**Hochschule Esslingen**  
Sommersemester 2023



**Sport Challenge Projekt**  
Softwareprojekt für doubleSlash

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

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

Stand: 22. Juni 2023

[Abbildungsverzeichnis 3](#_Toc138316724)

[1 Einführung 4](#_Toc138316725)

[1.1 Projektbeschreibung 4](#_Toc138316726)

[1.2 Anforderungen 4](#_Toc138316727)

[1.2.1 Beschränkungen 5](#_Toc138316728)

[1.3 Qualitätsziele 5](#_Toc138316729)

[1.4 Risiken 5](#_Toc138316730)

[1.5 Zielgruppe 6](#_Toc138316731)

[1.6 Personas 6](#_Toc138316732)

[2 Projektmanagement 9](#_Toc138316733)

[2.1 Agiles Projektmanagement 9](#_Toc138316734)

[2.1.1 Terminplan 9](#_Toc138316735)

[2.1.2 Rollenverteilung 9](#_Toc138316736)

[2.1.3 Definition of Ready 10](#_Toc138316737)

[2.1.4 Definition of Done 10](#_Toc138316738)

[2.2 Projektverwaltung 10](#_Toc138316739)

[2.2.1 User Story Regeln 10](#_Toc138316740)

[2.3 Versionsverwaltung 11](#_Toc138316741)

[2.3.1 Commit Regeln 11](#_Toc138316742)

[2.3.2 Branch Regeln 12](#_Toc138316743)

[2.4 Teamkommunikation 12](#_Toc138316744)

[2.5 Implementierung 12](#_Toc138316745)

[2.5.1 Tests 13](#_Toc138316746)

[2.6 Entwicklungsumgebung 13](#_Toc138316747)

[3 Funktionsumfang 14](#_Toc138316748)

[3.1 User Stories 14](#_Toc138316749)

[4 UI-Entwurf 22](#_Toc138316750)

[4.1 Prototyp 22](#_Toc138316751)

[5 Architektur 28](#_Toc138316752)

[5.1 Lizenzen 29](#_Toc138316753)

[5.1.1 Projekt 29](#_Toc138316754)

[5.1.2 React 29](#_Toc138316755)

[5.1.3 Java Spring Boot 29](#_Toc138316756)

[5.1.4 Keycloak 29](#_Toc138316757)

[5.1.5 PostgreSQL 30](#_Toc138316758)

[5.2 Logischer Aufbau 30](#_Toc138316759)

[5.3 Verteilungsansicht 31](#_Toc138316760)

[5.4 Datenhaltung 32](#_Toc138316761)

[5.5 ER-Modell der Datenbank 33](#_Toc138316762)

[5.6 Klassenstruktur der Datenbankentitäten 35](#_Toc138316763)

[5.7 Ablauf einer Challenge 36](#_Toc138316764)

[6 Aufwandschätzung 36](#_Toc138316765)

[7 API-Dokumentation 38](#_Toc138316766)

[7.1 Member 38](#_Toc138316767)

[7.2 Image 41](#_Toc138316768)

[7.3 Sport 42](#_Toc138316769)

[7.4 Challenge 44](#_Toc138316770)

[7.5 SAML 48](#_Toc138316771)

[7.6 Team 49](#_Toc138316772)

[7.7 Team-Member 50](#_Toc138316773)

[7.8 Challenge-Sport 53](#_Toc138316774)

[7.9 Bonus 55](#_Toc138316775)

[7.10 Aktivitäten 57](#_Toc138316776)

[Quellenverzeichnis 61](#_Toc138316777)

# Abbildungsverzeichnis

[Abbildung 1: Persona einer Sekretärin 7](#_Toc138316701)

[Abbildung 2: Persona eines Geschäftsleiters 7](#_Toc138316702)

[Abbildung 3: Persona eines Softwaretechnikers 8](#_Toc138316703)

[Abbildung 4: Beispiel für den Aufbau mit zwei laufenden Feature-Branches 12](#_Toc138316704)

[Abbildung 5: Ausgefülltes Beispiel für die Verwendung von Branches 12](#_Toc138316705)

[Abbildung 6: Beispiel in Java mit JavaDoc 13](#_Toc138316706)

[Abbildung 7: Verschiedene Einrückungsstile [1] 13](#_Toc138316707)

[Abbildung 8: Die Loginseite ist die erste geöffnete Seite 23](#_Toc138316708)

[Abbildung 9: Die Startseite wird nach dem Login geöffnet 23](#_Toc138316709)

[Abbildung 10: Die Seite einer einzelnen Challenge 24](#_Toc138316710)

[Abbildung 11: Die „Meine Challenges“-Seite erlaubt die Eingabe von Daten 25](#_Toc138316711)

[Abbildung 12: Eine neue Challenge erstellen 26](#_Toc138316712)

[Abbildung 13: Ein neues Team erstellen 27](#_Toc138316713)

[Abbildung 14: Das Benutzerprofil erlaubt das Bearbeiten der Nutzerdaten 27](#_Toc138316714)

[Abbildung 15: Grundlegende Architektur 28](#_Toc138316715)

[Abbildung 16: Architektur mit den von uns gewählten Frameworks 29](#_Toc138316716)

[Abbildung 21: Komponentendiagramm der Anwendung 30](#_Toc138316717)

[Abbildung 22: Verteilungsdiagramm der Anwendung 31](#_Toc138316718)

[Abbildung 19: Datenhaltungsdiagramm der Anwendung 32](#_Toc138316719)

[Abbildung 17: ER-Diagramm der Datenbankstruktur 33](#_Toc138316720)

[Abbildung 18: Klassendiagramm der Entitäten 35](#_Toc138316721)

[Abbildung 20: Ablaufdiagramm einer einzelnen Challenge 36](#_Toc138316722)

[Abbildung 23: Vorläufige Sprintplanung zum Stand 13. April 2023 37](#_Toc138316723)

# 1 Einführung

Das Sport Challenge Projekt wird im Auftrag der Firma doubleSlash Net-Business GmbH durchgeführt. Dieses Dokument dokumentiert die Umsetzung des Projekts und die 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. |  |
| 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. |
| 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 Personas

Um eine bessere Vorstellung der zukünftigen Nutzer zu bekommen, haben wir folgende Personas erstellt.

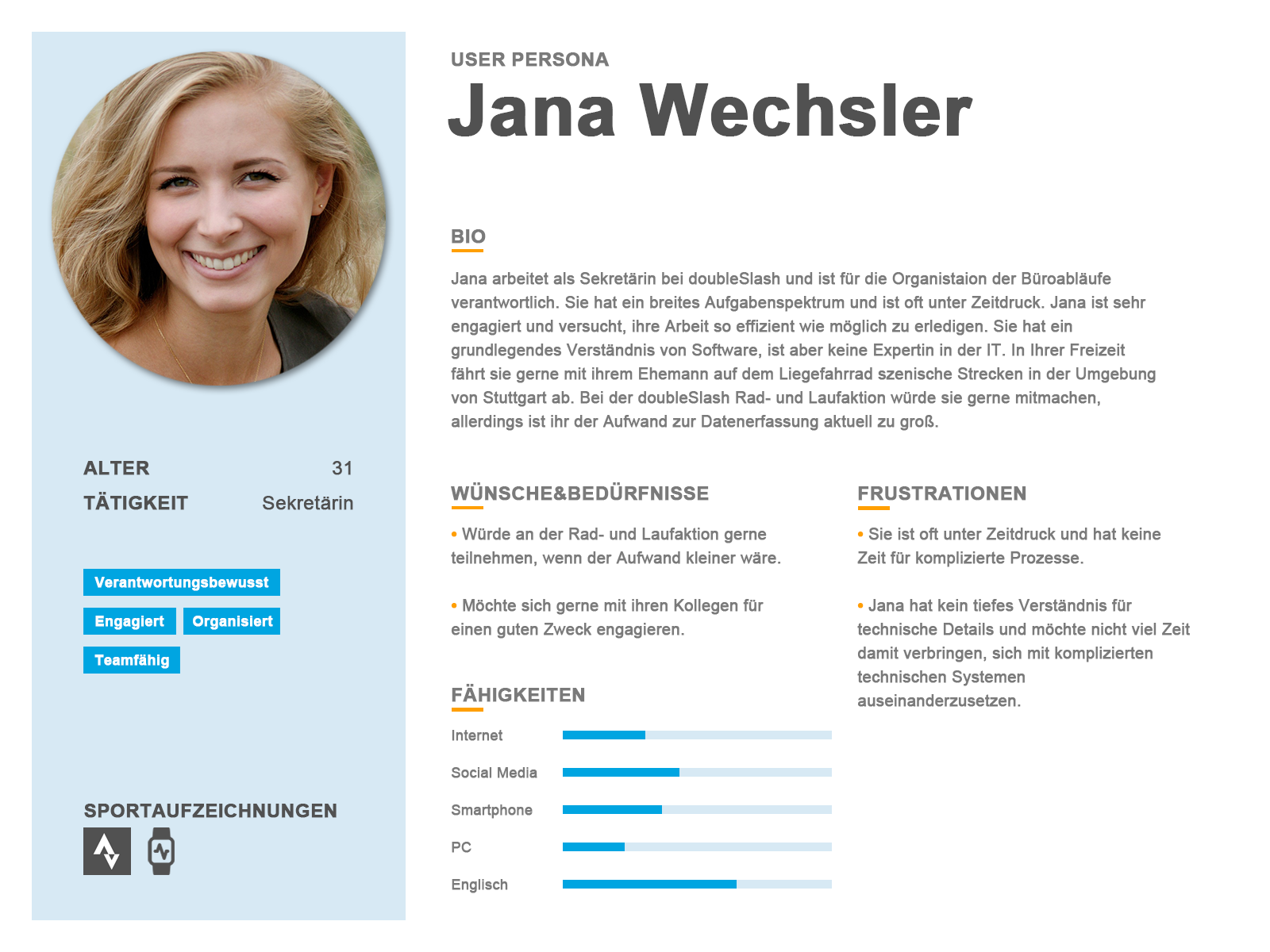


Abbildung 1: Persona einer Sekretärin

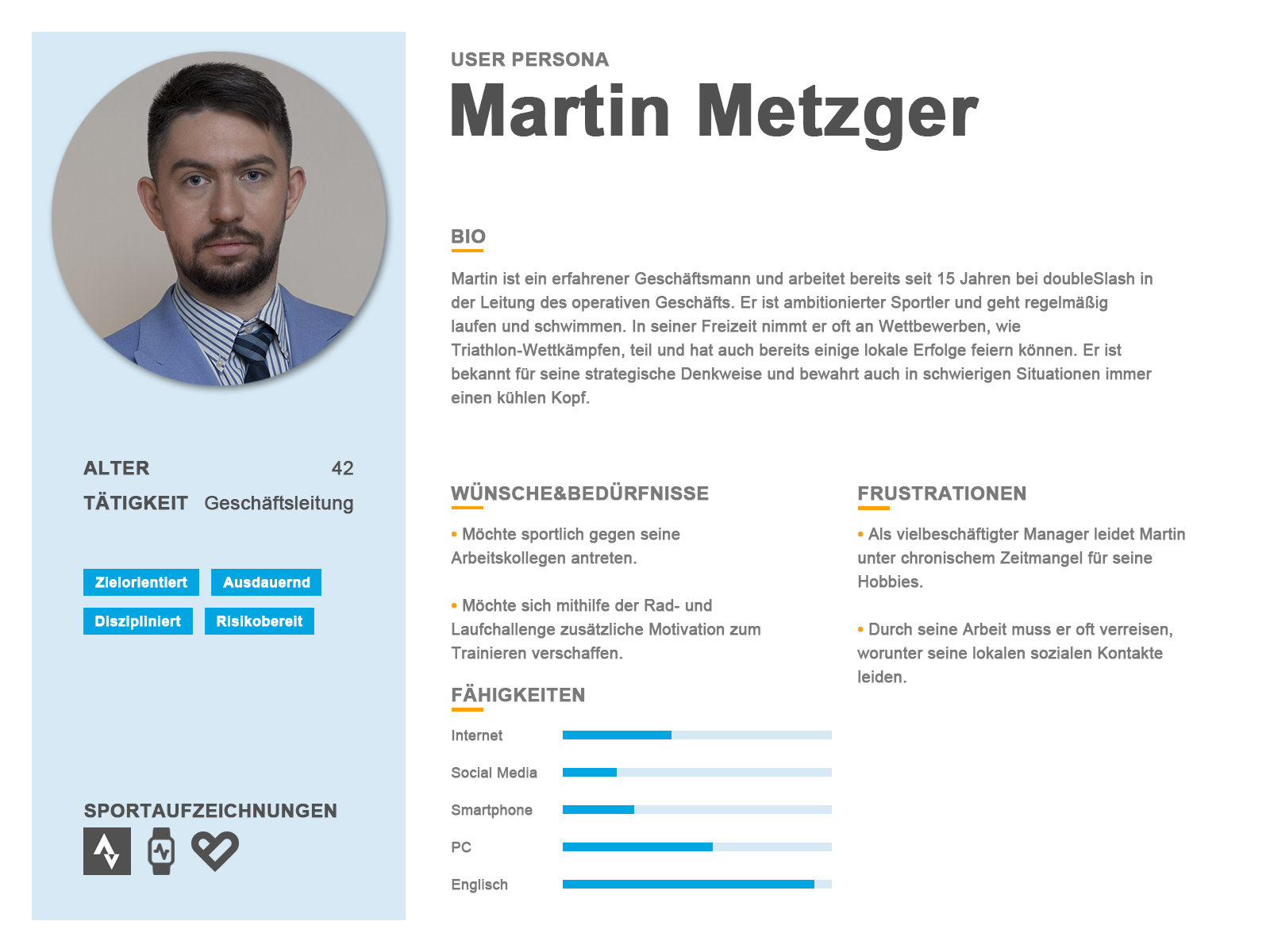


Abbildung 2: Persona eines Geschäftsleiters

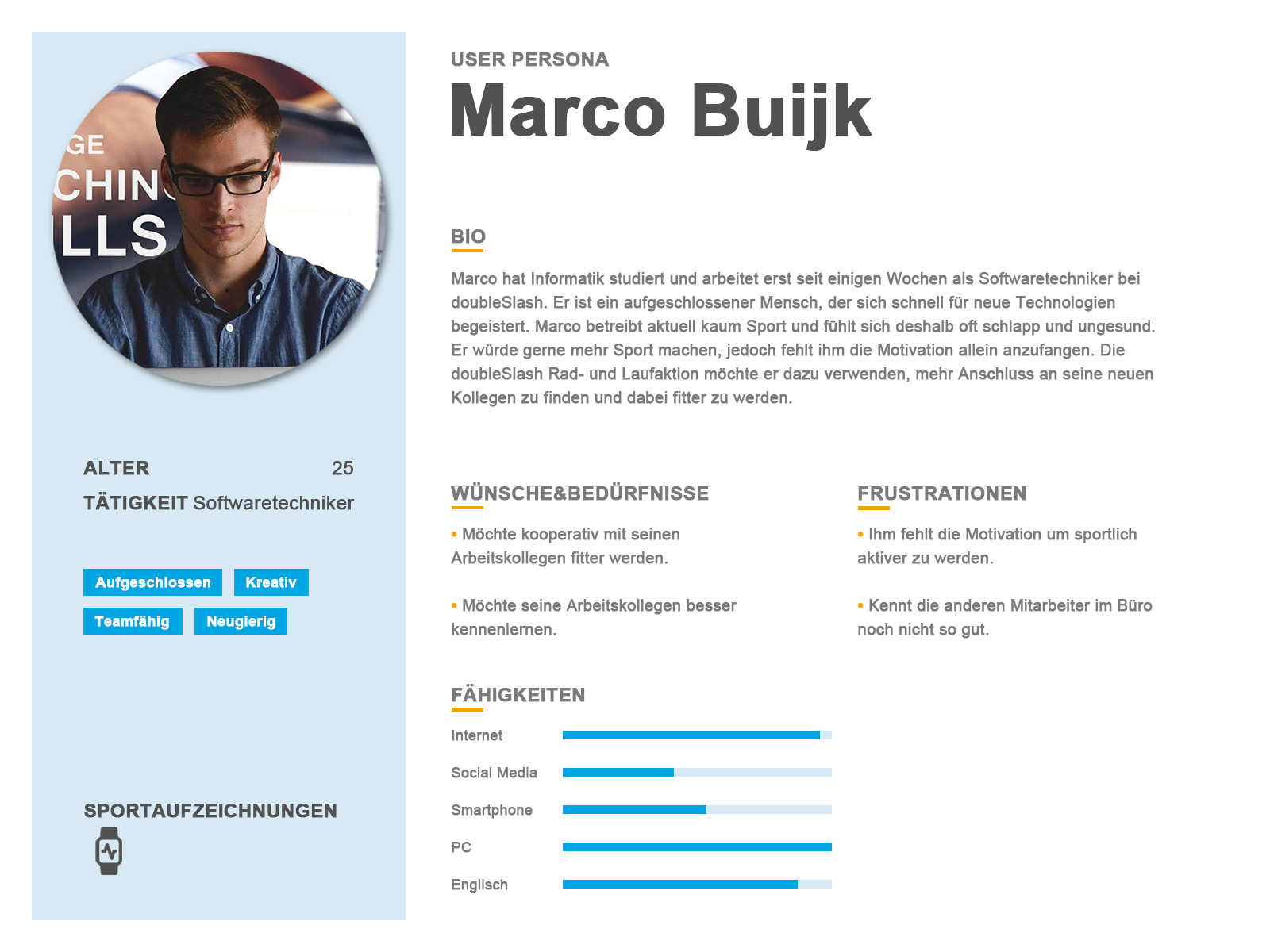


Abbildung 3: Persona eines Softwaretechnikers

# 2 Projektmanagement

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

## 2.1 Agiles Projektmanagement

Die Anforderung NF1 fordert die Verwendung agiler Methoden für das Projektmanagement. Hier haben wir uns für das Scrum Framework entschieden, da es uns eine gute Zusammenarbeit mit dem Kunden und Flexibilität bei der Umsetzung des Projekts ermöglicht und dennoch durchgeplant ist.

