

# Softwareprojekt Wintersemester 2019/2020

am Fachgebiet Software Engineering, Leibniz Universität Hannover

## Anforderungsspezifikation *GameDev1*

### Vorgelegt

am 12.11.2019  
von Team GameDev1

### Ausführende:

<i>Nachname</i>	<i>Vorname</i>	<i>Rolle</i>
<i>Andrae</i>	<i>Dominik</i>	<i>Projektleiter</i>
<i>Allermann</i>	<i>Lukas</i>	<i>Qualitätsbeauftragter</i>
<i>Curth</i>	<i>Leon</i>	<i>Entwickler</i>
<i>Gaire</i>	<i>Amrit</i>	<i>Entwickler</i>
<i>Hollmann</i>	<i>Tim</i>	<i>Entwickler</i>
<i>Niehus</i>	<i>Lukas</i>	<i>Entwickler</i>
<i>Volodarskis</i>	<i>Felix</i>	<i>Entwickler</i>

### Das Dokument enthält

- ☒ Die Anforderungen aus Kundensicht (User Requirements)
- ☒ Anforderungen, wie das zu System zu gestalten ist (System Requirements)

---

Datum, Unterschrift des Projektleiters, auch für die anderen Projektangehörigen

### Kunden-Bewertung

Der Kunde, Nico Suhl, im Auftrag der Adesso AG, bestätigt mit seiner Unterschrift, diese Anforderungsspezifikation erhalten, geprüft und für inhaltlich ☐ **in Ordnung** | ☐ **weitgehend in Ordnung** | ☐ **deutlich zu verbessern** | ☐ **nicht akzeptabel** befunden zu haben.

---

Datum, Unterschrift des Kunden; evtl. Vermerk.

**Inhaltsverzeichnis**

<b>1 Mission des Projekts</b>	<b>3</b>
1.1 Erläuterung des zu lösenden Problems	4
1.2 Wünsche und Prioritäten des Kunden	4
1.3 Domänenbeschreibung	4
1.4 Maßnahmen zur Anforderungsanalyse	4
<b>2 Rahmenbedingungen und Umfeld</b>	<b>5</b>
2.1 Einschränkungen und Vorgaben	5
2.2 Anwender	5
2.3 Schnittstellen und angrenzende Systeme	5
<b>3 Funktionale Anforderungen</b>	<b>6</b>
3.1 Use Case-Diagramm	6
3.2 Use Case-Beschreibungen	7
3.3 Hauptfunktionen des Programms	12
<b>4 Qualitätsanforderungen</b>	<b>13</b>
4.1 Qualitätsziele des Projekts	13
4.2 Qualitäts-Prioritäten des Kunden	13
4.3 Wie Qualitätsziele erreicht werden sollen	14
<b>5 Hinweise zur Umsetzung</b>	<b>14</b>
Weboberfläche	15
Architektur	17
Was sind Achievements?	17
Wie funktioniert die CONFIG-Datei?	18
<b>6 Probleme und Risiken</b>	<b>18</b>
<b>7 Optionen zur Aufwandsreduktion</b>	<b>19</b>
7.1 Mögliche Abstriche	19
7.2 Inkrementelle Arbeit	19
<b>8 Glossar</b>	<b>20</b>
<b>9 Abnahme-Testfälle</b>	<b>21</b>

# 1 Mission des Projekts

## 1.1 Erläuterung des zu lösenden Problems

Das Ziel des Projekts GameDev ist es, ein Programm zu entwickeln, das durch Vergabe von Achievements und Punkten Arbeitsschritte in einem Softwareentwicklungsprozess belohnt, indem verschiedene Metriken von bestehenden Systemen im Softwareentwicklungsprozess (wie Jira, BitBucket oder anderen Git-Hosts) gesammelt und in Spiel-Artefakte überführt werden. Dies soll Mitarbeiter/-innen mehr Motivation für die Arbeit geben, da ihre Bemühungen stets durch Achievements und Punkte Anerkennung bekommen und sie sich im Highscore mit anderen Mitarbeitern messen können.

## 1.2 Wünsche und Prioritäten des Kunden

Die Anwendung soll modular aufgebaut werden. Es existiert eine Hauptanwendung, die eine Weboberfläche zur Interaktion mit dem Anwender bereitstellt, beispielsweise errungene Preise, Level oder Ranglisten anzeigt.

Die einzelnen Anwendungen wie Jira, Bitbucket oder andere sollen modular nachträglich angebunden werden können (über spezielle Plug-Ins, die wir im Folgenden “Kollektoren” nennen), und zwar ohne die Hauptanwendung neu starten oder kompilieren zu müssen. Kollektoren für Bitbucket und Jira müssen auf jeden Fall entwickelt werden. Die Kollektoren sollen in der Lage sein, selbstständig (aktiv, regelmäßig) relevante Metriken aus ihrem jeweiligen System zu extrahieren (“Polling”) oder (passiv) Metriken über sogenannte Web-Hooks von diesen empfangen zu können. Passive Web-Hooks sollen aktiven Polling gegenüber bevorzugt werden, um die Auslastung der Systeme möglichst gering zu halten.

Als vorhandenes Authentifizierungssystem soll LDAP genutzt werden, sodass Benutzer in der Lage sind, sich in der Hauptanwendung ohne Vergabe von anwendungsspezifischen Anmeldedaten anzumelden.

## 1.3 Domänenbeschreibung

Das System läuft auf einem lokalen Server, der sich im selben Netzwerk befindet, wie alle Benutzer/-innen. Das System sollte bis zu 40 Nutzer/-innen gleichzeitig verwalten können, die sowohl Git und Jira, als auch ähnliche Systeme verwenden. Besonders zu beachten ist, dass das System leicht erlernbar ist und eine übersichtliche Anleitung zur Verfügung stellt.

