Gottfried Wilhelm
Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Praktische Informatik
Fachgebiet Software Engineering

Automatisierte Registrierung von studentischen Software-Projekten in GitLab und JIRA

Automated Registration of Student Software Projects in GitLab and JIRA

Bachelorarbeit

im Studiengang Informatik

von

Aymen Maamri

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüfer: Dr. Jil Klünder
Betreuer: M.Sc. Fabian Kortum

Hannover, 31.08.2020

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 31.08.2020

Aymen Maamri

Zusammenfassung

Die Projektverwaltungs- und Problemverfolgungsanwendung Jira sowie die kontinuierliche Integrations- und Einsatzanwendung GitLab sind in der Berufswelt, neben anderen wettbewerbsfähigen Anwendungen, ein wichtiger Bestandteil im Lebenszyklus von Softwareprojekten. Um die Studierenden im Umgang mit der agilen Softwareentwicklung zu schulen, können diese Anwendungen eine wesentliche Rolle spielen, indem sie ihre Entwicklungsgeschwindigkeit erhöhen und ihre Zusammenarbeit ausbauen. Sie werden auch in verschiedenen agilen Methoden wie SCRUM und Kanban eingesetzt.
Im Rahmen der alljährlichen Veranstaltung *Software-Projekt* begegnen die Studierenden diesen professionellen Anwendungen, insbesondere Jira und GitLab. Seit Jahren steigen die Teilnehmerzahlen, sodass der aktuell noch manuelle auszuführende Prozess (einfache Skripte zum Support) zur Registrierung von studentischen Software-Projekten in Jira und GitLab noch immer schwer zu handhaben ist und zunehmend Bearbeitungszeit einnimmt. Um diese Registrierung zu erleichtern, benötigt das Fachgebiet Software Engineering der Leibniz Universität Hannover eine neue Lösung, die diesen manuellen Aufwand auf ein Minimum reduziert und den Anlegeprozess für Projekte, Teams und Studenten möglichst vollautomatisiert übernimmt.
Diese Arbeit zielt auf die Entwicklung eines neuen Jira Plugins ab. Es soll dem Fachgebiet helfen, den Registrierungsprozess für Studierenden in Jira und GitLab zu erleichtern. Die Lösung besteht darin, die REST-APIs von Jira und GitLab zu verwenden, um den Anlegeprozess für studentische Software-Projekte möglichst zu automatisieren.

Abstract

Automated Registration of Student Software Projects in GitLab and JIRA

The Project management and problem tracking application Jira as well as the continuous integration and deployment application Gitlab, are in the professional world, with other competitive applications, important components of the software projects ecosystem. aiming to train the students on the agile software development,these applications can play an essential role,through increasing their development speed and expand collaboration. they can be also used in different Agile methodologies like SCRUM and Kanban.

As part of the annual software projects, the students encounter these professional applications in particular Jira and Gitlab. Over the years, however, the number of participants is also increasing, so that the currently manual process (simple scripts for support) for registration is always difficult to handle or takes increasing processing time. To facilitate the registration of students in these tools, the Leibniz University Software Engineering Department needs a new solution that reduces this manual effort to a minimum and takes over the creation process for projects, teams and students as fully automated as possible.

This work aims at the development of a new Jira plugin. The Plugin should help the department to make the students registration process in Jira and GitLab easier. The solution is to use the REST API of Jira and Gitlab to automate the creation process for student software projects.

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Problemstellung . . . 1
- 1.2 Lösungsansatz . . . 2
- 1.3 Struktur der Arbeit . . . 2

2 Grundlagen **3**
- 2.1 Jira Grundlagen . . . 3
  - 2.1.1 Studentische Softwareprojekte erstellen in Jira . . . 3
  - 2.1.2 Erweiterung von Jira Funktionalitäten durch Plugins . 4
- 2.2 GitLab Grundlagen . . . 5
  - 2.2.1 Studentische Softwareprojekte erstellen in GitLab . . . 5
  - 2.2.2 Schnittstellen für die Kommunikation mit Gitlab . . . 6
- 2.3 Verwandte Arbeiten . . . 6

3 Konzept und Planung **7**
- 3.1 Methodisches Vorgehen . . . 7
- 3.2 Analyse und Spezifikation von Anforderungen . . . 8
  - 3.2.1 Funktionale Anforderungen des Registrierungsplugins . 8
  - 3.2.2 Nicht-Funktionale Anforderungen des Registrierungsplugins . . . 10
  - 3.2.3 Analyse der Registrierungsplugin Use Cases . . . 10
  - 3.2.4 Modellierung der Benutzeroberfläche . . . 17
- 3.3 Modellierung und Entwurf des Lösungsansatzes . . . 19
  - 3.3.1 Phase 1: Allgemeine Konzeptionierung . . . 19
  - 3.3.2 Phase 2: Detaillierte Konzeptionierung . . . 24

4 Implementierung **27**
- 4.1 Eingesetzte Technologien . . . 27
  - 4.1.1 Atlassian Plugin SDK 6.3.10 . . . 27
  - 4.1.2 Representational State Transfer . . . 27
  - 4.1.3 JavaScript Object Notation . . . 29
- 4.2 Backend . . . 30
  - 4.2.1 Implementierung des WebWork Action . . . 30

4.2.2 Realisierung der Plugin REST Ressource 30
4.2.3 Interaktion mit der Datenbank durch Active Objects . 31
4.3 Frontend 32
4.3.1 Javascript Controller 32
4.3.2 Realisierung der Benutzeroberfläche 34

5 Validierung der Funktionalitäten 39
5.1 Validierungstest des Registrierungsplugins 39
5.2 Belastungstest des Registrierungsplugins 41
5.2.1 Angewendete methode 41
5.2.2 Erwartetes Ergebnis 42
5.2.3 Teststrategie 42

6 Zusammenfassung und Ausblick 49
6.1 Zusammenfassung 49
6.2 Ausblick 50

A Aktivitätsdiagramme 51
A.1 Anstehende Projekte editieren 51
A.2 Invalide Daten der Studierende editieren 52

B Sequenzdiagramme 53
B.1 Anstehende Projekte editieren 54
B.2 Invalide Daten der Studierende editieren 55

Kapitel 1

Einleitung

Das Fachgebiet Software Engineering bietet alljährlich im Rahmen des Bachelor-Studiums die Veranstaltung Software Projekt an. Daran nehmen jährlich etwa 150 Studierende teil und bilden Teams zur Entwicklung verschiedener Software-Projekte. Das Fachgebiet gewährleistet, dass die Studierenden im Rahmen des SWP die verschiedenen Funktionalitäten von Jira und GitLab Kennenlernen und ausüben. Dafür ist eine Registrierung der Studierenden und deren Projekte in Jira und GitLab erforderlich. Derzeit führt die SWP-Organisation den Registrierungprozess semi-automatisch aus. Hierzu verwendet sie zur Unterstützung Selenium Skripte[10], um die sich wiederholenden Aufgaben zu automatisieren. Die Analyse des aktuellen Registrierungprozesses für Studierende in Jira und GitLab hat es ermöglicht, eine Menge von Mängeln zu identifizieren.

1.1 Problemstellung

Die Teilnehmerzahl steigt jährlich an, die derzeitige Lösung ist nicht stabil und das Zeitfenster für den Registrierungsprozess ist konstant geblieben. Diese Punkte erfordern einen enormen manuellen Bearbeitungs- und Zeitaufwand für die Admins. Bisher wurden Skripte für die Testsoftware *Selenium* verwendet, die von der HTML-Struktur verschiedener Jira- und GitLab-Seiten abhängig sind, um die Registrierungsprozesse über die Webanwendung von Jira und GitLab durchzuführen. Das bewirkte, dass jede Änderung dieser Struktur, die sich immer bei einem System Upgrade ergeben kann, zu entsprechenden Änderungen in den Skripten führt. Darüber hinaus kann es in dem aktuellen Prozess zu einer unvollständigen Registrierung von Studierenden und Projekten kommen oder es kann eine Duplizierung von Benutzerkonten in einem oder beiden Systemen verursachen. Folglich muss die Nachregistrierung von ungültigen Studierendeneinträgen manuell erfolgen, da beide Systeme keinen Feedback darüber geben, welche Einträge ungültig sind und diese somit nicht in der Datenbank speichern.

Angesichts der im Laufe der Jahre gestiegenen Anzahl der Studierenden, könnte festgestellt werden, dass die von dem Fachgebiet angewandte Methode ineffizient geworden ist und nicht den Erwartungen an eine schnelle und zuverlässige Registrierung entspricht.

1.2 Lösungsansatz

Der allgemeine Lösungsansatz dieser Problemstellung ist ein Jira-Plugin, das den Registrierungsprozess direkt in Jira integriert. Dieser Ansatz soll die Ausführungszeit des Prozesses erheblich verkürzen und eine langfristige Einsetzbarkeit sicherstellen. Das Plugin hat das Ziel die manuellen Intervention des Registrierungsprozesses auf ein Minimum reduzieren. Gleichzeitig soll es nicht von der unterschiedlichen HTML-Struktur der Jira- und Gitlab-Seiten abhängig sein, sondern von ihren APIs, die konsistent sind und ihre Änderung und Dekommissionierung vorhersehbar sind.
Nach Eingabe der erforderlichen Daten der Registrierung wird eine Validierung stattfinden, um die syntaktische Korrektheit und die Integrität der Daten zu verifizieren. Die nächste Überprüfung soll zeigen, ob die Benutzer und Projekte bereits in einem oder beiden Systemen existieren und danach den Prozess entsprechend anpassen, so dass das Plugin mit dem Admin interagiert und ihm verschiedene Möglichkeiten anbietet, wie der Prozess fortgesetzt werden soll. Danach wird das Plugin die ungültigen Einträge speichern und dem Admin die Möglichkeit geben, sie jederzeit durch eine einfache Mensch-Maschine-Interaktion zu verwalten. Schließlich wird ein *'Audit Trail'* über jeden ausgeführten Registrierungsprozess in der Datenbank gespeichert.
Durch die Verwendung der REST Ressourcen von Jira und GitLab bleibt das Plugin unter Berücksichtigung der verschiedenen Jira- und GitLab Updates über die zeit stabil.

1.3 Struktur der Arbeit

Zunächst werden in Kapitel 2 die grundlegenden Begriffe und Definitionen von Jira und GitLab, die für dieser Arbeit relevant sind, dargestellt. Das Kapitel 3 beschreibt methodisch die Analyse und Konzeptionierung des zu implementierenden Plugins. Folgend wird im Kapitel 4 die Implementierung des Plugins beschrieben. Kapitel 5 enthält die Bewertung und Interpretation der durchgeführten Software-Validierung. Im letzten Kapitel 6 wird ein Fazit dieser Arbeit gezogen und mögliche Erweiterungen des Plugins vorgestellt.

Kapitel 2

Grundlagen

Dieses Kapitel beschreibt die wichtigsten thematischen Technologien und Grundkonzepte, welche für das Verständnis der Arbeit notwendig sind.

2.1 Jira Grundlagen

Jira ist ein *Application Lifecycle Management*[12] von Atlassian. Es beinhaltet verschiedene Pakete für unterschiedliche Kundenanforderungen und wird in verschiedenen Bereichen der Berufswelt eingesetzt. Zudem gibt es Jira in verschiedenen Varianten. Für diese Arbeit wurde Jira *Software* eingesetzt. Jira unterstützt den Teams bei der Planung und Verwaltung der Entwicklungsaufgaben im Rahmen von Software-Projekten. In diesem Abschnitt werden einige Kernbegriffe von Jira erläutert und ich werde einen Blick auf das Konzept Jira Add-on werfen.

2.1.1 Studentische Softwareprojekte erstellen in Jira

Ein studentisches Software Projekt (SWP) in Jira umfasst sich in verschiedenen Projektartefakte:

- *Jira Projekt*: Als Grundlage für das SWP wird ein Projekt vom Typ Software in Jira erstellt. Projekte in Jira werden anhand ihres Namens und Schlüssels identifiziert. Jira Software Projekte unterstützen die agile Entwicklung mit dem Einsatz von SCRUM und Kanban und sind damit für Studenten geeignet, denn die SWPs eine SCRUM ähnliche Entwicklung verfolgen.

- *Jira Projekt Rollen*: In Jira gibt es die drei Standardprojektrollen: *Administrators Developers* und *Users*. Ein Admin verwaltet ein Projekt, ein Developer arbeitet an *issues*(Aufgabe, Problem,Fehler,...) innerhalb eines Projekts und ein User protokolliert issues in einem Projekt.

- *Jira Benutzerkonto*: Für jeden Studenten, der Teilnehmer an einem SWP ist, wird ein Jira Benutzerkonto erstellt. Ein Student wird durch einen Benutzernamen identifiziert. Sämtliche Studenten werden innerhalb des Jira Projekts der Rolle des *Developer* zugeordnet.

- *Jira Gruppe*: Eine Gruppe in Jira versammelt Studenten aus einem SWP. Gruppen wurden in Jira hauptsächlich genutzt, um globale Berechtigungen oder projektspezifischen Zugriff zu gewähren.

- *Berechtigungsschema*: Ein Berechtigungsschema definiert die Berechtigungen für die Projektmitglieder, indem es der mit dem Projekt zugehörigen Gruppe zugewiesen wird. Berechtigungschemata umfassen verschiedene Arten von Berechtigungen, wie Projektberechtigungen, issues Berechtigungen Kommentarberechtigungen, Anhangsberechtigungen und Zeiterfassungsberechtigungen. Jedes erstellte Projekt wird einem personalisierten Berechtigungsschema zugeordnet.

- *Benachrichtigungsschema*: Mit einem Benachrichtigungsschema kann kontrolliert werden, welche Mitglieder des Projekts Benachrichtigungen über den Stand oder Änderungen erhalten. Jedes erstellte Projekt wird einem personalisierten Benachrichtigungsschema zugeordnet.

2.1.2 Erweiterung von Jira Funktionalitäten durch Plugins

Neben zahlreichen Funktionalitäten für verschiedene Arten von Benutzern, bietet Jira auch die Möglichkeit, die Software spezifischen Bedürfnissen anzupassen und zu erweitern. Atlassian stellt dafür ein SDK zur Verfügung, mit dem Entwickler Jira Plugins (Jira Add-ons)[4] erstellen, Installieren und bereitstellen. Zudem unterhält Atlassian Java-APIs und –Bibliotheken, auf die zugegriffen werden kann und die programmatisch mit Jira interagieren.

Ein Jira Plugin besteht aus mehreren Modulen. Die für diese Arbeit relevanten, werden in der folgenden Tabelle vorgestellt:

Tabelle 2.1: Jira Plugin Modulen

Das WebWork Action	Hierbei handelt es sich um ein einfaches Servlet, mit dem sich eine eigene, komplette Seite entwickeln lässt. Eine Klasse wurde definiert, die mit einem bestimmten URL verbunden ist. Es kann angegeben werden, ob die Seite öffentlich für alle Jira Users oder nur für Admins sichtbar sein soll.
REST Ressourcen	Jira setzt hierbei auf REST basierte Schnittstellen. Diese stellen eine Menge von Funktionen zur Verfügung, um Daten aus Jira zu lesen, Jira zu konfigurieren oder Daten zu schreiben. Auch besteht die Möglichkeit eigene REST Ressourcen zu entwickeln.
AJS	Hierbei handelt es sich um das Atlassian JavaScript Framework, das eine leichte jQuery-Bibliothek beinhaltet.
AUI	Ein AUI ist ein HTML/CSS Framework, das der Entwicklung von Benutzeroberflächen dient.
WebRessourcen	Statische Dateien, wie z.B. JavaScript Datein, werden definiert und über einen Schlüssel für die Nutzung, z.B. im JavaScript Code oder in einem Template, bereitgestellt.

2.2 GitLab Grundlagen

Gitlab bietet eine einfache Versionierung und Verwaltung des Codes. Es ermöglicht Benutzern, Git-Repositories schnell und effektiv zu verwalten. Außerdem verfügt GitLab über eine zuverlässige Rechteverwaltung, die die Verwaltung der Rechte der Studierenden erleichtert. In diesem Abschnitt werden einige Kernbegriffe von GitLab erläutert, die für die Arbeit relevant sind.

2.2.1 Studentische Softwareprojekte erstellen in GitLab

Studentische Software Projekte in GitLab sind grundsätzlich GitLab repositories, welche Gitlab-Projekte sind. Sie werden verwendet, um Probleme in der Entwicklung zu verfolgen. Für jedes SWP wird ein repository erstellt. In GitLab können Projekte in einer GitLab-Gruppe begrenzt werden. Dies dient dazu, SWPs aus einem bestimmten Jahr zusammenzuführen und so Projekte von Administratoren leichter verwalten zu können. Wie Jira werden auch in GitLab für die Studierenden Benutzerkonten erstellt und können durch den Benutzernamen identifiziert werden. Die Stundierenden werden

dann den jeweiligen repositories als *Developer* zugeordnet. Ein Developer in GitLab erhält Zugang zum jeweiligen Projekt und kann darin zu mehreren Aktivitäten beitragen, z.B. Verwaltung von Entwicklungsbranchen oder Erstellung von commits. SWP Mitglieder in GitLab erhalten einen zeitlich festgelegten Zugang und werden nach diesem Datum inaktiv gestellt, damit die Projekte nach dem Ende der Veranstaltung unverändert bleiben.

