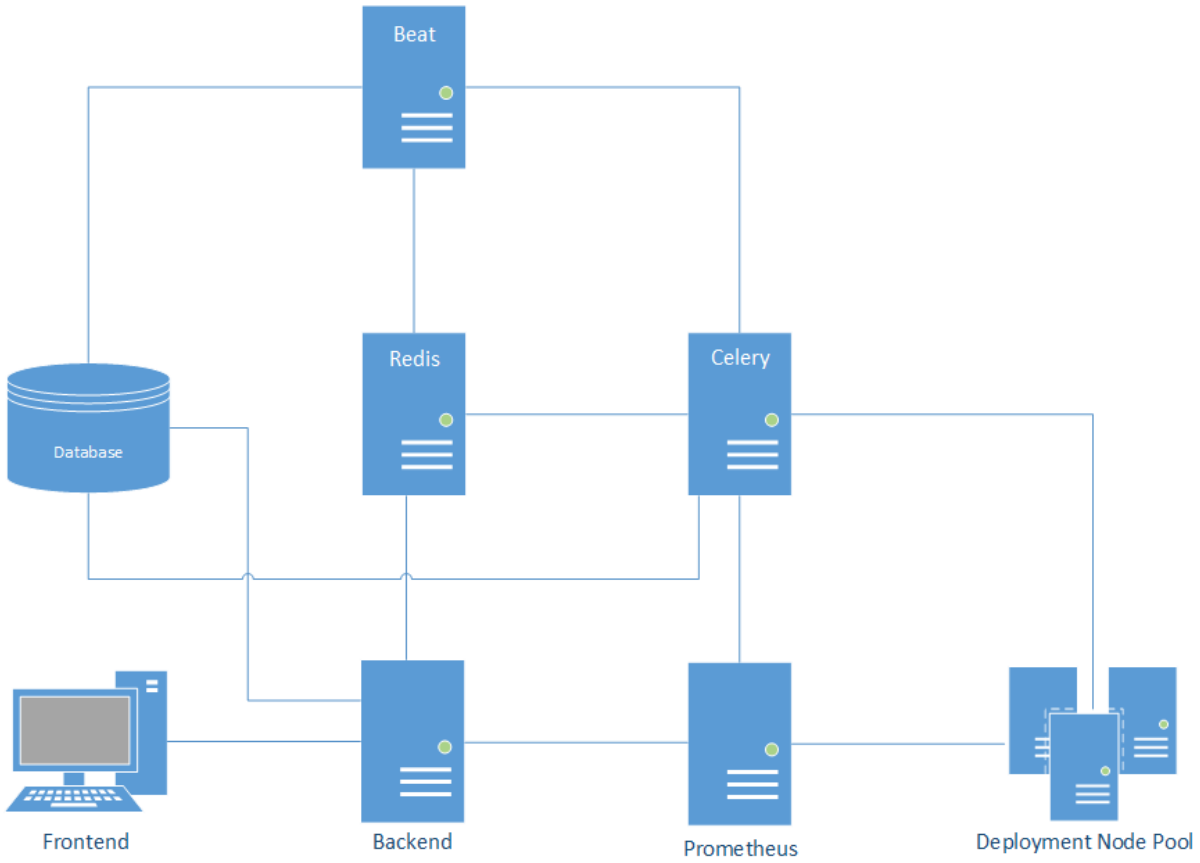

Abbildung 1: Technischer Kontext - Vereinfachte Darstellung

3.2.1 Frontend
Das Frontend ist eine interne React-Applikation, welche mit dem LTB Backend via http(s) über die Django-API kommuniziert.

3.2.2 Backend
Das Backend stellt die Django-API zur Verfügung und enthält alle Funktionen der Business Logik.

3.2.3 Database
Für die Speicherung der Daten wird eine interne Postgres-Datenbank verwendet. Die Celery Tasks werden in der Datenbank persistiert.

Die interne Postgres-Datenbank wird vom Backend sowie von Beat angesprochen.

3.2.4 Redis
Wird als Message-Broker für die Kommunikation mit Celery verwendet.

3.2.5 Beat
Ist ein Task Scheduler der die Celery-Tasks anstösst.

3.2.6 Celery
Task-Queue für das Ausführen der effektiven Deployments von Labs auf den Nodes im Deployment Node Pool.

3.2.7 Prometheus
Monitoring-System, das für die Überwachung der Auslastung der Deployment Nodes im Pool verwendet wird.

3.2.8 Deployment Node Pool
Alle Instanzen, die für ein Deployment genutzt werden können.

4 Lösungsstrategie

Dieser Abschnitt enthält die Definitionen für die Arbeitsweisen und eine grobe Übersicht der wichtigsten Ziele.

4.1 Projektorganisation

Für die grobe Planung wird der «Rational Unifed Process (RUP)» verwendet. Die Inception Phase ist im Sprint 0 gemacht, die weiteren Phasen Elaboration und Construction werden zeitlich den weiteren Sprints zugeordnet. Die Transition Phase wird mehrheitlich von der INS übernommen.

Die konkrete Planung wird Iterativ mit dem Projektmanagement Framework Scrum gemacht. Die Projektplanung wird ab Sprint 1 jeweils in 2-Wochen Sprints stattfinden. Dabei werden pro Sprint die zu erreichende Ziele festgehalten.

- **Product Owner:** Ist äquivalent zum Auftraggeber, wobei das Verwalten der Backlog-Items vom Projektteam übernommen wird.
- **Scrum Master:** Ist nicht vorhanden.
- **Developers:** Ist äquivalent zum Projektteam.

Die Arbeitspakete sind im Jira Tool mit folgenden Kategorien erfasst:

- **Epics:** Ohne Schätzung.
- **User Stories:** Schätzung mit Storypoints anhand der Fibonacci-Zahlen.
- **Tasks:** Schätzung auf Stundenbasis.

Für jeden Sprint wird ein Scrumboard mit 4 Swimlanes geführt.

- **To Do:** Arbeiten die noch zu erledigen sind.
- **In Progress:** Arbeiten die gerade erledigt werden.
- **Review:** Erledigte Arbeiten die von einer zweiten Person überprüft wird.
- **Done:** Erledigte Arbeiten.

4.1.1 Meetings Organisation / Timeboxing

Auf Daily Stand-Up Meetings wurde verzichtet, da die Mitglieder nicht täglich an dem Projekt arbeiteten.

Das Team hat sich jeweils jeden Mittwoch von 15:00 bis 16:00 Uhr getroffen. Da mit einem 2-Wochen Sprint-Rhythmus gearbeitet wurde, war folgerichtig jedes zweite ein Sprintmeeting mit den folgenden Traktanden:

- **Review:** Kurze Präsentation über das Erreichte im letzten Sprint.
- **Retrospektive:** Einholen gegenseitiger Feedbacks für Verbesserungen.
- **Planning:** Planung und Priorisierung des nächsten Sprints.
- **Fragen:** Die restliche Zeit wurde für offene Fragen genutzt.

Während eines Sprints diente diese Besprechung primär zur Klärung offenen Fragen.

Diese Meetings wurden jeweils auf 1 Stunde limitiert. Es wurde bei allen Meetings darauf geachtet, dass das geplanten Zeitlimit nicht überschritten wurde.

Aufgrund der grossen Komplexität und der diversen Ansatzlösungen waren mehrere Meetings ausserhalb der regulären Meetings notwendig. Diese wurden mit einer eigenen Traktandenliste und einer angepassten Zeitlimitierung gehalten.

In allen Meetings wurde Protokoll geführt. Dazu wurde das OneNote im Microsoft Teams Channel «LTB Organizer» verwendet.

4.1.2 Grobplanung

RUP	Sprint	Ziel	Estimate	Done
Inception	**Sprint 0**	Projektplan, Ready for first sprint planning	-	-
	Sprint 1	Konzept für Kosten der Labs und Kapazitäten der Nodes	10	14
Elaboration	**Sprint 2**	Implementierung des Analytics Board und des Deployment Jobs	16	24
	Sprint 3	Erster Entwurf Reservation erstellen und bereitstellen	26	26
	Sprint 4	Reservationen fertigstellen	23	23
Construction	**Sprint 5**	Production ready	16	16
	Sprint 6	Abschluss und Dokumentation der Arbeit	-	-

Tabelle 4: Grobe Übersicht der Sprintziele

Um das Vorgehen während der Planung und Umsetzung zu unterstreichen, wurde der weitere Verlauf dieses Dokumentes in die erarbeiteten Epics unterteilt.

Folgende Epics wurden identifiziert:

- Einarbeitung Lab Topology Builder (LTB)
- Ressourcenbasiertes Deployment
- Labs reservieren
- Priorisierung / Credit System

Das Epic «Priorisierung / Credit System» wurde in dieser Arbeit nicht umgesetzt. Aufgrund des Zeitaufwands und der entsprechenden Priorisierung während der Arbeit wurde auf eine Erarbeitung des Epics verzichtet, da dies den Zeitrahmen klar überschritten hätte.

5 Konzepte

In diesem Abschnitt werden allgemeine, systemweite Strukturen und Aspekte beschrieben.

5.1 Domain Model

Im Domain Model wird ausschliesslich auf die Änderungen bezüglich der Umsetzung des LTB Organizers eingegangen. Es wird bewusst darauf verzichtet, die komplette, existierende Domain abzubilden.

Bereits existierende Entitäten werden abgebildet, wenn sie für den Kontext dieses Projekts relevant sind. Grau hinterlegte Felder repräsentieren existierende Felder in der aktuellen LTB Domain. Die Felder in schwarz sind dementsprechend neu und wurden im Kontext dieser Arbeit umgesetzt.

Anschliessend wird die grobe Übersicht über die Problem-Domain noch textuell beschrieben.

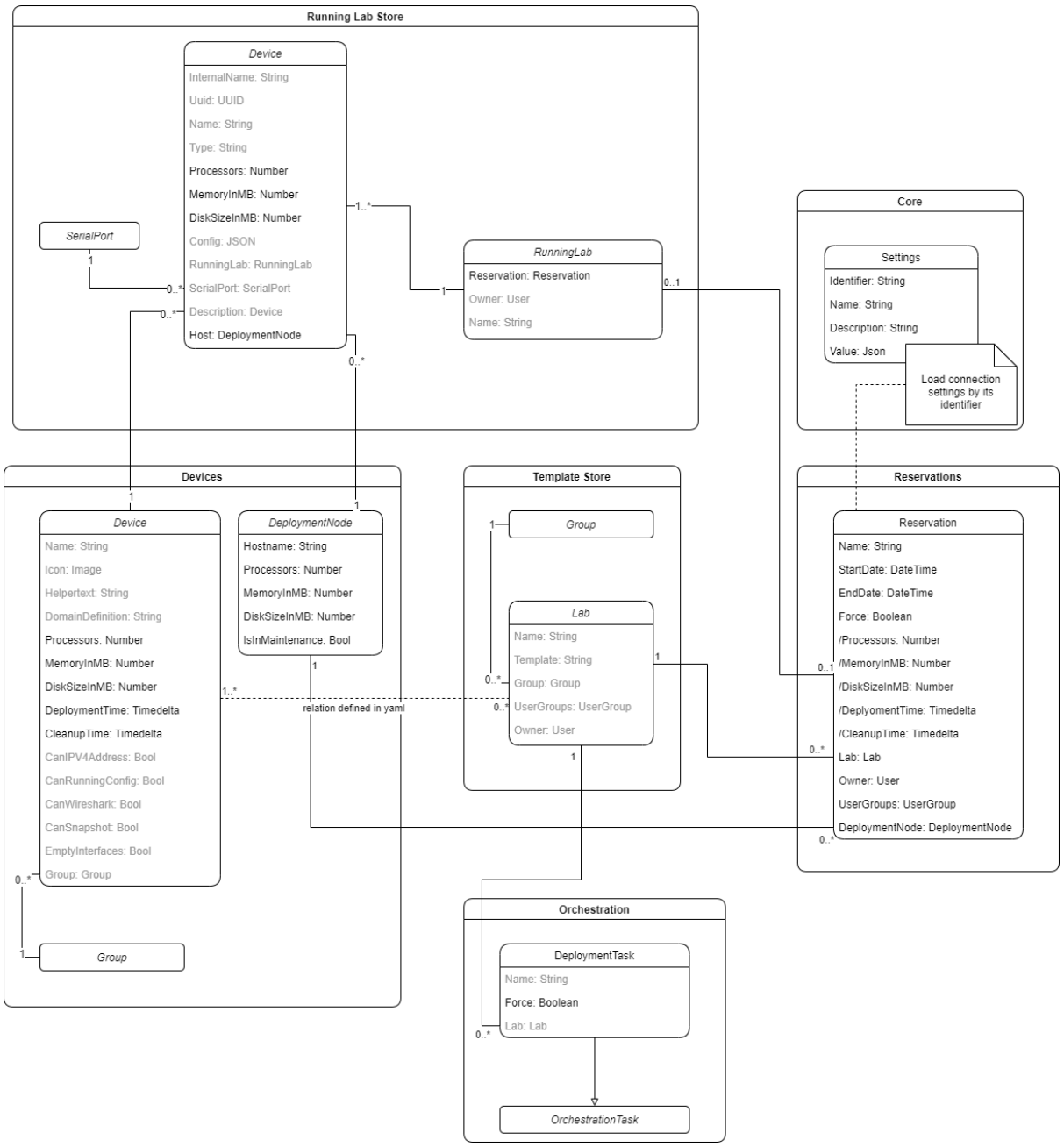

Abbildung 2: Domain Model - LTB Organizer

5.1.1 Running Lab Store

5.1.1.1 Device

Werte für die Kosten, Deployment- und Cleanup-Time werden auf dem "Running Device" absichtlich dupliziert. Es muss sichergestellt werden, dass Änderungen am ursprünglichen Device keinen Einfluss auf die aktuell laufenden Ressourcen haben.

Processors	Anzahl Prozessoren, die das Device benutzen kann
Memory	Menge an Arbeitsspeicher, die das Device benutzen kann in Megabytes
DiskSize	Platz auf der Disk die vom Device benutzt wird in Megabytes
Host	Deployment Node auf dem das Device läuft

Tabelle 5: Felder Running Device

5.1.1.2 Running Lab

Reservation	Reservation, die für die das Lab erstellt wurden

Tabelle 6: Felder Running Lab

5.1.2 Devices

5.1.2.1 Device

Processors	Anzahl Prozessoren, die das Device benutzen kann
Memory	Menge an Arbeitsspeicher, die das Device benutzen kann in Megabytes
DiskSize	Platz auf der Disk die vom Device benutzt wird in Megabytes
DeploymentTime	Notwendige Zeit, um dieses Device zu deployen
CleanupTime	Notwendige Zeit, um dieses Device zu entfernen

Tabelle 7: Felder Device

5.1.2.2 Deployment Node

Hostname	Hostname des Servers
Processors	Verfügbare Anzahl Prozessoren für das Deployment
Memory	Verfügbare Menge an Arbeitsspeicher für Deployments in Megabytes
DiskSize	Verfügbarer Platz auf der Disk für Deployments in Megabytes
IsInMaintenance	Gibt an, ob der Server sich im "Wartungsmodus" befindet und entsprechend für Deployments gesperrt ist

Tabelle 8: Felder Deployment Node

5.1.3 Reservations

5.1.3.1 Reservation

Name	Name für die Reservation
StartDate	Startdatum mit der Zeit, ab wann das Lab zur Verfügung stehen muss
EndDate	Enddatum, bis wann das Lab zur Verfügung stehen muss
Force	Gibt an, ob die Reservation forciert wurde
Processors	Summierte Anzahl Prozessoren
MemoryInMB	Summiertes Memory in MB aller Devices
DiskSizeInMB	Summierte Disk Size in MB aller Devices
DeploymentTime	Summierte Deployment Time aller Devices und Connections (via Settings)
CleanupTime	Summierte Cleanup Time aller Devices und Connections (via Settings)
Lab	Verknüpftes Lab
DeploymentNode	Node welcher für das Deployment der Reservation vorgesehen ist
Owner	Ersteller der Reservation
UserGroups	Benutzergruppen

Tabelle 9: Felder Reservation

5.1.4 Orchestration

5.1.4.1 Deployment Task

Reservation	Verknüpfte Reservation
Force	Gibt an, ob das Deployment forciert werden soll

Tabelle 10: Felder Deployment Task

5.1.5 Core

5.1.5.1 Settings

Identifier	Identifier der Konfiguration
Name	Lesbarer Name der Settings
Description	Kurzbeschreibung für was dieses Setting gebraucht wird
Value	Effektiver Value in JSON-Format

Tabelle 11: Felder Settings

5.2 Testbarkeit

5.2.1 Frontend

Für das Frontend existieren keine funktionalen Tests, lediglich ein Style-Checker, welcher bei jedem Merge-Request ausgeführt wird.

