Architektur-Dokumentation für „ShareIt“

Autor und Projektleiter: Daniel Gabl

Inhalt

Einführung und Ziele

Dieser Abschnitt führt in die Aufgabenstellung ein und skizziert die Ziele von „ShareIt“.

Aufgabenstellung

Ziel dieses Projektes ist die Erschaffung eines Medien-Verleih-Services „ShareIt“, in dem sich jede Person registrieren kann. Sofern registriert kann eine Person dann ein Exemplar eines Mediums wie beispielsweise Bücher und CDs ausleihen. Diese Exemplare müssen obligatorischerweise wieder zurückgeben werden.

Qualitätsziele

Ziel	**Konkrete Umsetzung**
Unabhängige Autorisierung	Eigener Microservice mit eigener REST-Schnittstelle für Autorisierung
Hohe Verfügbarkeit	Erreicht durch Deployment auf Heroku
Geringe Antwortzeit	Server reagiert in unter einer Sekunde auf jede Anfrage
Einfache Erweiterbarkeit	Funktionen wurden so definiert, dass sie das System einfach um neue Medien erweitern lässt

Stakeholder

Rolle	Kontakt	Erwartungen
Studierende	*„Stuve“*	*Hohe Benutzerfreundlichkeit*
Verwaltung	*info@fs.cs.hm.edu*	*Einfach verwaltbar*
Administratoren	*vogler@hm.edu*	*Einfach wartbar*

Randbedingungen

- Allgemein
    - Das System kommt mit einer „Donationware“ Lizenz
    - Es müssen sowohl neue Benutzer als auch Medien angelegt werden können
    - Das System muss einen eigenen Autorisierungsservice mitbringen
    - Die Daten müssen persistiert werden im Fall eines Systemausfalls
    - Das System muss sowohl unter Linux als auch unter Windows laufen
    - Die Implementierung dieser Aufgaben erfolgt in eigenen Microservices
- Entwicklungsspezifisch
    - Das Backend des Servers ist in Java geschrieben,
      das Frontend wird im Web mit JS und der JS-Bibliothek „jQuery“ realisiert
    - Es sollen Variablennamen definiert werden, die auch ohne jegliche Dokumentation klar machen, was diese Variable beinhaltet
    - Variablen sind standardmäßig private und final deklariert
    - Medien-Objekte sollen immutable sein
    - Benutzen des vorgegebenen Checkstyles
    - Verwenden von „Maven“ zur Dependency-Verwaltung
    - Verwenden von „Jackson“ zur einfachen (De)serialisierung von Objekten
    - Verwenden von „Hibernate“ zur Persistierung von Daten
    - Verwenden von „log4j“ als Logging / Debugging-Tool

Kontextabgrenzung

Dieser Abschnitt beschreibt das Umfeld von „ShareIt“. Für welche Benutzer ist es da, und mit welchen Fremdsystemen interagiert es?

Fachlicher Kontext

Akteure	Beschreibung
Studierende	Ausleihen und Zurückgeben von existierenden Medien, updaten ihrer Nutzerdaten
Moderatoren / Administratoren	Anlegen von neuen Medien, Verwalten der existierenden Medien, Rückgabe checken
Autorisierungs-System	Eigener Microservice zur Autorisierung der Benutzer in der Anwendung
Datenbank-System	Persistieren von Medien in eine Datenbank durch einen eigenen Microservice

Technischer Kontext

HTML und JavaScript-Frontend

Die Anwendung wird über einen Browser (Frontend) aufgerufen. Daten wie Buchtitel, ISBN, Nutzername, etc. werden in Formular-Felder eingetragen und dann an das Backend (JS client-seitig und dann an den Server) weitergeleitet und dort verarbeitet.

jQuery-Backend

Daten des Formulars werden an jQuery weitergegeben, es wird eine JSON-Anfrage gebildet und diese wird an den Server unter einer bestimmten Adresse (bspw. media/books) geschickt. Danach wird der Response des Java-Programms empfangen und an den Nutzer weitergegeben (bspw. das Buch das gesucht wurde).

Java-Backend als Server

Hier werden Daten bspw. in der Media-Ressource-Datei empfangen, es wird eine Autorisierung abgefragt und wenn vorhanden an die Media-Service-Routine weitergeleitet. Später wird dann das Ergebnis des Servers erhalten und an den Browser zurückgegeben.

