Microservice Weblog – Spezifikation

# Übersicht

Im Rahmen des Workshops ist es eure Aufgabe, das Back-end für ein einfaches Weblog-System zu entwickeln – als Microservice basierend auf Spring Boot. Der Service soll vier einfache Operationen unterstützen:

* Abruf einer Liste aller Weblog-Artikel, absteigend nach Datum sortiert.
* Abruf eines einzelnen Artikels, identifiziert über ein lesbares Identifikationsmerkmal
* Publikation eines neuen Artikels inklusive einfacher Validierung
* Aktualisieren bestimmter Eigenschaften eines bestehenden Artikels

Hinweis: Die in der Spezifikation beschriebenen „Sicherheitsmaßnahmen“ dienen ausschließlich dazu, die erforderliche Geschäftslogik komplizierter zu gestalten und sind in keiner Weise geeignet, das System vor Missbrauch zu schützen.

# Spezifikation

## Use Case 1 – Einen einzelnen Artikel abrufen

1. Wenn der Client einen bestehenden Artikel anhand seiner ID abruft, sollen folgende Informationen in der Antwort enthalten sein: ID des Artikels, Titel, Inhalt, voller Name des Autors, Email-Adresse des Autors, Datum der ersten Publikation.
2. Wenn der Client einen bestehenden Artikel abruft, dürfen folgende Informationen nicht in der Antwort enthalten sein: Evtl. vorhandene technische Primärschlüssel, Login-Name des Autors.
3. Falls der Client bei der Anfrage eine nicht vorhandene ID angibt, so muss das System die Anfrage mit einer Fehlermeldung zurückweisen.

## Use Case 2 – Alle Artikel abrufen

1. Wenn der Client alle Artikel abruft, dann soll das System zu jedem publizierten Artikel nur folgende Informationen zurückliefern: Titel, Datum der ersten Publikation, URL zum Abruf des Artikels selbst.
2. Wenn der Client alle Artikel abruft, dann soll das System alle publizierten Artikel zurückliefern – absteigend sortiert nach dem Zeitstempel der Veröffentlichung.
3. Falls das System noch keine publizierten Artikel gespeichert hat, so wird ein leeres Ergebnis zurückgeliefert, aber keine Fehlermeldung produziert.

## Use Case 3 – Einen neuen Artikel publizieren

1. Wenn der Client einen neuen Artikel publizieren möchte, dann muss das System sicherstellen, dass die Anfrage die Informationen den Login-Namen des Autors sowie den Titel und den Inhalt des neuen Artikels enthält.
2. Falls die Anfrage zur Erstellung eines neuen Artikel einen Login-Namen enthält, der dem System nicht bekannt ist, dann muss das System die Anfrage mit einer Fehlermeldung zurückweisen.
3. Das System soll aus dem Titel eines neuen Artikels automatisch dessen Identifikationsmerkmal erzeugen. Dabei gelten folgende Regeln:
   1. Alle Buchstaben werden in Kleinbuchstaben umgewandelt.
   2. Umlaute und 'ß' werden ersetzt: ä 🡪 ae, ö 🡪 oe, ü 🡪 ue, ß 🡪 ss
   3. Alle Sonderzeichen werden durch ein '-' ersetzt. Folgen mehrere Sonderzeichen aufeinander, so werden diese durch ein einzelnes '-' ersetzt.
   4. Alle Sonderzeichen am Ende des Titels werden gestrichen.
4. Falls aus dem Titel ein bereits bestehendes Identifikationsmerkmal erzeugt wird, dann soll das System die Anfrage mit einer Fehlermeldung zurückweisen.
5. Wenn ein neuer Artikel erfolgreich publiziert wird, dann soll die zentrale Redaktion darüber informiert werden. In der ersten Ausbaustufe genügt die tägliche Bereitstellung einer Log-Datei, die ausschließlich redaktionelle Ereignisse festhält.

## Use Case 4 – Einen existierenden Artikel ändern

1. Wenn der Client einen bestehenden Artikel ändern möchte, dann muss das System sicherstellen, dass die Anfrage die Informationen den Login-Namen des Autors sowie den Titel und den Inhalt enthält.
2. Falls die Anfrage zur Erstellung eines neuen Artikel einen Login-Namen enthält, der dem System nicht bekannt ist, dann muss das System die Anfrage mit einer Fehlermeldung zurückweisen.
3. Falls der in der Anfrage enthaltene Login-Name nicht mit dem des Originalautors übereinstimmt, so muss das System die Anfrage mit einer Fehlermeldung zurückweisen.
4. Falls der Client bei der Anfrage eine nicht vorhandene ID angibt, so muss das System die Anfrage mit einer Fehlermeldung zurückweisen..
5. Falls der in der Anfrage enthaltene Titel nicht mit dem Original übereinstimmt, dann soll das System zwar den geänderten Titel speichern, darf aber nicht die ID des Artikels verändern.
6. Wenn ein bestehender Artikel erfolgreich geändert wird, dann soll die zentrale Redaktion darüber informiert werden (s.o.).