## 1.4 Maßnahmen zur Anforderungsanalyse

Ein Webseiten-Entwurf wurde als Mockup implementiert und vorgestellt. Das Mockup enthält alle wichtigen Funktionen zur Benutzung des Programms und zeigt exemplarisch, wie die Benutzeroberfläche aussehen könnte. Das Mockup war für den Kunden sehr zufriedenstellend und kann in dieser Art ohne große Veränderungen implementiert werden. Zudem wurde eine Skizze zur Systemarchitektur vorgestellt (siehe Abschnitt “Architektur” in Kapitel 5), die umgesetzt wird.

## **2 Rahmenbedingungen und Umfeld**

### **2.1 Einschränkungen und Vorgaben**

Die SW läuft auf einem lokalen Server und soll Entwicklerteams im Größenbereich von ca. 8 bis 40 Entwickler/-innen verwalten können. Es wurden keine genaueren Ressourcen bzw. Kapazitäten angegeben, aber es ist davon auszugehen, dass die zur Verfügung stehenden Ressourcen definitiv ausreichend für das Projekt sind. Da das System gegebenenfalls weiterentwickelt werden soll, muss darauf geachtet werden, dass man weitere Kollektoren und Achievements möglichst einfach hinzufügen kann. Um dies zu erreichen ist eine möglichst modulare Architektur wünschenswert. Für die Authentifizierung sollen die schon bestehenden Login-Daten der Entwickler genutzt werden. Zur (initialen) Konfiguration des Systems wird eine Konfigurationsdatei verwendet.

### **2.2 Anwender**

Die Anwender/-innen sind SW-Entwickler/-innen. Dementsprechend kann man gute Kompetenz in der Bedienung von Computern und Webanwendungen voraussetzen. Jedoch sollte trotzdem darauf geachtet werden, dass die Weboberfläche einfach und intuitiv bedienbar ist.

Die Initialisierung des Systems wird von einem Admin auf der Kommandozeile ausgeführt. Ein/-e Administrator/-in sollte über genügend Erfahrung mit der Kommandozeile verfügen, sodass eine Dokumentation zur Inbetriebnahme des Systems genügt, um das System zu initialisieren.

### **2.3 Schnittstellen und angrenzende Systeme**

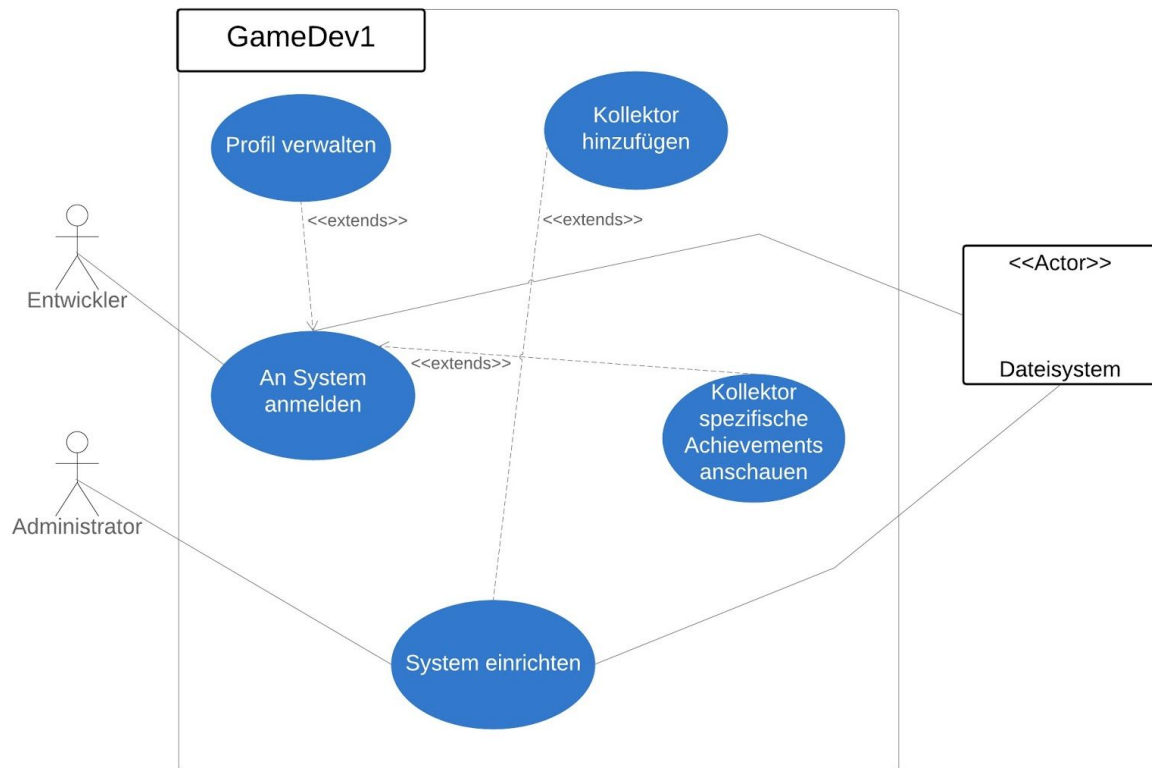
Jegliche Authentifizierungsdaten für einzelne Benutzer/-innen sind über LDAP erreichbar. Desweiteren sind die Entwickler/-innen in LDAP auch schon in die entsprechenden Gruppen von Entwickler/-innen.

Die SW soll Daten von Entwicklungstools wie GIT oder Jira auslesen können und um weitere Schnittstellen zu anderen Entwicklungstools erweiterbar sein.

Die SW benötigt einen Server mit Docker-Engine, auf dem sie laufen kann.

