Technology
Arts Sciences
TH Köln

DOKUMENTATION

Web-basierte Anwendungen 2: Verteilte Systeme

Prof. Dr. Kristian Fischer

Gruppe 11 (EMA):

Esranur Bulut

Mark Schwott

Aysenur Cavusoglu

01. Juli 2016

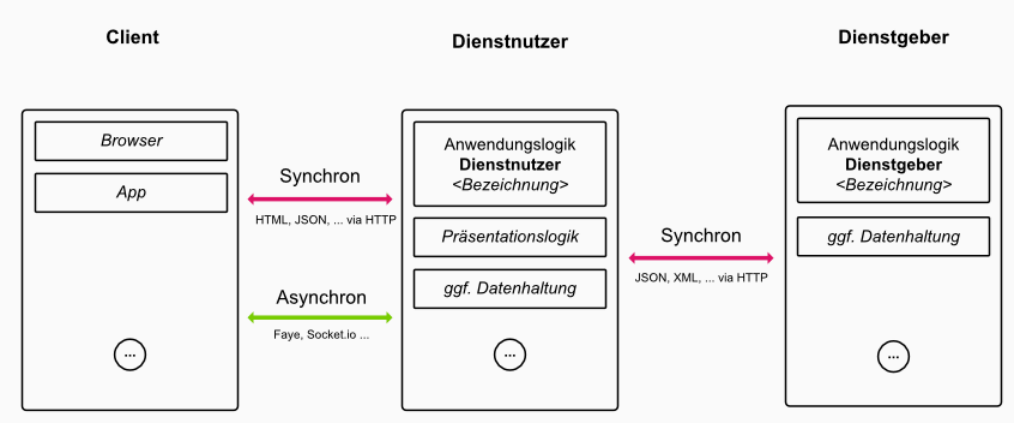
INHALTSVERZEICHNIS

EINLEITUNG

Diese Dokumentation beinhaltet eine detaillierte Beschreibung der praktischen Umsetzung eines verteilten Systems, das im Rahmen des Moduls “Web-basierte Anwendungen 2” entwickelt wird. Zunächst wird die Aufgabenstellung erläutert. Es werden Szenarien und Use-Cases beschrieben. Die Ressourcen werden mit ihrer Semantik und Methoden angegeben und spezifiziert. Dann wird das Arbeitsergebnis ausführlich dokumentiert. Anschließend wird eine Reflexion des Erreichens und des Nicht-Erreichens in einem Fazit beschrieben.

AUFGABENSTELLUNG

Das Ziel des Workshops ist es eine verteilte Web Anwendung zu entwickeln, die folgender Referenzarchitektur entspricht:



Es soll ein Konzept für ein verteiltes System ausgearbeitet werden. In der ersten Phase wird ein REST Dienstanbieter (Service) auf der Basis von *node.js* konzipiert und erstellt. In der zweiten Phase wird für den Dienst ein Dienstnutzer konzipiert. Dieser besteht seinerseits aus einem *node.js*-basierten Web Server und einem Browser als Nutzungsschnittstelle. Die Arbeitsergebnisse (Code) und der Projektdokumentation erfolgen mittels Github.

DOKUMENTATION DES SERVICE

Expose

Problemstellung:

Viele Medieninformatik-Studenten der TH-Köln müssen für bestimmte Module ein Video drehen, sei es ein Spielfilm oder ein Dokumentarfilm. Dazu kann das benötigte Equipment von der TH-Köln ausgeliehen werden. Jedoch gibt es das folgende Problem, dass die Betreuer im MI-Verleih nicht immer anwesend sind oder sogar auch manchmal Ausrüstungen für die Kamera, Ton etc. fehlen und bereits ausgeliehen worden sind.

