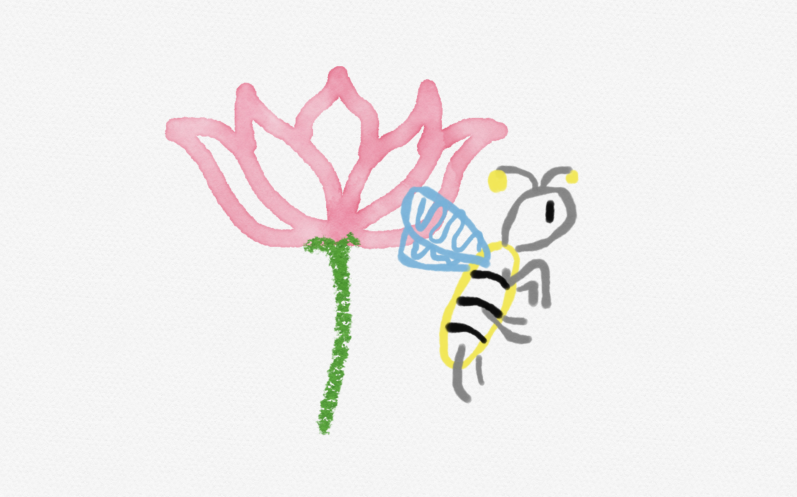
BiBi - Implementierungsbericht

Ivan Charviakou
León Liehr
Jonas Picker
Sergei Pravdin

25. Juni 2021
v1.0



Inhaltsverzeichnis

1 Einleitung

Autor: Sergei Pravdin

In diesem Dokument ist der Implementierungsbericht der Webanwendung **BiBi** dokumentiert. Dabei erfolgt die Gliederung der Implementierung in Milestones mit den geplanten und tatsächlichen Zeitaufwänden. Außerdem sind das **Änderungen gegenüber der Spezifikation**, die **Änderungen gegenüber dem Implementierungsplan** und die **Code-Metriken** zu sehen.

2 Meilensteine

Autor: Sergei Pravdin

In diesem Kapitel werden wir die geplanten Zeitaufwänden, bzw. Start- und Endzeiten, mit den tatsächlichen Werten verglichen.

Abbildung 1: Meilensteine 1 und 2

3 Änderungen gegenüber dem Implementierungsplan

Autor: Jonas Picker

Im Folgenden werden die Abweichungen vom Implementierungsplan

3.1 Milestone 1

Zu Begin des Milestones am Montag den 31.05.2021 war es zunächst notwendig, den bisherigen Implementierungsplan aufgrund von Unzulänglichkeiten komplett zu überarbeiten. Im Zuge dessen wurde der eigentliche Beginn der Implementierungsarbeit für 3 Teammitglieder deutlich verzögert. Das restliche Kapitel bezieht sich auf diese zweite, überarbeitete Version des Plans. Es war ursprünglich geplant, den Testspezialisten das Schreiben der Testpakete übernehmen zu lassen und ihn zum Ausgleich in den anderen Paketen zu entlasten. Nachdem dies wider Erwarten nicht stattfinden konnte, mussten alle Teammitglieder zusätzliche Zeit aufwenden, um sich teilweise parallel in das verwendete Testframework einzuarbeiten. Das Aufsetzen der Datenbank und der Systemstartfunktionen nahm ebenfalls deutlich mehr Zeit in Anspruch, als ursprünglich angenommen. Nachdem sich dieses Arbeitspaket ('SystemStartStop/DataLayerInitializer') auch als schwierig zu testen herausstellte, wurde der geplante Test stattdessen zunächst für den Logger erstellt. Durch die Entscheidung, die Tests nicht mit Auszuliefern und in ein eigenes Projekt zu verpacken, entstanden hier bereits die ersten Probleme beim Lesen von Dateien aus dem Hauptprojekt, jedoch wurde es aus Zeitmangel und der unerwarteten Parallelarbeit an den Tests nicht gleich kollektiv erkannt. Durch falsch kalkulierte Zeitschätzungen beim Anpassen des Implementierungsplans am Anfang der Phase, hatte der Testspezialist zum Schluss zu wenig Arbeit und es wurde bereits mit Teilen des Arbeitspakets 'Medienansicht' aus Milestone 2 begonnen. Neben Problemen bei der korrekten Fertigstellung des Arbeitspakets 'Login/Registrierung', hatte der verantwortliche Implementierer Schwierigkeiten, das Versionskontrollsystem 'git' im Einklang mit dem übrigen Team und dem Spezialisten über die Kommandozeile zu benutzen. Dadurch entstand ein deutlicher Mehraufwand für den Rest des Teams beim Reparieren unbedachter Operationen am gemeinsamen Repository.