In der Routine werden dann in den Methoden alle möglichen Fehler geprüft, bspw. ob die ISBN für ein Buch korrekt ist, ISBN als Primary-Key schon vorhanden, alle Daten vollständig sind und wenn alle Tests bestanden sind an die Persistierungs-Routine weitergeleitet. Nach der Persistierung wird dann das Ergebnis an die Medien-Ressource zurückgegeben.

In der Persistierungs-Routine werden die Daten dann nur noch persistiert oder aus der Datenbank abgefragt und zurück an die vorherige Instanz gegeben.

URI-Template	Verb	Wirkung
User		
/users	POST	Neuen User anlegen Möglicher Fehler: User existiert bereits Möglicher Fehler: Name u. od. Passwort vergessen
/users/login	POST	Einloggen und damit neuen OAuth-Token generieren
/users	GET	Alle User auflisten
/users	PUT	User-Daten modifizieren (automatische Prüfung ob User in Service-Routine existiert) Möglicher Fehler: User nicht gefunden Möglicher Fehler: Name und Passwort fehlen Möglicher Fehler: Neue Daten gleich mit alten Daten
/users/{name}	GET	Bestimmten User-Namen suchen
Books		
/media/books	POST	Neues Medium 'Buch' anlegen Möglicher Fehler: Ungültige ISBN Möglicher Fehler: ISBN bereits vorhanden Möglicher Fehler: Autor oder Titel fehlt
/media/books/{isbn}	GET	Eine JSON-Repräsentation eines gespeicherten Buches liefern, falls vorhanden
/media/books	GET	Alle Bücher auflisten

URI-Template	Verb	Wirkung
/media/books	PUT	Daten modifizieren (automatische Prüfung ob ISBN in Service-Routine existiert) Möglicher Fehler: ISBN nicht gefunden Möglicher Fehler: Autor und Titel fehlen Möglicher Fehler: Neue Daten gleich mit alten Daten
Discs		
/media/discs	POST	Neues Medium 'Disc' anlegen Möglicher Fehler: Ungültiger Barcode Möglicher Fehler: Barcode bereits vorhanden Möglicher Fehler: Director oder Titel fehlt
/media/discs	GET	Alle Discs auflisten
/media/discs/{barcode}	GET	Eine JSON-Repräsentation einer gespeicherten Disc liefern, falls vorhanden
/media/discs	PUT	Daten modifizieren (automatische Prüfung ob Barcode in Service-Routine existiert) Möglicher Fehler: Barcode nicht gefunden Möglicher Fehler: Director, FSK und Titel fehlen Möglicher Fehler: Neue Daten gleich mit alten Daten

Object	Parameter	Input
User		
	name	String
	pass	String
	role	String ("USER", "ADMIN" or "ROOT")

Object	Parameter	Input
Books		
	author	String
	isbn	String
	title	String
Discs		
	barcode	String
	director	String
	fsk	Integer
	title	String

Lösungsstrategie

Zu Anfang gab es 3 wesentliche Komponenten:

- Die **Medien-Ressource**, die von jQuery und Jackson Objekte übermittelt bekommt
- Das **Medien-Service-Interface**, das konkrete Methoden der API definiert
- Die **Medien-Service-Implementierung**, die alle Methoden des Interfaces definiert

