Hochschule Rhein-Waal
Fakultät Kommunikation und Umwelt

WiSe 2019/2020
Modul: Webentwicklung

Softwaredokumentation

lehrpool.nrw

Gruppenmitglieder:

Tilo Hauk 23792
Niclas Küppers 23249
Tobias Lehrke 23282

Sebastian Matt 23246
Judith Simon 21245

Abgabetermin:

07.02.2020

1 Projektbeschreibung

Innerhalb des Projektes ging es darum, die bestehende Java-App zur Verwaltung des virtuellen Lehrpools als Angular-Webapplikation zu implementieren. Dabei ging es lediglich um die Verwaltung der Metadaten von virtuellen Maschinen und den dazu in Relation stehenden Veranstaltungen, da ein bestehendes Backend bereits in der Produktivumgebung verwendet wird. Dieses Backend setzt sich aus dem sogenannten Master-Server und dem Satelliten-Server zusammen. Ersterer dient momentan ausschließlich zur Authentifikation, weshalb der Satelliten-Server letztendlich die eigentlichen Aufgaben zur Verwaltung der Metadaten übernimmt. Um mit diesen beiden Servern kommunizieren zu können, ist es erforderlich die Schnittstellenbeschreibungssprache „Thrift" der Apache Software Foundation zu benutzen.
Für dieses Projekt wurden folgende Meilensteine gesetzt, wobei sich einige dieser erst im Laufe der Projektdurchführung ergaben:

1. Java-App analysieren.
2. In die neue Sprache „Thrift" einarbeiten.
3. Mit Hilfe von „Thrift" eine Verbindung zum Backend via Node.js-Server etablieren.
4. Angular-Frontend programmieren.
5. Node.js-Server als Proxy zwischen Frontend und Backend schalten.

In der nachfolgenden Dokumentation wird auf die Aspekte Architektur, Schnittstellenbeschreibung, Verzeichnisstruktur, Dateienbeschreibung und gegebenenfalls erforderliche Benutzernamen und Passwörter eingegangen.

2 Architektur

Die vorliegende Software setzt sich aus dem clientseitigen Framework Angular, dem serverseitigen Framework Express und der dazugehörigen Serverlaufzeitumgebung Node.js zusammen. Des Weiteren kommt auf der Clientseite das JavaScript-Framework Bootstrap in Version 4.3.1 zum Einsatz. Die soeben aufgezählten Komponenten arbeiten gegen ein bereits existierendes Backend, welches sich aus einem Master-Server und einem Satelliten-Server zusammensetzt. Das Zusammenspiel dieser unterschiedlichen Komponenten wird in Abbildung 2.1 verdeutlicht.

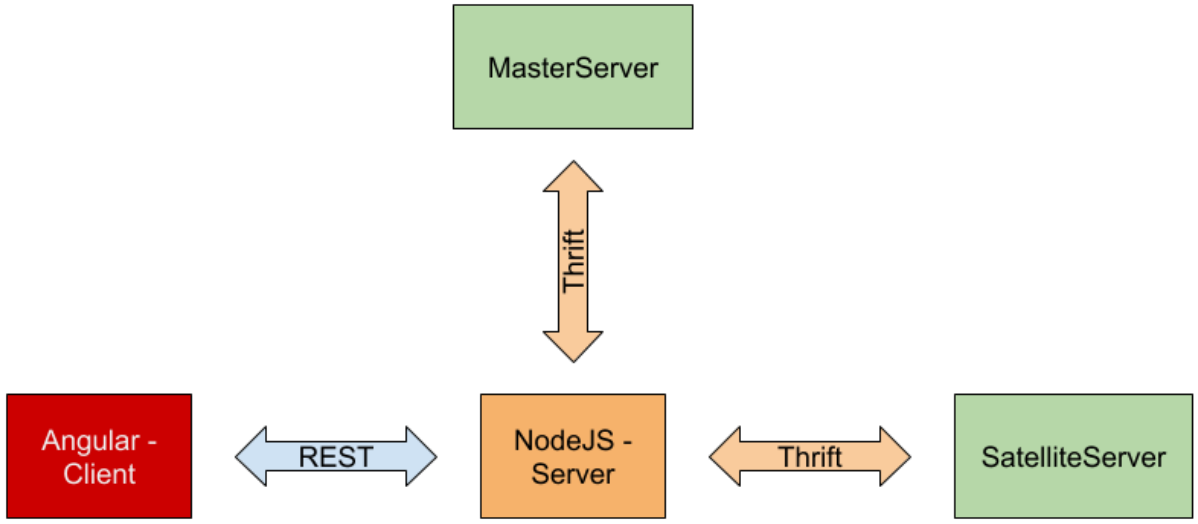


Abbildung 2.1: Architektur

So ist zu erkennen, dass der Angular-Client über das REST Interface mit dem Node.js-Server kommuniziert. Letzterer benutzt Apaches Schnittstellenbeschreibungssprache Thrift um mit dem Master-Server und dem Satelliten-Server zu interagieren. Aufgrund der Tatsache, dass die gesamte Struktur viele unterschiedliche Komponenten besitzt, wird nachfolgend auf jede eingegangen, damit die Funktion der jeweiligen Komponente verdeutlicht wird.

2.1 Angular

Angular ist ein TypeScript basiertes Front-End-Webapplikationsframework, welches für die Entwicklung von browserbasierten Single-Page-Applications, kurz SPA, verwendet wird. Es wird primär von Google entwickelt und als Open-Source mit MIT-Lizenz bereitgestellt. Unter Anderem geht Angular mit dem Ziel voran, Apps mit einer guten Testbarkeit zu entwickeln. Dies macht sich beispielsweise darin bemerkbar, dass zur Entwicklungszeit ein Server läuft, welcher das Transpilieren von TypeScript zu JavaScript beim Speichern des Quellcodes automatisiert durchführt und im Anschluss die

App im Browser aktualisiert. Hervorzuheben ist ebenfalls, dass Applikationen in mehreren Sprachen bzw. Sprachversionen entwickelt werden können. So steht es dem Entwickler frei zur Verfügung, ob er TypeScript oder eine andere Sprache verwenden möchte. Jedoch empfiehlt Angular die Verwendung von TypeScript und das aus gutem Grund.
Durch die simple Erlernbarkeit soll sich zeitgleich auch die Wahrscheinlichkeit von Programmfehlern verringern. Zusätzlich ist eine klassenbasierte und objektorientierte Programmierung möglich, welche perfekt zu dem Komponenten-System von Angular passt. So wird für jede Komponente der Applikation eine Angular-Komponente erzeugt. Beim Generieren einer neuen Komponente wird ein spezifischer Ordner innerhalb des app-Ordners erzeugt. In diesem befinden sich standardmäßig immer eine .component.html-, eine .component.css-, eine .component.spec.ts- und eine .component.ts-Datei. Eine übersichtliche Visualisierung zum Aufbau der Verzeichnisstruktur liefert dazu Kapitel 3.

2.2 Bootstrap

Bootstrap ist ein Frontend-Framework für Webanwendungen, das auf HTML und CSS basierende Gestaltungsvorlagen für einige Oberflächenelemente bereitstellt. Hauptmerkmale von Bootstrap sind das verwendete Grid-System zur Strukturierung der Oberfläche und die Möglichkeit des Responsive-Webdesigns, wodurch Webseiten, die Bootstrap verwenden, auf unterschiedlichen Displaygrößen sauber dargestellt werden können.

2.3 Express und REST

Express ist ein Framework zur Erstellung von Webanwendungen auf einem Node.js-Server, welches es ermöglicht REST-Interfaces erheblich angenehmer zu realisieren. Mit Hilfe von einem REST-Client können durch die Verwendung der Befehle GET, POST, PUT und DELETE die Ressourcen eines Servers angefordert, übergeben, bearbeitet oder gelöscht werden.

