

Implementierungsbericht

Team 2

SEP WS 2021/22



Betreuer:

PROF. DR. CHRISTIAN BACHMAIER

Projektphase	Leiter
Pflichtenheft	Johann Schicho
Entwurf	Stefanie Gürster
Feinspezifikation	Johannes Garstenauer
Implementierung	Thomas Kirz
Validierung	Sebastian Vogt

14. Januar 2021

Inhaltsverzeichnis

1	Tatsächlicher Aufwand	1
1.1	Allgemeines	1
1.2	Milestone 1	1
1.2.1	Geplante Arbeitspakete	1
1.2.2	Zusätzlicher Aufwand	3
1.2.3	Stundenanzahl	4
1.3	Milestone 2	4
1.3.1	Geplante Arbeitspakete	4
1.3.2	Zusätzlicher Aufwand	5
1.3.3	Stundenanzahl	5
1.4	Milestone 3	6
1.4.1	Geplante Arbeitspakete	6
1.4.2	Zusätzlicher Aufwand	7
1.4.3	Stundenanzahl	7
1.5	Stundenanzahl aller Milestones	7
2	Schwierigkeiten	8
2.1	Größere Abweichungen in der geschätzten Zeit	8
2.1.1	Arbeitspaket NewSubmission	8
2.1.2	Arbeitspaket Pagination & SortSearchColumn	8
2.1.3	Arbeitspaket Homepage	8
2.1.4	Arbeitspaket Submission ohne Gutachten	8
2.1.5	Arbeitspaket Profile	8
2.1.6	Fazit	9
2.2	Direkte Probleme bei der Programmierung	9
2.2.1	JSF	9
2.2.2	JDBC	9
2.2.3	Fatale Datenbankfehler	10
2.2.4	Testing	10
3	Code Metrics	11
4	Änderungen gegenüber Spezifikation	11
4.1	Klassen	11
4.1.1	business.internal	12
4.1.2	business.service	12
4.1.3	business.util	12
4.1.4	control.backing	12
4.1.5	control.conversion	14
4.1.6	control.internal	14
4.1.7	persistence.util	14
4.1.8	persistence.repository	14
4.1.9	global.transport	15
4.2	Funktionalitäten und Leistungen	16
4.3	Datenbankschema	16
4.4	Facelets	17
4.4.1	Composite Components	17

4.5	Views	18
4.6	Bibliotheken	18

1 Tatsächlicher Aufwand

Sebastian Vogt

1.1 Allgemeines

Zusätzlich zu den geplanten Arbeitspaketen sind in der Implementierung auch ungeplante Aufgaben aufgekommen. Diese sind in einer extra Tabelle aufgelistet:

- **Code Review:** Pull Requests von anderen Teammitgliedern reviewen.
- **Testing:** Hiermit ist weniger das Schreiben der Tests selbst und mehr das lauffähig machen von Tests aus dem Implementierungsplan sowie das Integrieren von CDI in die Tests gemeint.
- **Allgemeines Debugging:** Diese Debugging Arbeiten lassen sich nicht eindeutig einem Arbeitspaket zuordnen, da sie sich auf das System als ganzes beziehen.

Bei Arbeitspaketen ohne tatsächliche Zeit handelt es sich um Pakete, bei denen es schon Prototypen gab, die entgegen der Erwartung 1:1 übernommen werden konnten.

1.2 Milestone 1

1.2.1 Geplante Arbeitspakete

Tabelle 1: Milestone 1

Arbeitspaket	Start	Ende	Aufwand	Tatsächlicher Start	Tatsächliches Ende	Tatsächlicher Aufwand	Bearbeiter:in
NewSubmission	6.12.10 10:00 Uhr	8.12.2021 16:00 Uhr	10 h	06.12.2021 11:30 Uhr	10.12.2021 03:00 Uhr	23,7 h	Sebastian Vogt
Crypto	6.12.2021 17:00 Uhr	7.12.2021 14:00 Uhr	3 h	07.12.2021 21:50 Uhr	07.12.2021 22:30 Uhr	0,7 h	Sebastian Vogt

Fortsetzung auf der nächsten Seite

Tabelle 1: Milestone 1 (Fortsetzung)

Arbeitspaket	Start	Ende	Aufwand	Tatsächlicher Start	Tatsächliches Ende	Tatsächlicher Aufwand	Bearbeiter:in
Connection Pool	9.12.2021 8:00 Uhr	9.12.2021 19:00 Uhr	6 h	09.12.2021 08:00 Uhr	09.12. 19:00 Uhr	7,0 h	Sebastian Vogt
SQL Datenschema	6.12.2021 10:00 Uhr	6.12.2021 11:00 Uhr	0,5 h	06.12.2021 11:00 Uhr	08.12.2021 01:15	2,25 h	Johann Schicho
Pagination	6.12.2021 11:00 Uhr	6.12.2021 15:00 Uhr	2 h	-	-	0 h	Johann Schicho
Template: SortSearchColumn	6.12.2021 15:00 uhr	6.12.2021 16:00 Uhr	1 h	-	-	0 h	Johann Schicho
Welcome	6.12.2021 16:00 Uhr	7.12.2021 14:00 Uhr	5 h	06.12.2021 14:00 Uhr	08.12.2021 14:00 Uhr	7 h	Johann Schicho
LifeTime	7.12.2021 14:00Uhr	8.12.2021 15:00 Uhr	7,0 h	07.12.2021 13:00 Uhr	07.12.2021 14:30	1,5 h	Johann Schicho
UserGetLists	8.12.2021 15:00 Uhr	9.12.2021 16:00Uhr	5 h	07.12.2021 15:30 Uhr	08.12.2021 21:15 Uhr	6,25 h	Johann Schicho
LifeTime: Logger	6.12.2021 10:00 Uhr	6.12.2021 14:00 Uhr	3 h	06.12.2021 11:00 Uhr	06.12.2021 14:00 Uhr	1 h	Thomas Kirz
Homepage	6.12.2021 14:00 Uhr	9.12.2021 19:00 Uhr	13 h	06.12.2021 15:00 Uhr	09.12.2021 23:00 Uhr	21 h	Thomas Kirz
Message Bundle Producer	6.12.2021 17:00 Uhr	7.12.2021 10:00 Uhr	2 h	06.12.2021 14:00 Uhr	06.12.2021 15:00 Uhr	1 h	Thomas Kirz
Exceptions	7.12.2021 10:00 Uhr	7.10.2021 11:00 Uhr	0,5 h	Thomas Kirz			
Template: Main	6.12.2021 10:00 Uhr	7.12.2021 16:00 Uhr	2 h	06.12.2021 10:45 Uhr	07.12.2021 15:00 Uhr	4,5 h	Stefanie Gürster