Scrum-Rollenverteilung:

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

### 2.1.1 Terminplan

Wir haben uns für eine Sprintdauer von einer Woche entschieden, da es sich um ein zeitlich 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

Der Abgabetermin für das Projekt ist der 29. Juni 2023.

### 2.1.2 Rollenverteilung

Die Rollenverteilung ist in unserem Projekt sehr flexibel. Die hier genannten Abschnittsleiter sind die Entscheidungsträger bei Entscheidungen in den jeweiligen Abschnitten – beispielsweise haben sie bei Unstimmigkeiten über die Umsetzung eines Features im Team das letzte Wort – und kümmern sich um das Einhalten von Best Practices und den definierten Implementierungsregeln. Die meisten Entscheidungen werden trotzdem versucht im Gesamtteam getroffen zu werden. Zudem bedeutet es nicht, dass der Frontend-Lead nicht auch im Backend arbeiten kann. Die Rollen wurden möglichst nach Menge der Erfahrung im jeweiligen Bereich verteilt.

|  |  |  |
| --- | --- | --- |
| **Person** | **Rollen** | **Rollenbeschreibung** |
| Jason Patrick Duffy | Project Lead  Scrum Master  Backend- & DB-Lead  Full-Stack-Entwickler | Leitet das Projekt und kümmert sich um die Einhaltung der Scrum-Regeln. Ist der Entscheidungsträger bei Entscheidungen im Bezug zum Backend und der Datenbank. Arbeitet an allen Aspekten der Implementierung mit. |
| Robin Hackh | Frontend- & UX-Lead  Full-Stack-Entwickler | Trifft Entscheidungen über das Frontend und der Benutzererfahrung. Arbeitet an allen Aspekten der Implementierung mit. |
| Mason Schönherr | Documentation-Lead  Teamsprecher  Full-Stack-Entwickler | Kümmert sich darum, dass Implementierungen ausreichend dokumentiert werden. Spricht in der Öffentlichkeit für das Team. Arbeitet an allen Aspekten der Implementierung mit. |
| Tom Nguyen Dinh | QA-Lead  Full-Stack-Entwickler | Prüft, dass ausreichend und richtig getestet wird. Arbeitet an allen Aspekten der Implementierung mit. |

### 2.1.3 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 wurde 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.

### 2.1.4 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.
* Die Branch wurde mit der Main-Branch zusammengefügt.
* 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.
* Niedrigste: 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. Zudem benutzen wir das hochschulinterne GitLab, da wir ansonsten auf die Begrenzung der CI/CD Minuten achten müssten.

Link zum GitLab (nur aus dem HS Esslingen-VPN erreichbar):   
<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) mit einem kleinen Zusatz. Commits sollten immer nur einem Typ zugehörig sein. Commits sollten also regelmäßig gemacht werden, um übergroße Commits, der mehrere Typen angehören, zu verhindern.

**Aufbau eines Conventional Commits:**  
<Typ>: Beschreibung (Titel des Commits)  
Optionaler Textkörper, der den Commit genauer beschreibt.  
Refs: SCP-0

Time: X h

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** |
| Refs: SCP-0, SCP-1, … | 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.  Dies ist nicht Teil der Conventional Commits Spezifikation. |

**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.  
Refs: SCP-12  
Time: 4.7 h

### 2.3.2 Branch Regeln



Abbildung 4: Beispiel für den Aufbau mit zwei laufenden Feature-Branches

Wir verwenden eine primäre Branch namens *Main*.

Pro User Story wird für das Implementieren eine Feature-Branch abgeleitet. Diese Branches werden Feature *X* genannt, wobei *X* für das spezifische Feature steht.

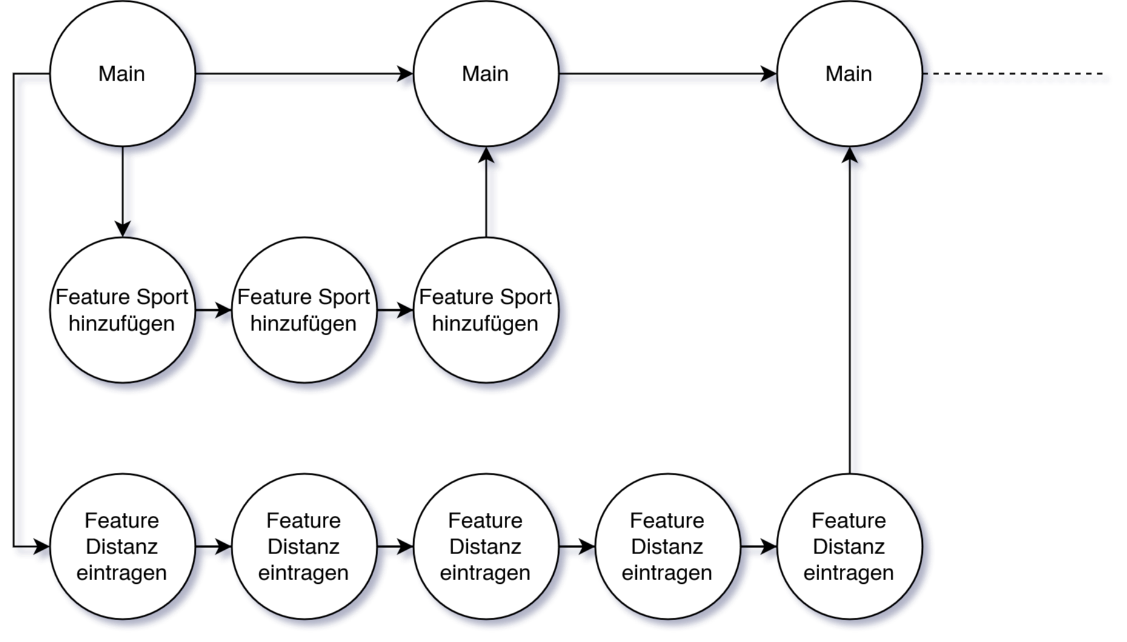


Abbildung 5: Ausgefülltes Beispiel für die Verwendung von Branches

Wenn das Feature implementiert ist, wird ein Merge Request durchgeführt, der von mindestens einem weiteren Teammitglied überprüft werden **muss**.

## 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. | Ein Bild, das Text enthält.  Automatisch generierte Beschreibung  Abbildung 6: Beispiel in Java mit JavaDoc |
| Einheitlichen Einrückungsstil verwenden | Zur besseren Lesbarkeit sollte stets der **Kernighan & Ritchie** Einrückungsstil verwendet werden. | Ein Bild, das Diagramm, Text, Brief enthält.  Automatisch generierte Beschreibung  Abbildung 7: Verschiedene Einrückungsstile [1] |

### 2.5.1 Tests

Tests haben das Ziel, sicherzustellen, dass die Implementierung richtig und fehlerfrei implementiert ist. Entsprechend werden alle nicht trivialen Funktionen getestet. Als nicht trivial sehen wir alle Funktionen an, die Logik beinhalten, also nicht beispielsweise simple Getter und Setter sind. Das Ziel ist es, eine Testabdeckung von 80% im Backend zu erreichen. Im Frontend testen wir nicht.

Beim Backend verwenden wir die *Spring Boot Starter Test* Abhängigkeit. Sie beinhaltet bereits alle für uns nötigen Funktionen und ist sehr gut mit Spring Boot integriert. Tests werden im Testpfad des Quellcodes gespeichert. Eine Testklasse beinhaltet stets alle Tests für eine ganze Klasse. Ihr Pfad sollte innerhalb des Testpfades dem der getesteten Klasse gleichen. Testet man also beispielsweise die Klasse Member mit dem Pfad /Database/Entities/, so sollte dieser Pfad im Testpfad exakt gleich sein. Der Name einer Testklasse ist dabei stets *getesteteKlasseTest*, also zum Beispiel MemberTest. Es ist zudem darauf zu achten, dass Tests immer das Profil „test“ verwenden.

Die Tests werden über GitLab CI/CD automatisiert. Ein Merge Request, der die Main-Branch als Ziel hat, wird geblockt, wenn mindestens ein Test fehlschlägt.

## 2.6 Entwicklungsumgebung

Zur Entwicklung verwenden wir im Frontend Microsoft Visual Studio Code, zusammen mit der von *dsznajder* entwickelten React Erweiterung (ID: dsznajder.es7-react-js-snippets).

Im Backend verwenden wir JetBrains IntelliJ IDEA, das native Unterstützung für Spring Boot anbietet.

Beide Umgebungen können beliebig mit anderen IDEs ersetzt werden, solange diese React, beziehungsweise Spring Boot mit Maven unterstützen.

Die REST-API kann nach einem Login über <http://localhost:8081/login> mittels Swagger-UI über <http://localhost:8081/swagger-ui/index.html> ausprobiert werden.

# 3 Funktionsumfang

In diesem Abschnitt werden unsere User Stories aufgelistet und mit der in [Abschnitt 2.2.1](#_2.2.1_User_Story) angesprochenen Priorität versehen. Die Ticket ID wird dabei automatisch von Jira generiert.

Die User Stories werden in folgende Epics eingeteilt:

|  |  |
| --- | --- |
| **Epic** | **Beschreibung** |
| Benutzerverwaltung | User Stories, die hauptsächlich mit der Verwaltung der Benutzer zusammenhängen. Beispiele hierfür sind Login oder Nutzerdaten bearbeiten. |
| Challengeverwaltung | User Stories, die sich hauptsächlich um die Verwaltung der Challenges drehen. Beispiele hierfür sind das Erstellen und Bearbeiten von Challenges. |
| Sportverwaltung | User Stories, die mit dem Verwalten von Sportarten zu tun haben. Beispiele sind das Erstellen einer neuen oder das Bearbeiten einer vorhandenen Sportart. |
| Teamverwaltung | User Stories, die das Verwalten von Teams definieren. Beispiele sind das Erstellen eines Teams oder das Hinzufügen von Mitgliedern. |

## 3.1 User Stories

Hier werden alle von uns erstellten User Stories zusammen mit ihrer Ticket ID, ihrer Epic-Zuweisung, dem Autor, der Priorität sowie der vorläufigen Zeitschätzung aufgelistet.

|  |  |
| --- | --- |
| Ticket ID | SCP-2 |
| Epic | Benutzerverwaltung |
| User Story | Um die Website verwenden zu können, möchte ich mich als Benutzer mit meinem doubleSlash Konto einloggen können. |
| Erstellt von | Jason Patrick Duffy |
| Priorität | Höchste |
| Zeitschätzung | 2 Tage 6 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-5 |
| Epic | Challengeverwaltung |
| User Story | Um einen Überblick über alle laufenden Challenges zu haben, möchte ich als Benutzer eine Übersicht mit allen aktiven Challenges und einer jeweiligen Kurzinfo dazu. |
| Erstellt von | Jason Patrick Duffy |
| Priorität | Höchste |
| Zeitschätzung | 1 Tag 1 Stunde |

|  |  |
| --- | --- |
| Ticket ID | SCP-7 |
| Epic | Sportverwaltung |
| User Story | Als Administrator möchte ich in der Lage sein, eine neue Sportart hinzuzufügen. |
| Erstellt von | Mason Schönherr |
| Priorität | Niedrig |
| Zeitschätzung | 4 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-10 |
| Epic | Teamverwaltung |
| User Story | Um an einer Challenge teilzunehmen, möchte ich als Benutzer einem Team beitreten können. |
| Erstellt von | Jason Patrick Duffy |
| Priorität | Hoch |
| Zeitschätzung | 2 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-13 |
| Epic | Challengeverwaltung |
| User Story | Als Benutzer will ich eine Challenge mit den gegebenen Spezifikationen anlegen können. |
| Erstellt von | Robin Hackh |
| Priorität | Höchste |
| Zeitschätzung | 1 Tag 1 Stunde |

|  |  |
| --- | --- |
| Ticket ID | SCP-14 |
| Epic | Challengeverwaltung |
| User Story | Als Benutzer möchte ich die Spezifikationen wie z.B. die Sportarten einer bereits existierenden Challenge ändern können. |
| Erstellt von | Robin Hackh |
| Priorität | Mittel |
| Zeitschätzung | 6 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-15 |
| Epic | Sportverwaltung |
| User Story | Um neue Anregung für die Challenge zu schaffen, möchte ich als Benutzer mehrere/einzelne temporäre Bonusaktionen für die Challenge erstellen können. |
| Erstellt von | Robin Hackh |
| Priorität | Hoch |
| Zeitschätzung | 1 Tag 4 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-16 |
| Epic | Sportverwaltung |
| User Story | Um eine Challenge fairer zu gestalten, möchte ich als Administrator für Sportarten einen Multiplikator festlegen können. |
| Erstellt von | Robin Hackh |
| Priorität | Mittel |
| Zeitschätzung | 2 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-17 |
| Epic | Teamverwaltung |
| User Story | Um an einer Challenge teilzunehmen, möchte ich als Benutzer ein Team erstellen können, das einer aktiven Challenge zugeordnet wird. |
| Erstellt von | Robin Hackh |
| Priorität | Hoch |
| Zeitschätzung | 6 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-18 |
| Epic | Challengeverwaltung |
| User Story | Um keine Bonusaktionen zu verpassen, möchte ich als Benutzer über anstehende/laufende Boni per Mail informiert werden. |
| Erstellt von | Robin Hackh |
| Priorität | Mittel |
| Zeitschätzung | 1 Tag 4 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-19 |
| Epic | Challengeverwaltung |
| User Story | Um nicht von der Sport Challenge abgelenkt zu werden, möchte ich als Benutzer die Benachrichtigungsmails abmelden können. |
| Erstellt von | Jason Patrick Duffy |
| Priorität | Niedrig |
| Zeitschätzung | 2 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-20 |
| Epic | Challengeverwaltung |
| User Story | Um einen Fortschritt in der Challenge zu erzielen, möchte ich als Benutzer meine zurückgelegten Kilometer in einer Challenge, in der ich teilnehme, eintragen können. |
| Erstellt von | Robin Hackh |
| Priorität | Höchste |
| Zeitschätzung | 2 Tage 5 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-21 |
| Epic | Challengeverwaltung |
| User Story | Um herauszufinden wie mein Team und ich bei der Challenge abschneiden, möchte ich als Benutzer eine mit Charts aufbereitete Seite der Challenges. |
| Erstellt von | Robin Hackh |
| Priorität | Mittel |
| Zeitschätzung | 3 Tage |

|  |  |
| --- | --- |
| Ticket ID | SCP-22 |
| Epic | Challengeverwaltung |
| User Story | Um meine Kilometer nicht jedes mal von Hand eingeben zu müssen, möchte ich als Benutzer diese mit einem Bild meines Tachos eintragen können. |
| Erstellt von | Robin Hackh |
| Priorität | Niedrigste |
| Zeitschätzung | 4 Tage |

|  |  |
| --- | --- |
| Ticket ID | SCP-23 |
| Epic | Benutzerverwaltung |
| User Story | Um mich von anderen Benutzern abzuheben, möchte ich als Benutzer mein Profilbild ändern können. |
| Erstellt von | Robin Hackh |
| Priorität | Niedrig |
| Zeitschätzung | 4 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-24 |
| Epic | Challengeverwaltung |
| User Story | Um Ordnung bei den Challenges zu waren, möchte ich als Administrator, dass nur ausgewählte Benutzer berechtigt sind Spezifikationen der Challenge zu ändern. |
| Erstellt von | Robin Hackh |
| Priorität | Niedrigste |
| Zeitschätzung | 6 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-25 |
| Epic | Benutzerverwaltung |
| User Story | Um mein Profil ansprechender zu gestalten, möchte ich als Benutzer meinem Profil ein Motto hinzufügen können. |
| Erstellt von | Robin Hackh |
| Priorität | Niedrigste |
| Zeitschätzung | 2 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-26 |
| Epic | Benutzerverwaltung |
| User Story | Um mein Profil auf dem aktuellsten Stand zu halten, möchte ich als Benutzer meine Profildaten ändern können. |
| Erstellt von | Robin Hackh |
| Priorität | Mittel |
| Zeitschätzung | 5 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-28 |
| Epic | Sportverwaltung |
| User Story | Um Fehler zu korrigieren, möchte ich als Administrator die Daten einer Sportart bearbeiten können. |
| Erstellt von | Jason Patrick Duffy |
| Priorität | Mittel |
| Zeitschätzung | 4 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-29 |
| Epic | Challengeverwaltung |
| User Story | Um die doppelte Datenerfassung zu ersparen, möchte ich als Benutzer meine Kilometerdaten aus der Strava-App importieren können. |
| Erstellt von | Jason Patrick Duffy |
| Priorität | Niedrig |
| Zeitschätzung | 1 Tag |

