API Dokumentation

Gruppe 4:	Benedikt Bakker <Benedikt.Bakker@stud.tu-darmstadt.de> Sascha Hans-Jürgen Dutschka <sascha.dutschka@arcor.de> Julian Harbarth <julian.harbarth@stud.tu-darmstadt.de> Sebastian Kropp <Sebastian.Kropp@stud.tu-darmstadt.de>
Teamleiter:	Benjamin Klein <klein.benjamin@gmail.com>
Auftraggeber:	Marc Werner <marc.werner@seemoo.tu-darmstadt.de> Secure Mobile Networking Lab - SEEMOO Fachbereich 20 <Informatik> Tobias Lange <tobias.lange@seemoo.tu-darmstadt.de> Secure Mobile Networking Lab - SEEMOO Fachbereich 20 <Informatik>
Stand:	30.03.2015

TECHNISCHE UNIVERSITÄT DARMSTADT

Bachelor-Praktikum WS 2014/2015
Fachbereich Informatik

Inhaltsverzeichnis

1 Dokumentation

1.1 Zustandsdiagramm

Das folgende Zustandsdiagramm zeigt die Zustandsübergänge für den Umgang mit einem Experiment. Zuerst wird gezeigt, welche Aktionen beim Anlegen des Experiments durchgeführt werden und dass sich das Experiment danach im Zustand ”Stopped” befindet. Danach erläutert das Diagramm, welche Zustandswechsel ein Experiment haben kann und welche weiteren Aktionen bei diesen Wechseln durchgeführt werden. Das Zustandsdiagramm zeigt, dass ein Experiment nur aus dem Zustand ”Stopped” auch wieder gelöscht werden kann.

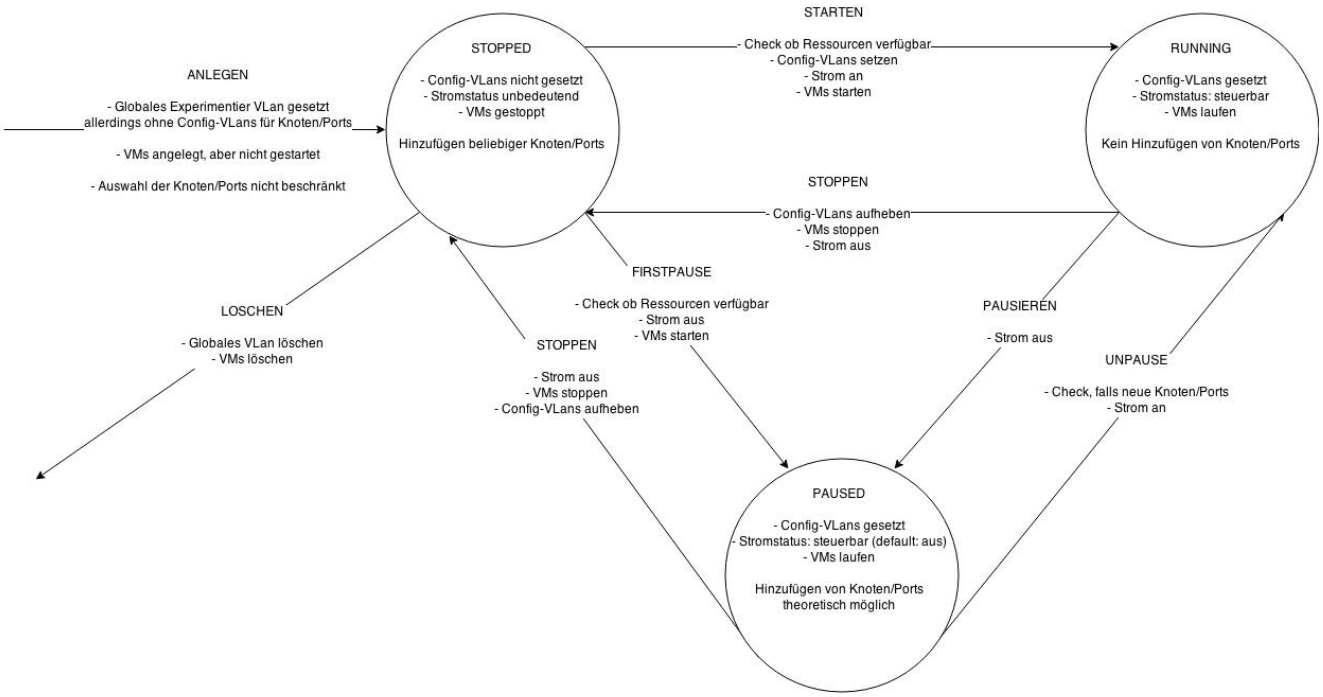

Abbildung 1.1: Zustandsdiagramm eines Experiments

1.2 Sequenzdiagramme

Folgender Abschnitt zeigt die Sequenzdiagramme, die für den Umgang mit Experimenten erstellt wurden. Es wird zuerst der Fall des Experiments anlegen gezeigt:

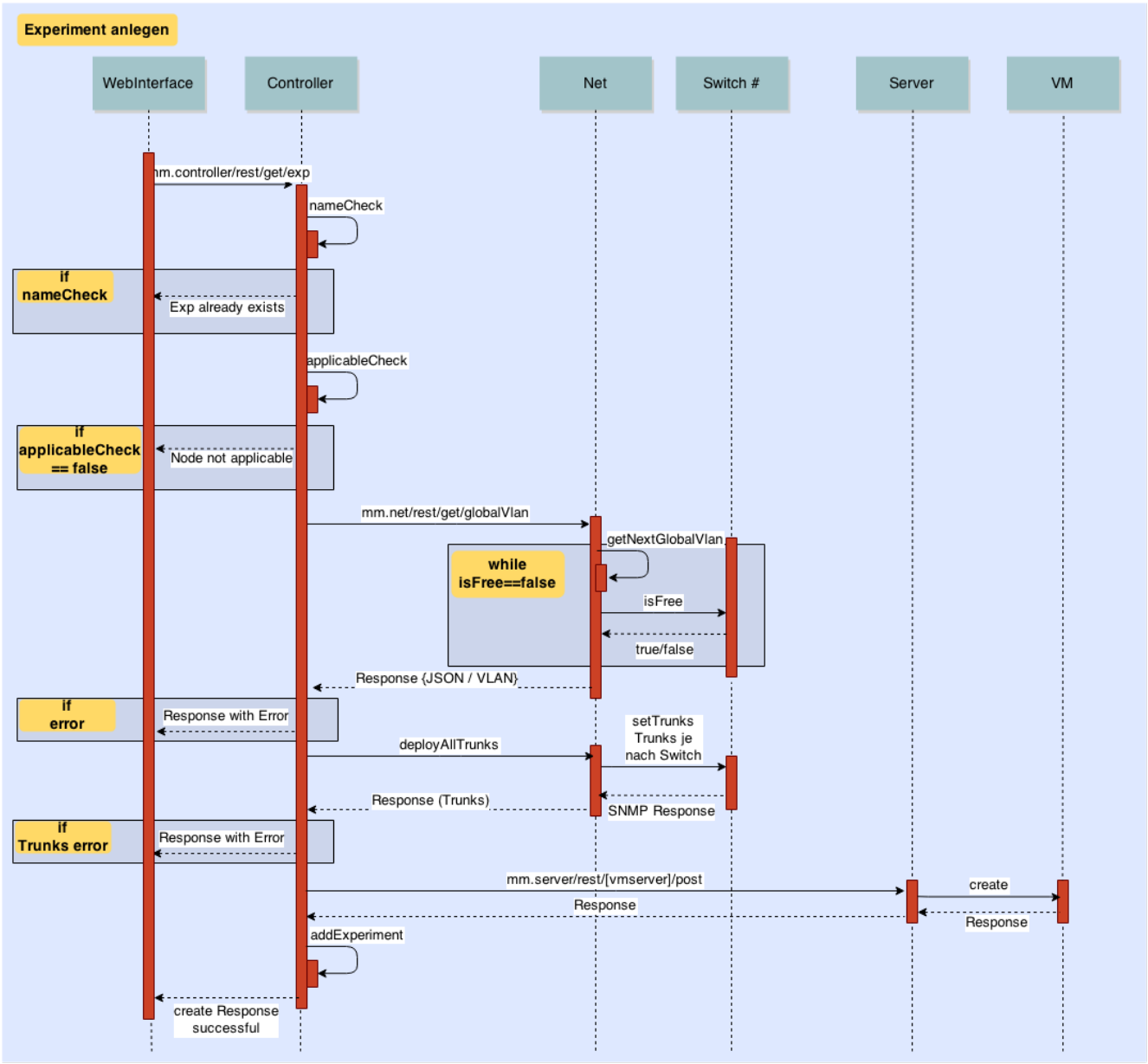

Abbildung 1.2: Experiment anlegen

Das nächste Diagramm zeigt den Anwendungsfall, wenn ein Experiment gestartet wird:

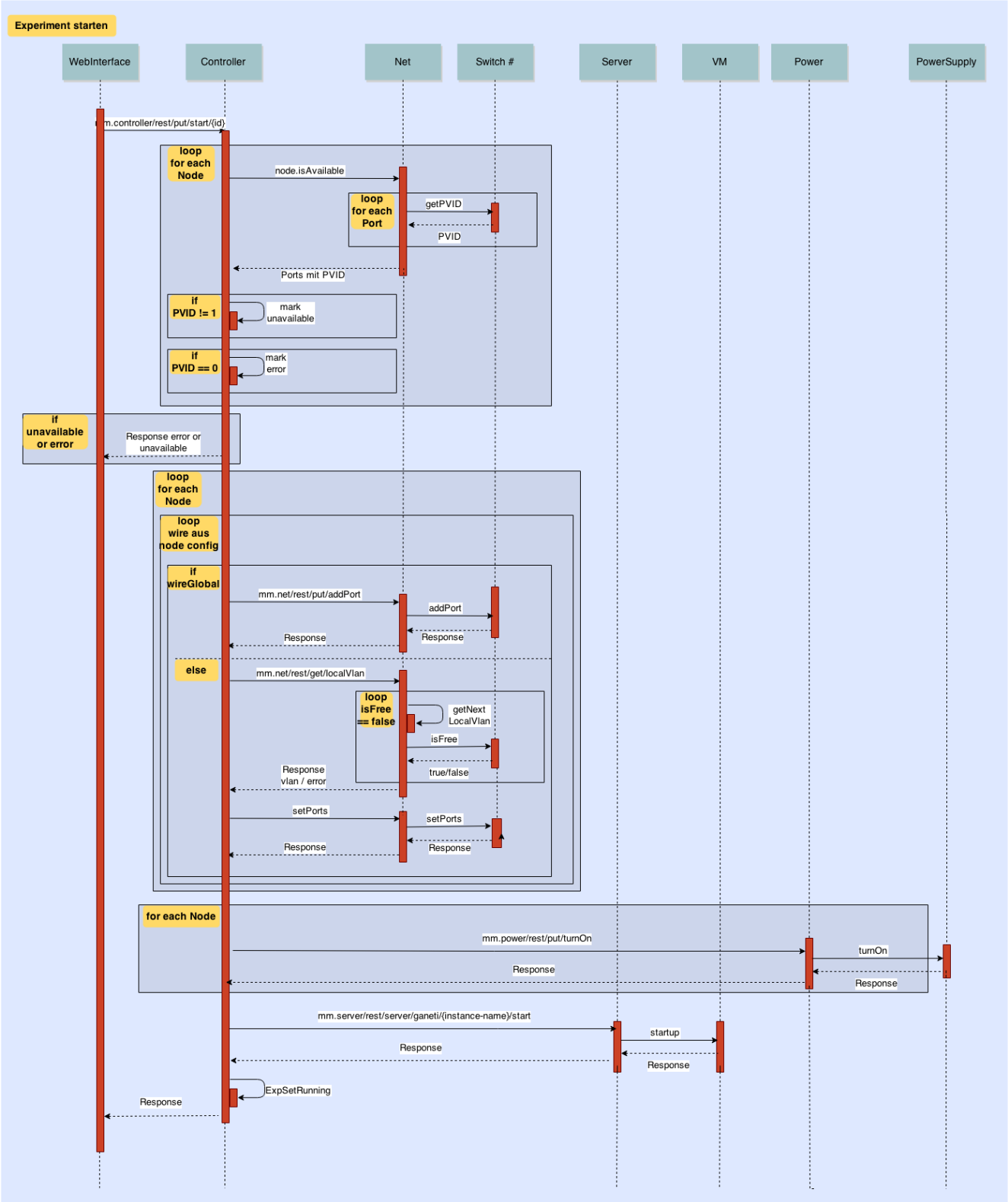

Abbildung 1.3: Experiment starten

Das folgende Diagramm zeigt den Anwendungsfall, wenn ein Experiment gestoppt wird:

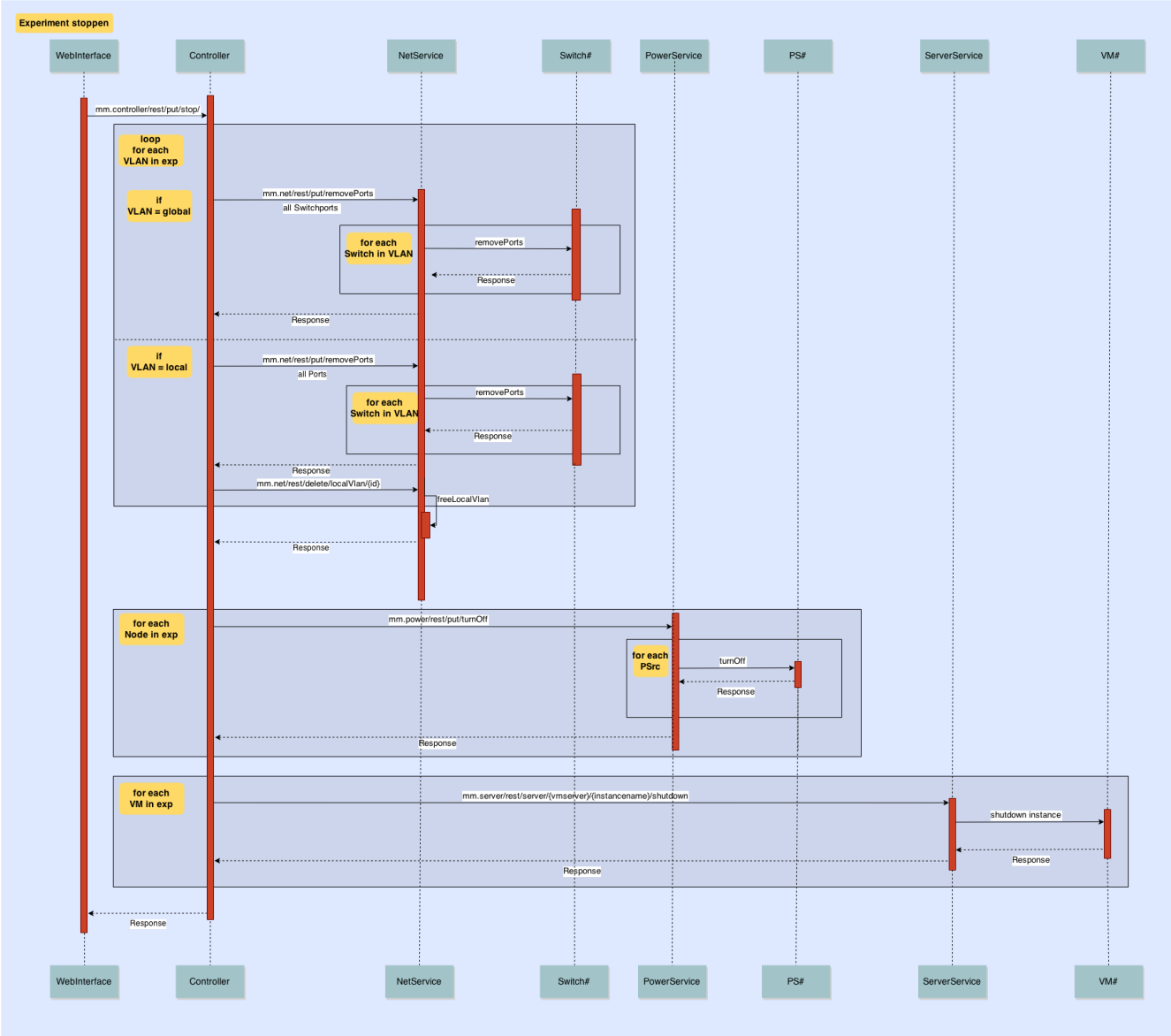

Abbildung 1.4: Experiment stoppen

Das kommende Diagramm zeigt den Anwendungsfall, wenn ein Experiment pausiert wird:

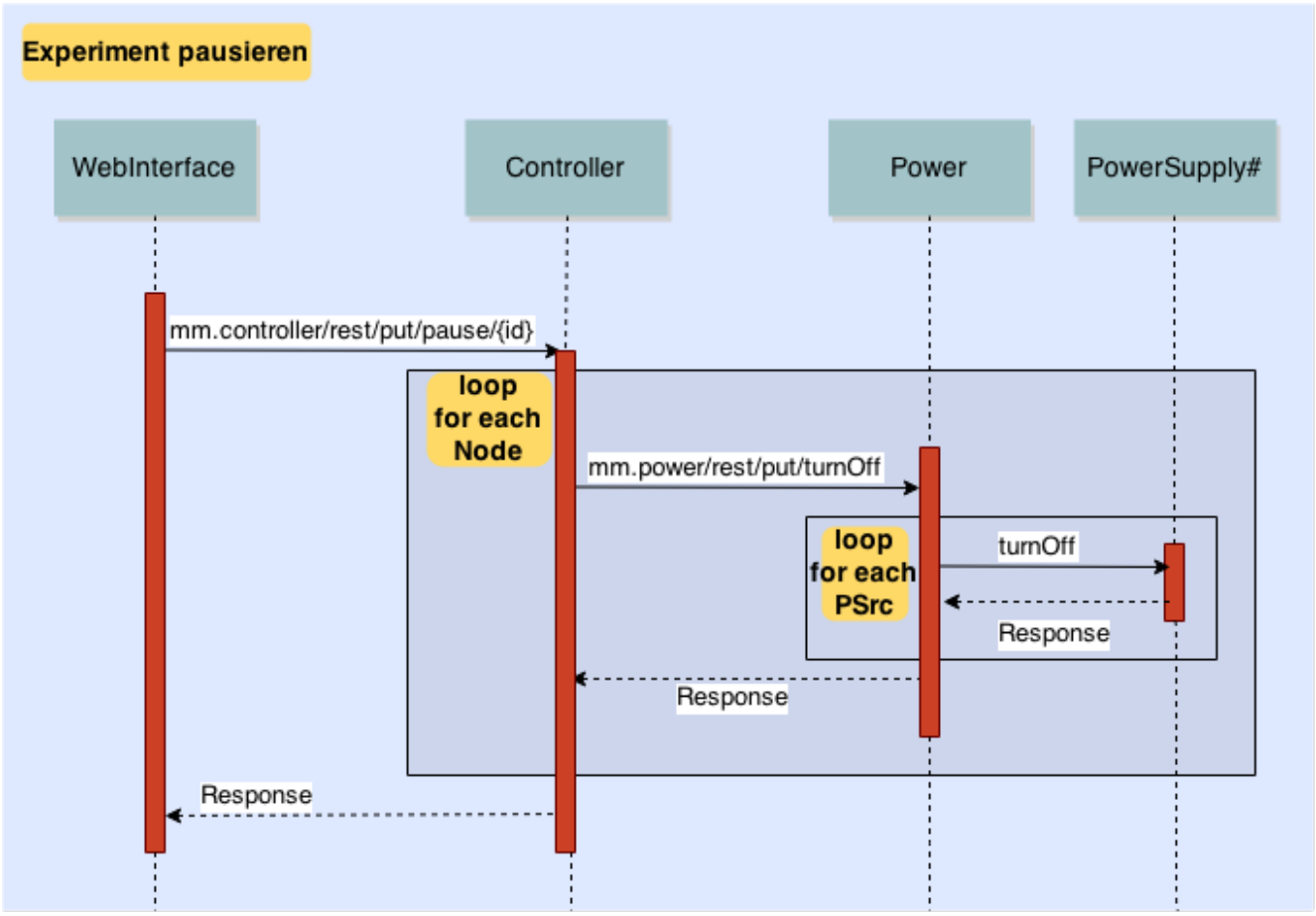

Abbildung 1.5: Experiment pausieren

Das letzte Diagramm zeigt den Anwendungsfall, wenn ein Experiment gelöscht wird:

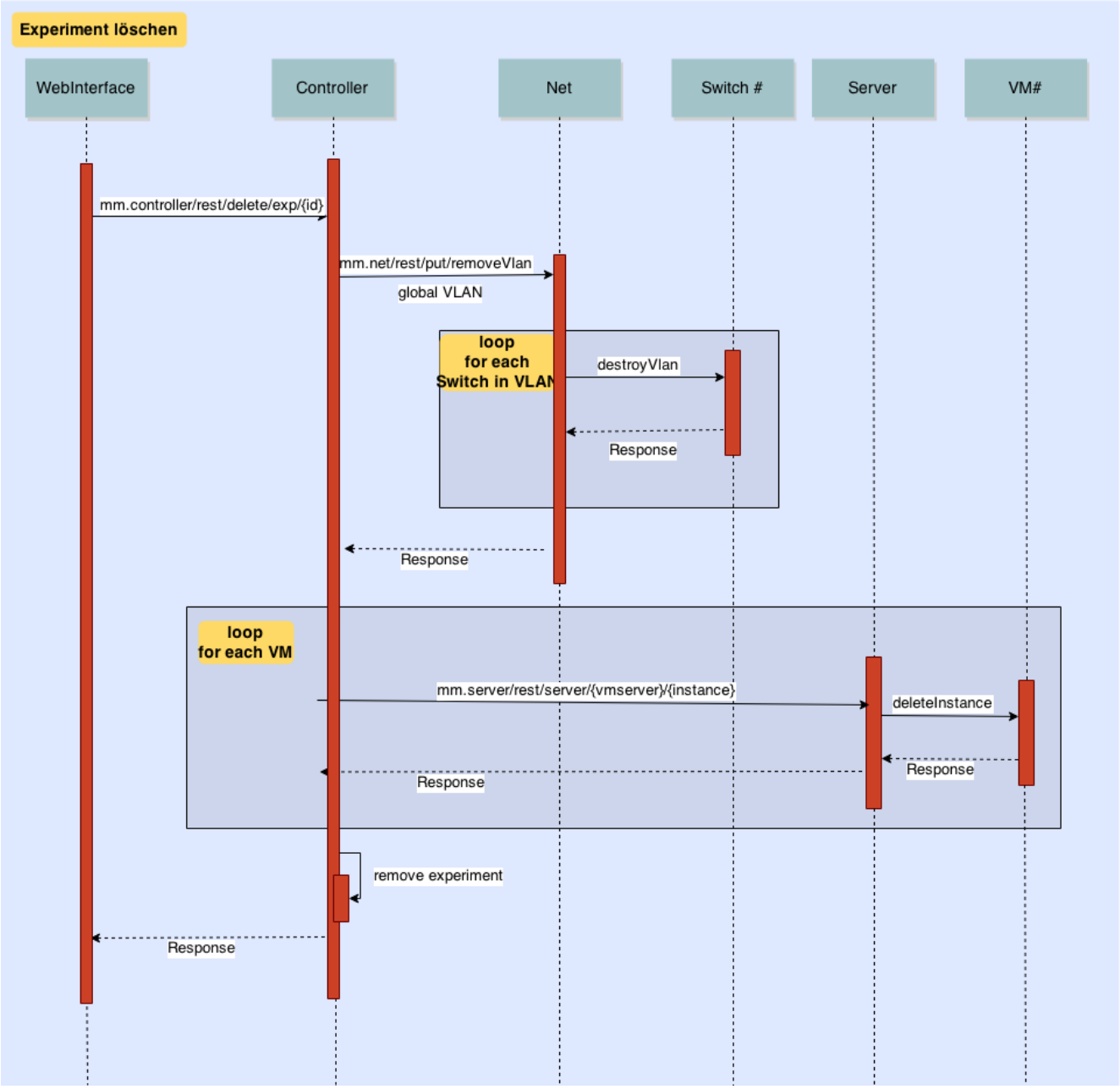

Abbildung 1.6: Experiment löschen

1.3 HTTP-Befehle

1.3.1 WebInterface

Das Webinterface dient der Bedienung der API und somit der Steuerung von Experimenten.
Es wurden hierbei AngularJS sowie Bootstrap verwendet und es ist als single-page Application aufgebaut.

/pages

- /index.html: Da das Webinterface als single-page aufgebaut ist, ist dies die einzige Seite, die man zunächst als Nutzer aufrufen muss.
- /login.html: Dies ist standardmäßig die Startseite und dient dem Login des Nutzers.
- /home.html: Diese Seite folgt nach dem Login und stellt die angelegten Experimente des eingeloggten Nutzer übersichtlich dar und bietet zu diesen Modifikationsmöglichkeiten.
- /newSetup.html: Auf dieser Seite kann der Nutzer ein neues Experiment konfigurieren.
- /editSetup.html: Auf dieser Seite kann der Nutzer ein gestopptes oder pausiertes Experiment editieren.
- /showExp.html: Auf dieser Seite kann der Nutzer ein aktives Experiment modifizieren, d.h. den Stromstatus aller Knoten des entsprechenden Experimentes steuern.
- /settings.html: Diese Seite dient der Konfiguration.
- /about.html: Information über das Projekt.

/css

/main.css: Diese Datei enthält die style-spezifischen Informationen

- Feste Leiste am unteren Bildschirmrand für Buttons.
- Container spezifische Größenanpassungen.
- Textstyle für die PPowerschalteräuf der /showExp.html Seite.

/img

Bilder für die aufklappbaren Listen

- collapsed.png
- expanded.png
- normal.png

/js

- jquery.min.js: JQuery-Bibliothek
- checklist-model.js: Eine Bibliothek für die Verknüpfung von Checkboxen mit einer Datenstruktur.
- extFunc.js: enthält folgende zwei Funktionen:
  - checkboxes: Erzeugt aus einer Listenstruktur verschachtelte Checkboxen mit entsprechender Aktivierungslogik.
  - toggleList: Erzeugt aufklappbare Listen.
- app.js: Enthält den AngularJS-code für das Webinterface (siehe folgenden Abschnitt).

app.js

.config:
Hier wird das Routing der Seiten gesteuert und die entsprechenden Controller zu jeder Seite spezifiziert.

.directive:
- nodes: Diese Direktive kompiliert die HTML-Struktur der Knoten-Daten.
- ports: Diese Direktive kompiliert die HTML-Struktur der Port-Daten.
- vms: Diese Direktive kompiliert die HTML-Struktur der VM-Daten.

.factory:
mainService: stellt für alle Controller übergreifende Funktionen zur Verfügung:

- verwaltet die SessionID.
- verwaltet das ausgewählte Experiment.
- getWelcomeMsg(): gibt die Willkommensnachricht zurück.
- getUsername(): gibt den eingeloggten Nutzer zurück.
- editSetup(): öffnet je nach Status des gewählten Experiment die /showExp.html oder /editSetup.html page.
- getSessionID(): gibt die SessionID zurück.
- setSessionID(id): setzt die id als SessionID.
- getSelectedExperimentID(): gibt die ID des ausgewählten Experiments zurück.
- getSelectedExperimentStatus(): gibt den Status des ausgewählten Experiments zurück.
- setSelectedExperiment(id, status): setzt die id und den status des ausgewählten Experiments.

.controller:

- mainController:
  - spezifiziert die URL der API.
  - Verwaltet die Willkommensnachricht (oben rechts).
- homeController:
  - radioChange(id, status): Verwaltet die Anzeige der Buttons auf der home-Seite, in Abhängigkeit des gewählten Experiments und dessen Status.
  - createSetupList(): Sendet eine Anfrage an die API und ruft die Liste der Experiment für den eingeloggten Nutzer auf.
  - startExperiment(): Sendet eine Anfrage an die API, dass ausgewählte Experiment zu starten.
  - pauseExperiment(): Sendet eine Anfrage an die API, dass ausgewählte Experiment zu pausieren.
  - unpauseExperiment(): Sendet eine Anfrage an die API, dass ausgewählte Experiment aus dem pausierten Zustand wieder zu starten.
  - stopExperiment(): Sendet eine Anfrage an die API, dass ausgewählte Experiment zu stoppen.
  - deleteExperiment(): Sendet eine Anfrage an die API, dass ausgewählte Experiment zu löschen.
  - callEditSetup(): ruft mainService.editSetup() auf.
- loginController:
  - callLogin(): Liest die Eingabefelder auf der Login-Seite aus und sendet eine enstprechende Login-Anfrage an die API. Bei erfolgreicher Antwort, wird die SessionID und der Nutzername in mainService gesetzt und auf die /home.html Seite weitergeleitet. Sollten die Login-Daten falsch sein, sendet die API den Fehlerstatus ”403” zurück. In allen anderen Fällen wird eine fehlende Verbindung zur API angenommen.
- newSetupController:
  - getPortList(): sendet eine Anfrage an die API, um die Liste aller verfügbaren Ports zu erhalten.
  - getVMList(): sendet eine Anfrage an die API, um die Liste aller verfügbaren VM-Templates zu erhalten.
  - getNodeList(): sendet eine Anfrage an die API, um die Liste aller verfügbaren Knoten zu erhalten.
  - createHTMLData(): Erzeugt die HTML-Struktur zur Anzeige der Knoten, Ports und VM-Templates, welche in den entsprechenden Direktiven dann kompiliert werden.
  - createNewExperiment(): Sendet eine Anfrage zur Anlegung eines neuen Experiments an die API: Dazu wird zunächst die Datenstruktur des Experiments initialisiert und dann mit folgenden Daten gefüllt:
    - name: der Name des Experiments.

    * nodes: die ausgewählten Knoten des Experiments.
    * wports: die ausgewählten Ports des Experiments.
    * vms: die ausgewählten VM-Templates des Experiments.
    * user: der eingeloggte Nutzer.