Problemlösung / Idee:

Zu Beginn des Workshops wurden sich Gedanken zwischen der Fachschaft und dem Mi-Verleihsystem gemacht. Nach langen Überlegungen hat sich das Team für das Mi-Verleihsystem entschieden. Die Medieninformatik-Studenten können ganz einfach über dieses System Equipments wie z.B. Kamera, Beleuchtung und Ton für ihre Filmprojekte ausleihen. Sie bekommen aktuell zu dem Zeitpunkt verfügbare Geräte angezeigt und sie können je nach Auswahl das benötigte Equipment ausleihen. Die schon bereits ausgeliehenen Geräte d.h. die nicht verfügbaren Geräte werden ebenfalls in der Anwendung angezeigt und mit einem voraussichtlichen Rückgabedatum gekennzeichnet. Eine Vormerkung der Equipments ist möglich. Die Anwendung wird ständig vom Dienstgeber aktualisiert und bearbeitet. Ein Equipment kann neu angelegt oder gelöscht werden. Falls Fragen zum Equipment auftreten, können die Studenten einfach eine Nachricht zurücklassen.

RESSOURCEN

Am Anfang dieses Projektes hat sich das Team Gedanken über Ressourcen passend für ihr Projekt gemacht. Anschließend wurde eine Ressourcentabelle mit den zugehörigen http-Verben und deren Semantik erstellt.

Diese Ressourcen, die zu Beginn erstellt wurden:

„/equipment/kamera“	→ Ober-Ressource
„/equipment/kamera/videokamera/:id/”	→ Unter-Ressource
„/equipment/kamera/fotokamera/:id/“	→ Unter-Ressource

„/equipment/ton“ → Ober-Ressource

„/equipment/ton/aufnahmegeraet/:id/” → Unter-Ressource

„/equipment/ton/richtmikrofon/:id/” → Unter-Ressource

“/equipment/ton/kabel/:id/” → Unter-Ressource

„/equipment/beleuchtung“ → Ober-Ressource

“/equipment/beleuchtung/led_beleuchtung/:id/” → Unter-Ressource

“/equipment/beleuchtung/scheinwerfer/:id/” → Unter-Ressource

Im Laufe des Projektes wurde es klar, dass die Ressourcen unnötig spezialisiert wurden. Deshalb hat das Team die Ressourcen geändert und allgemein gehalten, um eine bessere Umsetzung zu gewährleisten. Zusätzlich wurde die Ressourcentabelle noch mit den Ressourcen: „/nachrichten“ und „/equipment/ausleihliste“ erweitert. Des Weiteren enthält die Tabelle die Ressourcen „/users“ und „/users/:id“, die von Anfang an beibehalten wurden.

Endgültig ergaben sich also folgende Ressourcen mit den entsprechenden http-Verben und deren Semantik. Anhand dieser Spezifikation wird klar, welche Ressourcen letztendlich definiert wurden, welche Methoden verwendet wurden und die Bedeutung der Semantik.

Spezifikation REST Webservice

Ressource: Kameras

Ressource	Methode	Semantik	Content-type (req)	Content-type (res)
/equipment/kameras	GET	Liste aller verfügbaren Kameras	text/plain	application/json
/equipment/kameras	POST	Legt eine neue Kamera an	application/json	application/json
/equipment/kameras/:id	PUT	Aktualisiert die Informationen einer Kamera	application/json	application/json

/equipment/kameras/:id	GET	Gibt unter ID folgende Kamera aus	application/json	application/json
/equipment/kameras/:id	DELETE	Löscht eine Kamera aus der Liste	text/plain	text/plain

Ressource: Ton

Ressource	Methode	Semantik	Content-type (req)	Content-type (res)
/equipment/ton	GET	Liste aller Ton-Equipments	text/plain	application/json
/equipment/ton	POST	Legt eine neues Ton-Equipment an	application/json	application/json
/equipment/ton/:id	PUT	Aktualisiert die Informationen eines Ton-Equipments	application/json	application/json
/equipment/ton/:id	GET	Gibt unter ID folgendes Ton-Equipment aus	application/json	application/json
/equipment/ton/:id	DELETE	Löscht ein Ton-Equipment aus der Liste	text/plain	text/plain

Ressource: Beleuchtung

Ressource	Methode	Semantik	Content-type (req)	Content-type (res)
/equipment/beleuchtung	GET	Liste aller Beleuchtung-Equipments	text/plain	application/json
/equipment/beleuchtung	POST	Legt eine neues Beleuchtung-Equipment an	application/json	application/json
/equipment/beleuchtung/:id	PUT	Aktualisiert die Informationen	application/json	application/json

		eines Beleuchtung-Equipments		
/equipment/beleuchtung/:id	GET	Gibt unter ID folgendes Beleuchtung-Equipment aus	application/json	application/json
/equipment/beleuchtung/:id	DELETE	Löscht ein Beleuchtung-Equipment aus der Liste	text/plain	text/plain