|  |  |
| --- | --- |
| Ticket ID | SCP-30 |
| Epic | Challengeverwaltung |
| User Story | Um die Anwendung in Zukunft erweitern zu können, möchte ich als Entwickler bei doubleSlash eine API zum Anbinden weiterer Apps haben. |
| Erstellt von | Jason Patrick Duffy |
| Priorität | Niedrigste |
| Zeitschätzung | 1 Tag 6 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-31 |
| Epic | Benutzerverwaltung |
| User Story | Um mich von allem abzumelden, möchte ich als Benutzer mein Konto vollständig löschen können. |
| Erstellt von | Jason Patrick Duffy |
| Priorität | Niedrig |
| Zeitschätzung | 4 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-32 |
| Epic | Benutzerverwaltung |
| User Story | Um ungewollte Änderungen zu vermeiden, möchte ich als Administrator Benutzergruppen erstellen und ihnen bestimmte Rechte zuweisen können. |
| Erstellt von | Jason Patrick Duffy |
| Priorität | Niedrig |
| Zeitschätzung | 2 Tage 6 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-33 |
| Epic | Benutzerverwaltung |
| User Story | Um die Anwendung verwenden zu können, möchte ich als Benutzer meine bevorzugte Sprache auswählen können. |
| Erstellt von | Jason Patrick Duffy |
| Priorität | Niedrig |
| Zeitschätzung | 6 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-34 |
| Epic | Benutzerverwaltung |
| User Story | Um nachvollziehen zu können, wer welche Änderungen gemacht hat, möchte ich als Administrator eine Übersicht über alle gemachten Änderungen und wer diese getätigt hat haben. |
| Erstellt von | Jason Patrick Duffy |
| Priorität | Niedrigste |
| Zeitschätzung | 1 Tag 4 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-35 |
| Epic | Teamverwaltung |
| User Story | Um die Challenge zu organisieren, möchte ich als Administrator Benutzer aus Teams entfernen können. |
| Erstellt von | Jason Patrick Duffy |
| Priorität | Niedrig |
| Zeitschätzung | 4 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-36 |
| Epic | Teamverwaltung |
| User Story | Um meinen Kollegen einfach in mein Team zu bekommen, möchte ich als Benutzer andere Benutzer in eine Challenge und mein Team einladen können. |
| Erstellt von | Jason Patrick Duffy |
| Priorität | Niedrig |
| Zeitschätzung | 1 Tag |

|  |  |
| --- | --- |
| Ticket ID | SCP-37 |
| Epic | Teamverwaltung |
| User Story | Um Fehler zu korrigieren, möchte ich als Benutzer die Daten meines Teams ändern können. |
| Erstellt von | Jason Patrick Duffy |
| Priorität | Mittel |
| Zeitschätzung | 5 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-38 |
| Epic | Challengeverwaltung |
| User Story | Um die Daten der Challenge weiterverarbeiten zu können, möchte ich als Administrator die Daten der Challenge exportieren können. |
| Erstellt von | Jason Patrick Duffy |
| Priorität | Niedrigste |
| Zeitschätzung | 1 Tag |

|  |  |
| --- | --- |
| Ticket ID | SCP-40 |
| Epic | Teamverwaltung |
| User Story | Um eine faire Challenge zu gestalten, möchte ich als Administrator, dass die Kilometer jedes Teams als Durchschnitt der Mitglieder gerechnet werden. |
| Erstellt von | Tom Nguyen Dinh |
| Priorität | Mittel |
| Zeitschätzung | 1 Stunde 30 Minuten |

|  |  |
| --- | --- |
| Ticket ID | SCP-46 |
| Epic | Teamverwaltung |
| User Story | Um einen besseren Teamgeist zu fördern, möchte ich als Benutzer dem Team gerne einem Namen geben können. |
| Erstellt von | Tom Nguyen Dinh |
| Priorität | Niedrig |
| Zeitschätzung | 2 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-58 |
| Epic | Teamverwaltung |
| User Story | Um eine dynamischere Erfahrung zu haben, möchte ich als Benutzer für jede Challenge ein anderes Team auswählen können. |
| Erstellt von | Tom Nguyen Dinh |
| Priorität | Mittel |
| Zeitschätzung | - |

|  |  |
| --- | --- |
| Ticket ID | SCP-60 |
| Epic | Teamverwaltung |
| User Story | Um mir später Arbeit zu ersparen, möchte ich als Benutzer mich bereits vor Challengebeginn zur Challenge anmelden können. |
| Erstellt von | Tom Nguyen Dinh |
| Priorität | Niedrig |
| Zeitschätzung | 2 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-61 |
| Epic | Teamverwaltung |
| User Story | Um eine faire Challenge zwischen den Teams zu haben, möchte ich als Administrator, dass Teilnehmer ihr Team nicht während einer laufenden Challenge wechseln können. |
| Erstellt von | Tom Nguyen Dinh |
| Priorität | Niedrig |
| Zeitschätzung | - |

|  |  |
| --- | --- |
| Ticket ID | SCP-62 |
| Epic | Sportverwaltung |
| User Story | Bei Veränderungen, wie das neue Hinzukommen einer Sportart, möchte ich als Benutzer eine E-Mail erhalten. |
| Erstellt von | Tom Nguyen Dinh |
| Priorität | Niedrig |
| Zeitschätzung | 2 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-98 |
| Epic | Challengeverwaltung |
| User Story | Als User möchte ich in der Lage sein aus einer Challenge auszusteigen, falls ich bemerke, dass sie doch nichts für mich ist. |
| Erstellt von | Mason Schönherr |
| Priorität | Niedrigste |
| Zeitschätzung | 4 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-99 |
| Epic | Challengeverwaltung |
| User Story | Als aktiver User möchte in der Lage sein meine Aktivitäten auf mehreren Seiten der Website eintragen zu können, um beim Hin- und Herklicken Zeit zu sparen. |
| Erstellt von | Mason Schönherr |
| Priorität | Niedrig |
| Zeitschätzung | 1 Tag |

|  |  |
| --- | --- |
| Ticket ID | SCP-100 |
| Epic | Challengeverwaltung |
| User Story | Als User möchte ich eine graphische Historie meiner Aktivitäten sehen können. |
| Erstellt von | Mason Schönherr |
| Priorität | Niedrig |
| Zeitschätzung | 1 Tag |

|  |  |
| --- | --- |
| Ticket ID | SCP-103 |
| Epic | Benutzerverwaltung |
| User Story | Um weniger Probleme bei der Nutzung zu haben, möchte ich als Benutzer meine bevorzugten Einheiten (Kilometer, Meilen, Yards, ...) auswählen können. |
| Erstellt von | Jason Patrick Duffy |
| Priorität | Niedrigste |
| Zeitschätzung | 6 Stunden |

|  |  |
| --- | --- |
| Ticket ID | SCP-109 |
| Epic | Challengeverwaltung |
| User Story | Um Fehler zu korrigieren, möchte ich als Benutzer in der Lage sein, meine gutgeschriebenen Kilometer zu widerrufen oder zu ändern. |
| Erstellt von | Mason Schönherr |
| Priorität | Mittel |
| Zeitschätzung | 1 Tag |

|  |  |
| --- | --- |
| Ticket ID | SCP-110 |
| Epic | Sportverwaltung |
| User Story | Um die Challenge weiter personalisieren zu können, möchte ich als Administrator für jede Sportart einen eigenen Multiplikator auswählen können. |
| Erstellt von | Mason Schönherr |
| Priorität | Hoch |
| Zeitschätzung | 3 Stunden |

# 4 UI-Entwurf

In diesem Abschnitt wird unser Prototyp vorgestellt. Außerdem findet sich hier eine Tabelle der vom Kunden verwendeten Farben.

**Farbcodes**

|  |  |
| --- | --- |
| **Farbe** | **Farbcode** |
|  | Markenfarbe: #00A5E1 |
|  | Sekundär Unternehmensfarbe: #D7E9F4 |
|  | Sekundär Orange: #ff9f00 |
|  | Schrift Dunkelgrau: #515151 |
|  | Sekundär Dunkelgrau: #7A7A7A |
|  | Sekundär Hellgrau: #C6C6C6 |
|  | Akzentfarbe: #32547B |
|  | Akzentfarbe: #B9D478 |
|  | Akzentfarbe: #8B593E |
|  | Akzentfarbe: #C63328 |

Aus unserer Ausarbeitung der Personas haben wir folgende wichtige Punkte zur Usability ermittelt:

1. Diverses Benutzerfeld erfordert intuitive und einfache Anwendung.
2. Dateneingabe darf nicht zu schwer aufzufinden sein oder zu lange dauern.
3. Kooperative und kompetitive Aspekte müssen zentral sein.
4. Verschiedene Wege der Sportaufzeichnung benötigen Verbindung mit APIs unterschiedlicher Plattformen.

Die Umsetzung dieser Punkte wird durch Abstimmung mit dem Kunden überprüft.

## 4.1 Prototyp

Den Prototyp unserer Anwendung haben wir mithilfe von Figma erstellt. Die durch die Benutzeranalyse in den Personas erkannten Anforderungen sind hier umgesetzt.

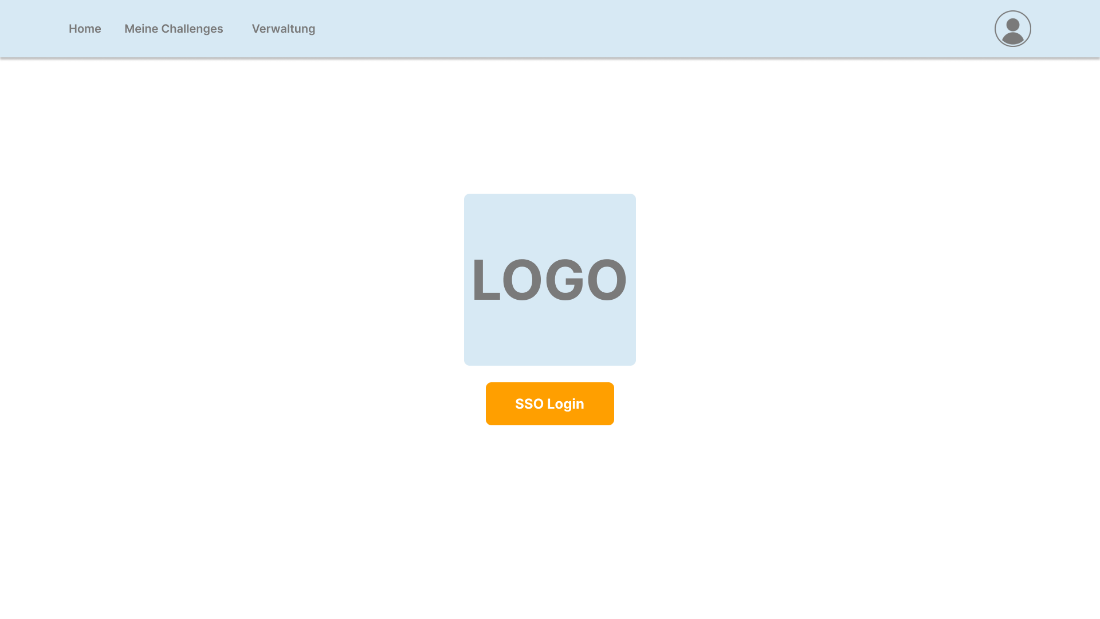


Abbildung 8: Die Loginseite ist die erste geöffnete Seite

Da die Anmeldung über SSO erfolgt, reicht eine einfache Loginseite mit einem einzelnen Knopf, der auf den Identity Provider weiterleitet.

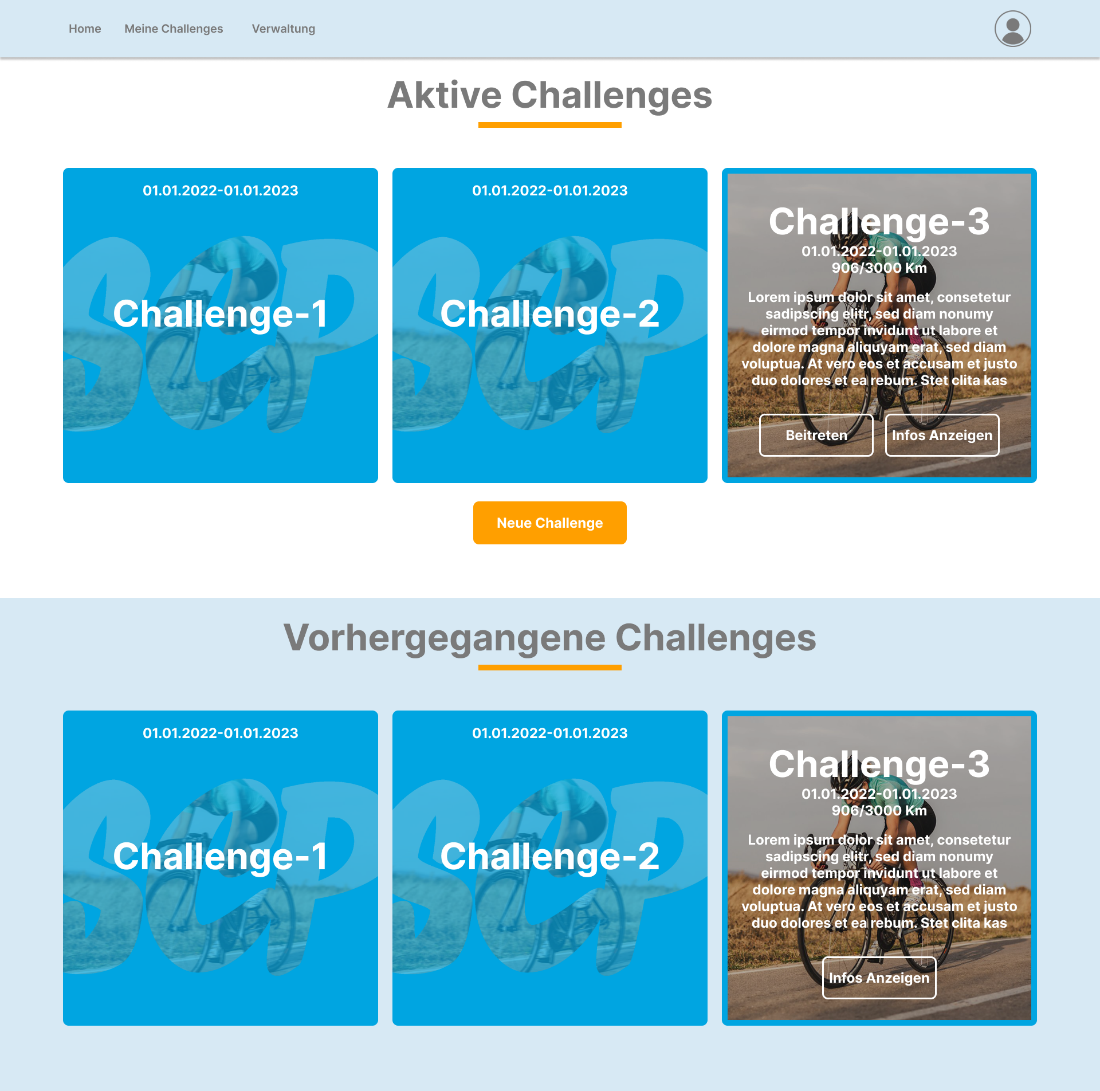


Abbildung 9: Die Startseite wird nach dem Login geöffnet

Große Anzeigekacheln sollen direkt die Aufmerksamkeit des Benutzers fangen und sofort über die laufenden und vergangen Challenges informieren.