- editSetupController:
  - getPortList(): sendet eine Anfrage an die API, um die Liste aller verfügbaren Ports zu erhalten.
  - getVMList(): sendet eine Anfrage an die API, um die Liste aller verfügbaren VM-Templates zu erhalten.
  - getNodeList(): sendet eine Anfrage an die API, um die Liste aller verfügbaren Knoten zu erhalten.
  - createHTMLData(): Erzeugt die HTML-Struktur zur Anzeige der Knoten, Ports und VM-Templates, welche in den entsprechenden Direktiven dann kompiliert werden.
  - loadExperiment(): Läd die Daten eines Experiments von der API und aktualisiert die ID, sowie die Knoten-, Ports-, und VM-Datenstrukturen.
  - changePowerStatus(location, node, comp): Sendet eine Anfrage an die API, den Stromstatus einer Komponente comp, welche zu dem Knoten node gehört, welcher wiederum zu der Ort-Gruppierung location gehört, zu ändern. Diese Methode wird zur Zeit weder bei pausierten noch gestoppten Experimenten genutzt.
  - submitExperiment(): Sendet eine Anfrage an die API, dass entsprechende Experiment zu aktualisieren.

- showExpController:
  - changePowerStatus(node): Sendet eine Anfrage an die API, den Stromstatus eine Knoten node zu ändern.
  - powerAllNodes(power): Sendet eine Anfrage an die API, den Stromstatus aller Knoten des aktiven, betrachteten Experiments zu ändern. Power ist hierbei entweder ”Off” oder ”On”, je nachdem, welcher Button diese Funktion aufgerufen hat.
  - loadExperiment(): Läd das aktive Experiment von der API und aktualisiert die Tabelle aller Knoten inklusive deren Stromstatus.

1.3.2 Controller

/mm.controller/rest/delete/exp/[id]

- Delete

Delete

Löscht ein Experiment.

Parameter:

- PathParam id: Die eindeutige ID des Experiments, die sich aus Usernamen und Experimentnamen zusammensetzt (user + name).

Precondition:

- Das Experiment mit der übergebenen ID existiert.
- Das Experiment ist im Zustand ”stopped”.

Postcondition:

- Das Experiment wird aus der Liste aller Experimente gelöscht, damit verschwinden alle Referenzen auf das Objekt. Es wird nun dem Benutzer nicht mehr angezeigt.
- Die zum Experiment zugehörigen VMs werden vom entsprechenden VM Server gelöscht.
- Das zum Experiment gehörigen globale Experimentier VLAN ist von allen beteiligten Netzwerkkomponenten gelöscht.

/mm.controller/rest/get/nodes

- Get

Get

Liefert eine Liste mit allen Knoten.

Parameter:

- HeaderParam sessionId: Eine gültige sessionId vom AuthService im Headerfeld ”sessionId”.

Precondition:

- Gültige Session ID eines Benutzers.

Postcondition:

- Eine Liste von Knoten wurde zurückgegeben.

Beispielrückgabe: Liste mit einem Knoten:


```
[
{
  ”id”: ”Node A”,
  ”typeName”: ”APU+WARP”,
  ”components”: [
  {
    ”type”: ”WARP”,
    ”interfaces”: [
    {
      ”name”: ”eth0”,
      ”switchport”: ”NetGear2;2”,
      ”role”: ”WARP A”,
```

```
      },
      {
        ”name”: ”eth1”,
        ”switchport”: ”NetGear2;3”,
        ”role”: ”WARP B”,
      }
      ],
      ”powerSource”: ”AeHome1;3”,
    },
    {
      ”type”: ”APU”,
      ”interfaces”: [
      {
        ”name”: ”eth2”,
        ”switchport”: ”NetGear2;4”,
        ”role”: ”APU 1”,
      },
      {
        ”name”: ”eth3”,
        ”switchport”: ”NetGear2;5”,
        ”role”: ”APU 2”,
      },
      {
        ”name”: ”eth4”,
        ”switchport”: ”NetGear2;6”,
        ”role”: ”APU 3”,
      }
      ],
      ”powerSource”: ”AeHome1;1”,
    }
    ],
    ”building”: ”Building 1”,
    ”room”: ”Room right”,        ”trunk”: ”NetGear2;1”     }
  ]
```


/mm.controller/rest/get/configs

- Get

Get

Liefert eine Liste mit allen Konfigurationen.

Parameter:

- HeaderParam sessionId: Eine gültige sessionId vom AuthService im Headerfeld ”sessionId”.

Precondition:

- Gültige Session ID eines Benutzers.

Postcondition:

- Eine Liste von Konfigurationen wurde zurückgegeben.

Beispielrückgabe: List mit zwei Konfigurationen


```
[
{
  ”name”: ”WARP+APU”,
  ”wires”: [
  {
    ”endpoints”: [
      ”APU 2”,
      ”WARP B”
    ],
  },
  {
    ”endpoints”: [
      ”APU 3”,
      ”WARP A”
    ]
  }
  ]
},
{
  ”name”: ”WARP”,
  ”wires”: [
  {
    ”endpoints”: [
      ”WARP A”,
      ”WARP B”
    ]
  }
  ]
}
]
```


/mm.controller/rest/get/ports

- Get

Get

Liefert eine Liste mit allen Ports.

Parameter:

- HeaderParam sessionId: Eine gültige sessionId vom AuthService im Headerfeld ”sessionId”.

Precondition:

- Gültige Session ID eines Benutzers.

Postcondition:

- Eine Liste von Ports wurde zurückgegeben.

Beispielrückgabe: List mit zwei Ports

```
[
{
  ”id”: ”testport”,
  ”port”: ”netgear1;2”,
  ”building”: ”building1”,
  ”room”: ”room1”,
  ”trunk”: ”netgear1;1”
},
{
  ”id”: ”testport1”,
  ”port”: ”netgear1;3”,
  ”building”: ”building1”,
  ”room”: ”room1”,
  ”trunk”: ”netgear1;1”
}
]
```

/mm.controller/rest/get/[exp]

- Get

Get

Liefert eine Übersicht über das Experiment mit aktuellen Informationen bezüglich des Stromstatus. Jede Komponente besitzt ein Status Feld welches den Stromstatus signalisiert. Jeder Knoten besitzt ebenfalls ein Statusfeld, welches auf ”true” gesetzt ist, sobald mindestens eine Komponente dieses Knotens ebenfalls den Status ”true” hält. Das Experiment wird im JSON Format übertragen.

Parameter:

- PathParam id : Die eindeutige ID des Experiments, die sich aus Usernamen und Experimentnamen zusammensetzt (user + name).
- HeaderParam sessionId: Eine gültige sessionId vom AuthService im Headerfeld ”sessionId”.

Precondition:

- Gültige Session ID eines Benutzers.
- Eine Experiment ID, die existiert wurde übergeben.

Postcondition:

- Ein Experiment im JSON Format.

Beispielrückgabe: Ein Beispiel mit einem Knoten, mit zwei Komponenten, und Status Informationen


```
[
{
  ”id”: ”Node A”,
  ”typeName”: ”APU+WARP”,
  ”components”: [      {
    ”type”: ”WARP”,
    ”interfaces”: [
    {
      ”name”: ”eth0”,
      ”switchport”: ”NetGear2;2”,
      ”role”: ”WARP A”
    },
    {
      ”name”: ”eth1”,
      ”switchport”: ”NetGear2;3”,
      ”role”: ”WARP B”
    }
    ],
    ”status”: false,
    ”powerSource”: ”AeHome1;3”
  },
  {
    ”type”: ”APU”,
    ”interfaces”: [
    {
      ”name”: ”eth2”,
      ”switchport”: ”NetGear2;4”,
      ”role”: ”APU 1”
    },
    {
      ”name”: ”eth3”,
      ”switchport”: ”NetGear2;5”,
      ”role”: ”APU 2”
    },
    {
      ”name”: ”eth4”,
      ”switchport”: ”NetGear2;6”,
      ”role”: ”APU 3”
    }
    ],
    ”status”: true,
    ”powerSource”: ”AeHome1;1”
  }
  ],
```

```
  "building": "Building 1",
  "room": "Room right",
  "status": true,
  "trunk": "NetGear2;1"
}
]
```


/mm.controller/rest/get/staticConsistency/net

- Get

Get

Überprüft die Konsistenz eines statischen VLANs. Statische VLANs sind solche, die während des Initialisieren des NetServices angelegt werden und nicht vom Benutzer veränderbar sind. Zur Zeit gibt es zwei statische Netze. Das Power Netz und das Management Netz.