Das Frontend wird daher weiterhin nur vom Entwickler manuell getestet, oder das Testing findet im Rahmen des Systemtestings statt.

5.2.2 Backend

Im Backend existieren bereits diverse Unit Tests, mit einer sehr hohen Code Coverage. Zudem ist der Deployment Prozess mit den Unit Tests relativ einfach und zeitsparend testbar.

Der Test-Driven-Deployment eignet sich daher sehr gut für die Realisierung des LTB Organizers.

6 Einarbeitung Lab Topology Builder (LTB)

Der LTB Organizer ist eine Erweiterung, die in das bestehende Projekt «Lab Topology Builder (LTB)» implementiert wird. Daher ist eine Einarbeitung in das bestehende Projekt erforderlich.

Das Frontend ist eine React-Applikation, die mit dem Backend über die entsprechende API, als auch über WebSockets, kommuniziert.

Das Backend wird mittels folgenden Docker-Container bereitgestellt.

- **db** Postgres Datenbank
- **pgadmin** Web UI Management Tool für die Entwicklung
- **redis** Database scheduler (Update channels mit Notifications)
- **backend** Django-API und Business Logic
- **celery** Für das Management von Running-Tasks, bez. Labs
- **beat** Scheduler
- **ngnix** Load Balancer/Reverse Proxy

Für dieses Epic sind keine Requirements definiert, da die LTB Applikation bereits existiert. Nachfolgend werden aber die verschiedenen Probleme beim Aufsetzen der lokalen Entwicklungsumgebung und deren Lösungen beschrieben.

6.1 Docker mit Linux

Das Deployment der Docker-Container auf einer Windows Distribution ist fehlerhaft und kann nicht genutzt werden. Somit kann das herkömmliche Windows nicht für die Entwicklung verwendet werden.

Für das lokale Entwickeln muss eine Linux Distribution verwendet werden, wobei auch die «Windows Subsystem for Linux (WSL)» nutzbar ist.

6.2 Deployment Node Mocking

Ein Deployment eines Labs kann einige Minuten in Anspruch nehmen und ein für die Entwicklung nutzbarer Deployment Node existiert nicht. Ausserdem kann das Deployment eines Labs in der bestehenden LTB Applikation auch nicht gemocked werden.

Für die lokale Entwicklung muss das Deployment im Source Code auskommentiert werden, dabei ist auch zu beachten, dass Prometheus nur im internen INS Netz zur Verfügung steht und entsprechend ebenfalls auskommentiert/gemocked werden muss.

7 Ressourcenbasiertes Deployment

Ziel des ressourcenbasierten Deployments ist die Verwaltung der Kosten von Labs und die Verwaltung der verfügbaren Ressourcen von Deployment Nodes. Dazu sollen die Kosten für ein Lab, sowie für die verfügbaren Ressourcen der Deployment Nodes, hinterlegt werden. Diese Informationen sollen beim Deployment genutzt werden, um die richtigen Deployment Nodes auszusuchen und eine Überbelastung eben dieser zu verhindern.

7.1 Analyse Problemstellung

In diesem Kapitel wird auf einige Besonderheiten bezüglich dem ressourcenbasierten Deployment eingegangen, insbesondere wird darauf eingegangen, wie die Berechnung der Kosten einheitlich gemacht werden kann.

7.1.1 Metriken

Für die Berechnung der Kosten anhand der gegebenen Ressourcen müssen Metriken definiert werden, die die Auslastung eines Deployment Nodes abbilden können. Dazu verwenden wird die drei Metriken: CPU, Memory und Disk Size.

7.1.1.1 CPU

Die Metrik CPU ist durch die Anzahl an verfügbaren oder benutzten logischen Prozessoren definiert.

Bei Docker entspricht die Angabe der CPUs den logischen Prozessor-Ressourcen, die der Container verwenden darf. Somit kann diese Angabe als absoluter Wert behandelt werden.

Für KVM gehen wir von der generellen Schätzung aus, dass 1 vCPU = 1 logischem Prozessor entspricht. Jedoch ist eine vCPU mächtiger als eine CPU/Core, gerade wenn mehrere Cores verfügbar sind.

7.1.1.2 Memory

Für das Memory verwenden wir eine Angabe in Megabytes.

Für Docker und KVM kann das Memory entsprechend ebenfalls mit einer Angabe in Bytes limitiert werden.

7.1.1.3 Disk

Für die Metrik Disk verwenden wir eine Angabe vom verfügbaren oder verwendeten Platz in Megabyte.

Für Docker kann die Grösse des Containers unter gewissen Vorbedingungen limitiert werden.

Für KVM ist der verwendete Platz durch die Grösse des Images / Disk der VM gegeben.

7.1.2 Berechnung

Grundsätzlich sind die Berechnungen sehr einfach gehalten und für alle Metriken äquivalent. Die Metrik Prozessoren in den folgenden Beispielen können auch durch die Metrik Memory oder Disk Size ersetzt werden.

Prüfen, ob genügend verfügbare Ressourcen für ein Deployment vorhanden sind:

$$Vorhandene\ Prozessoren\ -\ Verwendete\ Prozessoren\ >=\ Angeforderte\ Prozessoren$$

Prozentuale Auslastung:

$$Auslastung\ in\ \%\ =\ \frac{Verwendete\ Prozessoren}{Vorhandene\ Prozessoren} * 100$$

7.1.2.1 Overcommitment

Gewisse Ressourcen können überbeansprucht werden. Beispielsweise ist nicht jeder Container zu jeder Zeit zu 100% ausgelastet und nimmt sein zugewiesenes Memory und Prozessoren nicht voll in Anspruch. Daher kann es sich lohnen, etwas mehr Ressourcen zuzuweisen, als dass diese vorhanden sind.

Eine mögliche Lösung könnte sein, einen gewissen Prozentsatz festzulegen, bis zu welchem Grad eine Ressource überbeansprucht werden darf. Dies kann allgemein oder pro Node festgelegt werden. Andererseits wäre es bereits möglich, dies bei der Definition der verfügbaren Ressourcen einzurechnen.

7.1.2.2 Safety Zone

Dies entspricht dem genauen Gegenteil von Overcommitment. Bestehende und grundlegende Prozesse auf den Nodes sollen nicht durch zusätzliche Ressourcen gestört oder unbrauchbar gemacht werden. Beispielsweise darf die Disk nicht zu 100% ausgelastet werden.

Um dies sicherzustellen, müsste eine gewisse Reserve eingerechnet werden, welche sicherstellt, dass der Node an sich reibungslos funktionieren kann. Eine mögliche Lösung könnte sein, einen gewissen Prozentsatz festzulegen, bis zu welchem Grad eine Ressource überbeansprucht werden darf. Dies kann allgemein oder pro Node festgelegt werden. Andererseits wäre es bereits möglich, dies bei der Definition der verfügbaren Ressourcen einzurechnen.

(Docker Inc., 2021) (Docker Inc., 2021) (Hans, 2021) (Hyve, 2021) (Red Hat Inc., 2021) (Red Hat Inc., 2021)

7.2 Requirements

Die Requirements wurden in Form von Use-Cases definiert. Nachfolgend sind alle Use-Cases, welche in Zusammenhang mit dem Epic «Ressourcenbasiertes Deployment» erarbeitet wurden, beschrieben.

7.2.1 Use-Cases «Kosten pro Lab speichern»

Create Device

Brief description	Der Actor erstell ein neues Device mit dessen Kosten*.
Actors	Admin
Preconditions	Der Actor ist im LTB-Backend eingeloggt.
Basic flow of events	Devices sollen wie gehabt erstellt werden können. Zusätzlich sollen neu die Felder für die Kosten*, CPUs, Memory und Disk Size erfasst werden können.
Extensions	
Postconditions	Das Device wurde erfasst, zusammen mit den Kosten*.
Special requirements	

Tabelle 12: Use-Case Create Device

Read Device

Brief description	Der Actor sieht die Kosten* eines Devices ein.
Actors	Admin
Preconditions	Der Actor ist im LTB-Backend eingeloggt.
Basic flow of events	Beim Einsehen von einem Device sollen neu auch die Felder für die Kosten*, CPUs, Memory und Disk Size eingesehen werden können.
Extensions	
Postconditions	
Special requirements	

Tabelle 13: Use-Case Read Device

Update Device	
Brief description	Der Actor updated die Kosten* eines Devices.
Actors	Admin
Preconditions	Der Actor ist im LTB-Backend eingeloggt.
Basic flow of events	Beim Ändern eines Devices können auch die Felder für die Kosten*, CPUs, Memory und Disk Size aktualisiert werden.
Extensions	Das Device kann mit einem Running-Device verknüpft sein. • Das Running-Device speichert die aktuellen Kosten des Devices beim Deployment.
Postconditions	Die Kosten* des Devices sind aktualisiert.
Special requirements	Das Ändern der Kosten* darf ein laufendes Lab nicht beeinflussen.

Tabelle 14: Use-Case Update Device

7.2.2 Use-Cases «Ressourcen pro Server speichern»

Create Deployment Node	
Brief description	Der Actor erstellt ein Deployment Node.
Actors	Admin
Preconditions	Der Actor ist im LTB-Backend eingeloggt.
Basic flow of events	Deployment Nodes sollen mit den folgenden Feldern erfasst werden können: • (Host) Name • Verfügbare CPU (Cores) • Verfügbares Memory in MB • Verfügbare Disk Size in MB • Is in Maintenance Mode
Extensions	
Postconditions	Ein neuer Deployment Node Eintrag wurde erstellt.
Special requirements	

Tabelle 15: Use-Case Create Deployment Node

Update Deployment Node	
Brief description	Der Actor aktualisiert einen bestehenden Deployment Node
Actors	Admin
Preconditions	Der Actor ist im LTB-Backend eingeloggt.
Basic flow of events	Alle Felder des Deployment Node können geändert werden.
Extensions	
Postconditions	Der Deployment Node ist aktualisiert.
Special requirements	Es kann laufende Labs auf dem vom Deployment Node beschriebenen Server haben. Änderungen der verfügbaren CPU's (Cores), des verfügbaren Memory oder dem verfügbaren Disk Size hat keinen Einfluss auf die laufende Instanz, nur auf zukünftige Deployments.

Tabelle 16: Use-Case Update Deployment Node

Maintenance Mode	
Brief description	Der Actor versetzt den Deployment Node in den Maintenance Mode.
Actors	Admin
Preconditions	Der Actor ist im LTB-Backend eingeloggt.
Basic flow of events	Um Wartungen an einem Node vorzunehmen, soll dieser für Deyploments deaktiviert werden können. Aktuell laufende Labs werden nicht abgeräumt oder verschoben.
Extensions	
Postconditions	Keine weiteren Labs werden mehr auf diesem Deployment Node erstellt.
Special requirements	

Tabelle 17: Use-Case Maintenance Mode

Delete Deployment Node	
Brief description	Der Actor löscht ein Deployment Node.
Actors	Admin
Preconditions	• Der Actor ist im LTB-Backend eingeloggt. • Es darf keine laufenden Labs mehr auf dem Node existieren. • Deployment Node muss im Maintenance Mode sein.
Basic flow of events	Der Deployment Node kann gelöscht werden.
Extensions	
Postconditions	Datenbankeintrag ist gelöscht.
Special requirements	

Tabelle 18: Use-Case Delete Deployment Node

7.2.3 Use-Cases «Deployment anhand von erfassten Kosten»

Deployment anhand Kosten und Ressourcen

Brief description	Lab Deployment anhand Kosten und Ressourcen
Actors	User
Preconditions	• Der Nutzer ist autorisiert • Ein Node mit genügend freien Ressourcen ist vorhanden
Basic flow of events	Bei der Auswahl eines Labs für ein Deployment wird das Lab auf einem Node mit genügend freien Ressourcen bereitgestellt. Dieser Node muss genügend freie Ressourcen für die entsprechenden Kosten des Labs haben. Grundsätzlich wird der Node mit den meisten freien Ressourcen ausgewählt.
Extensions	• Beim Deployment müssen auf den Docker Container Limiten anhand der definierten Kosten gesetzt werden. Mindestens müssen CPUs und Memory begrenzt werden. Zusätzlich könnte Disk Size limitiert werden. • Für VMs sollten vCPUs und Memory anhand der definierten Kosten überschrieben oder hinzugefügt werden.
Postconditions	• Alle Devices des Labs sind auf einem Node bereitgestellt • Docker Container haben entsprechende Limiten • VMs haben die entsprechend definierten CPUs und Memory
Special requirements	Die bereits vorhandene Prüfung, ob genügend Ressourcen anhand der Prometheus-Daten vorhanden sind, soll beigehalten werden.

Tabelle 19: Use-Case Deployment anhand Kosten und Ressourcen

Force-Deployment für Administratoren	
Brief description	Labs deployen, ohne checks, ob genügend Ressourcen vorhanden sind.
Actors	Admin
Preconditions	Der Nutzer ist autorisiert
Basic flow of events	Die Berechnung, ob ein Deployment eines Labs möglich ist, ist von sehr vielen Faktoren abhängig und dadurch sehr komplex. Dies kann dazu führen, dass ein Deployment aufgrund der eingebauten Prüfung nicht mehr ausgeführt werden kann, obwohl ein Node noch genügend Ressourcen hätte. Daher muss ein Administrator ein Deployment eines Labs erzwingen (Force) können.
Extensions	Zusätzlich prüfen, ob durch das Force-Deployment Labs in einer Reservation nicht mehr ausgeführt werden können.
Postconditions	• Alle Devices des Labs sind auf einem Node bereitgestellt • Docker Container haben entsprechende Limiten • VMs haben die entsprechend definierten CPUs und Memory
Special requirements	

Tabelle 20: Use-Case Force-Deployment für Administratoren

Analytics Board	
Brief description	Übersicht der zurzeit verwendeten Ressourcen
Actors	Admin
Preconditions	Der Nutzer ist autorisiert
Basic flow of events	Dem Actor wird eine Dashboard-Übersicht mit den CPU, Memory, Disk Metriken pro Device dargestellt. Einerseits werden die vom Backend berechneten Daten dazu verwendet, andererseits werden die Live-Daten von Prometheus als Referenzdaten geladen.
Extensions	Daten anhand eines Tickers (Bsp. 5 Sekunden) aktualisieren.
Postconditions	
Special requirements	

Tabelle 21: Use-Case Analytics Board

7.3 Aspekte der Umsetzung

Nachfolgend wird auf einige Aspekte der Umsetzung näher eingegangen. Dies soll vor allem einen Einblick darin geben, wie die Logik der einzelnen Berechnungen aufgebaut ist.

7.3.1 Kosten für ein Lab

Die Kosten werden auf der Stufe eines Devices gespeichert, somit muss für die Berechnung der Kosten eines Labs über alle Devices iteriert werden. Die berechneten Kosten werden hingegen auf der Stufe der Reservation benötigt.

Ein Lab im Template Store hat zudem keine Datenbankrelation zu seinen Devices und Connections, sondern lediglich ein YAML-File, welches diese Informationen speichert (siehe dazu auch die Abbildung des Domain Models). Für die Berechnung muss also jeweils das YAML-File eingelesen und alle Devices müssen geladen werden.

Um die Performance der Applikation beizubehalten und um die Berechnung so einfach wie möglich zu halten, speichert die Reservation die aggregierten Kosten seines Labs. Damit lösen sich die folgenden Probleme:

- YAML-File und die Devices müssen nicht immer geladen und iteriert werden.
- Berechnungen auf Reservationsstufe müssen keine Unterscheidungen zwischen Devices und Connections machen.
- Berechnungen auf Reservationsstufe können die Relation zu den Labs vernachlässigen
- Reservation speichern die Kosten, die beim Zeitpunkt des Reservierens gültig waren.

7.4 Deployment validieren

Zusätzlich zur herkömmlichen Validierung, dass ein Lab nur dann deployed werden darf, wenn die CPU-Auslastung gemäss den Prometheus Date weniger als 80% ist, wird neu auch gegen die gespeicherten Ressourcen der Deployment Nodes validiert.

