# Dokumentation des Projektsteils von Datenbankenteam

# Die Struktur der Dokumentation

1. Anleitung
2. Architektur
3. Methoden

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

# *Anleitung*

Das Datenbankenteam beschäftigt sich damit, die geeignete Struktur für Datenbanken zu erschaffen und, wenn notwendig, zu pflegen. Dafür sind ein paar lebenswichtige für Datenbankenpflege Komponenten zu installieren und einzustellen.

Fürs Projekt sind folgende Komponenten unbedingt zu installieren:
- Apache Maven ab Version 3.5.0 (empfehlenswert ab 3.5.2)
- ElasticSearch (die Version 6.0 ist zu empfehlen)

Im Projekt wurde IntelliJ Idea mit allen dafür benötigten Komponenten (wie JDK von Oracle z.B.) benutzt.

Falls es wichtig ist das Terminal zu benutzen, wobei man nur Windows-Betriebssystem hat, dann kann zusätzlich cygwin installiert werden. Kurze Erklärung, wieso überhaupt das Terminal benutzt werden kann: im Terminal ist es leichter schnell Datenbanken mit Kommando „CURL“ zu überprüfen/verändern. Leider unterstützt die Eingabeaufforderung von Windows BS dieses Kommando nicht.

Wenn man mit der Installation von notwendigen Komponenten fertig ist, dann können die Einstellungen auf der Seite von Datenbankenteam in GitHub benutzt werden.

Vor allem ist ElasticSearch (5.6.3) auf folgende Weise einzustellen:
1 - In Rootverzeichnis von ElasticSearch findet man ein Verzeichnis „config“ und geht rein
2 - elasticsearch.yml lässt man voreingestellt
3 - in jvm.options soll man das Folgende hinzufügen:
   -Xms7g (unter Xms und Xmx)
   -Xmx7g
4 - ElasticSearch starte über bin mit dem Kommando ./elasticsearch in Terminal

Am Anfang ist es grundgesetzlich alles, was man zum Arbeiten in Datenbanken von dem Projekt braucht.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

## *Architektur*

Das Datenbankenschema sieht folgendermaßen aus:

- Index: semesterprojekt

- Type: utils
- Schema:
  - ResourceName

- Type: dokument
- Schema:
  - Abstract - String
  - Annotations - String[]
  - Author - String[]
  - Identifyer - String[]  (Synonyme)
  - Journal - String[]
  - Keywords - String[]
  - Link - String
  - MeshHeadings - String[]
  - PMID - Integer
  - PublikationsType - String
  - Substances - String[]
  - Suggest - String
  - TextminingVersion – String
  - Title - String

**Kurze Beschreibung**:

Ein Index ist die Sammlung von Dokumenten, die die ähnlichen Merkmale haben. Der Name vom Index ist mit lowercase-Buchstaben. In einem Cluster kann man so viele Indizes definieren, wie man will. ( https://www.elastic.co/guide/en/elasticsearch/reference/current/_basic_concepts.html ) Das Index, das im Projekt benutzt wird, heißt Articles.

Es gibt im Index „Articles“ zwei Typen, und zwar „Article“ und „FullText“. Ein Typ ist das logische Teil eines Index. Jedes Dokument muss einem Typ angewiesen werden.

Das Schema der Typen hat gewisse Merkmale. Die Namen der Merkmale entsprechen dem Inhalt von diesen Merkmalen.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

## *Pakete, Klassen und Methoden*

Die Klasse **ServerApp**:
1. Methode main
2. Methode start

Die Klasse „ServerApp“ dient dem Ausführen von Netty.

Die Klasse **ClientImportService:**
1. Methode parseImport

Die Klasse **ClientQueryService**:
1. Methode parseQuery

Die Klasse **TestClient**:
Hier testet man die Methoden, die von dem Client ausgeführt/aufgerufen werden.