## 3 Funktionale Anforderungen

### 3.1 Use Case-Diagramm



### 3.2 Use Case-Beschreibungen

<b>Use Case 1</b>	<b>System einrichten</b>
<b>Erläuterung</b>	Der Administrator richtet das System auf dem Server ein.
<b>Umfeld</b>	Kommandozeile
<b>Systemgrenzen (Scope)</b>	Der Server, auf dem die Software gestartet wird.
<b>Ebene</b>	Hauptaufgabe
<b>Vorbedingung</b>	Der Server läuft und unterstützt Docker-Container.
<b>Mindestgarantie</b>	Die Software teilt mit, ob das Programm erfolgreich eingerichtet wurde oder nicht.
<b>Erfolgsgarantie</b>	Die Software ist richtig eingerichtet und betriebsbereit.
<b>Stakeholder</b>	Administrator : möchte die Software für anderen Entwickler bereitstellen. Developer/User : möchte Code Achievements und Ranglisten als Motivation ansehen können. Teamleader/Chef : möchte die Entwickler motivieren, um Produktivität zu steigern.
<b>Hauptakteur</b>	Administrator
<b>Auslöser</b>	Administrator startet die Applikation zum ersten Mal.
<b>Hauptszenario</b>	<ol style="list-style-type: none"> <li>Administrator startet die Applikation zum ersten Mal.</li> <li>Das System führt das interne Setup durch und fragt den User, ob er Kollektoren hinzufügen möchte.</li> <li>Administrator tippt "y" ein, und fügt die Kollektoren hinzu.(Use case 2)</li> <li>Das System meldet, ob die Kollektoren erfolgreich hinzugefügt wurden.</li> </ol>
<b>Erweiterungen</b>	2a: WENN ein Fehler aufgetreten ist, DANN soll eine Fehlermeldung ausgegeben werden und weiter bei 2. 3a: WENN der Admin "n" eintippt, DANN meldet das System, dass kein Kollektor hinzugefügt wurde. 4a: WENN ein Fehler auftritt, DANN gibt das System eine Fehlermeldung aus und kehrt zurück zu 2.
<b>Priorität</b>	unverzichtbar
<b>Verwendungshäufigkeit</b>	einmalig am Anfang pro Team

<b>Use Case 2</b>	<b>Kollektor hinzufügen</b>
<b>Erläuterung</b>	Der Administrator möchte einen Kollektor hinzufügen.
<b>Umfeld</b>	Kommandozeile
<b>Systemgrenzen (Scope)</b>	Der Server, auf dem das Hauptsystem läuft.
<b>Ebene</b>	Nebenfunktion
<b>Vorbedingung</b>	Der Administrator hat das System erfolgreich eingerichtet.
<b>Mindestgarantie</b>	Die Software teilt mit, ob ein Kollektor erfolgreich hinzugefügt werden konnte oder nicht.
<b>Erfolgsgarantie</b>	Die Software unterstützt einen zusätzlichen Kollektor.
<b>Stakeholder</b>	Administrator : möchte die Software für anderen developer bereitstellen. Developer/User : möchte Code Achievements und Ranglisten als Motivation ansehen können. Teamleader/Chef : möchte die Entwickler motivieren, um Produktivität zu steigern.
<b>Hauptakteur</b>	Administrator
<b>Auslöser</b>	Administrator startet die Kollektor-Applikation zum ersten Mal.
<b>Hauptszenario</b>	<ol style="list-style-type: none"> <li>Administrator startet die Kollektor-Applikation zum ersten Mal.</li> <li>Das System führt den internen Setup durch und informiert den User, ob der Dienste wie erwartet laufen.</li> </ol>
<b>Erweiterungen</b>	2a: WENN ein Fehler aufgetreten ist, DANN soll eine Fehlermeldung ausgegeben werden und weiter bei 1.
<b>Priorität</b>	hoch
<b>Verwendungshäufigkeit</b>	Einmal pro neues Subsystem

<b>Use Case 3</b>	<b>An System anmelden</b>
<b>Erläuterung</b>	Der Benutzer möchte sich im System anmelden.
<b>Umfeld</b>	Weboberfläche.
<b>Systemgrenzen (Scope)</b>	Browser.
<b>Ebene</b>	Hauptfunktion
<b>Vorbedingung</b>	Der Server läuft und unterstützt Docker-Container und die Hauptsoftware läuft.
<b>Mindestgarantie</b>	Die Daten des Nutzers werden korrekt angezeigt oder es wird ein Fehler angezeigt
<b>Erfolgsgarantie</b>	Die Daten des Nutzers werden korrekt angezeigt
<b>Stakeholder</b>	Developer/User : möchte Code Achievements und Ranglisten als Motivation ansehen können. Teamleader/Chef : möchte die Entwickler motivieren, um Produktivität zu steigern.
<b>Hauptakteur</b>	Entwickler
<b>Auslöser</b>	Der Benutzer betritt die Webseite
<b>Hauptszenario</b>	<ol style="list-style-type: none"> <li>1. Der Benutzer betritt die Webseite.</li> <li>2. Das System öffnet die Login-Seite.</li> <li>3. Der Benutzer gibt seine Login-Daten ein und klickt auf "Login".</li> <li>4. Das System öffnet die Hauptseite.</li> </ol>
<b>Erweiterungen</b>	<ol style="list-style-type: none"> <li>2a. WENN der Benutzer bereits eingeloggt ist, DANN öffnet sich direkt die Hauptseite.</li> <li>4a. WENN die Login-Daten nicht validiert werden konnten, DANN kehrt das System zurück zu 2 und gibt eine Fehlermeldung aus.</li> </ol>
<b>Priorität</b>	unverzichtbar
<b>Verwendungshäufigkeit</b>	Regelmäßig