Für die Prüfung, ob ein Lab deployed werden kann, müssen alle Deployment Nodes betrachtet werden. Ein Lab kann nur dann nicht deployed werden, wenn kein einziger Deployment Node mehr freie Ressourcen hat.

Dazu wurde in einer ersten Version die zurzeit verwendeten Ressourcen der Deployment Nodes geladen und summiert. Diese, addiert mit den Kosten des zu deployenden Labs, müssen kleiner sein als die verfügbaren Ressourcen aller Deployment Nodes, damit das Deployment eines Labs gültig ist.

In einem zweiten Schritt wurde diese Validierung nicht mehr über alle, sondern über die einzelnen Deployment Nodes, gemacht.

7.5 Limitierung der Devices

Standardmässig würde ein Docker Container so viele Ressourcen in Anspruch nehmen, wie dies möglich ist. Durch die Definition der Ressourcen auf den Devices soll dies jedoch für VM's, als auch für Docker Container, limitiert werden.

Bei Docker Container wird dies durch die Angabe von «CPU Period» und «CPU Quota» in Bezug auf Prozessorleistung und mit der Angabe von «Memory» in Bezug auf Arbeitsspeicher erreicht. Die Disk Size kann nur schwierig und unter gewissen Rahmenbedingungen limitiert werden, wobei momentan bei der Device Definition die Grösse des Containers und des Images in Betracht gezogen werden sollten.

Die VM's werden anhand einer «Domain Definition» erstellt. Die entsprechenden Metriken werden dabei als Platzhalter zur Verfügung gestellt, wodurch die Limitierung innerhalb der «Domain Definition» vorgenommen werden kann.

8 Labs reservieren

Für die Reservationen sind die Kosten der Labs und die verfügbaren Ressourcen relevant. Eine Reservation darf nicht erstellt werden, wenn weniger Ressourcen zur Verfügung stehen als ein Lab kostet. Diese Kosten entsprechen den Metriken der Ressourcenberechnung.

Für die Reservationen müssen zusätzlich Deployment- und Cleanupzeiten berücksichtigt werden. Also die Zeit, welche ein Lab für das Bereitstellen aller enthaltenen Devices benötigt, sowie die Zeit, die für das Entfernen dieser erforderlich ist.

- Metriken der Ressourcenberechnung
- Deploymentzeit
- Cleanupzeit

8.1 Analyse Problemstellungen

Im Prinzip gäbe es verschiedene Möglichkeiten, zu welchen Zeitpunkten eine Reservation gestartet oder auch beendet werden kann. Diese zwei zu unterscheidenden Ansätze wurden hier beschrieben:

- Die freie Definition von Start- und Endzeitpunkten für eine Reservation durch den Erstellenden
- Festgelegte Start- und Endzeitpunkte für Reservationen durch die Administrierenden (Pufferzeiten)

8.1.1 Freie Definition der Laufzeit

Zur Beschreibung dieser Variante ist ein zeitlicher Verlauf eines Reservationsplans dargestellt. Als Ausgangslage für die verwendeten Labs kann eine Deploymentzeit von 20 Minuten und eine Cleanupzeit von 10 Minuten pro Lab angenommen werden.

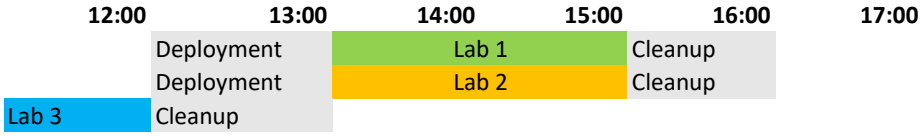

Abbildung 3: Ausgangslage freie Definition Laufzeiten

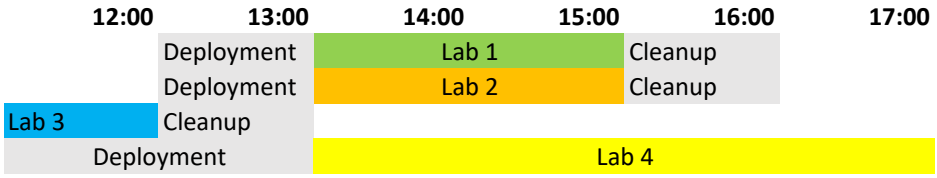

Abbildung 4: Freie Definition Laufzeiten mit zusätzlichem Lab

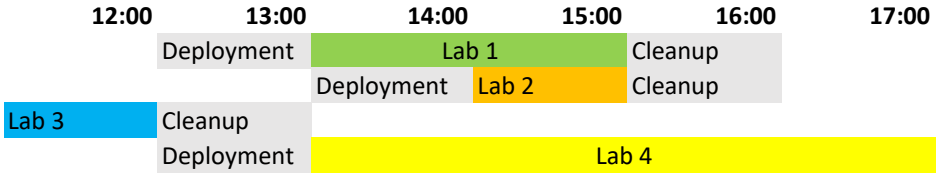

Abbildung 5: Freie Definition Laufzeiten mit Änderung

Als Ausgangslage dienen hier zwei Reservation von Labs (Lab 1 & 2) die zum selben Zeitpunkt starten und eine zusätzliche Reservation (Lab 3), dessen Cleanup an die Bereitstellung der neuen Reservationen angrenzt. In einem nächsten Schritt wird eine zusätzliche Reservation (Lab 4) erstellt. Diese hat dieselbe gewünschte Startzeit wie die beiden ersten Reservationen. Im letzten Schritt wird der Startzeitpunkt der Reservation «Lab 2» verschoben.

Dieser Ansatz hat diverse Herausforderungen:

- Es gibt keinen fest definierten Deploymentzeitpunkt. Potenziell müsste jede Minute geprüft werden, ob das Deployment oder der Cleanup für eine Reservation gestartet werden muss. Um den Start des Deployments oder des Cleanups zu errechnen, muss immer der Reservationsplan als Ganzes berücksichtigt werden.
- Eine Anpassung einer Reservation (Lab 2) hat Auswirkungen auf den ganzen Reservationsplan. In diesem Fall verschiebt sich der Deploymentstart nach vorne, da nun genügend Zeit zur Verfügung steht.
- Die Komplexität zur Berechnung der verfügbaren Ressourcen steigt massiv, da eine Reservation bereits vor dem geplanten Start Ressourcen in Anspruch nimmt. Nämlich während und nach dem effektiven Deployment. Im Beispiel von Lab 4 wäre dies bereits ca. 1 Stunde vor dem geplanten Start.
- Propagierung von Fehlern, wenn Deployments länger dauern.

8.1.2 Eingeschränkte Definition der Laufzeit

Bei der eingeschränkten Definition der Laufzeit sind die Start- und Endzeitpunkte einer Reservation fixiert. Ersteller der Reservation wählen ihre Laufzeit auf Tagesbasis, bzw. auf Basis der Daten.

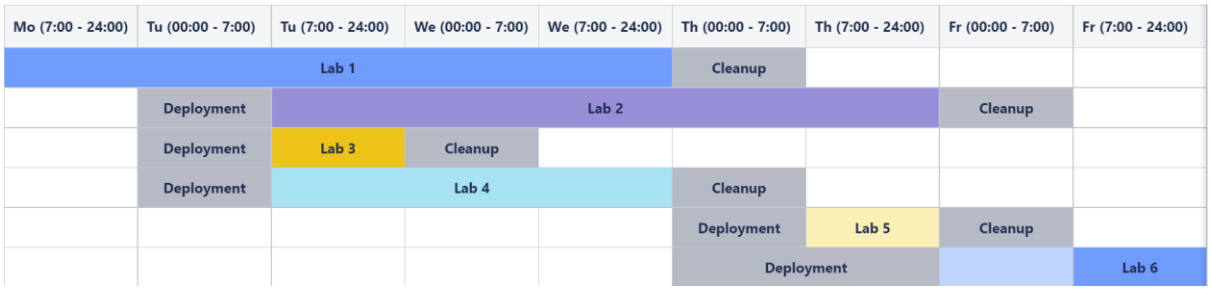

Abbildung 6: Eingeschränkte Definition der Laufzeit

Vorteile gegenüber der freien Definition der Laufzeiten:

- Die Administrierenden können die Start- und Endzeiten für Reservationen definieren. Dadurch können Deploymens den laufenden Betrieb weniger stark beeinflussen oder sogar stören.
- Die Deployments sind in Zeitfenster pro «Tag» aufgeteilt. Dadurch wird die Berechnung des Reservationsplans stark vereinfacht.
- Das Deployment kann einmal pro Tag zu einem fix definierten Zeitpunkt stattfinden. Dabei können zuerst alle Labs abgelaufener Reservationen entfernt werden, danach können die anstehenden Labs des kommenden Tages eingeplant werden.
- Das Vorgehen garantiert, dass auch während dem Deployment nicht mehr Ressourcen in Anspruch genommen werden, als dass diese verfügbar sind. Dabei können für die Berechnung der Ressourcen die geplanten Start- und Endzeitpunkte der Reservationen verwendet werden.
- Anpassungen von Reservationen, sowie auch Fehler bei Deployments welche länger andauern als definiert, sind auf das Deploymentfenster von jedem einzelnen Tag beschränkt.

Herausforderungen:

- Labs, welche an sich die vorgegebenen Deploymentzeiten überschreiten, müssten einen «Slot» früher bereitgestellt werden (Siehe Lab 6).
- Durch den Cleanup einer neuen Reservation kann es vorkommen, dass bestehende Reservationen später starten. Im obigen Beispiel könnte dies simuliert werden, indem eine neue Reservation am Montag enden würde. Entsprechend würden die Deployments der bestehenden Reservationen vom Dienstag später starten.

8.1.3 Granularität

Für die Reservationen können verschiedene Granularitäten/Zeiteinheiten gewählt werden. Beispielsweise könnten diese auf Tage, Stunden oder Minuten genau getätigt werden. Je kleiner die Einheiten gewählt werden, desto schwieriger wird die Planung. Für die aktuellen Verwendungszwecke ist die Einheit **Tage** völlig ausreichend.

8.1.4 Deploymentzeit / Cleanupzeit

Die Deploymentzeit und entsprechend auch die Cleanupzeit eines Labs errechnet sich aus der Summe der jeweiligen Zeiten der im Lab enthaltenen Devices. Die effektive Startzeit errechnet sich aus den bestehenden Deployments/Cleanups anderer Reservation, sowie der Deploymentzeit des Labs.

Wichtig ist auch, dass bei einem Lab die Deploymentzeiten der Connections mitberechnet werden müssen. Diese müssen entsprechend separat konfiguriert werden, da diese selbst als Device nicht existieren.

8.1.5 Paralleles Deployment

Momentan werden die Labs sequenziell bereitgestellt. Für die Reservationsplanung würde es Sinn machen, wenn Labs parallel bereitgestellt werden könnten. Momentan besteht die Problematik, dass ein Lab mit 6 Stunden Deploymentzeit viele andere Labs blockieren würde, die entsprechend in diesem Zeitraum nicht eingeplant werden können. Mit einem parallelen Deployment könnte diese Problematik etwas entschärft werden. Die Schwierigkeit des parallelen Deployments liegt in der Auslastung der einzelnen Nodes. Aus der bisherigen Erfahrung zeigt sich, dass es Probleme geben kann, wenn zu viele VM's parallel auf einem Node bereitgestellt werden und zu fehlerhaften Deployments führen kann.

Bisher wäre es möglich, dass die Deployments auf **unterschiedlichen Nodes** parallel ohne Probleme laufen könnten. Dadurch würde die Problematik der «blockierenden» Nodes bereits etwas entschärft werden. Im Weiteren könnten die Deployments weiter optimiert werden, indem ein Lab mit einer sehr langen Deploymentdauer in Bezug auf den Deployment Node von Labs mit einer kurzen Deploymentdauer getrennt wird.

Das parallele Deployment wurde für die Umsetzung in dieser Arbeit in Betracht gezogen, wobei gegen die Umsetzung entschieden wurde. Der Mehrwert der parallelen Deployments ist klar ersichtlich, jedoch ist die Komplexität der Umsetzung sehr gross. In der aktuellen Implementierung wird das sequenzielle Deployment durch die Anzahl der «Celery Worker» – momentan 1 Worker – forciert. Die Steuerung der Parallelität müsste somit in der Anwendung selbst implementiert werden, wobei dies eine massive Überarbeitung der bestehenden Logik für das Scheduling der Deployments erfordert.

8.2 Konzept Reservationsplan

Im Projekt wurde die Variante «Eingeschränkte Definition der Laufzeit» umgesetzt. Betreffend Granularität wurde definiert, dass eine Laufzeit einer Reservation immer in ganzen Tagen angegeben wird und entsprechend auf eine komplett dynamische Angabe der Laufzeit verzichtet werden kann. Besonders für die aktuelle Verwendung überwiegen die Vorteile dieser Variante stark gegenüber der Variante mit einer komplett freien Definition. Folgende Grundsätze wurden für das gewählte Konzept definiert:

- Eine Reservation beginnt um 7:00 Uhr (lokale Zeit)
- Das Ende einer Reservation ist um 00:00 (lokale Zeit)
- Die Reservationszeiten müssen konfigurierbar sein
- Über den ganzen Zeitraum einer Reservation müssen genügend freie Ressourcen vorhanden sein
- Eine Reservation kann nur an einem Datum starten, wenn die effektive Startzeit der Reservation noch vor der geplanten Startzeit (7:00 Uhr) ist
- Ein Lab mit einer längeren Deploymentzeit als 7 Stunden kann entsprechend nicht breitgestellt werden
- Enddaten werden nie blockiert. Das Risiko, dass eventuell der Start von Reservationen leicht verschoben wird, kann akzeptiert werden. Dieser Fall sollte äusserst selten auftreten. Zu beachten ist auch, dass die Ressourcen in Hinsicht auf Kosten und der Deploymentzeit in diesem Fall am Limit sind.
- Wenn das Startdatum einer Reservation dem Tag der Erstellung entspricht, soll die Reservation direkt bereitgestellt werden. Die Reservation darf nur dann für eine sofortige Bereitstellung erstellt werden, wenn diese nicht die bestehende Planung beeinflusst.

8.2.1 Beispielplan

Folgend wird ein Beispiel eines Reservationsplanes für den definierten Node und die entsprechenden Labs dargestellt. Für die Labs 7 und 8s werden die potenziell möglichen Startzeiten dargestellt. Das Lab 6 stellt den Fall eines Labs mit einer längeren Deploymentzeit als verfügbare Pufferzeit dar.

Node:

- Processors: 16
- Memory: 16 GB
- Disk: 100 GB

Lab	Startzeit	Endzeit	Deploymentzeit	Abräumzeit	Kosten (CPU / Memory / Disk)
Lab 1	Mo 7:00	We 24:00	2h	30m	4 / 4 GB / 30 GB
Lab 2	Tu 7:00	Th 24:00	3h	1h	6 / 6 GB / 50 GB
Lab 3	Tu 7:00	Tu 24:00	1h	10m	2 / 2 GB / 10 GB
Lab 4	Tu 7:00	We 24:00	2h	20m	2 / 2 GB / 10 GB
Lab 5	Th 7:00	Th 24:00	1h	10m	2 / 2 GB / 10 GB
Lab 6	Fr 7:00	Sa 24:00	8h	2h	6 / 5 GB / 20 GB
Lab 7			2h	20m	2 / 2 GB / 10 GB
Lab 8			1h	10m	3 / 5 GB / 12 GB

Tabelle 22: Konzept Reservationsplan

Abbildung 7: Konzept Reservationsplan

8.3 Requirements

Die Requirements wurden in Form von Use-Cases definiert. Nachfolgend sind alle Use-Cases, welche in Zusammenhang mit dem Epic «Labs reservieren» erarbeitet wurden, beschrieben.