Parameter:

- PathParam net: Kann einen der zwei Werte annehmen: "power" oder "management".
- HeaderParam sessionId: Eine gültige sessionId vom AuthService im Headerfeld "sessionId".

Precondition:

- Gültige Session ID eines Benutzers.

Postcondition:

- Die Konfigurationen der statischen VLANs werden überprüft.
- Eine Rückmeldung enthält eine Aufschlüsselung je nach Netzwerkkomponente ob ein VLAN den erwarteten Konfigurationen entspricht.

/mm.controller/rest/get/expConsistency/id

- Get

Get

Überprüft die Konsistenz eines Experimentes. Je nach Status werden verschiedene VLANs überprüft. Bei einem gestoppten Experiment wird das globale Experimentiernetz überprüft, ob es mit dem erwarteten Zustand übereinstimmt. Das Experimentiernetz sollte alle Trunkports aller Knoten und Ports mit dem VM Server verbinden.
Sollte das Experiment im Status "running" oder "pausiert" sein so werden zusätzlich noch die lokalen VLANs überprüft.

Parameter:

- PathParam id : Die eindeutige ID des Experiments, die sich aus Usernamen und Experimentnamen zusammensetzt (user + name).
- HeaderParam sessionId: Eine gültige sessionId vom AuthService im Headerfeld "sessionId".

Precondition:

- Gültige Session ID eines Benutzers.
- Ein Experiment mit der übergebenen ID existiert auf dem Server.

Postcondition:

- Die VLANs des Experiments wurden entsprechend des Status überprüft.
- Eine Rückmeldung zum Konsistenzstatus wurde übergeben.

/mm.controller/rest/get/exp

- Get

Get
Liefert eine Liste aller Experimente des eingeloggten Users. Dieser wird aus der Session ID abgelesen. Sollte der Benutzer Administrator Rechte besitzen werden alle existierenden Experimente angezeigt.

Parameter:

- HeaderParam sessionId: Eine gültige sessionId vom AuthService im Headerfeld ”sessionId”.

Precondition:

- Gültige Session ID eines Benutzers.

Postcondition:

- Eine Liste mit allen Experimenten des Nutzers wird zurückgeliefert oder eine komplette Liste aller existenten Experimente bei entsprechenden Rechten.

/mm.controller/rest/put/turnOff

- Put

Put
Schaltet alle Komponenten eines Knoten aus.

Parameter:

- data: Die eindeutige ID des Knoten im Body der HTTP Anfrage.

Precondition:

- Gültige Session ID eines Benutzers.
- Übergebene Knoten ID, die dem Controller bekannt ist, mit dazugehöriger Stromquelleninformation.

Postcondition:

- Die Komponenten des Knotens werden nicht mit Strom versorgt.

/mm.controller/rest/put/turnOn

- Put

Put
Schaltet alle Komponenten eines Knoten an.
Parameter:

- data: Die eindeutige ID des Knoten im Body der HTTP Anfrage.

Precondition:

- Gültige Session ID eines Benutzers.
- Übergebene Knoten ID, die dem Controller bekannt ist, mit dazugehöriger Stromquelleninformation.

Postcondition:

- Die Komponenten des Knotens werden mit Strom versorgt.

/mm.controller/rest/put/turnOff/[comp]

- Put

Put
Schaltet die übergebene Komponente eines Knoten aus.
Parameter:

- data: Die eindeutige ID des Knoten im Body der HTTP Anfrage.
- PathParam comp: Der Name einer Komponente.

Precondition:

- Gültige Session ID eines Benutzers.
- Übergebene Knoten ID, die dem Controller bekannt ist, mit dazugehöriger Stromquelleninformation einer Komponente.

Postcondition:

- Die Komponenten des Knotens werden nicht mit Strom versorgt.

/mm.controller/rest/put/turnOn/[comp]

- Put

Put
Schaltet die übergebene Komponente eines Knoten an.

Parameter:

- data: Die eindeutige ID des Knoten im Body der HTTP Anfrage.
- PathParam comp: Der Name einer Komponente.

Precondition:

- Gültige Session ID eines Benutzers.
- Übergebene Knoten ID, die dem Controller bekannt ist, mit dazugehöriger Stromquelleninformation einer Komponente.

Postcondition:

- Die Komponenten des Knotens werden mit Strom versorgt.

/mm.controller/rest/put/stop

- Put

Put
Stoppt das Experiment mit der übermittelten ID.
Parameter:

- expId: Die eindeutige ID eines Experimentes im Message Body. Die ID setzt sich aus Usernamen und Experimentnamen zusammensetzt (user+name).

Precondition:

- Gültige Session ID eines Benutzers.
- Das Experiment mit der übergebenen ID existiert.
- Das Experiment ist im Zustand ”running” oder ”paused”.

Postcondition:

- Die lokalen Konfigurationen der VLANs der Knoten im Experiment sind gelöscht, damit sind die Knoten wieder freigegeben.
- Die Virtuellen Maschinen, die dem Experiment zugeordnet sind, sind gestoppt.
- Die Knoten sind ausgeschaltet.

/mm.controller/rest/put/start

- Put

Put
Startet das Experiment mit der übermittelten ID.

Parameter:

- exp: Die eindeutige ID eines Experimentes im Message Body. Die ID setzt sich aus Usernamen und Experimentnamen zusammensetzt (user+name).

Precondition:

- Gültige Session ID eines Benutzers.
- Die Knoten, die dem Experiment zugeordnet sind, sind nicht in einem anderen laufenden Experiment.
- Das Experiment ist im Zustand ”stopped”.

Postcondition:

- Die Stromversorgung der Knoten ist eingeschaltet.
- Die Virtuellen Maschinen des Experimentes wurden gestartet.
- Die lokalen Konfigurationen der VLANs wurden gesetzt und die Knoten sind mit dem VM Server über ein globales Experimentier-VLAN verbunden.

/mm.controller/rest/put/pause

- Put

Put

Pausiert das Experiment mit der übermittelten ID.

Parameter:

- exp: Die eindeutige ID eines Experimentes im Message Body. Die ID setzt sich aus Usernamen und Experimentnamen zusammensetzt (user+name).

Precondition:

- Gültige Session ID eines Benutzers.
- Das Experiment ist im Status ”running” oder ”stopped”.
- Falls Experiment im Status ”stopped”: Knoten, die dem Experiment zugeordnet sind, sind keinem weiteren Experiment zugeordnet.

Postcondition:

- Die Stromversorgung der Knoten ist ausgeschaltet.
- Falls Status war ”stopped”: Die lokalen Konfigurationen der VLANs sind erstellt und konfiguriert. Das globale Experimentiernetz verbinden die Knoten mit dem VM Server.

/mm.controller/rest/put/unpause

- Put

Put

Aktiviert ein pausiertes Experiment mit der übermittelten ID, aber es wird nicht in den Zustand ”running” gesetzt.

Parameter:

- exp: Die eindeutige ID eines Experimentes im Message Body. Die ID setzt sich aus Usernamen und Experimentnamen zusammensetzt (user+name).

Precondition:

- Gültige Session ID eines Benutzers.
- Das Experiment ist im Status ”paused”.