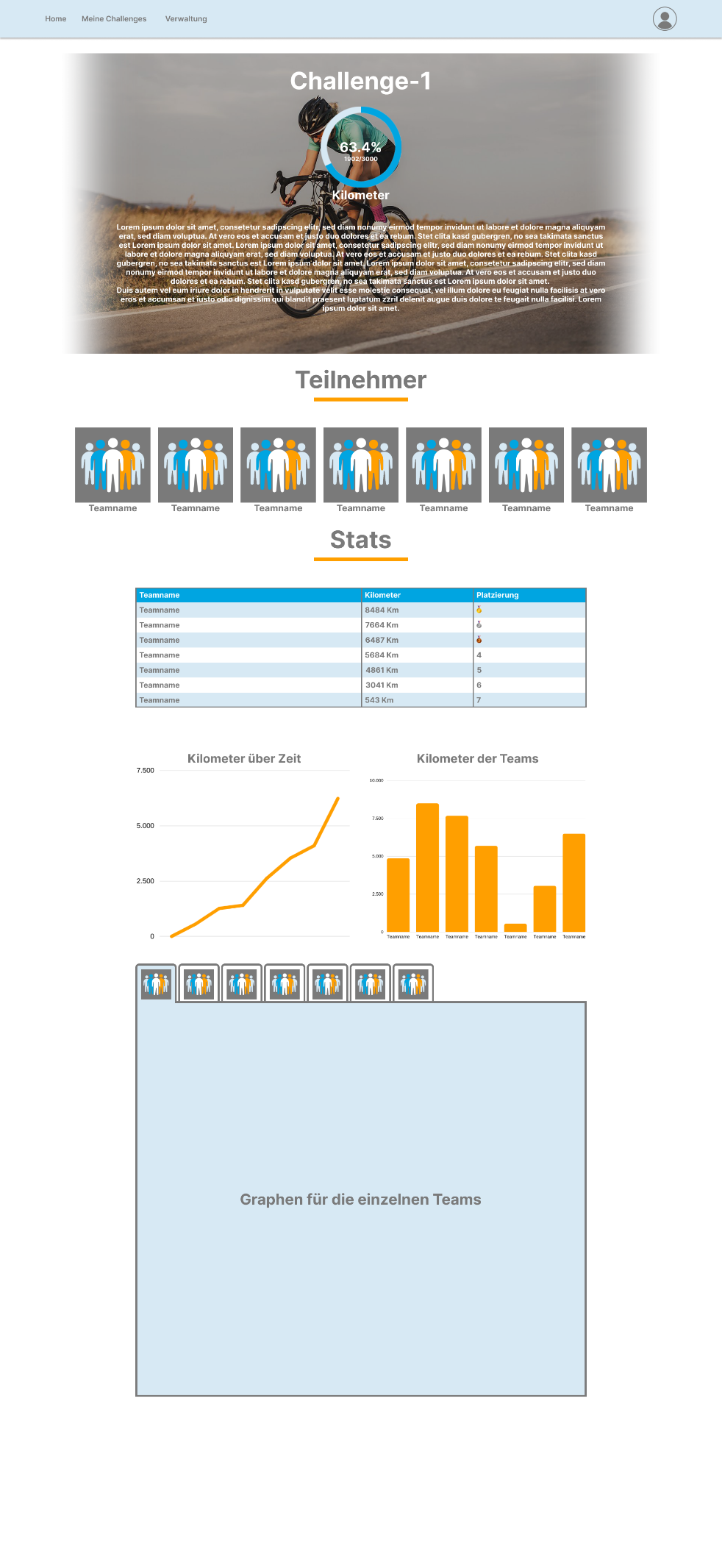


Abbildung 10: Die Seite einer einzelnen Challenge

Auf der Challengeseite wird der Verlauf der Challenge grafisch aufgearbeitet. So werden dem Benutzer schnell verschiedene Informationen mitgeteilt. Zudem kann man sich verschiedene Graphen für die unterschiedlichen Teams ansehen. So soll der kooperative und kompetitive Aspekt gefördert werden.

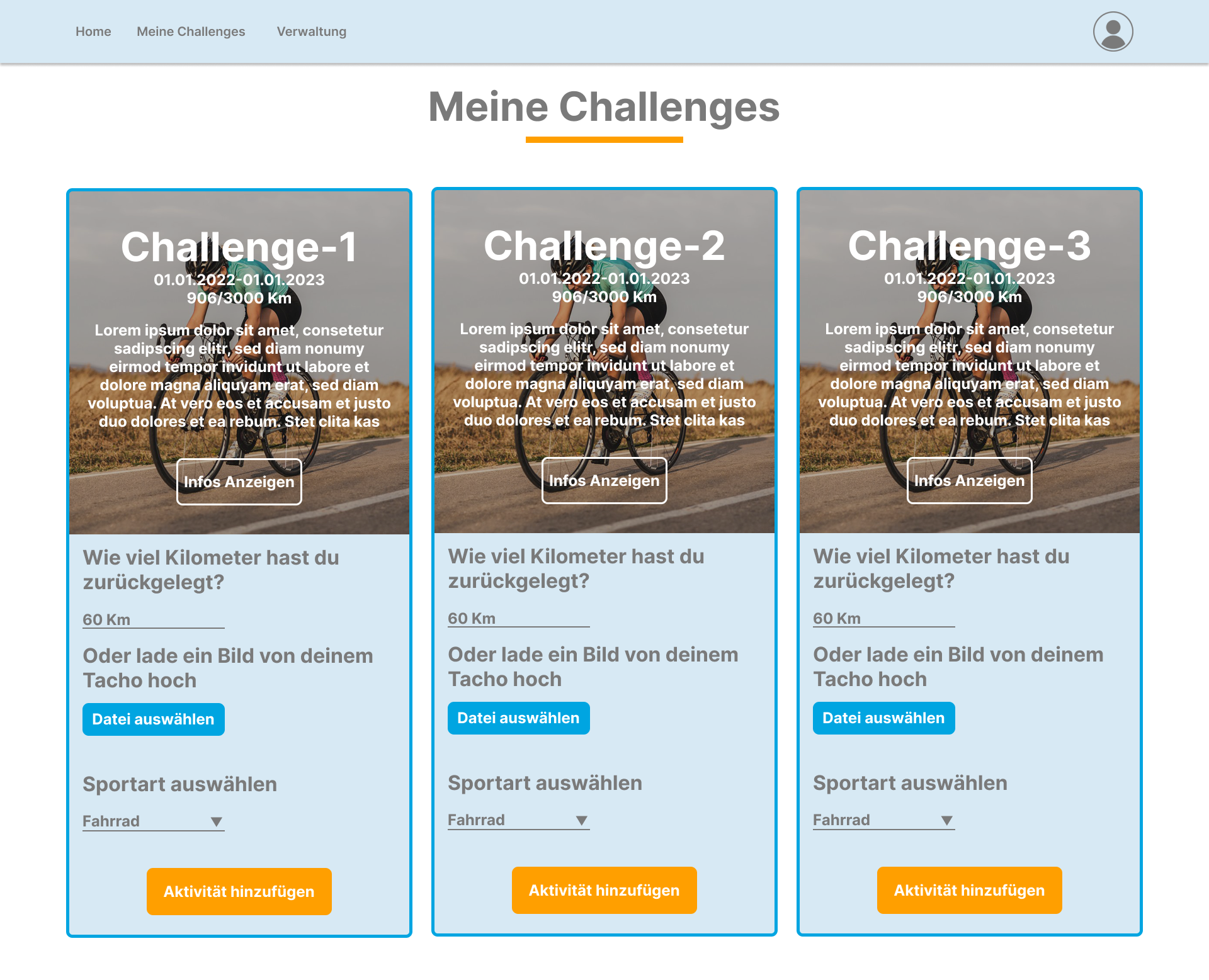


Abbildung 11: Die „Meine Challenges“-Seite erlaubt die Eingabe von Daten

Auf der „Meine Challenges“-Seite erhält man direkt einen Überblick über alle Challenges, an denen man teilnimmt. So kann man in kurzer Zeit die Eingabe für verschiedene Challenges durchführen.

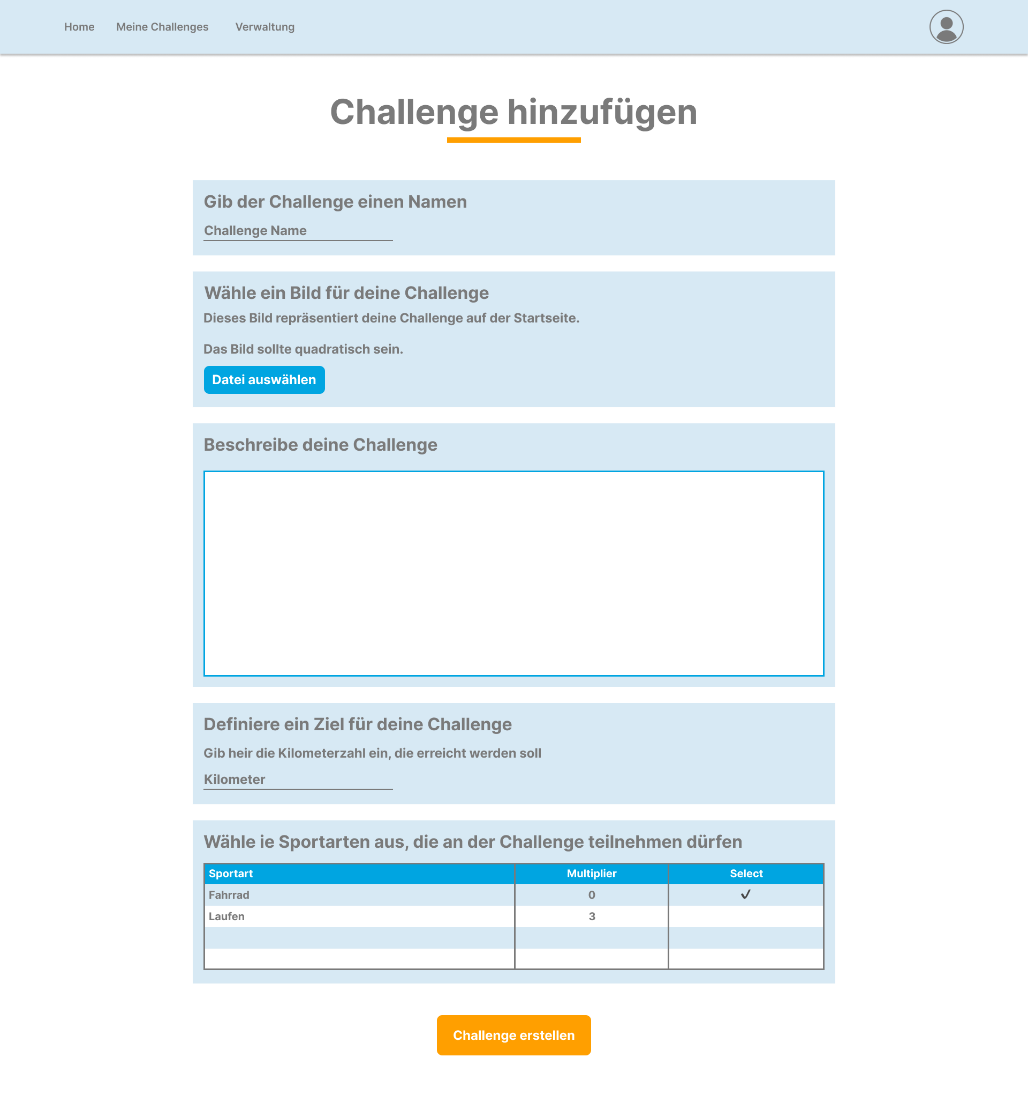


Abbildung 12: Eine neue Challenge erstellen

Die Dateneingabe für eine neue Challenge erfolgt auf einer einfachen Seite mit allen erforderlichen Daten aufgelistet.

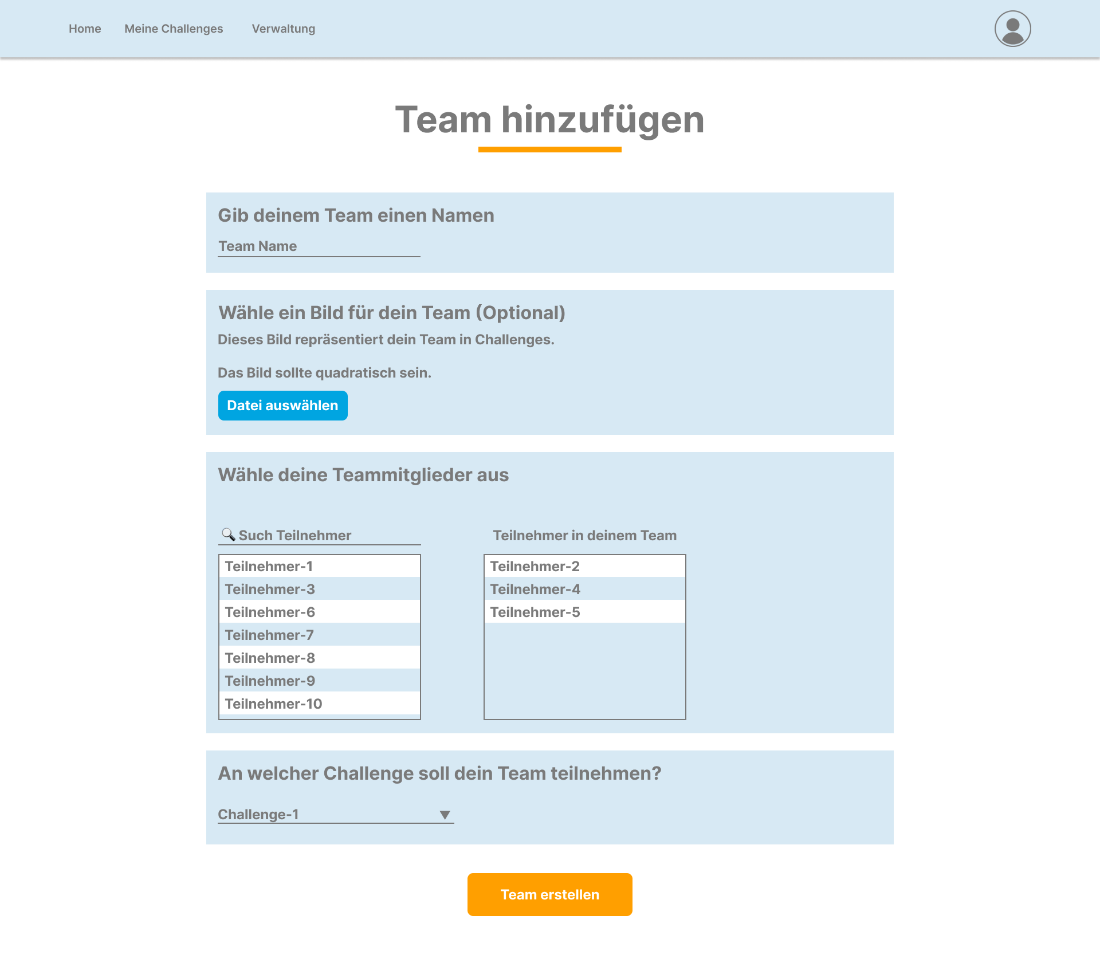


Abbildung 13: Ein neues Team erstellen

Wie beim Erstellen einer Challenge, erfolgt das Erstellen eines neuen Teams ebenfalls auf einer einfachen Seite, auf der man alle erforderlichen Daten eingeben kann. Hier kann man zudem Teilnehmer bereits zum Team hinzufügen.

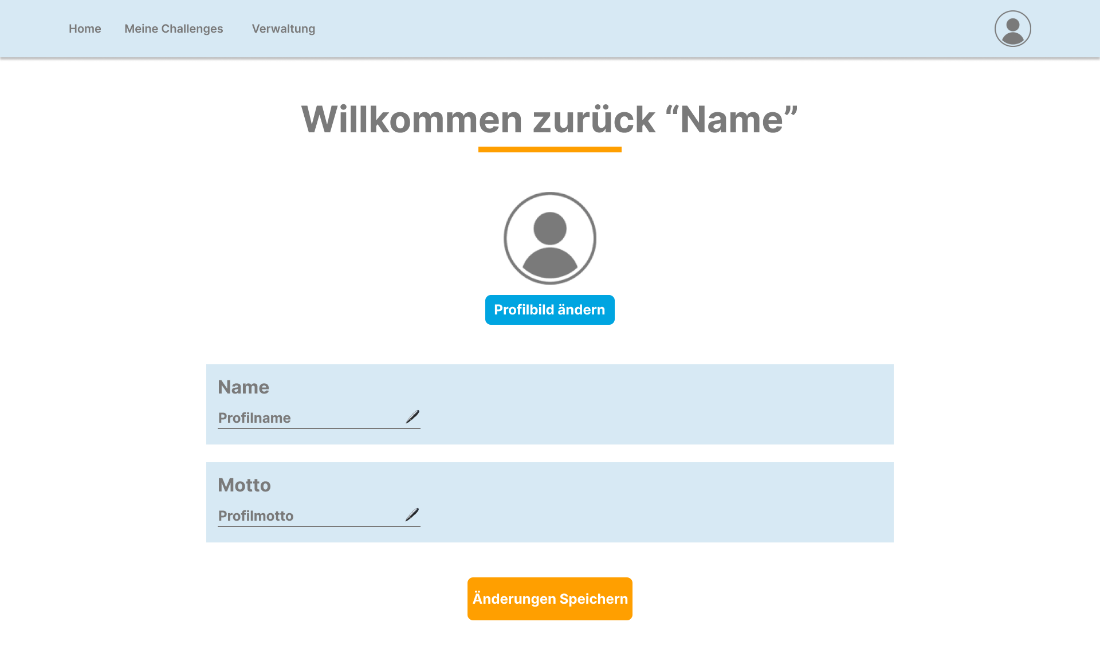


Abbildung 14: Das Benutzerprofil erlaubt das Bearbeiten der Nutzerdaten

Schließlich kann man auf Wunsch die eigenen Daten des Profils bearbeiten.