<b>Use Case 4</b>	<b>Kollektor spezifische Achievements anschauen</b>
<b>Erläuterung</b>	Der User möchte die kollektorspezifischen Achievements einsehen.
<b>Umfeld</b>	Weboberfläche
<b>Systemgrenzen (Scope)</b>	Browser
<b>Ebene</b>	Nebenfunktion
<b>Vorbedingung</b>	Der Server läuft und unterstützt Docker-Container. Der User ist im System eingeloggt. Mindestens ein Kollektor existiert im System.
<b>Mindestgarantie</b>	Die Webseite wird geladen.
<b>Erfolgsgarantie</b>	Der User kann seine Kollektor spezifische Achievements einsehen.
<b>Stakeholder</b>	Developer/User : möchte Code Achievements und Ranglisten als Motivation ansehen können.
<b>Hauptakteur</b>	Developer/User
<b>Auslöser</b>	User klickt auf "Haus".
<b>Hauptszenario</b>	<ol style="list-style-type: none"> <li>1. User klickt auf "Haus".</li> <li>2. Das System öffnet die Achievement-Seite.</li> <li>3. User klickt auf gewünschten Kollektor.</li> <li>4. Das System öffnet die jeweilige Seite mit den Achievements des gewählten Kollektors.</li> </ol>
<b>Erweiterungen</b>	
<b>Priorität</b>	unverzichtbar
<b>Verwendungshäufigkeit</b>	häufig

<b>Use Case 5</b>	<b>Namen ändern</b>
<b>Erläuterung</b>	Der User möchte sein Profil verwalten.
<b>Umfeld</b>	Weboberfläche
<b>Systemgrenzen (Scope)</b>	Browser
<b>Ebene</b>	Nebenfunktion
<b>Vorbedingung</b>	Der Server läuft und unterstützt Docker-Container. Der User ist im System eingeloggt.
<b>Mindestgarantie</b>	Die Webseite wird geladen.
<b>Erfolgsgarantie</b>	Der User kann seine Daten erfolgreich ändern.
<b>Stakeholder</b>	Developer/User : möchte seine Daten verändern können
<b>Hauptakteur</b>	Developer/User
<b>Auslöser</b>	Der User klickt auf seinen Namen.
<b>Hauptszenario</b>	<ol style="list-style-type: none"> <li>1. Der User klickt auf seinen Namen.</li> <li>2. Das System öffnet die Profilseite des Users.</li> <li>3. Der User ändert seinen Namen im Textfeld.</li> <li>4. Das System übernimmt die gewählten Einstellungen.</li> </ol>
<b>Erweiterungen</b>	3a. WENN Der User einen Haken bei "Anonym bleiben" setzt, DANN wird er aus allen Ranglisten entfernt.
<b>Priorität</b>	unverzichtbar
<b>Verwendungshäufigkeit</b>	häufig

<b>Use Case 6</b>	<b>Namen auf Ranking verstecken</b>
<b>Erläuterung</b>	Der User möchte seine Achievements auf dem Ranking verstecken.
<b>Umfeld</b>	Weboberfläche
<b>Systemgrenzen (Scope)</b>	Browser
<b>Ebene</b>	Nebenfunktion
<b>Vorbedingung</b>	Der Server läuft und unterstützt Docker-Container. Der User ist im System eingeloggt.
<b>Mindestgarantie</b>	Die Webseite wird geladen.
<b>Erfolgsgarantie</b>	Der User kann seine Achievements und Punkte erfolgreich vor anderen Benutzern verstecken.
<b>Stakeholder</b>	Developer/User : möchte seine Daten verstecken können
<b>Hauptakteur</b>	Developer/User
<b>Auslöser</b>	Der User setzt einen Haken bei der Checkbox "Daten verstecken".
<b>Hauptszenario</b>	<ol style="list-style-type: none"> <li>1. Der User klickt auf seinen Namen.</li> <li>2. Das System öffnet die Profilseite des Users.</li> <li>3. Der einen Haken bei der Checkbox "Daten verstecken".</li> <li>4. Das System übernimmt die gewählten Einstellungen.</li> </ol>
<b>Erweiterungen</b>	
<b>Priorität</b>	unverzichtbar
<b>Verwendungshäufigkeit</b>	selten

### 3.3 Hauptfunktionen des Programms

Konkret sollen folgende **funktionale** Anforderungen erzielt werden:

- System kann auf einem lokalen Server in Docker eingerichtet werden
- Kollektoren für neue Subsysteme können modular hinzugefügt werden
- Das Programm lässt sich anhand einer CONFIG-Datei konfigurieren
- User können ihren Fortschritt auf einer Weboberfläche einsehen
  - Dieser Fortschritt wird über Achievements gemessen, die jeweils ihren eigenen Fortschritt besitzen.
  - Zusätzlich besitzt jeder Benutzer ein "Level", welches durch das erfüllen von Achievements erhöht werden kann

## 4 Qualitätsanforderungen

### 4.1 Qualitätsziele des Projekts

Die Programmierung selbst unterliegt einer klar formulierten **Code Convention**, an die sich jedes Team-Mitglied hält und die so Qualität und Übersichtlichkeit des Codes steigert. Diese Convention umfasst u.a. klare Regeln bzgl. Kommentierung und Dokumentation des Codes, die einheitliche Benennung von Variablen und Methoden und die Regelung der verwendeten GIT-Repository.

Außerdem werden folgende **nicht-funktionale** Qualitätsmerkmale festgelegt:

- Die Weboberfläche soll einfach bedienbar sein
- Die Wartbarkeit des Systems soll gewährleistet sein
- Beim Empfangen von Metriken über WebHooks soll mittels eines Schlüssels / gemeinsamen Geheimnisses die Authentizität der Quelle festgestellt werden (Sicherheit)

