

---

# BACHELOR-PROJEKT

---

## HSF-STACK



**Matthias Maisch**

**matthias.maisch@informatik.hs-fulda.de**

**Annika Oeste**

**annika.oeste@informatik.hs-fulda.de**

**Fabian Karl**

**fabian.karl@informatik.hs-fulda.de**

**Lukas Kress**

**lukas.kress@informatik.hs-fulda.de**

**Stefan Friedmann**

**stefan.friedmann@informatik.hs-fulda.de**

# Inhaltsverzeichnis

---

<b>IST-ZUSTAND .....</b>	<b>1</b>
Struktur .....	1
Ablauf der Benutzung.....	1
Schwachstellen .....	1
 <b>AUFGABEN-STELLUNG .....</b>	 <b>2</b>
Soll.....	2
Projektziele.....	3
 <b>PROJEKTMANAGEMENT .....</b>	 <b>3</b>
Zeitplanung .....	4
Versionsverwaltung .....	4
Meilensteine .....	5
Teamkommunikation .....	5
 <b>KUNDENEINWEISUNG .....</b>	 <b>6</b>
„Normaler Nutzer“ .....	6
Admin .....	8
 <b>GENUTZTE TECHNOLOGIEN .....</b>	 <b>8</b>
Programmiersprachen .....	8
Dateistruktur .....	8
Datenbank.....	11

**TECHNISCHE ERSTE SCHRITTE ..... 11**

Notwendige Programme .....11

**PROJEKT-VERLAUF ..... 11**

**AUSBLICK..... 13**

## IST-ZUSTAND

---

### STRUKTUR

---

➤ AKTEURE

- Benutzer: Professoren und Dozenten des Fachbereichs Angewandte Informatik

➤ GRENZEN

- Webanwendung zur automatischen Bereitstellung und Verwaltung von Kurs-Umgebungen für die Unterstützung von Lehrveranstaltungen mit Mirantis Openstack

### ABLAUF DER BENUTZUNG

---

➤ Professor oder Dozent meldet sich an HSF-Stack Webseite an:

- Erstellt Kurs mit mind. einem Projekt, mit oder ohne Netzwerk
- Löschen von gesamten Kursen inkl. aller Projekte oder das Löschen einzelner Projekte
- Über Openstack/Horizon angelegte Instanzen können suspendiert bzw. wieder gestartet werden

### SCHWACHSTELLEN

---

- Anweisungen, die vom Webinterface zum Openstack-Server gesendet werden, brauchen sehr lange bis sie bearbeitet werden
- Benutzerverwaltung nur über Openstack/Horizon möglich
- Anlegen von Instanzen nicht über das Webinterface „HSF-Stack“ möglich

## AUFGABEN-STELLUNG

---

### SOLL

---

Entwicklung einer Web-Anwendung, die eine automatisierte Bereitstellung und Verwaltung von Kurs-Umgebungen für die Unterstützung von Lehrveranstaltungen mit Mirantis Openstack realisiert. Dabei sollen die Openstack APIs verwendet werden um zukünftige Versionen zu unterstützen.

Eine Authentifizierung und Autorisierung von Kursleitern und -teilnehmern soll idealerweise über die bestehende LDAP-Anbindung des Daten-Verarbeitungszentrums ermöglicht werden.

Über das Webinterface soll es möglich sein, Benutzer, Gruppen, Projekte und Rollen in Openstack für neue Kurse anzulegen, sowie die Zuweisung von virtuellen Netzwerken, Storage und das Erstellen von Instanzen für einzelne Projekte. Die Verwaltung der virtuellen Ressourcen der einzelnen Kurse soll das Starten, Stoppen und Suspendieren von Ressourcen (z.B. Instanzen) beinhalten und ggf. die Integration automatischer Snapshots/Backups etc. beinhalten. Am Schluss eines Kurses soll es möglich sein, die für alle Kursteilnehmer erstellten, Ressourcen zu entfernen.

