 Share-It

2017-06-28

# Einführung und Ziele

Ziel der Verleihplattform ShareIt ist es, den wechselseitigen Verleih von Lehr- und Lernmaterialien innerhalb einer eingegrenzten Gruppe von Studierenden zu organisieren und zu verwalten. Dieses Gesamtziel lässt sich gliedern in die folgenden drei Hauptaufgabenbereiche:

* Verwaltung von Benutzerdaten
* Verwaltung von Exemplaren
* Ausleihe und Ruckgabe der Exemplare

Standardablauf:

1. User loggt sich ein
2. User legt ein Buch an
3. User schaut nach ob dieses Buch eingetragen ist
4. User loggt sich aus

## Aufgabenstellung

**Inhalt.**

Siehe [Anforderungsdokument]

## Qualitätsziele

**Inhalt.**

|  |  |
| --- | --- |
| Qualitätsziele | Szenarien |
| Robustheit | System sollte bei fehlerhafter Eingabe weiter funktionieren |
| Modulares System | Einfach skalier- und erweiterbar |
| Anlegen von Büchern | User soll Bücher anlegen können |
| Einsehen von Büchern | User sollen den Bücherbestand angezeigt bekommen können. |
| Authenztifizierung von Usern | Zugangskontrolle für User |

## Stakeholder

|  |  |  |  |
| --- | --- | --- | --- |
| Rolle | Beschreibung | Abnahmerelevanz | Erwartungen |
| Administratoren | Wartung des Systems | Hoch | Einfach zu warten |
| Studenten – User | Benutzer des Systems | Hoch | Einfache Bedienbarkeit |
| Hochschulleitung | Finanzierer des Systems | Mittel | Kostengünstig |
|  |  |  |  |

# 

# Randbedingungen

**Lizenz**

GitHub repository.

Deployment auf Heroku. https://www.heroku.com/policy/tos

Nutzung von Hibernate (LGPL 2.1) und log42 (Apache License, Version 2.0).

**Technisch**

Modulares System.

Zielumgebung: Java auf Heroku.

Zeitliche Begrenzung bis zum 2.7.2017.

# Kontextabgrenzung

Standalone auf Heroku.

## Fachlicher Kontext

## Technischer Kontext

Deployment auf Heroku.

REST API mittels JSON

JSON Objekt Aufbau  
{  
 "name" : "bookName",  
 "author" : "name",  
 "ISBN" : number,  
}  
  
  
  
POST:  
Legt ein neues Buch an  
Respond: StatusCode:  
 - 200 alles ok buch wurde angelegt  
 - 400 es existiert bereits ein identisches buch  
 - 401 ungültige eingabe  
Respond JSON obj bei 400 und 401  
 {  
 "detail" : "Fehler beschreibung"  
 }  
  
Get:  
lieft information zu allen angelegten Büchern  
Resond Status Code:  
 - 200 alles ok  
 - 400 es gibt keine angelegten bücher  
Respond JSON obj bei 200  
 {  
 <Beliebige anzahl>  
 {  
 "name" : "bookName","author" : "name", "ISBN" : number,  
 }  
 <\Beliebige anzahl>

Aktuelle REST API im Repository.

