**Hochschule Esslingen**  
Sommersemester 2023



**Projekt Sport Challenge**  
Softwareprojekt für doubleSlash

Projektteam:   
Jason Patrick Duffy (jaduit00@hs-esslingen.de)  
Robin Hackh (rohait02@hs-esslingen.de)  
Tom Nguyen (tongit00@hs-esslingen.de)  
Mason Schönherr (mascit43@hs-esslingen.de)

Betreuer: Herr Prof. Dr. Nitzsche  
Kunden: doubleSlash Net-Business GmbH

Stand: 6. April 2023

Inhaltsverzeichnis

[1 Einführung 3](#_Toc131642009)

[1.1 Projektbeschreibung 3](#_Toc131642010)

[1.2 Anforderungen 3](#_Toc131642011)

[1.2.1 Beschränkungen 4](#_Toc131642012)

[1.3 Qualitätsziele 4](#_Toc131642013)

[1.4 Risiken 4](#_Toc131642014)

[1.5 Zielgruppe 5](#_Toc131642015)

[1.6 Stakeholder 5](#_Toc131642016)

[1.7 Personas 6](#_Toc131642017)

[2 Projektmanagement 7](#_Toc131642018)

[2.1 Agiles Projektmanagement 7](#_Toc131642019)

[2.1.1 Terminplan 7](#_Toc131642020)

[2.1.2 Definition of Ready 7](#_Toc131642021)

[2.1.3 Definition of Done 7](#_Toc131642022)

[2.2 Projektverwaltung 8](#_Toc131642023)

[2.2.1 User Story Regeln 8](#_Toc131642024)

[2.3 Versionsverwaltung 8](#_Toc131642025)

[2.3.1 Commit Regeln 8](#_Toc131642026)

[2.3.2 Branch Regeln 9](#_Toc131642027)

[2.4 Teamkommunikation 9](#_Toc131642028)

[2.5 Implementierung 9](#_Toc131642029)

[2.5.1 Testing Policy 10](#_Toc131642030)

[3 Funktionsumfang 11](#_Toc131642031)

[4 UI-Entwürfe 12](#_Toc131642032)

[5 Architektur 13](#_Toc131642033)

[5.1 Lizenzen 14](#_Toc131642034)

[5.1.1 Projekt 14](#_Toc131642035)

[5.1.2 React 14](#_Toc131642036)

[5.1.3 Java Spring Boot 14](#_Toc131642037)

[5.1.4 Keycloak 14](#_Toc131642038)

[5.1.5 PostgreSQL 14](#_Toc131642039)

[6 Aufwandschätzung 15](#_Toc131642040)

[Quellenverzeichnis 16](#_Toc131642041)

[Abbildungsverzeichnis 16](#_Toc131642042)

# 1 Einführung

Das Sport Challenge Projekt wird im Auftrag der Firma doubleSlash Net-Business GmbH durchgeführt. Dieses Dokument beschreibt die Umsetzung und Regeln, die wir für dieses Projekt festgelegt haben.

## 1.1 Projektbeschreibung

Bei der doubleSlash Rad- und Laufaktion können Benutzer zusammen mit einem Team an befristeten Challenges teilnehmen, bei denen ihnen und ihrem Team ihre zurückgelegte Strecke in Punkten gutgeschrieben wird. Je nach Sportart (Laufen, Fahrradfahren, etc.) wird zur Strecke eine unterschiedliche Gewichtung als Faktor verwendet.

Am Ende der Challenge erhält das Gewinnerteam eine Belohnung, wie ein kostenloses Abendessen, und der gesammelte Kilometerstand wird mit einem Faktor zu einer wohltätigen Spende umgerechnet.

Die Dateneingabe erfolgt auf Vertrauensbasis und benötigt keine weitere Verifikation.