## PROJEKTZIELE

---

- Webanwendung mit Anbindung an Mirantis OpenStack 9.x Umgebung des Fachbereichs Angewandte Informatik der Hochschule Fulda (Privat Cloud) **Erledigt!**
- Verwendung der OpenStack APIs. Wurde mit dem OpenStack Pythonclient umgesetzt. **Erledigt!**
- Proof-of-Concept inkl. Dokumentation **Erledigt!**
- Authentifizierung und Autorisierung von Kursleitern und -teilnehmern (ideal: über bestehende LDAP-Anbindung (DVZ)). Nachträglich mit Kunden (Prof. Sebastian Rieger) wurde die LDAP-Anbindung nicht umgesetzt.
- Anlegen von Benutzern und Kursen mit Projekten in OpenStack. **Erledigt!**
- Anlegen von Gruppen und Rollen wurde nicht umgesetzt, da für das Projekt nicht erforderlich.
- Einrichtung von Netzwerken. **Erledigt!**
- Einrichten von Storage und Zuweisungen von Images. Wurde nach Vereinbarung mit Kunden (Prof. Sebastian Rieger) nicht umgesetzt.
- Verwalten der virtuellen Ressourcen der Kurse, dass gesammelte suspendieren und starten von Instanzen wurde umgesetzt. **Erledigt!**
- Zeitgesteuertes Suspendieren und Starten von Instanzen wurde nach Vereinbarung vom Kunden nicht umgesetzt.
- Automatische Erstellung von Snapshots und Backups wurde nach Rücksprache mit Kunden nicht umgesetzt
- Entfernung aller eingerichteten Ressourcen am Ende eines Kurses wurde umgesetzt. **Erledigt!**

## PROJEKTMANAGEMENT

---

Als Team haben wir uns für ein agiles Projektmanagement entschieden. Wir haben uns in Gruppen aufgeteilt. Eine Gruppe übernahm die Entwicklung des Front-Ends und dessen Integration mit dem Back-End. Die andere Gruppe übernahm die Entwicklung des Back-Ends, also die Erstellung von Skripten für den OpenStack Pythonclient.

Der Team-Leiter koordinierte während des Projektes den Fortschritt der Entwicklung, machte Vorgaben für die einzelnen Gruppen und entwickelte beim Front-End mit.

Das gesamte Team traf sich jede Woche um den aktuellen Entwicklungsstand den anderen Mitgliedern zu präsentieren. Kurzfristige

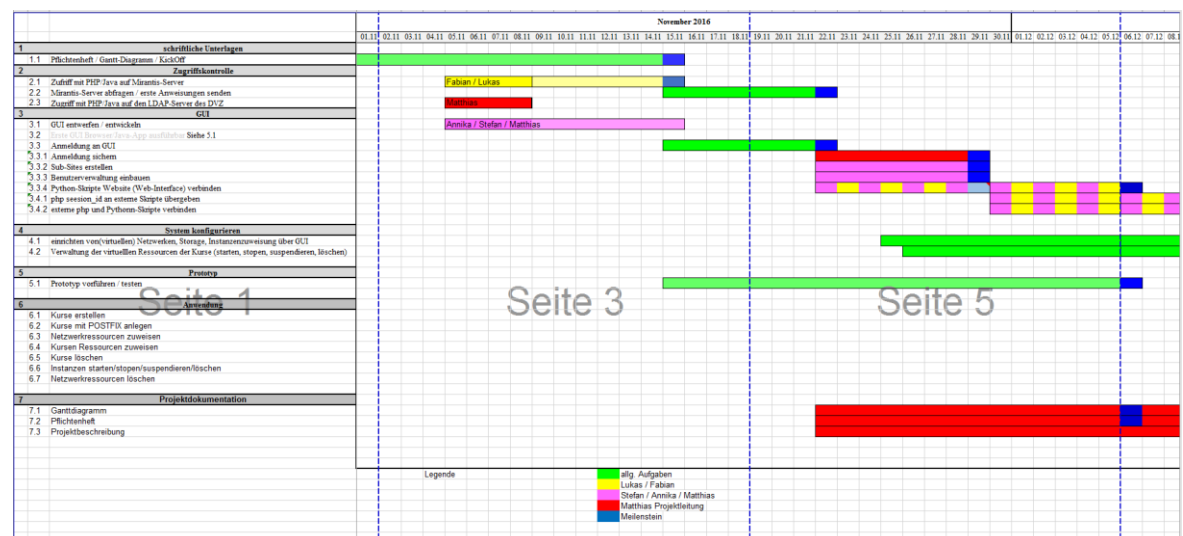
Fragen zwischen den einzelnen Mitgliedern wurden über den Messenger „Kik“ gestellt und beantwortet.