2.4 Node.js

Node.js ist eine serverseitige Plattform zum Betrieb von Netzwerkanwendungen, die zur Realisierung von Webservern dient. Die verwendete Skriptsprache ist JavaScript. Durch die ereignisgesteuerte Architektur verbraucht Node.js weniger Arbeitsspeicher pro Verbindung und ist performanter als vergleichbare Anwendungen. Zudem beinhaltet Node.js mit npm einen Paketmanager zur Verwaltung von Modulen. So sind über 750.000 Module zur einfachen Installation verfügbar.

3 Schnittstellenbeschreibung

Die Schnittstellenbeschreibung umfasst alle Inhalte, auf die in der index.js zugegriffen wird.

3.1 Class User

Properties

Die Properties beinhalten den Typen und einen beschreibenden Namen.

<string>name	//Anmeldename des Users (für Testacc)
<string>pass	//Passwort des Users (für Testacc)
<object>userData	//Userdatenobjekt
<string>satAddr	//Sat-Addresse auf die sich der User verbindet
<int>satAddr	//interne Id der Sat-Addresse auf die sich der User verbindet
<int>timeout	//Verbleibende Minuten bis zum Timeout
<int>setValTimeout	//Timeoutwert zum Reset des Timeouts

Methods

Auf der linken Seite befinden sich die Typen und Rückgabewerte, während auf der rechten Seite die Funktionsaufrufe samt Übergabeparameter zu sehen sind.

//Reset vom Timeout	
	resetTimeout()
//erstellt einen User	
static	createUser(<object>userData, <string>satAddr)

//setzt Timeoutwert zum Reset des Timeouts in Minuten	
static	setTimeout(<int>minuten)
//gibt den Timeoutwert zum Reset des Timeouts in Minuten zurück	
static <int>minuten	getTimeout()

3.2 Thrift

Methods

Auf der linken Seite befinden sich die Typen und Rückgabewerte, während auf der rechten Seite die Funktionsaufrufe samt Übergabeparameter zu sehen sind.

// Verbindungen aufbauen zum Sat- und Master-Server	
	connect()
// äquivalenter Login für Testacc, wie Shibboleth	
<object>userData	loginTestUser(<object>user)
// Erstellt eine Veranstaltung	
<string>lectureId	createLecture(<object>user, <object>Lecture)
// Updated eine Veranstaltung	
	updateLecture(<object>user, <string>lectureId, <object>lecture)
// Updated die Benutzerberechtigungen	
// für eine Veranstaltung	
	writeLecturePermissions(<object>user, <string>lectureId, <object>permissions)
// Liefert Benutzer (Sat)	

<object>satelliteClient	getSatelliteClient(<object>user)
// Liefert Benutzer (Master)	
<object>masterClient	getMasterClient()
// Liefert Liste der Betriebssysteme eines Users	
<object-list>OS	getOperatingSystems(<object>user)
// Liefert Eigenschaften der VM	
<object>imageDetails	getImageDetails(<object>user, <string>id)
// Liste mit berechtigten Benutzern	
// für eine VM zurückliefern lassen	
// Gibt alle Benutzer bis auf den Besitzer zurück	
<object>imagePermissions	imagePermissions(<object>user, <string>imageBaseId)
// Liste mit berechtigten Benutzern	
// für eine Veranstaltung zurückliefern lassen	
// Gibt alle Benutzer bis auf den Besitzer zurück	
<object>lecturePermissions	lecturePermissions(<object>user, <string>lectureId)
// Besitzer der VM ändern	
	setImageOwner(<object>user, <string>imageBaseId, <string>newOwnerId)
// Besitzer der Veranstaltung ändern	
	setLectureOwner(<object>user, <string>lectureId,<string>newOwnerId)
// Event anhand seiner ID löschen	
	deleteEvent(<object>user, <string>id)

// VM anhand ihrer ID löschen

deleteVm(<object>user, <string>id)

// Alle Benutzer zurückliefern

<object-list>userList getUserList(<object>user, <int>page)

// Räume zurückliefern

<object-list>locationList getLocations(<object>user)

// Liefert Liste der VMs

<object-list>imageList getImageList(<object>user,
<string>tagSearch, <int>page)

// Liefert Eigenschaften der Veranstaltung

<object>lectureDetails getLectureDetails(<object>user, <string>id)

// Liefert Liste der Veranstaltungen

<object-list>lectureList getLectureList(<object>user,
<string>tagSearch, <int>page)

// Updated eine VM

updateImageBase(<object>user,
<string>imageBaseId,
<object>imageBaseMetadata)

// Updated die Benutzer-Berechtigungen für eine VM

writeImagePermissions(<object>user,
<string>imageBaseId, <object>permissions)

// Löscht Version der VM

deleteImageVersion(<object>user, <int>id)

// Wandelt die SatIp in die SatId um	
<int>tmpId	getSatId(<string>ip)
// VM erstellen	
<string>imageId	createImage(<object>user, <string>imageName)
// Liefert Liste der Satellite-Server aus der config	
<object-list>send	getSats()
// Ankündigung an den SatelliteServer // für den Upload einer VM-Version	
<object>transferInformation	requestImageVersionUpload(<object>user, <string>imageBaseId, <int>fileSize, <binary-list>blockHashes, <binary>machineDescription)

3.3 Funktionen

Methods

Auf der linken Seite befinden sich die Typen und Rückgabewerte, während auf der rechten Seite die Funktionsaufrufe samt Übergabeparameter zu sehen sind.

//Attribute einer VM // für bessere Rückgabe bearbeiten	
<object>vm	prepareVm(<object>vm)
//Attribute aller VMs // für bessere Rückgabe bearbeiten	
<object-list>vms	prepareVms(<object-list>vms)
//Attribute einer Veranstaltung	

// für bessere Rückgabe bearbeiten	
<object>event	prepareEvent(<object>event)
//Attribute aller Veranstaltungen	
// für bessere Rückgabe bearbeiten	
<object-list>events	prepareEvents(<object-list>events)
//Ausgabe von Fehlermeldungen	
// mit Code und Nachricht über Rest	
	send(<object>response,<object>exception, <int>fehlercode)
//Erstellt einen User-Timeout-Countdown	
	scheduleInit(<object>user)

3.4 REST

Auf der linken Seite befinden sich die zugehörigen Werte, während auf der rechten Seite die URL-Routen zu sehen sind.

GET

// in der request wird die sessionId benötigt

//OS liefern	
<string-list> OS	/operatingSystems
// Alle Veranstaltungen liefern	
<object-list>Veranstaltungen	/events
// Eine Veranstaltung liefern	
<object>Veranstaltung	/events/<string>id

// Alle VMs liefern
<object-list>VMs /vms

// Benutzerrechte für eine VM liefern
<object>permissions /vms/permissions/<string>id

// Benutzerrechte für eine Veranstaltung liefern
<object>permissions /events/permissions/<string>id

// Eine VM liefern
<object>VM /vms/<string>id

// Alle Räume zurückliefern
<object-list> /locations

// Alle Benutzer zurückliefern
<object-list>userList /users

//Alle Sat-Verbindungen zurückliefern
<object-list>send /sats

DELETE

// Eine VM-Version löschen
/vms/version/<string>id

// Eine Veranstaltung löschen
/events/<string>id

// Eine VM löschen
/vms/<string>id

POST

// in den body kommen “username” und “password”
<object>userData /login

//Löscht de User aus Cache in dem request wird die sessionId benötigt
<object>exception /logout

//im body werden “userData” und “satAddr” benötigt
<object>user /SatAddr

//in der request wird die sessionId benötigt und im body ein LectureWrite-Objekt
<string>lectureId /event

//Erstellt eine neue VM; Ohne Version und Metadaten
<string>imageId /vm

// Dem Server den Upload einer VM ankündigen und das Upload Token erhalten
<object>transferInformation /vm/upload