8.3.1 Use-Cases «Reservationen»

Create Reservation

Brief description	Der Actor erstellt eine Reservation für ein Lab über einen beliebigen, zukünftigen Zeitraum
Actors	User
Preconditions	• Der Nutzer ist autorisiert • Es hat genügend freie Ressourcen über den ganzen Reservationszeitraum • Das Lab kann innerhalb der Deploymentzeiten bereitgestellt werden
Basic flow of events	Der Actor wählt ein Lab aus, für das er eine Reservation tätigen möchte. Dem Actor sollten die entsprechend freien Termine für eine Reservation angezeigt werden, wobei der Actor einen Start und Endzeitpunkt auswählen kann. Die Mindestlaufzeit für ein Lab beträgt einen Tag.
Extensions	• Ein Administrator kann die Reservation forcieren und die entsprechenden Prüfungen übersteuern. • Als Startzeitpunkt kann der heutige Tag gewählt werden, wobei das Deployment gleich bei der Erstellung der Reservation getätigt werden soll. Der Startzeitpunkt ist dabei die Zeit, welche für das Deployment ab dem aktuellen Zeitpunkt gebraucht wird.
Postconditions	Die Reservation wurde erfolgreich erstellt
Special requirements	

Tabelle 23: Use-Case Create Reservation

Update Reservation	
Brief description	Der Actor aktualisiert / verschiebt eine Reservation für ein Lab über einen Zeitraum
Actors	User
Preconditions	• Der Nutzer ist autorisiert • Der Actor hat eine gültige Reservation, die keinem Running Lab angehört • Es hat genügend freie Ressourcen über den ganzen Reservationszeitraum • Das Lab kann innerhalb der Deploymentzeiten bereitgestellt werden
Basic flow of events	Der Actor kann eine Reservation verschieben. Dem Actor sollen die entsprechenden freien Termine für eine Reservation angezeigt werden, wobei der Actor einen Start und Endzeitpunkt auswählen kann. Die Mindestlaufzeit für ein Lab beträgt einen Tag.
Extensions	• Für eine Reservation, die bereits gestartet wurde, kann nur der Endzeitpunkt der Reservation bearbeitet werden. Die Reservation kann verkürzt oder, wenn möglich, verlängert werden. • Ein Administrator kann die Reservation forcieren und die entsprechenden Prüfungen übersteuern. • Als Startzeitpunkt kann der heutige Tag gewählt werden, wobei das Deployment gleich bei der Erstellung der Reservation getätigt werden soll. Der Startzeitpunkt ist dabei die Zeit, welche für das Deployment ab dem aktuellen Zeitpunkt gebraucht wird.
Postconditions	Die Reservation wurde erfolgreich aktualisiert
Special requirements	

Tabelle 24: Use-Case Update Reservation

Terminate Reservation	
Brief description	Der Actor beendet eine laufende Reservation
Actors	User
Preconditions	• Der Nutzer ist autorisiert • Der Actor hat eine gültige Reservation, die einem Running Lab angehört
Basic flow of events	Die Reservation kann als gelöscht markiert werden / terminiert werden. Die Endzeit der Reservation wird auf die aktuelle Zeit gesetzt und das Running Lab wird abgeräumt.
Extensions	
Postconditions	• Reservation ist weiterhin vorhanden, wird nicht gelöscht • Running Lab ist gelöscht
Special requirements	

Tabelle 25: Use-Case Terminate Reservation

Delete Reservation	
Brief description	Der Actor löscht eine Reservation
Actors	User
Preconditions	• Der Nutzer ist autorisiert • Der Actor hat eine gültige Reservation, welche noch nicht gestartet wurde
Basic flow of events	Der Nutzer löscht eine Reservation, welche in der Zukunft liegt.
Extensions	
Postconditions	Datenbankeintrag ist gelöscht
Special requirements	

Tabelle 26: Use-Case Delete Reservation

Übersicht Reservation	
Brief description	Der Actor will eine Übersicht über aktuelle Reservationen
Actors	User
Preconditions	Der Nutzer ist autorisiert
Basic flow of events	Der Actor will Reservationen ansehen und bestehende Reservationen für eine Bearbeitung auswählen. Dabei sollen dem Actor alle Reservationen angezeigt werden, auf welche er Zugriff hat.
Extensions	
Postconditions	Der Actor hat eine Übersicht welche Reservationen wann eingeplant sind
Special requirements	

Tabelle 27: Use-Case Übersicht Reservation

8.3.2 Use-Cases «Settings»

CRUD Settings	
Brief description	Der Actor kann Settings-Einträge erstellen, bearbeiten und löschen
Actors	Admin
Preconditions	Der Nutzer ist autorisiert
Basic flow of events	Settings sollen mit den folgenden Feldern erfasst, eingesehen und aktualisiert werden können: • Identifier • Name • Description • Value Zudem können diese Einträge gelöscht werden.
Extensions	
Postconditions	Setting erstellt, aktualisiert, gelöscht
Special requirements	Der Identifier ist eindeutig und wird für die weitere Benutzung hinterlegt

Tabelle 28: Use-Case CRUD Settings

8.3.3 Use-Cases «Deployment»

Cleanup/Deployment Job	
Brief description	Abgelaufene Reservation von Nodes entfernen und neue Reservationen bereitstellen
Actors	Job
Preconditions	
Basic flow of events	Um 00:00 werden diejenigen Running Labs abgeräumt, für welche die Reservationen abgelaufen sind. Nachdem der Cleanup erfolgt ist, werden Reservationen für den anstehenden Tag bereitgestellt.
Extensions	
Postconditions	• Labs mit abgelaufenen Reservationen werden abgeräumt. • Labs mit anstehenden Reservationen für den nächsten Tag werden bereitgestellt
Special requirements	

Tabelle 29: Use-Case Cleanup/Deployment Job

8.3.4 Use-Cases «Devices»

Devices um Zeitkosten erweitern	
Brief description	Devices um Zeitkosten für Deployment und Cleanup erweitern
Actors	Admin
Preconditions	
Basic flow of events	Beim Erstellen und Bearbeiten eines Device sollen neu auch die Deploymentzeit und Zeit für den Cleanup angegeben werden.
Extensions	
Postconditions	Zeiten für Deployment und Cleanup sind erfasst
Special requirements	

Tabelle 30: Use-Case Devices um Zeitkosten erweitern

8.3.5 Use-Cases «Labs»

Bearbeitung von Labs sperren	
Brief description	Labs mit Reservationen dürfen nicht bearbeitet werden
Actors	User
Preconditions	
Basic flow of events	Ein Lab, welches geplante oder aktuelle Reservationen hat, darf nicht bearbeitet oder gelöscht werden. Die entsprechenden Aktionen werden bei diesem Zustand für den Benutzer gesperrt.
Extensions	
Postconditions	Aktionen für die Bearbeitung und Löschung von Labs sind unter den erwähnten Bedingungen gesperrt
Special requirements	

Tabelle 31: Use-Case Bearbeitung von Labs sperren

8.4 Aspekte der Umsetzung

Nachfolgend wird auf einige Aspekte der Umsetzung näher eingegangen. Dies soll vor allem einen Einblick geben, wie die Logik der einzelnen Berechnungen aufgebaut ist.

8.4.1 Reservationen in einem Zeitraum finden

Eine der grundlegendsten Funktionen ist die Abfrage von Reservationen für einen Zeitraum. Dabei müssen folgende Fälle unterschieden werden:

- Reservation endet im angegebenen Zeitraum
- Reservation startet im angegebenen Zeitraum
- Reservation ist komplett im angegebenen Zeitraum enthalten

In der folgenden Abbildung ist dies für einen Abfragezeitraum vom 04.01.2021 bis 07.01.2021 illustriert. Die im Zeitraum enthaltenen Reservationen sind in grün dargestellt, diejenigen ausserhalb sind rot dargestellt.

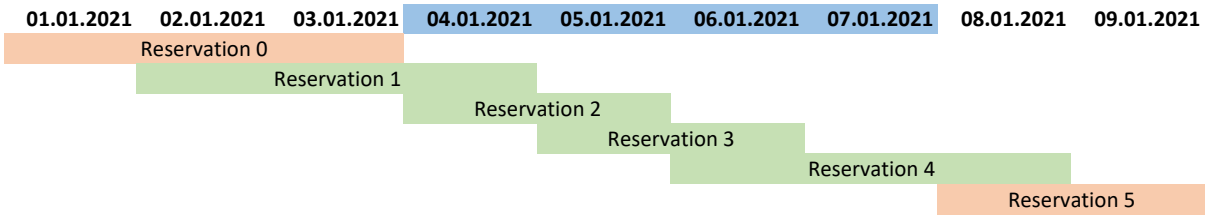

Abbildung 8: Abfrage Reservationen für Zeitraum

8.4.2 Zeitabschnitte für Ressourcenberechnung erstellen

Für die Ressourcenberechnung müssen die Zeiträume der verschiedenen Reservationen weiter unterteilt werden. Ein Zeitraum für die Ressourcenberechnung kann wie folgt definiert werden: Einen Zeitabschnitt, indem Ressourcen gleichermassen verwendet werden. Gleichermassen bedeutet dies, dass immer ein Abschnittswechsel stattfindet, wenn eine neue Reservation beginnt oder eine bestehende endet. Für die Abfrage eines definierten Zeitraums sind nur die Abschnitte relevant, welche sich auch wirklich in dem Zeitraum befinden. Dies ist entsprechend mit blau markiert.

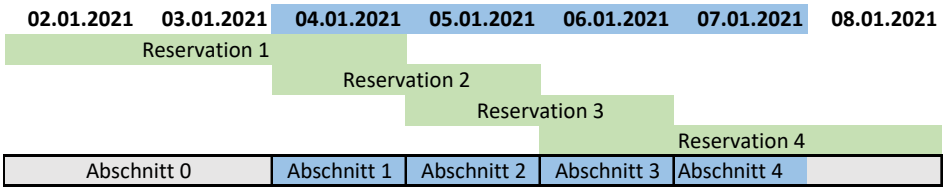

Abbildung 9: Zeitabschnitte Ressourcenberechnung

8.4.3 Geplante Kosten

Um die geplanten Kosten für einen definierten Zeitraum zu erhalten, werden zuerst die bestehenden Zeitabschnitte, wie zuvor erläutert, berechnet. Anhand dieser können die geplanten Kosten konstruiert werden. Jeder Planungseintrag ist durch den Zeitabschnitt, sowie durch die Summe der in diesem Zeitraum verwendeten Ressourcen, definiert. Zudem ist zu beachten, dass die geplanten Kosten immer pro Deployment Node berechnet werden, da nur so effektiv beurteilt werden kann, ob eine Reservation auf einem Node bereitgestellt werden kann.

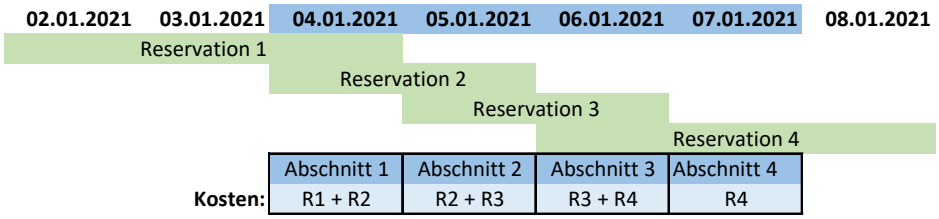

Abbildung 10: Geplante Kosten

8.4.4 Gesperrte Reservationszeiträume

Von Interesse ist nun, welche Zeiträume in einem definierten Zeitabschnitt und für die definierten Kosten gesperrt sind. Anders könnte man auch einfach sagen, in welchem Zeiträumen darf eine neue Reservation nicht laufen, da ansonsten die verfügbaren Ressourcen überschritten werden. Dies ist immer dann der Fall, wenn für einen definierten Zeitabschnitt die Kosten der neuen Reservation, addiert mit den bestehenden Kosten des Zeitabschnitts, die verfügbaren Ressourcen überschreitet. Und dies für alle Deployment Nodes gilt, also entsprechend kein Deployment Node genügend freie Ressourcen für den Zeitabschnitt hat. Die folgenden Abbildungen sollten etwas Klarheit bringen.

	02.01.2021	03.01.2021	04.01.2021	05.01.2021	06.01.2021	07.01.2021	08.01.2021
Node 1	Abschnitt 1	Abschnitt 2			Abschnitt 3		
	Verfügbar	R(N1) - C(A2) < C(Y)			R(N1) - C(A3) < C(Y)		
Node 2	Abschnitt 1			Abschnitt 2	Abschnitt 3		
	R(N2) - C(A1) < C(Y)			Verfügbar	R(N2) - C(A3) < C(Y)		Verfügbar
Node 3		Abschnitt 1			Abschnitt 2		
		R(N3) - C(A1) < C(Y)			R(N3) - C(A2) < C(Y)		

Y = Neue Reservation
R(x) = Verfügbare Ressourcen von x
C(x) = Kosten von x

Abbildung 11: Gesperrte Abschnitte pro Node

In einem ersten Schritt wird pro Deployment Node bestimmt, welche Zeitabschnitte gesperrt sind. Dies ist dann der Fall, wenn die Ressourcen des Nodes *R(NX)*, minus die bestehenden Kosten des Abschnitts *C(AX)*, kleiner sind als die erforderlichen Kosten der neuen Reservation *C(Y)*.

	02.01.2021	03.01.2021	04.01.2021	05.01.2021	06.01.2021	07.01.2021	08.01.2021
Node 1		Gesperrter Abschnitt					
Node 2	Gesperrter Abschnitt				Gesperrter Abschnitt		
Node 3		Gesperrter Abschnitt					
	Verfügbar	Gesperrt		Verfügbar	Gesperrt		Verfügbar

Abbildung 12: Gesperrte Abschnitte Schnittmenge

Aus den gesperrten Abschnitten können dann die Abschnitte konstruiert werden, welche die Schnittmenge aller sich überdeckender Abschnitte bilden. Also die Zeitabschnitte, welche auf jedem verfügbaren Node gesperrt sind.

8.4.5 Auswahl Deployment Node

Wie im vorhergehenden Abschnitt erwähnt werden die geplanten Kosten pro Deployment Node berechnet. Dies bedingt, dass bereits bei der Erstellung der Reservation bekannt ist, welcher Deployment Node die Reservation erhalten wird. Der Node wird anhand der geplanten Kosten ausgesucht, wobei der Node mit den meisten freien Ressourcen ausgesucht wird, welcher die Kosten der Reservation tragen kann. Aktuell wird dies über eine Sortierung gelöst, wobei nach CPU, Memory und dann Disk Size priorisiert wird. In Zukunft könnte man sich überlegen, eine Gewichtung für die Werte zu definieren, damit die Nodes besser ausgelastet sind.

8.4.6 Berechnung Deploymentzeiten

Für die Berechnung der Deploymentzeiten wird unterschieden, ob bereits ein Task für das Deployment erstellt wurde oder ob der Start des Deployments erst geplant ist. Entsprechend gibt es folgende zwei Ausgangslagen für den geplanten Start/Ende:

- Task für Deployment/Cleanup erstellt: Zeit in der das Deployment real geplant wurde, bzw. bei dem der Task mit Celery Scheduled wurde
- Start/Ende rein geplant: Geplanter Start entspricht der Angabe des Settings für Deploymentstartzeit

Grundsätzlich spielt das für die Berechnung keine Rolle, jedoch wurde in der Vergangenheit mit dem dazumal realen Startzeitpunkten gerechnet. Dies ist vor allem für Reservationen relevant, welche am aktuellen Tag gestartet werden und ausserhalb der geplanten Deploymentzeiten liegen. Für die Berechnung in die Zukunft können sich die Deploymentzeiten entsprechend ändern, wenn die Deploymentstartzeit beeinflusst wird.

Für jede Reservation wird ein Zeitraum erstellt, wobei der Beginn dem geplanten Startdatum oder Enddatum entspricht und das Ende dem geplanten Start/Ende der Reservation, addiert mit der Zeit, welche für das Deployment oder Cleanup gebraucht wird. Wenn sich zwei Bereiche überschneiden, bedeutet dies, dass diese verkettet werden müssen. Dies ist durch das sequenzielle Deployment bedingt.

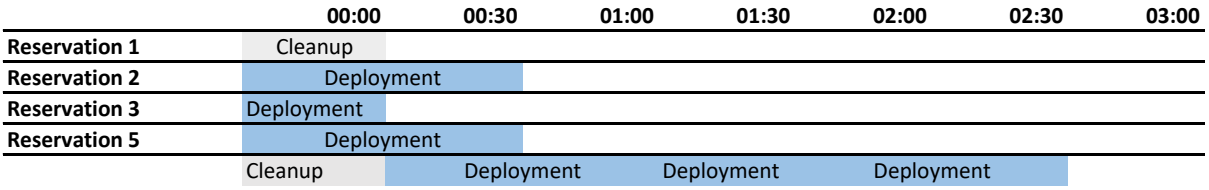

Abbildung 13: Berechnung Deploymentzeiten

Anhand der Deploymentzeiten kann festgestellt werden, ob für den geplanten Start die vorgesehene Startzeit der Reservation überschritten wird. Andererseits wird dadurch auch bestimmt, welche Startdaten für einen Zeitraum gesperrt sind.