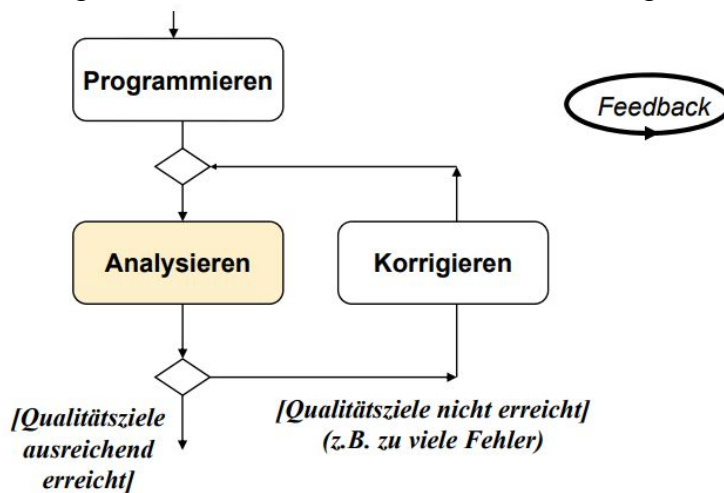
### 4.2 Qualitäts-Prioritäten des Kunden

Die Qualitätsziele sind wie folgt absteigend priorisiert:

1. System ist auf lokalem Server funktionsfähig
2. User können ihren Fortschritt auf einer Weboberfläche einsehen
3. Kollektoren für neue Subsysteme können modular hinzugefügt werden
4. Die Weboberfläche soll einfach bedienbar sein
5. Die Wartbarkeit des Systems soll gewährleistet sein

### 4.3 Wie Qualitätsziele erreicht werden sollen

Die Qualitätsziele werden für jeden Teilschritt anhand von Analysen und Tests überprüft und ggf. korrigiert. Dieser iterative Arbeitsschritt ist in folgendem Diagramm abgebildet:



Bereits in den frühen Projektphasen werden Unit-Tests für die einzelnen Module erstellt, sodass das Backend mit einer Testabdeckung von mind. 80% überprüft wird. Anhand der formulierten Qualitätsziele wird jeweils geprüft, ob diese erreicht wurden. Ist dies nicht der Fall, werden Fehler korrigiert, bis die gewünschten Testergebnisse erzielt werden.

Neben den einzelnen Modultests werden auch in diesen Phasen die Integrationstests mit anderen bereits bestehenden Modulen bzw. der vorausgesetzten Systemumgebung durchgeführt. Vor Abschluss des Projekts findet außerdem ein Systemtest statt, der die Anwendung in ihrer Gesamtheit testet und die Erfüllung aller Anforderungen gewährleistet.

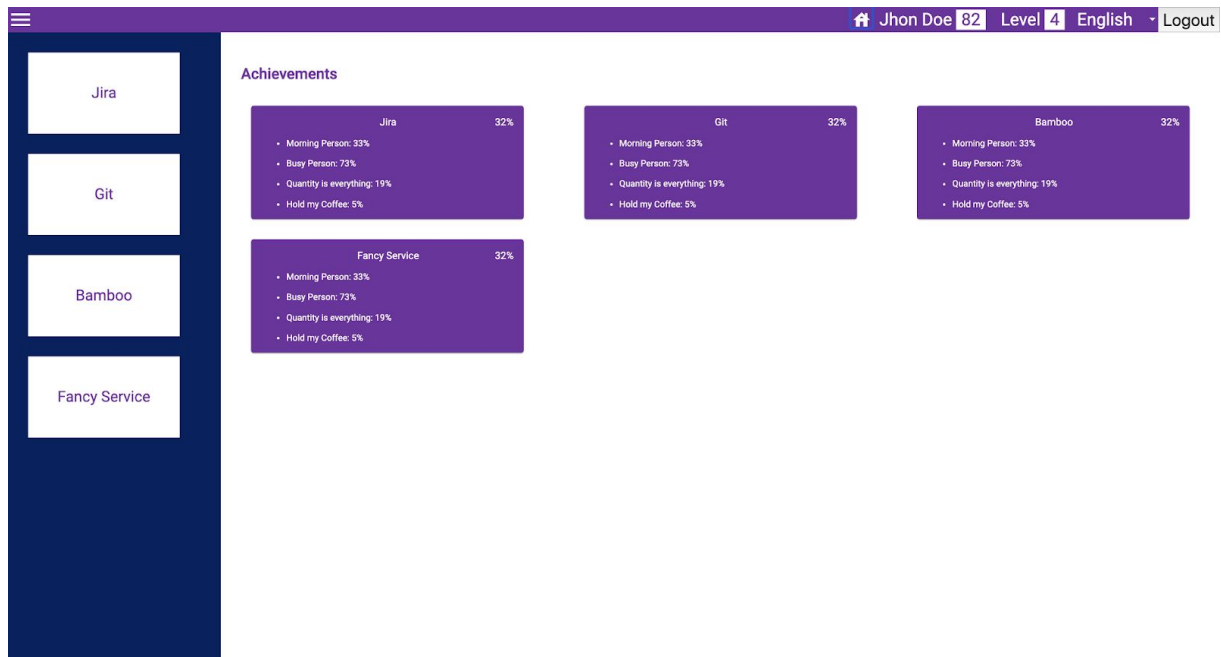
So stellen wir sicher, dass wir *Bottom-Up* bereits im Kleinen testen, ob einzelne Module wie gewünscht funktionieren, ob sie im nächsten Schritt miteinander korrekt interagieren und zuletzt im gesamten Systemkontext die Anforderungen erfüllen. Auf diese Weise können wir bereits im Entwicklungsprozess frühzeitig auf mögliche Fehler reagieren und ggf. korrigieren.

Testfälle beinhalten neben der eigentlich Funktionalität des jeweiligen Moduls mindestens die in dieser Spezifikation dokumentierten Use Cases und Abnahmetestfälle.