# 5 Architektur

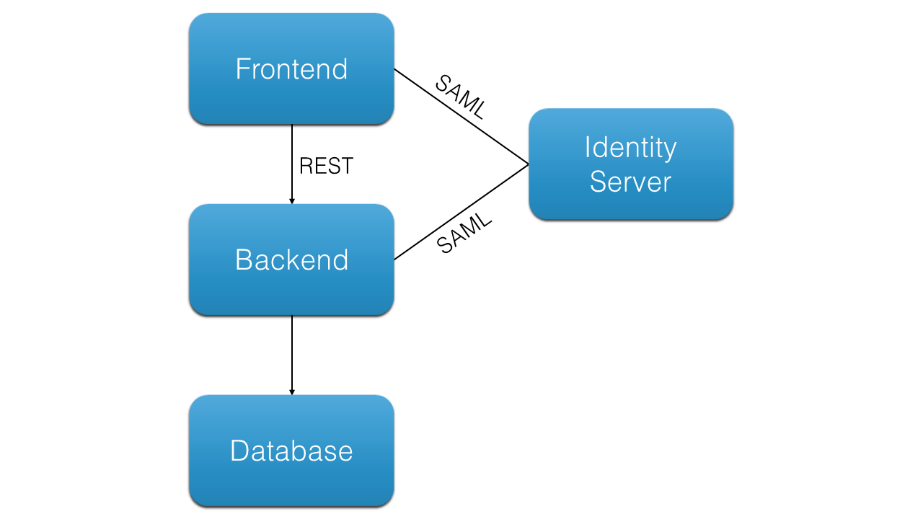


Abbildung 15: Grundlegende Architektur

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 daher 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 Funktionsvielfalt von PostgreSQL die spätere Erweiterung der Anwendung mit komplexeren Aufgabenstellungen – es ist also zukunftsorientiert.

Daraus ergibt sich folgende Architektur:

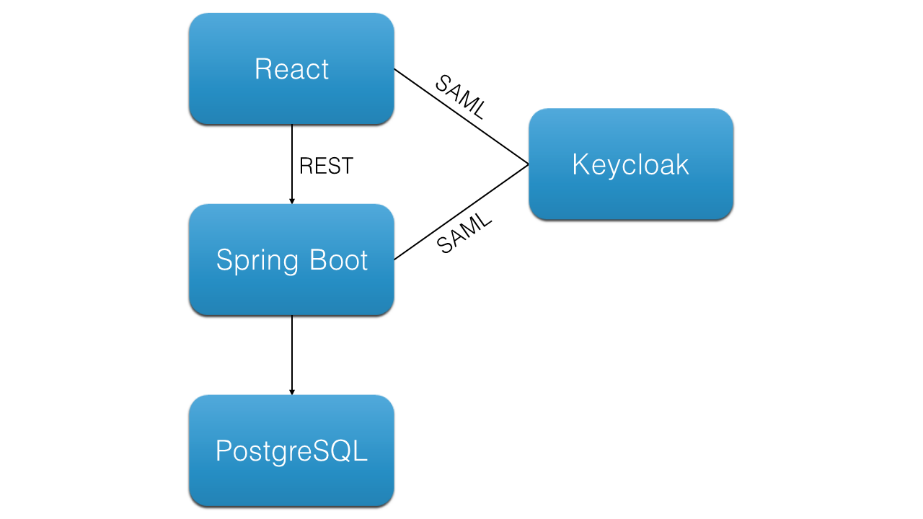


Abbildung 16: Architektur mit den von uns gewählten Frameworks

## 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 [3] 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 der Anforderung F2 entspricht.

### 5.1.2 React

React wird ebenfalls mit der MIT Lizenz ausgeliefert [4]. 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 [5]. 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 lizenzieren 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 [6]. 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 [7]. 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.

## 5.2 Logischer Aufbau

Ein Bild, das Text, Diagramm, Rechteck, Plan enthält.

Automatisch generierte Beschreibung

Abbildung 21: Komponentendiagramm der Anwendung

Die vier Komponenten befinden sich jeweils in ihren eigenen Docker-Containern. Die Komponenten kommunizieren im Docker miteinander. Alle Komponenten laufen dabei zusammen in einem Docker-Compose.

## 5.3 Verteilungsansicht

Ein Bild, das Screenshot, Text, Design enthält.

Automatisch generierte Beschreibung

Abbildung 22: Verteilungsdiagramm der Anwendung

Der in [5.6 Logischer Aufbau](#_5.6_Logischer_Aufbau) angesprochene Aufbau wird vom Benutzer über HTTP über das Internet angesteuert. Das React Frontend läuft über einen Node.js-Server auf dem Port 3000. Die Authentifizierung über Keycloak läuft über den Port 8080. Das Backend wird über Apache Tomcat auf Port 8081 gehostet. Schließlich läuft die PostgreSQL-Datenbank auf Port 5432.

## 5.4 Datenhaltung

Ein Bild, das Text, Screenshot, Schrift, Design enthält.

Automatisch generierte Beschreibung

Abbildung 19: Datenhaltungsdiagramm der Anwendung

Zum Einloggen sendet der Client erst seine Logindaten an den Identity Provider. Dort wird der Client anhand der internen Datenbank authentifiziert. Der Identity Provider sendet dann an den Client einen Cookie mit der Session zurück. Die Session wird beim Server vom Identity Provider registriert. Anhand dieser Registrierung kann der Server dann den Benutzer selbst authentifizieren. Der Server fragt zudem die Benutzerdaten der Session beim Identity Provider an.

Auf dem Client wird nur ein Cookie mit Session-Informationen gespeichert. Diesen Cookie sendet der Client dann zusammen mit der Anfrage zum Server. Anhand des Cookies authentifiziert der Server den Client und führt dann die Anfrage aus. Danach sendet der Server das Ergebnis der Anfrage an den Client zurück.

## 5.5 ER-Modell der Datenbank

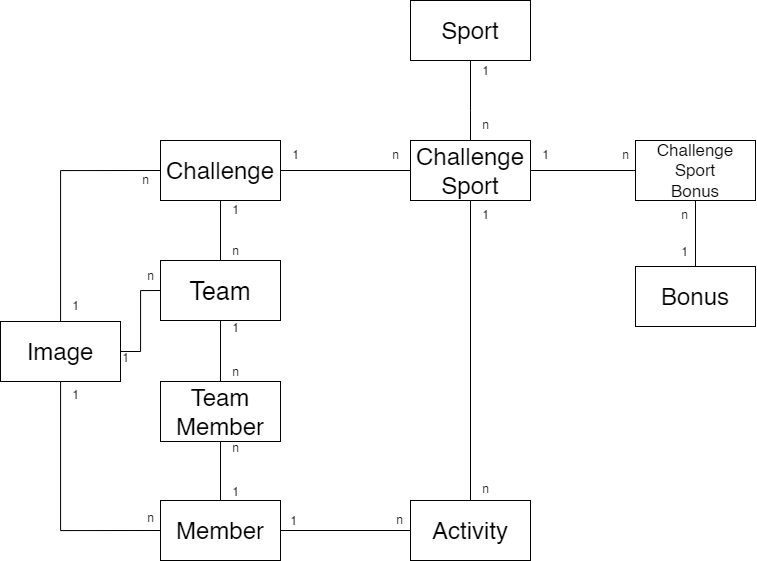


Abbildung 17: ER-Diagramm der Datenbankstruktur

Die Datenbank besteht aus den folgenden Hauptentitäten: Challenge, Team, Image, Member, Activity, Sport und Bonus. Dazu kommen noch drei Zwischentabellen, die zum Auflösen von n:m-Beziehungen verwendet werden: Challenge Sport, Challenge Sport Bonus und Team Member.

In der Challenge-Tabelle werden alle für eine einzelne Challenge benötigten Daten gespeichert. Dazu gehören die Start- und Enddaten, der Name, die Beschreibung, die Zielpunktzahl und das dazugehörige Bild als Foreign Key.

Bei der Sport-Tabelle wird der Name der Sportart und der bei der Challenge-Erstellung voreingestellte Faktor gespeichert.

Die Challenge Sport-Zwischentabelle löst die n:m-Beziehung zwischen der Challenge- und Sport-Tabelle auf und beinhaltet zudem noch den challengespezifischen Faktor für die Sportart. Darin werden also alle für die Challenge relevanten Sportarten mitsamt ihres Faktors gespeichert.

In der Team-Tabelle wird der Name eines Teams und die dazugehörige Challenge und das Bild als Foreign Keys gespeichert.

Die Member-Tabelle beinhaltet die Daten für alle Benutzer der Anwendung. Dazu gehören die E-Mail-Adresse, der Vorname, der Nachname, die Zustimmung zur E-Mail-Kommunikation und das Bild als Foreign Key.

Die Team Member-Tabelle löst die n:m-Beziehung zwischen der Team- und Member-Tabelle auf. Sie beinhaltet keine weiteren Daten.

In der Image-Tabelle werden die Bilddaten, der Bildname und der Datentyp des Bildes gespeichert. Die Tabelle beinhaltet alle in der Anwendung hochgeladenen Bilder.

Aktivitäten werden in der Activity-Tabelle gespeichert. Die Daten darin beinhalten das Datum, die eingetragene Distanz, die zum Zeitpunkt der Eintragung berechnete Gesamtpunktzahl sowie die dazugehörige Challenge Sport und der Member als Foreign Keys.

Die Bonus-Tabelle beinhaltet eine Beschreibung, die Start- und Enddaten, den Faktor und den Name des Bonuses.

Schließlich wird die Challenge Sport Bonus-Tabelle zur Auflösung der n:m-Beziehung zwischen der Bonus- und Challenge Sport-Tabelle verwendet.

## 5.6 Klassenstruktur der Datenbankentitäten

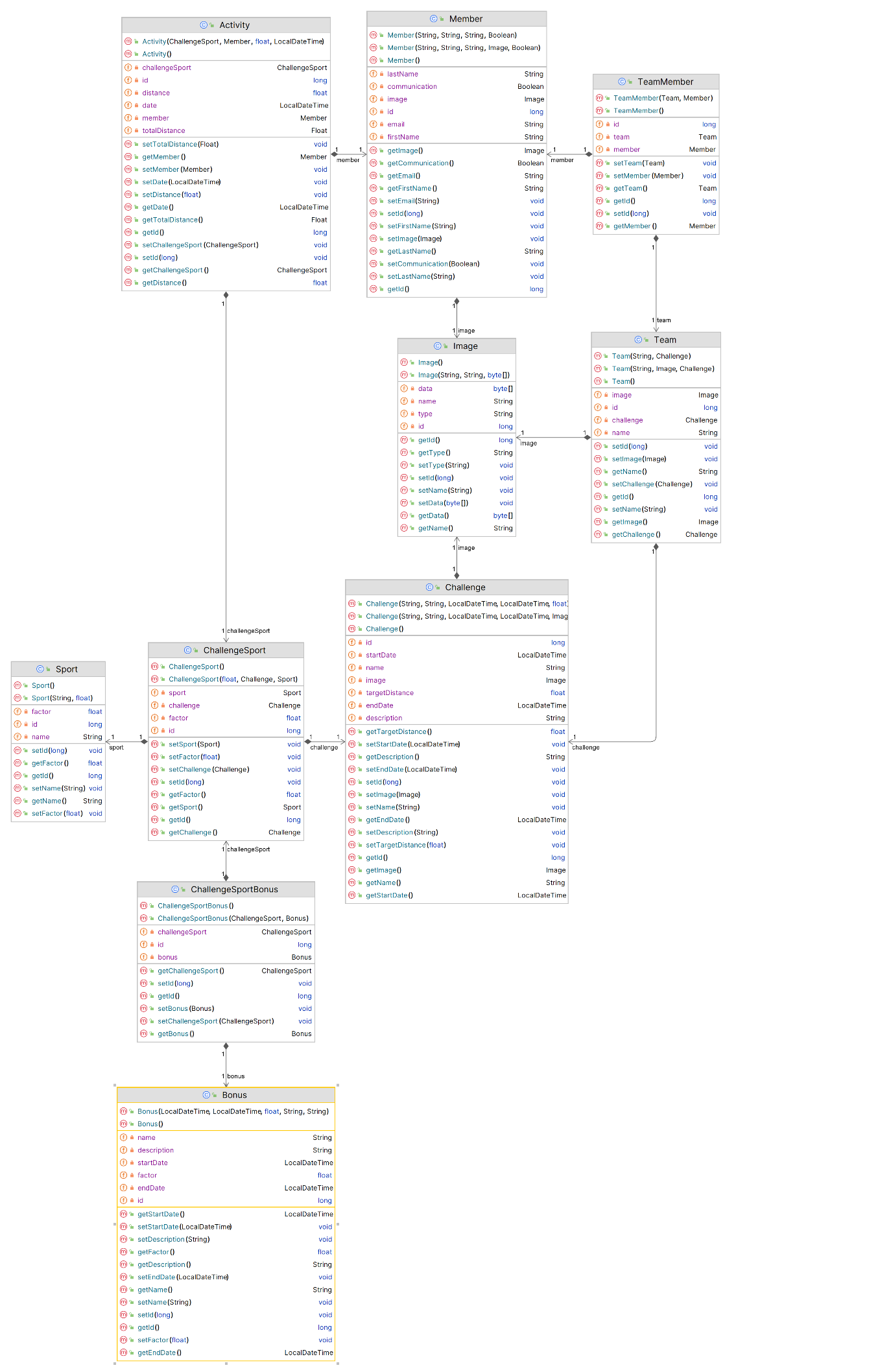


Abbildung 18: Klassendiagramm der Entitäten

Die in Kapitel [5.2 ER-Modell der Datenbank](#_5.2_ER-Modell_der) beschriebene Datenbankstruktur wurde im Projekt umgesetzt, wo es mittels Hibernate und JPA in die PostgreSQL-Datenbank geschrieben wird und darüber auch abrufbar ist. Im oberen Diagramm kann man nun die genauen Bezeichnungen der Datenbankelemente und deren Datentypen erkennen.

## 5.7 Ablauf einer Challenge

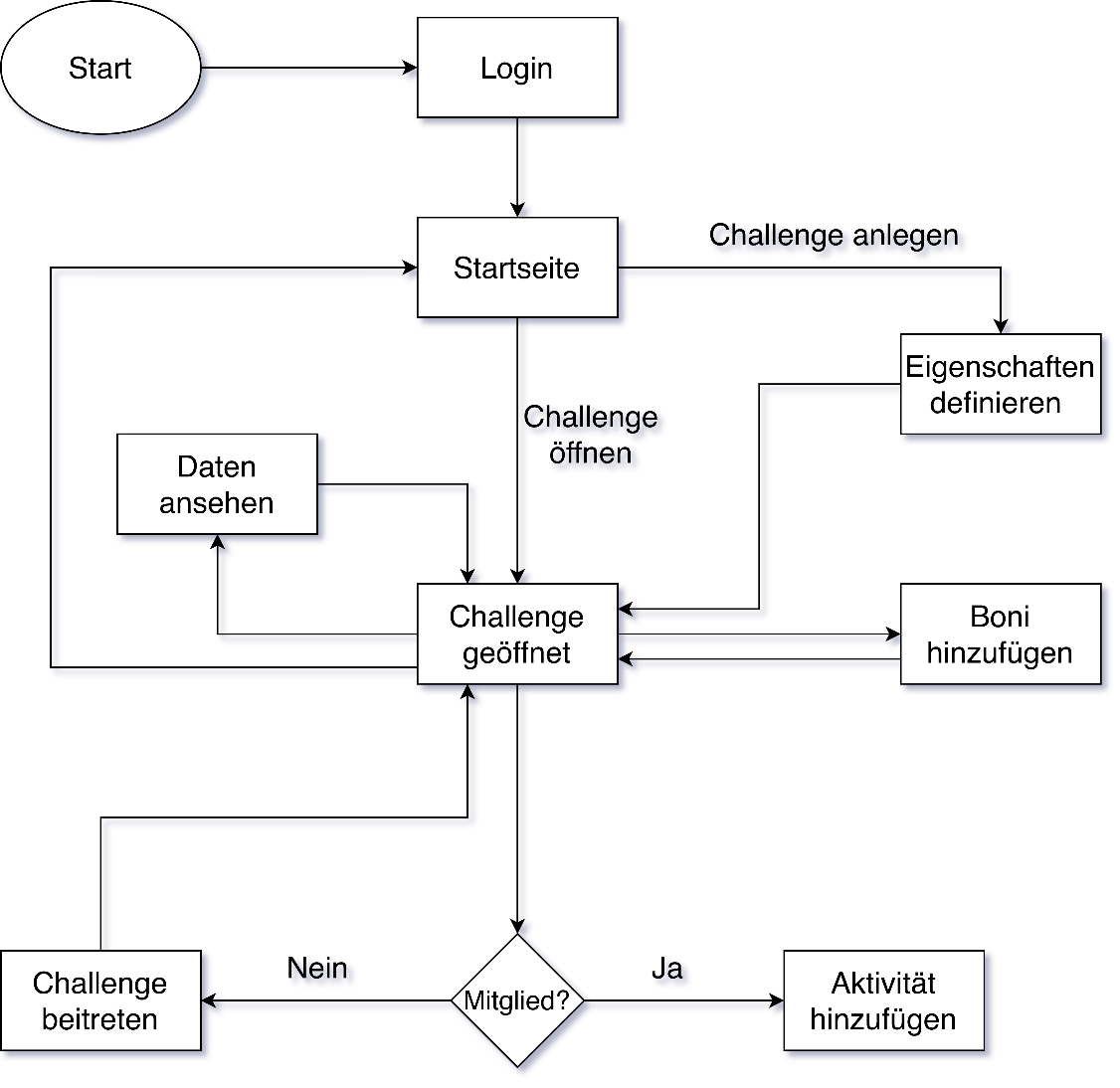


Abbildung 20: Ablaufdiagramm einer einzelnen Challenge

Der Ablauf einer einzelnen Challenge sieht folgendermaßen aus: Der Benutzer loggt sich ein, woraufhin der zur Startseite geleitet wird. Von dort kann der Benutzer dann entweder eine Challenge öffnen und die Daten einsehen oder eine neue Challenge anlegen. In der geöffneten Challenge kann er zudem einen Bonus hinzufügen. Ist er bereits Mitglied der Challenge, kann er Aktivitäten hinzufügen; andererseits kann er der Challenge beitreten.