Fortsetzung auf der nächsten Seite

Tabelle 1: Milestone 1 (Fortsetzung)

Arbeitspaket	Start	Ende	Aufwand	Tatsächlicher Start	Tatsächliches Ende	Tatsächlicher Aufwand	Bearbeiter:in
Submission ohne Gutachten	6.12.2021 16:00 Uhr	9.12.2021 18:00 Uhr	17 h	06.12.2021 14:15 Uhr	09.12.2021 03:30 Uhr	25 h	Stefanie Gürster
SessionInformation	6.12.2021 10:00 Uhr	6.12.2021 14:00 Uhr	2 h	06.12.2021 10:50 Uhr	07.12.2021 15:20 Uhr	9,17 h	Johannes Garstenauer
TrespassListener	6.12.2021 14:00 Uhr	6.12.2021 19:00 Uhr	4 h	07.12.2021 15:30 Uhr	07.12.2021 18:30 Uhr	3 h	Johannes Garstenauer
UIMessage Generator	7.12.2021 10:00Uhr	7.12.2021 17:00 Uhr	5 h	-	-	0 h	Johannes Garstenauer
Templates: Header, Footer	7.12.2021 17:00 Uhr	8.12.2021 16:00 Uhr	4 h	08.12.2021 13:00 Uhr	09.12.2021 20:00 Uhr	13,5 h	Johannes Garstenauer
Config Reader	8.12.2021 16:00 Uhr	9.12.2021 14:00 Uhr	3 h	07.12.2021 18:40 Uhr	07.12.2021 19:00	0,33 h	Johannes Garstenauer

3

1.2.2 Zusätzlicher Aufwand

Aufgabe	Sebastian Vogt	Johann Schicho	Thomas Kirz	Stefanie Gürster	Johannes Garstenauer
Code Review	1,8 h	1,5 h	1 h	1 h	1 h
Testing	-	3,25 h	1 h	1 h	-
Allgemeines Debugging	-	2,75 h	-	-	2 h

1.2.3 Stundenanzahl

Teammitglied	Stundenanzahl Geplant	Stundenanzahl Tatsächlich	
Sebastian Vogt	19 h	33,2 h	33
Johann Schicho	20,5 h	22 h	28
Thomas Kirz	18,5 h	25 h	27
Stefanie Gürster	19 h	31,5 h	30 } ?
Johannes Garstenauer	18 h	32 h	27 } ?
	Σ	Σ	

vs h-Tag

1.3 Milestone 2

1.3.1 Geplante Arbeitspakete

Tabelle 2: Milestone 2

Arbeitspaket	Start	Ende	Aufwand	Tatsächlicher Start	Tatsächliches Ende	Tatsächlicher Aufwand	Bearbeiter:in
ScienceFields	13.12.2021 10:00	13.12.2021 20:00	5 h	13.12.2021 16:00 Uhr	14.12.2021 14:00 Uhr	4,5 h	Sebastian Vogt
Profile	14.12.2021 08:00	16.12.2021 11:30	12 h	14.12.2021 14:00 Uhr	17.12.2021 02:15 Uhr	18 h	Sebastian Vogt
NewReview	13.12.2021 14:00	13.12.2021 18:00	4 h	14.12.2021 19:00 Uhr	15.12.2021 19:30 Uhr	4 h	Johann Schicho
Submission: Reviews	13.12.2021 18:00	15.12.2021 15:00	10 h	13.12.2021 13:00 Uhr	15.12.2021 21:30 Uhr	14 h	Johann Schicho
NewScientificForum	16.12.2021 10:00	16.12.2021 15:30	4 h	16.12.2021 13:30 Uhr	17.12.2021 00:30 Uhr	7,5 h	Johann Schicho
Registration	13.12.2021 14:00	14.12.2021 17:00	10 h	13.12.2021 13:00 Uhr	15.12.2021 11:00 Uhr	11 h	Thomas Kirz

Fortsetzung auf der nächsten Seite

Tabelle 2: Milestone 2 (Fortsetzung)

Arbeitspaket	Start	Ende	Aufwand	Tatsächlicher Start	Tatsächliches Ende	Tatsächlicher Aufwand	Bearbeiter:in
Verification	15.12.2021 13:00	16.12.2021 17:00	10 h	14.12.2021 15:00 Uhr	16.12.2021 19:00 Uhr	13 h	Thomas Kirz
Administration	13.12.2021 10:00	14.12.2021 14:00	6 h	15.12.2021 21:40 Uhr	16.12.2021 23:30 Uhr	9,5 h	Stefanie Gürster
Toolbar	14.12.2021 15:00	16.12.2021 19:00	10 h	13.12.2021 17:00 Uhr	17.12.2021 02:00 Uhr	13 h	Stefanie Gürster
ScientificForum	13.12.2021 16:00	15.12.2021 18:00	15 h	13.12.2021 13:00 Uhr	16.12.2021 19:45 Uhr	22,5 h	Johannes Garstenauer
ScientificForumList	16.12.2021 13:00	16.12.2021 18:00	5 h	16.12.2021 21:00 Uhr	17.12.2021 02:30 Uhr	5,5 h	Johannes Garstenauer