PUT

// Änderung der Rechte von Veranstaltungen
/events/permissions

// Änderung des Besitzers von Veranstaltungen
/events/owner

// Änderung der Veranstaltungs-Einstellungen
/events/<string>id

// Änderung der Rechte von VMs
/vms/permissions

// Änderung des Besitzers von VMs

/vms/owner

// Änderung der VM-Einstellungen

/vms/<string>id

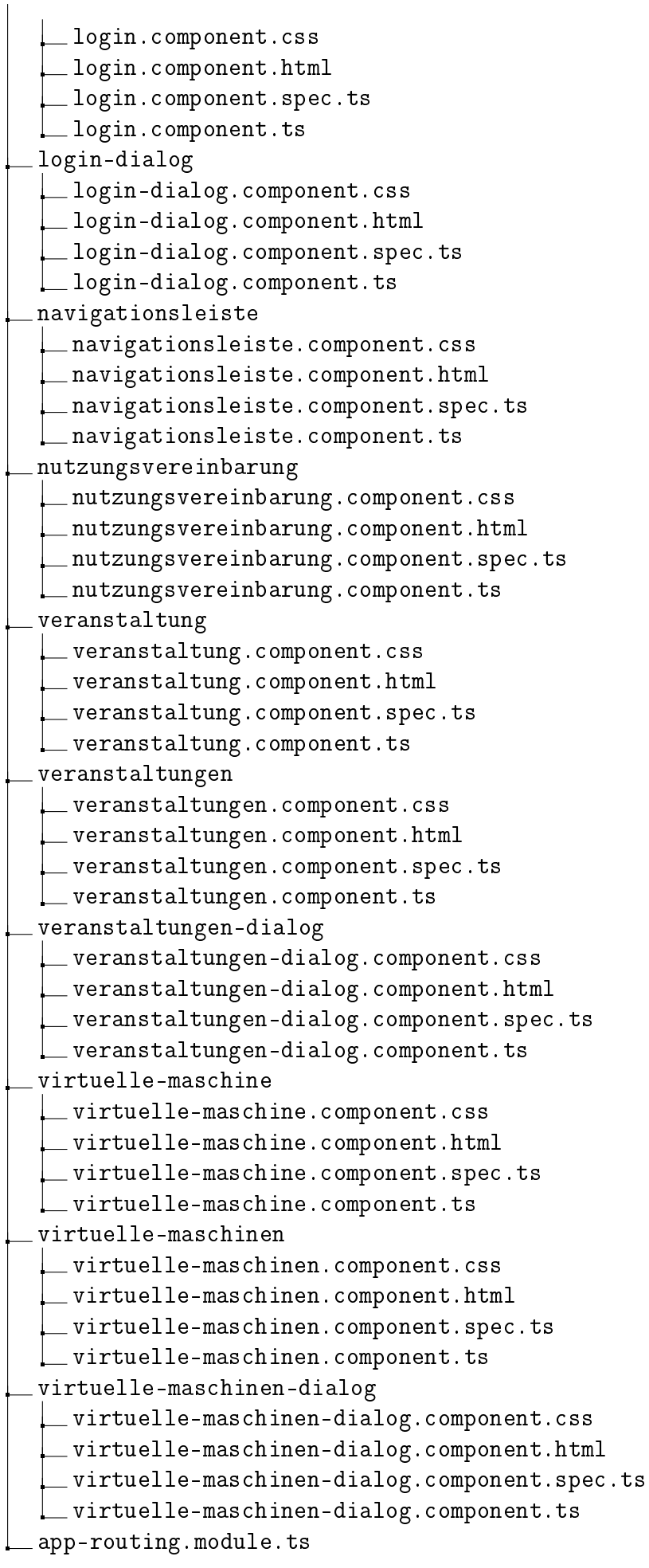
4 Verzeichnisstruktur

Im Folgenden wird die Verzeichnisstruktur des Angular-Clients sowie des Node.js-Servers festgehalten.

4.1 Angular-Client

Nachfolgend befindet sich die Verzeichnisstruktur des Angular-Clients:

```
lehrpool-nrw
├── e2e
│   ├── src
│   │   ├── app.e2e-spec.ts
│   │   └── app.po.ts
│   ├── protractor.conf.js
│   └── tsconfig.json
└── src
    └── app
        ├── aenderungen-verwerfen-dialog
        │   ├── aenderungen-verwerfen-dialog.component.css
        │   ├── aenderungen-verwerfen-dialog.component.html
        │   ├── aenderungen-verwerfen-dialog.component.spec.ts
        │   └── aenderungen-verwerfen-dialog.component.ts
        ├── change-owner
        │   ├── change-owner.component.css
        │   ├── change-owner.component.html
        │   ├── change-owner.component.spec.ts
        │   └── change-owner.component.ts
        ├── change-vm
        │   ├── change-vm.component.css
        │   ├── change-vm.component.html
        │   ├── change-vm.component.spec.ts
        │   └── change-vm.component.ts
        ├── create-veranstaltung
        │   ├── create-veranstaltung.component.css
        │   ├── create-veranstaltung.component.html
        │   ├── create-veranstaltung.component.spec.ts
        │   └── create-veranstaltung.component.ts
        ├── create-vm
        │   ├── create-vm.component.css
        │   ├── create-vm.component.html
        │   ├── create-vm.component.spec.ts
        │   └── create-vm.component.ts
        ├── datenschutzerklaerung
        │   ├── datenschutzerklaerung.component.css
        │   ├── datenschutzerklaerung.component.html
        │   ├── datenschutzerklaerung.component.spec.ts
        │   └── datenschutzerklaerung.component.ts
        ├── login
```


login.component.css
login.component.html
login.component.spec.ts
login.component.ts
login-dialog
login-dialog.component.css
login-dialog.component.html
login-dialog.component.spec.ts
login-dialog.component.ts
navigationsleiste
navigationsleiste.component.css
navigationsleiste.component.html
navigationsleiste.component.spec.ts
navigationsleiste.component.ts
nutzungsvereinbarung
nutzungsvereinbarung.component.css
nutzungsvereinbarung.component.html
nutzungsvereinbarung.component.spec.ts
nutzungsvereinbarung.component.ts
veranstaltung
veranstaltung.component.css
veranstaltung.component.html
veranstaltung.component.spec.ts
veranstaltung.component.ts
veranstaltungen
veranstaltungen.component.css
veranstaltungen.component.html
veranstaltungen.component.spec.ts
veranstaltungen.component.ts
veranstaltungen-dialog
veranstaltungen-dialog.component.css
veranstaltungen-dialog.component.html
veranstaltungen-dialog.component.spec.ts
veranstaltungen-dialog.component.ts
virtuelle-maschine
virtuelle-maschine.component.css
virtuelle-maschine.component.html
virtuelle-maschine.component.spec.ts
virtuelle-maschine.component.ts
virtuelle-maschinen
virtuelle-maschinen.component.css
virtuelle-maschinen.component.html
virtuelle-maschinen.component.spec.ts
virtuelle-maschinen.component.ts
virtuelle-maschinen-dialog
virtuelle-maschinen-dialog.component.css
virtuelle-maschinen-dialog.component.html
virtuelle-maschinen-dialog.component.spec.ts
virtuelle-maschinen-dialog.component.ts
app-routing.module.ts

```
    ├── app.component.css
    ├── app.component.html
    ├── app.component.spec.ts
    ├── app.component.ts
    ├── app.module.ts
    ├── login.service.ts
    ├── user.service.ts
    ├── user.ts
    ├── veranstaltung.ts
    ├── veranstaltungen.service.ts
    ├── vm.ts
    └── vm.service.ts
  ├── assets
  │   ├── bootstrap/4.3.1
  │   │   └── css
  │   │       ├── bootstrap-grid.css
  │   │       ├── bootstrap-grid.css.map
  │   │       ├── bootstrap-grid.min.css
  │   │       ├── bootstrap-grid.min.css.map
  │   │       ├── bootstrap-reboot.css
  │   │       ├── bootstrap-reboot.css.map
  │   │       ├── bootstrap-reboot.min.css
  │   │       ├── bootstrap-reboot.min.css.map
  │   │       ├── bootstrap.css
  │   │       ├── bootstrap.css.map
  │   │       ├── bootstrap.min.css
  │   │       ├── bootstrap.min.css.map .5 js
  │   │       ├── bootstrap.bundle.js
  │   │       ├── bootstrap.bundle.js.map
  │   │       ├── bootstrap.bundle.min.js
  │   │       ├── bootstrap.bundle.min.js.map
  │   │       ├── bootstrap.js
  │   │       ├── bootstrap.js.map
  │   │       ├── bootstrap.min.js
  │   │       ├── bootstrap.min.js.map
  │   │       └── jquery-3.4.1.min.js
  │   ├── images
  │   │   └── lehrpool-nrw.png
  │   └── .gitkeep
  ├── environments
  │   ├── environment.prod.ts
  │   └── environment.ts
  ├── favicon.ico
  ├── index.html
  ├── main.ts
  ├── polyfills.ts
  ├── style.css
  └── test.ts
├── .editorconfig
└── .gitignore
```

```
├── .README.md
├── angular.json
├── browserlist
├── karma.conf.js
├── package-lock.json
├── package.json
├── tsconfig.app.json
├── tsconfig.json
├── tsconfig-spec.json
└── tslint.json
```