2.2.2 Schnittstellen für die Kommunikation mit Gitlab

GitLab bietet die Möglichkeit an, mit ihm remote zu kommunizieren und seine Ressourcen von anderen Anwendungen zu manipulieren. Diese Kommunikation erfolgt über REST Ressourcen, die GitLab seinen Benutzern zur Verfügung stellt. Um diese REST Aufrufe erfolgreich auszuführen, muss ein Sicherheitstoken bereitgestellt werden. Dieses Token kann durch Administratoren in GitLab erstellt werden und kann Zugriffsrechte auf verschiedene Gitlab Ressourcen festlegen.

2.3 Verwandte Arbeiten

Der Fokus dieser Arbeit liegt auf der Automatisierung des Registrierungsprozesses für studentische Software-Projekte in Jira und GitLab. Eine Dokumentation des Themas ist notwendig, um mit der Konzeptionierung und der Implementierung beginnen zu können. In diesem Abschnitt werden Referenzen zitiert, die verwendet wurden, um zu verstehen, wie ein Jira-Plugin entwirft und implementiert wurde und die Verwaltung von Benutzern und Projekten in Jira und GitLab. *Doar* [3] erläutert wie verschiedene Jira-Objekte miteinander interagieren, wie die Rolle von Schemata und ihre Verwaltung in Jira. Er beschreibt auch die verschiedenen Funktionen, die ein Administrator in Jira ausführen kann und die verschiedene Möglichkeiten, Projekte und Gruppen in Jira zu verwalten. Außerdem erläutert *Kuruvilla* [6] die Grundlagen des Entwicklungsprozesses von Jira-Plugins. Dafür wird ausführlich auf die Einrichtung einer Entwicklungsumgebung eingegangen, mit der kundenspezifischen Plugins erstellt und getestet werden können. Er behandelt ausführlich die Jira-Architektur und die verschiedenen Möglichkeiten zur optimalen Nutzung dieser bei der Entwicklung von Plugins. Ebenfalls wird die remote Kommunikation zwischen Jira und anderen Anwendungen behandelt. Schließlich erläutert *Hethey* [5] die Verwaltung von GitLab Repositories und Gruppen. mit der Einführung von Rollen und Berechtigungen innerhalb von GitLab erklärt er, wie Teams in GitLab zusammenarbeiten können.

Kapitel 3

Konzept und Planung

In diesem Kapitel wird auf die Konzeptionierung des Registrierungsplugins näher eingegangen. Der Hauptzweck besteht darin, die Anforderungen des Registrierungsplugins zu extrahieren und zu analysieren, die Hauptmodule des Plugins zu entwerfen und zu definieren, wie sie miteinander interagieren. Das Endergebnis dieser Konzeptionierung soll ein zielgerichtetes Lösungskonzept für die Problemstellung sein

3.1 Methodisches Vorgehen

Zur Konzeptionierung des Plugins wurde ein iteratives Software Entwicklungsmodell[7] verwendet. Dieses Modell zielt darauf ab, ein Softwareprodukt anhand von Iterationen zu entwickeln. Jede Iteration basiert auf der vorhergehenden Iteration und bringt neue Funktionalitäten und Optimierungen in das entwickelte Projekt. Die initiale Iteration soll eine einfache Grundlage für das Projekt unter Bezugnahme der Basisanforderungen liefern. Darüber hinaus wird der Grad der Entwurfspräzision mit jeder Iteration immer präziser.

Auswahl des iterativen Modells

Für diese Arbeit wurde das iterative Modell ausgewählt, da die Hauptanforderungen bereits bekannt sind, wohingegen einige Funktionalitäten und gewünschte Erweiterungen erst im Laufe des Entwicklungsprozesses entstehen. Zudem garantiert es, das in jeder Phase oder jedem Schritt erstellte Modell verfeinert und verbessert werden kann. Nach der Analyse der Basisanforderungen des Registrierungsplugins werden in erster Linie Iterationen ausgenutzt, um schrittweise die Probleme und Fehler zu identifizieren, die während des Registrierungsprozesses auftreten können, und anschließend evaluiert, wie das Registrierungsplugin damit umgeht. Diese Probleme können unabsichtlich von dem Benutzer erzeugt werden, außerhalb der Kontrolle des Benutzers liegen oder können unerwartet auftreten.

Anwendung des iterativen Modells auf diese Arbeit

Damit die Arbeit die Anforderungen an das Registrierungsplugin erfüllt, wurde des iterative Modell gewählt, um bei jeder Iteration die geleistete Arbeit mit den Wünschen der zukünftigen Plugin-Benutzer verfeinern zu können. In dieser Arbeit wurden drei Iterationen durchgeführt. In der ersten Iteration wurde ein nominales Registrierungsszenario entworfen (Basisregistrierungsprozess), d.h. die Behandlung der von einer csv-Datei gelieferten Daten, die über das Anmeldungsportal für das SWP erzeugt werden und die notwendige Daten von Studierenden und Projekten für die Registrierung enthält, ohne mögliche Probleme zu berücksichtigen. In der zweiten Iteration wurden nach der Durchführung von Interviews mit den Verantwortlichen für die Studentenregistrierung in Jira und GitLab und Selbstdurchführung des Registrierungsprozesses unter verschiedenen Umständen, mögliche Probleme identifiziert, die unabsichtlich von dem Benutzer erzeugt und während des Registrierungsprozesses auftreten könnten. Gemäß den neuen Informationen wurde das Modell verbessert, um diese Probleme vorhersehen zu können (erweiterter Registrierungsprozess). Schließlich wurde in der letzten Iteration die Möglichkeit der menschlichen Interaktion hinzugefügt, um alle Probleme, die nicht automatisiert gelöst werden können, zu bewältigen. Beispielsweise besteht es die Möglichkeit, ungültige Daten von Studierenden einfach mittels des Plugins zu ändern und direkt dem zugehörigen Projekt zuzuweisen (interaktiver Registrierungsprozess).

3.2 Analyse und Spezifikation von Anforderungen

Der erste Schritt bei der Entwicklung einer Anwendung ist die Identifizierung der Benutzeranforderungen, um die Liste der Funktionalitäten zu ermitteln, die in der Anwendung vorhanden sein müssen. Diese Phase ist unerlässlich, um die Möglichkeiten der Anwendung zu bestimmen, was eine Validierung der Anwendung nach ihrer Entwicklung ermöglicht. Die Anforderungen des Registrierungsplugins wurden anhand von Interviews mit der SWP-Organisation identifiziert. Im Folgenden wird die Analyse der verschiedenen funktionellen und nicht-funktionellen Anforderungen der Anwendung identifiziert. Danach werden die Akteure und die zugehörigen Use Cases vorgestellt.

3.2.1 Funktionale Anforderungen des Registrierungsplugins

Funktionale Anforderungen identifizieren die Funktionalitäten, die das System bereitstellen muss. Im folgenden sind die von der Anwendung erwarteten funktionale Anforderungen :

- **A1**: Das Plugin soll ausschließlich vom Administrator gesteuert sein.

- **A2**: Das Plugin soll in der Lage sein, die Verbindungsdaten für GitLab aufzunehmen.
- **A3**: Studenten, deren E-Mail Adresse, Vorname oder Nachname syntaktisch ungültig sind, sollten vom Registrirungsprozess ausgelassen werden.
- **A4**: Es soll geprüft werden, ob Studenten schon Konten in Jira oder GitLab haben, in diesem Fall müssen die bestehenden Konten verwendet werden.
- **A5**: Alle Ereignisse des Registrierungsprozesses sollen in einem *Logger* Bereich geloggt werden.
- **A6**: Der Zustand des Registrierungsprozess soll jederzeit sichtbar und nachvollziehbar sein.
- **A7**: Jedes Projekt erhält sein eigenes Berechtigungs- und Benachrichtigungsschema.
- **A8**: Die erstellten Studentenkonten in GitLab bekommen ein Ablaufdatum von genau sechs Monaten.
- **A9**: Die erstellten Projekte in GitLab sollen unter einer Gruppe, die als *SWP<Veranstaltungsjahr>* genannt wird, gruppiert werden.
- **A10**: Studenten erhalten in Jira und GitLab jeweils den gleichen Username.
- **A11**: In Jira werden Berechtigungsschema, Benachrichtigungsschema und Gruppen den jeweiligen Projektname erhalten.
- **A12**: Die ungültigen Studierenden-Einträge sollen jederzeit vom Admin durch eine einfache grafische Oberfläche nachgearbeitet und zu den Projekten zugeordnet werden.
- **A13**: Bei einem unerwarteten Fehler im Registrierungsprozess, soll das Plugin in der Lage sein, bei der nächsten Ausführung die schon erstellten Projekte zu erkennen und dem Admin die Möglichkeit geben, diese Projekte zu löschen, zu benutzen oder ihre Namen zu ändern.
- **A14**: Die Registrierungsdatei und das Registrierungslog sollen am Ende des Prozesses in der Datenbank gespeichert werden.
- **A15**: Die notwendigen Daten für den Registrierungsprozess müssen in einer *CSV-Datei* enthalten sein.

- **A16**: Die Header der Registrierungsdatei lauten (Mat, Firstname, Lastname, Uniqueprojectname, Projectname). Das Plugin soll in der Lage sein, die Datei zu analysieren, unabhängig von der Reihenfolge und die Groß- und Kleinschreibung der Header. Jede Zeile der Registrierungsdatei soll einen Studierenden und das Projekt, zu dem er gehört, darstellen.
- **A17**: Das Plugin soll im *Mozilla Firefox* ab Version 79 funktionieren.

3.2.2 Nicht-Funktionale Anforderungen des Registrierungsplugins

- **N1**: Der Quellcode soll gut dokumentiert und von anderen Entwicklern leicht zu verstehen sein.
- **N2**: Das Plugin muss flexibel sein, um die Registrierungsdatei Format problemlos ändern zu können.
- **N3**: das Registrierungslog soll verständlich und gut gegliedert sein.

3.2.3 Analyse der Registrierungsplugin Use Cases

Anhand von Use Cases wird das System abgegrenzt und sein Verhalten beschrieben. Use Cases stellen eine funktional orientierte Darstellung des Systems dar.

Identifizierung von Akteuren

Ein Akteur ist eine Person, Hardware oder Software, die mit dem System interagiert, um einen Mehrwert zu erzielen. Die zukünftigen Akteure des Registrierungsplugins sind:

- Administrator: der am meisten privilegierte Benutzer, er verwaltet die Zugriffsrechte und die Benutzerprivilegien. Außerdem kann er alle Jira und GitLab Funktionen ausführen.
- Jira: Die im Fachgebiet installierte Jira Anwendung
- GitLab: Die im Fachgebiet installierte GitLab Anwendung

Der Basisregistrierungsprozess

Der Basisregistrierungsprozess stellt die Erstellung der studentischen Softwareprojekte in Jira und GitLab dar, ohne mögliche Probleme oder Fehler zu berücksichtigen, die während des Prozesses auftreten können. Dieses Szenario umfasst nur das Hochladen und Parsen der Registrierungsdatei, um SWP-Objekte vorzubereiten, die die Informationen über das Projekt und seine

Teilnehmer enthalten, sowie die Kommunikation mit Jira und GitLab, um die Projekte, die zugehörigen Objekte und die Studentenkonten in beiden Systemen zu erstellen. Die Basisregistrierung berücksichtigt auch nicht die Rückmeldungen aus der Kommunikation mit Jira und GitLab. Dieser Prozess führt nur dann zu einer erfolgreichen Registrierung, wenn es keine Probleme mit den bereitgestellten Daten gibt und alle Kommunikationen von Jira und GitLab ebenfalls erfolgreich waren. In Abbildung 3.1 stellen die blau markierten Aktionen den Basisregistrierungsprozess dar. Die Abbildung 3.1 stellt den Basisregistrierungsprozess und den erweiterten Registrierungsprozess dar. Beide Prozesse werden nur in einem Use Case zusammengefasst, da die erweiterte Registrierung auf die Basisregistrierung gründet.

Der erweiterte Rregistrierungsprozess

Der erweiterte Registrierungsprozess bringt neue Funktionen mit sich. Der überwiegende Teil davon bringt die Möglichkeit, die geparsten Daten auf verschiedene Weise zu testen, um Fehler aufzufinden. Die vom Registrierungsplugin durchzuführenden Überprüfungen sollen vor und während des Registrierungsprozesses stattfinden und die Gültigkeit der Registrierungsdatei, die Überprüfung der bereitgestellten GitLab-Verbindungsdaten und die syntaktische Korrektheit und die Integrität der verschiedenen Einträge in der Registrierungsdatei umfassen. Weiterhin erfolgt vor Beginn des Prozesses die Erstellung von Benutzernamen für Benutzerkonten, wobei die extrahierten Benutzernamen gegen Benutzernamen in Jira und GitLab getestet werden müssen, mit dem Ziel, eindeutige Benutzernamen für die SWP-Teilnehmer zu definieren. Während des Registrierungsprozesses müssen weitere interne Überprüfungen durchgeführt werden. Vor Erstellung eines Studierendenkontos muss geprüft werden, ob der Studierende bereits ein Konto in Jira oder GitLab besitzt. In diesem Fall muss das bestehende Konto verwendet werden. Die gleiche Logik wird bei anderen Registrierungsobjekten verwendet. Schließlich muss die Erweiterung des Basisregistrierungsprozesses das von Jira und GitLab erhaltene Feedback verarbeiten und den Prozess gegebenenfalls entsprechend anpassen. In Abbildung 3.1 stellen die grau markierten Aktionen den Erweiterungen an den Basisregistrierungsprozess dar.

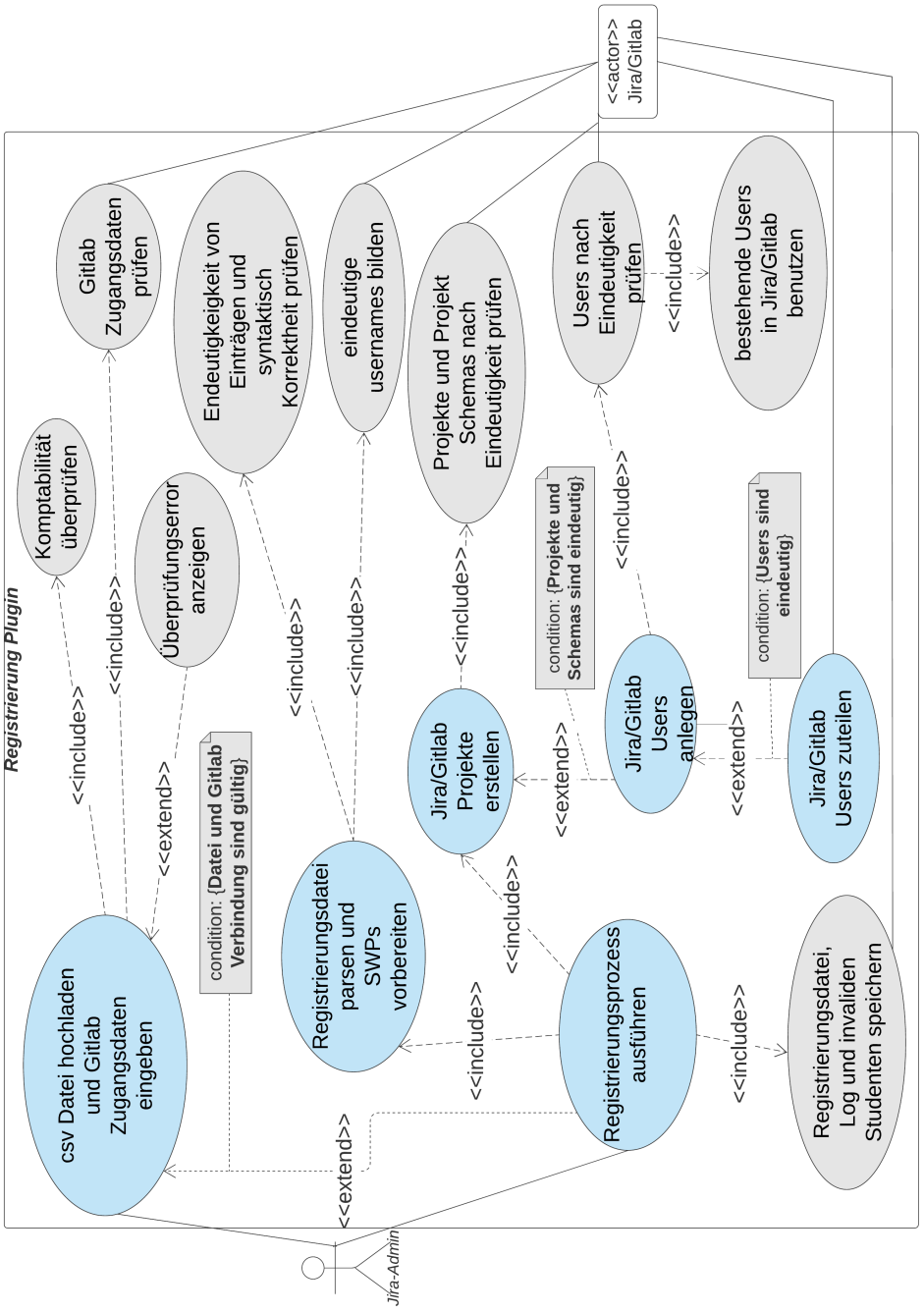


Abbildung 3.1: Erweiterter Registrierungsprozess

Die Tabelle 3.1 stellt eine ausführliche Beschreibung zum Use Case in der Abbildung 3.1 bereit.