# Lösungsstrategie

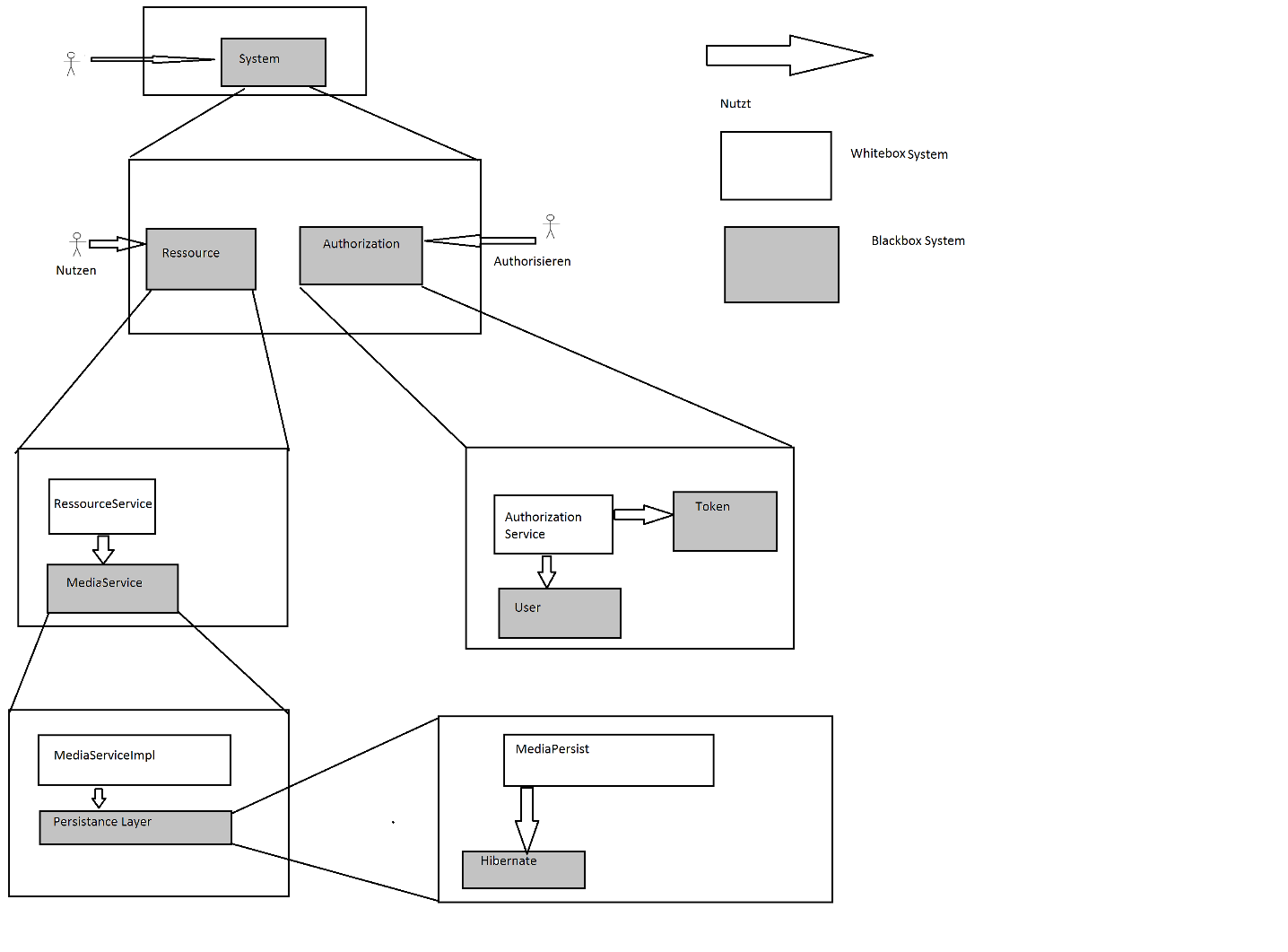
Nutzung der Microservices mit Schichtenarchitektur.

Datenhaltung mit Hibernate.

|  |  |
| --- | --- |
| Aufgabe/Problem | Lösungsansatz |
| Skalierfähigkeit/Modularität | Mircoservices |
| Robustheit | Prüfung der Eingabe |
| Schnittstelle | REST Schnittstelle mit JSON |

|  |  |  |
| --- | --- | --- |
| Qualitätsmerkmal | Szenario | Maßnahmen |
| Robustheit | Fehlerhafte Implementirerung | Junit Tests |
|  |  |  |