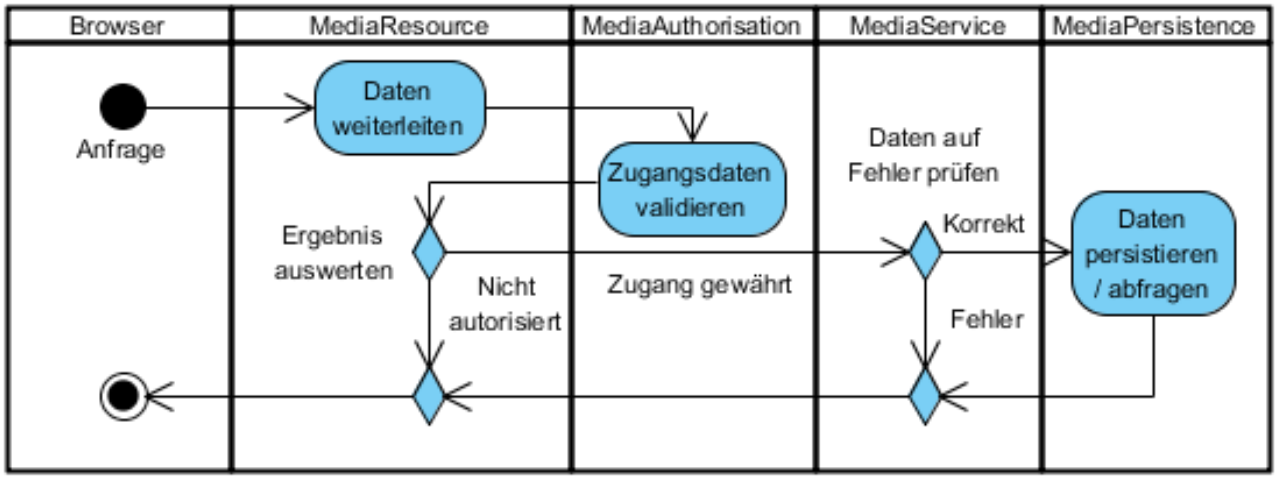
Die Ressource bekommt ein Medien-Service-Interface als Parameter übergeben, dieses wird beim Start mit einer neuen Medien-Service-Implementierung initialisiert.
Die Klasse hatte für jede Funktion der API (Medium hinzufügen, Medium aktualisieren, Medium suchen, etc.) eine Methode definiert, die als Parameter das Objekt von jQuery bekommen hat. Nun wurde dieses Objekt als Parameter weiter an das Interface gegeben, wo eine passende Methode mit diesem Objekt aufgerufen wird.

In der Medien-Service-Implementierung wurden nun die Daten auf ihre Korrektheit geprüft (bspw. ob ein Buch eine korrekte ISBN hat, alle Daten vollständig sind, etc.). Waren die Daten korrekt wurden diese gespeichert.

Die erste Umsetzung fand über Listen statt, in der Medien gespeichert worden sind. Im Fall eines korrekten Zugriffs (es wurden korrekte Daten an das System übermittelt) wurde das Medien-Objekt der Liste hinzugefügt. Beim Suchen nach diesem Medium wurde durch diese Liste iteriert und passende Objekte wurden zurückgegeben.

Nach dem Einbau einer Persistierungs-Ebene wurde das System mit Listen verworfen. Statt nun ein Objekt einer Liste hinzuzufügen wurde es zur Persistierung an die Datenbank-Schicht weitergereicht bzw. wurden bspw. Suchanfragen auch an die Persistenz-Schicht weitergeleitet.

Bausteinsicht



Whitebox Gesamtsystem

Begründung

Die Anwendung wurde in mehrere „Micro-Services“ unterteilt um das Schichtenmodell zu garantieren und einen Einstieg in das Projekt und die Wartbarkeit dieses zu vereinfachen.

Enthaltene Bausteine

- Medien-Ressource (wartet auf eingehende Requests über das Formular von ShareIt)
- Medien-Service (prüft weitergeleitete Eingaben / Objekte der Medien-Ressource)
- Autorisierungs-Service (autorisiert und validiert Nutzer von ShareIt)
- Persistierungs-Service (persistiert von Medien-Service geprüfte Objekte)

Wichtige Schnittstellen

3 Interfaces für die Autorisierung, die Persistierung und für den Medien-Service

Autorisierungs-Service

Erstellt Tokens für Nutzer bei einem Login und kann später prüfen, ob ein Benutzer einen gültigen Token hat bzw. ob für diesen Benutzer ein gültiger Token definiert ist.

Schnittstellen: MediaAuthorisationInterface

Erfüllte Anforderungen: Unabhängige Autorisierung

Ebene 1

Medien-Ressource

Wartet auf eingehende Requests. Per @Annotations wird der Pfad / die Request-URI definiert, auf die eine Methode horchen soll. Wird an diese URL Daten geschickt, so werden diese an die tiefer-liegende Schicht (Medien-Service) weitergeleitet.