Tabelle 3.1: Use Case: Erweiterter Registrierungsprozess

UC	Erweiteter Registrierungsprozess
Umfeld	Registrierung Plugin unter Jira Software
System Grenzen	Jira-Admin, Registrierungsdatei und Gitlabverbindung existieren bereits
Ebene	Hauptebene
Hauptakteure	Jira-Admin
Stakeholder u. interessen	Jira-Admin möchte schnell und problemlos den Registrierungsprozess der Studenten ausführen
Voraussetzungen	1- Registrierungsdatei ist gültig 2- Jira-Admin ist eingeloggt und hat Verbindung zu Jira 3- Jira-Admin hat Verbindung zu Gitlab
Garantie	Der Prozess wird erfolgreich ausgeführt und positiv gemeldet oder es wird eine Fehlermeldung angezeigt
Erfolgsfall	Der Prozess wurde erfolgreich ausgeführt
Auslöser	Jira-Admin lädt eine csv Datei in der Plugin Seite hoch und gibt die Gitlab Zugangsdaten an
Beschreibung	1- Jira-Admin lädt eine csv Datei hoch und gibt Gitlabzugangsdaten ein 2- Jira-Admin klickt Start 3- Plugin prüft, ob Datei und Zugangsdaten gültig sind. 4- Plugin konsumiert Jira REST Ressourcen und passt den Prozess aufgrund der Responses an 5- Plugin konsumiert Gitlab REST Ressourcen und passt den Prozess aufgrund der Responses an 6- Plugin speichert den Registrierung Datei un Log und den invaliden Studeten in der DB 7- Plugin meldet Prozess war erfolgreich
Erweiterungen	3a-1 Plugin meldet Datei und/oder Zugangsdaten sind ungültig

Der interaktive Registrierungsprozess

Der interaktive Registrierungsprozess hat den Hauptzweck, eine einfache Interaktion mit dem Admin zu ermöglichen, um anstehende Probleme, die nicht automatisiert gelöst werden können, zu lösen. Diese Interaktion wird in zwei Bereichen unterteilt, nämlich am Anfang und am Ende des Prozesses Interaktionen. Die Interaktion, die am Anfang des Prozesses stattfindet, besteht in der Bearbeitung bereits bestehender Projekte in Jira oder GitLab. Dieses kann dadurch verursacht werden, dass ein Name oder ein Schlüssel eines Projekts in der Registrierungsdatei mit einem bereits existierenden Projekt in einem oder beiden Systemen übereinstimmt. In diesem Fall soll der Admin in der Lage sein, das Projekt entweder aus dem System zu löschen,

es zu verwenden oder den Namen des Projekts in der Registrierungsdatei zu ändern. Die Interaktion, die am Ende des Prozesses stattfindet, geht es um die ungültigen Daten des Studierenden, die beim Parsen der Datei ausgeschlossen werden. Der Admin soll dann in der Lage sein, das durch das Plugin identifizierte Problem zu korrigieren . Das Plugin muss danach einige interne Überprüfungen durchführen, um zu überprüfen, ob die eingegebenen Daten gültig sind und gegebenenfalls den Studierenden in Jira und GitLab bearbeiten und ihn dem zugehörigen Projekt zuweisen. Ein weiterer Aspekt, der diesem interaktiven Prozess hinzugefügt wird, ist die Protokollierung der Registrierungsereignisse. Sämtliche Ereignisse, die während des Prozesses auftreten, müssen klar und aussagekräftig protokolliert werden.

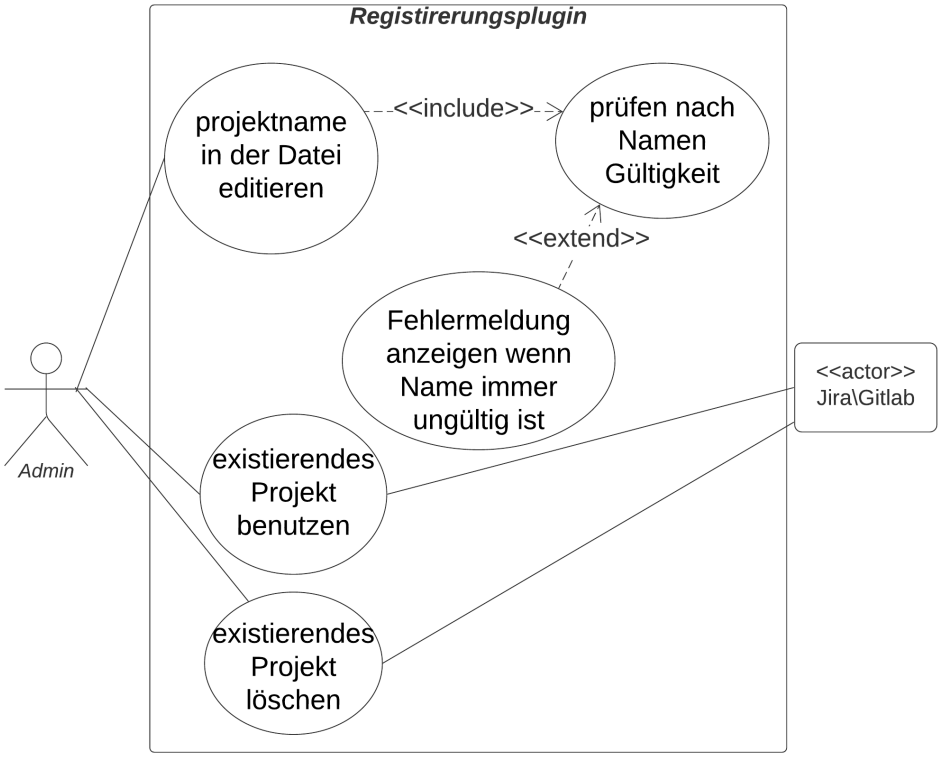


Abbildung 3.2: Use Case: Anstehende Projekte auslösen

Die Abbildung 3.2 stellt die Use Case für das Auslösen von Konflikten, die durch bereits existierende Projekte verursacht werden. Die Tabelle 3.2 stellt eine ausführliche Beschreibung zum Use Case in der Abbildung 3.2 bereit.

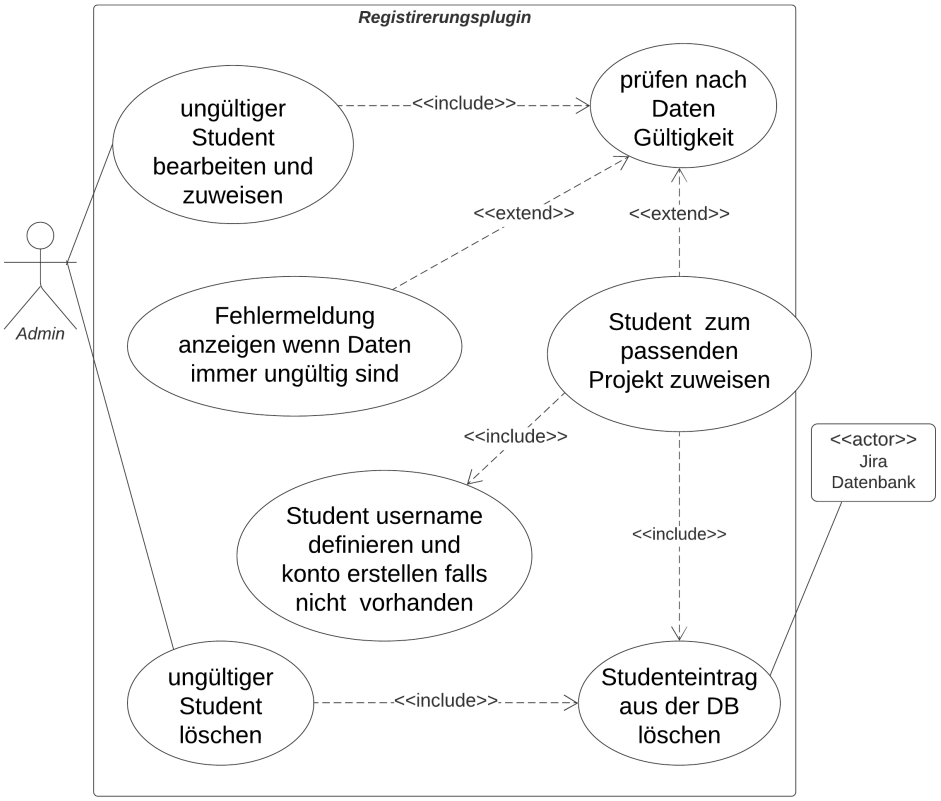


Abbildung 3.3: Use Case: Invalide Studierendendaten editieren

Die Abbildung 3.3 stellt eine Use Case für das Editieren von invaliden Studenteneinträgen und zu den Projekten zuweisen. Die Tabelle 3.3 stellt eine ausführliche Beschreibung zum Use Case in der Abbildung 3.3 bereit.

Tabelle 3.2: Use Case: Anstehende Projekte auslösen

UC	Anstehende Projekte auslösen
Umfeld	Registrierung Plugin unter Jira Software
System Grenzen	Jira-Admin ist eingellogt, Registrierungsdatei wurde verifiziert und geparst und Gitlab Zugangsdaten sind gültig
Ebene	Hauptebene
Hauptakteure	Jira-Admin
Stakeholder u. interessen	Jira-Admin möchte schnell und problemlos die existierenden Projektkonflikte auslösen
Voraussetzungen	1- Datei ist verifiziert und geparst 2- Jira-Admin ist eingeloggt und hat Verbindung zu Jira 3- Jira-Admin hat Verbindung zu Gitlab
Garantie	Die existierende Projekte werden gelöscht oder verwendet, oder die zu erstellenden Projekten werden bearbeitet
Erfolgsfall	Der Prozess wird mit dem Registrierungsprozess weitermachen
Auslöser	Jira-Admin klickt *Delete*, *use existing prroject* oder *edit*
Beschreibung	1- Jira-Admin klickt *edit* 2- Ein Pop-up Fenster öffnet sich für das Editieren von Projektdaten 3- Admin modifiziert den Daten und clickt *continue* 4- Plugin prüft, ob Projektdaten eindeutig sind 5- Plugin meldet Daten OK
Erweiterungen	1a-1 Jira-Admin klickt*use existing prroject* 1a-2 Existierendes Projekt zum SWP Referenzieren 1b-1 Jira-Admin klickt *Delete* 1b-2 plugin wird das existierende Projekt aus Jira und/oder Gitlab löschen 1b-3 Plugin meldet Projekt gelöscht 5a-1 Plugin meldet Daten sind immer ungültig

Tabelle 3.3: Use Case: Invalide Studenteneinträge editieren

UC	Invalide Studentendaten editieren
Umfeld	Registrierung Plugin unter Jira Software
System Grenzen	Jira-Admin, Registrierungsprozess ist bereits fertig
Ebene	Hauptebene
Hauptakteure	Jira-Admin
Stakeholder u. interessen	Jira-Admin möchte schnell und problemlos die invaliden Studenten editieren und zu Projekten zuweisen
Voraussetzungen	1- Registrierungsprozess ist bereits fertig 2- Jira-Admin ist eingeloggt und hat Verbindung zu Jira 3- Gitlab Zugangsdaten sind immer noch gültig
Garantie	Die invaliden Studenten werden entweder editiert und zum Projekt zugewiesen oder gelöscht
Erfolgsfall	Invalide Studenten sind zu den Projekten zugewiesen
Auslöser	Jira-Admin klickt *Edit* oder *Delete*
Beschreibung	1- Jira-Admin klickt *Edit* 2- Ein Pop-up Fenster öffnet sich für das Editieren von Studentendaten 3- Admin editiert die Daten und clickt *edit and assign* 4- Plugin prüft, ob Studentendaten gültig sind und bildet seiner *username* 5- Plugin prüft, ob Student schon ein Konto in Jira oder Gitlab hat. 6- Plugin erstellt ein Konto für den Student und weist ihm zum Projekt zu
Erweiterungen	1a-1 Jira-Admin klickt *Delete* 1a-2 Studenteintrag aus der DB löschen 4a-1 plugin meldet Daten sind immer ungültig 5a-1 Plugin weist existierendem Studentenkonto zum Projekt zu

3.2.4 Modellierung der Benutzeroberfläche

Eine Benutzeroberfläche, die die beschriebenen Anforderungen des Registrierungsplugin übersetzt, ist unerlässlich, um den Bedürfnissen des Benutzers gerecht zu werden. In diesem Teil wird ein Mockup vorgestellt, das einen Überblick über die zukünftige GUI enthält. Das Design des Mockups wurde mit der SWP-Organisation abgestimmt und unter Berücksichtigung der von Jira angebotenen technischen Schnittstellen zur Entwicklung einer Benutzeroberfläche erstellt. Dieses Mockup wurde während der verschiedenen Iterationen der Entwicklung des Plugins modifiziert und bereichert. Das allgemeine Design und die Erwartungen an die Benutzeroberfläche wurden

aus den während der Iterationen durchgeführten Interviews mit der SWP-Organisation abgeleitet.
Das Registrierungsplugin soll dem Admin eine einfache und adäquate Benutzeroberfläche bereitstellen. Der Admin soll in der Lage sein, alle seine Interaktionen mit dem Plugin problemlos zu absolvieren. Schließlich soll die Schnittstelle aussagekräftige Erfolgs-, Fehler- und Warnmeldungen liefern, um den Admin während des Prozesses zu leiten. Die Gestaltung der Benutzeroberfläche erfolgt ebenfalls während der Iterationen. Im Folgenden wird die Entwicklung des Designs anhand von Abbildung 3.4 beschrieben.

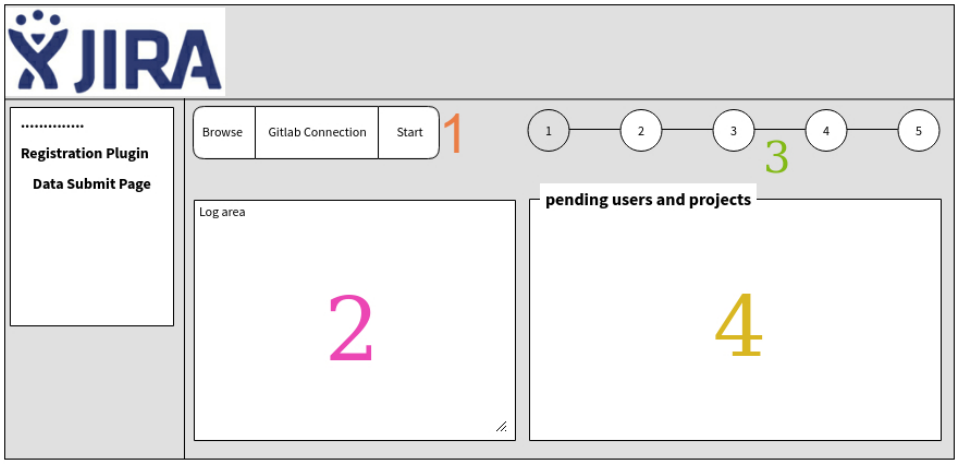


Abbildung 3.4: Plugin Hauptseite Mockup

1.Iteration Auf dem Mockup stellen die Elemente 1 und 2 die Basiselemente dar und wurden in der ersten Iteration entworfen. Zunächst stellt das Element **1** ein *ButtonsBar*, das die Schaltflächen zur Eingabe der Daten und zum Start des Prozesses enthält. Die *Browse* und *Gitlab Connection* Tasten dienen jeweils zur Auswahl der Registrierungsdatei und zur Eingabe der GitLabzugangsdaten. Mit der *Start* Taste kann den Registrierungsprozess gestartet werden. Weiterhin steht das Element **2** für einen *Logger*, in dem relevante Prozessinformationen in Echtzeit geloggt werden. Im Laufe des Prozesses muss immer automatisch nach unten gescrollt werden, so dass der Administrator immer die neuesten Ankündigungen sieht.