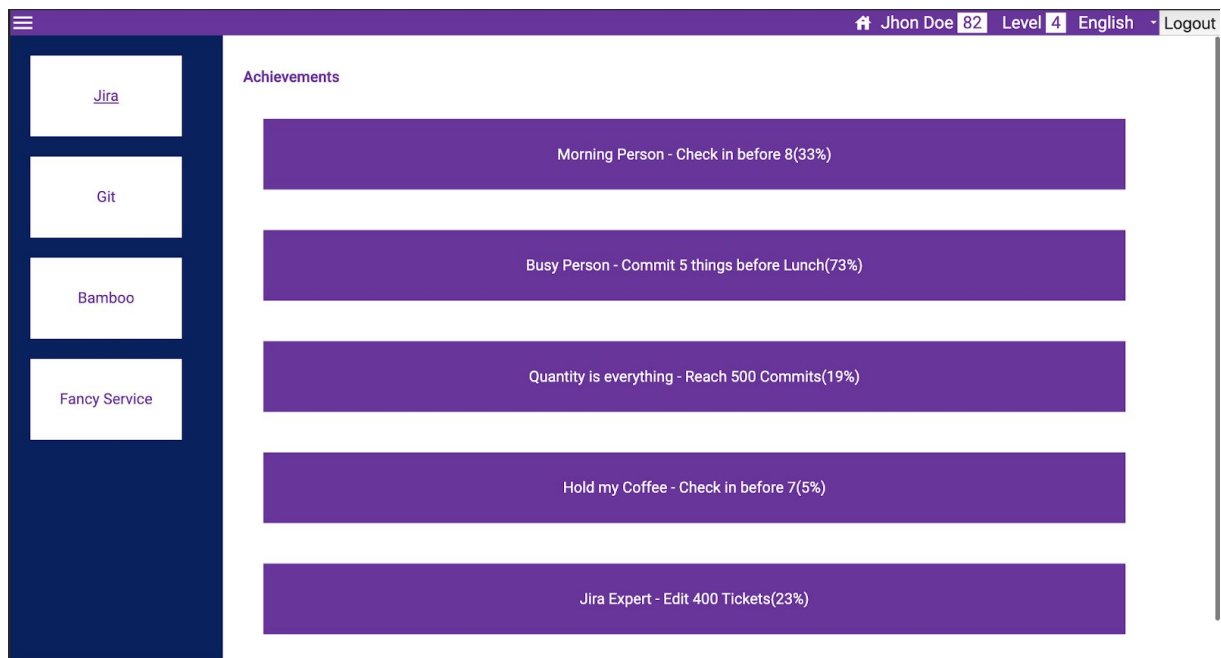
Darüber hinaus wird innerhalb des Teams anhand von klaren Aufgabenverteilungen und Dokumentation der einzelnen Aufgaben sichergestellt, dass Prozesse übersichtlich und nachvollziehbar bleiben. Dazu kommen Systeme wie Jira und Git zum Einsatz.

## 5 Hinweise zur Umsetzung

### Weboberfläche

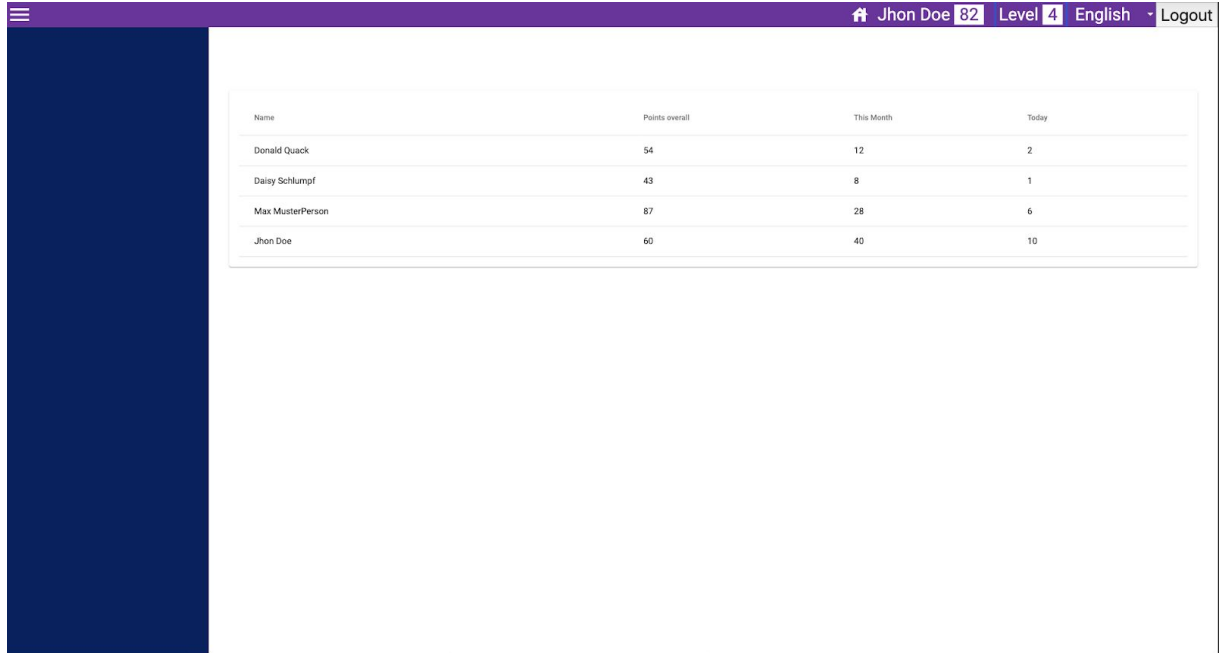


Auf der Webseite sieht man eine Übersicht aller Achievements für alle installierten Kollektoren. Außerdem sieht man in der oberen Zeile seinen Namen, die bereits erreichte Punktzahl und sein aktuelles Level. Ein Klick auf einen Kollektor führt zu einer erweiterten Ansicht.



In dieser erweiterten Ansicht werden alle Achievements für den ausgewählten Kollektor angezeigt. Zusätzliche Informationen wie ein Beschreibungstext können mit einem Klick auf das jeweilige Achievement ausgeklappt werden.

Mit einem Klick auf sein Level wird die Ranglistenübersicht angezeigt.