Die Klasse **ImportService**:
1. Methode parseImport
2. Methode createDocument
3. Methode createUtils

Die Klasse **QueryService**:
1. Methode parseQuery
2. Methode getVersion
3. Methode getDocumentByID
4. Methode getUtils
5. Methode getDocumentByTitle
6. Methode deleteDocumentByID
7. Methode deleteDocumentByTitle

Die Klasse **TestService**
Hier testet man die Methoden, die den Service ausführen/benutzen.

Die Klasse **TestApplication**

Die Klasse **App**
1. Methode searchDocumentByTitle
2. Methode searchDocumentByPMID

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

3. Methode deleteDocumentByPMID
4. Methode deleteDocumentByTitle
5. Methode searchInES
6. Methode searchByMeshTerms
7. Methode searchByMoreLikeThis
8. Methode searchByRelevanceList
9. Methode searchDocumentMLT
10. Methode getCommonTermSearch
11. Methode createDataWithBulk
12. Methode createType

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

public static void main(String[] args)

Mit dieser „main“-Methode startet man Netty.
Parameter:

1. String[] args. Der Parameter wird nicht benutzt.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

public static void start(ResteasyDeployment deployment)

„start“-Methode wird in „main“-Methode der Klasse „ServerApp“ benutzt, um den Dienst von Netty aufrufen.
Parameter:

1. ResteasyDeployment deployment. Der Parameter ist voreingestellt.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

public Response parseImport(@QueryParam("type") String type, String document)

parseImport-Methode fügt ein Dokument entweder dem Typ „Document“ oder dem Typ „utils“ hinzu.
Parameter:

1. String type. Der Parameter dient dazu, um den Typ festzustellen, auszuwählen und dort ein Dokument zu speichern.
2. String document. Das Dokument wird unter dem Namen „document“ gespeichert.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

public Response createDocument(String document)

Die Methode speichert ein Dokument in dem Typ „document“.
Parameter:

1. String document. Der Name vom zu speichernden Dokument ist „document“.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

public Response createUtils(String document)

Die Methode speichert ein Dokument in dem Typ „utils“.
Parameter:

1. String document. Der Name vom zu speichernden Dokument ist „document“.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

```
public Response parseQuery(@QueryParam("type") String type,
                           @QueryParam("utilsName") String utilsname,
                           @QueryParam("PMID") @DefaultValue("-1") int pMID,
                           @QueryParam("docTitle") String title)
```

Die Methode gibt eine Antwort auf die Anfrage zurück, falls die Anfrage nicht fehlerhaft ist.
Parameter:

1. String type. Mit diesem Parameter wird entschieden, welcher Typ benutzt wird.
2. String utilsname. Dieser Parameter wird nicht benutzt.
3. int pMID. PMID ist der Schlüssel zu Dokumenten. Also, die Dokumente werden entweder nach dem Schlüssel oder nach dem Titel gesucht. Der voreingestellte Wert ist gleich -1.
4. String title. Alternative Möglichkeit etwas zu finden (oder eventuell nicht) ist die Suche nach dem Titel von dem Dokument. Darum wird auch der Titel als Parameter angefordert.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

public Response getVersion(@QueryParam("utilsname") String utilsname)

Die Methode gibt die Version von dem Typ „utils“ zurück.
Parameter:

1. String utilsname. „utilsname“ wird benutzt, um den Typ zu finden und seine Version als Antwort auf die Anfrage zu bekommen.

## Die Methode kann eventuell fehlerhaft sein!!!

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

public Response getDocumentByID(int pMID)

Die Methode gibt das angeforderte Dokument zurück, falls das existiert.
Parameter:

1. int pMID. PMID ist der Schlüssel eines Dokumentes.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

public Response getUtils(String utilsname)
Die Methode gibt ein Dokument vom Typ „utils“ mit dem Namen „utilsname“ zurück.
Parameter:

1. String utilsname. „utilsname“ ist der Name von dem zu suchenden Dokument.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

public Response getDocumentByTitle(String title)