Folgende Features sind dabei von doubleSlash gewünscht:

* Verwalten von Sportarten mit unterschiedlicher Gewichtung
* Verwalten von Teilnehmern und Teams
* Verwalten von Bonusaktionen (z.B. Tage, an denen andere Gewichtungen gelten)
* Hinzufügen von Daten
* Benachrichtigungen bei Updates und Bonusaktionen
* Übersicht aller Teilnehmer und Teams und deren Platzierung
* Auswertung aller Eingaben und deren Ergebnissen
* Authentifizieren der Teilnehmer über SAML oder Windows AD

Folgende Features sind „nice to have”:

* API-Endpunkte zum Anbinden eines Datenimporters
* Einbindung der Strava-API
* Benutzergruppen mit unterschiedlichen Rechten

Ziel ist es, eine neue Full-Stack-Anwendung zum Koordinieren der firmeninternen Rad- und Laufaktionen umzusetzen und die zuvor verwendete Excel-Tabelle abzulösen.

## 1.2 Anforderungen

In diesem Abschnitt werden die von doubleSlash definierten funktionalen und nicht-funktionalen Anforderungen definiert und erläutert.

|  |  |  |
| --- | --- | --- |
| **ID** | **Anforderung** | **Erläuterung** |
| F1 | Im Browser aufrufbar | Die Anwendung soll im Web-Browser aufgerufen und vollständig verwaltet werden können. |
| F1.1 | Responsive | Da die Anwendung im Web-Browser angezeigt werden soll, muss sie sich responsiv dem Browserfenster anpassen und auch auf Mobiltelefonen problemlos funktionieren. |
| F2 | Erweiterbarkeit | Die Anwendung soll zukünftig erweiterbar, beispielsweise durch das Hinzufügen neuer Sportarten oder Anbinden einer Drittanbieter-API, sein. |
| F3 | Containerized | Die gesamte Anwendung soll in einem Docker-Container laufen. |
| NF1 | Agile | Das Projekt soll mithilfe agiler Methoden durchgeführt werden. |

### 1.2.1 Beschränkungen

Aus den definierten Anforderungen ergeben sich folgende Beschränkungen.

Die Anwendung sollte…

* …Plattformunabhängig über den Webbrowser aufgerufen werden können.
* …in einem Docker-Container laufen.
* …eine Lizenz verwenden, die dem Kunden das Verändern und Einsetzen ermöglicht.

## 1.3 Qualitätsziele

In diesem Abschnitt werden die Qualitätsziele nach ISO 25010[[1]](#footnote-2) beschreiben und einige dazugehörige Szenarien definiert.

|  |  |  |  |
| --- | --- | --- | --- |
| **Kategorie** | **Qualität** | **Beschreibung** | **Szenario** |
| Compatibility | Interoperability | Wenn möglich, sollte eine Schnittstelle zum Import von Daten bereitgestellt werden. | C1 |
| Usability | Learnability | Das System muss intuitiv sein und keine vorherige Einführung benötigen. | U1 |
| Usability | Operability | Die Dateneingabe sollte nicht zu viel Zeit in Anspruch nehmen. |  |
| Reliability | Recoverability | Es müssen regelmäßige automatische Backups der Daten erstellt werden. | R1 |
| Security | Integrity | Der Zugriff auf die Anwendung soll auf die Mitarbeiter von doubleSlash begrenzt sein. |  |
| Maintainability | Modularity | Das System muss einfach erweiterbar sein. | M1 |
| Portability | Installability | Die gesamte Anwendung muss in einem Container laufen. |  |

**Szenarien**

|  |  |
| --- | --- |
| **ID** | **Szenario** |
| C1 | Das Unternehmen kann eine eigene Importanwendung schreiben, die die gesammelten Daten aus anderen Anwendungen in diese Anwendung importiert. |
| U1 | Ein neuer Benutzer kann ohne Einführung innerhalb von 10 Minuten mit der Anwendung umgehen und Daten erfassen. |
| R1 | Bei Datenverlust aufgrund von Fehlern oder externer Einflüsse müssen die Daten wiederhergestellt werden können. |
| M1 | Erweiterungen nach Durchführung des ursprünglichen Projekts können ohne größere Änderungen an der Codebasis gemacht werden. |

## 1.4 Risiken

Da es sich für alle Mitglieder des Projektteams um das erste große Softwareprojekt handelt, gibt es einige Risiken. Die folgenden Risiken haben wir identifiziert und Präventionsmaßnahmen definiert.