Postcondition:

- Alle Stromversorgungen aller Knoten des Experimentes sind eingeschaltet.

/mm.controller/rest/put/exp

- Put

Put

Erstellt ein neues Experiment.

Parameter:

- data: Ein Experiment als JSON.

Precondition:

- Gültige Session ID eines Benutzers.
- Der Name des Experimentes unter diesem Benutzer ist noch nicht vergeben.
- Alle Knoten sind fähig, die ihnen zugewiesene Konfiguration anzunehmen.

Postcondition:

- Ein globales Experimentiernetz mit allen Trunkports aller Knoten, die Mitglied des Experimentes sind, wurde erstellt.
- Alle Virtuellen Maschinen, die im Experiment angegeben wurden, wurden erstellt.
- Ein Experiment wurde im Controller angelegt und kann nun vom Benutzer manipuliert werden.

1.3.3 Auth-Service

/mm.auth/rest/session/createSession/[user]/[pw]

- Get

Get

Erstellt eine gültige Session ID für einen authentifizierten User.
Parameter:

- PathParam user: Der Name des Benutzers.
- PathParam pw: Das Passwort des Benutzers.

Precondition:

- Gültiger Name des Benutzers.
- Gültiges Passwort zu dem Benutzer.

Postcondition:

- Eine gültige Session ID wurde für den Benutzer angelegt.
- Die Session ID hat ein maximale Gültigkeitsspanne.

/mm.auth/rest/session/validation

- Get

Get

Überprüft, ob eine übergebene Session ID noch gültig ist.
Parameter:

- HeaderParam sessionId: Die zu überprüfende Session ID.

Precondition:

- Gültige Session ID eines Benutzers.

Postcondition:

- Sollte die Session ID noch gültig sein, wird der HTTP Status 200 als Response geschickt.
- Sollte die Session ID abgelaufen sein, wird der HTTP Status 408 als Response geschickt.
- Sollte der Header keine Session ID enthalten, wir der HTTP Status 401 als Response geschickt.

/mm.auth/rest/session/getRole

- Get

Get

Liefert die Rolle eines Benutzer anhand der Session ID als HTTP Response.
Parameter:

- HeaderParam sessionId: Die zu überprüfende Session ID.

Precondition:

- Gültige Session ID eines Benutzers.

Postcondition:

- Sollte die Session ID noch gültig sein, wird der HTTP Status 200 mit der Rolle als Response geschickt.
- Sollte der Header keine Session ID enthalten, wir der HTTP Status 401 als Response geschickt.

/mm.auth/rest/session/getUser

- Get

Get
Liefert den Namen eines Benutzer anhand der Session ID als HTTP Response.
Parameter:

- HeaderParam sessionId: Die zu überprüfende Session ID.

Precondition:

- Gültige Session ID eines Benutzers.

Postcondition:

- Sollte die Session ID noch gültig sein, wird der HTTP Status 200 mit dem Benutzernamen als Response geschickt.
- Sollte der Header keine Session ID enthalten, wird der HTTP Status 401 als Response geschickt.

1.3.4 Net-Service

/mm.net/rest/get/vlanStatus/[incoming]

- Get

Get
Liefert eine Liste von Interfaces mit Ports und deren PV-IDs.
Parameter:

- incoming: Eine Liste von Ports.

Precondition:

- Gültige Liste von Ports.

Postcondition:

- Sollte die Methode erfolgreich ausgeführt worden sein, wird der HTTP Status 200 mit einer Liste von Interfaces mit Ports und deren PV-ID geschickt.
- Sollte der Datentransfer zu der Netzwerkkomponente nicht vollständig sein, wird der HTTP Status 400 geschickt.
- Sollte die Netzwerkkomponente nicht existieren, wird der HTTP Status 404 geschickt.
- Sollte ein interner Fehler produziert worden sein, wird der HTTP Status 500 geschickt.

/mm.net/rest/get/globalVlan

- Get

Get
Überprüft ob das globale VLAN auf der Netzwerkkomponente frei ist.
Postcondition:

- Sollte das zu überprüfende globale VLAN frei sein, wird der HTTP Status 200 mit VLAN geschickt.
- Sollte das zu überprüfende globale VLAN belegt sein, wird der HTTP Status 404 geschickt.

/mm.net/rest/get/localVlan

- Get

Get
Überprüft ob das lokal VLAN auf der Netzwerkkomponente frei ist.
Postcondition:

- Sollte das zu überprüfende lokale VLAN frei sein, wird der HTTP Status 200 mit VLAN geschickt.
- Sollte das zu überprüfende lokale VLAN belegt sein, wird der HTTP Status 404 geschickt.

/mm.net/rest/get/consistency/[enc]

- Get

Get
Überprüft ob die Daten eines übergebenem VLAN konsistent sind.
Parameter:

- PathParam enc: Ein Base64 codiertes VLAN.

Postcondition:

- Es wird der HTTP Status 200 geschickt für die erfolgreiche Abfrage mit der Response Nachricht, ob es konsistent ist.

/mm.net/rest/delete/globalVlan/[id]

- Delete

Delete

Löscht ein globales VLAN auf allen Netzwerkkomponenten und gibt es wieder frei.

Parameter:

- PathParam id: Die ID des zu löschenden globalem VLAN.

Postcondition:

- Es wird der HTTP Status 200 geschickt für das erfolgreiche Löschen des VLANs.
- Es wird der HTTP Status 500 geschickt, falls das Löschen nicht erfolgreich war.

/mm.net/rest/delete/globalVlan/[id]

- Delete

Delete

Löscht ein lokales VLAN auf allen Netzwerkkomponenten und gibt es wieder frei.

Parameter:

- PathParam id: Die ID des zu löschenden lokalem VLAN.

Postcondition:

- Es wird der HTTP Status 200 geschickt für das erfolgreiche Löschen des VLANs.
- Es wird der HTTP Status 500 geschickt, falls das Löschen nicht erfolgreich war.

/mm.net/rest/put/setTrunkPort

- Put

Put

Setzt die Ports eines übergebenen VLAN als Trunks. Falls das VLAN schon existiert, wird es gelöscht und neu erstellt.

Parameter:

- incoming: Ein VLAN als JSONObject.

Postcondition:

- Es wird der HTTP Status 200 geschickt für das erfolgreiche Setzen der Trunkports.
- Es wird der HTTP Status 500 geschickt, falls das Setzen nicht erfolgreich war.

/mm.net/rest/put/addTrunkPort

- Put

Put

Fügt die Ports eines übergebenen VLAN als Trunkports hinzu. Ports die dem VLAN schon angehören werden mit der übergebenen Konfiguration des VLANs ersetzt.

Parameter:

- incoming: Ein VLAN als JSONObject.

Postcondition:

- Es wird der HTTP Status 200 geschickt für das erfolgreiche Hinzufügen der Trunkports.
- Es wird der HTTP Status 500 geschickt, falls das Hinzufügen nicht erfolgreich war.

/mm.net/rest/put/setPort

- Put

Put

Setzt die Ports eines übergebenen VLANs als unmarkierte Mitglieder des VLANs.

Parameter:

- incoming: Ein VLAN als JSONObject.

Postcondition:

- Es wird der HTTP Status 200 geschickt für das erfolgreiche Setzen der Ports.
- Es wird der HTTP Status 500 geschickt, falls das Setzen nicht erfolgreich war.

/mm.net/rest/put/addPort

- Put

Put

Fügt die Ports eines übergebenen VLANs als unmarkierte Mitglieder des VLANs hinzu.

Parameter:

- incoming: Ein VLAN als JSONObject.

Postcondition:

- Es wird der HTTP Status 200 geschickt für das erfolgreiche Hinzufügen der Ports.
- Es wird der HTTP Status 500 geschickt, falls das Hinzufügen nicht erfolgreich war.