# REST API

## Einen einzelnen Artikel abrufen

### Methode

GET /articles/{articleId}

### Parameter

*articleId*: Eindeutiges Identifikationsmerkmal des Artikels

### Ergebnis

Ein JSON-Objekt mit folgenden Eigenschaften:

|  |  |
| --- | --- |
| **Property** | **Beschreibung** |
| *articleId* | Eindeutiges Identifikationsmerkmal des Artikels (Format s.o.) |
| *title* | Titel des Artikels |
| *content* | Inhalt des Artikels |
| *author* | Voller Name des Autors (Vorname Nachname) |
| *emailAddress* | Korrespondenzadresse des Autors |
| *created* | Zeitstempel der ersten Publikation (YYYY-MM-DD HH:mm:ss) |

### HTTP Response Codes

|  |  |
| --- | --- |
| **Code** | **Beschreibung** |
| *200* | Artikel gefunden |
| *404* | ID ist nicht bekannt |

### Beispiel

#### Request

GET /articles/**is-tdd-dead**

#### Response

## { "articleId":"is-tdd-dead", "title":"Is TDD Dead?", "content":"David Heinemeier Hansson, the creator of Ruby on Rails, …", "author": "Martin Fauler", "emailAddress": "martin\_f@gmx.de", "created":"2015-03-30 17:27:14" }

## Alle Artikel abrufen

### Methode

GET /articles

### Ergebnis

Eine Liste von JSON-Objekten mit folgenden Eigenschaften:

|  |  |
| --- | --- |
| **Property** | **Beschreibung** |
| *title* | Titel des Artikels |
| *created* | Zeitstempel der ersten Publikation (YYYY-MM-DD, HH:mm:ss) |
| *link* | URL des Artikels |

### HTTP Response Codes

|  |  |
| --- | --- |
| **Code** | **Beschreibung** |
| *200* | Erfolgreich ausgeführt |

### Beispiel

#### Request

GET /articles

#### Response

[  
 {  
 "title": "Is TDD dead?",  
 "created": "2015-03-30 17:27:14",  
 "link": "http://localhost:8080/articles/is-tdd-dead"  
 },  
 {  
 "title": "Goto Fail, Heartbleed, and Unit Testing Culture",  
 "created": "2015-03-29 16:12:37",  
 "link": "http://localhost:8080/articles/goto-fail-heartbleed-and-unit-testing-culture"  
 },  
 {  
 "title": "The Dependency Injection Pattern",  
 "created": "2015-02-19 23:05:57",  
 "link": "http://localhost:8080/articles/the-dependency-injection-pattern"  
 }  
]

## Einen neuen Artikel publizieren

### Methode

POST /articles

### Request Body

Ein JSON-Objekt mit folgenden Eigenschaften:

|  |  |
| --- | --- |
| **Property** | **Beschreibung** |
| *nickName* | Eindeutiges Identifikationsmerkmal des Artikels (Format s.o.) |
| *title* | Titel des Artikels |
| *content* | Inhalt des Artikels |

### Ergebnis

Bei erfolgreicher Ausführung enthält der Location-Header der http-Antwort die URL des neuen Artikels

### HTTP Response Codes

|  |  |
| --- | --- |
| **Code** | **Beschreibung** |
| *201* | Artikel wurde erfolgreich gespeichert |
| *403* | Artikel wurde nicht angelegt, da Autor unbekannt |
| *409* | Artikel wurde nicht angelegt, da bereits ein anderer Artikel mit gleicher ID existiert |

### Beispiel

#### Request

POST /articles

{  
 "nickName": "martin",  
 "title": "Is TDD dead?",  
 "content":"David Heinemeier Hansson, the creator of Ruby on Rails, **…"**}

#### Response

Location: http://<HOST>:<PORT>/articles/is-tdd-dead

## Einen existierenden Artikel ändern

### Methode

PUT /articles/{articleId}

### Parameter

*entryId*: Eindeutiges Identifikationsmerkmal des Artikels

### Request Body

Ein JSON-Objekt mit folgenden Eigenschaften:

|  |  |
| --- | --- |
| **Property** | **Beschreibung** |
| *nickName* | Eindeutiges Identifikationsmerkmal des Artikels (Format s.o.) |
| *title* | Neuer Titel des Artikels |
| *content* | Neuer Inhalt des Artikels |

### Ergebnis

Ein JSON-Objekt mit folgenden Eigenschaften:

|  |  |
| --- | --- |
| **Property** | **Beschreibung** |
| *entryId* | Unverändertes Identifikationsmerkmal des Artikels (Format s.o.) |
| *title* | Neuer Titel des Artikels |
| *content* | Neuer Inhalt des Artikels |
| *author* | Voller Name des Autors (Vorname Nachname) |
| *emailAddress* | Korrespondenzadresse des Autors |
| *created* | Zeitstempel der ersten Publikation (YYYY-MM-DD, HH:mm:ss) |

### HTTP Response Codes

|  |  |
| --- | --- |
| **Code** | **Beschreibung** |
| *200* | Artikel wurde erfolgreich gespeichert |
| *403* | Autor unbekannt oder nicht der Autor des Originals |
| *404* | ID des Artikels existiert nicht |

### Beispiel

#### Request

PUT /articles/**is-tdd-dead**

{  
 "nickName": "martin",  
 "title": "TDD considered harmful?",  
 "content": "For a number of years I have been familiar with …**"**}

#### Response

### { "articleId":"is-tdd-dead", "title": "TDD considered harmful?", "content": "For a number of years I have been familiar with …", "author": "Martin Fauler", "emailAddress": "martin\_f@gmx.de", "created":"2015-03-30 17:27:14" }

# Datenbankschema

Die Anwendung soll auf einem bestehenden Datenbankschema aufgebaut werden.