Um in Zukunft ein paralleles Deployment auf unterschiedlichen Nodes zu unterstützen, müsste die Berechnung pro Deployment Node erfolgen. Dies analog zu der Berechnung der Ressourcen pro Node.

8.4.7 Settings

Für die Planung der Reservationen und zur Berechnung der Deploymentzeiten sind fünf Einstellungen relevant.

Feld	Wert
Identifier	DEPLOYMENT_START_TIME
Name	Startzeit Deployment
Description	Gibt an, zu welcher Tageszeit (UTC) das Deployment angestossen wird.
Default Value	"23:00"

Tabelle 32: DEPLOYMENT_START_TIME Setting

Feld	Wert
Identifier	RESERVATION_START_TIME
Name	Startzeit Reservationen
Description	Gibt an, zu welcher Tageszeit (UTC) Reservationen starten.
Default Value	"06:00"

Tabelle 33: RESERVATION_START_TIME Setting

Feld	Wert
Identifier	RESERVATION_END_TIME
Name	Endzeit Reservationen
Description	Gibt an, zu welcher Tageszeit (UTC) Reservationen enden.
Default Value	"23:00"

Tabelle 34: RESERVATION_END_TIME Setting

Feld	Wert
Identifier	LTB_CONNECTION_DEPLOYMENT_TIME
Name	Deploymentzeit Connections
Description	Gibt an, wie lange das Deployment einer Connection dauert.
Default Value	"00:05:00"

Tabelle 35: LTB_CONNECTION_DEPLOYMENT_TIME Setting

Feld	Wert
Identifier	LTB_CONNECTION_CLEANUP_TIME
Name	Cleanupzeit Connections
Description	Gibt an, wie lange der Cleanup einer Connection dauert.
Default Value	"00:01:00"

Tabelle 36: LTB_CONNECTION_CLEANUP_TIME Setting

8.4.8 Jobs

Um die Reservationen bereitzustellen und nach ihrem Ablauf auch wieder zu entfernen, wurde zwei Jobs definiert.

Feld	Wert
Name	Abgelaufene Reservationen entfernen
Task (registered)	orchestration.tasks.remove_expired_reservations
Task (custom)	orchestration.tasks.remove_expired_reservations
Enabled	True
Description	Prüft, ob aktuell laufende Reservation existieren, welche abgelaufen sind und startet die Entfernung dieser.
Crontab Schedule	* 23 * * * (m/h/dM/MY) UTC

Tabelle 37: Job Abgelaufene Reservationen entfernen

Feld	Wert
Name	Anstehende Reservationen bereitstellen
Task (registered)	orchestration.tasks. deploy_pending_reservations
Task (custom)	orchestration.tasks. deploy_pending_reservations
Enabled	True
Description	Startet das Deployment für anstehende Reservationen des nächsten Tages.
Crontab Schedule	5 23 * * * (m/h/dM/MY) UTC

Tabelle 38: Job Anstehende Reservationen bereitstellen

9 Sprintplanung im Detail

9.1 Sprint 0 (22.10.2021 - 06.10.2021)

Ziel: Projektplan, Ready for first sprint planning

Im Sprint 0 ging es vor allem um die Einarbeitung in das Projekt und die verwendeten Technologien. Zusätzlich wurde ein initialer Projektplan erarbeitet und JIRA für das Backlog Management eingerichtet.

Nummer	Titel
SLO-1	Setup JIRA
SLO-2	Document Project plan
SLO-3	Get started with Python / Django
SLO-4	Get started with react
SLO-5	Setup local dev environment

Tabelle 39: Items Sprint 0

9.2 Sprint 1 (06.10.2021 – 20.10.2021)

Ziel: Konzept für Kosten der Labs und Kapazitäten der Nodes

Im Sprint 1 wurde die Grundlage für die Berechnung der Ressourcen geschaffen. Dies wurde konzeptionell erarbeitet und auch bereits umgesetzt.

Nummer	Titel	SP
SLO-13	Konzeptionelle Erarbeitung Ressourcen pro Node speichern	3
SLO-17	Konzeptionelle Erarbeitung Kosten pro Lab speichern	3
SLO-18	Konzeptionelle Deployment anhand von erfassten Kosten und Ressourcen	3
SLO-19	Device um Kosten erweitern	2
SLO-20	Running Device um Kosten erweitern	1
SLO-21	Deployment Nodes CRUD	2
	Total	**14**

Tabelle 40: Items Sprint 1

9.3 Sprint 2 (20.10.2021 - 03.11.2021)

Ziel: Implementierung des Analytics Board und des Deployment Jobs

Im Sprint 2 wurde die konzeptionelle Grundlage für die Erstellung von Reservationen geschaffen. Das Deployment wurde angepasst, um die entsprechenden Ressourcen zu berücksichtigen. Zusätzlich wurde eine «Analytics Board» zur Visualisierung der verwendeten Ressourcen implementiert.

Nummer	Titel	SP
SLO-14	Konzeptionelle Erarbeitung Labs Reservieren	8
SLO-22	Deployment anhand von erfassten Kosten und Ressourcen	8
SLO-23	Analytics Board	8
	Total	**24**

Tabelle 41: Items Sprint 2

9.4 Sprint 3 (03.11.2021- 17.11.2021)

Ziel: Erster Entwurf Reservation erstellen und bereitstellen

Im Sprint 3 wurde die erste Version der Reservationen implementiert mit dem Ziel, das Minimum umzusetzen, damit Reservationen getestet werden können.

Nummer	Titel	SP
SLO-32	Reservationen erstellen	10
SLO-36	CRUD Settings	3
SLO-37	Cleanup / Deployment Job	8
SLO-38	Devices um Zeitkosten erweitern	2
SLO-40	Bearbeitung von Labs sperren	2
SLO-43	Running Devices um Zeitkosten erweitern	1
	Total	**26**

Tabelle 42: Items Sprint 3

9.5 Sprint 4 (17.11.2021 – 01.12.2021)

Ziel: Reservationen fertigstellen

Im Sprint 4 wurden die zusätzlichen Funktionalitäten der Reservationen fertiggestellt. Zusätzlich wurde neben der Ressourcenberechnung nun auch die Limitierung der Deploymentzeiten umgesetzt.

Nummer	Titel	SP
SLO-33	Reservation anpassen	3
SLO-34	Reservation beenden	2
SLO-35	Reservationen löschen	2
SLO-50	Übersicht Reservationen	3
SLO-61	Reservations-Kosten pro Deployment Node	5
SLO-62	Deployment und Cleanupzeiten für Reservation berücksichtigen	8
	Total	**23**

Tabelle 43: Items Sprint 4

9.6 Sprint 5 (01.12.2021 – 15.12.2021)

Ziel: Production ready

Im Sprint 5 wurden kleinere Fehler behoben und minimale Ergänzungen vorgenommen. Zusätzlich wurde das Deployment auf Testing vorbereitet und durchgeführt.

Nummer	Titel	SP
SLO-66	Running Lab announcement	1
SLO-72	Startdatum blockiert wenn cleanup time sehr klein ist	1
SLO-73	Blockierte Ressourcen im UI sollten Startzeit berücksichtigen	1
SLO-74	Reservation Unterscheidung zwischen Scheduled Start/Ende und realem Start/Ende	5
SLO-76	Manuelles Deployment / Cleanup Reservation	2
SLO-77	Connections Deploymentzeit-Kosten	1
SLO-78	Lab/Gruppe löschen	2
SLO-79	Deployment auf Testumgebung und Testing	3
	Total	**16**

Tabelle 44: Items Sprint 5

9.7 Sprint 6 (15.12.2021 – 24.12.2021)

Ziel: Projektbericht fertigstellen und Abgabe

Der letzte Sprint widmet sich der Dokumentation des Projekts und der notwendigen Dokumente für die Abgabe. Zusätzlichen wurden noch kleinere Fixes aus den Resultaten des Testings umgesetzt.

Nummer	Titel
SLO-70	Projektbericht
SLO-71	Präsentation
SLO-80	Plakat
SLO-81	Abstract
SLO-84	Fixes aus Testing

Tabelle 45: Items Sprint 6

10 Resultat

Die bestehende LTB Webapplikation ist um die zwei Webpages Analytics und Reservations ergänzt worden. Der Dialog für das Deployment eines Labs ist durch den Dialog für das Erstellen einer Reservation ersetzt und die Running Lab Ansicht wurde für Reservationen aktualisiert.

Auf der Analytics Page wird die aktuelle Auslastung der einzelnen Deployment Nodes dargestellt. Die freien, benutzten und verfügbaren dargestellten Zahlen sind die sich aus den Reservationen errechneten Daten. Die Graphik wiederum zeigt den Vergleich der errechneten Daten der Reservationen mit den aktuellen Prometheus-Daten.

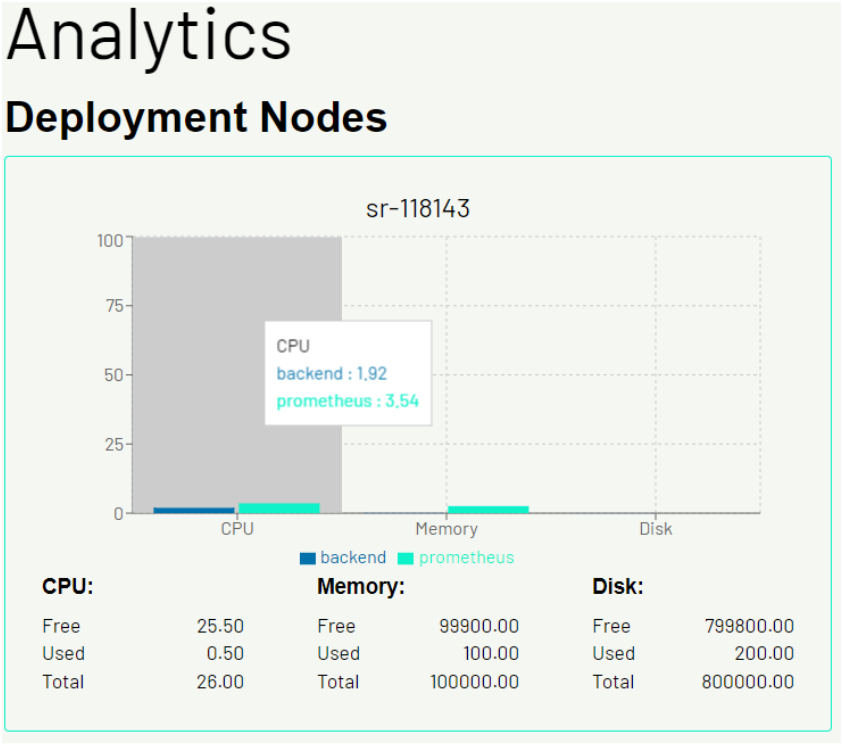

Abbildung 14: Analytics

Die beiden Dialoge, Create Reservation und Update Reservation, sind im Aufbau gleich, der User kann alle möglichen Daten für eine Reservierung selektieren. Im Update Reservation Dialog kann zusätzlich das Löschen und Terminieren der Reservationen gemacht werden.

Der Administrator kann zudem im Update Reservation Dialog ein manuelles Deployment oder Removal ausführen.

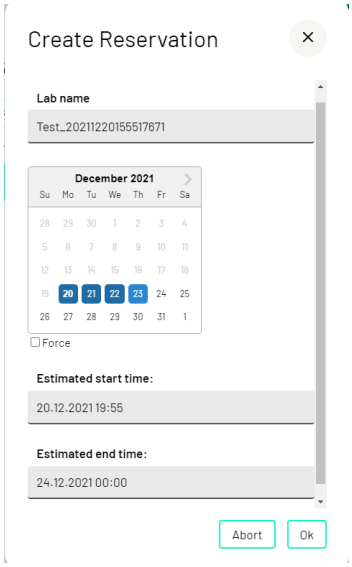

Abbildung 15: Create Reservation Dialog

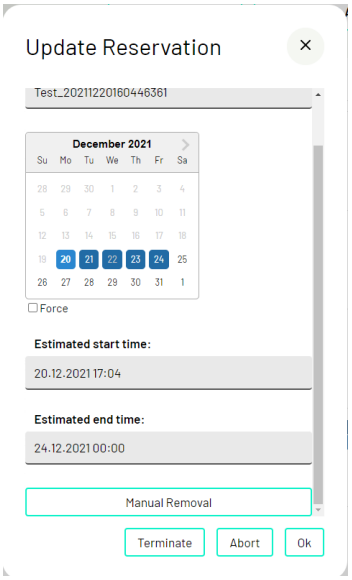

Abbildung 16: Update Reservation Dialog (running)

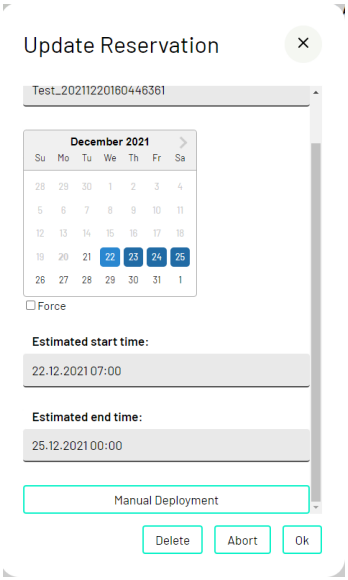

Abbildung 17: Update Reservation Dialog

Die Reservations Page zeigt alle Reservationen des Users. Bei der Selektion einer Reservation wird der «Update Reservation» Dialog geöffnet.

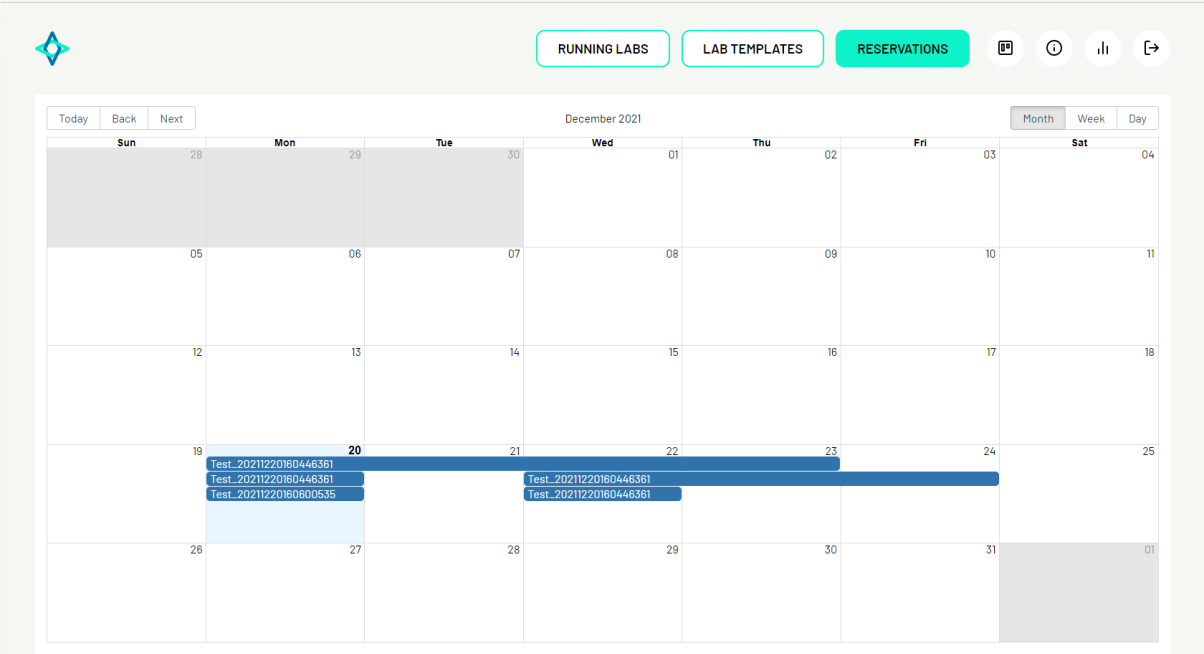

Abbildung 18: Screenshot Reservations Page

Die Running Labs Page wurde angepasst, so dass für Running Labs, die über eine Reservation erstellt wurden, die Option für das Stoppen eines Labs nicht mehr zur Verfügung steht, da diese neu via dem Update Reseravtion Dialog gemacht werden muss.