Ressource: User

Ressource	Methode	Semantik	Content-type (req)	Content-type (res)
/users	GET	Liste aller User	text/plain	application/json
/users	POST	Legt einen neuen User an	application/json	application/json
/users/:id	PUT	Aktualisiert die Informationen eines Users	application/json	application/json
/users/:id	DELETE	Löscht User	text/plain	text/plain
/users/:id	GET	User anzeigen	application/json	application/json

Ressource: Nachrichten

/nachrichten/	GET	Liste aller Nachrichten	text/plain	application/json
/nachrichten/	DELETE	Löscht eine Nachricht	text/plain	text/plain
/nachrichten/	POST	Versendet Nachricht	application/json	application/json

Ressource: Ausleihliste

Ressource	Methode	Semantik	Content-type (req)	Content-type (res)
/equipment/ausleihliste	GET	Ausleihinformation über alle verliehene Equipments	text/plain	application/json

GET: Anforderung der Repräsentation einer Ressource (Lesezugriff)
POST: Anlegen einer neuen Ressource. Rückgabe ist URI (Schreibzugriff)
PUT: Erstellen und bearbeiten von Ressourcen (Aktualisierung)
DELETE: Löschen von Ressourcen

ANWENDUNGSLOGIK UND DATENHALTUNG DES SERVICE

Im Folgenden sollen die Anwendungslogik und die Datenhaltung des implementierten Webservice beschrieben und Überlegungen zu dieser aufgezeigt werden. Der Webservice kümmert sich um die persistente Datenhaltung der erstellten Ressourcen und stellt diese auch entsprechend der Anfragen des Dienstnutzers zur Verfügung.

Bei einem GET z.B. auf die Ressource „/equipment/kameras“ kann nun über die ID einfach der passende Inhalt gefunden und an den Sender der Anfrage gesendet werden. Mit PUT und DELETE können somit auch einzelne Elemente der Equipments nach ihrer ID geändert, bzw. gelöscht werden.

Bei einem GET auf den „/user/:id“ wird der Username vorausgesetzt. Hier lädt der Server wieder das Userverzeichnis aus der Datenbank und geht dieses wieder durch. Wenn er den passenden User gefunden hat filtert er diesen heraus und schickt ihn an den Client zurück. Falls kein User mit diesem Namen vorhanden sein sollte wird ein Fehlercode gesendet. PUT und DELETE arbeiten bei „/user/:id“ nach der ID. Hier wird der User nach der ID, wie bei einem GET, gesucht und es können Daten geändert oder im Falle von DELETE einzelne User aus der Liste aller User gelöscht werden.
Zudem besteht die Möglichkeit, sich über eine GET Anfrage auf die Ressource „/users“ ohne ID oder Namen eine Liste aller User ausgeben zu lassen. Falls der User nicht gefunden wird, wird ein Status 404 an den Dienstnutzer zurück geschickt mit der Meldung, dass der User nicht gefunden werden konnte.

Für die Persistente Datenhaltung werden die Objektdaten im JSON-Format in der REDIS-Datenbank gespeichert. Die Objekte können durch ansprechen der definierter Keys

referenziert werden. Für die Equipments werden diese mit dem Key: „/equipments/kameras/“ in der Datenbank gespeichert. Jedes Equipments bekommt eine eindeutige ID.

Um eine persistente Datenhaltung zu ermöglichen wurden drei separate Dateien auf dem eigenen Rechner erstellt.