# Bausteinsicht



# Laufzeitsicht

1. Anmelden

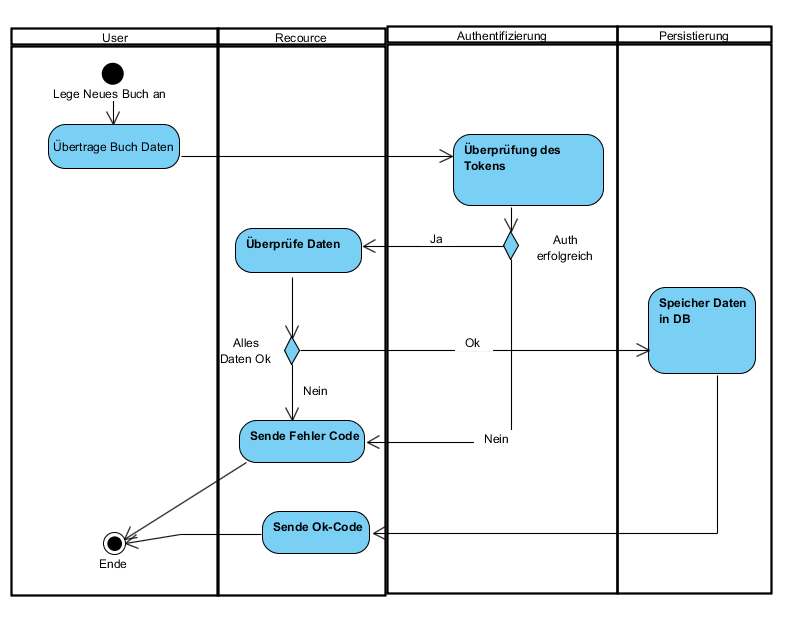
2. Bücher anzeigen lassen

3. Buch anlegen

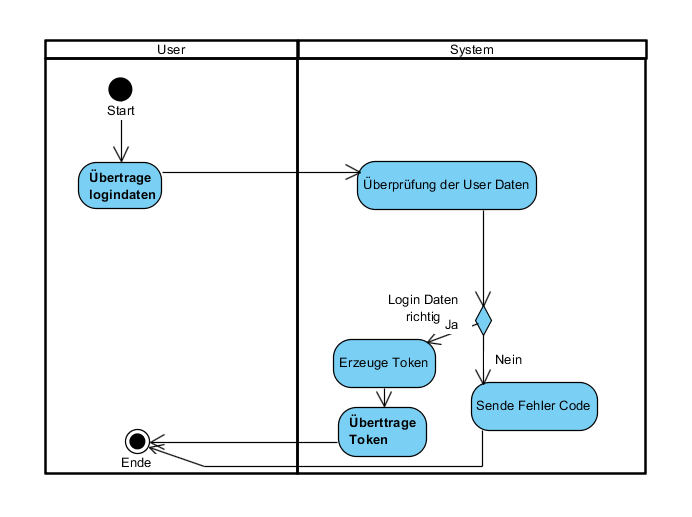
4. Buch speziell suchen

5. Ausloggen

## *Buch anlegen*



## *Login*



# Verteilungssicht

Sämtliche module warden auf Heroku deployed.

<https://www.logicline.de/blog/2015/11/heroku_technical_view/>

# Querschnittliche Konzepte

1. Logging mittels Log42j
2. Schichtenarchitektur
3. Microservice-Ansatz
4. Persistierung mit Hibernate
5. Deployment auf Heroku
6. Dependency injection mittels Guice

# Entwurfsentscheidungen

Entscheidungsmöglichkeiten sind im [Anforderungsdokument] festgelegt.

# Qualitätsanforderungen

Beispiel:

|  |  |
| --- | --- |
| Robustheit | System sollte bei fehlerhafter Eingabe weiter funktionieren |

1. Anwendungszenario:

User gibt Isbn falsch ein.

Die Anfrage wird auf Richtigkeit überprüft bevor sie vom System weiterverarbeitet wird.

User bekommt eine Ausagekräftige Fehlermeldung die klarstellt wo das Problem lag.

User kann danach normal weiter mit dem System arbeiten.

1. Änderungszenario:

Verleihen von anderen Medien.

-> Identifizierbarkeit schwieriger

-> Richtigkeit nicht definiert

# Risiken und technische Schulden

1. **Authentification Service** nicht als Microservice implementiert.
2. **Token Klasse beinhaltet zu viel Funktionalität** – Tokenklasse aufsplitten in handling und Token
3. **User steht derzeit in dem Authorisation Package**

–ungünstig bei der Buch zu Besitzer Zuordnung.

1. **Bisher keine JUnits die über HTTP**, die die REST Schnittstelle testen - bisher nur händisch mithilfe Postman
2. **Persitierung der User fehlt**
3. **Es können keine Exemplare angelegt oder verliehen werden**

# Glossar

|  |  |
| --- | --- |
| **Begriff** | **Definition** |
| Medium | Oberbegriff für Elemente die von ShareIt verliehen werden können |
| Microservice | Microservices sind voneinander unabhängige Dienste |
| Heroku | Platform auf der ShareIt gehostet wird |
| Book | Eindeutiges Buch(über die ISBN) zu dem es mehrere Copy‘s geben kann |
| Disc | Eindeutige Disc(über den Barcode) zu dem es mehrere Copy‘s geben kann |
| Copy | Eine ausleihbare Einheit eines bestimmten Types |
|  |  |