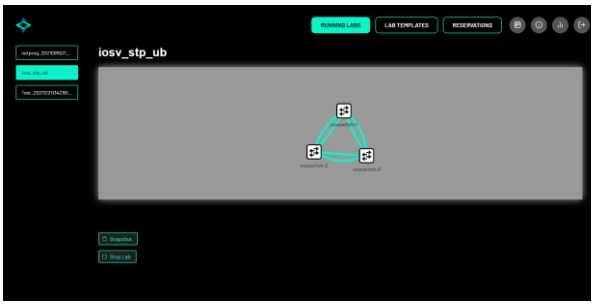

Abbildung 19: Altes Running Lab ohne Reservation

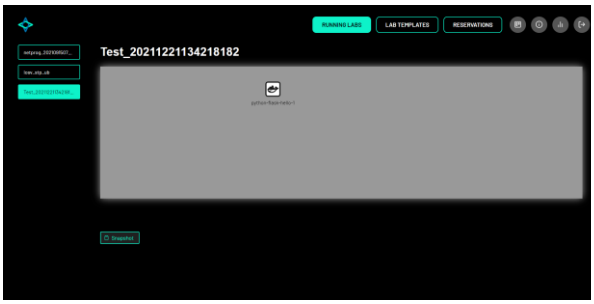

Abbildung 20: Neues Running Lab mit einer Reservation

11 Hinweis Quellenangabe

Es wurden ausschliesslich Abbildungen und Tabellen aus eigener Darstellung verwendet, weshalb auf eine explizite Quellenangabe «eigene Darstellung» verzichtet wurde. Jegliche fremden Quellen werden explizit angegeben.

12 Abbildungsverzeichnis

13 Tabellenverzeichnis

14 Literaturverzeichnis

Docker Inc. (21. 12 2021). *docker run*. Von Docker Documentation: https://docs.docker.com/engine/reference/commandline/run/#set-storage-driver-options-per-container abgerufen

Docker Inc. (21. 12 2021). *Runtime options with Memory, CPUs, and GPUs*. Von Docker Documentation: https://docs.docker.com/config/containers/resource_constraints/ abgerufen

Hans, T. (21. 12 2021). *Docker Container CPU Limits Explained*. Von Thorsten Hans' blog: https://www.thorsten-hans.com/docker-container-cpu-limits-explained/ abgerufen

Hyve. (21. 12 2021). *What is a VMware vCPU?* Von Hyve Managed Hosting: https://www.hyve.com/what-is-a-vmware-vcpu/ abgerufen

Red Hat Inc. (21. 12 2021). *Overcommiting memory*. Von Red Hat | Customer Portal: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/virtualization_deployment_and_administration_guide/sect-overcommitting_with_kvm-overcommitting_memory abgerufen

Red Hat Inc. (21. 12 2021). *Overcommitting virtualized cpus*. Von Red Hat | Customer Portal: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/virtualization_deployment_and_administration_guide/sect-overcommitting_with_kvm-overcommitting_virtualized_cpus abgerufen

15 Anhang

15.1 Team Report

Als wir mit der Arbeit begonnen haben, gab es noch ein paar Unsicherheiten. Einerseits war uns anfangs nicht bewusst, dass wir mindestens zwei Tage pro Woche aufwenden müssen. Andererseits hatten wir Bedenken bezüglich der Entwicklung mit Python, da wir bisher wenig bis gar keine Erfahrungen damit hatten. Durch eine Reduktion auf ein 40% Pensum in unserer Berufstätigkeit und eine Einarbeitung in das Projekt, sowie auch in Django, konnten wir das weitere Vorgehen gut aufgleisen.

Mit dem agilen Vorgehen selbst waren wir sehr zufrieden, da am Anfang des Projekts noch sehr unklar war, was überhaupt in vorgegeben Zeit erarbeitet oder umgesetzt werden kann. Dabei haben wir auch stark darauf geachtet in einem ersten Schritt immer nur so viel umzusetzen, wie auch absolut nötig (Minimum Vaiable Product). Dies hat uns auch extrem bei der Erarbeitung der doch sehr komplexen Thematik geholfen. Eine Schätzung auf Basis von Story Points war für uns ebenfalls sehr hilfreich. Dabei konnten wir bereits ab dem Sprint 3 schon sehr gute Einschätzungen vornehmen.

Die Zusammenarbeit im Projektteam war für uns beide sehr angenehm. An diesem Punkt möchten wir auch ein grosses Dankeschön an Yannick und Sebastian aussprechen, die uns während der Arbeit als Ansprechpartner unterstützt haben. Ihr Feedback während der Retrospektiven war für uns sehr wertvoll.

In Zukunft würden wir sicher mehr Zeit für das Deployment und Testing einplanen. Dies war gegen Ende doch eher etwas knapp, wobei wir auch noch kleinere Korrekturen durchführen mussten.

Mit dem Resultat der Arbeit sind wir äusserst zufrieden. Der minimale Anspruch war eher ein Proof of Concept, wobei wir nun eine produktive Version des Systems zur Verfügung stellen können.

15.2 Wireframes

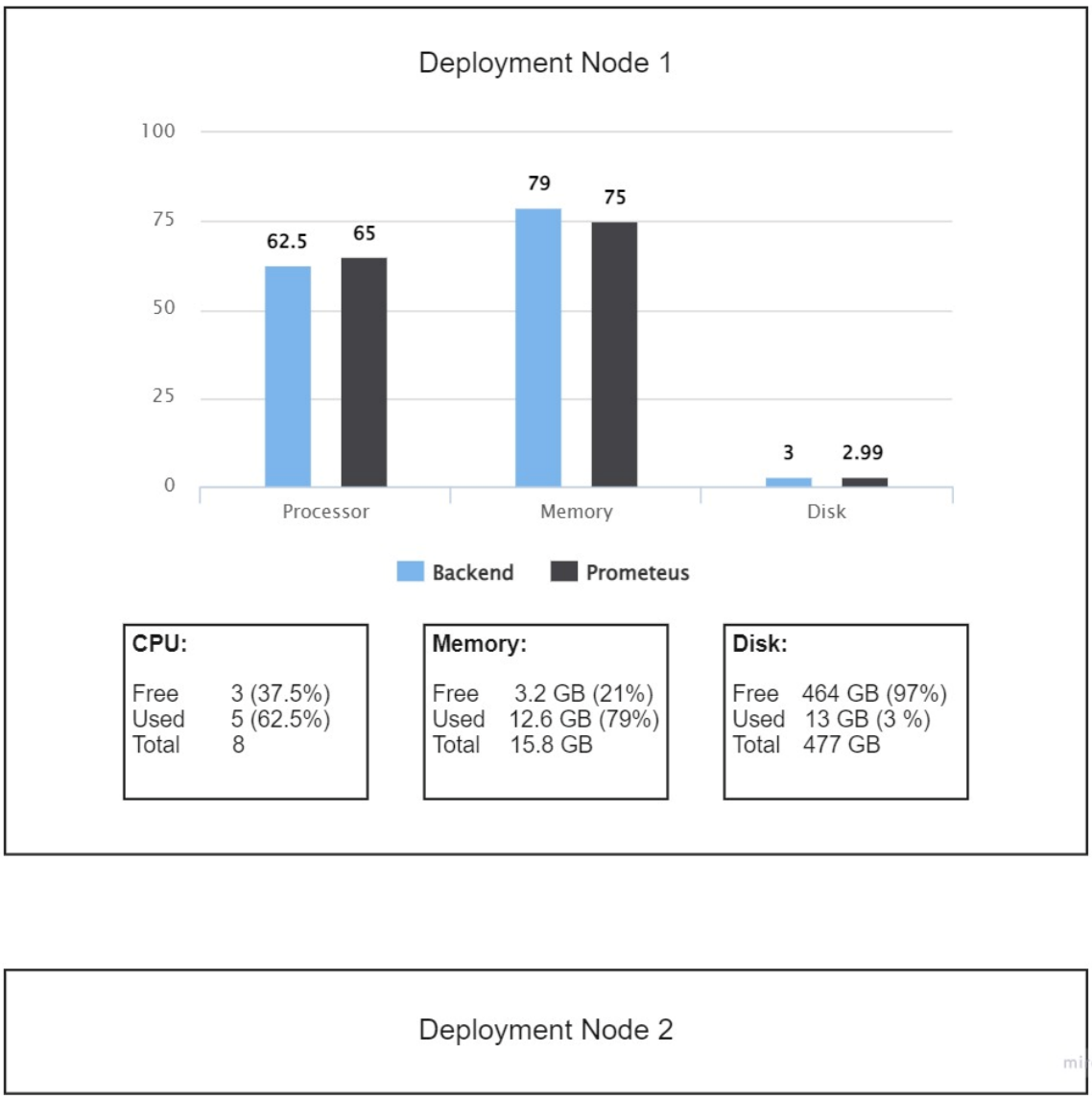

Abbildung 21: Wireframe Analytics Board

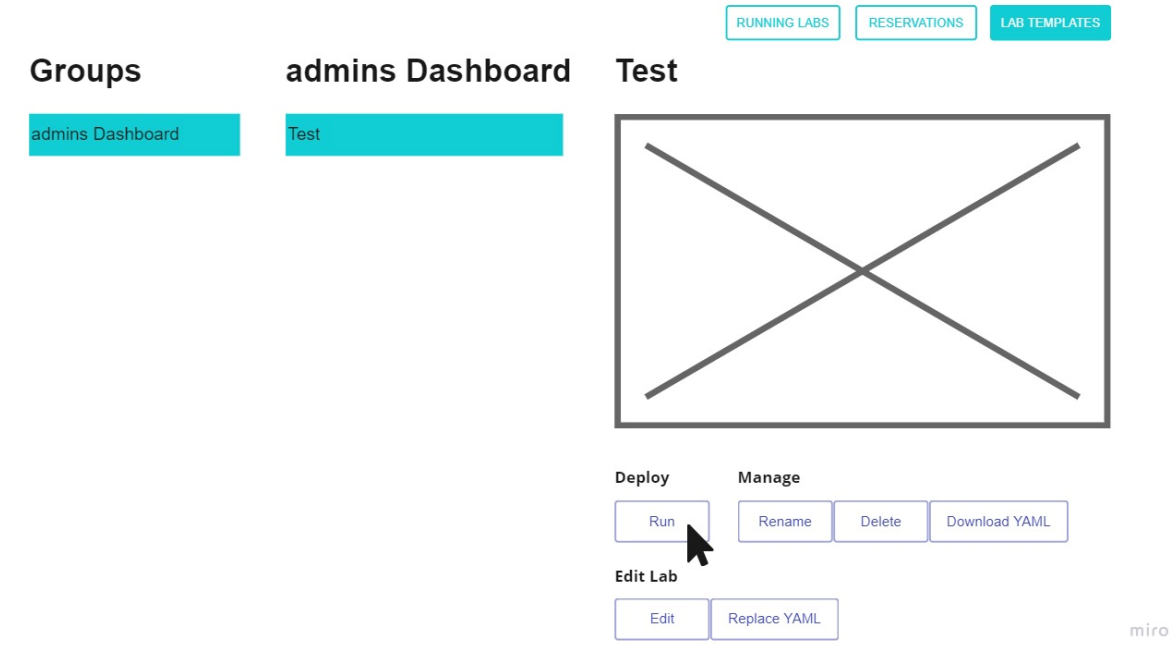

Abbildung 22: Wireframe Labs Page

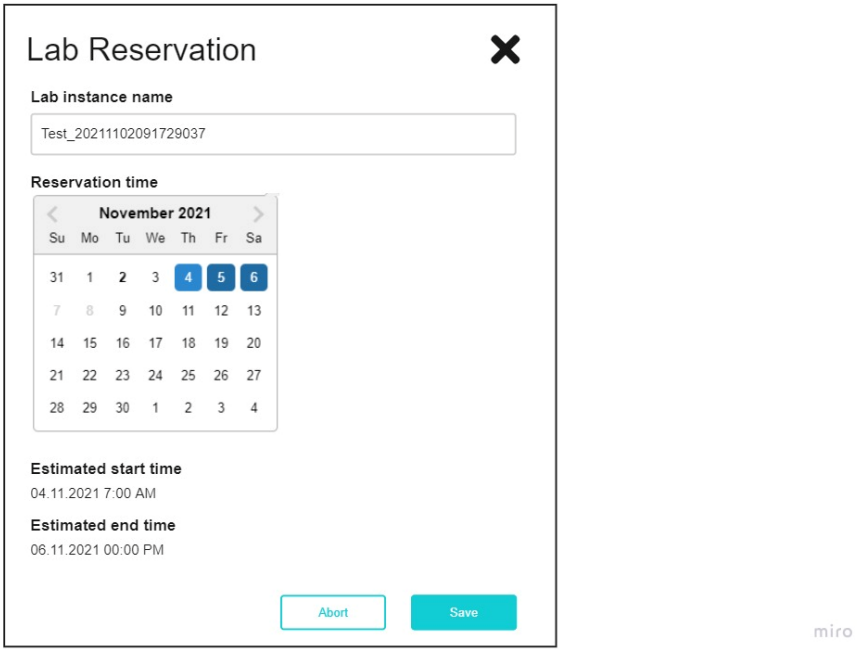

Abbildung 23: Wireframe Run Lab Dialog (Later Date)

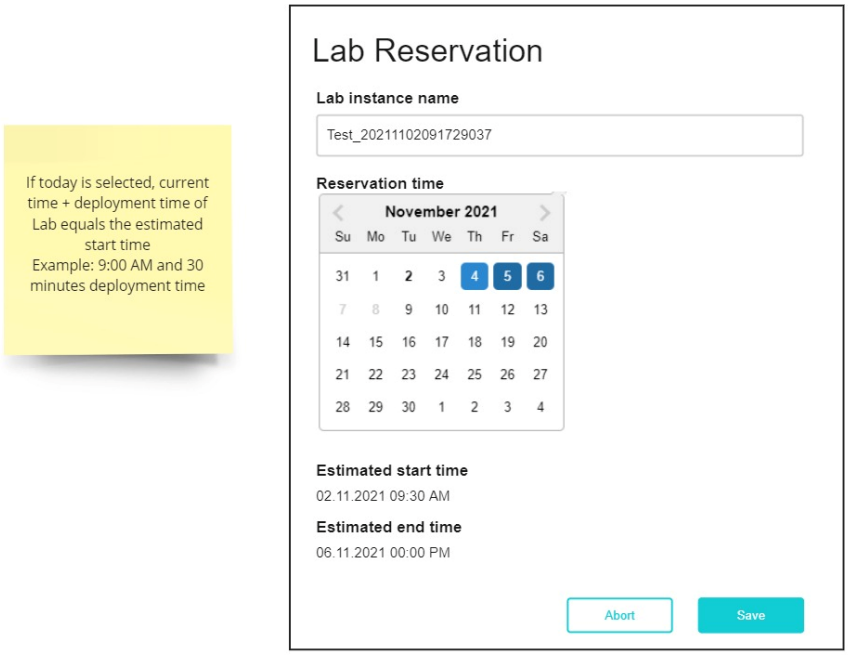

Abbildung 24: Wireframe Run Lab Dialog (Today)

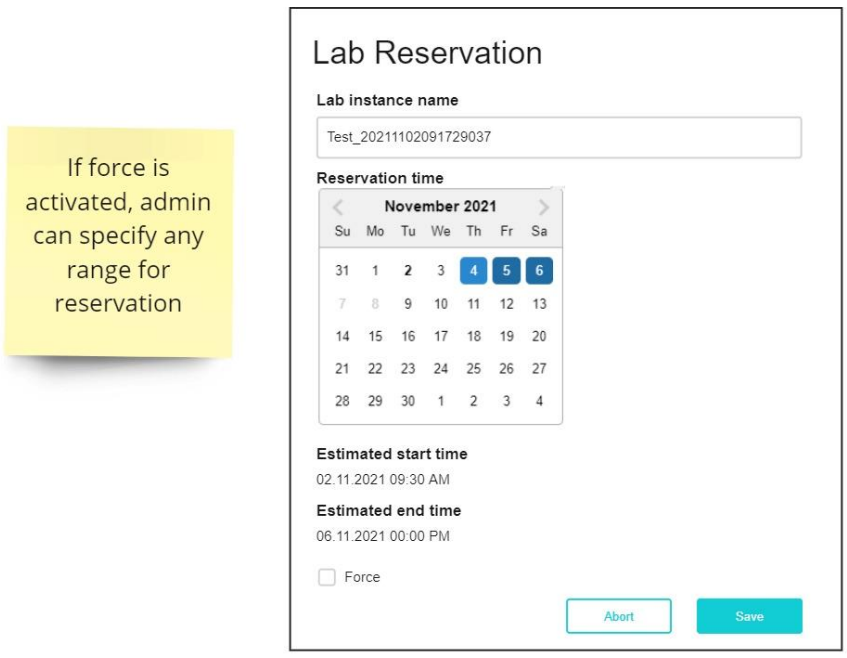

Abbildung 25: Wireframe Run Lab Dialog (Admin)