- ‚kameras.json'
  Diese Datei beschreibt die Listeressource von Kamera-Objekten. Folgende Syntax beschreibt ein solches Objekt:

  {"kameras":[{"bezeichnung":"Nikon D5300","sensor":"Halbformat","megapixel":"24"}]}

- ‚ton.json'
  Diese Datei beschreibt die Listeressource von Ton-Objekten. Folgende Syntax beschreibt ein solches Objekt:

  {"ton":[{"equipment":"Mikrofon","name":" M 130 DYNAMISCHES DOPPELBAENDCHEN","merkmale":" Uebertragungsbereich Mikrofone: 40 - 18,000 Hz, Richtcharakteristik Mikrofon:8"]}

- 'beleuchtung.json'
  Diese Datei beschreibt die Listenressource der von Beleuchtung-Objekten. Folgende Syntax beschreibt ein solches Team-Objekt:

  {"beleuchtung":[{"name":"Tec Take 3er SET Profi", "infos":"Leuchtmittel ca. 55 W, Stativ ausziehbar"}]}

- ‚users.json'
  Diese Datei beschreibt die Listenressource von den User-Objekten. Folgende Syntax beschreibt ein solches Objekt:

  {"users":[{"name":"Max"},{"studiengang":"Medieninformatik"},{"semester":"4"},{"funktion":"Administrator"}]}

Ziel des Teams war es das Verwalten der Daten möglichst einfach und mit wenig Aufwand zu gestalten und hierfür wurde eben die Möglichkeit der separaten JSON-Dateien gewählt.

NICHT IMPLEMENTIERTE FUNKTIONALITÄT

Einige Funktionalitäten konnten auf Grund Zeitmangel nicht umgesetzt werden. Jedoch bemüht sich das Team die Funktionen im Laufe des Arbeitsprozesses hinsichtlich der Logik zu verbessern. Da Unklarheiten bei der Benennung der Ressourcen aufgetreten sind und diese nicht dem REST-Prinzip entsprachen, wurden einige Funktionalitäten verworfen Dies war auch gleichzeitig die größte Herausforderung bei der Definition der Ressourcen.

DOKUMENTATION DES DIENSTNUTZERS

Szenarien

Szenario 1: Equipment ausleihen

Max studiert an der Technischen Hochschule Köln und muss für das Modul AVM einen Spielfilm drehen. Dafür muss er von der TH-Köln Kamera-Equipments ausleihen. Er loggt sich mit seiner Mi-Kennung im System der MI-Verleih ein. Nach einem erfolgreichen Login ist er berechtigt Equipments für die Ausleihe auszuwählen. Die ausgewählten Equipments werden in einer Ausleihliste abgespeichert und für das Buchen bereitgestellt. Sobald er mit dem Auswählen fertig ist, kann er per Klick auf den Button der Ausleihliste diese Selektion beenden. Anschließend wird der Dienstgeber benachrichtigt. Der Server schickt eine Benachrichtigungsmail an den Dienstgeber und eine Bestätigungsmail an Max (User). Falls die Ausleihe fehlschlägt, bekommt Max eine Fehlermeldung 404 mit dem Inhalt, dass er zurzeit nicht ausleihberechtigt ist und er sich an den Dienstgeber wenden kann.

Szenario 2: Equipments vormerken

Max sucht in der Navigation des MI-Verleihs eine bestimmte Kamera und entscheidet sich anschließend für „Canon EOS 5D Mark III“. Aktuell ist das Equipment nicht verfügbar. Jedoch bemerkt Max, dass er sein gewünschtes Equipment über das „Vormerkung“ Button vormerken lassen kann. Damit er nachher das verfügbare Equipment buchen kann, muss Max für die Vormerkung seine E-Mail-Adresse angeben. Falls das Equipment wieder vorhanden ist, bekommt Max per E-Mail eine Benachrichtigung.

Szenario 3: Nachricht verfassen

Max hat seinen Spielfilmdreh früher beendet als erwartet, aufgrund dessen möchte er das Equipment vor dem vereinbarten Rückgabedatum zurückgeben. Er loggt sich im System ein und geht in der Navigation auf den Nachrichten-Button und sendet eine Nachricht mit der Bitte für eine vorzeitige Rückgabe. Der Dienstgeber wird somit für die gewünschte Rückgabe

informiert, somit kann der Dienstgeber mit Max einen erneuten Rückgabetermin vereinbaren und es anschließend im System vermerken.