Die Mitglieder der Front-End Gruppe haben ihre Entwicklung mit GitHub geteilt und synchron gehalten.

Der Teamleiter hatten die Zeitplanung vorgegeben und diese ständig im Blick. Dafür hat er sich ein Gantt Diagramm erstellt und alle Teammitglieder haben sich daran orientiert (siehe nächstes Kapitel).

## ZEITPLANUNG

### GANTDIAGRAMM:



## VERSIONSVERWALTUNG

GitHub: <https://github.com/spawnms/HSF-Stack>

## MEILENSTEINE

---

Die Meilensteine waren wichtige Merkmale, um den Entwicklungsfortschritt sichtbar zu machen. Dafür definierte der Teamleiter während des Projekts und mit Vorgabe des Kunden mehrere Meilensteine

- 15.11 -> Erstellung vom Pflichtenheft, Gantt-Diagramm und KickOff Präsentation, sowie das Einrichten des OpenStack Pythonclients
- 22.11 -> Erstellung der ersten Pythonskripte um mit Mirantis OpenStack zu kommunizieren, Benutzeranmeldung an die Webanwendung
- 29.11 -> Anmeldung an GUI absichern (Verschlüsseln des Benutzerpassworts mit hmac-sha3-512), weitere GUI-Ansichten erstellen, Benutzerverwaltung mittels MySQL Datenbank erstellen,
- 06.12 -> mittels WebInterface die ersten PythonSkripte ausführen, Kundenpräsentation des Prototyps, Gantt-Diagramm und Pflichtenheft aktualisieren
- 23.12 -> PHP-Session für alle Seiten, externe php und python-Skriptaufrufe verknüpfen, Einrichten von virtuellen Netzwerken, Kurse erstellen, Kurse als Präfix der Projekte erstellen, Netzwerkressourcen erstellen, Gantt-Diagramm und Pflichtenheft aktualisieren
- 10.01 -> Löschen von Kursen

## TEAMKOMMUNIKATION

---

- Treffen während der Stunde
- Zwischendurch kürzere Meetings -> neuster Stand, Fragen an andere Teams
- Messenger App KIK
- Teilen, Weiterleiten von Emails
- Microsoft Office 365 für die Doku



# KUNDENEINWEISUNG

---

## „NORMALER NUTZER“

---

### Möglichkeiten auf der Seite *Main*:

→ **Zugang:** Seite direkt nach dem anmelden.  
Über HSF-Stack Logo

Es wird eine Liste mit allen Projekten angezeigt, sowie deren gesamte Instanzen. Diese sind zusätzlich aufgeteilt in gestoppte/suspendierte und aktive.

### Möglichkeiten auf der Seite *Kurs*:

→ **Zugang:** Menüpunkt Kurs in der Navbar

Die Kurse können erweitert werden, sodass eine Tabelle mit den zugehörigen Projekten angezeigt wird. In dieser Ansicht können einzelnen Projekte mit dem Mülleimer- Button gelöscht werden.

Mit dem Button „NEU“ können Benutzer oder Kurse angelegt werden.