Schnittstellen: Keine

Erfüllte Anforderungen: Schnelle Antwortzeit
- Kann mehrere Requests parallel handlen, gibt eine Antwort in unter einer Sekunde zurück

Ebene 2

Medien-Service

Wartet auf Aufruf durch die Media-Resource. Prüft erhaltene Daten auf Fehler (bspw. ISBN falsch, Medium schon vorhanden, etc.) und gibt bei diesen Rückmeldung an den Nutzer.

Schnittstellen: MediaServiceInterface

Ebene 3

Persistierungs-Service

Wartet auf Aufruf durch den Medien-Service zur Persistierung von geprüften Daten. Kann auch Suchanfragen zu Mediem bekommen, prüft diese und gibt eine Antwort zurück.

Schnittstellen:

MediaServiceInterface

Definiert Methoden zum Hinzufügen, Aktualisieren und Finden von Medien.

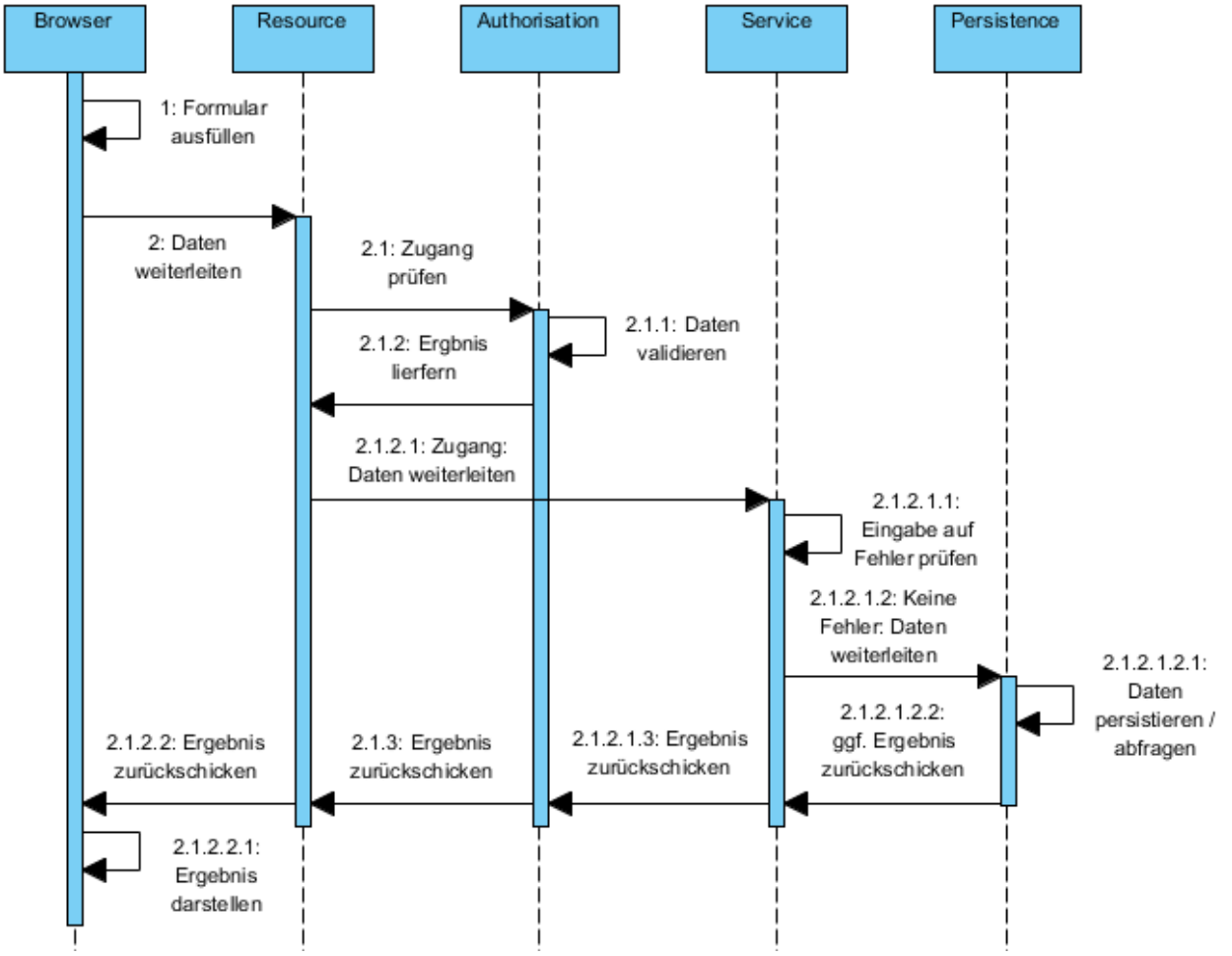
MediaAuthorisationInterface

Definiert Methoden zur Autorisierung (Registration, Login) und zur Validierung.