4.2 Node.js-Server

Nachfolgend befindet sich die Verzeichnisstruktur des Node.js-Servers:

```
node-js
├── gen-nodejs
│   ├── MasterServer.js
│   ├── SatelliteServer.js
│   └── bwlp_types.js
├── module
│   ├── functions.js
│   ├── thrift.js
│   └── user.js
├── .gitignore
├── package.json
└── tconfig.json
```

5 Dateienbeschreibung: Angular-Client

Im Nachfolgenden werden alle Dateien beschrieben, welche sich im Verzeichnis lehrpool-nrw-master/src/app/ befinden. Falls es extravagante Besonderheiten zu einer Komponente gibt, werden diese genannt. Andernfalls wird lediglich das Nötigste zu jeder Komponente erwähnt.

5.1 aenderungen-verwerfen-dialog

Diese Komponente ist dafür zuständig, den Benutzer zu fragen, ob er die jeweilige Seite wirklich verlassen möchte. Dieser Dialog wird ausgelöst sofern der Anwender mindestens eine Änderung vorgenommen hat, welche noch nicht gespeichert wurde. So kommt diese Komponente in Form eines Pop-Ups zum Einsatz, wenn der Benutzer beispielsweise die Metadaten einer Veranstaltung verändert hat und die Seite verlassen möchte, ohne zuvor den Button zum Speichern gedrückt zu haben.

aenderungen-verwerfen-dialog.component.css

Diese Datei besitzt derzeit keinen Inhalt.

aenderungen-verwerfen-dialog.component.html

Erzeugt zwei Buttons damit der Benutzer entweder auf „Ja" oder „Nein" klicken kann, um die Änderungen zu verwerfen oder auf der Seite zu bleiben.

aenderungen-verwerfen-dialog.component.spec.ts

Automatisch generierte Datei für Unit-Tests.

aenderungen-verwerfen-dialog.component.ts

Enthält die Klasse „AenderungenVerwerfenDialogComponent", welche zum Erzeugen des Dialogs verwendet wird. Hier sind entsprechende Methoden implementiert, um alle benötigten Interaktionen aus der zugehörigen .html-Datei zu realisieren.

5.2 change-owner

Diese Komponente ermöglicht es den Besitzer einer VM oder einer Veranstaltung zu ändern. Erzeugt eine Tabelle, in welcher die zur Auswahl stehenden Benutzer enthalten sind. Ebenfalls ist in jeder Reihe ein Radiobutton enthalten um den neuen Benutzer auszuwählen. Über der Tabelle ist eine

Suchleiste um nach Benutzern zu filtern. Zum Festlegen des neuen Benutzers oder zum Abbrechen der Aktion sind zwei Buttons implementiert worden.

change-owner.component.css

Diese Datei besitzt derzeit keinen Inhalt.

change-owner.component.html

Diese Datei erzeugt die benötigte/n Tabelle, Buttons und Input-Felder um den Benutzer einer virtuellen Maschine oder einer Veranstaltung zu ändern.

change-owner.component.spec.ts

Automatisch generierte Datei für Unit-Tests.

change-owner.component.ts

Enthält die Klasse „ChangeOwnerComponent", welche die Auswahl eines neuen Benutzers ermöglicht. Hier sind entsprechende Methoden implementiert, um alle benötigten Interaktionen aus der zugehörigen .html-Datei zu realisieren. Ebenso, um die benötigten Daten über den Node.js-Server vom Satelliten-Server abrufen zu können.

5.3 change-vm

Diese Komponente ermöglicht es die verknüpfte VM von einer Veranstaltung zu ändern. Sie erzeugt eine Tabelle, in welcher die zur Auswahl stehenden VMs enthalten sind. Ebenfalls ist in jeder Reihe ein Radiobutton enthalten um die neue VM auszuwählen. Über der Tabelle ist eine Suchleiste um nach VMs zu filtern. Zum Festlegen der neuen VM oder Abbrechen der Aktion sind zwei Buttons implementiert worden.

change-vm.component.css

Diese Datei besitzt derzeit keinen Inhalt.

change-vm.component.html

Diese Datei erzeugt die benötigte/n Tabelle, Buttons und Input-Felder, um die verknüpfte virtuelle Maschine einer Veranstaltung zu ändern.

change-vm.component.spec.ts

Automatisch generierte Datei für Unit-Tests.

change-vm.component.ts

Enthält die Klasse „ChangeVmComponent", welche die Auswahl eines neuen Benutzers ermöglicht. Hier sind entsprechende Methoden implementiert, um alle benötigten Interaktionen aus der zugehörigen .html-Datei zu realisieren. Ebenso, um die benötigten Daten über den Node.js-Server vom Satelliten-Server abrufen zu können.

5.4 create-veranstaltung

Diese Komponente ermöglicht es dem Anwender eine Veranstaltung zu erzeugen. Sie enthält zwei Input-Felder, um den Namen und eine Beschreibung der neuen Veranstaltung einzugeben. Des Weiteren werden zwei Felder erzeugt, die das Startdatum und Enddatum festlegen. Neben beiden Feldern für das Datum wird jeweils ein Feld für die Uhrzeit des gewählten Datums generiert. Das Startdatum gibt immer das aktuelle Datum mit einer Uhrzeit von 00:00 Uhr vor. Das Enddatum gibt ebenfalls das aktuelle Datum vor, jedoch mit einer Uhrzeit von 23:59 Uhr. Der Benutzer kann das Start- und Enddatum jedoch nach seinen Wünschen anpassen.
Im nächsten Teilbereich der HTML-Datei wird eine Tabelle erzeugt, in welcher die zur Auswahl stehenden VMs enthalten sind. Zur Auswahl ist in jeder Reihe ein Radiobutton enthalten, um die neue VM auszuwählen. Über der Tabelle ist eine Suchleiste um nach VMs zu filtern.
Im Anschluss daran werden mehrere Checkboxen erzeugt, welche zusätzliche Einstellungen für die Veranstaltung ermöglichen, wie beispielsweise den Prüfungsmodus auf aktiv zu setzen.
Als Nächstes wird eine weitere Tabelle erzeugt, welche dazu dient, Benutzer einzutragen, welche daraufhin spezielle Berechtigungen erhalten können. Dafür enthält die Datei ein Input-Feld direkt unter der Tabelle. Daneben ist ein Button implementiert, um den eingetragenen Benutzer zur Tabelle hinzuzufügen. Sobald ein Benutzer in der Tabelle erscheint, können Checkboxen angeklickt werden, um einzelne Rechte hinzuzufügen oder zu entfernen. Zusätzlich befinden sich unterhalb der Tabelle mehrere Checkboxen, um die Standardberechtigungen aller anderen Benutzer festzulegen.
Im letzten Abschnitt können Räume angeklickt werden, damit die Veranstaltung in den jeweiligen Räumen sichtbar ist. Zuletzt wird ein Button zum Erstellen der Veranstaltung erzeugt. Falls der Benutzer auf diesen Button klickt und fehlerhafte Einträge getätigt hat oder Pflichtfelder nicht ausgefüllt wurden, färbt sich der Button rot und es werden Fehlermeldungen neben den einzelnen Abschnitten des Formulars angezeigt. Für den Fall, dass sich der Anwender dazu entscheidet keine neue virtuelle