USE-CASES

Use Case: Equipment ausleihen

actors: User (Max)

precondition: Max muss auf dem Server angemeldet sein.

main flow:

1. Max wählt unter „Equipments“ in der Navigation die gewünschten Equipments zum Ausleihen aus.
2. Die ausgewählten Equipments werden in einer Ausleihliste abgespeichert.
3. Per Klick auf den Button der Ausleihliste wird die Ausleihe durchgeführt.
4. Wenn alles erfolgreich durchgeführt wurde, wird der Dienstgeber vom Server benachrichtigt und Max bekommt eine Bestätigungsmail zurück.

alternative flow: Ausleihe nicht erfolgreich

5a. Max bekommt eine Fehlermeldung 404 und muss sich an den Dienstgeber wenden.

postcondition: Max hat erfolgreich Equipment ausgeliehen.

end: Equipment ausleihen

Use Case: Equipment vormerken

actors: User (Max)

precondition: Max muss auf dem Server angemeldet sein.

main flow:

1. Max sucht nach einer bestimmten Kamera für sein Filmprojekt.
2. Per Klick auf „Kameras“ in der Navigation, werden alle vorhandenen Kameras mit Informationen angezeigt.
3. Jedoch findet er sein gewünschtes Equipment nicht. Dafür bemerkt er, dass eine „Vormerkung“-Button existiert.
4. Max gibt seine E-Mail-Adresse an und das Equipment wird vorgemerkt.

5. Falls Equipment wieder vorhanden ist, bekommt Max per Mail eine Benachrichtigung.

postcondition: passende Kamera erfolgreich gefunden

end: passende Kamera suchen

Use Case: Nachricht verfassen

actors: User (Max)

precondition: Max muss auf dem Server angemeldet sein und bereits ausgeliehen haben.

main flow:

1. Max möchte das ausgeliehene Equipment vor dem vereinbarten Rückgabedatum zurückgeben.
2. In der Navigation kann Max unter „Nachrichten“ eine Nachricht verfassen.
3. Max schreibt eine Nachricht in das leere Eingabefeld mit der Bitte für eine vorzeitige Rückgabe.
4. Anschließend bestätigt er die Eingabe und wartet evtl. auf eine Antwort.

postcondition: Max hat erfolgreich eine Nachricht verfasst

end: Nachricht verfassen

ANWENDUNGSLOGIK UND DATENHALTUNG DES DIENSTNUTZERS

Die Anwendungslogik des Dienstnutzers besteht darin, dass Equipments von Usern ausgewählt, in einer Auswahlliste gespeichert und schließlich ausgeliehen werden können. Falls gesuchte Equipments nicht zu dem Zeitpunkt vorhanden sind, können die User diese dann vormerken lassen. Es besteht die Möglichkeit User, welche Equipments vorgemerkt haben mit Benachrichtigungsmail über wieder vorhandene Equipments zu informieren. Des Weiteren wird das Modul Faye verwendet und dies ermöglicht dem User Nachrichten an den Administrator zu schicken. Für die Datenhaltung wurde das Modul Faye als Engine verwendet, welches mit dem Redis-Server kommuniziert.

PRÄSENTATIONSLOGIK DES DIENSTNUTZERS

Das System wurde mit Hilfe von Bootstrap realisiert, um ein einfaches Responsive Webdesign zu ermöglichen. Um die Ressourcen im Browser nutzbar zu machen wurden verschiedene EJS-Dateien angelegt. Das Modul EJS ist ein Template Engine basierend auf JavaScript und ermöglicht verschiedene Views und Darstellungen z.B. bei einem GET auf /users, eine Schleife zu durchlaufen und alle User mit den eingestellten Informationen darzustellen. Im Dienstnutzer findet hierbei das Rendern der EJS-Datei in HTML statt. Für den Abruf der Dienstnutzer-Seite im Browser wird „localhost:3001“ verwendet.

BESCHREIBUNG DER ASYNCHRON IMPLEMENTIERTEN TEILE

Die asynchrone Kommunikation wird hier bei der späteren Information per Nachricht angewendet. Ist das gewünschte Equipment nicht zu diesem Zeitpunkt vorhanden, also bereits ausgeliehen, bekommt der Dienstnutzer eine Nachricht sobald dieses Equipment wieder ausleihbar ist. Der Dienstnutzer wartet also auf die Antwort, somit ist die Kommunikation zeitlich versetzt. Des Weiteren kann der Dienstnutzer Nachrichten an Administratoren senden. Diese Funktion wurde mit Hilfe von Faye implementiert.