/mm.net/rest/put/removePort

- Put

Put
Löscht die Ports eines übergebenen VLANs.
Parameter:

- incoming: Ein VLAN als JSONObject.

Postcondition:

- Es wird der HTTP Status 200 geschickt für das erfolgreiche Löschen der Ports.
- Es wird der HTTP Status 500 geschickt, falls das Löschen nicht erfolgreich war.

/mm.net/rest/put/removeVlan

- Put

Put
Löscht das übergebenen VLAN.
Parameter:

- incoming: Ein VLAN als JSONObject.

Postcondition:

- Es wird der HTTP Status 200 geschickt für das erfolgreiche Löschen des VLANs.
- Es wird der HTTP Status 500 geschickt, falls das Löschen nicht erfolgreich war.

1.3.5 Power-Service

/mm.power/rest/get/[incoming]

- Get

Get
Liefert den Status einer Stromquelle anhand der ID als HTTP Response.
Parameter:

- PathParam incoming: Die ID der Stromquelle.

Precondition:

- Gültige ID einer Stromquelle.

Postcondition:

- Sollte die Methode erfolgreich ausgeführt worden sein, wird der HTTP Status 200 mit einer Liste der Informationen bzgl. der Stromquelle geschickt.
- Sollte die ID der Stromquelle nicht existieren, wird der HTTP Status 404 geschickt.
- Sollte der Datentransfer von der Stromquelle nicht vollständig sein, wird der HTTP Status 500 geschickt.

/mm.power/rest/put/turnOn

- Put

Put

Schaltet die Stromversorgung einer Stromquelle an.

Parameter:

- incoming: Eine Liste mit Stromquelle und deren Socket.

Precondition:

- Die Stromquelle existiert.

Postcondition:

- Sollte die Methode erfolgreich ausgeführt worden sein, wird der HTTP Status 200 geschickt.
- Sollte die Stromquelle nicht existieren, wird der HTTP Status 404 geschickt.

/mm.power/rest/put/turnOff

- Put

Put

Schaltet die Stromversorgung einer Stromquelle aus.

Parameter:

- incoming: Eine Liste mit Stromquelle und deren Socket.

Precondition:

- Die Stromquelle existiert.

Postcondition:

- Sollte die Methode erfolgreich ausgeführt worden sein, wird der HTTP Status 200 geschickt.
- Sollte die Stromquelle nicht existieren, wird der HTTP Status 404 geschickt.

1.3.6 Server-Service

In den folgenden HTTP Befehlen wird Ganeti als VM-Server genutzt, daher steht in dem HTTP Aufruf auch ”ganeti”. Der Controller kann anhand einem dementsprechenden übergebenden String auch mit anderen VM-Servern kommunizieren.

/mm.server/rest/server/ganeti

- Get
- Post

Get
Liefert eine Liste der VM’s, die auf dem Ganeti-Server angelegt sind.
Beispielrückgabe: Liste mit drei VM’s
[instance1.seemoo.tu-darmstadt.de, instance2.seemoo.tu-darmstadt.de, instance3.seemoo.tu-darmstadt.de]
Postcondition:

- Liste der angelegten Instanzen wurde zurückgeschickt.

Post
Erstellt eine neue VM auf dem Ganeti-Server.
Parameter: Ein JSONObject mit folgenden Attributen:

- size: Die Größe der Festplatte für die Instanz.
- template: Der Name des Templates, welche die Einstellungen für die Instanz enthält.
- bridge: Der Name der Netzwerkbrücke.
- name: Der Name für die zu anlegende Instanz.
- ip: Die gewünschte IP-Adresse, kann auch leer bleiben.

Precondition:

- JSONObject enthält alle benötigten Parameter.

Postcondition:

- Instanz wurde angelegt.

/mm.server/rest/server/ganeti/[instance-name]

- Delete
- Post

Delete
Löscht die ausgewählte VM.
Parameter:

- PathParam instance-name: Der Name der Instanz, die gelöscht werden soll.

Precondition:

- Die zu löschende Instanz existiert.

Postcondition:

- Instanz wurde gelöscht.

Post
Startet die ausgewählte VM neu.
Parameter:

- PathParam instance-name: Der Name der Instanz, die gelöscht werden soll.
- type: Muss entweder soft, hard oder full sein.

Precondition:

- Die ausgewählte Instanz existiert.

Postcondition:

- Instanz wurde neu gestartet.

/mm.server/rest/server/ganeti/[instance-name]/start

- Put

Put
Startet die ausgewählte VM.
Parameter:

- PathParam instance-name: Der Name der Instanz, die gelöscht werden soll.
- type: Ein JSONObject mit gewünschten Einstellungen, kann auch leer sein.

Precondition:

- Die ausgewählte Instanz existiert.

Postcondition:

- Instanz wurde gestartet.

/mm.server/rest/server/ganeti/[instance-name]/stop

- Put

Put
Stopt die ausgewählte VM.
Parameter:

- PathParam instance-name: Der Name der Instanz, die gelöscht werden soll.
- type: Ein JSONObject mit gewünschten Einstellungen, kann auch leer sein.

Precondition:

- Die ausgewählte Instanz existiert.

Postcondition:

- Instanz wurde gestoppt.

/mm.server/rest/server/ganeti/[instance-name]/rename

- Put

Put
Nennt die ausgewählte VM um.
Paramter:

- PathParam instance-name: Der Name der Instanz, die gelöscht werden soll.
- newName: Der neue Name der ausgewählten VM.

Precondition:

- Die ausgewählte Instanz existiert.
- Der neue Name darf nicht leer sein.

Postcondition:

- Instanz wurde umbenannt.

/mm.server/rest/server/template

- Get
- Put

Get
Liefert eine Liste von Templates der Instanzen aus einer XML Datei inklusiver aller Attribute.
Postcondition:

- Liefert eine Liste der angelegten Templates.

Beispielrückgabe: Ein Beispiel mit drei Templates.


```
[
{
  ”mode”: ”create”,
  ”server”: ”ganeti”,
  ”disk_template”: ”plain”,
  ”ip_check”: false,
  ”name_check”: false,
  ”os_type”: ”debootstrap+wheezy”,
  ”start”: false,
  ”nics”: [
  {
    ”mode”: ”bridged”
  }
  ],
  ”__version__”: 1,
  ”conflicts_check”: false,
  ”id”: ”Instanz1”,
  ”pnode”: ”pxhost01.seemoo.tu-darmstadt.de”
},
{
  ”mode”: ”create”,
  ”server”: ”ganeti”,
  ”disk_template”: ”plain”,
  ”ip_check”: false,
  ”name_check”: false,
  ”os_type”: ”debootstrap”,
  ”start”: false,
  ”nics”: [
  {
    ”mode”: ”bridged”
  }
  ],
  ”__version__”: 1,
  ”conflicts_check”: false,
  ”id”: ”Instanz2”,
  ”pnode”: ”pxhost01.seemoo.tu-darmstadt.de”
},
{
  ”mode”: ”create”,
  ”server”: ”ganeti”,
  ”disk_template”: ”plain”,
  ”ip_check”: false,
  ”name_check”: false,
  ”os_type”: ”debootstrap+wheezy”,
  ”start”: false,
```

```
    ”nics”: [
    {
       ”mode”: ”bridged”
    }
    ],
    ”__version__”: 1,
    ”conflicts_check”: false,
    ”id”: ”Instanz3”,
    ”pnode”: ”pxhost01.seemoo.tu-darmstadt.de”
  },
  ]
```


Put

Aktualisiert die Liste von Templates, falls Änderungen an der XML Datei vorgenommen wurden.

Postcondition:

- Die Liste der Templates wurde aktualisiert.