51

1.3.2 Zusätzlicher Aufwand

Aufgabe	Sebastian Vogt	Johann Schicho	Thomas Kirz	Stefanie Gürster	Johannes Garstenauer
Code Review	-	-	1 h	1 h	-
Testing	4,6 h	2,5 h	1 h	2,5 h	-
Allgemeines Debugging	3,4 h	1 h	4 h	1 h	-

1.3.3 Stundenanzahl

Teammitglied	Stundenanzahl Geplant	Stundenanzahl Tatsächlich
Sebastian Vogt	17 h	30,5 h
Johann Schicho	18 h	29 h
Thomas Kirz	20 h	30 h
Stefanie Gürster	16 h	27 h
Johannes Garstenauer	20 h	28 h

h. p. g.
30
26
30
26
27

1.4 Milestone 3

1.4.1 Geplante Arbeitspakete

Tabelle 3: Milestone 3

Arbeitspaket	Start	Ende	Aufwand	Tatsächlicher Start	Tatsächliches Ende	Tatsächlicher Aufwand	Bearbeiter:in
ResultList	03.01.2022 10 Uhr	06.01.2022 14 Uhr	15 h	20.12.2021 13:30 Uhr	06.01.2022 12:00	6,9 h	Sebastian Vogt
PeriodicWorker	03.01.2022 10 Uhr	04.01.2022 18 Uhr	4 h	21.12.2021 13:30 Uhr	21.12.2021 15:30 Uhr	2 h	Johann Schicho
InitialConfig	05.01.2022 10 Uhr	06.01.2022 14 Uhr	4 h	21.12.2021 15:30 Uhr	23.12.2021 17:15 Uhr	7,5 h	Johann Schicho
ErrorPage	03.01.2022 10 Uhr	03.01.2022 18 Uhr	3 h	20.12.2021 11:00 Uhr	20.12.2021 18:00 Uhr	3,5 h	Thomas Kirz
E-Mail	04.01.2022 12 Uhr	05.01.2022 14 Uhr	5 h	21.12.2021 13:00 Uhr	06.01.2022 16:00 Uhr	8 h	Thomas Kirz
NewUser	05.01.2022 14 Uhr	06.01.2022 14 Uhr	5 h	22.12.2021 11:00 Uhr	04.01.2022 13:00 Uhr	4 h	Thomas Kirz
Imprint	03.01.2022 10 Uhr	03.01.2022 18 Uhr	3 h	21.12.2021 15:10 Uhr	21.12.2021 16:10 Uhr	1 h	Stefanie Gürster
CSS Themes	04.01.2022 10 Uhr	06.01.2022 14 Uhr	10 h	20.12.2021 14:16 Uhr	04.01.2022 17:00 Uhr	12 h	Stefanie Gürster
UncheckedExceptionHandler	03.01.2022 10 Uhr	04.01.2022 18 Uhr	7 h	04.01.2022 11:00 Uhr	04.01.2022 17:00 Uhr	4,25 h	Johannes Garstenauer
UserList	05.01.2022 10 Uhr	06.01.2022 14 Uhr	4 h	18.12.2021 15:00 Uhr	06.01.2022 17:00 Uhr	13,5 h	Johannes Garstenauer

1.4.2 Zusätzlicher Aufwand

In diesem Meilenstein musste zusätzlich noch die Versendung der E-Mails in bestehenden Service Code eingearbeitet werden.

Aufgabe	Sebastian Vogt	Johann Schicho	Thomas Kirz	Stefanie Gürster	Johannes Garstenauer
Code Review	-	-	1,5 h	-	-
Testing	-	-	0,5 h	-	-
Allgemeines Debugging	8,1 h	12,5 h	5 h	5,75 h	7,5 h
E-Mail Versendung	4,4 h	-	-	4,3 h	-

1.4.3 Stundenanzahl

Teammitglied	Stundenanzahl Geplant	Stundenanzahl Tatsächlich
Sebastian Vogt	15 h	19,4 h
Johann Schicho	8 h	22 h
Thomas Kirz	13 h	22,5 h
Stefanie Gürster	13 h	23,05 h
Johannes Garstenauer	11 h	25,25 h

15
22
23
23
25

Team meeting
Zählt
auch...

1.5 Stundenanzahl aller Milestones

Teammitglied	Stundenanzahl Geplant	Stundenanzahl Tatsächlich
Sebastian Vogt	51 h	83,1 h
Johann Schicho	46,5 h	73 h
Thomas Kirz	51,5 h	77,5 h
Stefanie Gürster	48 h	81,55 h
Johannes Garstenauer	49 h	85,25 h

Σ
~ 700 h 1.75

Σ

2 Schwierigkeiten

Johann Schicho

Wie bereits aus Kapitel 1 hervorgeht, gab es teilweise größere Abweichungen von den geplanten Zeiten. Diese werden hier nochmal aufgefasst und genauer beschrieben. Ziel ist es zu reflektieren und die Gründe für größere Verzögerungen zu erörtern. Zusätzlich wird auch auf Probleme in der Programmierung selbst eingegangen.

2.1 Größere Abweichungen in der geschätzten Zeit

2.1.1 Arbeitspaket NewSubmission

Dieses Arbeitspaket benötigte 13 Stunden mehr als ursprünglich geplant. *NewSubmission* war Teil des ersten Milestones. Dadurch war viel Code noch nicht vorhanden. Da wir alle Arbeitspakete vertikal implementiert haben, das heißt zu jedem Frontend-Code wurde der zugehörige Backend-Code von der gleichen Person implementiert, war es sehr viel Arbeit für einen Einzelnen. Zusätzlich benötigte dieses Arbeitspakete auch viele unterschiedliche *Services* und damit verschob sich der Zeitaufwand immens.