Maschine anzulegen, wurde ein Button zum Abbrechen implementiert.

create-veranstaltung.component.css

Diese Datei besitzt die Klasse „ul", welche das Attribut „list-style" des jeweiligen Elements auf „none" setzt.

create-veranstaltung.component.html

Diese Datei erzeugt alle benötigten Felder, Buttons und Tabellen um das Formular zum Erzeugen einer neuen Veranstaltung anforderungsgerecht zu erstellen.

create-veranstaltung.component.spec.ts

Automatisch generierte Datei für Unit-Tests.

create-veranstaltung.component.ts

Enthält die Klasse „CreateVeranstaltungComponent", welche das Erstellen einer neuen Veranstaltung ermöglicht. Hier sind entsprechende Methoden implementiert, um alle benötigten Interaktionen aus der zugehörigen .html-Datei zu realisieren. Ebenso, um die benötigten Daten über den Node.js-Server vom Satelliten-Server abrufen zu können. Des Weiteren befinden sich in dieser Datei mehrere Klassen, um die Baumstruktur, die Anzeige und die Auswahl der Räume dynamisch zu ermöglichen.

5.5 create-vm

Mit dieser Komponente wird es möglich gemacht, eine virtuelle Maschine samt ihren Metadaten zu erzeugen. Aufgrund der Tatsache, dass noch keine Lösung für den Upload einer VM gefunden wurde, wird die erstellte VM bislang ohne eine VM-Version erzeugt. Innerhalb dieser Komponente wird ein Formular erzeugt, welches alle notwendigen Felder deklariert, um eine neue virtuelle Maschine anzulegen. Dazu wurde vorbehaltlich ein Feld implementiert mit welchem Dateien zum Hochladen ausgewählt werden können, falls es in Zukunft möglich gemacht wird, eine VM-Version hochzuladen. Des Weiteren wird ein Input-Feld erzeugt, um die Namenseingabe der neuen virtuellen Maschine zu ermöglichen. Darunter befindet sich eine Text-Area um eine Beschreibung hinzuzufügen.
Im nächsten Teilbereich des Formulars können Benutzerberechtigungen festgelegt werden. Dazu wird eine Tabelle erzeugt, welche genutzt wird, um Benutzer einzutragen, welche im Anschluss spezifische Berechtigungen erhalten können. Dafür enthält die Datei ein Input-Feld direkt unter der Tabelle. Daneben ist ein Button implementiert, um den eingetragenen Benutzer in die Tabelle zu schreiben.

Sobald ein Benutzer in der Tabelle erscheint, können Checkboxen angeklickt werden, um einzelne Rechte hinzuzufügen oder zu entfernen. Die einzelnen Rechte heißen „Download", „Verlinken", „Bearbeiten" und „Admin". Zusätzlich befinden sich unterhalb der Tabelle mehrere Checkboxen um die Standardberechtigungen aller anderen Benutzer festzulegen.
Als Letztes befindet sich auf der Seite ein Button zum Fertigstellen, welcher das Formular zum Server sendet. Falls die Pflichtfelder unvollständig oder nicht ausgefüllt worden sind, färbt sich der Button rot und es erscheint eine Fehlermeldung. Für den Fall, dass sich der Anwender dazu entscheidet, keine neue virtuelle Maschine anzulegen, wurde ein Button zum Abbrechen implementiert.

create-vm.component.css

Diese Datei besitzt derzeit keinen Inhalt.

create-vm.component.html

In dieser Datei werden alle geforderten Felder, Buttons und Tabellen erstellt, um die Visualisierung anforderungsgerecht zu gestalten.

create-vm.component.spec.ts

Automatisch generierte Datei für Unit-Tests.

create-vm.component.ts

Enthält die Klasse „CreateVmComponent", welche das Erstellen einer neuen virtuellen Maschine ermöglicht. Hier sind entsprechende Methoden implementiert, um alle benötigten Interaktionen aus der zugehörigen .html-Datei zu realisieren. Ebenso, um die benötigten Daten über den Node.js-Server vom Satelliten-Server abrufen zu können.

5.6 datenschutzerklaerung

Diese Komponente enthält die Datenschutzerklärung.

datenschutzerklaerung.component.css

Diese Datei besitzt derzeit keinen Inhalt.

datenschutzerklaerung.component.html

Hier werden mehrere Paragraphen deklariert, welche die Datenschutzerklärung in Form eines Textes beinhalten.

datenschutzerklaerung.component.spec.ts

Automatisch generierte Datei für Unit-Tests.

datenschutzerklaerung.component.ts

Enthält die Klasse „DatenschutzerklaerungComponent".

5.7 login

Diese Komponente ermöglicht derzeit den Login mit einem Testaccount.

login.component.css

Diese Datei besitzt derzeit keinen Inhalt.

login.component.html

In dieser Datei wird das Logo des Projekts lehrpool.NRW eingebunden. Des Weiteren erzeugt diese Datei ein Formular, in welchem die Benutzerdaten für einen Testaccount eingegeben werden können. Zum Anmelden wird ein Login-Button und zum Registieren ein Registrieren-Button hinzugefügt.

login.component.spec.ts

Automatisch generierte Datei für Unit-Tests.

login.component.ts

Enthält die Klasse „LoginComponent", welche das Anmelden auf dem Master-Server ermöglicht. Hier sind entsprechende Methoden implementiert, um alle benötigten Interaktionen aus der zugehörigen .html-Datei zu realisieren. Ebenso, um die User-Daten nach einem erfolgreichen Login über den Node.js-Server vom Master-Server abrufen zu können.

5.8 login-dialog

Diese Komponente erzeugt einen Dialog, welcher nach einem erfolgreichen Login aufgerufen wird. In diesem werden dem Benutzer die zur Auswahl stehenden Satelliten-Server angezeigt. Davon muss sich der Anwender einen aussuchen oder bei Bedarf einen eigenen unterhalb des Dialogs eintragen.

login-dialog.component.css

Diese Datei besitzt derzeit keinen Inhalt.

login-dialog.component.html

In dieser Datei werden die zur Auswahl stehenden Satelliten-Server mit dem zusätzlichen Input-Feld für einen eigenen Satelliten-Server erzeugt. Um den jeweiligen Server auszuwählen, steht neben jeder Option ein Radiobutton. Damit die Auswahl bestätigt werden kann, wird der Button „Weiter" erzeugt.

login-dialog.component.spec.ts

Automatisch generierte Datei für Unit-Tests.

login-dialog.component.ts

Enthält die Klasse „LoginDialogComponent", welche das Senden der Serverauswahl ermöglicht. Hier sind entsprechende Methoden implementiert, um alle benötigten Interaktionen aus der zugehörigen .html-Datei zu realisieren. Ebenso, um die notwendigen Daten nach einem erfolgreichen Login an den Node.js-Server zu senden, damit dieser den aktuell angemeldeten Benutzer in seine Benutzerverwaltung hinzufügen kann.