# 6 Aufwandschätzung

Für die Sprintplanung haben wir darauf geachtet, dass jeder Sprint eine Arbeitszeit von etwa vier Tagen aufweist. Das entspricht pro Woche und pro Teammitglied etwa einem Arbeitstag, also acht Stunden.

So stellen wir sicher, dass wir etwas Überschuss bei der Umsetzung haben, falls wir auf unvorhergesehene Probleme stoßen sollten.

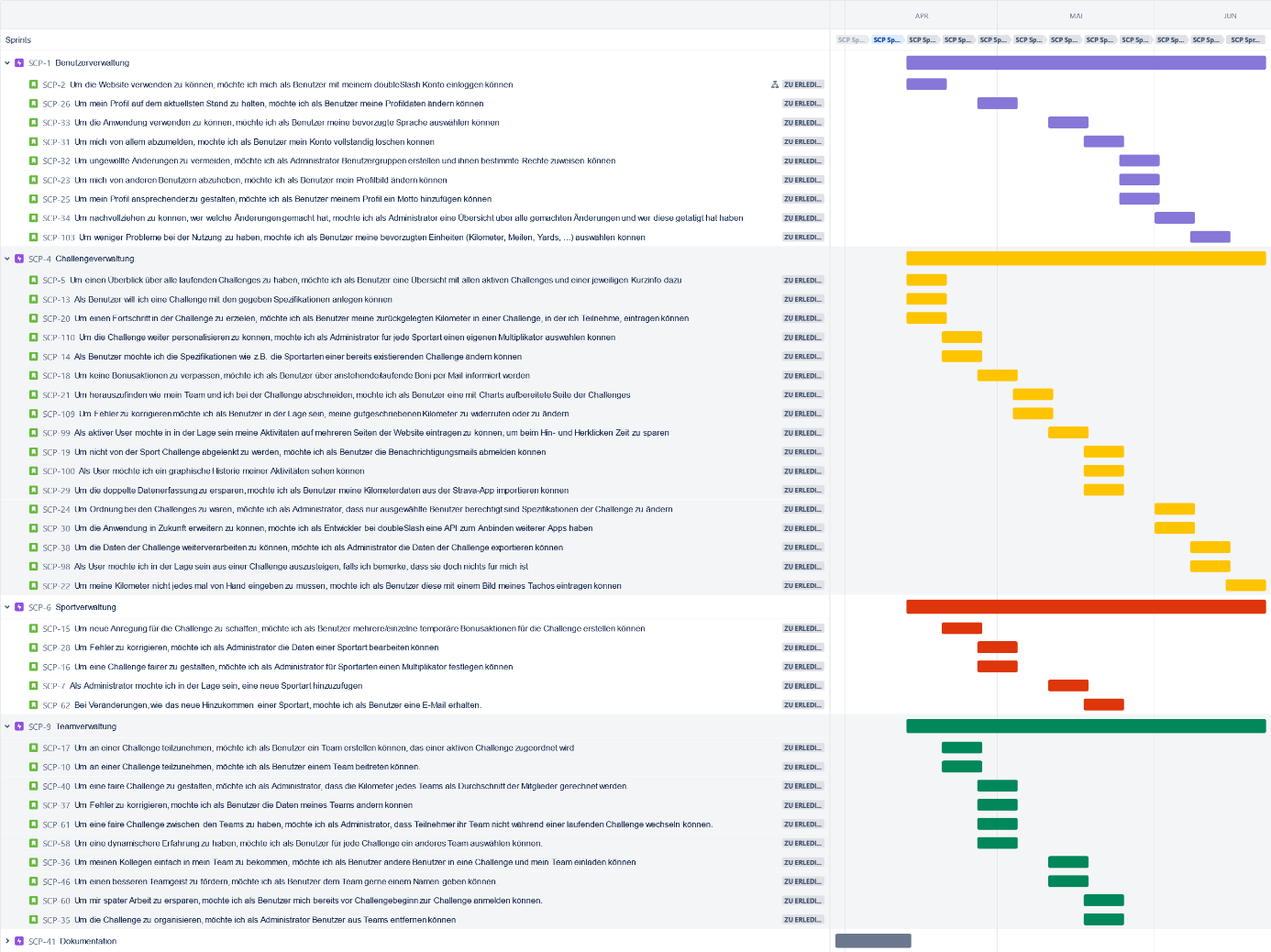


Abbildung 23: Vorläufige Sprintplanung zum Stand 13. April 2023

Rechnet man die Zeitschätzungen der [User Stories](#_3.1_User_Stories) zusammen, kommen wir so auf einen Arbeitsaufwand von *290.5* Stunden, was einer wöchentlichen Arbeitszeit von etwa *29* Stunden und *7* Stunden pro Person entspricht.

Rechnet man dazu noch wöchentlich folgende Punkte pro Teammitglied:

* Treffen mit dem Betreuer: *0.75* Stunden
* Sprint Review: *1* Stunde
* Sprint Retrospektive: *0.75* Stunden
* Sprint Planung: *1* Stunde
* Stand-up-Meetings: *2 \* 0.5* Stunden

Ergibt sich über zehn Wochen ein wöchentlicher Arbeitsaufwand von etwa *12* Stunden pro Teammitglied.

Es ist allerdings wichtig zu beachten, dass dieser Zeitplan die Umsetzung aller User Stories beinhaltet. Also auch die, die nicht direkt Teil der Kundenforderungen sind. Rechnet man nur mit den explizit geforderten Funktionen, kommt man auf eine Arbeitszeit von *98.5* Stunden, was einem wöchentlichen Arbeitsaufwand von etwa *7* Stunden pro Person entspricht.

# 7 API-Dokumentation

Wir bieten für alle Datenbankentitäten und die SAML-Verbindung entsprechende REST-Schnittstellen an. Diese Schnittstellen erlauben das Erstellen und Bearbeiten sowie Löschen von Datenbankobjekten, jedoch auch komplexere Vorgänge, wie das Berechnen von Distanzen und können über das Frontend oder Swagger-UI angesteuert werden.

## 7.1 Member

|  |  |
| --- | --- |
| **Zugriff** | GET /members/{id}/ |
| **Beschreibung** | Gibt die Benutzerdaten des Members mit der gegebenen E-Mail-Adresse zurück. |
| **Parameter** | UserID (long) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * email (String): E-Mail Adresse * firstName (String): Vorname des Bentuzers * lastName (String): Nachname des Benutzers * userid (long): ID des Benutzers * ImageID (long): ID des Bildes |
| **Statuscodes** | 200 – Member gefunden  403 – Nicht eingeloggt  404 – Member nicht gefunden |
| **Beispielantwort** | {  "email": "John@example.com",  "firstName": "John",  "lastName": "Doe",  “userID”: 1,  “imageID”: 1  } |

|  |  |
| --- | --- |
| **Zugriff** | PUT /members/{id}/ |
| **Beschreibung** | Bearbeitet die Benutzerdaten des gegebenen Members. |
| **Parameter** | * E-Mail-Adresse (String) * Vorname (String) * Nachname (String) * ImageID (long) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * email (String): E-Mail Adresse * firstName (String): Vorname des Bentuzers * lastName (String): Nachname des Benutzers * userid (long): ID des Benutzers * ImageID (long): ID des Bildes |
| **Statuscodes** | 200 – Member geupdated  403 – Nicht eingeloggt  404 – Member nicht gefunden |
| **Beispielantwort** | {  "email": "John@example.com",  "firstName": "John",  "lastName": "Doe",  “userID”: 1,  “imageID”: 1  } |

|  |  |
| --- | --- |
| **Zugriff** | POST /members/ |
| **Beschreibung** | Fügt einen neuen Member hinzu. |
| **Parameter** | * E-Mail-Adresse (String) * Vorname (String) * Nachname (String) * ImageID (Long): ID des Bildes |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * email (String): Eindeutige ID des Benutzers * firstName (String): Vorname des Benutzers * lastName (String): Nachname des Benutzers * userid (long): ID des Benutzers * ImageID (long): ID des Bildes |
| **Statuscodes** | 201 – Member hinzugefügt  403 – Nicht eingeloggt  400 – Benutzer existiert bereits oder das Bild kann nicht gefunden werden. |
| **Beispielantwort** | {  "email": "Test@example.com",  "firstName": "Max",  "lastName": "Mustermann",  “userID”: 1,  “imageID”: 1  } |

|  |  |
| --- | --- |
| **Zugriff** | DELETE /members/ |
| **Beschreibung** | Löscht alle Members. |
| **Parameter** | - |
| **Antwort** | - |
| **Statuscodes** | 200 – Member gelöscht  403 – Nicht eingeloggt  500 – Member konnte nicht gelöscht werden |
| **Beispielantwort** | - |

|  |  |
| --- | --- |
| **Zugriff** | DELETE /members/{id}/ |
| **Beschreibung** | Löscht einen Member. |
| **Parameter** | UserID (long) |
| **Antwort** | - |
| **Statuscodes** | 200 – Member gelöscht  403 – Nicht eingeloggt  404 – Member konnte nicht gefunden werden |
| **Beispielantwort** | - |

|  |  |
| --- | --- |
| **Zugriff** | GET /members/{id}/challenges/{chID}/rawDistance/ |
| **Beschreibung** | Gibt die Distanz eines Benutzers in einer Challenge ohne eingerechnete Boni zurück. |
| **Parameter** | * id (Long): ID des Members * chID (Long): ID der Challenge |
| **Antwort** | Text mit der gerechneten Zahl als Float. |
| **Statuscodes** | 200 – Distanz wurde erfolgreich errechnet  403 – Nicht eingeloggt  500 – Nicht alle Aktivitäten sind Teil derselben Challenge |
| **Beispielantwort** | 12 |

|  |  |
| --- | --- |
| **Zugriff** | GET /members/{id}/challenges/{chID}/distance/ |
| **Beschreibung** | Gibt die Distanz eines Benutzers in einer Challenge ohne eingerechnete Boni zurück. |
| **Parameter** | * id (Long): ID des Members * chID (Long): ID der Challenge |
| **Antwort** | Text mit der gerechneten Zahl als Float. |
| **Statuscodes** | 200 – Distanz wurde erfolgreich errechnet  403 – Nicht eingeloggt  500 – Nicht alle Aktivitäten sind Teil derselben Challenge |
| **Beispielantwort** | 24 |

|  |  |
| --- | --- |
| **Zugriff** | GET /members/{id}/activities/ |
| **Beschreibung** | Gibt alle Aktivitäten des gegebenen Benutzers zurück. |
| **Parameter** | * id (ID des Members) |
| **Antwort** | JSON-Liste mit folgenden Eigenschaften:   * id (Long): Eindeutige ID der Aktivität * challengeSportID (Long): Eindeutige ID der Challenge Sport * memberID (Long): Eindeutige ID des Members * distance (Float): Eingetragene Distanz der Aktivität * date (LocalDateTime): Datum der Eintragung der Aktivität |
| **Statuscodes** | 200 – Distanz wurde erfolgreich errechnet  403 – Nicht eingeloggt  404 – Keine Aktivitäten gefunden |
| **Beispielantwort** | [  {  "id": 1,  "challengeSportID": 1,  "memberID": 1,  "distance": 4,  "date": "04.05.2023,12:00"  }  ] |

|  |  |
| --- | --- |
| **Zugriff** | GET /members/loggedIn/ |
| **Beschreibung** | Gibt die Benutzerdaten des eingeloggten Members zurück |
| **Parameter** | - |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * email (String): Eindeutige ID des Benutzers * firstName (String): Vorname des Bentuzers * lastName (String): Nachname des Benutzers * userid (long): ID des Benutzers * ImageID (long): ID des Bildes |
| **Statuscodes** | 200 – Member gefunden  403 – Nicht eingeloggt  404 – Member konnte nicht gefunden werden |
| **Beispielantwort** | {  "email": "Test@example.com",  "firstName": "Max",  "lastName": "Mustermann",  “userID”: 1,  “imageID”: 1  } |

## 7.2 Image

|  |  |
| --- | --- |
| **Zugriff** | POST /images/{id}/ |
| **Beschreibung** | Lädt ein neues Bild hoch und fügt es in die Datenbank hinzu. |
| **Parameter** | Datei (Binär) |
| **Antwort** | - |
| **Statuscodes** | 200 – Datei erfolgreich hochgeladen  403 – Nicht eingeloggt  417 – Datei konnte nicht gespeichert werden |
| **Beispielantwort** | - |

|  |  |
| --- | --- |
| **Zugriff** | GET /images/{id}/ |
| **Beschreibung** | Lädt das in der Datenbank gespeicherte Bild mit der gegebenen ID runter. |
| **Parameter** | id (Long) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (Long): Eindeutige ID des Bildes * name (String): Name des Bildes * type (String): Datentyp des Bildes * data (String): Bild als Base64-String |
| **Statuscodes** | 200 – Datei gefunden  403 – Nicht eingeloggt  404 – Datei nicht gefunden |
| **Beispielantwort** | {  "id": 1,  "name": "userImage.png",  "type": "image/png",  "data": "DATA"  } |

|  |  |
| --- | --- |
| **Zugriff** | GET /images/ |
| **Beschreibung** | Lädt alle in der Datenbank gespeicherte Bild runter. |
| **Parameter** | - |
| **Antwort** | JSON-Liste mit folgenden Eigenschaften:   * id (Long): Eindeutige ID des Bildes * name (String): Name des Bildes * type (String): Datentyp des Bildes * data (String): Bild als Base64-String |
| **Statuscodes** | 200 – Datei gefunden  403 – Nicht eingeloggt |
| **Beispielantwort** | [  {  "id": 1,  "name": "userImage.png",  "type": "image/png",  "data": "DATA"  },  {  "id": 2,  "name": "teamImage.png",  "type": "image/png",  "data": "DATA"  } ] |

|  |  |
| --- | --- |
| **Zugriff** | PUT /images/{id}/ |
| **Beschreibung** | Updates ein Bild in der Datenbank |
| **Parameter** | id (Long) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (Long): Eindeutige ID des Bildes * name (String): Name des Bildes * type (String): Datentyp des Bildes * data (String): Bild als Base64-String |
| **Statuscodes** | 200 – Datei gefunden  403 – Nicht eingeloggt  404 – Datei nicht gefunden  417 – Datei konnte nicht gespeichert werden |
| **Beispielantwort** | {  "id": 1,  "name": "userImage.png",  "type": "image/png",  "data": "DATA"  } |

|  |  |
| --- | --- |
| **Zugriff** | DELETE /images/{id}/ |
| **Beschreibung** | Löscht ein Bild mit der gegebenen ID in der Datenbank |
| **Parameter** | id (Long) |
| **Antwort** | - |
| **Statuscodes** | 200 – Datei gefunden  403 – Nicht eingeloggt  404 – Datei nicht gefunden |
| **Beispielantwort** | - |

|  |  |
| --- | --- |
| **Zugriff** | DELETE /images/ |
| **Beschreibung** | Löscht alle Bilder in der Datenbank |
| **Parameter** | - |
| **Antwort** | - |
| **Statuscodes** | 200 – Datei gefunden  403 – Nicht eingeloggt |
| **Beispielantwort** | - |

