

# Agiler Publikationsprozess

Praktikum Werkzeuge für Agile Modellierung

Anna Ostrovskaya | 24. Juli 2019

FAKULTÄT FÜR INFORMATIK / INSTITUT FÜR PROGRAMMSTRUKTUREN UND DATENORGANISATION



- 1 Motivation
- 2 Grundlagen einer webbasierten Anwendung
- 3 Umsetzung
- 4 Live-Demo
- 5 Fazit

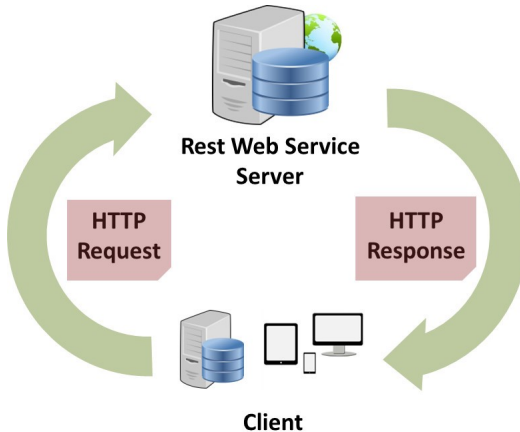
- Forschungsprozesse mit Reviews sind oft schwergewichtig, mit langen Iterationen, intransparent.



- Webbasierte agile Lösung für transparenten Publikationsprozess von wissenschaftliche Papiere

- Vereinfachter Registrierung/Login-Prozess
- Hochladen von Artikeln
- Versionierung von Artikeln
- Personalisierte Kommentare
- Bewertungssystem für Artikeln
- Unterstützung von Tagging

# Grundlagen einer webbasierten Anwendung



# Wie funktioniert es?

Name	x	Headers	Preview	Response	Timing
<input type="checkbox"/> tags/ <input type="checkbox"/> documents/		General			
		Request URL:	https://agilpub-backend.annaos.dev/tags/		
		Request Method:	GET		
		Status Code:	200		
		Remote Address:	18.208.41.204:443		
		Referrer Policy:	no-referrer-when-downgrade		
		Response Headers (7)			
		Request Headers			
		Provisional headers are shown			
		Accept:	application/json, text/plain, */*		
		Origin:	https://agilpub.annaos.dev		
		Referer:	https://agilpub.annaos.dev/documents		
		User-Agent:	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit		

Name	x	Headers	Preview	Response	Timing
<input type="checkbox"/> tags/ <input type="checkbox"/> documents/				<pre>[[{"id": 6, "name": "SUM", "..."}, {"id": 11, "name": "Projectional", "..."}, {"id": 16, "name": "Signatures", "..."}], ...]</pre>	
				<pre>0: {"id": 6, "name": "SUM", "..."}   documents: [{"id": 4, "name": "Single Underlying Models for Projectional, Multi-View Environments", "score": 1,...}]     0: {"id": 4, "name": "Single Underlying Models for Projectional, Multi-View Environments", "score": 1,...}       createdAt: "2019-07-17T09:02:11.963+0000"       id: 4       name: "Single Underlying Models for Projectional, Multi-View Environments"       score: 1     id: 6     name: "SUM"     1: {"id": 11, "name": "Projectional", "..."}     2: {"id": 16, "name": "Signatures", "..."}     3: {"id": 17, "name": "Kryptographie", "..."}]</pre>	

# Wie funktioniert es?

Name	×	Headers	Preview	Response	Timing
<input type="checkbox"/> tags/					
<input type="checkbox"/> documents/					
	▼	[,...]			
	▼	0:	{id: 4, name: "Single Underlying Models for Projectional, Multi-View Environments", score: -0.6666667,...} createdDate: "2019-07-17T09:02:11.963+0000" id: 4 name: "Single Underlying Models for Projectional, Multi-View Environments" ▶ owner: {id: 3, name: "Anna", username: "anna"} score: -0.6666667 ▶ tags: [{id: 11, name: "Projectional"}, {id: 6, name: "SUM"}] ▼ versions: [...] ▼ 0: {id: 5, createdDate: "2019-07-17T09:02:11.963+0000", version: 1, filename: "eeq9fbnfs8b4nl63307579"} createdDate: "2019-07-17T09:02:11.963+0000" filename: "eeq9fbnfs8b4nl63307579" id: 5 version: 1 ▶ 1: {id: 7, createdDate: "2019-07-17T09:02:45.753+0000", version: 2, filename: "rq9ehl2coecihibzf01zfcf"} ▶ 1: {id: 14, name: "Homomorphic Signature", score: 3, createdDate: "2019-07-17T18:49:51.034+0000",...}		