5.9 navigationsleiste

Diese Komponente wird benötigt, damit der Benutzer von jeder Seite der Anwendung zu anderen Seiten navigieren kann. Die Navigationsleiste enthält das Logo des Projekts lehrpool.NRW. Daneben befinden sich zwei Buttons, um auf die Übersichtsseite der virtuellen Maschinen oder der Veranstaltung zu gelangen. Zusätzlich gibt es ein Dropdown-Menü, welches zu der Nutzungsvereinbarung, der Datenschutzerklärung und dem E-Learning Wiki führt. Zusätzlich befindet sich in der Navigationsleiste ein Button zum Abmelden des eingeloggten Benutzers. Damit der Benutzer weiß, auf welcher Seite er sich befindet, wird der jeweilige Buttons der Navigationsleiste mit einem routerLinkActive versehen. Außerdem hat die Navigationsleiste eine feste Position und wandert nicht mit, wenn der Benutzer auf der Seite nach unten scrollt.

navigationsleiste.component.css

In dieser Datei befindet sich die Klasse „.navbarSelect", welche das Attribut „display" des jeweiligen Elements auf „none" setzt.

navigationsleiste.component.html

Hier werden alle benötigten Attribute deklariert und falls nötig die erforderlichen Klassen zugewiesen, um die Navigationsleiste zu generieren.

navigationsleiste.component.spec.ts

Automatisch generierte Datei für Unit-Tests.

navigationsleiste.component.ts

Enthält die Klasse „NavigationsleisteComponent", welche eine unkomplizierte Navigation innerhalb der Anwendung möglich macht. Hier sind entsprechende Methoden implementiert, um alle benötigten Interaktionen aus der zugehörigen .html-Datei zu realisieren.

5.10 nutzungsvereinbarung

Diese Komponente enthält die Nutzungsvereinbarung.

nutzungsvereinbarung.component.css

In dieser Datei ist die Klasse „a" enthalten, welche die Schriftfarbe des jeweiligen Elements auf „whitesmoke" setzt.

nutzungsvereinbarung.component.html

Hier werden mehrere Paragraphen deklariert, welche die Nutzungsvereinbarungen in Form eines Textes beinhalten.

nutzungsvereinbarung.component.spec.ts

Automatisch generierte Datei für Unit-Tests.

nutzungsvereinbarung.component.ts

Enthält die Klasse „NutzungsvereinbarungComponent".

5.11 veranstaltung

Diese Komponente ermöglicht es eine einzelne Veranstaltung anzusehen, zu bearbeiten und zu aktualisieren. Hier können mehr Metadaten angepasst werden, als bei der Erstellung einer neuen Veranstaltung. Für die bessere Lesbarkeit wird nachfolgend auf alle möglichen Interaktionen in den einzelnen Abschnitten eingegangen.

Allgemein:

Hier kann der Anwender, sofern er ausreichende Berechtigungen besitzen sollte, den Veranstaltungsnamen sowie die zugehörige Beschreibung ändern. Ebenfalls ist es möglich, die Besitzrechte auf einen anderen Benutzer zu übertragen. Zusätzlich lässt sich die verknüpfte virtuelle Maschine ändern und das Startdatum sowie das Enddatum können inklusive ihrer Startzeit und Endzeit bearbeitet werden. Des Weiteren lässt sich hier ändern, ob die Veranstaltung automatisch die neueste VM-Version der verknüpften virtuellen Maschine verwenden soll.

Beschränkungen:

Unter diesem Reiter kann der Anwender auswählen, ob er externe Speichermedien in der virtuellen Maschine zulassen möchte. Des Weiteren kann er festlegen, ob diese Veranstaltung eine E-Prüfung ist. Falls die Veranstaltung keinen Zugriff auf das Netzwerk/ Internet haben soll, kann hier ebenfalls eine Checkbox angeklickt werden. Zusätzlich könnten Netzwerkregeln definiert werden, jedoch steht die Funktion im derzeitigen Entwicklungsstadium der Webapplikation noch nicht zur Verfügung.

Firewall:

In diesem Bereich könnten die Firewall-Regeln definiert werden, jedoch steht die Funktionalität im derzeitigen Entwicklungsstadium der Webapplikation noch nicht zur Verfügung.

Raumauswahl:

Hier können die Räume ausgewählt werden, in welchem die Veranstaltung sichtbar sein soll.

VM-Start:

Bei dem Start der verknüpften virtuellen Maschine könnten ebenfalls Einstellungen vorgenommen werden, jedoch steht diese Funktion im derzeitigen Entwicklungsstadium der Webapplikation noch nicht zur Verfügung.

Berechtigungen:

Unter diesem Abschnitt können die einzelnen Berechtigungen für die Veranstaltung eingestellt werden. So können Benutzer für eine spezifische Rechtevergabe einer Tabelle hinzugefügt werden oder allgemeine Standardberechtigungen für alle Benutzer festgelegt werden. Zu den Berechtigungen, welche eingestellt werden können, gehören „Bearbeiten" und „Admin".

Netzlaufwerke:

An dieser Stelle könnten Netzlaufwerke und sogar Drucker einer Veranstaltung zugewiesen werden. Allerdings steht diese Funktion im derzeitigen Entwicklungsstadium der Webapplikation noch nicht zur Verfügung.

LDAP-Filter:

Im letzten Abschnitt könnten LDAP-Filter eingestellt werden, jedoch steht die Funktion im derzeitigen Entwicklungsstadium der Webapplikation noch nicht zur Verfügung.

veranstaltung.component.css

In dieser Datei befindet sich die Klasse „.nav-link", welche die Schriftfarbe des jeweiligen Elements auf „whitesmoke" setzt. Zusätzlich enthält diese .css-Datei die Klasse „ul", welche das Attribut „list-style" des jeweiligen Elements auf „none" setzt.

veranstaltung.component.html

Diese Datei erzeugt das geforderte Formular und generiert alle benötigten Felder, um eine ordentliche Bearbeitung der Veranstaltung zu ermöglichen.

veranstaltung.component.spec.ts

Automatisch generierte Datei für Unit-Tests.

veranstaltung.component.ts

Enthält die Klasse „VeranstaltungComponent", welche das Bearbeiten einer Veranstaltung ermöglicht. Hier sind entsprechende Methoden implementiert, um alle benötigten Interaktionen aus der zugehörigen .html-Datei zu realisieren. Ebenso, um die benötigten Daten über den Node.js-Server vom Satelliten-Server abrufen zu können. Des Weiteren befinden sich in dieser Datei mehrere Klassen, um die Baumstruktur, die Anzeige und die Auswahl der Räume dynamisch zu ermöglichen.

5.12 veranstaltungen

Diese Komponente ist für die Auflistung der vorhandenen Veranstaltungen zuständig. Diese werden allesamt in einer Angular-Material-Table aufgelistet und dem Benutzer angezeigt. Sobald auf eine Veranstaltung geklickt wird, gelangt der Benutzer zur Detailansicht der jeweiligen Veranstaltung. Die erste Spalte jeder Zeile enthält eine Checkbox, welche benutzt wird, um die ausgewählten Veranstaltungen zu löschen. Dafür muss nach dem Auswählen auf den roten Löschen-Button geklickt werden, welcher sich rechts neben der Suchleiste befindet. Diese wiederum befindet sich oberhalb der Tabelle und sorgt dafür, dass der Anwender die Tabelle mit Stichworten filtern kann. Über dem Löschen-Button befindet sich ein Button um eine neue Veranstaltung zu erzeugen. Das Dropdown-Menü rechts neben der Suche ist derzeit deaktiviert, da die Funktionalität im aktuellen Entwicklungsstadium der Webapplikation nicht gewährleistet werden kann.

veranstaltungen.component.css

Diese Datei beinhaltet die Klasse „tr[ng-reflect-router-link]" und setzt das Attribut „cursor" des betroffenen Elements auf den Wert „pointer".

veranstaltungen.component.html

Mit dieser Datei werden die benötigten Elemente erzeugt, welche für eine ordentliche Visualisierung der Veranstaltungsübersicht zuständig sind.

veranstaltungen.component.spec.ts

Automatisch generierte Datei für Unit-Tests.

veranstaltungen.component.ts

Enthält die Klasse „VeranstaltungenComponent", welche das Auflisten der Veranstaltungen ermöglicht. Hier sind entsprechende Methoden implementiert um alle benötigten Interaktionen aus der zugehörigen .html-Datei zu realisieren. Ebenso um die benötigten Daten über den Node.js-Server vom Satelliten-Server abrufen zu können.