## 7.3 Sport

|  |  |
| --- | --- |
| **Zugriff** | POST /sports/ |
| **Beschreibung** | Fügt eine neue Sportart hinzu. |
| **Parameter** | * Name (String) * Faktor (Float) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (Long): Eindeutige ID der Sportart * name (String): Name der Sportart * factor (Float): Faktor der Sportart |
| **Statuscodes** | 201 – Sportart erfolgreich hinzugefügt  403 – Nicht eingeloggt |
| **Beispielantwort** | {  "id": 2,  "name": "Testsport",  "factor": 4.7  } |

|  |  |
| --- | --- |
| **Zugriff** | GET /sports/{id}/ |
| **Beschreibung** | Erhalte die Daten der Sportart mit ID {id}. |
| **Parameter** | id (long) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (Long): Eindeutige ID der Sportart * name (String): Name der Sportart * factor (Float): Faktor der Sportart |
| **Statuscodes** | 200 – Sportart gefunden  403 – Nicht eingeloggt  404 – Sportart nicht gefunden |
| **Beispielantwort** | {  "id": 1,  "name": "Fahrradfahren",  "factor": 4  } |

|  |  |
| --- | --- |
| **Zugriff** | DELETE /sports/{id}/ |
| **Beschreibung** | Lösche die Daten der Sportart mit ID {id}. |
| **Parameter** | id (long) |
| **Antwort** | - |
| **Statuscodes** | 20 – Sportart gelöscht  403 – Nicht eingeloggt  404 – Sportart nicht gefunden |
| **Beispielantwort** |  |

|  |  |
| --- | --- |
| **Zugriff** | DELETE /sports/ |
| **Beschreibung** | Löscht die Daten aller Sportarten. |
| **Parameter** | - |
| **Antwort** | - |
| **Statuscodes** | 200 –alle Sportarten gelöscht  403 – Nicht eingeloggt |
| **Beispielantwort** | - |

|  |  |
| --- | --- |
| **Zugriff** | GET /sports/ |
| **Beschreibung** | Erhalte die Daten aller Sportarten. |
| **Parameter** | - |
| **Antwort** | JSON-Liste mit folgenden Eigenschaften:   * id (Long): Eindeutige ID der Sportart * name (String): Name der Sportart * factor (Float): Faktor der Sportart |
| **Statuscodes** | 200 – alle Sportarten gefunden  403 – Nicht eingeloggt |
| **Beispielantwort** | [  {  "id": 1,  "name": "Fahrradfahren",  "factor": 4  },  {  "id": 3,  "name": "Testsport",  "factor": 4.7  }  ] |

|  |  |
| --- | --- |
| **Zugriff** | PUT /sports/{id}/ |
| **Beschreibung** | Bearbeiten der Sportartdaten der gegebenen Sportart. |
| **Parameter** | * id(Long) * name (String) * factor (Float) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (Long): Eindeutige ID der Sportart * name (String): Name der Sportart * factor (Float): Faktor der Sportart |
| **Statuscodes** | 200 – Sportart updated  403 – Nicht eingeloggt  404 – Sportart nicht gefunden |
| **Beispielantwort** | {  "id": 1,  "name": "Laufen",  "factor": 10  } |

## 7.4 Challenge

|  |  |
| --- | --- |
| **Zugriff** | GET /challenge/?type=all |
| **Beschreibung** | Erhalte die Daten aller Challenges. |
| **Parameter** | Type (String) |
| **Antwort** | JSON-Liste mit folgenden Eigenschaften:   * id (Long): Eindeutige ID der Challenge * name (String): Name der Sportart * description (String): Beschreibung der Challenge * startDate (LocalDate): Startdatum der Challenge * endDate (LocalDate): Enddatum der Challenge * targetDistance (Float): Zieldistanz der Challenge |
| **Statuscodes** | 200 – Suche erfolgreich  403 – Nicht eingeloggt |
| **Beispielantwort** | [  {  "id": 2,  "name": "Super Challenge",  "description": "Challenge",  "startDate": "04.05.2023,12:00",  "endDate": "07.07.2023,12:00",  "targetDistance": 0  },  {  "id": 3,  "name": "Mega Challenge",  "description": "Challenge2",  "startDate": "18.01.2021,12:00",  "endDate": "12.05.2022,12:00",  "targetDistance": 0  }  ] |

|  |  |
| --- | --- |
| **Zugriff** | GET /challenge/?type=current |
| **Beschreibung** | Erhalte die Daten aller laufenden Challenges. |
| **Parameter** | Type (String) |
| **Antwort** | JSON-Liste mit folgenden Eigenschaften:   * id (Long): Eindeutige ID der Challenge * name (String): Name der Sportart * description (String): Beschreibung der Challenge * startDate (LocalDate): Startdatum der Challenge * endDate (LocalDate): Enddatum der Challenge * targetDistance (Float): Zieldistanz der Challenge |
| **Statuscodes** | 200 – Suche erfolgreich  403 – Nicht eingeloggt |
| **Beispielantwort** | [  {  "id": 2,  "name": "Super Challenge",  "description": "Challenge",  "startDate": "04.05.2023,12:00",  "endDate": "07.07.2023,12:00",  "targetDistance": 0  }  ] |

|  |  |
| --- | --- |
| **Zugriff** | GET /challenge/?type=past |
| **Beschreibung** | Erhalte die Daten aller vergangenen Challenges. |
| **Parameter** | Type (String) |
| **Antwort** | JSON-Liste mit folgenden Eigenschaften:   * id (Long): Eindeutige ID der Challenge * name (String): Name der Sportart * description (String): Beschreibung der Challenge * startDate (LocalDate): Startdatum der Challenge * endDate (LocalDate): Enddatum der Challenge * targetDistance (Float): Zieldistanz der Challenge |
| **Statuscodes** | 200 – Suche erfolgreich  403 – Nicht eingeloggt |
| **Beispielantwort** | [  {  "id": 3,  "name": "Mega Challenge",  "description": "Challenge2",  "startDate": "18.01.2021,12:00",  "endDate": "12.05.2022,12:00",  "targetDistance": 0  }  ] |

|  |  |
| --- | --- |
| **Zugriff** | POST /challenge/ |
| **Beschreibung** | Fügt eine Challenge mit den gegebenen Daten hinzu. |
| **Parameter** | * SportID (long [ ]) * SportFactor (long [ ]) * file (Binär) * json (Challenge) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (Long): Eindeutige ID der Challenge * name (String): Name der Sportart * description (String): Beschreibung der Challenge * startDate (LocalDateTime): Startdatum der Challenge * endDate (LocalDateTime): Enddatum der Challenge * targetDistance (Float): Zieldistanz der Challenge |
| **Statuscodes** | 201 – Challenge erfolgreich erstellt  403 – Nicht eingeloggt  417 – Challenge nicht erstellt |
| **Beispielantwort** | {  "id": 6,  "name": "string",  "description": "string",  "startDate": "20.05.2023,12:00",  "endDate": "12.08.2024,12:00",  "imageID": 3,  "targetDistance": 0  } |

|  |  |
| --- | --- |
| Zugriff | DELETE /challenge/{id}/ |
| Beschreibung | Lösche die Daten der Challenge mit ID {id} |
| Parameter | Id (Long) |
| Antwort | - |
| Statuscodes | 200 – Challenge gelöscht  403 – Nicht eingeloggt  404 – Challenge nicht gefunden |
| Beispielantwort |  |

|  |  |
| --- | --- |
| Zugriff | GET /challenges/{id}/teams/ |
| Beschreibung | Erhalte alle Teams der Challenge |
| Parameter | id (Long): ID der Challenge |
| Antwort | JSON-Liste mit folgenden Eigenschaften:   * id (Long): Eindeutige ID des Teams * name (String): Name des Teams * imageID (Long): Eindeutige ID des Bildes des Teams * challengeID (Long): Eindeutige ID der Challenge |
| Statuscodes | 200 – Suche des Teams erfolgreich!  403 – Nicht eingeloggt  404 – Kein Team gefunden |
| Beispielantwort | [  {  "id": 3,  "name": "blue",  "imageID”: 1,  “challengeID”: 4  }  ] |

|  |  |
| --- | --- |
| Zugriff | DELETE /challenges/{id}/teams/ |
| Beschreibung | Lösche alle Teams der Challenge |
| Parameter | id (Long): ID der Challenge |
| Antwort | - |
| Statuscodes | 204 – Teams der challenge gelöscht  403 – Nicht eingeloggt  404 – Keine Challenge gefunden |
| Beispielantwort | - |

|  |  |
| --- | --- |
| Zugriff | GET /challenges/{id}/rawDistance/ |
| Beschreibung | Erhalte Distanz der Challenge ohne eingerechnete Boni |
| Parameter | id (Long): ID der Challenge |
| Antwort | Text mit der gerechneten Zahl als Float. |
| Statuscodes | 200 – Aktivität für Benutzer gefunden  403 – Nicht eingeloggt  500 – Nicht alle Aktivitäten sind Teil derselben Challenge |
| Beispielantwort | 4 |

|  |  |
| --- | --- |
| Zugriff | GET /challenges/{id}/members/ |
| Beschreibung | Erhalte alle Mitglieder der Challenge |
| Parameter | id (Long): ID der Challenge |
| Antwort | JSON-Liste mit folgenden Eigenschaften:   * email(String): Email des Mitglieds * firstName (String): Vorname des Mitglieds * lastName (String): Nachname des Mitglieds * userID (long): Eindeutige ID des Mitglieds * imageID (Long): Eindeutige ID des Profilbildes |
| Statuscodes | 200 – Mitglied gefunden!  403 – Nicht eingeloggt  404 – Mitglied nichtKefunden |
| Beispielantwort | [  {  "email”: “max.mustermann@exp.de”,  "firstName": "Max",  “lastName”: “Mustermann”,  “userID”: 1,  “imageID”: 4  }  ] |

|  |  |
| --- | --- |
| Zugriff | GET /challenges/{id}/distance/ |
| Beschreibung | Erhalte Distanz der Challenge mit eingerechneten Boni |
| Parameter | id (Long): ID der Challenge |
| Antwort | Text mit der gerechneten Zahl als Float. |
| Statuscodes | 200 – Aktivität für die Challenge gefunden  403 – Nicht eingeloggt  500 – Nicht alle Aktivitäten sind Teil derselben Challenge |
| Beispielantwort | 8 |

|  |  |
| --- | --- |
| Zugriff | GET /challenges/{id}/activities/ |
| Beschreibung | Erhalte alle Aktivitäten einer Challenge |
| Parameter | id (Long): ID der Challenge |
| Antwort | JSON-Liste mit folgenden Eigenschaften:   * id (Long): Eindeutige ID der Challenge * challengeSportID (Long): * memberID (Long): Eindeutige ID des Benutzers * distance (float): Distanz der Aktivität * date (LocalDateTime): Datum der Aktivität |
| Statuscodes | 200 – Aktivität für Challenge gefunden!  403 – Nicht eingeloggt  404 – Keine Aktivität gefunden |
| Beispielantwort | [  {  "id": 1,  "challengeSportID": 1,  "memberID”: 1,  “distance”: 4,  “date”: “04.05.2023,12:00”  }  ] |

|  |  |
| --- | --- |
| Zugriff | GET /challenges/members/{id}/ |
| Beschreibung | Erhalte alle Challenges für diesen Benutzer |
| Parameter | id (Long): ID des Members |
| Antwort | JSON-Liste mit folgenden Eigenschaften:   * email(String): Email des Benutzers * firstName(String): Vorname des Nutzers * lastName(String): Nachname des Nutzers * userID(Long): Eindeutige ID des Nutzers   imageID(Long): Eindeutige ID des Profilbildes des Nutzers |
| Statuscodes | 200 – Suche der Challenges für den Nutzer erfolgreich!  403 – Nicht eingeloggt  404 – Keine Challenges für diesen Nutzer gefunden |
| Beispielantwort | [  {  "email": “Hanna.Montana@exp.de”,  "firstName": "Hannah",  "lastName”: “Montana”,  “userID”: 2.  “imageID”: 0  }  ] |

## 7.5 SAML

|  |  |
| --- | --- |
| **Zugriff** | GET /saml/ |
| **Beschreibung** | Erhalte die von SAML übergebenen Nutzerdaten zurück. |
| **Parameter** | - |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * email (String): E-Mail des SAML-Benutzers * firstName (String): Vorname des SAML-Benutzers * lastName (String): Nachname des SAML-Benutzers |
| **Statuscodes** | 200 – User ist eingeloggt  403 – Nicht eingeloggt |
| **Beispielantwort** | {  "email": "max@mustermann.test",  "firstName": "Max",  "lastName": "Mustermann"  } |

## 7.6 Team

|  |  |
| --- | --- |
| **Zugriff** | GET /teams/{id}/ |
| **Beschreibung** | Erhalte die Daten vom Team mit der gegebenen ID |
| **Parameter** | Id (long) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (long): TeamID * Name (String): Name des Teams * ImageID (long): Id des Bildes * ChallengeID (long): Id der zugehörigen Challenge |
| **Statuscodes** | 200 – Team gefunden  403 – Nicht eingeloggt  404 – Team nicht gefunden |
| **Beispielantwort** | {  "id": 1,  "name": "Team Rot",  "imageID": 1,  "challengeID": 1  } |

|  |  |
| --- | --- |
| **Zugriff** | GET /teams/ |
| **Beschreibung** | Erhalte die Daten aller Teams |
| **Parameter** | - |
| **Antwort** | JSON-Liste mit folgenden Eigenschaften:   * id (long): TeamID * Name (String): Name des Teams * ImageID (long): Id des Bildes * ChallengeID (long): Id der zugehörigen Challenge |
| **Statuscodes** | 200 – alle Teams gefunden  403 – Nicht eingeloggt |
| **Beispielantwort** | [  {  "id": 1,  "name": "Team Rot",  "imageID": 1,  "challengeID": 1  },  {  "id": 2,  "name": "Team Blau",  "imageID": 2,  "challengeID": 1  } ] |

|  |  |
| --- | --- |
| **Zugriff** | PUT /teams/{id}/ |
| **Beschreibung** | Update ein Team mit der gegebenen ID |
| **Parameter** | * Id (long) * file (Binär) * json (Team) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (long): TeamID * Name (String): Name des Teams * ImageID (long): Id des Bildes * ChallengeID (long): Id der zugehörigen Challenge |
| **Statuscodes** | 200 – Team erfolgreich updated  403 – Nicht eingeloggt  417 – Team konnte nicht verändert werden |
| **Beispielantwort** | {  "id": 1,  "name": "Team Rot",  "imageID": 1,  "challengeID": 3 } |

|  |  |
| --- | --- |
| **Zugriff** | DELETE /teams/{id}/ |
| **Beschreibung** | Lösche die Daten vom Team mit der gegebenen ID |
| **Parameter** | Id (long) |
| **Antwort** | - |
| **Statuscodes** | 200 – Team gelöscht  403 – Nicht eingeloggt  404 – Team nicht gefunden |
| **Beispielantwort** | - |

|  |  |
| --- | --- |
| **Zugriff** | DELETE /teams/ |
| **Beschreibung** | Lösche die Daten aller Teams |
| **Parameter** | - |
| **Antwort** | - |
| **Statuscodes** | 200 – Alle Teams gelöscht  403 – Nicht eingeloggt |
| **Beispielantwort** | - |

|  |  |
| --- | --- |
| **Zugriff** | POST /teams/ |
| **Beschreibung** | Füge ein Team mit den gegebenen Daten zu |
| **Parameter** | * file (Binär) * json (Team) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (long): TeamID * Name (String): Name des Teams * ImageID (long): ID des Bildes * ChallengeID (long): ID der zugehörigen Challenge |
| **Statuscodes** | 200 – Team erfolgreich hinzugefügt  403 – Nicht eingeloggt  404 – Team nicht gefunden |
| **Beispielantwort** | {  "id": 3,  "name": "Team Superman",  "imageID": 3,  "challengeID": 1 } |