3.2 Milestone 2

Wegen des Fehlens eines funktionierenden Logins am Ende von Milestone 1, musste der Verantwortliche dessen Fertigstellung zunächst vervollständigen, wodurch seine geplanten Arbeitspakete in diesem Milestone nicht angefangen wurden. Durch dieses Fehlen einer Grundfunktionalität, wurde das manuelle Ausprobieren von sessionabhängigen Funktionen ebenfalls erschwert. Ansonsten konnten die übrigen Teammitglieder ihre Pakete des zweiten Milestones relativ unabhängig voneinander implementieren, da die Arbeitspakete im Gegensatz zu Milestone 1 horizontal aufgebaut waren. Am Donnerstag den 10.06.2021 verließ der nacharbeitende Implementierer das Team, ohne eines seiner bisherigen Arbeitspakete vervollständigt zu haben. Dies hatte sowohl eine Reduzierung des Funktionsumfangs des Systems, als auch ein erneutes Umstrukturieren des Implementierungsplans zur Folge, um die fehlende Arbeitskraft zu kompensieren und wichtige Kernfunktionalitäten auf die übrigen Entwickler zu verteilen. Hierbei entstand ein zu großer Mehraufwand für einen unserer Entwickler, der bereits Zeitprobleme hatte und so ebenfalls seine Arbeitspakete zum Ende von Milestone 2 nicht komplett fertigstellen konnte. Im Testprojekt kristallisierte sich gegen Ende des Milestones heraus, dass der Pfad zur Konfigurationsdatei aus dem Hauptprojekt trotz mehrstündiger Versuche

nicht aufgelöst werden konnte und deshalb der ConnectionPool keine Verbindung zur Datenbank bekam. Die Lösung dieser Schwierigkeit wurde aus Zeitmangel in den letzten Milestone verschoben.

3.3 Milestone 3

Neben der Nacharbeit der fehlenden Pakete aus dem letzten Abschnitt, war es dem Entwickler diesmal möglich, auch seine übrigen Arbeitsaufträge zu erfüllen. Hierzu wurden nicht nennenswerte Teile von Arbeitspaketen als kleine Entlastung umverteilt. Das Leseproblem im Testprojekt wurde umgangen, indem im PreTest die Konfigurationsdatei hartkodiert übergeben wurde. Ohne eine explizite Lösung des Problems war jedoch ein Fertigstellen des im Abschnitt 1 erwähnten Loggertests nicht möglich, dieser wurde gestrichen und gegen einen Test für den DataLayerInitializer ausgetauscht. Außerdem wurde die Reihenfolge der Pakete verändert, z.B. wurde die Implementierung des TrespassListeners nach hinten verlegt, um manuelle URL-Eingaben für unkompliziertes Testen bis zum Ende des Milestones möglich zu machen. Es trat dann Mitte der Woche eine bis jetzt ungeklärte Anomalie bei zwei unserer Entwickler auf, in Folge derer sich das System lokal nicht mehr starten lies. Da unser Paginationsspezialist, der das Gros der verbleibenden technisch komplizierten Teile nach der Funktionsreduktion zu tragen hatte, so noch zusätzlich Zeit aufwenden musste, geriet er mit den Tests in Verzug und es wurde beschlossen, diese zeitnah nachzuliefern. Ohne die Möglichkeit, produzierten Code zu testen, verschob sich ebenfalls die Fertigstellung des Kategoriebrowsers über das Ende des Milestones hinaus.

3.4 Allgemeine Schwierigkeiten

Eigene Implementierungen der nötigen Validatoren und Converter für das reibungslose Funktionieren der Facelets wurden in vorhergehenden Phasen großteils gar nicht spezifiziert und stellten einen ungeplanten Mehraufwand in vielen Paketen dar. Ebenfalls ungenügend und zeitaufwendig waren die Kommentare und Methodenrümpfe aus der Spezifikation, die beim Erreichen des Arbeitspakets dann angepasst werden mussten, obwohl diese bereits manchmal als Schnittstelle in anderen Paketen benutzt wurden. Da die Verantwortlichkeiten in den Dao-Klassen methodenweise verteilt waren, kam es teils vor, dass identische Hilfsmethoden redundant parallel entwickelt wurden oder im umgekehrten Fall, Hilfsmethoden von anderen verwendet und dann vom Originalautor abgeändert wurden. Durch die Reduzierung der Teamgröße und damit des Funktionsumfangs mitten in der Implementierungsphase, entstanden einige unbenutze Artefakte im Code und der Datenbank, deren Entfernung weitreichende Änderungen in den Zuständigkeitsbereichen unterschiedlicher Entwickler zur Folge gehabt hätten. Da dies oft nicht mit dem jeweiligen Zeitplan vereinbar war, wurde von deren Entfernung zunächst abgesehen. Infolgedessen rutschten Fertigstellungen von Paketen gegen Ende der Milestones oft in Zeitknappheit und aus Prioritätsgründen wurde dann die Lauffähigkeit und der Umfang der geplanten Testfälle vernachlässigt. Über alle Milestones hinweg gab es immer wieder verschiedene Probleme, die mit der Interaktion unseres Versionskontrollsystems und den unterschiedlichen Entwicklungsumgebungen der Teammitglieder zusammenhing. Hierbei wurden vor allem Projektkonfigurationen die in proprietären Metadatenformaten der jeweiligen IDEs gespeichert waren, immer wieder gegenseitig überschrieben und es gab Unterschiede in der Interpretierung der Packagestruktur. All dies resultierte in der Notwendigkeit für einige Mitglieder, mitten in der Implementierungsphase auf die Referenzentwicklungsumgebung zu wechseln, womit weitere ungeplante Einarbeitungszeit einherging.