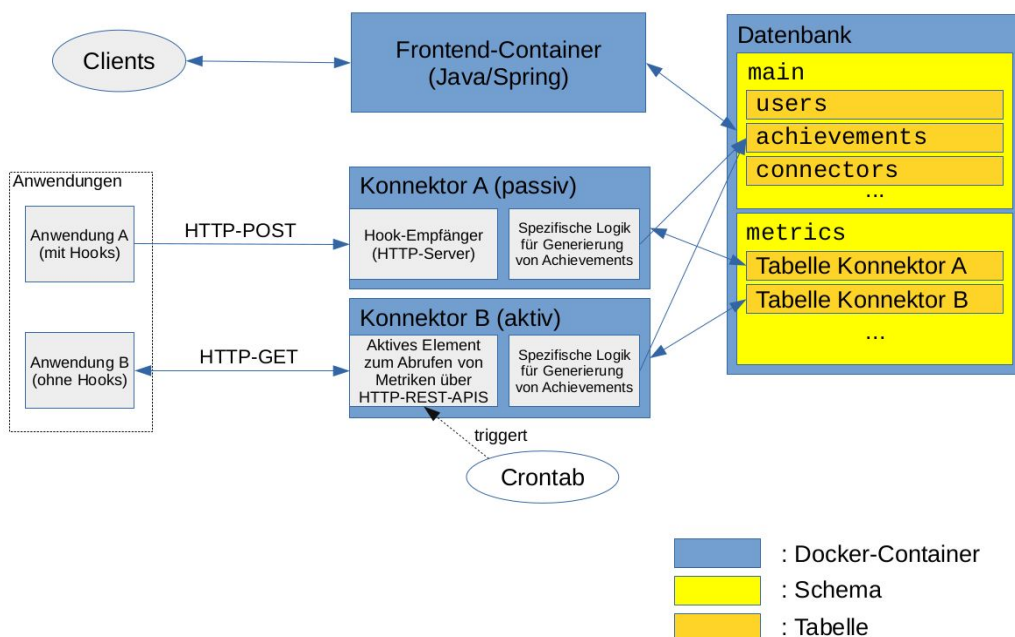


Name	Points overall	This Month	Today
Donald Quack	54	12	2
Daisy Schlumpf	43	8	1
Max MusterPerson	87	28	6
Jhon Doe	60	40	10

In der Ranglistenübersicht werden alle Teilnehmer des Projekts in einer Rangliste absteigend dargestellt.

## Architektur

Zur Umsetzung wird unter anderem die Technologie Docker verwendet. Die Hauptanwendung (in der Abbildung “Frontend-Container”), die Datenbank und die einzelnen Kollektoren laufen in isolierten und voneinander unabhängigen “Docker-Containern”. Kollektoren verwalten ihre Datenbank-Tabellen selbstständig und bieten im passiven Fall (Empfang der Metriken über Web-Hooks) einen Webserver mit (dienstabhängiger) Schnittstelle bereit. Aktive Kollektoren können beispielsweise per Cronjob angestoßen werden. Die Hauptanwendung stellt selbst eine REST-API bereit, um die Kommunikation mit den Clients auch auf anderem Weg als über die Weboberfläche zu ermöglichen und um den Kollektoren eine Möglichkeit zu geben, mit ihr zu kommunizieren. Die spezifische Logik zur Generierung von Achievements aus den Metriken ist in den Kollektoren enthalten.



### Was sind Achievements?

- Achievements besitzen einen Namen, eine Beschreibung und eine Bedingung das Achievement zu erreichen.
- Achievements können unterschiedlich oft erreicht werden (täglich, wöchentlich, monatlich).
- Achievements geben eine bestimmte Anzahl an Punkten die sich auf das Level und die Rangliste auswirken.
- Achievements werden von ihrem zugehörigen Kollektor implementiert und hinzugefügt.

### Was sind Kollektoren?

- Kollektoren sammeln Daten von verschiedenen Diensten, wie z.B. Git-Hosts oder Ticketsystemen (Jira)
- Kollektoren können einzeln hinzugefügt werden, auch (nachträglich) zur Laufzeit
- Kollektoren haben jeweils eine eigene Datenbanktabelle zur Speicherung ihrer Metriken
- Kollektoren vergeben anhand ihrer eigenen Metriken Punkte und Achievements



### **Wie funktioniert die CONFIG-Datei?**

Die Hauptanwendung wird durch eine Konfigurations-Datei vorkonfiguriert und beinhaltet (unvollständige Liste)

- Den eigenen Hostname
- Hostname und Zugangsdaten zur Datenbank
- Zur Authentifizierung der Benutzer über LDAP notwendige Informationen (uid-Attribut, Gruppen-Konfiguration, Filter, ...)
- API-Tokens zur Kommunikation mit den Kollektoren
- ...

(Diese Konfiguration kann beispielsweise beim Start an eine spezielle Stelle innerhalb des Docker-Containers gemountet werden, sodass das Image zustandslos ist und gleichzeitig die Konfiguration einen Neustart des Containers überlebt).

Die einzelnen Kollektoren enthalten ihre eigenen Konfigurationsdateien mit spezifischen Informationen wie

- Im passiven Fall (Web-Hooks):
  - API-Token(s) zur Hinterlegung bei den Diensten
- Im aktiven Fall (Polling):
  - Hostnames und Zugangsdaten, abhängig vom jeweiligen Dienst
  - ggf. Polling-Intervall
- Ggf. eine Liste der (nicht)zu beachtenden Metriken/Ereignisse (sofern implementiert)
- API-Token zur Authentifizierung an der REST-API der Hauptanwendung
- ...

Da die Implementierung der Hauptanwendung und Kollektoren noch nicht im Detail geklärt ist, ist diese Auflistung nicht endgültig und kann noch vielen Ergänzungen unterzogen werden, ist also eher als Hinweis zu verstehen.