2.Iteration Die zweite Iteration befasst sich mit der Logik der Registrierungsprozess und betrifft nicht die Gestaltung von der Hauptseite des Plugins.

3.Iteration Auf dem Mockup stellen die Elemente 3 und 4 die erweiterten Elemente dar und wurden in der dritten Iteration entworfen. Zunächst

zeigt das Element **3** ein *ProgressBar* dar. Dies dient dazu, dass den Stand des Prozesses jederzeit sichtbar und einfach zu verfolgen ist. Schließlich repräsentiert das Element **4** einen *Interaktionsbereich*, auf dem Tabellen mit ungültigen Einträgen gezeigt wurden. Der Administrator kann dann bestimmte Aktionen auf den Einträgen durchführen.

3.3 Modellierung und Entwurf des Lösungsansatzes

Nach der Phase der Problemerkennung, der Äußerung von funktionellen und nicht-funktionellen Anforderungen und der Lösungsdefinition, wird nun die Phase des Lösungsentwurfs und der Modellierung eingetreten. Der konzeptionelle Entwurf besteht darin, das Registrierungsplugin zu gestalten und ihm eine Form und Architektur zu geben, die allen in der Analysephase geäußerten und formulierten Bedürfnissen und Anforderungen entspricht. In diesem Teil wird die UML-Sprache als grafische Modulationssprache verwendet.
Dieser Entwurf besteht aus zwei Phasen, wobei die erste Phase die allgemeine oder auch vorläufige Phase ist. In dieser werden die verschiedenen Komponenten der Lösung modelliert [9] und in ihrer Systemumgebung definiertAbbildung 3.5. Dieses ermöglicht im Anschluss die Struktur jeder Komponente im Detail verfolgen zu können. In dieser ersten Phase werden auch die Aktivitätsdiagramme verwendet, um die Aktivitäten der Lösung mit einer zeitlichen Dimension zu modellieren. Damit können die verschiedenen Phasen der Ausführung des Registrierungsprozesses, die die Lösung ausmachen, aufgezeigt werden. Dieser Ansatz ermöglicht es , die verschiedenen Interaktionen zwischen den Komponenten der Lösung im Detail zu konzipieren.
Dieses führt nun zur zweiten Entwurfsphase. Wie im vorherigen Absatz erwähnt, wird dabei die Struktur der Komponenten im Detail entworfen. Dafür wurden Klassendiagramme ausgewählt, um den JAVA-Teil der Lösung zu modellieren. Außerdem werden Sequenzdiagramme verwendet, um die Interaktionen zwischen den Komponenten der Lösung zu entwerfen. Jede Phase hat einen statischen Aspekt, der die Komponenten und ihre Struktur definiert, und einen dynamischen Aspekt, der die Schritte des Registrierungsprozesses definiert und diese in die Form von Interaktionen zwischen den Komponenten übersetzt.

3.3.1 Phase 1: Allgemeine Konzeptionierung

In der allgemeinen Konzeptionierung werden die verschiedenen Komponenten, die zur Funktionsweise des Registrierungsplugins gehören, konzipiert, ohne im Detail auf deren Zusammenspiel einzugehen. Darüber hinaus werden die verschiedenen Schritte des Registrierungsprozesses anhand von Aktivitätsdiagrammen dargestellt.

Architekturaufbau

die Abbildung 3.5 stellt ein UML-Komponentendiagramm dar. Das Diagramm zeigt die verschiedenen Komponenten, die für das Funktionieren des Registrierungsplugins eine Rolle spielen, sowie die Schnittstellen, die sie verbinden.
Grundsätzlich ist das Registrierungsplugin eine Webanwendung. Webanwendungen bestehen in der regel aus einem Frontend sowie einem Backend. Der Backend-Teil des Registrierungsplugins enthält die *Webwork Action* des Registrierungsplugins und die definierten REST-Ressourcen, die zur Erfüllung des zwecks des Plugins benötigt werden. Die *Registration REST Resource*(RRS) wird für den Prozessablauf notwendige Funktionalitäten realisieren. Diese Funktionalitäten sind die Erstellung von *Notification schemes*, die für die Erstellung eines Jira-Projekts notwendig sind, aber nicht durch den REST Ressourcen von Jira zur Verfügung gestellt wurden, das Speichern ungültiger Daten der Studierenden in der Datenbank und das Speichern von Daten des ausgeführten Registrierungsprozesses in der Datenbank. Der *Javascript-Controller* ruft die Funktionalitäten der RRS über REST-Aufrufe auf. Der Frontend-Teil des Plugins besteht aus einem *View* und einem JS-Controller. Der JS-Controller führt alle Vorprozess-Validierungen der Daten durch und ruft die REST Ressourcen von Jira und GitLab auf. Er definiert den Aktionsfluss entsprechend der vom Server erhaltenen Rückmeldungen. Der JS-Controller aktualisiert auch den *HTML-View*, was ein Echtzeit-Feedback des Prozesses garantiert. Außerdem dynamisiert er den *HTML-View*, um den Admin verschiedene Interaktionen anzubieten. Die REST Ressourcen von sowohl Jira als auch GitLab werden durch den JS-Controller aufgerufen und nicht durch den Backend-teil des Plugins, um die kaskadierenden Aufrufe der REST Ressourcen zu vermeiden, was das Debuggen im Fehlerfall erschwert und die Kopplung zwischen den Komponenten stark macht. Einige Komponenten der Architektur, wie das *Webwork Action*, werden durch den von Atlassian Architektur auferlegt. Andere Komponenten existieren und werden bereits in dem Fachgebiet eingesetzt.

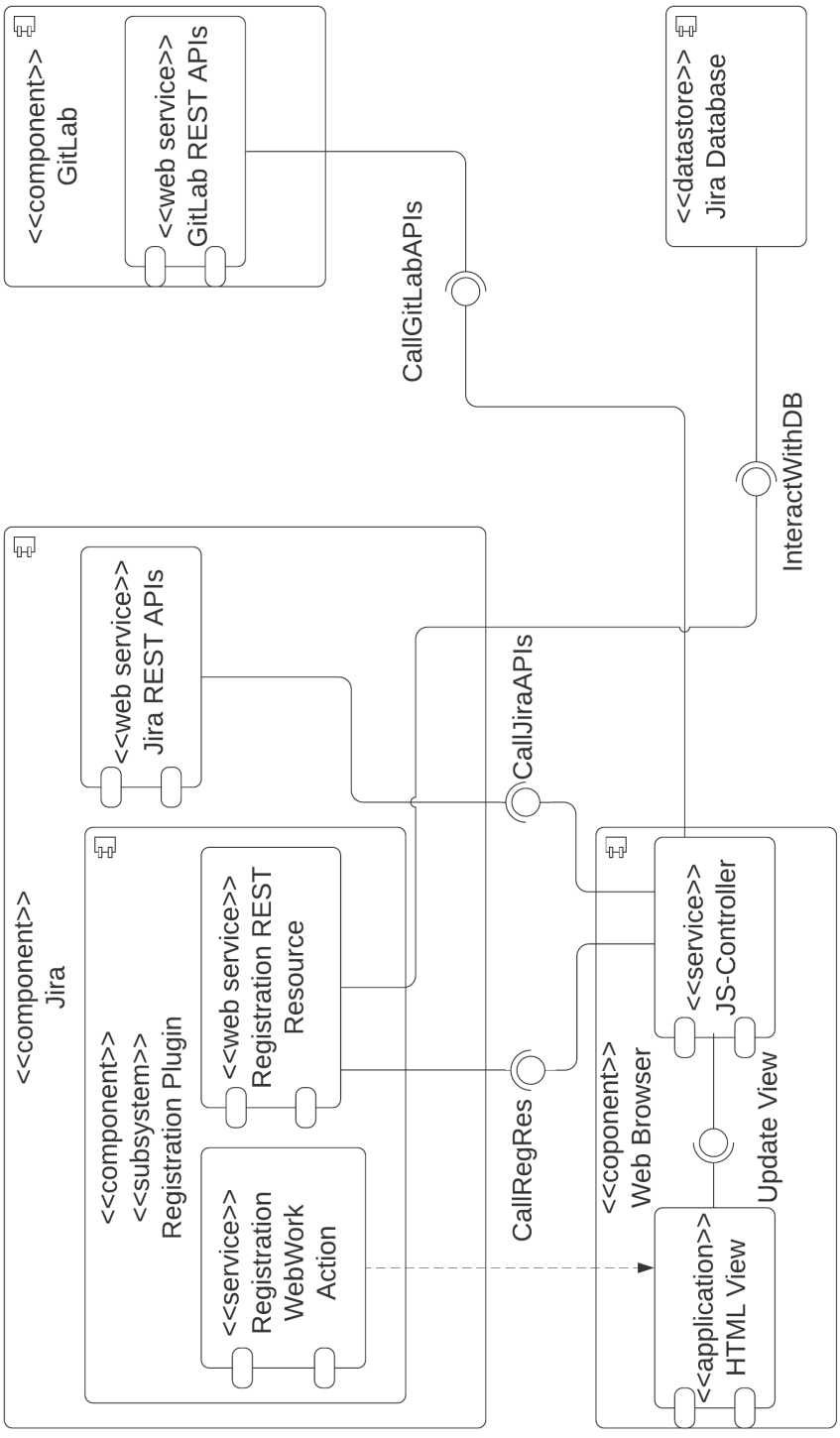


Abbildung 3.5: Registrierungsplugin Komponentendiagramm

Aktivitätsdiagramme des Registrierungsplugins

Das dynamische Modell stellt die Abläufe von Ereignissen, Zuständen und Reaktionen dar, die im System geschehen müssen. Es steht in engem Zusammenhang mit dem Objektmodell und beschreibt die Steuerungsaspekte eines Systems unter Berücksichtigung von Zeit, Abfolge von Operationen und Interaktionen zwischen Objekten. In der Entwurfsphase eignen sich Aktivitätsdiagramme besonders gut zur Beschreibung von Use Cases. Insbesondere veranschaulichen und verdichten sie die textliche Beschreibung von Use Cases und ihre Darstellung in Form von Flussdiagrammen, was sie leicht verständlich macht. Im folgenden wird sich auf die Aktivitäten aus der Sicht des Jira-Administrators konzentriert.
Das in Abbildung 3.6 dargestellte Aktivitätsdiagramm zeigt die Auslösung von dem erweiteten Registrierungsprozess, basierend auf Systemzuständen.

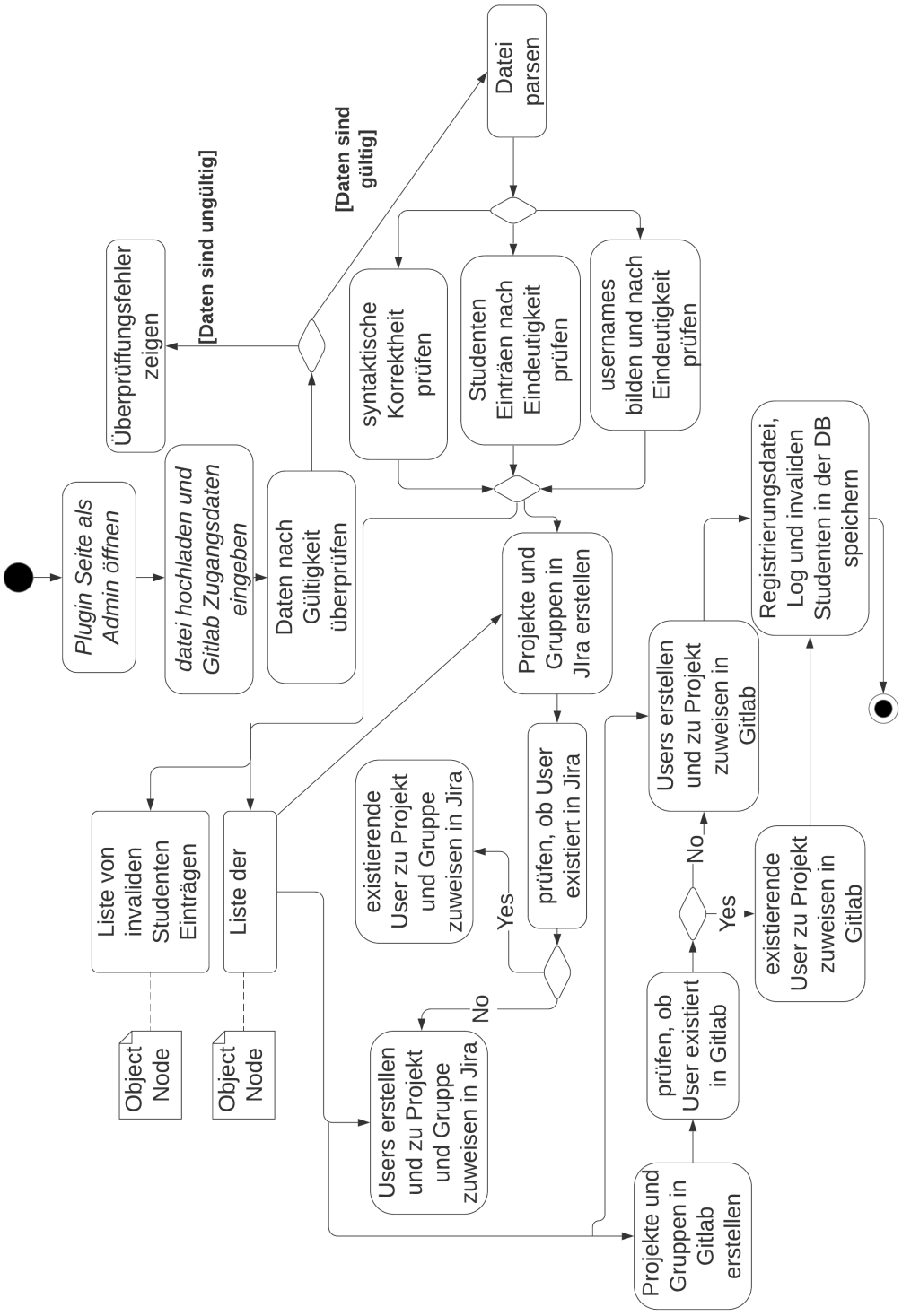


Abbildung 3.6: Aktivitätdiagramm: Erweiterter Registrierungsprozess

3.3.2 Phase 2: Detaillierte Konzeptionierung

In diesem Teil werden die JAVA-Klassen, ihre Methoden und Attribute in Form eines Klassendiagramms im Detail modelliert. Darüber hinaus wird anhand von Sequenzdiagrammen ein detaillierter Entwurf der Interaktionen zwischen den verschiedenen Komponenten, die beim Betrieb des Registrierungsplugins eine Rolle spielen, vorgenommen.

Klassendiagramm der Registration REST Resource

In diesem Teil wird die Komponente *Registration REST Resource* aus dem Komponentendiagramm Abbildung 3.5 im Detail dargestellt. Das Klassendiagramm Abbildung 3.7 ermöglicht es , die verschiedenen Klassen, Schnittstellen, ihre Attribute und Methoden sowie die Beziehungen zwischen ihnen zu veranschaulichen. Eine textuelle Interpretation des Diagramms wird in Form einer Tabelle Tabelle 3.4 bereitgestellt.