4 Code-Metriken

Autor: Ivan Charviakou

Im Folgenden werden verschiedene Code-Metriken allgemein vorgestellt und für das erstellte Projekt angegeben. Darüber hinaus lassen sich diese Metriken in drei allgemeine Untersuchungsbereiche unterteilen: Allgemeine Metriken, Komplexitäts-bezogene Metriken, und Abhängigkeits-bezogene Metriken. Durch Angabe dieser Metriken wird eine genauere Analyse des Codes aus verschiedenen Sichten ermöglicht.

4.1 Allgemeine Metriken

Facelets	
Anzahl Facelets	29
Anzahl Zeilen XML Code	1664

Metriken nach Paket

Paketname	**Anzahl Java-Zeilen**	**Anzahl Java-Methoden**	**Anzahl Java-Klassen**
de.dedede.model.data.dtos	852	200	23
de.dedede.model.data.exceptions	4	1	1
de.dedede.model.logic.converters	218	14	10
de.dedede.model.logic.exceptions	183	25	6
de.dedede.model.logic.managed_beans	2203	240	27
de.dedede.model.logic.util	369	23	14
de.dedede.model.logic.validators	443	16	11
de.dedede.model.persistence.daos	2518	111	4
de.dedede.model.persistence.exceptions	407	76	19
de.dedede.model.persistence.util	950	51	10
Insgesamt	8147	757	125

4.2 Komplexitäts-bezogene Metriken

Metriken nach Paket

Paketname	**Durchschnittliche zyklomatische Komplexität***
de.dedede.model.data.dtos	1.14
de.dedede.model.logic.converters	1.64
de.dedede.model.logic.exceptions	1.24
de.dedede.model.logic.managed_beans	1.44
de.dedede.model.logic.util	2.48

Paketname	Durchschnittliche zyklomatische Komplexität
de.dedede.model.logic.validators	3.06
de.dedede.model.persistence.daos	2.88
de.dedede.model.persistence.exceptions	1
de.dedede.model.persistence.util	2.14

Die 10 komplexeste Methoden nach kognitiver Komplexität	
Methodenname	**Kognitive Komplexität***
de.dedede.model.persistence.util. DataLayerInitializer.consoleDialogue(Connection)	21
de.dedede.model.persistence.util. DataLayerInitializer.setUpDatabase()	20
de.dedede.model.logic.managed_beans. Lending.lendCopies()	18
de.dedede.model.logic.util. TrespassListener.afterPhase(PhaseEvent)	18
de.dedede.model.logic.managed_beans. ReturnForm.returnCopies()	15
de.dedede.model.persistence.daos. CategoryDao.createCategory(CategoryDto)	14
de.dedede.model.persistence.daos. MediumDao.getAppliedLendingLimit(Connection,String,int)	14
de.dedede.model.persistence.util. MaintenanceProcess.MaintenanceThread.run()	12
de.dedede.model.persistence.daos. MediumDao.translateNuancedSearchQuery (StringBuilder,List,NuancedSearchQuery)	11
de.dedede.model.logic.managed_beans. CategoryCreator.onload()	10

Zyklomatische Komplexität Als zyklomatische Komplexität bezeichnet man die maximale Anzahl an Pfaden im entsprechenden Kontroll-Fluss Grahpen, die sich jeweils im Vergleich zu allen anderen Pfaden in der Menge an besuchten Knoten um mindestens einen unterscheiden.

Kognitive Komplexität Im Code setzt sich die Metrik zur kognitiven Komplexität aus der Anzahl an Brüchen im Kontrollfluss und Verschachtelungen von entsprechenden Kontrollflussanweisungen zusammen. Im Gegensatz zur zyklomatischen Komplexität fasst diese Metrik die Schwierigkeit, mit der das gegebene Code zum Lesen und Verstehen ist.

4.3 Abhängigkeits-bezogene Metriken

Metriken nach Paket		
Paketname	**Anzahl Paket-Abhängigkeiten**	**Anzahl Paketen mit dieser Abhängigkeit**
de.dedede.model.data.dtos	1	8
de.dedede.model.data.exceptions	0	2
de.dedede.model.logic.converters	2	0
de.dedede.model.logic.exceptions	2	1
de.dedede.model.logic.i18n	0	0
de.dedede.model.logic.managed_beans	6	2
de.dedede.model.logic.util	4	6
de.dedede.model.logic.validators	4	0
de.dedede.model.persistence.daos	4	4
de.dedede.model.persistence.exceptions	1	6
de.dedede.model.persistence.util	4	4