## 7.7 Team-Member

|  |  |
| --- | --- |
| **Zugriff** | GET /teamMembers/{id}/ |
| **Beschreibung** | Erhalte die Daten vom Team-Members mit der gegebenen ID |
| **Parameter** | Id (long) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (long): Team-Member ID * teamID (long): Id des Teams * memberID (long): Id des Teams |
| **Statuscodes** | 200 – Team-Member gefunden  403 – Nicht eingeloggt  404 – Team-Member nicht gefunden |
| **Beispielantwort** | {  "id": 1,  " teamID ": 1,  " memberID ": 1  } |

|  |  |
| --- | --- |
| **Zugriff** | GET /teamMembers/ |
| **Beschreibung** | Erhalte die Daten von allen Team-Members |
| **Parameter** | - |
| **Antwort** | JSON-Liste mit folgenden Eigenschaften:   * id (long): Team-Member ID * teamID (long): Id des Teams * memberID (long): Id des Teams |
| **Statuscodes** | 200 – Suche erfolgreich  403 – Nicht eingeloggt |
| **Beispielantwort** | [  {  "id": 1,  " teamID ": 1,  " memberID ": 1  },  {  "id": 2,  " teamID ": 1,  " memberID ": 2  } ] |

|  |  |
| --- | --- |
| **Zugriff** | PUT /teamMembers/{id}/ |
| **Beschreibung** | Update eines Team-Members mit der gegebenen ID |
| **Parameter** | * id (Long): ID des zu bearbeitenden Teammitglieds   Body: JSON-Objekt mit folgenden Eigenschaften:   * teamID (Long) * memberID (Long) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (long): Team-Member ID * teamID (long): Id des Teams * memberID (long): Id des Teams |
| **Statuscodes** | 200 – Team-Member updated  403 – Nicht eingeloggt  404 – Team-Member nicht gefunden |
| **Beispielantwort** | {  "id": 1,  " teamID ": 2,  " memberID ": 1  } |

|  |  |
| --- | --- |
| **Zugriff** | DELETE /teamMembers/{id}/ |
| **Beschreibung** | Lösche die Daten vom Team-Members mit der gegebenen ID |
| **Parameter** | Id (long) |
| **Antwort** | - |
| **Statuscodes** | 200 – Team-Member gelöscht  403 – Nicht eingeloggt |
| **Beispielantwort** | - |

|  |  |
| --- | --- |
| **Zugriff** | DELETE /teamMembers/ |
| **Beschreibung** | Lösche die Daten von allen Team-Members |
| **Parameter** | - |
| **Antwort** | - |
| **Statuscodes** | 200 – alle Team-Member gelöscht  403 – Nicht eingeloggt |
| **Beispielantwort** | - |

|  |  |
| --- | --- |
| **Zugriff** | POST /teamMembers/ |
| **Beschreibung** | Erhalte die Daten vom Team-Members mit der gegebenen ID |
| **Parameter** | Body: JSON-Objekt mit folgenden Eigenschaften:   * teamID (Long) * memberID (Long) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (Long): Team-Member ID * teamID (Long): Id des Teams * memberID (Long): Id des Teams |
| **Statuscodes** | 200 – Team-Member erfolgreich hinzugefügt  403 – Nicht eingeloggt  404 – Team-Member nicht gefunden |
| **Beispielantwort** | {  "id": 1,  "teamID": 1,  "memberID": 1  } |

|  |  |
| --- | --- |
| **Zugriff** | GET /teamMembers/challenges/{id}/ |
| **Beschreibung** | Erhalte die Daten vom Team-Members von einer Challenge mit der gegebenen Challenge ID |
| **Parameter** | Id (long) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (long): Team-Member ID * teamID (long): Id des Teams * memberID (long): Id des Teams |
| **Statuscodes** | 200 – Suche erfolgreich  403 – Nicht eingeloggt |
| **Beispielantwort** | {  "id": 1,  " teamID ": 1,  " memberID ": 1  } |

## 7.8 Challenge-Sport

|  |  |
| --- | --- |
| **Zugriff** | GET /challenge-sports/{id}/ |
| **Beschreibung** | Erhalte die Daten vom Challenge-Sport mit der gegebenen ID |
| **Parameter** | Id (long) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (long): Challenge-Sport ID * Factor (float): Faktor der Sportart für die Challenge * ChallengeID (long): Id der zugehörigen Challenge * sportID (long): Id der Sportart |
| **Statuscodes** | 200 – Challenge-Sport gefunden  403 – Nicht eingeloggt  404 – Challenge-Sport nicht gefunden |
| **Beispielantwort** | {  "id": 1,  "factor": 1.4,  "challengeID ": 1,  "sportID": 1  } |

|  |  |
| --- | --- |
| **Zugriff** | GET /challenge-sports/ |
| **Beschreibung** | Erhalte alle Daten von Challenge-Sport |
| **Parameter** | Id (long) |
| **Antwort** | JSON-Liste mit folgenden Eigenschaften:   * id (long): Challenge-Sport ID * Factor (float): Faktor der Sportart für die Challenge * ChallengeID (long): Id der zugehörigen Challenge * sportID (long): Id der Sportart |
| **Statuscodes** | 200 – alle Challenge-Sports gefunden  403 – Nicht eingeloggt |
| **Beispielantwort** | [  {  "id": 1,  "factor": 1.4,  "challengeID ": 1,  "sportID": 1  },  {  "id": 2,  "factor": 1.4,  "challengeID ": 1,  "sportID": 2  }  ] |

|  |  |
| --- | --- |
| **Zugriff** | PUT /challenge-sports/{id}/ |
| **Beschreibung** | Update die Daten von Challenge-Sport mit der gegebenen ID |
| **Parameter** | Id (long) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (long): Challenge-Sport ID * Factor (float): Faktor der Sportart für die Challenge * ChallengeID (long): Id der zugehörigen Challenge * sportID (long): Id der Sportart |
| **Statuscodes** | 200 – Challenge-Sport updated  403 – Nicht eingeloggt  404 – Challenge-Sport nicht gefunden |
| **Beispielantwort** | {  "id": 1,  "factor": 1.4,  "challengeID ": 1,  "sportID": 1 } |

|  |  |
| --- | --- |
| **Zugriff** | DELETE /challenge-sports/{id}/ |
| **Beschreibung** | Lösche die Daten vom Team mit der gegebenen ID |
| **Parameter** | - |
| **Antwort** | - |
| **Statuscodes** | 200 – Challenge-Sport gelöscht  403 – Nicht eingeloggt  404 – Challenge-Sport nicht gefunden |
| **Beispielantwort** | - |

|  |  |
| --- | --- |
| **Zugriff** | DELETE /challenge-sports/ |
| **Beschreibung** | Lösche alle Daten von Challenge-Sport |
| **Parameter** | - |
| **Antwort** | - |
| **Statuscodes** | 200 – alle Challenge-Sports gelöscht  403 – Nicht eingeloggt |
| **Beispielantwort** | - |

|  |  |
| --- | --- |
| **Zugriff** | POST /challenge-sports/ |
| **Beschreibung** | Fügt eine Challenge-Sport hinzu. |
| **Parameter** | Body: JSON-Objekt mit folgenden Eigenschaften:   * id (Long): Challenge-Sport ID * factor (Float): Faktor der Sportart für die Challenge * challengeID (Long): ID der zugehörigen Challenge * sportID (Long): ID der zugehörigen Sportart |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (Long): Challenge-Sport ID * factor (Float): Faktor der Sportart für die Challenge * challengeID (Long): ID der zugehörigen Challenge * sportID (Long): ID der zugehörigen Sportart |
| **Statuscodes** | 201 – Challenge-Sport hinzugefügt  403 – Nicht eingeloggt  500 – Challenge-Sport konnte nicht erstellt werden |
| **Beispielantwort** | {  "id": 4,  "factor": 1.4,  "challengeID ": 1,  "sportID": 5  } |

|  |  |
| --- | --- |
| **Zugriff** | GET /challenge-sports/challenges/{id}/ |
| **Beschreibung** | Erhalte alle Challenge Sports für eine gegebene Challenge |
| **Parameter** | * Id (Long): ID der Challenge |
| **Antwort** | JSON-Liste mit folgenden Eigenschaften:   * id (Long): Eindeutige ID der Challenge-Sport * factor (Float): Faktor der Sportart für die Challenge * challengeID (Long): ID der zugehörigen Challenge * sportID (Long): ID der zugehörigen Sportart |
| **Statuscodes** | 200 – Challenge-Sport gefunden  403 – Nicht eingeloggt |
| **Beispielantwort** | [  {  "id": 1,  "factor": 3,  "challengeID": 1,  "sportID": 1  }  ] |

## 7.9 Bonus

|  |  |
| --- | --- |
| **Zugriff** | GET /bonuses/{id}/ |
| **Beschreibung** | Erhalte die Daten vom Bonus mit der gegebenen ID |
| **Parameter** | Id (long) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (long): Bonus ID * challengeSportID (long): Id von Challenge-Sport * startDate (LocalDate): Startdatum des Bonus * endDate (LocalDate): Enddatum des Bonus * factor (float): Faktor der Sportart für die Challenge * name (string): Name des Bonus * description (string): Beschreibung des Bonus |
| **Statuscodes** | 200 – Bonus gefunden  403 – Nicht eingeloggt  404 – Bonus nicht gefunden |
| **Beispielantwort** | {  "id": 1,  “challengeSportID”: 1,  "startDate": "04.05.2023,12:00",  "endDate": "07.07.2023,12:00",  "factor": 2,  "name": “Double Laufen”,  "description": “Man bekommt doppelt so viel ” } |

|  |  |
| --- | --- |
| **Zugriff** | GET /bonuses/ |
| **Beschreibung** | Erhalte alle Daten von Bonus |
| **Parameter** | - |
| **Antwort** | JSON-Liste mit folgenden Eigenschaften:   * id (long): Bonus ID * challengeSportID (long): Id von Challenge-Sport * startDate (LocalDate): Startdatum des Bonus * endDate (LocalDate): Enddatum des Bonus * factor (float): Faktor der Sportart für die Challenge * name (string): Name des Bonus * description (string): Beschreibung des Bonus |
| **Statuscodes** | 200 – alle Bonuses gefunden  403 – Nicht eingeloggt  404 – Bonus nicht gefunden |
| **Beispielantwort** | [  {  "id": 1,  “challengeSportID”: 1,  "startDate": "04.05.2023,12:00",  "endDate": "07.07.2023,12:00",  "factor": 1.4,  "name": “Double Laufen”,  "description": “Man bekommt doppelt so viel ”  },  {  "id": 2,  “challengeSportID”: 1,  "startDate": "04.05.2023,12:00",  "endDate": "07.07.2023,12:00",  "factor": 3,  "name": “Triple Fahren”,  "description": “Man bekommt dreifach so viel ”  } ] |

|  |  |
| --- | --- |
| **Zugriff** | PUT /bonuses/{id}/ |
| **Beschreibung** | Update die Daten von Bonus mit der gegebenen ID |
| **Parameter** | Id (long) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (long): Bonus ID * challengeSportID (long): Id von Challenge-Sport * startDate (LocalDate): Startdatum des Bonus * endDate (LocalDate): Enddatum des Bonus * factor (float): Faktor der Sportart für die Challenge * name (string): Name des Bonus * description (string): Beschreibung des Bonus |
| **Statuscodes** | 200 – Bonus updated  403 – Nicht eingeloggt  404 – Bonus nicht gefunden |
| **Beispielantwort** | {  "id": 1,  “challengeSportID”: 1,  "startDate": "04.05.2023,12:00",  "endDate": "07.07.2023,12:00",  "factor": 3,  "name": “Triple Fahren”,  "description": “Man bekommt dreifach so viel ”  } |

|  |  |
| --- | --- |
| **Zugriff** | DELETE /bonuses/{id}/ |
| **Beschreibung** | Lösche die Daten von Bonus mit der gegebenen ID |
| **Parameter** | Id (long) |
| **Antwort** | - |
| **Statuscodes** | 200 – Bonus gelöscht  403 – Nicht eingeloggt  404 – Bonus nicht gefunden |
| **Beispielantwort** | - |

|  |  |
| --- | --- |
| **Zugriff** | DELETE /bonuses/ |
| **Beschreibung** | Lösche alle Daten von Bonus |
| **Parameter** | - |
| **Antwort** |  |
| **Statuscodes** | 200 – alle Bonuses gelöscht  403 – Nicht eingeloggt |
| **Beispielantwort** | - |

|  |  |
| --- | --- |
| **Zugriff** | POST /bonuses/ |
| **Beschreibung** | Füge ein Bonus hinzu |
| **Parameter** | * challengeSportID (long): Id von Challenge-Sport * startDate (LocalDate): Startdatum des Bonus * endDate (LocalDate): Enddatum des Bonus * factor (float): Faktor der Sportart für die Challenge * name (string): Name des Bonus * description (string): Beschreibung des Bonus |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (long): Bonus ID * challengeSportID (long): Id von Challenge-Sport * startDate (LocalDate): Startdatum des Bonus * endDate (LocalDate): Enddatum des Bonus * factor (float): Faktor der Sportart für die Challenge * name (string): Name des Bonus * description (string): Beschreibung des Bonus |
| **Statuscodes** | 200 – Bonus erfolgreich hinzugefügt  403 – Nicht eingeloggt  404 – Bonus nicht gefunden |
| **Beispielantwort** | {  "id": 1,  “challengeSportID”: 1,  "startDate": "04.05.2023,12:00",  "endDate": "07.07.2023,12:00",  "factor": 3,  "name": “Triple Fahren”,  "description": “Man bekommt dreifach so viel ”  } |

## 7.10 Aktivitäten

|  |  |
| --- | --- |
| **Zugriff** | GET /activities/{id}/ |
| **Beschreibung** | Erhalte die Daten von Aktivitäten mit der gegebenen ID |
| **Parameter** | Id (long) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (long): activity ID * challengeSportID (long): Id des zugehörigen Challenge-Sports * memberID (long): Id des Users für die Aktivität * distance (float): eingegebene Distanz * date (string): Datum wann Aktivität erstellt wurde |
| **Statuscodes** | 200 – Aktivität gefunden  403 – Nicht eingeloggt  404 – Aktivität nicht gefunden |
| **Beispielantwort** | {  "id": 1,  "challengeSportID ": 1,  "memberID": 1,  “distance”: 10,  “date”: "04.05.2023,12:00"  } |

|  |  |
| --- | --- |
| **Zugriff** | GET /activities/ |
| **Beschreibung** | Erhalte alle Daten von Aktivitäten |
| **Parameter** | - |
| **Antwort** | JSON-Liste mit folgenden Eigenschaften:   * id (long): activity ID * challengeSportID (long): Id des zugehörigen Challenge-Sports * memberID (long): Id des Users für die Aktivität * distance (float): eingegebene Distanz * date (string): Datum wann Aktivität erstellt wurde |
| **Statuscodes** | 200 – alle Aktivitäten gefunden  403 – Nicht eingeloggt |
| **Beispielantwort** | [  {  "id": 1,  "challengeSportID ": 1,  "memberID": 1,  “distance”: 10,  “date”: "04.05.2023,12:00"  },  {  "id": 2,  "challengeSportID ": 1,  "memberID": 3,  “distance”: 100,  “date”: "04.05.2023,12:00"  } ] |

|  |  |
| --- | --- |
| **Zugriff** | POST /activities/ |
| **Beschreibung** | Eine Aktivität hinzufügen |
| **Parameter** | * challengeSportID (long): * memberID (long) * distance (float) * date (string) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (long): activity ID * challengeSportID (long): Id des zugehörigen Challenge-Sports * memberID (long): Id des Users für die Aktivität * distance (float): eingegebene Distanz * date (string): Datum wann Aktivität erstellt wurde |
| **Statuscodes** | 200 – Aktivität erfolgreich hinzugefügt  403 – Nicht eingeloggt  404 – Aktivität nicht gefunden |
| **Beispielantwort** | {  "id": 1,  "challengeSportID ": 1,  "memberID": 3,  “distance”: 600,  “date”: "04.05.2023,12:00"  } |

|  |  |
| --- | --- |
| **Zugriff** | PUT /activities/{id}/ |
| **Beschreibung** | Update die Daten von Aktivitäten mit der gegebenen ID |
| **Parameter** | Id (long) |
| **Antwort** | JSON-Objekt mit folgenden Eigenschaften:   * id (long): activity ID * challengeSportID (long): Id des zugehörigen Challenge-Sports * memberID (long): Id des Users für die Aktivität * distance (float): eingegebene Distanz * date (string): Datum wann Aktivität erstellt wurde |
| **Statuscodes** | 200 – Aktivität updated  403 – Nicht eingeloggt  404 – Aktivität nicht gefunden |
| **Beispielantwort** | {  "id": 1,  "challengeSportID ": 1,  "memberID": 1,  “distance”: 4000,  “date”: "04.05.2023,12:00" } |

|  |  |
| --- | --- |
| **Zugriff** | DELETE /activities/{id}/ |
| **Beschreibung** | Lösche die Daten von Aktivitäten mit der gegebenen ID |
| **Parameter** | Id (long) |
| **Antwort** | - |
| **Statuscodes** | 200 – Aktivität gelöscht  403 – Nicht eingeloggt  404 – Aktivität nicht gefunden |
| **Beispielantwort** | - |

|  |  |
| --- | --- |
| **Zugriff** | DELETE /activities/ |
| **Beschreibung** | Lösche alle Daten von Aktivitäten |
| **Parameter** | - |
| **Antwort** | - |
| **Statuscodes** | 200 – alle Aktivitäten gelöscht  403 – Nicht eingeloggt |
| **Beispielantwort** | - |

# 8 Handbücher

123

## 8.1 Installation

123

## 8.2 Administration

123

# 9 Fazit

Fazit usw. Aufteilung der Teams

## 9.1 Reflektion Projektmanagement

123

## 9.2 Reflektion Lernfortschritt

123

## 9.3 Ausblick

123

# 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] | Keycloak, „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]. |
| [9] | Meta, „React Testing Environments,“ [Online]. Available: https://legacy.reactjs.org/docs/testing-environments.html. [Zugriff am 10 April 2023]. |

1. [9] Öffentlich zugängliche Beschreibung der ISO-Norm 25010 [↑](#footnote-ref-2)
2. [8] Conventional Commits v1.0.0 Spezifikation [↑](#footnote-ref-3)