Dabei müssen bei der Benutzererstellung angegeben werden:

- Benutzername
- Passwort
- Beschreibung
- Zugehöriges Projekt

Und bei der Kurserstellung:

- Kursname
- Projektname
- Projektanzahl

Außerdem kann man in der Checkbox markieren, ob man ein Default Netzwerk haben möchte.

Mit dem Button „INSTANZEN BEARBEITEN“ können alle Instanzen eines Kurses suspendiert oder neugestartet werden.

Mit dem „LÖSCHEN“ Button können ganze Kurse gelöscht werden.

### Möglichkeiten auf der Seite *Userverwaltung*:

→ **Zugang:** *Navbar Admin, DropDown Menü:*  
„Benutzername“

Benutzer werden in einer Tabelle angezeigt mit *Email, letztem Login* und *Rolle*. Bei Klicken auf die Buttons bekommt man aber eine Meldung, dass diese Funktionen nur von einem Administrator genutzt werden darf.

## ADMIN

---

Wie beim „normalen Nutzer“, allerdings kann ein Benutzer mit der Rolle „Admin“ unter der Userverwaltung (**Zugang:** *Navbar Admin, DropDown Menü: „Benutzername“*), Benutzer anlegen und löschen. Benutzer sind hier die Benutzer, die das Webinterface „HSF-Stack“ nutzen können.

## GENUTZTE TECHNOLOGIEN

---

### PROGRAMMIERSPRACHEN

---

#### Front-End

- HTML, php und JavaScript

#### Back-End

- Python und bash-Skript

### DATEISTRUKTUR

---

Alle Ordner im Arbeitsverzeichnis:

#### Ordner **config**

##### **useradd.php**

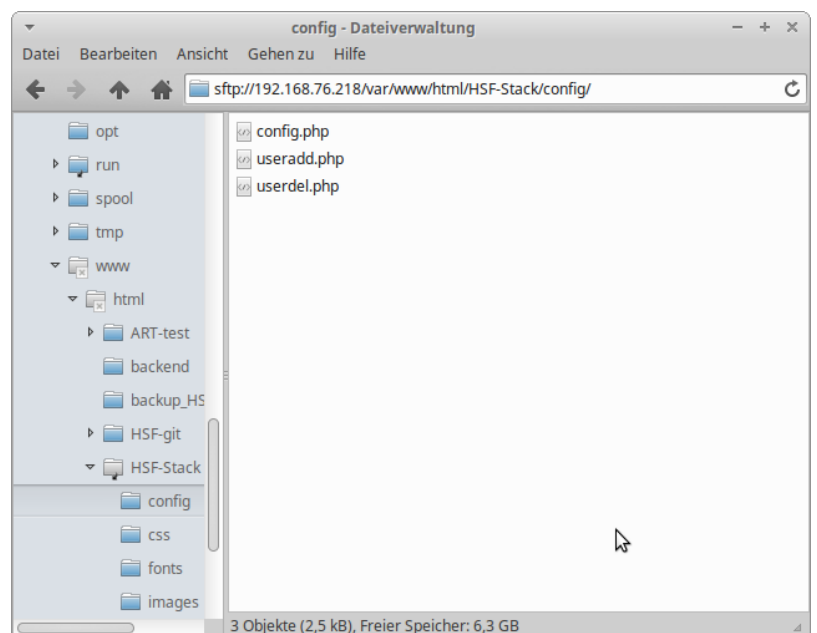
- Zum Erstellen neuer Benutzer für HSF-Stack

##### **userdel.php**

- Zum Löschen von Benutzern von HSF-Stack

##### **config.php**

- Datenbanktreiber mit Zugangsdaten für die MySQL-Datenbank, dem Salt zum hashen des Benutzerpasswortes und einem globalen Array welches Ausnahmen enthält, die innerhalb des Webinterfaces nicht angezeigt werden sollen.