## **6 Probleme und Risiken**

WENN die Daten von den Kollektoren falsch oder nicht ausgewertet werden (beispielsweise aufgrund einer Fehlkonfiguration von URL oder Zugangsdaten), DANN können Achievements nicht sinngemäß generiert werden.

Abhilfe: Sinnvolle Fehlermeldung mit Angabe zur genauen Fehlerquelle oder Behandlung von Exceptions.

WENN keine Achievements sinnvoll generiert werden können (siehe oberes Risiko), DANN kann die Webseite nicht sinnvoll mit diesen gefüllt werden.

Abhilfe: Sinnvolle Fehlermeldung an den Anwender mit Angabe zur genauen Fehlerquelle oder Behandlung von Exceptions.

WENN die Authentifizierung durch fehlerhafte Einbindung des vorhandenen LDAPs nicht richtig überprüft werden kann, DANN kann kein Zugriff auf die Webseite erfolgen.

Abhilfe: Notfalls eigenes Authentifizierungssystem (parallel zum vorhandenen LDAP-System) implementieren oder dem Anwender eine Information anzeigen, er solle den Administrator benachrichtigen.

WENN die empfangenen Metriken nicht korrekt einem Entwickler zugeordnet werden können, DANN können die Achievements nicht sinnvoll zugewiesen werden.

Abhilfe: Daten zu einem Entwickler so ändern, dass diese identifizierbar sind (z.B. durch E-Mail) und die Achievements ausgeben, so dass diese trotzdem gesehen werden.

## 7 Optionen zur Aufwandsreduktion

### 7.1 Mögliche Abstriche

Komplexe Datenauswertung und die Anzahl der Achievements insgesamt können reduziert werden.

Die Webseite kann simpler gestaltet werden.

Es können weniger Kollektoren implementiert werden, da diese auch später hinzugefügt werden können.

Achievements und Ranglisten können vereinfacht werden.

### 7.2 Inkrementelle Arbeit

In der *ersten Iteration* sollen folgende Funktionen implementiert werden:

- Hauptanwendung auf Spring Boot Basis (Web-Oberfläche sowie REST-API)
- Datenbank-Server und Initialisierung der Datenbank-Tabellen
- Modulare Erweiterung von Kollektoren, diese werden von der Hauptanwendung erkannt und können ihre Metriken über eigene Datenbank-Tabellen speichern
- Kommunikation zwischen Kollektoren und Hauptanwendung ist spezifiziert (zum Beispiel über die REST-API und/oder über die Datenbank)

In der *zweiten Iteration* sollen folgende Funktionen implementiert werden:

- Die einzelnen Kollektoren werden im Detail implementiert (mindestens Jira und BitBucket), sie können Metriken empfangen oder sammeln und daraus (ggf. auf Anfrage der Hauptanwendung) Achievements generieren und diese einem Benutzer zuordnen
- Die Weboberfläche kann die Errungenschaften eines Benutzers anzeigen
- Es gibt Level und Ranglisten sowie verschiedene Typen von Achievements (wiederholende, einmalige).

In der *dritten Iteration* (Polish) *können* - abhängig vom Fortschritt - folgende Funktionen implementiert werden:

- Kollektoren für weitere Systeme als die mindestens verlangten (z.B.: "Bare" Git, GitLab, SonarCube, Redmine, Jenkins, SVN)
- GUI für die Konfiguration der Hauptanwendung und/oder für die Konnektoren
- Bessere Auswertungen und Views können entwickelt werden, um die Daten attraktiver darzustellen.

## 8 Glossar

<b>Kollektor</b>	Modulare Komponente, die Daten von einem Speziellen Dienst (zb Git) abfragt oder empfängt, verwertet und in Achievements umwandelt und dann an den Hauptserver sendet.
<b>SW</b>	Software

## 9 Abnahme-Testfälle

### Login

Setup: Es existiert in LDAP user mit email "abc@gmx.de" und passwort "abc",

Eingabe	Soll
User gibt bei e-mail "abc@gmx.de" und bei passwort "abc" ein	Der User wird auf die Achievementsseite vom Benutzer abc weitergeleitet.
User klickt auf seinen Namen "abc"	Profilseite von "abc" wird angezeigt
Der User ändert seinen Namen von "abc" zu "overachiever"	In der Rangliste taucht sein Name jetzt als "overachiever" auf

Setup: Es existiert kein user mit E-mail "abcd@gmx.de" und passwort "abcd"

Eingabe	Soll
User gibt email "abcd@gmx.de" und passwort "abcd" auf der Loginseite der Webseite ein	Der User kriegt eine Fehlermeldung angezeigt und bleibt auf der Loginseite.

Setup: Es existiert in LDAP user mit email "abc@gmx.de" und passwort "abc"

Eingabe	Soll
User gibt seine E-Mail "abc@gmx.de" und passwort "passwort" ein	Der User kriegt eine Fehlermeldung und bleibt auf der Loginseite

**Achievements anschauen**

Setup:

User ist im System eingeloggt und befindet sich auf der Webseite. Er will seine Achievements von Git anschauen.

Eingabe	Soll
User klickt auf das Haus	Die Achievements-Seite wird angezeigt
User klickt auf "Git"	Die Git-Achievements-Statistik von dem User werden angezeigt.

**Kollektor hinzufügen**

Setup:

Das Hauptsystem läuft.

Eingabe	Soll
Der Administrator startet einen Kollektor auf dem Server.	Der Dienst taucht auf der Webseite unter Achievements auf.

**User löst ein Jira-Achievement aus**

Setup:

Das Hauptsystem läuft.

Es wurde ein Jira-Kollektor hinzugefügt.

Der User hat das Achievement: "Ein Ticket anlegen" noch nicht abgeschlossen.

Eingabe	Soll
Der User legt in Jira ein Ticket an.	Auf der Webseite wird nun unter Jira-Achievements das Achievement: "Ein Ticket anlegen" als abgeschlossen angezeigt.