5.13 veranstaltungen-dialog

Diese Komponente bittet den Benutzer um eine Bestätigung, ob die ausgewählten Veranstaltungen wirklich gelöscht werden sollen. Der Anwender hat die Möglichkeit zu bestätigen oder abzubrechen.

Im Falle einer Bestätigung, werden alle Veranstaltungen gelöscht für welche der Benutzer eine ausreichende Berechtigung besitzt.

veranstaltungen-dialog.component.css

Diese Datei besitzt derzeit keinen Inhalt.

veranstaltungen-dialog.component.html

In dieser Datei werden die jeweiligen Elemente generiert, welche für den Löschvorgang notwendig sind.

veranstaltungen-dialog.component.spec.ts

Automatisch generierte Datei für Unit-Tests.

veranstaltungen-dialog.component.ts

Enthält die Klasse „VeranstaltungenDialogComponent", welche das Löschen der Veranstaltungen ermöglicht. Hier sind entsprechende Methoden implementiert um alle benötigten Interaktionen aus der zugehörigen .html-Datei zu realisieren. Im Anschluss gibt die Komponente die getätigte Auswahl wieder zurück, damit die veranstaltungen-Komponente das Löschen übernehmen kann.

5.14 virtuelle-maschine

Diese Komponente ermöglicht es die Metadaten einer virtuellen Maschine zu bearbeiten und abzuspeichern. In dieser Detailansicht gibt es mehrere Reiter, welche die einzelnen Kategorien voneinander trennen. Fortan wird auf die vereinzelten Abschnitte separat eingegangen.

Übersicht:

Sofern der jeweilige Anwender über die erforderlichen Rechte verfügt, kann er in diesem Abschnitt den Namen, die Beschreibung, den Besitzer und das Betriebssystem der virtuellen Maschine ändern.

VM-Versionen:

Unter diesem Reiter können die einzelnen VM-Versionen in einer Tabelle angesehen und gelöscht werden. Sofern der Benutzer die benötigten Berechtigungen besitzt, kann er eine VM-Version löschen in dem er auf den „-"-Button klickt, welcher sich immer in der ersten Spalte einer Reihe der VM-Versionen befindet. Sofern keine Rechte vorhanden sind, wird der Button zum Löschen nicht

angezeigt.

Berechtigungen:
Unter diesem Abschnitt können die einzelnen Berechtigungen für die virtuelle Maschine eingestellt werden. So können Benutzer für eine spezifische Rechtevergabe einer Tabelle hinzugefügt werden oder allgemeine Standardberechtigungen für alle Benutzer festgelegt werden. Zu den Berechtigungen, welche eingestellt werden können, gehören „Bearbeiten", „Download", „Verlinken" und „Admin".

virtuelle-maschine.component.css

Diese Datei enthält die Klasse „.nav-link", welche das Attribut „color" des jeweiligen Elements auf den Wert „whitesmoke" setzt.

virtuelle-maschine.component.html

Innerhalb dieser Datei werden alle benötigten Elemente generiert, welche für die Visualisierung der Metadaten einer virtuellen Maschine notwendig sind.

virtuelle-maschine.component.spec.ts

Automatisch generierte Datei für Unit-Tests.

virtuelle-maschine.component.ts

Enthält die Klasse „VirtuelleMaschineComponent", welche das Bearbeiten einer virtuellen Maschine ermöglicht. Hier sind entsprechende Methoden implementiert um alle benötigten Interaktionen aus der zugehörigen .html-Datei zu realisieren. Ebenso um die benötigten Daten über den Node.js-Server vom Satelliten-Server abrufen zu können.

5.15 virtuelle-maschinen

Diese Komponente ist für die Auflistung der vorhandenen virtuellen Maschinen zuständig. Diese werden allesamt in einer Angular-Material-Table aufgelistet und dem Benutzer angezeigt. Sobald auf eine virtuelle Maschine geklickt wird, gelangt der Benutzer zur Detailansicht der jeweiligen Veranstaltung. Die erste Spalte jeder Zeile enthält eine Checkbox, welche benutzt wird um die ausgewählten virtuellen Maschinen zu löschen. Dafür muss nach dem Auswählen auf den roten Löschen-Button geklickt werden, welcher sich rechts neben der Suchleiste befindet. Diese wiederum befindet sich oberhalb der Tabelle und sorgt dafür, dass der Anwender die Tabelle mit Stichworten filtern kann.

Über dem Löschen-Button befindet sich ein Button um eine neue Veranstaltung zu erzeugen. Das Dropdown-Menü rechts neben der Suche, ist derzeit deaktiviert da die Funktionalität im aktuellen Entwicklungsstadium der Webapplikation nicht gewährleistet werden kann.

virtuelle-maschinen.component.css

Diese Datei beinhaltet die Klasse „tr[ng-reflect-router-link]" und setzt das Attribut „cursor" des betroffenen Elements auf den Wert „pointer".

virtuelle-maschinen.component.html

Mit dieser Datei werden die benötigten Elemente erzeugt, welche für eine ordentliche Visualisierung der virtuellen Maschinen zuständig sind.

virtuelle-maschinen.component.spec.ts

Automatisch generierte Datei für Unit-Tests.

virtuelle-maschinen.component.ts

Enthält die Klasse „VirtuelleMaschinenComponent", welche das Auflisten der virtuellen Maschinen ermöglicht. Hier sind entsprechende Methoden implementiert um alle benötigten Interaktionen aus der zugehörigen .html-Datei zu realisieren. Ebenso um die benötigten Daten über den Node.js-Server vom Satelliten-Server abrufen zu können.

5.16 virtuelle-maschinen-dialog

Diese Komponente bittet den Benutzer um eine Bestätigung, ob die ausgewählten virtuellen Maschinen wirklich gelöscht werden sollen. Der Anwender hat die Möglichkeit zu bestätigen oder abzubrechen. Im Falle einer Bestätigung, werden alle virtuellen Maschinen gelöscht für welche der Benutzer eine ausreichende Berechtigung besitzt.

virtuelle-maschinen-dialog.component.css

Diese Datei besitzt derzeit keinen Inhalt.

virtuelle-maschinen-dialog.component.html

In dieser Datei werden die jeweiligen Elemente generiert, welche für den Löschvorgang notwendig sind.

virtuelle-maschinen-dialog.component.spec.ts

Automatisch generierte Datei für Unit-Tests.

virtuelle-maschinen-dialog.component.ts

Enthält die Klasse „VirtuelleMaschinenDialogComponent", welche das Löschen der virtuellen Maschinen ermöglicht. Hier sind entsprechende Methoden implementiert um alle benötigten Interaktionen aus der zugehörigen .html-Datei zu realisieren. Im Anschluss gibt die Komponente die getätigte Auswahl wieder zurück, damit die virtuelle-maschinen-Komponente das Löschen übernehmen kann.

5.17 login.service.ts

Diese Datei ermöglicht eine Kommunikation mit dem Node.js-Server und sorgt dafür, dass sich ein Benutzer anmelden und abmelden kann. Zusätzlich kann durch diese Datei der Satelliten-Server gesetzt werden, welcher vom Benutzer verwendet wird.

5.18 user.service.ts

Diese Datei dient dazu, die Liste der Benutzer über den Node.js-Server vom Satelliten-Server zu holen.

5.19 user.ts

Innerhalb dieser Datei befinden sich drei Klassen:

UserInfo:

Enthält Attribute für demographische Daten eines Benutzers sowie die „userId".

Satellite:

Beschreibt einen Satelliten-Server, welcher in einem UserData Objekt enthalten ist.

UserData:

Enthält die aktuelle „sessionId", das „authToken", ein Array von „Satellite" und ein Objekt „UserInfo".

5.20 veranstaltung.ts

Diese Datei enthält mehrere Klassen, welche alle in Relation zu einer Veranstaltung stehen. Nachfolgend wird auf jede dieser Klassen kurz eingegangen:

LectureSummary:
Alle Veranstaltungen für die Auflistung in der veranstaltungen-Komponente, welche vom Server abgerufen werden, werden in diesem Format zurückgeliefert.