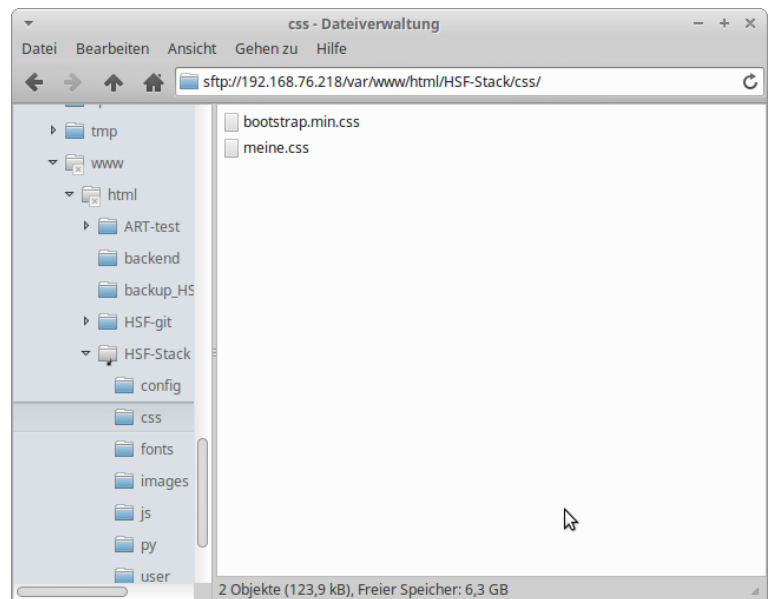
## Ordner **css**

### **Bootstrap.min.css**

- Cascading Style Sheet vom Bootstrap Framework

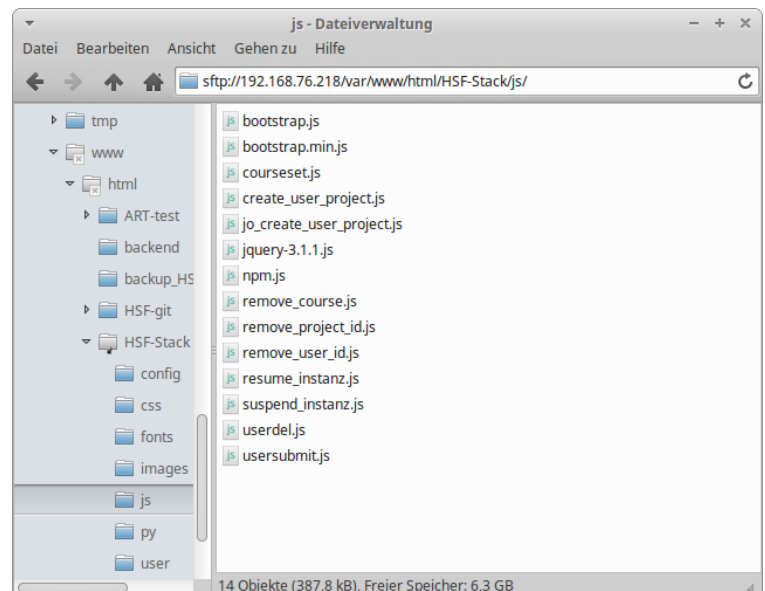
### **meine.css**

- Alle grafischen Anpassungen von unserem Front-End-Team



## Ordner **js**

- Hier liegen die von bootstrap und jquery genutzten js Dateien sowie alle js Dateien die für die Benutzereingaben über die Bootstrap-Modals notwendig sind. Ihre Aufgaben ergeben sich anhand der Dateinamen (siehe Kommentare im Quellcode).