MediaPersistenceInterface

Definiert Methoden zum Persistieren und Datenbank-Abfragen von Medien.

Laufzeitsicht



Das Formular wird vom Benutzer über den Browser abgeschickt. Die Daten werden erst an die JavaScript-Bibliothek geschickt, dort wird eine Funktion aufgerufen, in der ein JSON-String gebildet und eine URL definiert wird und sendet den String an diese URL (bspw. /media/books/). Nun horcht eine Funktion in Server auf diese URL und behandelt eingehende Requests. Es wird erst geprüft, ob der Nutzer zu diesem Aufruf autorisiert ist, wenn ja werden die eingegangenen Daten eine Schicht nach unten an den Media-Service durchgereicht, wenn nicht wird ein Autorisierungs-Fehler an den User zurückgegeben.

Im Service werden nun die eingegangenen Daten auf Richtigkeit geprüft (ISBN korrekt, nicht bereits vorhanden, Daten vollständig, etc.). Falls dies der Fall ist, werden die Daten zur Persistierung frei gegeben bzw. für Abfragen an die Persistierung weitergeleitet, falls nicht wird ein entsprechender Fehler an den Clienten zurückgegeben (bspw. ISBN nicht korrekt).

In der Persistierungs-Routine werden Daten entweder in eine Datenbank geschrieben oder aus der Datenbank gelesen und an die höhere Schicht (Media-Service) zurückgegeben.

Verteilungssicht

Infrastruktur Ebene 1

<Übersichtsdiagramm>

Begründung
<Erläuternder Text>

Qualitäts- und/oder Leistungsmerkmale
<Erläuternder Text>

Zuordnung von Bausteinen zu Infrastruktur
<Beschreibung der Zuordnung>

Infrastruktur Ebene 2

<Infrastrukturelement 1>

<Diagramm + Erläuterungen>

<Infrastrukturelement 2>

<Diagramm + Erläuterungen>

...

<Infrastrukturelement n>

<Diagramm + Erläuterungen>

Querschnittliche Konzepte

<Konzept 1>

<Erklärung>

<Konzept 2>

<Erklärung>

...

<Konzept n>

<Erklärung>

Entwurfsentscheidungen

Qualitätsanforderungen

Qualitätsbaum

Qualitätsszenarien

Risiken und technische Schulden

Folgende technische Schulden sind definitiv **nicht** vorhanden:

nicht-vorhandene technische Schuld	**Begründung**
Implementationsschulden	Checkstyle benutzt, wirft keine Fehler mehr Code-Redundanzen weitestgehend ausgelagert
Dokumentationsschulden	Alle Funktionen mit Parametern und Klassenvariablen dokumentiert
Testschulden	89,6% (4611 Inst) Testabdeckung mit EclEmma trotz vieler ungetesteter Ressourcen-Klassen

Design- und Architekturschulden sollten ebenfalls nicht vorhanden sein, es wurde darauf geachtet, ein Schichtenmodell durchzusetzen / Package-Strukturen zu benutzen, jede Klasse hat genau eine Funktion und nicht mehr (Daten werden durchgereicht).

Glossar

Begriff	Definition
ShareIt	*Name der Anwendung / des Services*
Medium / Medien	*Inhalte unseres Services.* *Können Bücher, CDs, DVDs, etc. sein (nur einmal vorhanden)*
Exemplar	*Konkretes Medium wie bspw. ein Buch / Kopie eines Mediums*
Stuve	*Studierendenvertretung der Hochschule München*
Donationware	*Donationware ist eine Unterart der Freeware. Die Nutzung der Software ist grundsätzlich kostenlos, es wird jedoch um eine Spende gebeten, um die Urheber finanziell zu unterstützen und somit die Entwicklung der Software zu erleichtern.*
.	.