2.1.2 Arbeitspaket Pagination & SortSearchColumn *gleichzeitige, oder?*

Diese Arbeitspakete benötigten während der Implementierungsphase keine weitere Zeit. Grund dafür war, dass bereits Prototypen in den Wochen davor programmiert worden sind. Damit war die Arbeit bereits abgeschlossen und konnte direkt übernommen werden.

2.1.3 Arbeitspaket Homepage *Spezialist?*

Hier wurden acht Stunden mehr benötigt, als ursprünglich angenommen. Grund dafür war ähnlich wie bei *NewSubmission*. Die Übersichtsseite sammelt Informationen aus vielen Ecken des Systems und stellt diese dar, dafür musste viel Code geschrieben werden. Außerdem wurde hier zum ersten mal die Paginierung implementiert, was viele Rückfragen im Team und dadurch extra Zeit benötigte. Im Rahmen des Implementierungszeitraums wurde durch die gesammelte Erfahrung das Implementieren dieser einfacher. Den Aufwand, besonders während des ersten Milestones, haben wir hier unterschätzt.

2.1.4 Arbeitspaket Submission ohne Gutachten

Wie der Name schon impliziert, wurde hier bereits ein größeres Arbeitspaket aufgespalten. Die Seite einer Einreichung ist das Kernstück der Anwendung und umfasst damit viel Funktionalität. Der große Aufwand war uns bewusst, dennoch wurden zehn Stunden mehr benötigt als geplant. Grund hier war neben den bereits genannten Gründen natürlich auch Startschwierigkeiten mit *Jakarta Server Faces*. Der Projektumfang ist wesentlich angestiegen und somit auch die Anforderungen an den Programmierer, sich tiefer mit der Technologie auseinander zu setzen.

2.1.5 Arbeitspaket Profile

Der zusätzliche Zeitaufwand kam hier durch die Notwendigkeit eines *Servlets* hinzu, welches die Profilbilder zur Verfügung stellt. Das war ein Exkurs von der bisherigen

Arbeit mit JSF und beanspruchte mehr Zeit als angenommen.

man sieht
was ein JSF abnimmt (1)

2.1.6 Fazit

Der Zeitaufwand für Arbeitspakete, bei denen bereits viel Zeit eingeschätzt war, hat sich häufig um viel mehr Zeit verlängert, als bei Arbeitspaketen, bei welchen die Zeit von Beginn an geringer eingeschätzt war. Eine weitere Aufteilung der Arbeitspakete hätte das gegebenenfalls verhindern können, hätte aber auch gegenseitige Abhängigkeiten während der Implementierung erzeugt. Das hätte zu anderweitigen Verzögerungen geführt.

Abweichungen von zwei bis drei Stunden waren abzusehen und sind auch bei den meisten Arbeitspaketen aufgetreten, sowohl als Verlängerung als auch Verkürzung der tatsächlichen Arbeitszeit. Häufiger wurde allerdings mehr Zeit benötigt als ursprünglich angenommen. Die großen Ausreißer in der benötigten Zeit gab es nach Milestone 1 nicht mehr. Grund dafür ist sicherlich die Vertrautheit mit der Systemarchitektur und der bereits vorhandenen Code Infrastruktur.

2.2 Direkte Probleme bei der Programmierung

2.2.1 JSF

Jakarta Server Faces stellte häufig größere Herausforderungen. Besonders bei der Verwendung von *Custom Components* wurde oftmals mehr Zeit benötigt. Dazu gehört unter anderem die *Pagination*. Wobei der Prototyp bereits bestand und lauffähig war, müssen im Einsatz mehrere Objekte durch die `@PostConstruct init()` Methode und zusätzlich einer `viewAction onLoad()` Methode initialisiert werden. Die Komplexität davon wurde im Voraus nicht richtig eingeschätzt und führte damit zu erheblichen Verlängerungen in Arbeitspaketen, in welchen diese verwendet wurde.

ja, eine der komplexesten Stellen

2.2.2 JDBC

Die Verbindung zur Datenbank wurde nach Voraussetzung mit *Java Database Connectivity* umgesetzt. Die Technologie war im geringen Maße bereits bekannt und konnte sehr schnell eingesetzt werden. Probleme entstanden durch benötigte Aufrufe von *Close Methoden*. Diese werden benötigt um sowohl *Connections* an den *Connection Pool* zurückzugeben, als auch *SQL Statements* zu schließen.

Anfangs führte das zu vielen *Leaks* an Datenbankverbindungen, die nicht mehr ordnungsgemäß zurückgegeben wurden. Gelöst wurde das Problem durch besseres *Logging* der Datenbankverbindungen und Verwendung von *Try-with-Resource* Codeblöcken. Das benötigte zusätzliche Zeit zu debuggen und implementieren und hat die Arbeitszeit bei vielen Arbeitspaketen gestreckt.

ja, leider

Ein weiteres Problem war das *Error-Handling* von SQL Errors. *PostgreSQL* und *JDBC* geben hier nur numerische Fehlercodes zurück und keine direkte Information, ob es sich um eine *Checked-* oder *Unchecked Exception* handelt. Hier wurde ein Prototyp von Sebastian Vogt entwickelt, allerdings zuerst nur vereinzelt von ihm umgesetzt. Es benötigte anschließend einen gemeinsamen Einsatz aller Teammitglieder alle Datenbankzugriffsmethoden mit der neuen Fehlerbehandlung auszustatten.

JDBC benötigt die manuelle Formulierung von *SQL Statements*. Für die Implementierung der Paginierung mit den zusätzlich filterbaren Spalten ist eine dynamische Generierung des *SQL Statements* nötig. Dadurch wurde der Code viel länger und komplizierter als ursprünglich angenommen. Dadurch wurde in vielen Arbeitspaketen die Zeit rein durch das Schreiben und *Refactoren* der *SQL Statements* gestreckt.