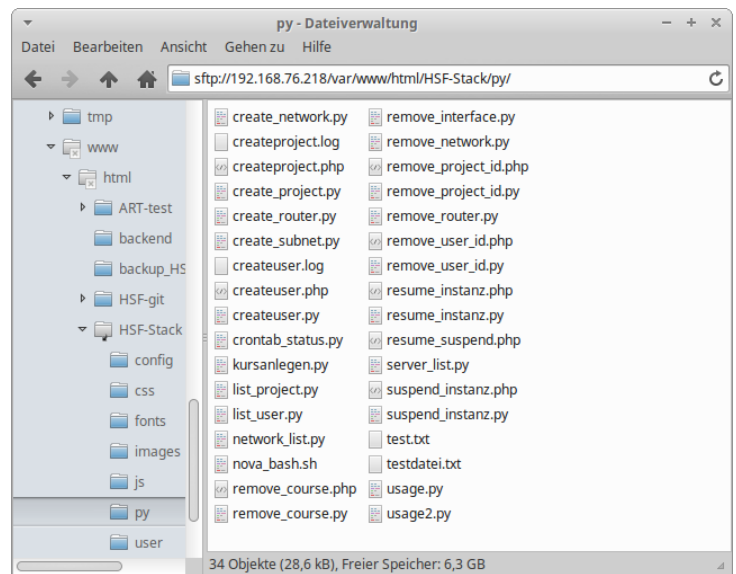
## Ordner **py**

- Hier liegen alle php **und** Python Skripte die für die Abfrage, das Erstellen von Benutzern und Kursen, für das Suspendieren und Fortführen von Instanzen, einem bash Skript welches den Status von Instanzen abfragt und in einer Textdatei (data.tmp) speichert. Die Benennung orientiert sich an der Funktion, für die das jeweilige Script verwendet wird. Beispielsweise wurden erstellende Skripte mit dem Präfix create versehen, gefolgt von einem Komponentennamen der Softwareumgebung, etwa USER, PROJECT usw.

Um die Skripte auszuführen, benötigen diese einige Eingabeparameter, etwa den Namen des Projekts, Nutzernamen, Passwort und Weitere. Diese werden von den im Frontend eingebetteten php-Formularen übergeben, welche namentlich eng mit dem dazugehörigen Script verknüpft sind.

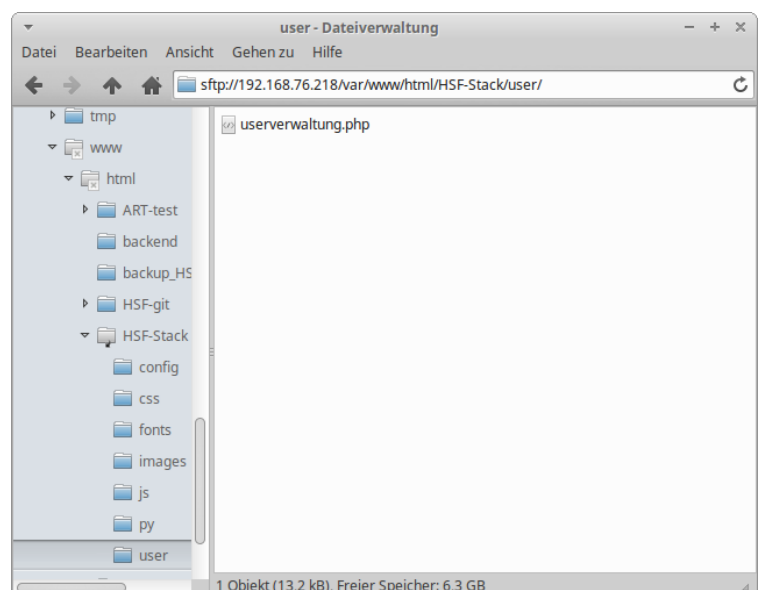
Die Skripte an sich führen hauptsächlich Konsolenbefehle mit den ihnen gegebenen Parametern aus, teilweise ist jedoch eine Schleifen-Funktion enthalten, um eine Aktion mehrmals ausführen zu können, zum Beispiel um mehrere Projekte ohne großen Nutzermehraufwand zu löschen.