NICHT IMPLEMENTIERTE FUNKTIONALITÄT

Funktionalitäten wie z.B. das „Ausleihen der Equipments von Usern“ oder „Bestätigung-Mails vom Server an den User schicken“ konnte auf Grund Zeitmangel nicht umgesetzt werden. Das Team bemüht sich aber im Laufe des Arbeitsprozesses diese Funktionalitäten noch umzusetzen. Ständig wurden die Funktionen umgeschrieben und geändert, somit war es schwierig ein funktionales Endergebnis zu erzielen. Einige Ideen wurden verworfen, da sie viel zu komplex für das Team waren.

DOKUMENTATION DES PROZESSES

Die Architektur des MI-Verleih Systems basiert auf dem Framework Node.js. Dieses Framework ermöglicht externe Module zu installieren. Die Module können mit dem integrierten Paketmanager „NPM“ erweitert werden und werden über eine externe JavaScript Datei eingebunden. Zum Einsatz kommen externe Module wie express und REDIS. Die Daten, wie die registrierten User und die Ausleihliste werden in einer Datenbank REDIS gespeichert.

VORGEHENSWEISE UND IRRWEGE

Die Vorgehensweise richtet sich generell an die Vorgaben, welche im Workshop gemacht wurden. Hierzu gehören zum einen die Erstellung einer Ressourcentabelle und zum anderen eine Auflistung von Use-Cases. Im weiteren Verlauf des Projekts hat die Ressourcentabelle eine große Mitwirkung der Erstellung des Service. Viele Ressourcen wurden abgeändert, da sie in der Anwendung nicht so verwendet werden konnten, wie es im Vorfeld geplant war.

Im Laufe des Projekts gab es nicht nur zeitlich Probleme, sondern auch beim Programmieren stellten sich Probleme heraus. Das System sollte ursprünglich dem User eine Erinnerungsmail senden, damit die Leihfrist nicht überschritten wird. Diese Funktion stellt die asynchrone Kommunikation dar. Aus zeitlichen Gründen konnte das Team diese Funktion nicht implementieren.

Zu Beginn der Programmierung des Service musste sich das Team zuerst mit Node.js vertraut machen, da bei einigen Mitgliedern des Teams bereits Vorkenntnisse in JavaScript vorhanden waren. Diese Sprache serverseitig zu verwenden war für alle unbekannt. Weitere Probleme entstanden auch bei REST, weil REST für das Team zunächst um ein unbekanntes Prinzip handelte, dementsprechend wurden auch Fehler gemacht. Das System wurde aber ständig überarbeitet und verbessert. Die Kommunikation zwischen den Mitgliedern und die Arbeitsverteilung wurde problemlos durchgeführt.

FAZIT

Das Projekt hat die Möglichkeit geboten, einen Einblick in die Technologien moderner Anwendungen zu bekommen. Zudem hat das Modul gezeigt, dass ein verteiltes System komplexer sein kann, als bisher gedacht. Zwar hatte das Team Probleme ein gutes Konzept zu entwickeln und in das in den Workshops vermittelte Wissen einzuarbeiten, doch jedes Teammitglied hat mit voller Disziplin gearbeitet. Diese Probleme konnten jedoch Dank der Tutoren Jorge Pereira und Sheree Saßmannshausen beseitigt werden. Der Prototyp funktioniert weitergehend aber für eine endgültige Realisierung müssten dennoch die Probleme überarbeitet und die Erweiterungsideen umgesetzt werden, um die Applikation für die Benutzer interessant zu machen. Letztlich wurde ein funktionales System erstellt, welches den aus der Vorlesung behandelnden Stoff umsetzt. Das Projekt in seiner Gesamtheit hat sich für das Team als schwierig erwiesen, da das Semester viel zu kurz war und die angesetzte Zeit für die Umsetzung des Projekts nicht ausreichend war.

ARBEITSMATRIX

Person	Dienstgeber	Dienstnutzer	Dokumentation
Esranur Bulut	33,3%	15%	42,5%
Mark Schwott	33,3%	70%	15%
Aysenur Cavusoglu	33,3%	15%	42,5%