Die Methode ist noch zu implementieren. Anfänglich soll diese Methode ein Dokument mit dem Titel „title“ zurückgeben.
Parameter:

1. String title. „title“ ist der Name des zu suchenden Dokuments.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

public Response deleteDocumentByID(int pMID)

Die Methode löscht ein Dokument mit dem Schlüssel „pMID“.
Parameter:

1. int pMID. PMID ist der Schlüssel eines Dokumentes.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

public Response deleteDocumentByTitle(String title)

Die Methode löscht ein Dokument mit dem Titel „title“.
Parameter:

1. String title. „title“ ist der Name des zu löschenden Dokuments.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

private static JsonObject searchDocumentByTitle(TransportClient client, String title)

Die Methode gibt das Dokument mit dem Titel „title“ zurück, falls das existiert.
Parameter:

1. TransportClient client. Der Transportclient stellt die Verbindung zu Elasticsearch-Cluster mit Hilfe von Transportmodulen her. ( https://www.elastic.co/guide/en/elasticsearch/client/java-api/current/transport-client.html )
2. String title. „title“ ist der Name des zu löschenden Dokuments.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

private static JsonObject searchDocumentByPMID(TransportClient client, int PMID)

Die Methode gibt das Dokument mit dem Schlüssel „PMID“ zurück, falls das existiert.
Parameter:

1. TransportClient client. Der Transportclient stellt die Verbindung zu Elasticsearch-Cluster mit Hilfe von Transportmodulen her. ( https://www.elastic.co/guide/en/elasticsearch/client/java-api/current/transport-client.html )
2. int PMID. PMID ist der Schlüssel eines Dokumentes.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

private static void deleteDocumentByPMID(TransportClient client, int pMID)

Die Methode löscht ein Dokument mit dem Schlüssel „pMID“, wobei ein Transportclient benutzt wird.
Parameter:

1. TransportClient client. Der Transportclient stellt die Verbindung zu Elasticsearch-Cluster mit Hilfe von Transportmodulen her. ( https://www.elastic.co/guide/en/elasticsearch/client/java-api/current/transport-client.html )
2. int PMID. PMID ist der Schlüssel eines Dokumentes.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

private static void deleteDocumentByTitle(TransportClient client, String title)

Die Methode löscht ein Dokument mit dem Titel „title“, wobei ein Transportclient benutzt wird.
Parameter:

1. TransportClient client. Der Transportclient stellt die Verbindung zu Elasticsearch-Cluster mit Hilfe von Transportmodulen her. ( https://www.elastic.co/guide/en/elasticsearch/client/java-api/current/transport-client.html )
2. String title. „title“ ist der Name des zu löschenden Dokuments.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

private static ArrayList<JsonObject> searchInES(TransportClient client, int pMID)

Die Methode benutzt MeshTerms und den Suchalgorithmus (Query) MoreLikeThis von Elasticsearch, um die ähnlichen zu vorgegebenem Artikel zu finden. MeshTerms sind bereits vorgegeben und werden benutzt, um die gewissen Artikel zu finden, die die wichtigen Begriffe enthalten. searchInEs gibt entweder ein Array von Artikeln oder den leeren Zeiger (null Pointer) zurück.
Parameter:

1. TransportClient client. Der Transportclient stellt die Verbindung zu Elasticsearch-Cluster mit Hilfe von Transportmodulen her. ( https://www.elastic.co/guide/en/elasticsearch/client/java-api/current/transport-client.html )
2. int PMID. PMID ist der Schlüssel eines Dokumentes.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

private static ArrayList<String> searchByMeshTerms(TransportClient client, JSONObject obj)

Die Methode such nach Artikeln, die mit den MeshTerms in dem Feld „MeshHeadings“ in gewissem Prozentsatz (z.B., 30%) übereinstimmen. searchByMeshTerms gibt entweder die Liste von Artikeln oder den leeren Zeiger (null Pointer) zurück.
Parameter:

1. TransportClient client. Der Transportclient stellt die Verbindung zu Elasticsearch-Cluster mit Hilfe von Transportmodulen her. ( https://www.elastic.co/guide/en/elasticsearch/client/java-api/current/transport-client.html )
2. JSONObject obj. obj ist der in JSON-Format codierte Artikel mit notwendigen MeshTerms. MeshTerms von obj werden für die Suche nach ähnlichen Artikeln benutzt.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

private static ArrayList<String> searchByMoreLikeThis(TransportClient client, JSONObject obj,
ArrayList<String> ids)

Die Methode benutzt MoreLikeThisQuery von Elasticsearch. Hier spielen Übereinstimmungen von Feldern mit dem Namen „Abstract“ wichtige Rolle. Zuerst werden Begriffe aus „Abstract“ von obj herausgezogen und dann mit den Zusammenfassungen von anderen Artikeln mit Hilfe von MoreLikeThisQuery verglichen. Letztendlich gibt diese Methode entweder die Liste von IDs, gespeichert als String, oder null-Zeiger/Pointer.
Parameter:

1. TransportClient client. Der Transportclient stellt die Verbindung zu Elasticsearch-Cluster mit Hilfe von Transportmodulen her. ( https://www.elastic.co/guide/en/elasticsearch/client/java-api/current/transport-client.html )
2. JSONObject obj. obj ist der in JSON-Format codierte Artikel, dessen Information zum Suchen und zum Vergleichen angewandt wird.
3. ArrayList<String> ids. In dieser List sind die Dokumenten gespeichert, die zum Vergleichen dienen.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

```
private static ArrayList<JsonObject> searchByRelevanceList(TransportClient client,
                                                           ArrayList<String> ids)
```

# Die Methode ist noch zu implementieren!!!

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

private static ArrayList<JsonObject> searchDocumentMLT(TransportClient client, int pMID)

Die Methode ist eine andere Variante von searchByMoreLikeThis. Zuerst wird ein Dokument mit dem Schlüssel „pMID“ gefunden und dann mit dem Feld „AbstractContent“ von anderen Artikeln verglichen. Zurückgegeben wird entweder die Liste von JSON-Objekts, oder null-Zeiger.
Parameter:

1. TransportClient client. Der Transportclient stellt die Verbindung zu Elasticsearch-Cluster mit Hilfe von Transportmodulen her. ( https://www.elastic.co/guide/en/elasticsearch/client/java-api/current/transport-client.html )
2. int PMID. PMID ist der Schlüssel eines Dokumentes.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

private static JsonObject getCommonTermSearch(TransportClient client, int pMID)

## Die Methode ist noch zu implementieren!!!

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

private static void createDataWithBulk(BulkProcessor bulkRequest, JSONObject jsonObject)

## Die Methode ist noch zu implementieren!!!

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden

private static void createType()
private static void createType(String index, String type)

Die Methode ist überladen.

Die erste voreingestellte Methode createType, die keine Argumente in Signatur hat, erschafft einen neuen Index, falls der Index noch nicht existiert. Sonst wird ein Exception ausgelöst und aufgefangen.
Parameter: keine

Die zweite Signatur besagt, dass man zusätzlich den Namen von Index und den Namen von Typ eingeben kann. Falls Index noch nicht existiert, dann wird der mit dem Typ „type“ erstellt. Falls Index bereits existiert, dann wird der Typ „type“ eventuell erschafft oder verändert abhängig davon, ob der Typ nicht existiert oder schon erstellt. Das Schema von Typ ist voreingestellt und, um das zu verändern, muss man selbst den Code anpassen.
Parameter:

1. String index. Der Index mit dem Namen „index“ wird erstellt bzw. benutzt.
2. String type. Der Typ mit dem Namen „type“ wird erstellt bzw. verändert. Das Schema vom Typ ist voreingestellt.

zurück zur Startseite
zurück zur Anleitung
zurück zur Architektur
zurück zur Methoden