

Dokumentenanalyse mit ElasticSearch

Von Stefan Beigel und Kevin Edinger





Backend

- NodeJS Webserver
 - Schnittstelle mit ExpressJS
- ElasticSearch
 - ElasticSearch Server
 - ElasticSearch JavaScript Client
 - Attachment-Plugin für ElasticSearch
 - Benutzt Apache Tika für die Text-Extrahierung

Frontend

- AngularJS
 - Controller
 - Services
- Bootstrap



REST API

Collections

GET	/documents	//all documents
GET	/documents?page={page}	//10 documents per page
GET	/documents?search={searchtext}&page={page}	

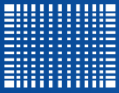
Single

GET	/documents/{id}	//document information
GET	/documents/{id}/file	//download file
DELETE	/documents/{id}	
POST	/documents	//Document als JSON im Body



Probleme

- Reihenfolge der Suchergebnisse
- Dateiübertragung
 - Wie soll eine Datei übertragen werden?



Fine-Tuning der Suche

- Score ist für die Sortierung zuständig
- Der Score wird aus drei Teilen Berechnet
 - Häufigkeit des Suchbegriffs im Dokument (Term frequency)
 - Textlänge des Dokuments (Field-length norm)
 - Häufigkeit des Suchbegriffs in anderen Dokumenten auf diesem Shard vor (Inverse document frequency)



Fine-Tuning der Suche

Field-length norm führt zu falschen Suchergebnissen:

- Deaktivieren der norms

Inverse document frequency führt zu falschen Suchergebnissen:

- Index auf ein Shard begrenzen
- Globale Inverse document frequency aktivieren



Dateiübertragung

- Base64 Kodierung der Datei auf der Client Seite
- POST der Datei mittel JSON
- Dekodieren der Datei auf der Serverseite

Live Demo