2.2.3 Fatale Datenbankfehler

Ein Nutzer wird auf eine Fehlerseite geleitet, falls auf eine nicht existierende Seite navigiert wird oder ein interner Error in LasEs auftritt. Ein schwerwiegender Fehler kann dabei der Abbruch der Verbindung zum Datenbankserver sein. Tritt ein solcher Fehler auf können keine Daten aus der Datenbank geladen werden und es wird auf die Fehlerseite weitergeleitet. Hier ergab sich das Problem eines rekursiven Fehlers.

Da diese Seite das *Templating* verwendet, wurde wiederum versucht das Firmenlogo und die Auswahl des *Styles* aus der Datenbank zu laden. Hier trat wiederum ein Datenbankfehler auf und es wurde wieder auf die Fehlerseite weitergeleitet, auf welcher sich der Prozess wiederholte.

Gelöst wurde das Problem durch die Verwendung von *Default* Werten, falls die Datenbankverbindung auf der Fehlerseite nicht erfolgreich ist. ✓

2.2.4 Testing

Viele Testfälle basieren auf Mocking mit der Softwarebibliothek *Mockito*. Hier entstanden Zeitverzögerungen dadurch, dass alle Entwickler vorher keinen bis wenigen Kontakt zu diesem Framework hatten. Damit streckte sich der Zeitaufwand in die Länge.

Ein weiteres Problem waren "*Phantom Bugs*". *Mockito* injiziert zum Mocking *Byte Code* direkt in den auszuführenden Code. Das hat zur Folge, dass in Testfunktionen in welchen *Mockito* nur lokal verwendet wird, auch in anderen Funktionen der gleichen Klasse zur Ausführung von gemockten Code führt.

Das führte zu langen Suchen nach Bugs, in denen sogar im Debug Modus anderer Code ausgeführt wurde. Anschließend gelöst wurde das Problem durch Auftrennen des Codes in unterschiedliche Testklassen, mit oder ohne *Mockito*.

Die Abhängigkeit von *CDI* in Tests stellte ein weiteres Problem dar. Grundsätzlich wird es im Produktiv Code durch den *Application Server* zur Verfügung gestellt. Wurde nun aber versucht ein von *CDI* abhängige Methode in einem Test aufzurufen, sind diese immer fehlgeschlagen.

Durch die Verwendung einer weiteren Softwarebibliothek¹, wird das Testen von *CDI* Komponenten vereinfacht und konnte damit auch umgesetzt werden.

¹<https://mvnrepository.com/artifact/org.jboss.weld/weld-junit-parent>

Tabelle position
hier
unverändert

Tabelle 4: Lines of Code

Java (Main)			
<i>Gesamt</i>	<i>Code</i>	<i>Kommentare</i>	<i>Leerzeilen</i>
17030	8451 (50 %)	6131 (36 %)	2372 (14 %)
Java (Test)			
<i>Gesamt</i>	<i>Code</i>	<i>Kommentare</i>	<i>Leerzeilen</i>
4066	2701 (66 %)	532 (13 %)	833 (20 %)
Facelets (XML)			
<i>Gesamt</i>	<i>Code</i>	<i>Kommentare</i>	<i>Leerzeilen</i>
3300	2860	139	301
CSS			
<i>Gesamt</i>		<i>Allgemein</i>	<i>Themes</i>
868		78	790
Properties			
<i>Konfiguration</i>		<i>Resource Bundles</i>	
71		1082	
SQL-Schema			
174			



3 Code Metrics

Thomas Kirz

Die Metriken in Tabelle 4 wurden mit dem IntelliJ-IDEA-Plugin *Statistic* zur Zeit der vorläufigen Abgabe von LasEs generiert.

4 Änderungen gegenüber Spezifikation

4.1 Klassen

Stefanie Gürster

Veränderungen bezüglich der Parameterübergabe von Methoden treten im Allgemeinen nur bedingt auf. Ein Beispiel dafür wäre, dass in der Methode *addReviewer()* in der Klasse *SubmissionService* kein *Submission-DTO* mehr mit übergeben wird, da lediglich die Id der Einreichung zum Abspeichern in der Datenbank benötigt wird. Diese Id wird aber bereits mit einem *ReviewedBy-DTO* übergeben. Die Submission wäre also hier überflüssig.

Damit ein Weiterleiten auf die nächste Seite einer *Pagination* funktioniert, müssen alle zugehörigen Einträge gezählt werden. Dies übernehmen Methoden im Service sowie in den Repositories mit der Namenskonvention *count...()* o.ä.. Eine solche Methode wird zum Beispiel im *PaperService* verwendet, um alle Paper einer Submission zu zählen.

71
Zahl

gut: rel. detailliert

Um Redundanzen im Folgenden zu vermeiden, wird diese Methode im Weiteren nicht mehr angesprochen.

4.1.1 business.internal

Lifetime In der Methode *startup()* wird nun zusätzlich ein FileDTO, welches *logger properties* als *InputStream* enthält, als Parameter übergeben.

PeriodicWorker Um den PeriodicWorker zu implementieren wird ein Timer Objekt verwendet. Dieser Timer wird durch *init()* gestartet und durch *stop()* gestoppt.

4.1.2 business.service

PaperService Für die Methode *getLatest()* wird nur ein Submission-DTO benötigt und kein User-DTO wie zuvor angenommen.

Die Methode *remove()* wurde entfernt, da sie nicht benötigt wird.

ScienceFieldService Die Funktion, ein einzelnes Fachgebiet mit *get()* zu laden wurde nicht benötigt.

SubmissionService In *add()* werden zusätzlich ein Paper-DTO, sowie ein File-DTO übergeben. Im Gegensatz dazu wird eine Liste von Gutachter nicht mehr benötigt.

Der Rückgabewert der Methode *change()* wurde von *void* zu *boolean* verändert, um Rückmeldung der Änderung zu erhalten.

Außerdem wird in der Toolbar eine Liste von Gutachtern der jeweiligen Einreichung geladen. Daher wurde die Methode *getList(Submission submission)* neu hinzugefügt.

Die Methode *releaseReview()* wird bereits im *ReviewService* durch *change()* implementiert und ist somit hier überflüssig.