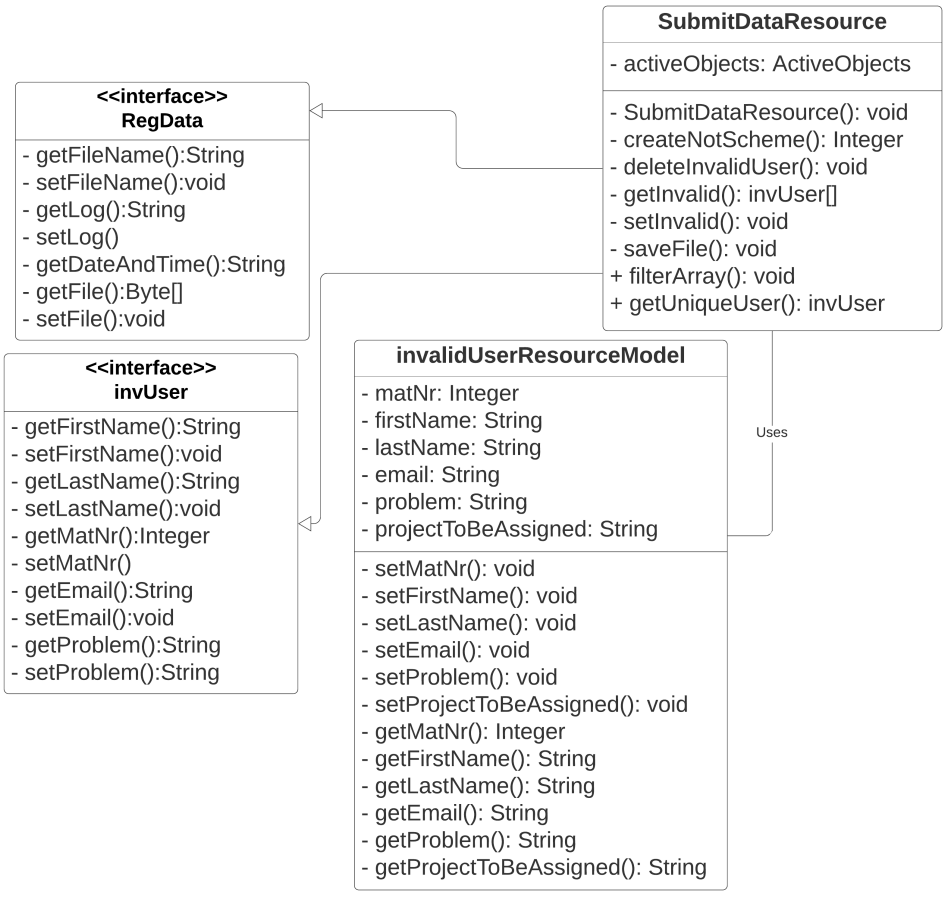


Abbildung 3.7: Klassendiagramm: Plugin Registration REST Ressource

Tabelle 3.4: Klassendiagramm Beschreibung

Klasse	**Beschreibung**
RegData	repräsentiert die Tabelle der Registrierungsdaten in der Datenbank
invUser	repräsentiert die Tabelle der invaliden Studenten in der Datenbank
SubmitData-Resource	repräsentiert die benögtige Methoden für die Registration REST Rresource
InvalidUser-ResourceModel	repräsentiert, wie ein invalider Student aus der Datenbank abgerufen werden sollte

Squenzsdiagramme

In diesem Abschnitt werden die Sequenzdiagramme als dynamische Diagramme vorgestellt. Diese Diagramme stellen die Interaktion zwischen Akteuren und Komponenten durch die verschiedenen logischen Schichten dar Die Abbildung 3.8 stellt ein Sequenzdiagramm für den Basis Registrierungsprozess sowie die benötigten REST-Aufrufe für die Erstellung von Jira und GitLab Objekten.

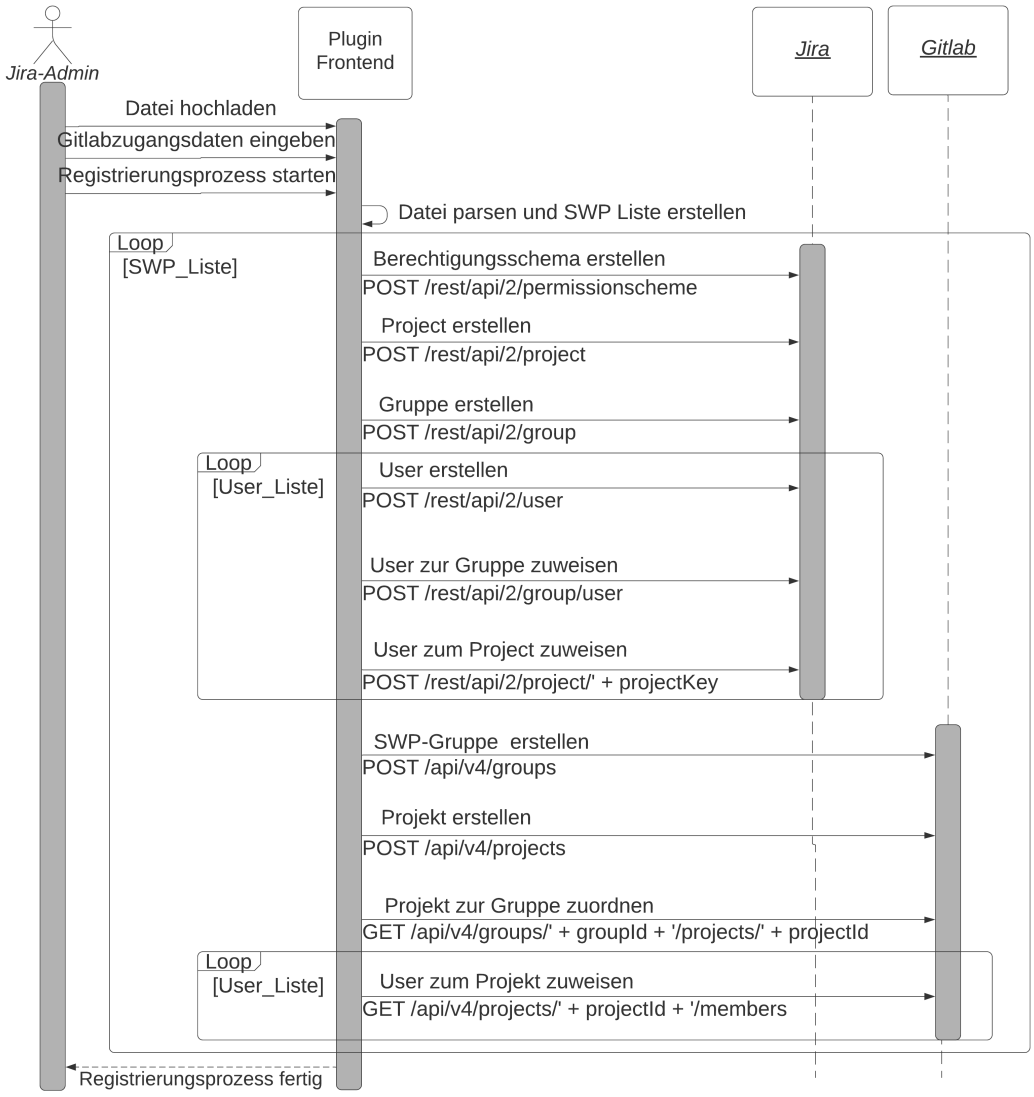


Abbildung 3.8: Sequenzdiagramm: BasisRegistrierungsprozess

Kapitel 4

Implementierung

In diesem Kapitel wird die Realisierung des Plugins befasst. Zunächst werden die eingesetzten Technologien beschrieben. Danach wird die Implementierung des im vorherigen Kapitel vorgestellten Konzepts in zwei Abschnitten untersucht, nämlich Backend und Frontend.

4.1 Eingesetzte Technologien

Bei der Implementierung vom Plugin wurden verschiedene Technologien verwendet. In diesem teil werden sie erläutert.

4.1.1 Atlassian Plugin SDK 6.3.10

Das Plugin SDK von Atlassian verfügt über zahlreiche Funktionalitäten und Schnittstellen. Davon sind für die Entwicklung des Plugin das Projektmanagement-Tool *Maven* und der integrierte Server *Apache Tomcat* wesentlich. Das SDK wurde mit JAVA geschrieben und verwendet die Funktionen vom Spring Framework. Es besitzt verschiedene JAVA Schnittstellen, mit denen die Entwickler Jira Objekte manipulieren können, z.B. die Erstellung von einem Projekt oder das Löschen von einem Nutzer.

4.1.2 Representational State Transfer

Bei der Entwicklung einer API für eine Webanwendung müssen einige Einschränkungen beachtet werden. Eine Variation dieser Einschränkungen wird als *Representational State Transfer* oder REST bezeichnet[11]. Es handelt sich dabei um einen populäreren Standard zur Entwicklung von APIs, um eine Verbindung zwischen Servern und Clients herzustellen. Ein wichtiger Aspekt von REST ist das *URL-Mapping*, ein Beispiel dafür wäre, wenn Daten vom Server nachgefragt sind (wird durch *Rquest* bezeichnet), muss eine entsprechende URL existieren, um die nachgefragte Daten zu

liefern (wird durch *Response* bezeichnet). Ein Request setzt sich aus vier Elementen zusammen:

- Der angefragten Endpunkt (abgerufene URL).
- Die verwendete Methode, die den Typ der Request definiert. Die häufig benutzten sind GET, POST, PUT und DELETE. In dieser Arbeit wurden GET Requests verwendet, um Daten vom server zu erhalten, POST Requests, um Daten an den Server zu senden und DELETE Requests, um einige Daten aus der Datenbank zu löschen.
- Der Körper von dem Request, der die zugehörige Request-Daten für die Anfrage enthalten kann (nur für POST-, PUT- und Delete-requests). Diese Daten können auch in der URL übertragen(GET Requests)und unter verschiedenen Formaten gesendet werden, z.B. text, XML, JSON, binär, etc. Der Typ der Datenkodierung wird in den *Request headers* angegeben.
- Die Headers, die die wichtigen Informationen über den Nutzer und den Server enthalten, z.B. Authentifizierungsinformationen oder den Typ von Datenkodierung.

Nach der Verarbeitung der Request sendet der Server dann sein Response an den Nutzer. Das Response enthält auch seine eigenen Headers und seinen Körper. Der Status des Response wird durch einen HTTP-Statuscode definiert, der in der Regel einer der folgenden sein kann:

- 2XX: impliziert, dass die Verarbeitung des Request erfolgreich war, und das passende Response wurde gesendet.
- 3XX: bedeutet, dass das nachgefragte Request zu einer anderen URL umgeleitet wurde und dort verarbeitet wird.
- 4XX: impliziert, dass die Verarbeitung des Request wegen einen Fehler an der Nutzerseite nicht erfolgreich war.
- 5XX: impliziert, dass die Verarbeitung des Request wegen einen Fehler an der Serverseite nicht erfolgreich war.

In der implementierung des Registrierungsplugins wurden Funktionen definiert, die die REST-Ressourcen von Jira und GitLab aufrufen. Die Abbildung 4.1 zeigt die Funktion *createUserInJira*. Diese Funktion ruft die entsprechende URL zur Erstellung von einem neuen User in Jira auf.

```
var createUserInJira = function (user) {
    var toBeReturned = {};
    var toBeCreated = {name: user.username, password: "", emailAddress: user.email,
        displayName: user.firstName, notification: true, applicationKeys: ["jira-software"]};
            AJS.$.ajax({
                async: false,
                contentType: 'application/json',
                url: baseUrl + "/rest/api/2/user",
                type: "POST",
                dataType: 'json',
                data: JSON.stringify(toBeCreated),
                success: function (data) {
                    toBeReturned =  {name: data.name, key: data.key};
                },
                error: function (error) {
                    toBeReturned = {error: error};
                }
            })
    return toBeReturned;
};
```

Abbildung 4.1: Aufruf zur Erstellung von einem neuen User

Die für der Aufruf verwendete Methode ist *POST* und die zu versendenden Daten sind vom Typ *JSON*. Nach der Ausführung des Aufrufs wird die Rückmeldung entweder ein Erfolg oder ein Misserfolg sein. Die *success* Methode impliziert, dass der Aufruf erfolgreich ausgeführt wurde. Die *error* Methode impliziert, dass der Aufruf nicht erfolgreich war und der Statuscode des *Response* wird dann untersucht. Die Abbildung 4.2 zeigt die Einstellung der angewendten Methode, den URL-Pfad und den Typ der Daten, die beim Aufruf der zugeordnete URL konsumiert und von der implementierten REST-Ressource erzeugt werden sollen.

4.1.3 JavaScript Object Notation

Abgekürzt JSON[1], diese Datenstruktur wird verwendet, um Javascript-Objekte in reinen Text umzuwandeln. JSON-Objekte können jede Art von Daten enthalten, sowohl primitive als auch komplexe. Sie werden oft bei der Entwicklung von Webanwendungen verwendet, da sie leicht und schnell zwischen Servern und Clients übertragen werden können. Der Hauptvorteil dieser Objekte besteht darin, dass sie problemlos im geeigneten Datenformat der Zielprogrammiersprache interpretierbar sind. Die Syntax ist minimalistisch und wird immer in der Form einer Eigenschaft und seinem Wert geschrieben.

```
@POST
@Path("/saveFile")
@AnonymousAllowed
@Produces({MediaType.APPLICATION_JSON})
@Consumes({MediaType.APPLICATION_JSON})
public Response saveFile (@Context HttpServletRequest request) throws IOException, ServletException {
```

Abbildung 4.2: Verwendung von JSON in der Implementierung der REST Ressource

die REST-APIs sowohl von Jira als auch von GitLab verwenden JSON als Datentyp, der in den Körper der *Requests* und *Responses* aufgenommen werden soll. Beim Aufruf einer REST-Ressource vom Typ POST müssen die zu sendenden Daten zunächst in ein JSON-Objekt transformiert werden, andernfalls antwortet der Server mit einem *Content-Type* Fehlermeldung. Die implementierte REST-Ressource des Registrierungsplugins verwendet ebenfalls JSON-Objekte, um Daten zu senden und zu empfangen. Die Abbildung 4.2 zeigt, dass die Methode *saveFile* ein JSON-Objekt als Input nehmen muss und ein JSON-Objekt erzeugt, das in der *Response* gesendet wird.

4.2 Backend

In dem Plugin-Backend wurde die REST Ressource implementiert, die notwendig ist, damit das Plugin seinen Zweck erfüllen kann.

4.2.1 Implementierung des WebWork Action

Das *WebWork Action* ist eine JAVA-Servlet, das die Hauptseite des Plugins lädt, wenn die zugehörige URL von einem Jira Admin abgerufen wird. Die URL, auf der das Servlet reagiert, lautet:
https://host/jira/secure/RegistrationDataSubmitWebworkAction.jspa
Wenn diese URL von einem normalen Benutzer(also kein Jira Admin) abgerufen wird, liefert Jira die Fehlermeldung 'Seite wurde nicht gefunden'.

4.2.2 Realisierung der Plugin REST Ressource

Obwohl Jira zahlreiche REST Ressourcen anbietet, müssen noch einige spezielle Ressourcen implementiert werden, um die Bedürfnisse des Plugins zu erfüllen. Das Atlassian SDK bietet JAVA-Bibliotheken zur Unterstützung der Entwicklung von REST Ressourcen.
Das *SubmitDataResource* (SDR) wurde entwickelt, um die Anforderungen des Registrierung Plugins zu erfüllen. Das SDR stellt fünf Methoden zur Verfügung, die auf bestimmte HTTP *Requests* von dem Plugin JS-Controller reagieren. Die entwickelten Methoden sind die folgenden:

- ***saveFile***: die URL lautet:
  https://host/jira/rest/submitdata/1.0/saveFile
  Diese Methode wird verwendet, um die Registrierungsdatei und den erzeugten Log in der Datenbank zu speichern. Die Datei und der Log müssen im Körper vom *Request* enthalten sein. Das SDR wird im Erfolgsfall ein *Response* mit dem Statuscode 201 senden.

- ***saveInvalid***: die URL lautet:
  https://host/jira/rest/submitdata/1.0/saveInvalid
  Diese Methode wird verwendet, um die Liste der invaliden Studierenden in der Datenbank zu speichern. Diese Liste wird in Form eines JSON Objekts im Körper vom *Request* enthalten sein. Das SDR wird im Erfolgsfall ein *Response* mit dem Statuscode 201 senden.

- ***getInvalid***: die URL lautet:
  https://host/jira/rest/submitdata/1.0/getInvalid
  Diese Methode wird verwendet, um die Liste der invaliden Studierenden aus der Datenbank zu filtern. Das SDR wird im Erfolgsfall ein *Response* mit dem Statuscode 200 senden. Das *Response* wird die Liste in Form von einem JSON-Objekt bereitstellen.

- ***deleteInvalidUser***: die URL lautet:
  https://host/jira/rest/submitdata/1.0/deleteInvalid/<matNr>
  Diese Methode wird verwendet, um einen invaliden Studierenden aus der Liste von invaliden Studierenden in der Datenbank zu löschen. Der Eintrag, der *matNr* als ID hat, wird gelöscht. Das SDR wird im Erfolgsfall ein *Response* mit dem Statuscode 200 senden.

- ***createNotScheme***: die URL lautet:
  https://host/jira/rest/submitdata/1.0/getNotScheme/<schemeName>
  Diese Methode wird verwendet, um ein Benachrichtigungsschema mit *schemeName* als Name zu erstellen. Das SDR wird im Erfolgsfall ein *Response* mit dem Statuscode 200 senden. Das *Response* wird die ID von der Schema in Form von einem JSON-Objekt bereitstellen.

Bei Invaliden *Requests* wird das SDR ein *Response* mit dem Statuscode 400 senden.