|  |  |
| --- | --- |
| **Risiko** | **Präventionsmaßnahmen** |
| Aufgrund mangelnder Erfahrung könnte die korrekte Zeiteinschätzung für das Umsetzen von User Stories schwierig sein und zu Problemen führen. | * Zusammenarbeit mit dem Betreuer bei den ursprünglichen Schätzungen. * Kontinuierliches Anpassen der Schätzungen, aufgrund von gesammelten Erfahrungen. |
| Ebenso könnte es aufgrund mangelnder Erfahrung mit den verwendeten Technologien und Frameworks zu Schwierigkeiten bei der Umsetzung kommen, was die Qualität des Endprodukts beeinträchtigen würde. | * Workshops mit dem Kunden, bei dem das Projektteam in die neuen Technologien eingeführt wird. * Verwendung von Best Practices. * Regelmäßige Code Reviews im Team, um gegenseitiges Lernen zu fördern. |
| Es könnte zu Problemen kommen, alle geplanten Features bis zum Ende des Projekts vollständig umzusetzen, da die Zeit der Teammitglieder aufgrund anderer akademischer Verpflichtungen begrenzt ist. | * Priorisierung der Features, damit die wichtigsten Funktionen zuerst implementiert werden. * Kontinuierliche Überwachung des Fortschritts, damit Probleme früh identifiziert werden können. * Offene Kommunikation im Team, sodass man sich gegenseitig aushelfen kann. |

## 1.5 Zielgruppe

Da das Projekt nur intern verwendet werden soll, sind nur die Mitarbeiter der Firma doubleSlash, die an der firmeninternen Sport Challenge teilnehmen möchten, die Zielgruppe.

## 1.6 Stakeholder

Es gibt nur wenige Stakeholder, da es sich um ein Projekt für eine firmeninterne Anwendung handelt. Hier werden diese aufgelistet und ihre Kontaktdaten sowie Erwartungen angegeben.

|  |  |  |
| --- | --- | --- |
| **Rolle** | **Kontakt** | **Erwartungen** |
| Projektteam | Jason Patrick Duffy (jaduit00@hs-esslingen.de)  Robin Hackh  (rohait02@hs-esslingen.de)  Tom Nguyen  (tongit00@hs-esslingen.de)  Mason Schönherr  (mascit43@hs-esslingen.de) | * Neue Technologien kennenlernen und anwenden * Einblicke in die angewandte Softwaretechnik erhalten |
| Kunde / Benutzer  doubleSlash Net-Business GmbH | Andreas Heinrich (Andreas.Heinrich@doubleslash.de)  Johannes Mayer (Johannes.Mayer@doubleslash.de)  Mattias Abramovic (Mattias.Abramovic@doubleslash.de) | * Eine Full-Stack-Anwendung, die intern verwendet werden kann * Wöchentliches Meeting über den Projektstand * Einhalten des Zeitplans |
| Betreuer | Prof. Dr. Jörg Nitzsche (Joerg.Nitzsche@hs-esslingen.de) | * Wöchentliches Meeting über den Projektstand * Pünktliche Abgabe der Meilensteine * Teilnahme an den angesetzten Veranstaltungen |

## 1.7 Personas

TODO

# 2 Projektmanagement

In diesem Abschnitt werden die für das Projekt verwendeten Werkzeuge und Methoden definiert und erläutert.

## 2.1 Agiles Projektmanagement

Eine der Anforderungen an das Projekt, ist die Verwendung agiler Methoden für das Projektmanagement. Hierfür haben wir uns für das Scrum Framework entschieden, da es uns eine gute Zusammenarbeit mit dem Kunden, Flexibilität bei der Umsetzung des Projekts ermöglicht und dennoch durchgeplant ist.

Rollenverteilung:

* Product Owner: doubleSlash Net-Business GmbH
* Scrum Master: Jason Patrick Duffy
* Entwicklungsteam: Jason Patrick Duffy, Robin Hackh, Tom Nguyen, Mason Schönherr

### 2.1.1 Terminplan

Wir haben uns für eine Sprintdauer von einer Woche entschieden, da es sich um ein zeitlich relativ kurzes Projekt handelt und auch unsere Treffen mit dem Betreuer und dem Kunden wöchentlich stattfinden.