Ebenso wird die Methode *addCoAuthor()* nicht mehr verwendet, da Co-Autoren nur bei der Einreichung selbst hinzugefügt werden. Die Funktionalität liegt also schon in der Methode *add()*.

UserService Um den Avatar eines Nutzers zu löschen, wird *setAvatar()* verwendet. Daher ist *deleteAvatar()* überflüssig.

Die Methode *getVerification()*, welche bestimmt, ob eine E-Mail-Adresse schon verifiziert ist, wird nicht mehr benötigt, da die Verifikation bereits über *isVerified()* aus dem User DTO abgefragt, oder über *verify()* eine E-Mail-Adresse verifiziert werden kann.

CustomizationService Um alle angebotenen Farbthemen in einem Dropdown darzustellen, werden die Namen der Stylesheets mit Hilfe von *loadStyles()* geladen.

4.1.3 business.util

EmailUtil Aufgrund von Benutzerfreundlichkeit werden in einigen E-Mails auch passend generierte Links mit versendet. Dafür wurden folgende Methoden neu hinzugefügt: *generateLinkForEmail*, *generateSubmissionURL*, *generateForumURL*.

4.1.4 control.backing

Diesem Paket wurden drei weitere Interfaces hinzugefügt: *ScientificForumPaginationBacking*, *SubmissionPaginationBacking* und *UserPaginationBacking*. Der Grund dafür war die

Erweiterung der *Templates* um drei *Composite Components*. Die einzelnen *Paginations* werden an mehreren Orten verwendet. Um Codeduplikationen zu vermeiden und für die jeweilige Komponente die benötigten Methoden zur Verfügung zu stellen, wurden die genannten Interfaces implementiert. Die folgenden Klassen implementieren ein oder mehrere Interfaces: *ResultListBacking*, *UserListBacking* und *ScientificForumList*.

Im Weiteren folgt aus dem Verwenden der oben genannten Interfaces, dass Methoden wie *getDateSelect()* (also alle Methoden, die zur Filterung dienen (Dropdown)) nur bedingt in den einzelnen *Backing Beans* implementiert werden.

War auf einer Seite keine *Pagination* vorgesehen und es war nötig jedoch die übergebenen *Viewparameter* überprüfen oder die Zugangsrechte eines Benutzers, so wurde in manchen Fällen noch eine *onLoad()* oder eine *preRenderViewListener()* Methode hinzugefügt.

AdministrationBacking Um die Funktionalität zu erweitern, ein neues Logo hochladen zu können, wurde zusätzlich *uploadNewLogo()* hinzugefügt. Damit das momentan ausgewählte Stylesheet geladen wird und dem Benutzer angezeigt werden kann, wurde *getPathToStyle()* hinzugefügt.

HomepageBacking Der Name eines zugehörigen Forums zu einer Einreichung wird über *getForumName()* geladen, damit dieser in der *Pagination* angezeigt werden kann. Um einzelne Tabs per CSS dynamisch zu aktivieren, wird dies über *get...CssClassSuffix()* gesetzt. Dieselbe Funktionalität mit verschiedenen Reitern wurde auch beim *ResultlistBacking*, sowie dem *ScientificForumBacking* benötigt, daher wurde die Klasse dementsprechend erweitert.

NavigationBacking Hier wurde die Methode *init()* noch hinzugefügt.

NewScientificForumBacking Damit ein neues Fachgebiet hinzugefügt werden kann, wurde die Methode *createNewScienceField()* implementiert. Wurde ein neues Fachgebiet erstellt, erscheint es in einer Art Liste. Hier kann der Benutzer ein Gebiet auswählen und hinzufügen, indem er jenes nicht in ein Feld eingibt, sondern anklickt. Dementsprechenden wurden auch die Methoden im *Backing Bean* angepasst.

NewSubmissionBacking Sollte ein Eingabefehler seitens des Benutzers erfolgen, so wird *initNewSubmission()* aufgerufen, um den zuvor erhaltenen *Viewparameter* nun händisch setzen zu können.

Zudem reicht hier eine Methode zum Laden einer Liste von Editoren aus, da die Auswahl über ein *selectOneMenu* erfolgt.

ProfileBacking Damit Adminrechte an Nutzer vergeben werden können, wurden *Getter* und *Setter* für den jeweiligen betroffenen Nutzer hinzugefügt.

ScientificForumBacking Editoren, sowie Fachgebiete werden in einer Liste wiedergegeben, welche durch ein Eingabefeld ergänzt werden können. Hierzu wurden entsprechende Methoden hinzugefügt.

Zusätzlich wurde die Forumsseite um Informationen wie der Deadline eines Forums erweitert.

was sende mal }

Die Information, ob ein Nutzer Gutachter ist, wird hier nicht benötigt, sondern lediglich ob dieser Admin oder Editor ist.

SubmissionBacking Je nach Status einer Einreichung sollte diese entsprechend mit Hil-

fe von `styleSubmissionState()` farblich markiert werden.
`disableReviewUploadButton` steuert die Möglichkeit zum Hochladen eines Gutachtens und wurde neu hinzugefügt.

Außerdem wird mit `loggedInUserHasPendingReviewerRequest` geprüft ob ein angefragter Gutachter bereits der Rolle zugestimmt hat.

Alle Methoden, welche die Filterauswahl bereitstellen, wurden hier, falls noch nicht existent, hinzugefügt, da das Backing Bean kein Interface implementiert.

Damit beim Klicken auf einen Namen oder E-Mail-Adresse über ein MailTo-Link weitergeleitet wird, wurde `sentMailTo()` implementiert.

ToolbarBacking Zur Unterstützung der Benutzerfreundlichkeit ist es möglich schon hinzugefügte Gutachter leichter zu bearbeiten, indem dessen Daten wieder in das jeweilige Eingabefeld geschrieben werden. Zusätzlich wird, abhängig von der Sprache, das Datumsformat als *Placeholder* angezeigt und generiert.