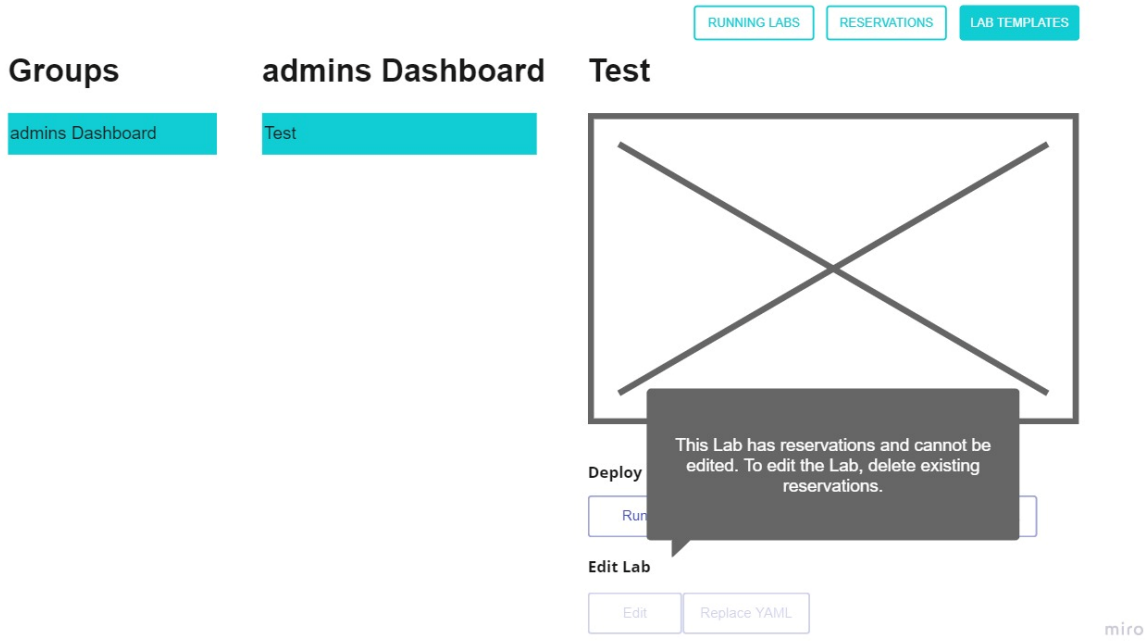

Abbildung 26: Wireframe Labs Page (Reservation exists)

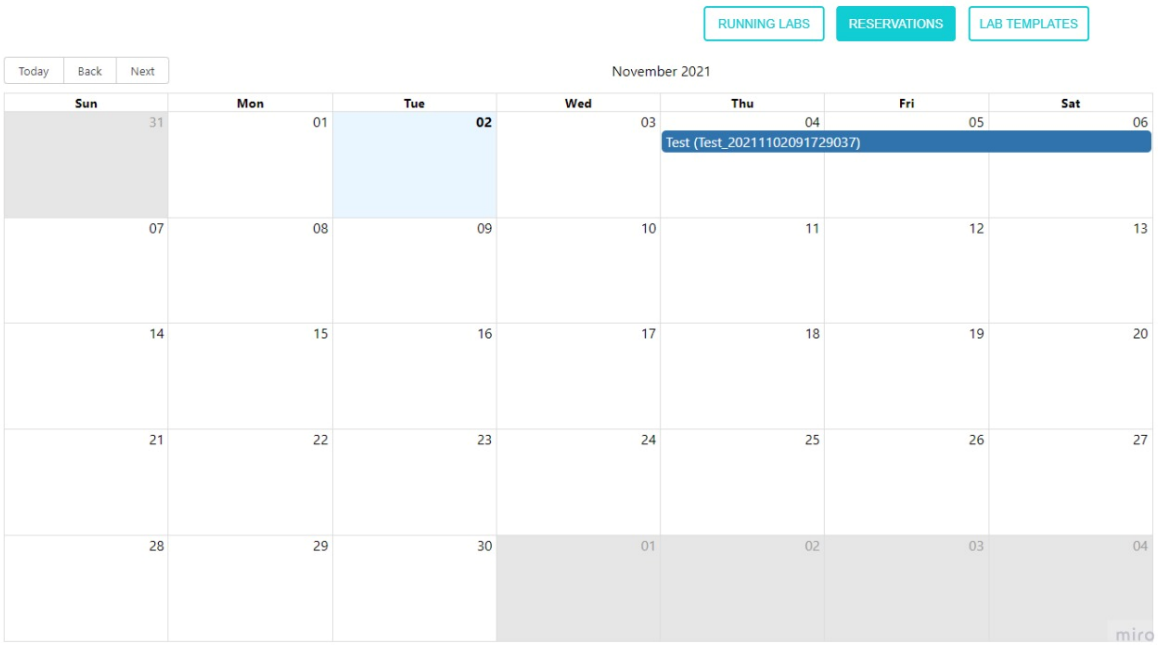

Abbildung 27: Wireframe Reservations Page (User)

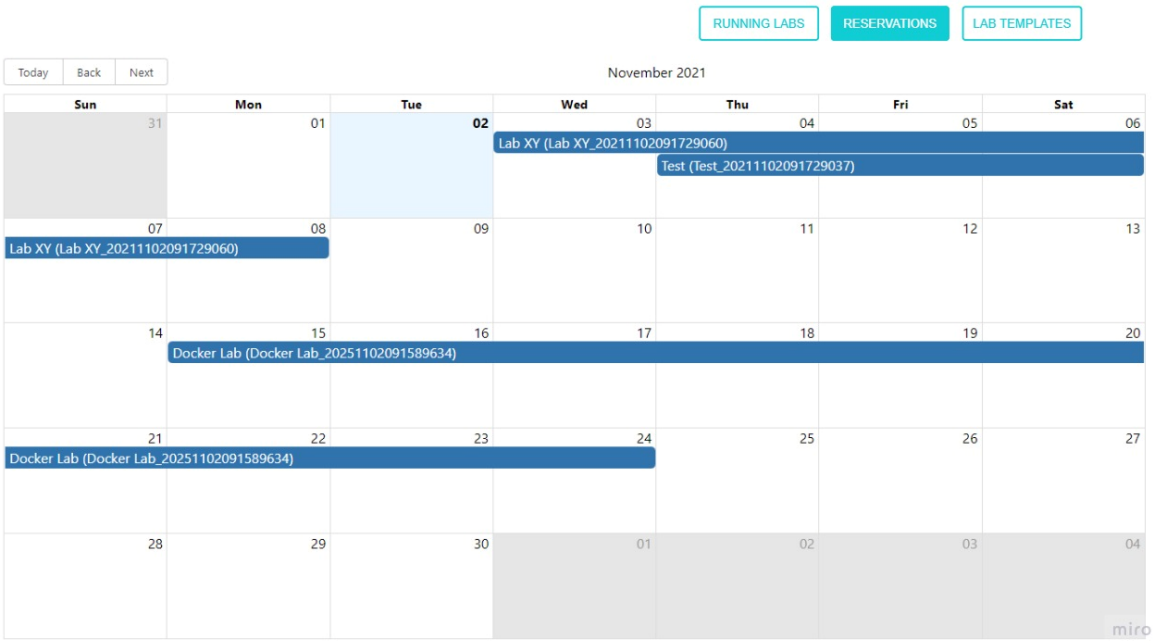

Abbildung 28: Wireframe Reservations Page (Admin)

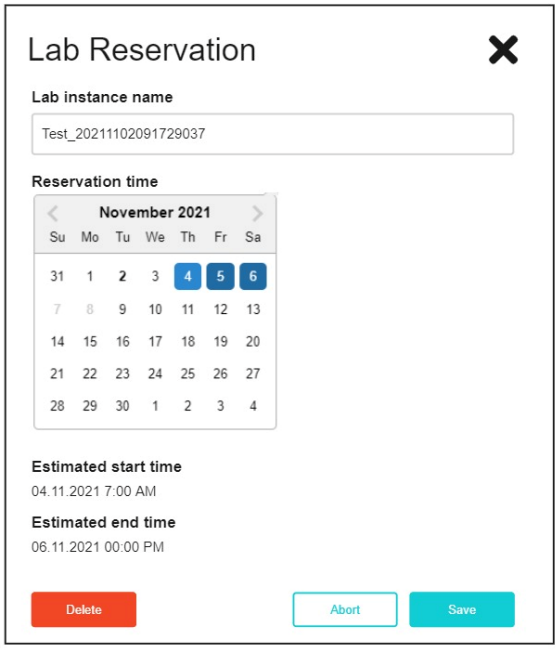

Abbildung 29: Wireframe Reservation Edit Dialog (Not Running)

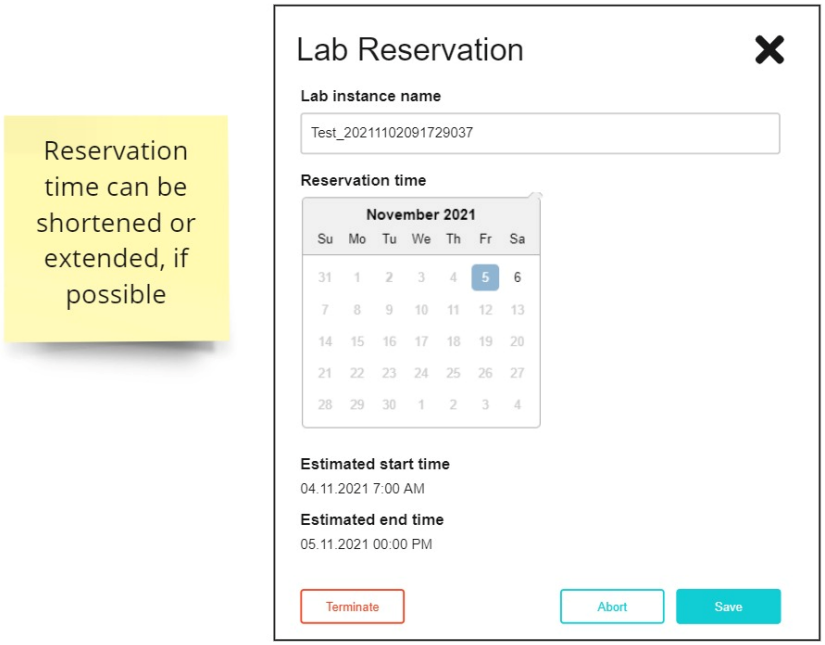

Abbildung 30: Wireframe Reservation Edit Dialog (Running, Today is 05.11.2021)

15.3 Screenshots Anwendung

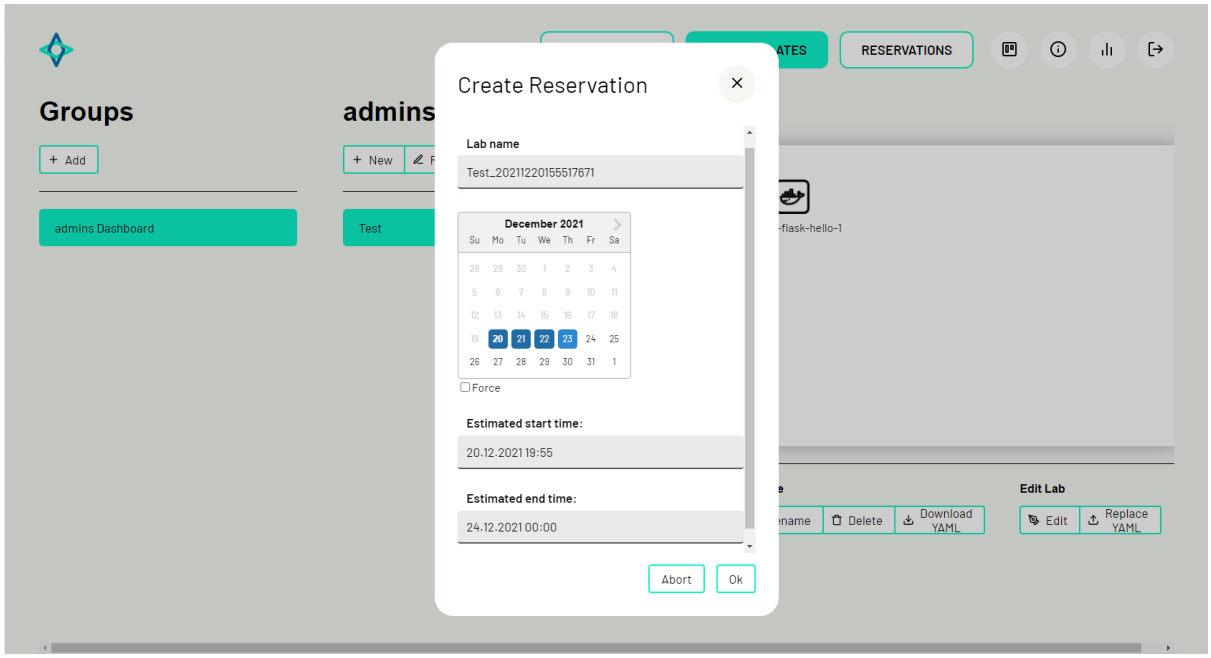

Abbildung 31: Screenshot Create Reservation Dialog (Start Date Today)

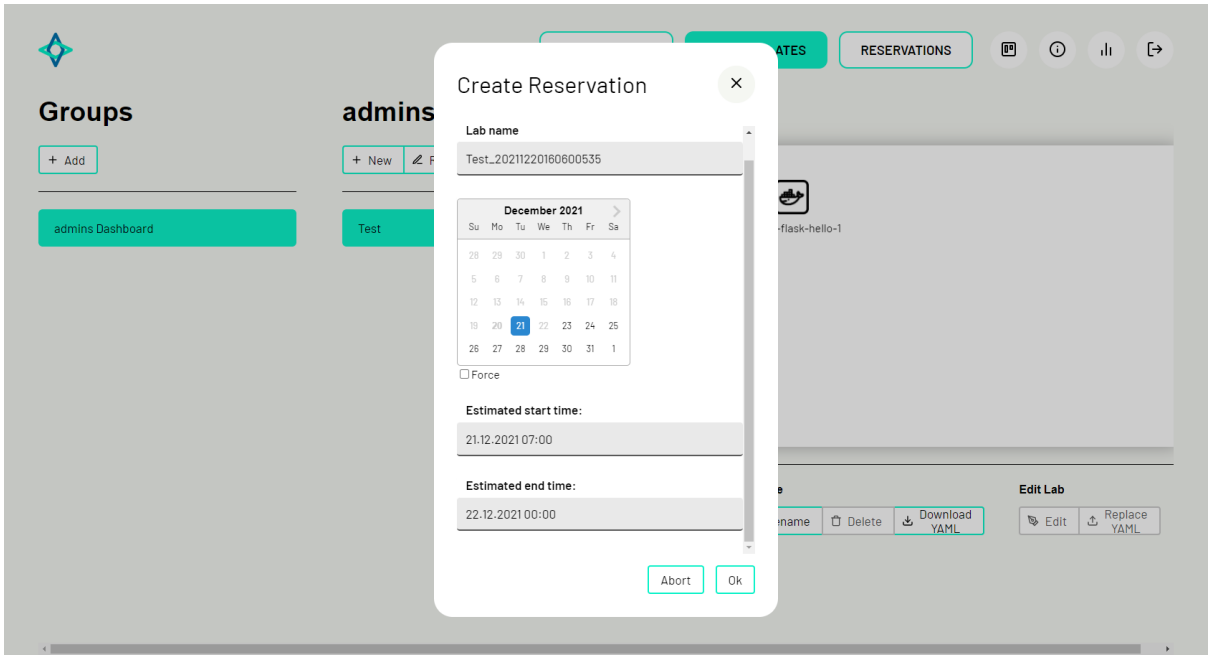

Abbildung 32: Screenshot Create Reservation Dialog (Locked Dates)