Der Sprint beginnt wöchentlich donnerstags. Von 10:15 Uhr bis 11:00 Uhr findet das Treffen mit dem Betreuer statt. Danach findet von 11:30 Uhr bis 12:30 Uhr die Sprint Review mit dem Kunden statt. Ab 19:15 Uhr trifft sich das Team für die Retrospektive, woraufhin wir gegen 20 Uhr den nächsten Sprint planen.

Aufgrund des engen Terminplans der Teammitglieder haben wir uns gegen ein tägliches Meeting entschieden, weshalb wir uns innerhalb des Sprints nur zweimal die Woche – sonntags um 18:00 Uhr und dienstags um 13:30 Uhr – treffen.

Zusammengefasst:

* Sprintdauer: 1 Woche
* Sprintbeginn: Donnerstag
* Review: Donnerstag ab 11:30 Uhr
* Retrospektive: Donnerstag ab 19:15 Uhr
* Planung: Donnerstag ab 20:00 Uhr
* Treffen mit dem Betreuer: Donnerstag ab 10:15 Uhr
* Stand-up-Meetings: Sonntag ab 18:00 Uhr und Dienstag ab 13:30 Uhr

### 2.1.2 Definition of Ready

Damit eine User Story in den Sprint aufgenommen werden kann, müssen folgende Kriterien erfüllt sein:

* Die User Story ist für alle Beteiligten klar, verständlich und vollständig beschrieben.
* Der Aufwand für die User Story ist vom Projektteam geschätzt.
* Alle sicherheits- und datenschutzrelevanten Aspekte wurden geprüft und berücksichtigt.
* Es wurde geprüft, dass die User Story innerhalb der Architektur umsetzbar ist.
* Es wurden klare Akzeptanzkriterien formuliert, die von allen Beteiligten verstanden werden und testbar sind.
* Der Aufwand für die User Story wurde geschätzt.

### 2.1.3 Definition of Done

Eine User Story gilt als fertiggestellt, wenn folgende Kriterien erfüllt sind:

* Alle Akzeptanzkriterien der User Story sind erfüllt.
* Die für die Umsetzung benötigte Zeit wurde eingetragen.
* Eine Code Review wurde durchgeführt.
* Implementierungsregeln wurden eingehalten.
* Der Code steht in der Versionsverwaltung zur Verfügung.
* Die Implementierung der User Story hat keine kritischen Fehler.
* Alle Tests waren erfolgreich und folgen der Testing Policy.

## 2.2 Projektverwaltung

Zur Projektverwaltung verwenden wir Atlassian Jira. Dies ermöglicht uns an einem Ort unsere Epics und User Stories festzuhalten sowie unsere Sprints zu planen. Zudem erhalten wir so einen schnellen Überblick über den aktuellen Fortschritt und können Probleme schneller identifizieren.

Link zum Jira-Board: <https://jduffy.atlassian.net/jira/software/projects/SCP/boards/2>

### 2.2.1 User Story Regeln

Beim Erstellen von User Stories sollte darauf geachtet werden, immer eine Priorität anzugeben.

Die Prioritäten haben folgende Bedeutung:

* Höchste: Grundlegende Funktion. Muss implementiert werden.
* Hoch: Wichtige Funktion. Sollte auf jeden Fall implementiert werden.
* Mittel: Geforderte Funktion. Sollte implementiert werden.
* Niedrig: Nicht geforderte Funktion. Kann implementiert werden.
* Sehr niedrig: Nicht geforderte Funktion. Sollte nur bei Zeitüberschuss implementiert werden.

## 2.3 Versionsverwaltung

Zur Versionsverwaltung verwenden wir das interne GitLab der Hochschule Esslingen. Wir haben uns für GitLab entschieden, da wir dort einen Code Runner zur Verfügung haben, der uns das Umsetzen von CI/CD ermöglicht, was mit GitHub nicht möglich gewesen wäre.