4.2.3 Interaktion mit der Datenbank durch Active Objects

Die *Active Objects*-Schnittstelle ist ein integriertes Atlassian-Plugin und stellt eine Reihe von Funktionen, die die Interaktion mit der DB ermöglichen, bereit. Die zuvor beschriebene implementierte REST Ressource Unterabschnitt 4.2.2 setzt Atlassian *Active Objects* zur Durchführung von Operationen mit der Datenbank ein. Bei der Verwendung von *Active Objects* erfolgt das Aktualisieren des DB-Schemas automatisch beim Einsatz

des Plugins im Jira System. Die Felder der Tabelle werden durch die Definition einer JAVA-Interface festgelegt, die die *Getters* und *Setters* der Tabellenfelder definiert. Bei der Implementierung von der Plugin REST Ressource Unterabschnitt 4.2.2 müssen zwei Tabellen erstellt werden.
Die erste Tabelle heißt *INV-USER* und wird die ungültigen Studierendeneinträge enthalten. Die *JAVA-Interface*, die diese Tabelle beschreibt, wurde in der Abbildung 3.7 vorgestellt. Zur Identifizierung von Einträgen wurde die Matrikulnummer verwendet.
Die zweite Tabelle heißt *REG-DATA*. Diese Tabelle speichert die Informationen der ausgeführten Registrierungsprozesse. Die *JAVA-Interface*, die diese Tabelle beschreibt, wurde ebenfalls in der Abbildung 3.7 vorgestellt.

4.3 Frontend

Das Frontend des Plugin besteht aus der Plugin-Hauptseite und dem JS-Controller. Im Folgenden wird die Implementierung des Frontends erläutert.

4.3.1 Javascript Controller

Der JS-Controller führt den großen Teil der Logik hinter dem Registrierung Plugin aus, einschließlich der Validierung der Daten, der Einrichtung der SWPs und ihrer Erstellung in Jira und GitLab durch REST-Aufrufe. Er aktualisiert auch die *View* entsprechend und dynamisiert die Plugin-Hauptseite, so dass der Admin mit dem Registrierungsprozess interagieren kann.

Validierung der Daten

Eine der Hauptfunktionalitäten des JS-Controllers ist die Validierung der Daten und die Einrichtung der SWP-Objekte. Die Validierung erfolgt in mehreren Schritten. Nach dem Extrahieren der Daten aus der Registrierungsdatei wird eine Reihe von Überprüfungen durchgeführt, um die ungültigen Daten zu isolieren. Die ungültigen Studenteneinträgen werden in einer separaten Liste gespeichert. JS bietet nützliche Funktionalitäten, um diese Art von Verarbeitungen zu leisten. Beispielsweise bietet JS *Regular expressions*[13] eine einfache Möglichkeit, die syntaktische Korrektheit der Daten durch übereinstimmende Zeichenkombinationen in Zeichenketten zu testen.

```
552 var verifyName = function (name) {
553     var reg = /^[A-Za-züöäß]+$/;
554     return reg.test(name);
555 };
```

Abbildung 4.3: Namensüberprüfung durch JS *Regular expressions*

Die Abbildung 4.3 beschreibt die Implementierung einer Namensüberprüfung. Nach diesem Beispiel darf der Name nur aus Buchstaben und Umlauten bestehen. Die Methode *test* prüft, ob der gegebene Name mit den Zeichenkombinationen übereinstimmt, und gibt dann ein boolean zurück. Das gleiche Prinzip kann angewendet werden, um andere Daten zu überprüfen, wie Emails und Matrikulnummer. Zudem werden doppelte Studenten Emails und Matrikelnummern als ungültig betrachtet.

Einrichten von Projekten

Nachdem die Daten validiert wurden, beginnt die Einrichtung von den SWP-Objekten. Ein SWP enthält den Projektnamen, einen Projektschlüssel und eine Liste von Studierenden, die zum Projekt gehören. Die usernames für die Studierenden wurden ebenfalls erstellt und getestet, ob sie bereits in Jira oder GitLab schon existieren. Der Standard username besteht aus dem ersten Buchstaben des Vornamens und dem kompletten Nachnamen.
Beispielsweise wäre für **max mustermann**der username **mmustermann**
Mit dem REST Ressourcen von Jira und GitLab kann einfach eine *Request* mit dem username gesendet werden, um zu prüfen, ob dieser bereits existiert. Diese URLs werden dazu verwendet:

- In Jira: *https://host/jira/rest/api/2/user?username=<username>*
  Wenn der username schon existiert, würde ein *Response* mit dem Statuscode 200 erhalten, ansonsten mit dem Statuscode 404.

- In Gitlab: *https://host/api/v4/users?search=<username>*
  Gleiches Verhalten wie Jira.

Sollte der username bereits in Jira oder GitLab vorhanden sein, so wird dieser wie folgt geändert:

- **mmustermann: mmustermann<i><SWP-Jahr>**

Das **i** bezeichnet hierbei eine Generatorzahl, die bei eins beginnt. Das **SWP-Jahr** steht für die letzten zwei Ziffern eines Jahres.
Am Ende dieses Schrittes wird dann eine Liste mit den SWP-Objekten bereitgestellt.

Erstellung von SWPs in Jira und Gitlab

In diesem Schritt sind die SWP Objekte bereits eingerichtet und bereit, in Jira und GitLab erstellt zu werden. Zunächst wurden die SWPs in Jira und danach in GitLab erstellt. Die Liste der SWPs wurde über jeden SWP iteriert. In Jira und GitLab wird anhand REST Ressourcen getestet, ob bereits Projekte mit dem gleichen Namen existieren. In einem solchen Fall müssten dann diese Konflikte über die Benutzeroberfläche, die vom Plugin

angeboten wird, gelöst werden, bevor es fortgesetzt werden kann.
In Jira werden die Benachrichtigungen- und Berechtigungensschema erstellt und dem Projekt bei der Erstellung zugewiesen. Ebenso wird auch Eine Gruppe erstellt. Danach beginnt die Einrichtung von Studierendenkonten. Ein Studierendenkonto wird bei der Erstellung sowohl der Gruppe als auch dem Projekt zugewiesen. Bereits existierende Studierendenkonten werden direkt zugewiesen. Die Erstellung von diesen Objekten, bzw. die Zuweisung von Objekten untereinander, erfolgt über die Jira REST Ressourcen.
In GitLab wird zunächst eine Gruppe angelegt, die die zu erstellenden Projekten gruppiert. Danach wird das Projekt eingerichtet. Das Repository des Projekts wird mit einer Readme-Datei initialisiert. Zuletzt werden die Studierendenkonten erstellt und dem projekt zugewiesen. Ähnlich zu Jira werden existierende Studierendenkonten direkt dem projekt zugewiesen. Deaktivierte und gesperrte Konten werden aktiviert, bzw. entsperrt. Jeder zugewiesene Studierende erhält einen Aktivierungszeitraum in Höhe von sechs Monaten. Genau wie in Jira erfolgt alle dieser Aktionen über GitLab REST Ressourcen.

4.3.2 Realisierung der Benutzeroberfläche

Im vorherigen Kapitel wurden die Anforderungen zur Benutzeroberfläche erläutert. Daraus wurde ein Design für die Plugin Hauptseite abgeleitet. In diesem Abschnitt wurde die endgültige Implementierung der Hauptseite des Plugins erläutert.

Einsetzung von Soy Templates

Mit der Einsetzung von *Soy templates* [2] können dynamisch wiederverwendbarer HTML- und UI-Elemente generiert werden. Diese Vorlagen können sowohl serverseitig als auch nutzerseitig manipuliert werden. In der Implementierung von der Benutzeroberfläche werden diese auf der Nutzerseite eingesetzt, damit kann der JS-Controller HTML-Elemente im Laufe des Prozesses aktualisieren, einfügen und löschen kann. Ein wichtiger Aspekt dieser Vorlagen ist die hohe Modularität von Webseiten, d.h. eine Neugestaltung der Seite durch Änderung der Platzierung der Elemente oder das Hinzufügen neuer Elemente ohne Beeinträchtigung bereits vorhandener Elemente sind leicht durchführbar.
Für die Implementierung der Plugin-Hauptseite wurden zwei Soy templates verwendet. Die erste Vorlage enthält das *Buttonsbar*, den *Logger* Bereich und das *Progress Tracker*. Die zweite Vorlage enthält die Interaktionstabellen und die zugehörigen Popup Fenster zur Dateneingabe. die Elemente, mit denen die Vorlagen gefüllt werden, werden aus der Atlassian AUI-Bibliothek importiert, so dass eine Konsistenz zwischen der Plugin-Hauptseite und anderen Jira Seiten entsteht.

Implementierung der Plugin-Hauptseite

Die Abbildung 4.4 stellt dar, wie die Hauptseite gemäß das in der Analyse von der graphischen Oberfläche beschriebenen Design implementiert wurde:

Abbildung 4.4: Hauptseite des Plugins

1. im ersten Bereich ist der Pluginname im Pluginmenü zu sehen. Oben rechts ist sichtbar, dass sich das Plugin in dem Administrationsbereich befindet, und somit nur für Admins zugänglich ist.

2. im zweiten Bereich ist eine Sammlung von Tasten zu erkennen. In der Abbildung 4.4 steht *continue* anstelle vom *Start*, denn das Plugin wartet auf eine Admin Interaktion und kann danach den Prozess fortfahren. Die Gitlabszugangsdaten könene mit Hilfe eines Pop-up Fensters eingegeben werden, das beim Drücken der textitconfigure Gitlab Connection Taste auftaucht.

3. Der dritte Bereich ist der *Logger* Bereich, der durch ein HTML *textarea* realisiert wird. Darin werden alle Ereignisse vom Anfang bis zum Ende des Registrierungsprozesses erfasst. Im Folgenden sind Beispiele für die Protokollierung von verschiedenen Prozessereignissen zu sehen.

   - **9 August 2020, 19:32:5**
     assigning User lbaumann to group Project Example 6
   - **9 August 2020, 19:32:6**
     creating User Liam,Ludwig...
   - **9 August 2020, 19:32:13**
     creating permission scheme for Project Example 8

4. Der vierte Bereich zielt darauf ab, eine einfache Interaktion mit dem Admin zu ermöglichen, um die ungültigen Studenten- oder Projektdaten durch die Verwendung von Popup Fenstern zu korrigieren.

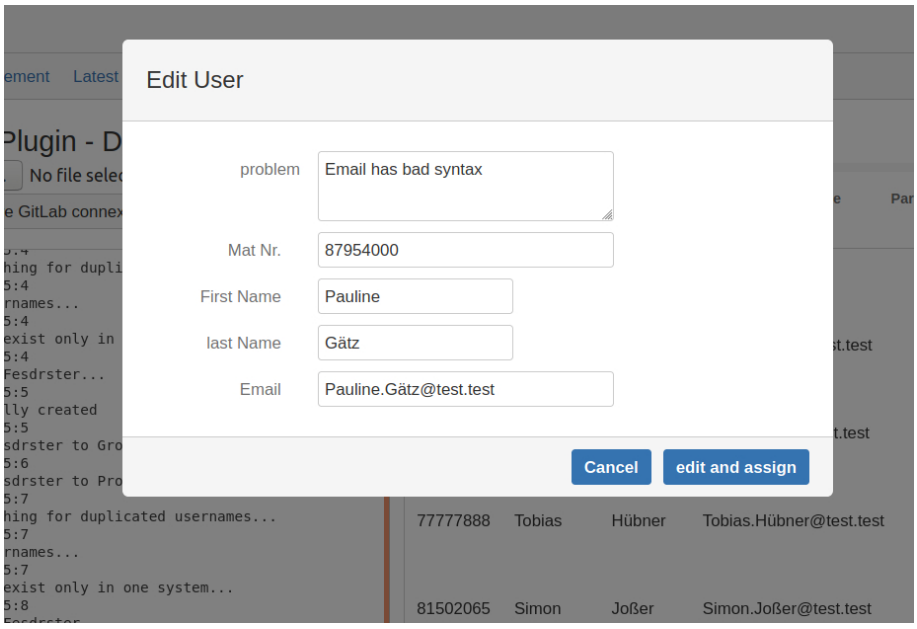


Abbildung 4.5: User editieren Pop-up Fenster

Die Abbildung 4.5 zeigt ein Pop-up-Fenster an, um die Daten eines ungültigen Students zu bearbeiten und ihn dem entsprechenden Projekt zuzuweisen. Nach der Modifizierung der Daten wird geprüft, ob die Daten immer ungültig sind. In dem Fall wird einen Fehlermeldung gezeigt.

5. Der fünfte Bereich handelt sich um ein *Progress Tracker*. Es unterteilt den Prozess in sechs verschiedenen Zuständen. Es wird sich aktualisieren, wenn ein Übergang zwischen den Zuständen stattfindet, damit hat der Admin einen kontinuierliches Feedback über den Prozessdurchlauf.

Kapitel 5

Validierung der Funktionalitäten

Nach der Implementierung des Registrierungsplugins, erfolgt nun eine Validierung der Funktionalitäten des Registrierungsplugins. Der erste Teil dieses Kapitels wird die Validierung der Funktionalitäten des Registrierungsplugins erläutert. Im zweiten Teil wird sich daher mit einem Belastungstest befassen, der die Ausführungszeit des Registrierungsplugins unter verschiedenen Variationen von Registrierungsdateien identifizieren soll.

5.1 Validierungstest des Registrierungsplugins

Um die Registrierungsplugin-Funktionalitäten zu validieren, muss eine finale Validierungstest durchgeführt werden. Das Ziel dieser Validierung besteht darin, die Fähigkeit des Registrierungsplugins zu überprüfen, verschiedene Arten von Registrierungsfehlern zu finden und zu behandeln, die absichtlich oder versehentlich von Testpersonen erzeugt wurden. Zwölf Probanden haben an der Validierung teilgenommen. Alle Probanden sind Informatik-Studierende im Bachelor. Das Alter der Testpersonen lag zwischen 20 und 25 Jahren. Die Testpersonen wurden über die Erstellung der Registrierungsdateien und die Anforderungen des Registrierungsplugins informiert. Die Testpersonen wurden gebeten Registrierungsdateien zu erstellen und zu versuchen, Fehler nach eigenem Ermessen einzufügen. Das Ziel ist Fehler zu provozieren, die das Registrierungsplugin nicht abdeckt und die während des Registrierungsprozesses auftreten können. Außerdem wurden den Testpersonen eine Liste von bereits existierenden Benutzerkonten und Projekten in Jira und GitLab zur Verfügung gestellt, so dass sie absichtlich einige Registrierungsfehler in ihre Dateien einbauen konnten. Auch die Fehler, die sich aus der syntaktischen Korrektheit von Matrikelnummern, Emails und Namen ergeben, wurden eingeführt.

Tabelle 5.1: Dokumentation der in den Dateien enthaltenen Fehler

Häufigkeit	**Fehler und Besonderheiten**	**Erwartetes Verhalten**	**Ergebniss**	**Kommentare**
12	Email ist ungültig	Eintrag vom Prozess ausnehmen	Erfolgreich	Eintrag steht am Ende des Prozess zum Editieren bereit
3	Projekt existiert schon in Jira	Plugin stellt existierendes Projekt zum Editieren bereit	Erfolgreich	Projekt kann gelöscht, genutzt oder editiert werden.
2	Projekt existiert schon in GitLab	Plugin stellt existierendes Projekt zum Editieren bereit	Erfolgreich	Projekt kann gelöscht, genutzt oder editiert werden.
6	Benutzername exitiert schon	Benutzername nach Muster ändern	Erfolgreich	
14	Student hat bereits ein Konto in Jira und Gitlab	Die existierende Konten werden benutzt	Erfolgreich	
18	Vor- und Nachname sind ungültig	Eintrag vom Prozess ausnehmen	Erfolgreich	Eintrag steht am Ende des Prozess zum Editieren bereit
5	Projekt existiert in Jira und Gitlab	Plugin stellt existierendes Projekt zum Editieren bereit	Erfolgreich	Projekt kann gelöscht, genutzt oder editiert werden.
2	Benutzer hat schon ein Konto in Jira, aber nicht in Gitlab	Ein Konto wird nur in Jira erstellt	Erfolgreich	Das Konto hat den gleichen Benutzernamen wie im Jira
4	Eine Gruppe mit dem selben Projektnamen existiert in Jira	Plugin stellt das invalide Projekt zum Editieren bereit	Erfolgreich	Gruppe kann gelöscht, genutzt oder editiert werden.
2	Benutzer ist in Gitlab blockiert	Benutzer wird entsperrt	Erfolgreich	
1	Ein Header fällt in der Datei aus	Fehlermeldung anzeigen	Erfolgreich	

Die Probanden haben unterschiedliche Registrierungsdateien mit verschiedenen Registrierungsfehlern zum Validierungstest erzeugt. Die Dateien umfassten 50 bis 100 Studierenden und 5 bis 10 Projekte. Sie haben auch eine seperate Textdatei bereitgestellt, die einen Bericht über den in der Registrierungsdatei verbauten Fehler enthielt. In diesem Test wurde das Verhalten des Plugins bei der Ausführung von Registrierungsprozessen mittels der bereitgestellten Registrierungsdateien beobachtet. Alle erwarteten Fehler und Besonderheiten wurden vom Plugin erkannt und korrekt behandelt. Die Tabelle 5.1 enthält einen Bericht über den Validierungstest. Sie besteht aus fünf Spalten. Die erste Spalte stellt die Häufigkeit von Fehlern in den Dateien dar, die zweite Spalte beschreibt den Fehler, der in dem Registrierungsprozess zu erwarten ist, die dritte Spalte stellt das vom Registrierungsplugin erwartete Verhalten zur Behandlung des Fehlers dar, die vierte Spalte gibt an, ob das erwartete Verhalten erfolgreich ausgeführt wurde und die letzte Spalte enthält Kommentare über Fehler oder Verhalten des Registrierungsplugins. Beispielsweise beschreibt die erste Zeile der Tabelle den Fehler 'ungültige Email'. Dieser Fehler trat zwölf Mal nach der Ausführung von allen Registrierungsprozessen auf. Die Behandlung dieses Fehlers besteht darin, den Studierendeneintrag mit der ungültigen Email aus dem Registrierungsprozess auszuschließen. Nach Beendigung des Prozesses können die Daten dann korrigiert werden.