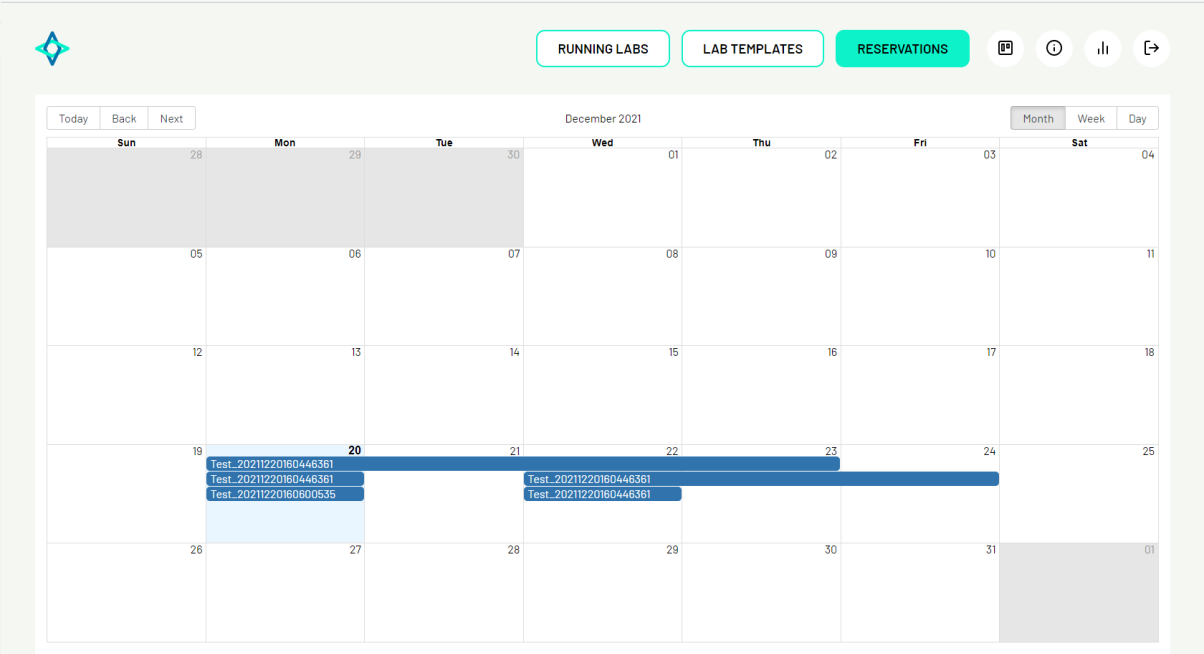

Abbildung 33: Screenshot Reservations Page

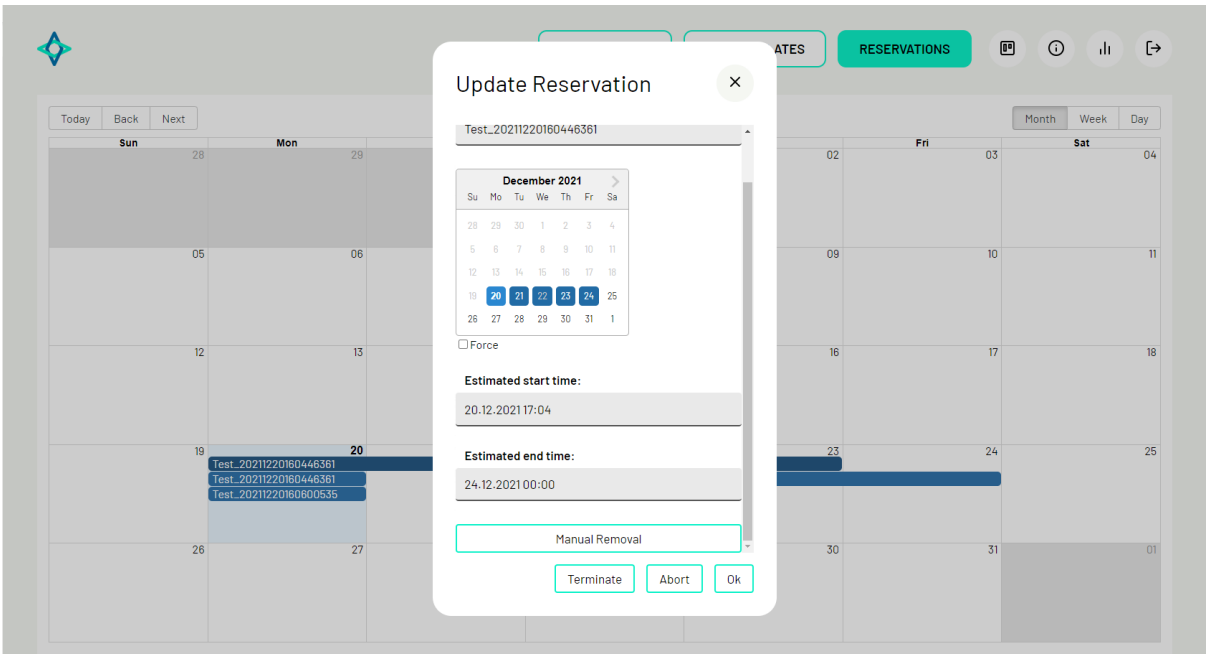

Abbildung 34:Screenshot Update Reservation Dialog (Running)

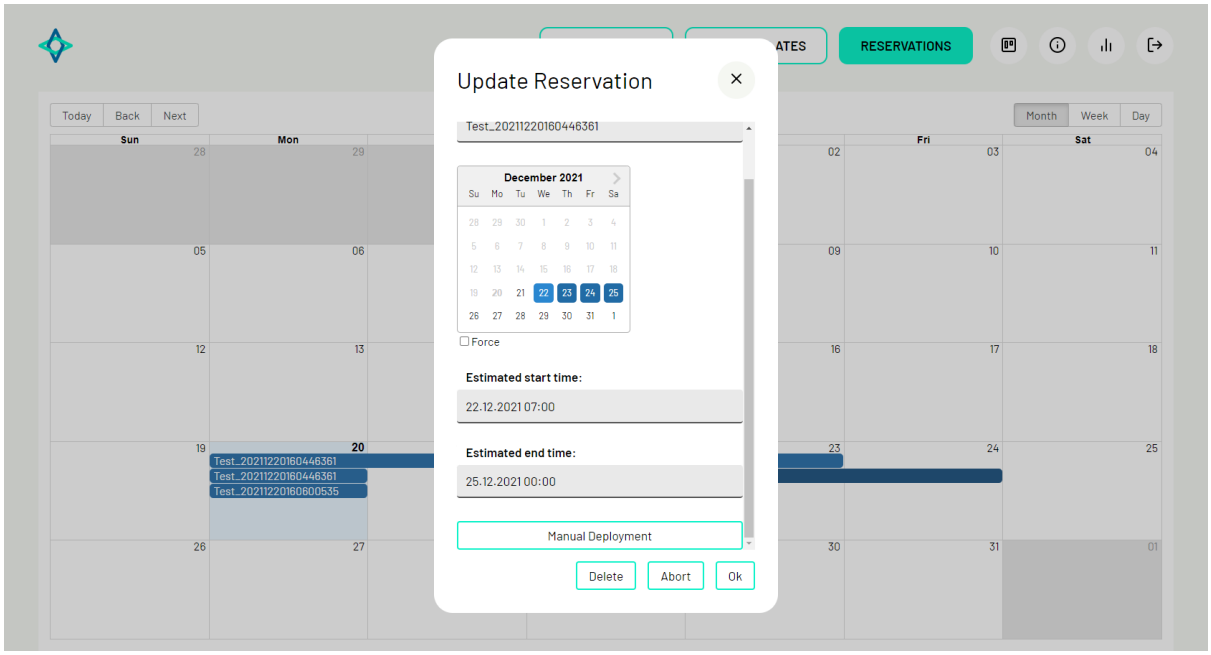

Abbildung 35: Screenshot Update Reservation Dialog (Scheduled)

Abbildung 36: Screenshot Analytics Board

15.4 Coverage Report

21.12.21, 16:10 Coverage report

Coverage report: 95% filter...

Module ↑	statements	missing	excluded	coverage
authentication/__init__.py	2	0	0	100%
authentication/admin.py	10	0	0	100%
authentication/auth.py	28	1	0	96%
authentication/filters.py	11	1	0	91%
authentication/migrations/0001_initial.py	8	0	0	100%
authentication/migrations/0002_genericchild_genericparent.py	6	0	0	100%
authentication/migrations/0003_remove_user_is_orchestrator.py	4	0	0	100%
authentication/migrations/__init__.py	0	0	0	100%
authentication/models.py	16	0	0	100%
authentication/permissions.py	57	4	0	93%
authentication/roles.py	12	0	0	100%
authentication/serializers.py	60	0	0	100%
authentication/urls.py	5	0	0	100%
authentication/views.py	4	0	0	100%
backend/__init__.py	0	0	0	100%
backend/celery.py	21	11	0	48%
backend/settings.py	94	0	0	100%
backend/urls.py	6	0	0	100%
core/__init__.py	0	0	0	100%
core/admin.py	3	0	0	100%
core/fields.py	25	6	0	76%
core/migrations/0001_initial.py	6	0	0	100%
core/migrations/__init__.py	0	0	0	100%
core/models.py	28	1	0	96%
core/serializers.py	6	0	0	100%
core/urls.py	6	0	0	100%
core/views.py	11	0	0	100%
devices/__init__.py	0	0	0	100%
devices/admin.py	18	0	0	100%
devices/migrations/0001_initial.py	6	0	0	100%
devices/migrations/0002_auto_20200710_1144.py	4	0	0	100%
devices/migrations/0003_auto_20200813_0503.py	4	0	0	100%
devices/migrations/0004_auto_20200827_1102.py	5	0	0	100%
devices/migrations/0005_device_empty_interfaces.py	4	0	0	100%
devices/migrations/0006_auto_20200925_1444.py	4	0	0	100%
devices/migrations/0007_auto_20210621_1510.py	4	0	0	100%
devices/migrations/0008_auto_20210824_1024.py	4	0	0	100%
devices/migrations/0009_auto_20211017_0910.py	5	0	0	100%
devices/migrations/0010_auto_20211024_1137.py	5	0	0	100%
devices/migrations/0011_auto_20211107_1021.py	5	0	0	100%
devices/migrations/__init__.py	0	0	0	100%
devices/models.py	126	5	0	96%
devices/serializers.py	54	2	0	96%
devices/static.py	1	0	0	100%
devices/urls.py	12	0	0	100%
devices/views.py	35	3	0	91%
Total	6664	340	0	95%

Abbildung 37: First Page Coverage Report with Total

15.5 Zeitauswertung

Labels	Time spent
Abgabe	62h
Backend	283h 50m
Conception	55h
Construction	292h 20m
Frontend	180h
Meeting	34h 45m
Setup	50h 30m
	494h 35m

Abbildung 38: Zeitaufwand nach Labels

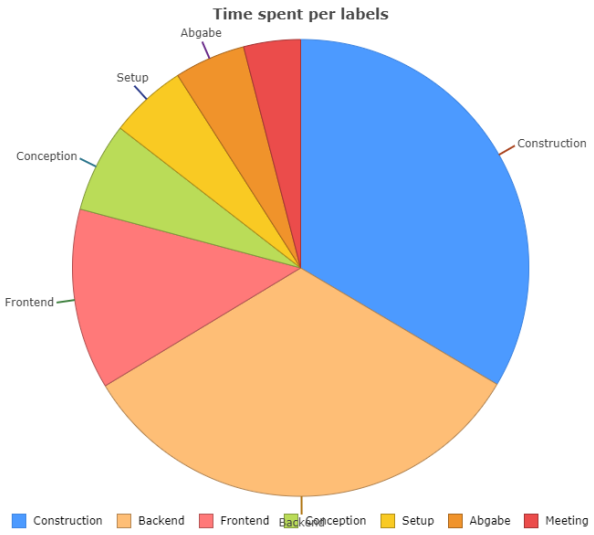

Abbildung 39 Zeitaufwand nach Labels als Diagramm

Daniel Els	Loris Keller	Total
248h 45m	245h 50m	494h 35m
248h 45m	245h 50m	494h 35m

Abbildung 40: Zeitaufwand pro Person

15.6 DB-Model

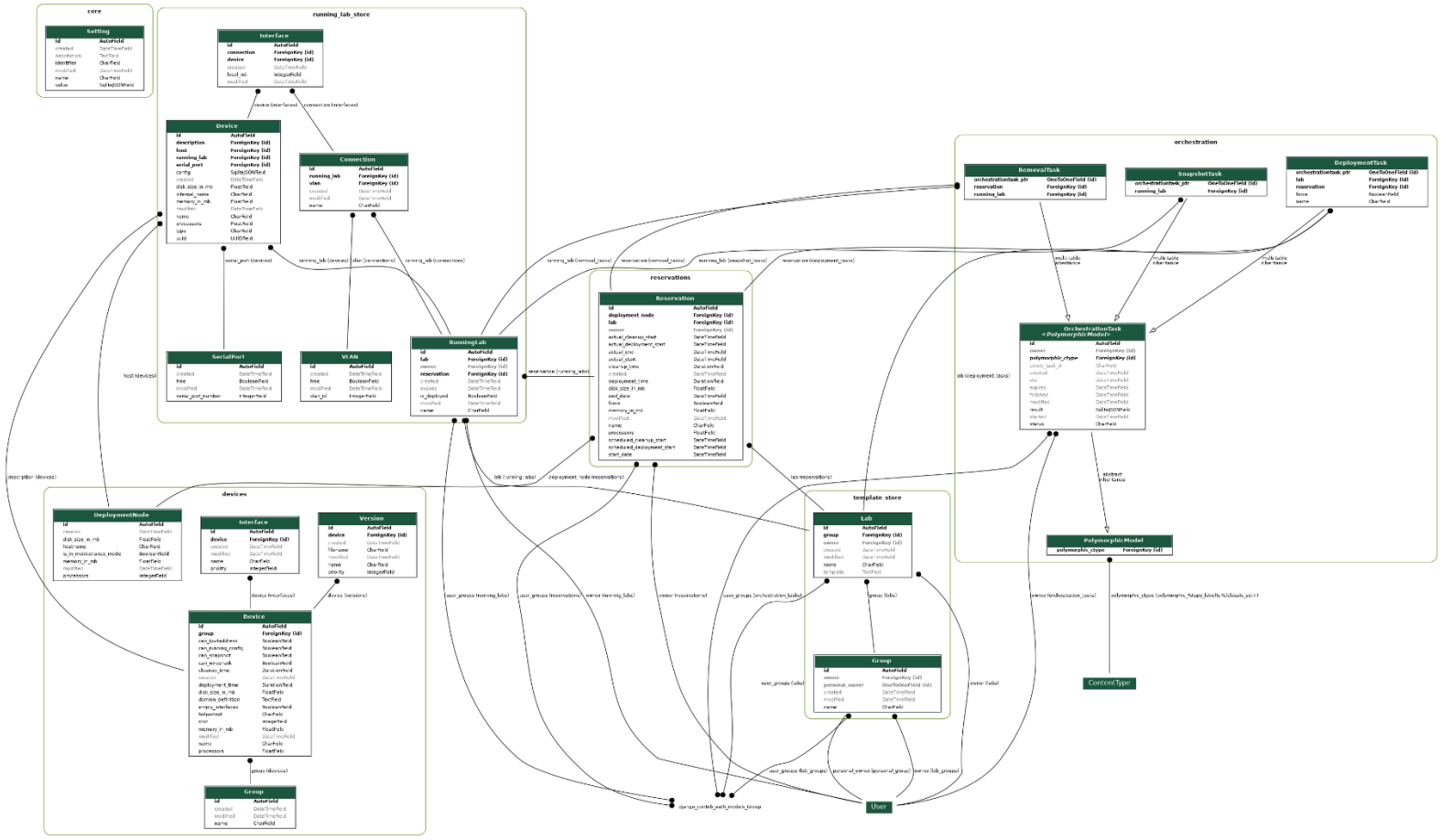

Abbildung 41: DB-Model