Wir planen die Auto DevOps-Funktion von GitLab zu verwenden, um Änderungen im Code automatisch zu testen, zu bauen und bereitzustellen.

Link zum GitLab: <https://gitlab.hs-esslingen.de/jaduit00/Sport-Challenge-Projekt>

### 2.3.1 Commit Regeln

Um Commits einheitlich und übersichtlich zu halten, verwenden wir die Conventional Commits-Spezifikation[[2]](#footnote-3). Commits sollten immer nur einem Typ zugehörig sein, Commits sollten also regelmäßig gemacht werden, um übergroße Commits, die mehrere Typen haben, zu verhindern.

**Aufbau eines Conventional Commits:**  
<Typ>: Beschreibung (Titel des Commits)  
Optionaler Textkörper, der den Commit genauer beschreibt.  
Optionale Fußzeile

Folgende Typen verwenden wir:

|  |  |
| --- | --- |
| **Typ** | **Beschreibung** |
| fix | Ein Fehler in der Codebasis wird behoben. |
| feat | Eine neue Funktion wurde in die Codebasis eingeführt. |
| ci | Änderungen an der CI/CD Pipeline. |
| docs | Ein Dokument wurde verändert oder hinzugefügt. |
| test | Ein Test wurde verändert oder hinzugefügt. |
| revert | Ein Commit wurde rückgängig gemacht. |

In der Fußzeile verwenden wir drei Typen:

|  |  |
| --- | --- |
| **Typ** | **Beschreibung** |
| BREAKING CHANGE | Es werden tiefgründige Änderungen in der API eingeführt.  Immer in Kombination mit einem ! beim Typ. |
| Refs: A, B, … | In Jira referenzierte Tickets, die zum Commit gehören. |
| Time: X h | Die für die Umsetzung des Commits benötigte Zeit in Stunden.  Zur Verbesserung der Lesbarkeit, ist die kleinste mögliche Angabe „0.1 h“ und kann von dort in 0.1-Schritten erhöht werden. Hat ein Commit also nur eine Minute gebraucht, gibt man dennoch „0.1 h“, statt „0.0166 h“, an. |

**Beispiel**

feat!: Neues System zur Verwaltung von Nutzerdaten implementiert  
Nutzerdaten werden jetzt über das neue System verwaltet. Durch effizientere Nutzung der Datenbank, ist die Reaktionsgeschwindigkeit der Anwendung nun deutlich schneller.  
BREAKING CHANGE: Alte Nutzerdaten sind mit der neuen Version inkompatibel.  
Refs: SCP-12  
Time: 4.7 h

### 2.3.2 Branch Regeln

Git Flow ! -> **BESPRECHEN – FUNKTIONIERT WOHL NICHT GUT MIT CI/CD**

**Ggf. Lieber eine Master Branch, von der man sich wenn man dran arbeitet eine neue Branch ableitet**

## 2.4 Teamkommunikation

Die teaminterne Kommunikation findet über eine Discord Gruppe und persönliche Treffen statt.

## 2.5 Implementierung

Um die Lesbarkeit der Implementierung zu garantieren, haben wir einige Regeln definiert.

|  |  |  |
| --- | --- | --- |
| **Regel** | **Beschreibung** | **Beispiel** |
| Code ausreichend kommentieren | Code, der nicht selbsterklärend ist, sollte immer eine kurze Beschreibung, sowie den Autor beinhalten. Wenn möglich sollten sprachenspezifische Features, wie JavaDoc, eingebunden werden.  Selbsterklärender Code sind beispielsweise simple Getter und Setter. | Abbildung 1: Beispiel in Java mit JavaDoc |
| Einheitlichen Einrückungsstil verwenden | Zur besseren Lesbarkeit sollten entweder der **Allman** oder **Kernighan & Ritchie** Einrückungsstil konsistent verwendet werden. | Abbildung 2: Verschiedene Einrückungsstile [1] |