4.1.5 control.conversion

Neu ist hier ein *Converter* für ein Fachgebiet, um dieses als String darstellen oder als DTO wieder auswerten zu können.

4.1.6 control.internal

Constants Hier werden alle Längen, die zur Validierung benötigt werden, in Form von Konstanten erfasst.

Pagination Anstatt dass die Methode `applyFilter()` in den einzelnen Backing Beans implementiert wird, wurde sie nun allgemein in dieser Klasse eingefügt.

4.1.7 persistence.util

Von hier an: Johannes Garstenauer

DataSourceUtil Es ist die Methode `logSQLException()` hinzugekommen, welche die Erstellung semantisch aussagekräftiger Logs für *SQL-Exceptions* ermöglicht.

Hierbei ist auch die neue Klasse **TransientSQLExceptionChecker** zu erwähnen, welche die Methode `isTransient()` anbietet. Hierdurch ist es möglich, zu überprüfen, ob ein Vorgang, welcher eine *SQL-Exception* hervorgerufen hat, bei einem erneuten Versuch mit gleichen Parametern funktionieren könnte. Dies ermöglicht es den Entwicklern einen Nutzer darauf hinzuweisen, eine fehlgeschlagene Operation erneut zu versuchen.

4.1.8 persistence.repository

PaperRepository Es gibt keine `setPDF()`-Methode mehr. Generell wurden im gesamten Entwicklungsprozess des öfteren individuelle Setter für bestimmte Datenelemente zugunsten von allgemeineren `add()` und `change()` Methoden ersetzt. Diese enthalten dann alle Datenelemente eines zugehörigen Facelets. Ein weiteres Beispiel hierfür sind die Methoden `setPDF()` und `remove()` im **ReviewRepository**. Änderungen dieser Art wurden zugunsten übersichtlicherer Interfaces gemacht.

ConnectionPool Die neue Methode `getNumberOfFreeConnections()` wurde hinzugefügt um ein aussagekräftigeres Logging der Verwendung von Connections zu ermöglichen.

Dies fand vor allem im Finden von *Connection-Leaks* Verwendung. Hierbei wurde nun geloggt, wann, von welcher Methode und in welcher Zeile eine Connection geöffnet wurde. Außerdem wurde festgehalten wie viele offene Connections noch bestehen. Hierdurch konnte die Korrektheit des *Connection-Pool* bestätigt und *Connection-Leaks*, verursacht durch Programmierfehler, schnell behoben werden.

Es ist unter anderem manchmal vorgekommen, dass sich die Notwendigkeit bestimmter Methoden erst während der Entwicklung ergeben hat. Allerdings beschränkt sich das nur auf eine handvoll Methoden, wodurch wir die Genauigkeit der Spezifikation bestätigt sehen. Ein Beispiel hierfür ist das **ReviewedByRepository**. Die *getList()*-Methode hat sich hier erst später als notwendig erwiesen. Andererseits ist es seltener auch zu Verschiebung von Methoden zwischen Klassen gekommen. Die *removeReviewer()*-Methode im **SubmissionRepository** war hier schlicht an der falschen Stelle. Sie hätte sich schon in der Spezifikation im *ReviewedByRepository* befinden sollen. Änderungen dieser Art wurden zugunsten des *single-responsibility principle* gemacht, wonach eine Klasse nur Methoden zu einem bestimmten Zweck besitzen sollte.

Es kam an wenigen Stellen auch vor, dass Methoden schlichtweg nicht benötigt wurden. Hier ist die *get()* Methode aus dem **ScienceFieldRepository** zu nennen. Sie diente dem Erhalten eines wissenschaftlichen Themas aus der Datenbank. Ursprünglich war diese Methode in der Spezifikation enthalten, da bei der Spezifikation der Repositories generell vorgesehen war *CRUD*-Operationen:

- *CREATE*
- *READ*
- *UPDATE*
- *CHANGE*

anzubieten. Allerdings wurde ein wissenschaftliches Thema nie alleine sondern immer nur im Verbund (mithilfe der *getList()*-Methoden) benötigt. Somit lässt sich feststellen, dass der Versuch gewisse Patterns konsequent umzusetzen, in wenigen Fällen zu Unstimmigkeit mit der Spezifikation geführt hat.

UserRepository Die neuen *changeVerification()* und *getVerification()* Methoden erlauben die Nutzervalidierung. Auch wenn diese Funktionalität in der Spezifikation nicht vergessen wurde, musste hier nachgearbeitet werden. In diesem Fall wurden die zwei genannten Methoden hinzugefügt um die Funktionalität zu implementieren.

4.1.9 global.transport

In der Enumeration **Privilege** musste eine geringfügige Funktionalitätserweiterung vorgenommen werden. Es wurde die Methode *getPrivilege()* hinzugefügt, um die Umwandlung von Datenbankspalten in das Modell der Businesslogik zu ermöglichen. Der Grund hierfür war in der Spezifikation im Vorhinein schwer zu antizipieren, und kann daher als ein erwartbarer Wandel im Entwicklungsprozess gewertet werden.

Es ist ebenfalls aufgetreten, dass neue Repräsentationen benötigt wurden. **Recommendation** modelliert das Empfehlungsverhalten von Gutachtern. Die **Visibility**-Enumeration wurde hinzugefügt um die Sichtbarkeit eines Papers innerhalb einer Einreichung zu repräsentieren.

Auch das Entfernen von Repräsentationen ist aufgetreten. Dies geschah zu Gunsten einer besseren Erweiterbarkeit der Anwendung, in diesem Fall um das dynamische Hinzufügen neuer *CSS-Stylesheets* zu ermöglichen. Es wurde die **Style** Enumeration entfernt. Die Auswahl von Farbschemata des Systems basiert nun dynamisch auf den im Anwendungsverzeichnis existierenden zugehörigen CSS-Dateien.