Als Ausgabeformat der Skripte wurde falls möglich JSON gewählt, um die Einbettung der Daten im Frontend zu ermöglichen.



## Ordner **user**

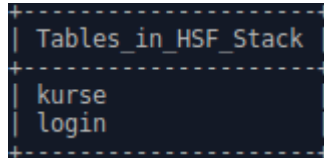
- Userverwaltung.php dient zur Verwaltung der „HSF-Stack“ Benutzer (**Zugang:** *Navbar Admin, DropDown Menü: „Benutzername“*)



## DATENBANK

---

- „HSF-Stack“ benutzt für die interne Benutzerverwaltung und für das Speichern von OpenStack – Projekten eine MySQL Datenbank mit zwei Tabellen.



```
Tables_in_HSF_Stack
kurse
login
```

- Tabelle **login**:
  - Enthält den Benutzernamen, das hmac-sha3(Passwort), E-Mail-Adresse, Datum/Uhrzeit des letzten Logins und die Benutzerrolle
- Tabelle **kurse**:
  - Enthält das Präfix des Projekts (als Kursname), den Projektnamen, die OpenStack Tenant-ID (eindeutige Projektbezeichnung innerhalb von OpenStack), die Netzwerk-ID (wenn einem Kurs bei Erstellung oder später manuell über OpenStack/Horizon ein Netzwerk zugeordnet wurde), den Routernamen und die Subnet-ID

## TECHNISCHE ERSTE SCHRITTE

---

### NOTWENDIGE PROGRAMME

---

- Webserver, MySQL-Server und php (aktuelle Version), müssen installiert sein.
- MySQL Datenbank anlegen
- Installieren vom OpenStack Python Client aus den Repositories von Ubuntu/Linux (python-openstackclient)
- config.php Datei im Ordner „config“ muss mit Zugangsdaten der MySQL Datenbank aktualisiert werden

## PROJEKT-VERLAUF

---

Hier wollen wir uns etwas Zeit nehmen und über den Projektverlauf berichten. Es war nicht immer einfach während des Projekts die Zeit zu finden, die wir gebraucht hätten. Am Anfang waren alle Mitglieder erst einmal erschlagen von der Fülle der Funktionen, die OpenStack mitbringt. Es galt also erst einmal, sich mit OpenStack auseinander zu setzen. Dann haben wir versucht eine geeignete API anhand des Wissenstandes der Mitglieder zu finden. Da war eine der beiden php-APIs erste Wahl. Später stellte sich aber heraus, dass die php-APIs nicht mehr gepflegt werden

und auf Grund fehlender Dokumentation haben wir schnell nach Alternativen gesucht. Zusammen mit dem Kunden haben wir uns dann den Openstack-Pythonclient angesehen. Dieser Client hat den Vorteil, dass er sehr einfach per Terminal zu bedienen ist und es eine sehr gute Dokumentation gibt. Dies ist auf den Umstand zurückzuführen, dass OpenStack zum einem großen Teil mit Python programmiert wurde.

Nachdem dieser Punkt geklärt war, ging es nun daran das Team so aufzuteilen, dass jeder eine Aufgabe hat. Wir entschieden uns dafür eine Gruppe für das Front-End (GUI) und eine Gruppe für das Back-End zu bilden. So waren die Zuständigkeiten geklärt und die ersten ansehnlichen Ergebnisse ließen nicht lange auf sich warten.

Die Back-End Gruppe stellte der Front-End Gruppe sehr schnell Pythonskripte zur Verfügung, die nur noch über ein Interface (hier eine Webseite) aufgerufen werden mussten. Dafür mussten Sie aber erst mit unserem Kunden zusammen dafür sorgen, dass alle virtuellen Rechner, die wir zur Entwicklung benötigten, untereinander kommunizieren konnten. Das war zum Anfang des Projekts noch nicht möglich. Es fehlten noch ein paar Netzwerkinterfaces.