### 2.5.1 Testing Policy

TODO – Testing so und so viel Coverage, so und so

# 3 Funktionsumfang

Unsere Anwendung muss dies das können, weil dies das. User Stories

Siehe ggf. <https://docs.arc42.org/examples/overview-example-3/>

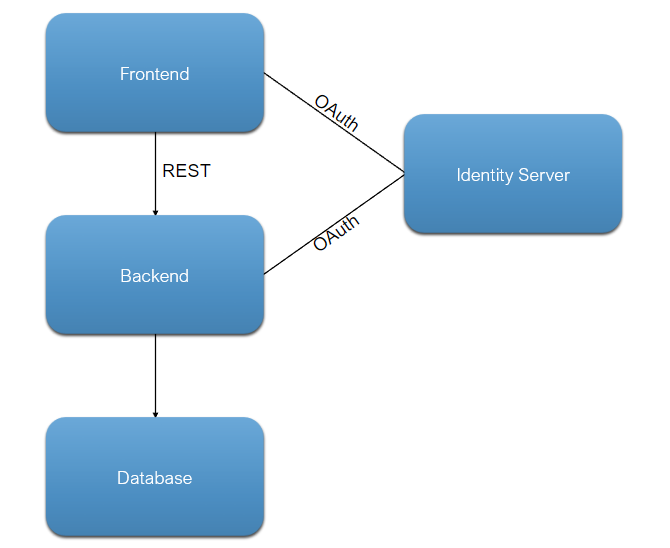
# 4 UI-Entwürfe

Hier einfach Bilder von unseren Entwürfen, ggf. kurze Erklärungen. **Schreibt hier irgendwo zum Referenzieren die Farbcodes und ggf. eine kurze Erklärung, für was sie verwendet werden hin.**

**Farbcodes**



# 5 Architektur



Bei der Wahl der Architektur war es besonders wichtig, dass die Teammitglieder bereits grundlegende Kenntnisse in den zugrundeliegenden Programmiersprachen haben und die kommerzielle Nutzung der Frameworks erlaubt ist. Wenn möglich wollten wir zudem Open Source Frameworks verwenden.

Für das Frontend haben wir uns auf das React Framework geeinigt, da jedes Teammitglied bereits Erfahrung bei der Arbeit mit JavaScript hat und die hohe Verbreitung des Frameworks uns bei Fragen und Problemen helfen sollte.

Beim Backend fiel die Wahl auf das Spring Boot Framework, da das Team ebenfalls bereits Erfahrung bei der Entwicklung mit Java hat und auch hier der hohe Grad der Verbreitung das Problemlösen erleichtern sollte. Das Spring Boot Framework eignet sich zudem ideal zum Aufbau einer REST-Schnittstelle.

Da nur Mitarbeiter des Kunden Zugriff auf das System haben dürfen, stellt die Authentifizierung von Benutzern einen wichtigen Punkt dar. Hier haben wir uns für Keycloak entschieden, da es sich sehr gut mit der von den Kunden verwendeten SAML-Schnittstelle verbinden lässt.

Schließlich entschieden wir uns beim Datenbanksystem für PostgreSQL. Auch hier sollte der hohe Verbreitungsgrad bei der Problemlösung helfen. Zudem ermöglicht die Funktionsreichheit von PostgreSQL die spätere Erweiterung der Anwendung mit komplexeren Aufgabenstellungen – es ist also zukunftsorientiert.

Daraus ergibt sich folgende Architektur:  
Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

## 5.1 Lizenzen

In diesem Abschnitt werden die Lizenzen der verwendeten Frameworks und des Projekts aufgelistet und beschrieben.

### 5.1.1 Projekt

Für unser Projekt haben wir uns für die MIT Lizenz [2] entschieden. Die Lizenz erlaubt die kommerzielle und private Nutzung und das Modifizieren und Verteilen der Software. Es wird keine Garantie oder Haftung angeboten. Eine Kopie der Lizenz muss mitgeliefert werden.

Mit der MIT Lizenz kann der Kunde in Zukunft das Projekt frei weiterentwickeln und erweitern, was eine der Anforderungen an das Projekt ist.