4.2 Funktionalitäten und Leistungen

Johannes Garstenauer

Es wurde größtenteils darauf geachtet den, im Pflichtenheft definierten, Produktfunktionen treu zu bleiben. Dies ist der **LasEs**-Anwendung größtenteils gelungen. Vorgenommene Änderungen sind:

Das Weglassen bestimmter Wunschfunktionen, wie:

*sind das alle
weggelassenen?*

- /FW061/ Optional ist das Einfügen eines Avatarbildes, siehe /DW015/, bei der Registration.
- /FW062/ Weiterhin kann der Nutzer optional weitere Daten, wie in /FW240/ beschrieben, angeben.

In diesem Fall dienten die Änderungen der Übersichtlichkeit der Registrationsseite und der Verringerung des Arbeitsaufwandes in diesem Arbeitspaket.

Eine weitere nennenswerte Änderung liegt in den Produktleistungen. Hier wurden die Ansprüche an die Pagination festgelegt:

- /L042/ Listen sind filterbar. Eine Spalte kann also auf einen bestimmten Wert gefiltert werden.
- /L045/ Alle Listen sind nach ihren Spalten auf- oder absteigend sortierbar.

Diese Anforderungen beziehen sich auf alle Spalten der jeweiligen Pagination. Im Entwicklungsprozess ist klar geworden, dass das nicht im Sinne der Übersichtlichkeit und Nutzerfreundlichkeit des Systems ist. Daher sind nun nicht mehr alle Spalten einer Pagination filter- und sortierbar, sondern nur noch diejenigen wo es Sinn ergibt.

4.3 Datenbankschema

Johannes Garstenauer

Das spezifizierte Datenbankschema wurde größtenteils unverändert in die *LasEs*-Anwendung übernommen.

Die Änderungen betreffen die Ergänzung um einen Eintrag zum *Look* des Systems.

```
TABLE system:  
    css_theme VARCHAR(20)
```

Weiterhin wurde eine *View* eingeführt:

~~Wird nicht mehr benötigt~~

```
create view user_data as  
select u.id,  
       u.email_address,  
       u.firstname,  
       u.lastname,
```

```

        u.title,
        u.birthdate,
        u.employer,
        u.is_registered,
        count(distinct s.id) as count_submissions,
        (
            case
                when u.is_administrator then 'admin'
                when exists(select * from member_of mo
                    where mo.editor_id = u.id) then 'editor'
                when exists(select * from reviewed_by rb
                    where rb.reviewer_id = u.id) then 'reviewer'
                else 'none'
            end) as user_role
from "user" u
    left JOIN submission s
        on u.id = s.author_id
    left join co_authored ca
        on (u.id = ca.user_id and ca.submission_id = s.id)
group by u.id;

```

user_data bietet einige für den Entwickler nützliche Zusatzinformationen über User des Systems an, wie:

- die Gesamtanzahl von Einreichungen,
- die höchste Rolle, welche der Nutzer im System innehat.

Die *view* wurde eingeführt, nachdem diese während der Entwicklung mehrfach angefragt wurden.

Letztlich wurde das Datenbankschema so erweitert, dass beim initialen Setup ein generischer Administrator als Nutzer hinzugefügt wird.

```

INSERT INTO "user"(id, email_address, is_administrator, firstname, lastname,
    is_registered, password_hash, password_salt)
VALUES (-1, 'admin@example.com', TRUE, 'Administrator', 'Administrator', TRUE,
    'CWDkErZ4M1iW4LJjHB8kAA==', '7aj1fcaMrRpJrtS9ZNsIsQ==');

```

```

INSERT INTO verification(id, is_verified)
VALUES (-1, TRUE);


```



4.4 Facelets

Stefanie Gürster

4.4.1 Composite Components

Neben der *Pagination* wurden zusätzliche *Composite Components* erstellt. Alle Komponenten: *scientificForumTable*, *submissionTable* und *userTable* verwenden eine *Pagination* und wurden erstellt, da die einzelnen Tabellen an mehreren Stellen benötigt werden.  Beispielsweise wird eine *submissionTable* auf der Homepage, auf der Seite eines Forums und

auf der Seite, welche die Suchergebnisse anzeigt, verwendet. Ähnlich geschieht dies bei den anderen Tabellen.

Die Facelets wurden zusätzlich mit einer *content.xhtml* ergänzt, welche als eine Art *fallback* Seite fungiert. Sollten bei Änderungen der *main.xhtml* Fehler auftreten wird als default eben dieses Facelet angezeigt. 2 -

4.5 Views

Stefanie Gürster

Im Allgemeinen wurden hier nur Änderungen unter dem Motiv der Benutzerfreundlichkeit vorgenommen. Das heißt beispielsweise, dass zusätzlicher Text oder kleinere Funktionalitäten mit Hilfe von Buttons hinzugefügt worden sind. Außerdem wurde die Funktionalität von einigen *Paginations* in, wie oben beschrieben, *Composite Components* ausgelagert.

Änderungen, welche im oberen Teil zu den *Backing Beans* bereits beschrieben wurden haben Teils kleine Auswirkungen auf das jeweilige Facelet. Diese werden aber dann hier nicht weiter erläutert, um Duplikation zu vermeiden.

errorPage.xhtml Hier wurde noch ein zusätzlicher *outputLink* hinzugefügt, der den *Stack Trace* nur im *developmentMode* ausklappen kann.

scientificForum.xhtml Die Anleitung zum Erstellen eines Gutachtens wird nicht in einer *textArea* angezeigt, sondern es erscheint beim Klicken des entsprechenden Buttons ein *Dialog* mit den zugehörigen Informationen.

4.6 Bibliotheken

Stefanie Gürster

Weld JUnit Extentions Um Methoden, welche CDI benötigen, ebenfalls testen zu können, haben wir zusätzlich *Weld JUnit Extentions*² verwendet.

↳ Lizenz

vs. alle Libs von Maven?
weil die anderen nur zum testen

²<https://mvnrepository.com/artifact/org.jboss.weld/weld-junit-parent>