- Backend
  - Java Spring Boot Rest Service
  - In-memory H2 Datenbank (6 Tabellen)
  - Lombok
- Frontend
  - Angular (10 Komponenten)
  - ng2-file-upload
  - ng2-pdf-viewer PDF Darstellung
  - Rangy (Bibliothek für Textselektion)



- Übergabe von verknüpften Daten: *JsonIgnore* und *JsonIgnoreProperties*

```
10  @Entity
11  @Table
12  @Data
13  public class Document {
14
15      @Id
16      @GeneratedValue(strategy = GenerationType.AUTO)
17      private long id;
18      private String name;
19      private float score;
20      private final Date createdAt;
21
22      @ManyToOne(fetch=FetchType.EAGER)
23      @JoinColumn(name="OWNER_ID")
24      @JsonIgnoreProperties({"files", "comments", "scores"})
25      private final User owner;
26
27      @ManyToOne(fetch = FetchType.LAZY)
28      @JoinTable(
29          name="DOCUMENT_TAG",
30          joinColumns=@JoinColumn(name="DOCUMENT_ID"),
31          inverseJoinColumns=@JoinColumn(name="TAG_ID"))
32      @JsonIgnoreProperties("documents")
33      private Set<Tag> tags = new HashSet<>();
34
35      @OneToMany(mappedBy = "document", fetch=FetchType.LAZY)
36      @JsonIgnoreProperties({"document", "comments"})
37      private List<DocumentVersion> versions;
38
39      @OneToMany(mappedBy = "document")
40      @JsonIgnore
41      private List<Score> scores;
```

Name	×	Headers	Preview	Response	Timing
<input type="checkbox"/> tags/					
<input type="checkbox"/> documents/					
	▼	[,...]			
	▼	0:	{id: 4, name: "Single Underlying Models for Projectional, Multi-View Environments", score: -0.6666667,...} createdDate: "2019-07-17T09:02:11.963+0000" id: 4 name: "Single Underlying Models for Projectional, Multi-View Environments" ▶ owner: {id: 3, name: "Anna", username: "anna" score: -0.6666667 ▶ tags: [{id: 11, name: "Projectional"}, {id: 6, name: "SUM"}] ▼ versions: [...] ▼ 0: {id: 5, createdDate: "2019-07-17T09:02:11.963+0000", version: 1, filename: "eeq9fbnfs8b4nl63307579" createdDate: "2019-07-17T09:02:11.963+0000" filename: "eeq9fbnfs8b4nl63307579" id: 5 version: 1 ▶ 1: {id: 7, createdDate: "2019-07-17T09:02:45.753+0000", version: 2, filename: "rq9ehl2coecihibzf01zfcf" ▶ 1: {id: 14, name: "Homomorphic Signature", score: 3, createdDate: "2019-07-17T18:49:51.034+0000",...}		

## ■ Unifizierung von Datumsdarstellung

```
<td>{{ version.createdDate | date:'dd.MM.yyyy \ 'um\ ' HH:mm:ss' }}</td>  
<td>{{ version.createdDate | createdDateFormat }}</td>
```

```
@Pipe({  
  name: 'createdDateFormat',  
})  
export class createdDateFormatPipe implements PipeTransform {  
  transform(value: Date) {  
    registerLocaleData(localeDe);  
    let datePipe = new DatePipe( locale: "de-DE");  
    return datePipe.transform(value, format: 'dd.MM.yyyy \ 'um\ ' HH:mm:ss');  
  }  
}
```

- Übergabe PDF-Datei per REST-Service
  - Backend: FileSystemStorageService
    - erhält MultipartFile, Speicherung im Dateisystem
    - Übertragung per HTTP-Response als Content Type application/pdf
  - Frontend
    - Upload
    - Empfang per HTTP-Response als Content Type application/pdf

- Speichern von Kommentare  
mit **Rangy**: cross-browser JavaScript range and selection library

```
14/1/3:62,14/1/3:66 { f910c7cc }  
start      ,end      {checksum }
```

```
14/1/3      :62  
node_path : index
```



## DONE

- REST-Service Backend
- Single-Site-Webanwendung mit Basisfunktionalität

## TODO

- Datenbank-Anbindung
- Login- / Registrierungsprozess
- Backend-Autorisierungsprozess
- Performance
- Dateigröße beschränken
- Überlappende Kommentare ermöglichen