5.2 Belastungstest des Registrierungsplugins

5.2.1 Angewendete methode

Um die Leistung des Plugins zu testen, wurde einen Ansatz gewählt, der auf Beobachtungen basiert. Anhand dieser kann festgestellt werden, ob sich Leistungsprobleme zeigen. Hierbei wurden mehrere Dateien mit unterschiedlichen Datenmengen verarbeitet. Für jede Datei wurde die Ausführungszeit von dem zugehörigen Registrierungsprozess gemessen. Die Verteilung der Daten in den einzelnen Dateien erfolgt in der Weise, dass die Anzahl der Studierenden im Verhältnis zur Anzahl der Projekte mit großen Abweichungen variiert. Die zu ausführenden Dateien enthalten keine Fehler, die zu einer Unterbrechung des Registrierungsprozesses führen würden, was die Ausführungszeit des Registrierungsprozesses beeinflussen würde. Der Registrierungsprozess kann dann von Anfang bis Ende ohne Unterbrechungen ablaufen. Es soll sichergestellt werden, dass die Ausführungszeit der Registrierungsprozessen im Verhältnis zur Anzahl der Projekte oder Studierenden nicht exponentiell ansteigt.

5.2.2 Erwartetes Ergebnis

Das Ziel ist, dass die Graphen, welche die Ausführungszeiten darstellen, einen linearen Verlauf aufzeigen. Das bedeutet, dass die gesamte Ausführungszeit der Datenverarbeitung das Ergebnis der Multiplikation einer Konstanten bleibt, die die Zeit der Verarbeitung einer Dateneinheit darstellt. Auf diese Weise kann festgestellt werden, ob das Registrierungsplugin in seinem Quellcode Algorithmen von exponentieller Komplexität enthält, die ein Leistungsproblem für das Plugin verursachen können. Der Belastungstest gilt als erfolgreich, wenn die Ausführungszeit linear ansteigt und als nicht erfolgreich, wenn sich die Ausführungszeit exponentiell erhöht.

5.2.3 Teststrategie

Die eingegebenen Registrierungsdaten sind von zwei Parametern abhängig, nämlich der Anzahl der Projekte und der Anzahl der Studierenden. Es wurden drei Belastungstests durchgeführt. Im ersten Test wurde die Anzahl der Projekte festgelegt und die der Studierenden erhöht, um Leistungsprobleme im Zusammenhang mit der Anzahl der Studierenden zu erkennen, die sich aus deren Erhöhung ergeben können. Beim zweiten Test wurde die Anzahl der Projekte erhöht und die Anzahl der Studierenden festgelegt, um Leistungsprobleme im Zusammenhang mit der Anzahl der Projekte zu erkennen. Beim letzten Test wurden beide Parameter erhöht, um sicherzustellen, dass es keine Leistungsprobleme bei der Ausführung von Registrierungsprozessen gibt.

Belastungstest 1:

In diesem Test wird die Anzahl der Projekte fixiert und die Anzahl der Studierenden erhöht. Bei Leistungsproblemen soll festgestellt werden, dass dieses Problem auf die Erhöhung der Studierendenzahl zurückzuführen ist und die Behandlung des Problems wird dann zielgerichtet sein. Tabelle 5.2.3 beschreibt die Zuteilung der Projekten und Studierenden in den Testdateien 1 bis 4.

Tabelle 5.2: Die Projekte und Studierende Zuteilung für Test 1

	Anzahl von Projekten	**Anzahl von Studierenden**
1	1	10
2	1	100
3	1	200
4	1	500

Die Abbildung 5.1 stellt die Ausführungszeit als eine Funktion der im Test verwendeten Datenmenge dar, die x-Achse stellt die Anzahl der Studierenden in der Datei dar und die y-Achse die Ausführungszeit aus dem mit der Registrierungsdatei ausgeführten Registrierungsprozess.

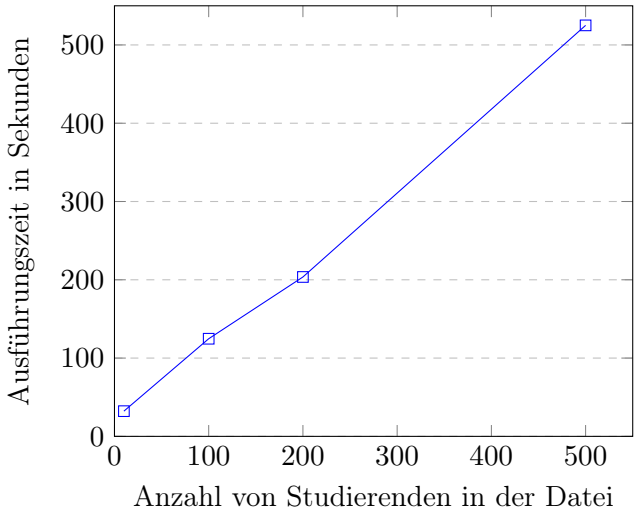


Abbildung 5.1: Ausführungszeiten in Abhängigkeit von der Anzahl der Studierenden

Die in der Abbildung 5.1 dargestellte Ausführungszeiten sind:

- Datei 1: 32.2 s
- Datei 2: 124.7 s
- Datei 3: 203.6 s
- Datei 4: 525 s

Beobachtungen

- es kann beobachtet werden, dass der Graph einen linearen Verlauf hat, was dem erwarteten Ergebnis entspricht. Daraus kann festgestellt werden, dass der Test erfolgreich war.

Belastungstest 2:

Im zweiten Test wird die Anzahl der Projekte erhöht und die Anzahl der Studierenden fixiert. Die Anzahl von Studierenden wird gleichmäßig auf die Projekten verteilt. Bei Leistungsproblemen soll festgestellt werden, dass dieses Problem auf die Erhöhung der Projektzahl zurückzuführen ist und die Behandlung des Problems wird dann zielgerichtet sein. Tabelle 5.2.3 beschreibt die Zuteilung der Projekte und Studierenden in den Testdateien 5 bis 8.

Tabelle 5.3: Die Projekte und Studierende Zuteilung für Test 2

	Anzahl von Projekten	**Anzahl von Studierenden**
5	10	500
6	100	500
7	200	500
8	500	500

Die Abbildung 5.2 stellt die Ausführungszeit als eine Funktion der im Test verwendeten Datenmenge dar, die x-Achse stellt die Anzahl der Projekte in der Datei dar und die y-Achse die Ausführungszeit aus dem mit der Registrierungsdatei ausgeführten Registrierungsprozess.

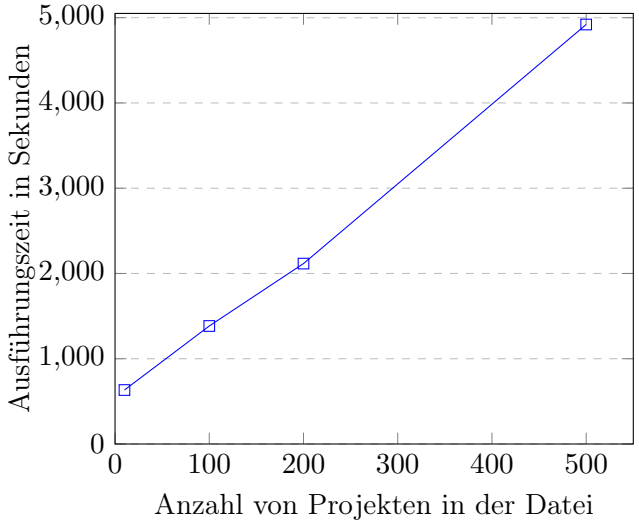


Abbildung 5.2: Ausführungszeiten in Abhängigkeit von der Anzahl der Projekte

Die in der Abbildung 5.2 dargestellte Ausführungszeiten sind:

- Datei 5: 633.8 s
- Datei 6: 1384.1 s
- Datei 7: 2116 s
- Datei 8: 4920 s

Beobachtungen

- es kann beobachtet werden, dass der Graph einen linearen Verlauf hat, was dem erwarteten Ergebnis entspricht. Daraus kann festgestellt werden, dass der Test erfolgreich war.

- Es ist zu beachten, dass mit der Erhöhung der Anzahl der Projekte die Laufzeit des Plugins zunimmt. Diese Zunahme ist auf die Verarbeitung eines Projekts und seiner verwandten Objekte zurückzuführen.

Belastungstest 3:

Im dritten werden sowohl die Anzahl der Studierenden als auch die Anzahl der Projekte erhöht. Die Anzahl von Studierenden wird gleichmäßig auf die Projekten verteilt. Bei Leistungsproblemen soll festgestellt werden, dass dieses Problem auf die Erhöhung der Studierenden- und der Projektezahl zurückzuführen ist und die Behandlung des Problems wird dann zielgerichtet sein. Tabelle 5.2.3 beschreibt die Zuteilung der Projekte und Studierenden in den Testdateien 9 bis 12.

Tabelle 5.4: Die Projekte und Studierende Zuteilung für Test 1

	Anzahl von Projekten	**Anzahl von Studierenden**
9	1	10
10	10	200
11	20	400
12	50	1000

Die Abbildung 5.3 stellt die Ausführungszeit als eine Funktion der im Test verwendeten Datenmenge dar, die x-Achse stellt den Nummer der ausgeführten Datei dar und die y-Achse die Ausführungszeit aus dem mit der Registrierungsdatei ausgeführten Registrierungsprozess.

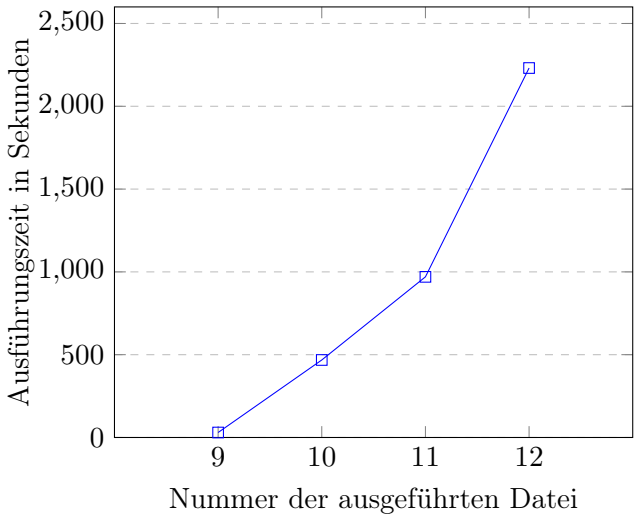


Abbildung 5.3: Ausführungszeiten in Abhängigkeit von der Anzahl der Studierenden und Projekte

Die in der Abbildung 5.3 dargestellte Ausführungszeiten sind:

- Datei 9: 30.8 s
- Datei 10: 468.4 s
- Datei 11: 970 s
- Datei 12: 2231.8 s

Beobachtungen

- es kann beobachtet werden, dass der Graph einen linearen Verlauf hat, was dem erwarteten Ergebnis entspricht. Daraus kann festgestellt werden, dass der Test erfolgreich war.

Anmerkungen:

- Es ist zu beachten, dass das Registrierungsplugin fünf REST-Aufrufe zur Bearbeitung eines einzelnen Studierendes und zwölf REST-Aufrufe zur Bearbeitung eines einzelnen Projekts ausführt. Dementsprechend sind die Leistung des Netzwerks und die Antwortzeit von Jira und GitLab kritische Faktoren für die Leistung des Plugins. Diese Tests wurden auf einem lokalen Rechner mit begrenzter Rechen- und Netzwerkkapazität durchgeführt, daher ist es deutlich, dass die überprüften Ausführungszeiten in der produktiven Umgebung schneller sein werden.

Vergleich mit der Ausführungszeit des aktuellen Prozesses

Die SWP aus dem Wintersemester 2020 umfasste insgesamt 17 Projekte und 152 Studierende. Die Ausführungszeit des Registrierungsprozesses betrug 4 Stunden und 48 Minuten. Es ist zu beachten, dass diese Ausführungszeit die manuellen Anpassungen einschließt, die von der SWP-Organisation vorgenommen werden müssen, um während des Registrierungsprozesses aufgetretene Konflikte zu lösen. Für einen Idealfall, der keine Fehler enthält, gibt es keinen direkten Bezug auf die Ausführungszeit des Registrierungsprozesses. Nach den durchgeführten Belastungstests kann davon ausgegangen werden, dass das Registrierungsplugin eine deutliche Verbesserung gegenüber dem eigentlichen Prozess aufweist.

Fazit

Aus den durchgeführten Belastungstests lassen sich einige Fakten ableiten. Aus den Graphen, die die Ausführungszeiten darstellen, ließ sich folgendes feststellen:

- Das Registrierungsplugin hat eine lineare Ausführungszeit.

- Das Registrierungsplugin hat keine Performance Probleme bei der Erhöhung der Anzahl von Studierenden oder Projekten
- Der Quellcode des Registrierungsplugins enthält keine Algorithmen, die zur Komplexitätsklasse EXP [8] gehören.
- Die Belastungstest wurden lokal durchgeführt. Es ist somit sicherzustellen, dass die Gesamtperformance des Registrierungsplugins im produktiven Umgebung besser sein wird.

Kapitel 6

Zusammenfassung und Ausblick

Dieses Kapitel dient dazu, einen abschließenden Überblick über die Inhalte und die wichtigsten Ergebnisse der Arbeit zu geben.

6.1 Zusammenfassung

Im Rahmen dieser Arbeit wurde ein Plugin für das Fachgebiet Software Engineering der Leibniz Universität Hannover sowohl konzipiert als auch implementiert, um den Registrierungsprozess für studentische Software-Projekte möglichst zu automatisieren. Das Plugin zielt darauf ab, die aktuellen Probleme der studentischen Registrierung zu lösen. Diese Probleme bestehen in erster Linie im Zeitaufwand, der für die Absolvierung des Registrierungsprozesses benötigt wird. Zusätzlich können unvollständige Registrierungen und Duplizierung von Studierendenkonten in Jira oder GitLab verursacht werden. Der vorgeschlagene Ansatz zur Lösung dieser Probleme war ein integriertes Jira-Plugin, das den Registrierungsprozess so weit wie möglich automatisieren sollte. Zunächst wurden die notwendigen Konzepte hinter dem Registrierungsprozess erläutert. Darunter der Ablauf des Registrierungsprozesses in Jira und GitLab. Das Konzept der Erweiterung der Jira-Funktionalitäten durch Plugins wurde ebenfalls untersucht.
Um das Registrierungsplugin realisieren zu können, musste zunächst hierfür ein vollständiges Konzept erstellt werden. Zu diesem Zweck wurde das iterative Modell eingesetzt, um eine systematische und inkrementelle Konzeption des Lösungsansatzes zu gewährleisten. Als Grundlage für diesen Schritt wurden die Anforderungen des Studierenden-Registrierungsprozesses gesammelt. Für diese Anforderungen wurden Use Cases erstellt, um sie in einer funktionalen Perspektive darzustellen. Außerdem wurde eine Benutzeroberfläche, die den Anforderungen der Benutzer entspricht, im laufe der Iterationen entworfen und verfeinert. Im weiteren Verlauf der Konzeptionierung wurde

dann der interne Aufbau des Registrierungsplugins erarbeitet und erläutert. Daraus wurde geschlossen, dass der JS-Controller des Plugins die Mehrheit der Logik hinter dem Registrierungsprozess ausführt. Gleichzeitig wurden die veschiedenen Registrierungsprozesseszenarien ausführlich untersucht. Die Konzeption des Plugins erfolgte schließlich durch die Verwendung verschiedener UML-Diagramme.
Im nächsten Schritt der Arbeit ging es um die Implementierung vom Plugin. Um diese zu erfüllen, wurden verschiedene Technologien eingesetzt, die identifiziert und erörtert wurden. Die Implementierung der Funktionalitäten unterteilt sich in zwei Bereiche, Frontend und Backend. Jeder Bereich wurde ebenfalls im Detail betrachtet und die implementierten Module dann erläutert. Bei einer detaillierten Betrachtung des implementierten Funktionalitäten war festzustellen, dass das entstandene Plugin sämtliche zu ermittelnden Anforderungen erfüllt.
Am Ende dieser Arbeit fand eine abschließende Validierung der Plugin-Funktionalitäten statt. Zuerst wurde die Leistung des Plugins unter Verwendung verschiedener Registrierungsdateien gemessen. Um zu verifizieren, dass das Plugin die geforderten Anforderungen erfüllt, wurde ein Validierungstest durchgeführt. Für diesen Test war die Hilfe von 12 Probanden erforderlich, die verschiedene Registrierungsdateien mit eingebauten Fehlern bereitgestellt haben. nach Durchführung des Tests wurden alle Fehler erfolgreich vom Plugin gefunden und behandelt.