### 5.1.2 React

React wird ebenfalls mit der MIT Lizenz ausgeliefert [3]. Die Lizenz ist identisch zu unserem Projekt. Die Lizenz erlaubt die kommerzielle und private Nutzung und das Modifizieren und Verteilen der Software. Es wird keine Garantie oder Haftung angeboten. Eine Kopie der Lizenz muss mitgeliefert werden.

### 5.1.3 Java Spring Boot

Java Spring Boot verwendet die Apache Lizenz 2.0 [4]. Die Lizenz erlaubt die kommerzielle und private Nutzung, das Modifizieren und Verteilen der Software sowie die Verwendung in Patenten. Nicht erlaubt ist die Verwendung von Apache Trademarks. Zudem wird keine Garantie oder Haftung angeboten. Software, die Anwendungen mit der Apache Lizenz 2.0 verwendet, muss eine Kopie der Lizenz beinhalten. Änderungen an der lizenzierten Software müssen klar markiert sein. Hat die lizenzierte Software eine „NOTICE“-Textdatei, muss diese ebenfalls mitgeliefert werden.

### 5.1.4 Keycloak

Bei Keycloak wird ebenfalls die Apache Lizenz 2.0 verwendet [5]. Der Lizenztext ist identisch zu dem von Java Spring Boot. Die Lizenz erlaubt die kommerzielle und private Nutzung, das Modifizieren und Verteilen der Software sowie die Verwendung in Patenten. Nicht erlaubt ist die Verwendung von Apache Trademarks. Zudem wird keine Garantie oder Haftung angeboten. Software, die Anwendungen mit der Apache Lizenz 2.0 verwendet, muss eine Kopie der Lizenz beinhalten. Änderungen an der lizenzierten Software müssen klar markiert sein. Hat die lizenzierte Software eine „NOTICE“-Textdatei, muss diese ebenfalls mitgeliefert werden.

### 5.1.5 PostgreSQL

PostgreSQL wird unter der PostgreSQL Lizenz veröffentlicht [6]. Die Lizenz erlaubt das Verwenden, Kopieren, Modifizieren und Verteilen der Software für jeglichen Verwendungszweck ohne Gebühr oder Einwilligung der Entwickler, solange der Urheberrechtshinweis mitverteilt wird.

# 6 Aufwandschätzung

Keine Ahnung, was wir hier genau machen sollen…

# Quellenverzeichnis

|  |  |
| --- | --- |
| [1] | Fermat's Library, „Darstellung von Einrückungsstilen,“ 10 Oktober 2022. [Online]. Available: https://twitter.com/fermatslibrary/status/1582719971146309632/. [Zugriff am 4 April 2023]. |
| [2] | „MIT Lizenz,“ [Online]. Available: https://opensource.org/license/mit/. [Zugriff am 6 April 2023]. |
| [3] | Meta, „React Lizenz,“ [Online]. Available: https://github.com/facebook/react/blob/main/LICENSE. [Zugriff am 6 April 2023]. |
| [4] | Spring Boot, „Lizenz,“ [Online]. Available: https://github.com/spring-projects/spring-boot/blob/main/LICENSE.txt. [Zugriff am 6 April 2023]. |
| [5] | Keybloack, „Lizenz,“ [Online]. Available: https://github.com/keycloak/keycloak-community/blob/main/LICENSE.txt. [Zugriff am 6 April 2023]. |
| [6] | PostgreSQL, „Lizenz,“ [Online]. Available: https://www.postgresql.org/about/licence/. [Zugriff am 6 April 2023]. |
| [7] | Conventional Commits, „Spezifikation,“ [Online]. Available: https://www.conventionalcommits.org/de/v1.0.0/. [Zugriff am 4 April 2023]. |
| [8] | ISO, „ISO25010,“ [Online]. Available: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010/. [Zugriff am 4 April 2023]. |

# Abbildungsverzeichnis

[Abbildung 1: Beispiel in Java mit JavaDoc 9](#_Toc131641978)

[Abbildung 2: Verschiedene Einrückungsstile [1] 10](#_Toc131641979)

1. [8] Öffentlich zugängliche Beschreibung der ISO-Norm 25010 [↑](#footnote-ref-2)
2. [7] Conventional Commits v1.0.0 Spezifikation [↑](#footnote-ref-3)