Die Front-End Gruppe beschäftigte sich zu Anfang mit der Benutzeranmeldung für das Webinterface. Danach wurde nach einer ansprechenden Möglichkeit gesucht die Benutzereingaben anschaulich zu gestalten. Dafür entschied sich die Gruppe für ein Bootstrap – Modal. Diese Entscheidung führte dann zum nächsten Schwierigkeitsgrad, denn nun kam das jQuery Framework (JavaScript) hinzu.

Nach einer kurzen Einarbeitungszeit war auch dieses Problem Geschichte. Wir konnten nun dazu übergehen die Pythonskripte per PHP auszuführen. Dabei stellte sich schnell heraus, dass die Ausgabe der Skripte im JSON Format erfolgen musste. Somit konnten wir deren Rückgabewerte einfacher nutzen.

Der erste Termin für eine Kundenpräsentation näherte sich. Bis zu diesem Zeitpunkt hatten wir die GUI so gut wie fertig und auch die ersten Skripte wurden ausgeführt und innerhalb der GUI präsentiert. Bis dahin konnten wir recht schnell alle Probleme lösen. Jetzt wurde es etwas schwieriger.

Kurz vor Weihnachten kam dann noch ein Zwischenfall dazu, der am praktischen Beispiel zeigte, wie wichtig es ist immer und ständig Backups

durchzuführen. Ausgerechnet der Teamleiter hat unbewusst dafür gesorgt das alle, bis zu diesem Zeitpunkt erstellten, Pythonskripte gelöscht wurden.

Die Folge war eine langwierige aber erfolgreiche Datenrettung. Die Back-End Gruppe hatte versäumt regelmäßige Backups ihrer Daten zu machen und der Teamleiter hat versäumt darauf zu achten, dass diese durchgeführt wurden.

Wir mussten z.B. dafür sorgen, dass innerhalb von OpenStack Instanzen (virtuelle Rechner) zu verwalten sind. Dafür gibt es für den Pythonclient zwei unterschiedliche Befehle. Der eine Befehl war zuerst für uns nicht zugänglich, da noch ein Zertifikat, welches der Pythonclient benötigte, fehlte. Der andere Befehl funktionierte, aber die Ausgabe konnte nicht in das JSON – Format geändert werden. Wir haben uns dann dafür entschieden die Ausgabe mittels eines bash-Skriptes so zu verändern, dass wir eine für PHP verwendbare Datenstruktur bekamen. Das fehlende Zertifikat haben wir dann mit Hilfe unseres Kunden installiert bekommen. Dabei handelte es sich um einen Bug im Pythonclient. Das letzte größere Problem war das Löschen von Kursen, die ein Netzwerk inkl. Router und Subnetz zugeordnet hatten. Dabei musste man beim Löschen die richtige Reihenfolge der einzelnen Komponenten einhalten. Also erst das Subnetz, dann den Router, dann das Netzwerk entfernen um einen Kurs mit all seinen Projekten zu löschen.

Letztendlich war das Projekt eine Herausforderung, zum Teil für jedes Teammitglied, da jeder sich mit Programmiersprachen/Skriptsprachen und Systemen auseinandersetzen musste, mit denen er noch nicht oder nicht ausreichend gearbeitet hatte. Für den Teamleiter lag die Herausforderung darin, einen vernünftigen Zeitplan zu erstellen und nicht zu viel Zeit mit programmieren zu verbringen, sondern auch Zeit zu finden, die Projektdokumentation zu pflegen.

---

## AUSBLICK

---

An die bestehende Anwendung können noch weitere Funktionen in Zukunft hinzugefügt werden:



- Dazu zählen das zeitlich gesteuerte Suspendieren und Neustarten
- Auch kann die Funktionalität erweitert werden, indem man einzelne Instanzen je Projekt erzeugen oder löschen kann
- Möglich ist auch eine detailliertere Ansicht für die einzelnen Projekte und Instanzen
- Das Erstellen und Zuweisen von Volumes