LectureRead:
Wenn die veranstaltung-Komponente für die Detailansicht und Bearbeitung einer einzelnen Veranstaltung aufgerufen wird, dann ist das Veranstaltungs-Objekt, welches vom Server gesendet wird, von diesem Datentyp.

LdapFilter:
Diese Klasse wird innerhalb der LectureRead-Klasse verwendet um die LDAP-Filter zu verwalten.

NetShare:
Diese Klasse wird innerhalb der LectureRead-Klasse verwendet um die Netzlaufwerke zu verwalten.

NetRule:
Diese Klasse wird innerhalb der LectureRead-Klasse verwendet um die Netzwerkregeln zu verwalten.

LecturePermissions:
Wird in mehreren Klassen von einer Veranstaltung für die Standardberechtigungen und die benutzerspezifischen Berechtigungen verwendet.

LectureWrite:
Wenn eine neue Veranstaltung erstellt werden soll oder die Metadaten einer bestehenden Veranstaltung geupdated werden sollen, muss das zum Server gesendete Objekt von dieser Klasse sein.

Location:
Diese Klasse ist für die Raumauswahl der Veranstaltung essentiell und kommt gleich in mehreren Klassen zum Einsatz.

5.21 veranstaltungen.service.ts

Diese Datei ermöglicht den Datenaustausch zwischen dem Angular-Client und dem Node.js-Server bezogen auf die Metadaten einer Veranstaltung. Hier befinden sich die jeweiligen Methoden für POST, GET, PUT und DELETE.

5.22 vm.ts

In dieser Datei befinden sich mehrere Klassen, welche alle im Zusammenhang mit einer virtuellen Maschine stehen. Nachfolgend wird auf jede dieser Klassen kurz eingegangen:

ImageSummaryRead:

Alle virtuelle Maschinen für die Auflistung in der virtuelle-maschinen-Komponente, welche vom Server abgerufen werden, werden in diesem Format zurückgeliefert.

ImageDetailsRead:

Wenn die virtuelle-maschine-Komponente für die Detailansicht und Bearbeitung einer einzelnen VM aufgerufen wird, dann ist das VM-Objekt, welches vom Server gesendet wird, von diesem Datentyp.

ImageVersionDetails:

Diese Klasse gibt Auskunft über eine einzelne VM-Version.

OperatingSystems:

Die Liste der Betriebssysteme, welche vom Server abgerufen wird, wird in diesem Format zurückgegeben.

ImagePermissions:

Wird in mehreren Klassen einer VM für die Standardberechtigungen und die benutzerspezifischen Berechtigungen verwendet.

ImageBaseWrite:

Wenn eine neue VM erstellt werden soll oder die Metadaten einer bestehenden VM geupdated werden sollen, muss das zum Server gesendete Objekt von dieser Klasse sein.

5.23 vms.service.ts

Diese Datei ermöglicht den Datenaustausch zwischen dem Angular-Client und dem Node.js-Server bezogen auf die Metadaten einer virtuellen Maschine. Hier befinden sich die jeweiligen Methoden für POST, GET, PUT und DELETE.

6 Dateienbeschreibung: Node.js-Server

Im Nachfolgenden werden alle Dateien beschrieben, welche sich im Verzeichnis lehrpool-nrw-NodeJS/ befinden.

6.1 gen-nodejs/bwlp_types.js

In dieser Datei sind alle Objekte für die Übertragung via Thrift beschrieben. Diese Datei wurde von Thrift erstellt.

6.2 gen-nodejs/MasterServer.js

Diese Datei beinhaltet alle Funktionen, zur Kommunikation mit dem Master-Server. Diese Datei wurde von Thrift erstellt.

6.3 gen-nodejs/SatelliteServer.js

Diese Datei beinhaltet alle Funktionen, zur Kommunikation mit dem Satelliten-Server. Diese Datei wurde von Thrift erstellt.

6.4 module/functions.js

In dieser Datei liegen Funktionen, die einige Daten der virtuellen Maschinen, der Veranstaltungen und der Fehlermeldungen vorbereiten und so anpassen, dass sie anschließend im richtigen Format an Angular weitergegeben werden können. Dies ist nötig damit Angular die Daten korrekt in der Web-Anwendung darstellen kann. Des Weiteren gibt es in der Datei eine Timeout-Schedule-Funktion für den User, die dafür sorgt, dass jede Minute der Countdown um eins dekrementiert wird.

6.5 module/thrift.js

Die Datei „thrift.js" ist grundsätzlich für die Verbindung zwischen dem Satelliten-Server, dem Master-Server und dem Node.js-Server da. Hier liegen alle benötigten Funktionen zum Kommunizieren mit den Servern vor. Falls es zu einem Fehler kommt, wird dieser direkt in dem Ordner „logs" in der Datei „tserverlog.log", samt aktueller Zeit geloggt. Dieses Modul beinhaltet eine Klasse, von der ein Objekt erstellt werden kann, welches die genannten Funktionen beinhaltet. Zudem bietet diese Klasse die Möglichkeit, Verbindungen zu den, in der „tconfig.json" genannten Servern, herzustellen. Auch für das Einlesen der Config-Datei ist die Klasse zuständig. In der Datei „thrift.js" befindet sich zudem eine Funktion, mit der es möglich ist, eine Exception in dem benötigten Format zu erstellen.

Als Übergabeparameter werden dort eine Nachricht für den Benutzer, ein HTTP-Fehlercode und ein weiterer Fehlercode zur Kommunikation mit dem Frontend verwendet. Um nicht bei jedem Funktionsaufruf neu überprüfen zu müssen, ob alle benötigten Voraussetzungen erfüllt sind, sind diese in „Check"-Funktionen ausgelagert, die die Verfügbarkeit der Informationen testen und je nach Bedarf am Anfang der Funktionen aufgerufen werden können und gegebenenfalls eine Exception werfen.

6.6 module/user.js

In der Datei „user.js" werden die Benutzer verwaltet. Sobald ein neuer Benutzer erstellt wird, wird hier ein neues „User"-Objekt mit dem eingegebenen Satelliten-Server erstellt. Sollte ein Test-User verwendet werden, kann vorher ein „User"-Objekt mit Name und Passwort für die Anmeldung auf dem Master-Server erstellt werden. Die Adresse des jeweiligen Satelliten-Servers wird vom Client mit übergeben. Mit dieser Serveradresse wird anschließend die zugehörige interne Server-ID gesucht. Außerdem werden hier die Timeout-Werte der User erstellt und verwaltet.

6.7 index.js

Die Indexdatei bildet die Hauptdatei des Node.js-Servers. Das bedeutet, dass sämtliche Module in dieser Datei geladen und angesprochen werden. Hier liegen die Routing-Funktionen für alle Datenzugriffe, wie z.B. die der Veranstaltungen und der virtuellen Maschinen. Diese Routen greifen auf die verschiedenen Methoden der „thrift.js" zu und senden das Ergebnis dieser Methoden zurück. Sollte dabei ein Fehler entstehen wird auf die Funktion „send" der „functions.js" zurückgegriffen. Außerdem werden zeitgleich Fehler inklusive der aktuellen Zeit geloggt. Diese Logs sind anschließend in dem Ordner „logs" in der Datei „serverlog.log" zu finden.

6.8 tconfig.json

In dieser Konfigurationsdatei sind sowohl alle Satelliten-Server samt Namen, als auch der Master-Server notiert.

7 Benutzernamen und Passwörter

Nachfolgend werden alle Benutzernamen und Passwörter aufgelistet, welche für die Verwendung des bereitgestellten Softwarepakets von Nöten sein könnten.

7.1 Satelliten-Server

Innerhalb der VM:

Benutzername: root

Passwort: root

Webschnittstelle:

Benutzername: admin

Passwort: root

7.2 Testaccount für den Angular-Client

Benutzername: niclas@test

Passwort: root