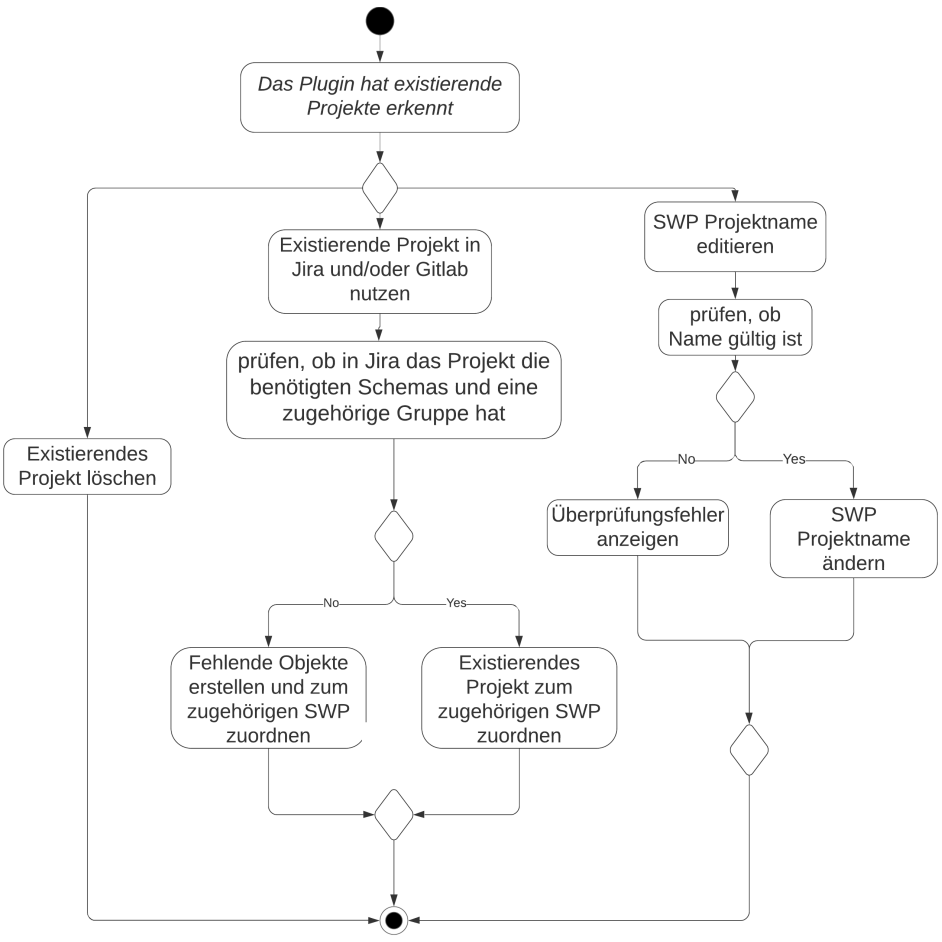
6.2 Ausblick

Im Rahmen einer zükunftigen Arbeit könnte das Plugin, um die im Folgenden beschriebenen Funktionen, erweitert werden.
Das Plugin könnte mit einem Modul zum Parsen von unterschiedlichen Registrierungsdatein ausgestattet werden, so dass das Plugin verschiedene Dateitypen, wie *xml* oder *docx*, unterstützt. Der Registrierungsprozess wird derzeit nur von einem Admin ausgeführt. Das bedeutet, um einen neuen Registrierungsprozess ausführen zu können, muss zunächst der vorherige Prozess abgeschlossen werden. Somit gibt es keine anhängigen, ungültigen Studierendeneinträge. Im Falle des Bestehens mehrerer Admins könnte das Plugin dahingehend erweitert werden, dass jeder Admin separate Registrierungsprozesse, unabhängig von anderen Admins, ausführen kann. Dadurch würden die ungültigen Daten der Studierenden nur für den zugehörigen Admin sichtbar sein und jeder Admin könnte einen Registrierungsprozess ausführen, solange er keine anhängigen ungültigen Studierenden hat.

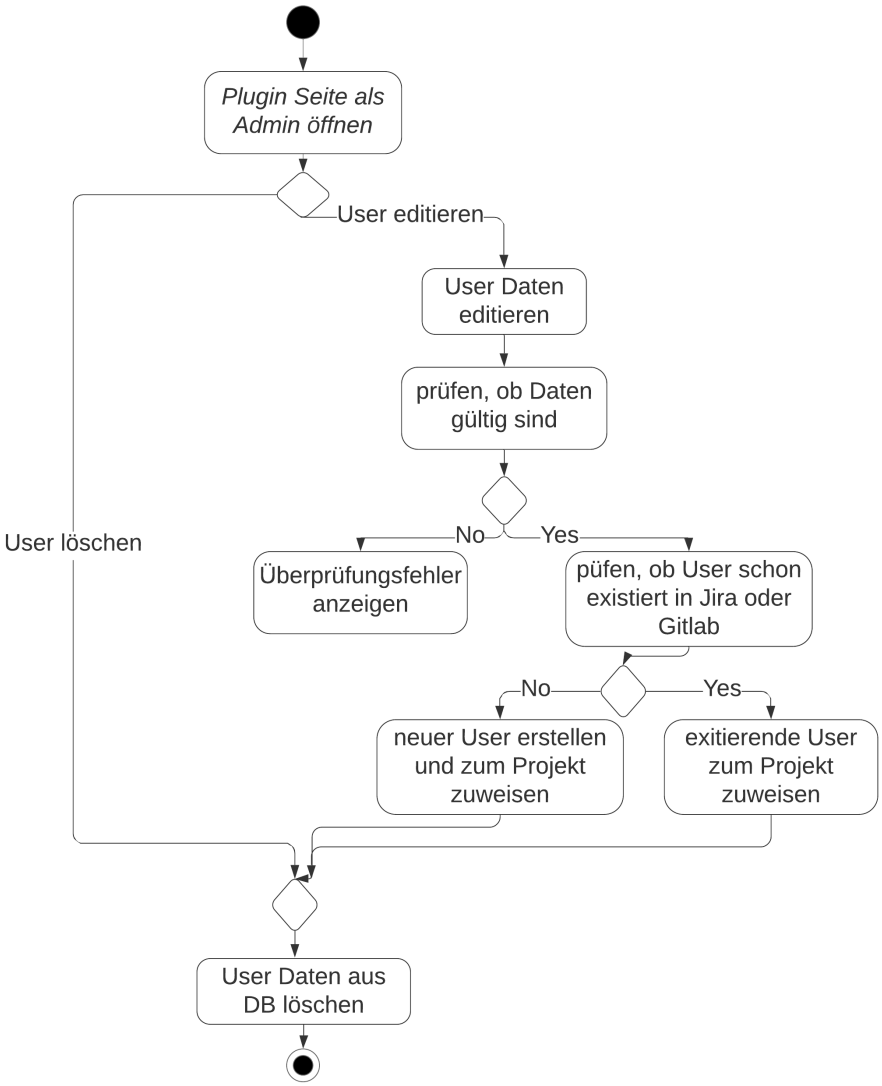
Anhang A

Aktivitätsdiagramme

A.1 Anstehende Projekte editieren



A.2 Invalide Daten der Studierende editieren



Anhang B

Sequenzdiagramme

B.1 Anstehende Projekte editieren

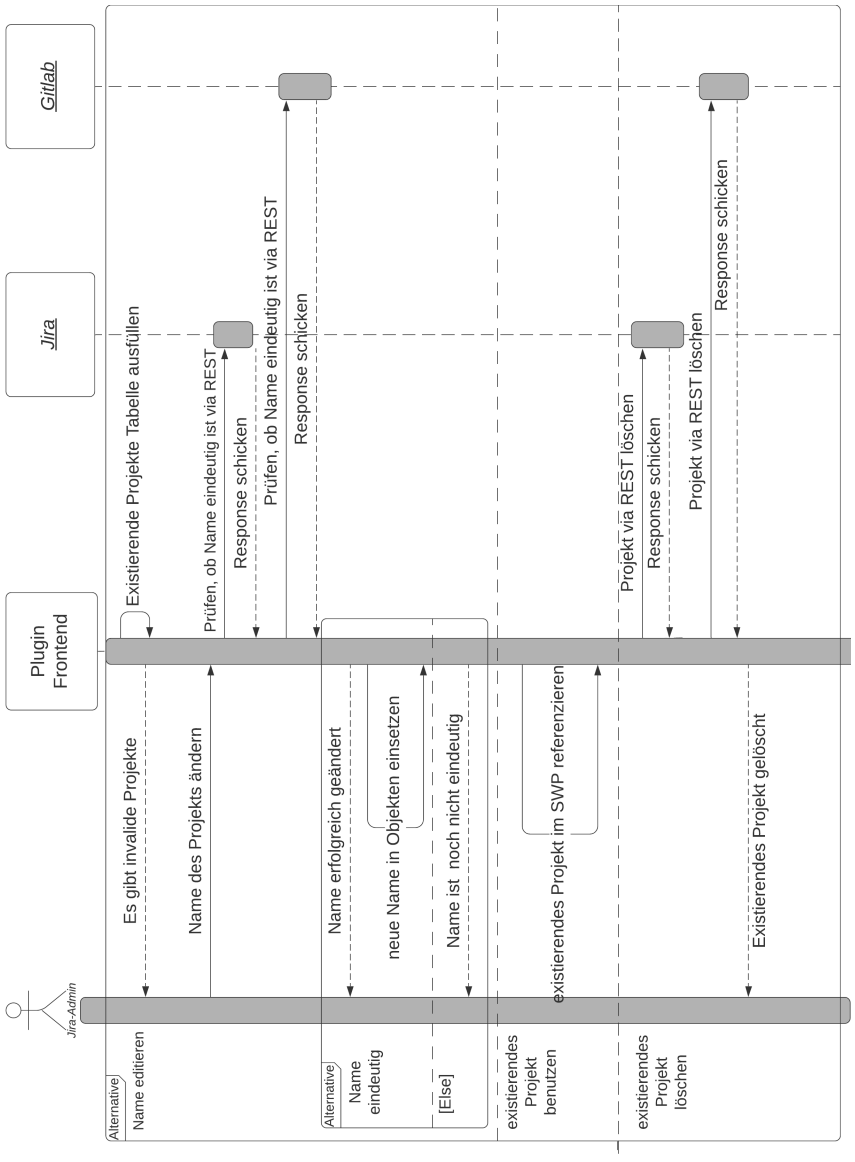


Abbildung B.1: Sequenzdiagramm: Anstehende Projekte auslösen

B.2 Invalide Daten der Studierende editieren

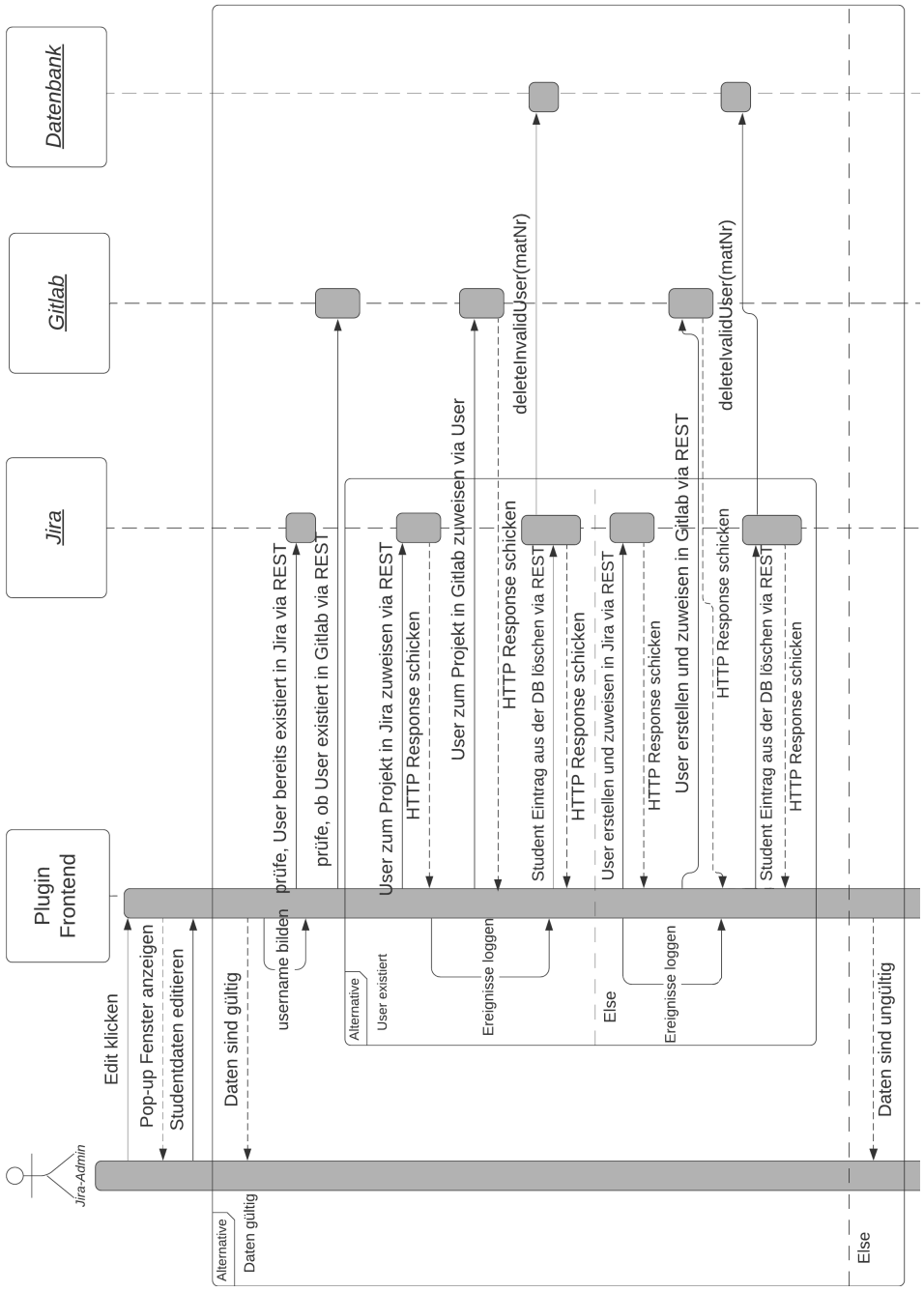


Abbildung B.2: Sequenzdiagramm Editieren von Invaliden Studenten

Literaturverzeichnis

[1] V. David. *Json: Main Principals*. CreateSpace Independent Publishing Platform, North Charleston, SC, USA, 2016.

[2] J. S. Developer. Using soy templates, 2019. [Online; letzter Zugriff 20.08.2020].

[3] M. Doar. *Practical JIRA Administration: Using JIRA Effectively: Beyond the Documentation*. O'Reilly Media, Inc., 2011.

[4] L. Gotter. Jira mit plugins erweitern – Überblick und möglichkeiten, 2015. [Online; letzter Zugriff 20.07.2020].

[5] J. M. Hethey. *GitLab Repository Management*. Packt Publishing Ltd, 2013.

[6] J. Kuruvilla. *JIRA Development Cookbook*. Packt Publishing Ltd, 2016.

[7] C. Larman and V. R. Basili. Iterative and incremental development: A brief history. 36(6), 2003.

[8] A. Meier, H. Vollmer, and U. Schöning. *Komplexität von Algorithmen: Mathematik für Anwendungen Band 4*. Mathematik für Anwendungen. Lehmanns, 2015.

[9] J. Osis and U. Donins, editors. *Bibliography*. Computer Science Reviews and Trends. Elsevier, Boston, 2017.

[10] N. Palani. *Selenium Webdriver: Software Automation Testing Secrets Revealed Part 2*. eBooks2go, Schaumburg, IL, 2016.

[11] L. Richardson and S. Ruby. *Restful Web Services*. O'Reilly, first edition, 2007.

[12] J. Rossberg. *Beginning Application Lifecycle Management*. Apress, USA, 1st edition, 2014.

[13] K. Thompson. Programming techniques: Regular expression search algorithm. *Commun. ACM*, 11(6):419–422, June 1968.